Université De Montpellier -Faculté Des Sciences
L2 informatique
Année Universitaire 2021 - 2022
Projet informatique — HLIN405

Groupe Y
Ayoub Chenini
Ali BA FAQAS
Ayman Benazzouz
Abdullah Obad

March 2022

Contents

1 Introduction

Dans le cadre de notre projet de deuxième année S4 nous avons développé un jeu de société "qui-est-ce?" Ce jeu a été codé en JAVASCRIPT Notre groupe est composé de quatre étudiants, Ayoub, Ali Bafqas, Ayman Benazzouz, et Abdullah Obad. La réalisation de ce projet s'est déroulée sur une période de 16 semaines, de mi-janvier à mi-Avril 2020.

1.1 Technologies utilisées et organisation

1.2 Langages de programmation

- **JavaScript** : un langage côté client omniprésent sur le web. Il nous permet de développer un site dynamique, avec mise à jour des pages sans rechargement. De plus, la plateforme Node.JS qui permet de gerer le côté serveur nous a permis de réaliser l'extension jeu à deux.
- **PHP** :est un langage de scripts généraliste et Open Source, spécialement conçu pour le développement d'applications web.

1.3 framework et bibliothèque

- **Bootstrap** :une collection d'outils utiles à la création du design de sites et d'applications web.
- **jQuery** une bibliothèque JavaScript conçue pour simplifier la traversée et la manipulation de l'arborescence HTML DOM.
- **Express.js** :framework d'application Web back-end pour Node.js, publié en tant que logiciel gratuit et open-source sous la licence MIT. Il est conçu pour créer des applications Web et des API.
- **Socket.IO** :bibliothèque JavaScript pilotée par les événements pour les applications Web en temps réel.

1.4 Outils

GitHub : un service web d'hébergement basé sur le logiciel de gestion de versions Git. Il facilite le développement concurrent grâce au système de branches.

excalidraw : un tableau blanc collaboratif simple et sans inscription, proposant simplement d'écrire, dessiner ou construire des diagrammes.

npm : un gestionnaire de packages pour le langage de programmation JavaScript géré par npm, Inc.

1.5 Organisation du travail

1.5.1 Jeu en solo

Nous avons réparti notre travail selon la division Back-end/Front-end de notre jeu:d'un côté toute la partie que l'utilisateur ne voit pas, mais qui lui permet de réaliser des actions sur le jeu, de l'autre la création de l'interface de jeu.
Nous avons commencé par réaliser ce que nous avions illustrer sur notre maquette de jeu, tout d'abord nous avons réaliser le design en html et CSS permettant d'avoir l'aspect de base du jeu sans les images bien évidemment. Enfin nous avons fini par le JavaScript qui permet lui de faire fonctionner le jeu.
Pour tester le premier le jeu en ligne : https://abdulllahobad.github.io/Projet-infoL2/

1.5.2 Générateur de JSON

En ce qui concerne cette partie tout comme la partie précédente, le plus gros du travail s'effectue en back-end. Dans un premier nous avons réalisé le visuel qui rappel celui du jeu en HTML et CSS. Ici, le JS nous a permis d'afficher les changements effectuer par l'utilisateur (faire apparaître les cases, les images,etc...) Enfin, le principal du générateur et réaliser grâce à PHP en effet c'est l'un des seuls langage qui permet d'écrire dans un fichier, ce qui n'est malheureusement pas le cas de Javascript.
Pour tester le générateur en ligne : https://generateurjson.000webhostapp.com/

1.5.3 Jeu à deux

Dans cette partie nous avons utilisé le jeu que nous avons construit en premier lieu et nous l'avons transformer en mode jouer-à-deux.
le travail a commencé par construire notre serveur en node.js, puis nous avons modifié le fichier javascript de la premiere partie pour lui permettre de gérer le côté client ce fichier est devenu `client.js`. puis nous avons crée les événements suivants:
1)créer des salons.
2)rejoindre un salon.
3) commencer le jeu.
4)gagner ou perdre la partie.
5)demander une nouvelle partie.
6)accepter la demande d'une nouvelle partie.

Pour tester l'extension en lignehttps://jouer-a-deux.herokuapp.com/

MSGR

Diagramme de Gantt

2 Permettre à l'utilisateur de jouer

2.0.1 Importation du JSON :

L'importation du JSON afin d'importer les informations nécessaires au jeu s'est effectuée grâce à la méthode getJSON() disponible sur la bibliothèque JQUERY cependant cela nous oblige à utiliser le jeu sur un serveur sinon ma méthode ne fonctionne malheureusement pas.**Affichage des personnages :** Afin d'afficher les personnages il nous suffit d'obtenir le nom de l'image de chaque personnage et la mettre dans une balise `<img>`. Cependant, il nous faut absolument respecter les lignes et les colonnes disponible sur le json pour cela nous avons appliqué l'algorithme suivant :

Algorithm 1 Afficher personnages

- **for** i allant de 0 à lignes **do**
  Créer une variable ligneDIV
  j=i + ((colonnes-1) * i)
  while (j - i - ((colonnes-1) * i)) inférieur à colonnes **do**
  Ajouter à ligneDIV la jème image du JSON
  j++
  end while
  end for

- **Affichage des questions :**

  Etant donné que tous les personnages ont les mêmes caractéristiques il suffit d'obtenir les caractéristiques du premier pour les avoir toutes. Ainsi il suffit d'ajouter dans une valise `<select>` l'ensemble des caractéristiques hormis la caractéristique fichier du premier personnage.

Algorithm 2 Afficher caractéristiques

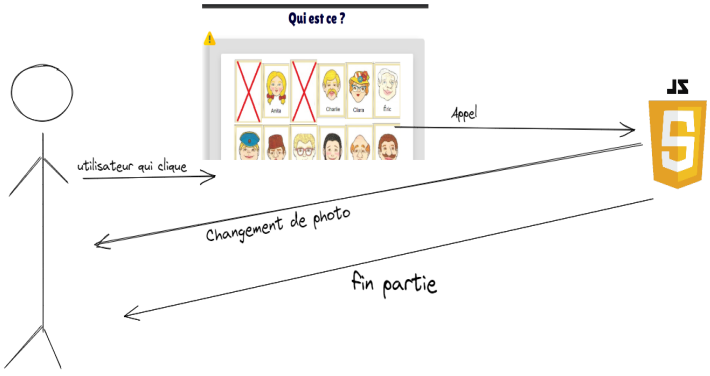
CaracteristquePerso (tableau du premier perso)
for i de 0 à nombreCaracteristiques **do**
  if CaracteristquePerso[i]!=”fichier” **then**
    On ajoute la caractéristique à la balise `<select>`
  end if
end for

2.0.2 Élimination de personnages:

Afin que le joueur puisse éliminer un personnage dont il est sûr qu'il n'a pas été choisi par l'ordinateur,il peut cliquer sur n'importe quelle photo et l'image du personnage sera remplacée par la même image mais avec cette fois-ci une croix rouge indiquant que le personnage est éliminé (le phénomène inverse n'est cependant pas autorisé par le jeu).
Pour changer l'emplacement de l'image du personnage cliqué par le joueur, intuitivement nous sommes partie avec l'idée que chaques personnage dispose de 2 photos une avec son prénom et l'autre avec son prénom avec un X rouge dessus.

```
var image = document.getElementById(clickeId);
image.src = data['images']+data['nom']+'X.png';
```

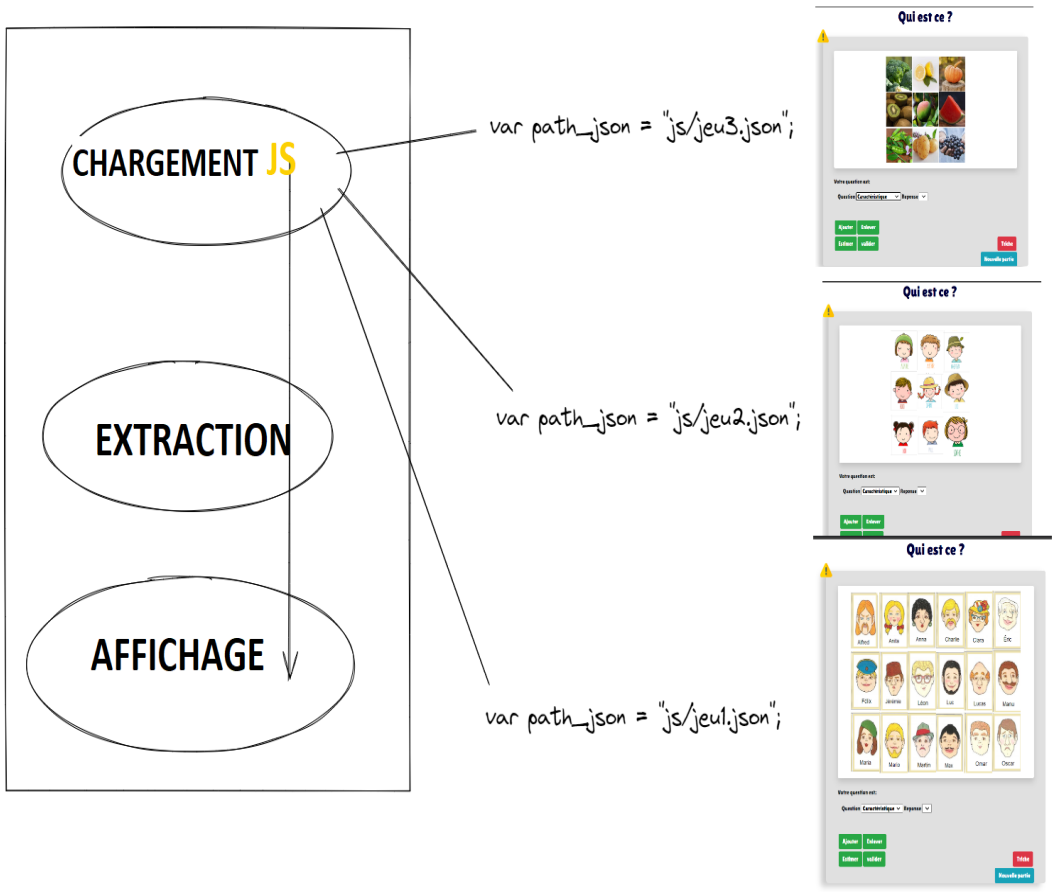
Problème:
Cette idée est géniale et fonctionne très bien, mais pour que le jeu soit dynamique avec le générateur, Le générateur doit copier chaque image en deux images, l'une avec le même nom que le personnage et l'autre avec le nom + ”X” Devant le nom du personnage.
de ce fait,nous avons donc pensé à une manière qui serait moins compliquée, pour qu'il y ait ”une seule image” representant la croix afin d'indiquer q'un personnage est éliminé.



2.0.3 La généricité du jeu

Il suffit juste de changer la variable globale "Path.json"par le nom d'un fichier json que vous voulez appliquer et une fonction jQuery va aller chercher les objets existants dans le fichier choisi. Une seule condition est nécessaire c'est que le fichier json doit obligatoirement disposer d'un attribut "prenom" si c'est un jeu de personnages par exemple ou "nom" si c'est un jeu de n'importe quels objets(voitures , légumes, instruments....etc) , car le joueur peut poser au minimum des questions sur le nom ou le prénom, en plus il existe des fonctions (mode triche , estimer, fin parie) qui reçoivent le nom(respectivement prenom) de l'objet choisi pour pouvoir fonctionner correctement. Pour celà, une fonctionne Prenom.existe qui reçoit les données du JSON et cherche l'existence de la clé "prénom" dans le fichier si on le trouve on change notre variable globale en prénom.
`if (i =` prenom")"

id.Personnage = "prenom;



3 Aider à la saisie des personnages

Il s'agit ici de permettre la création de grilles de personnages personnalisées.

3.1 Principe de base

L'utilisateur sélectionne le nombre de lignes et de colonnes dont il a besoin, ainsi que les photos et les caractéristiques dont il a besoin. Ensuite, avant de télécharger le fichier, l'algorithme vérifie si chaque personnage peut réellement être distingué en fonction des caractéristiques et des photos définies.

3.2 Le rôle du JavaScript

Nous ne attarderons pas très longtemps sur cette partie, en effet le rôle du Javascript est ici plus ou moins identique au rôle qu'il joue dans le jeu. Nous avons utilisé le JS afin simplement d'afficher le contenu du fichier JSON temporaire dans la page web en particulier les caractéristiques saisies des personnages.

3.3 Création du fichier JSON

L'utilisation de PHP est ici primordial pour nous.En effet, contrairement au JavaScript le langage PHP permet l'écriture/la création de fichier au sein du serveur. Vous l'aurez compris, la création d'un fichier JSON dans le serveur était ici primoridal pour nous.

3.3.1 L'utilisation de la variable globale $_SESSION

Après avoir comparé le fichier JSON avec une variable tableau, nous avons décidé d'utiliser un tableau pour stocker les informations qui seront stockées dans le JSON.
Cependant, nous ne savions toujours pas comment enregistrer les informations sans les perdre lorsque l'utilisateur actualise la page.
Mais nous nous sommes souvenu des variables globales. Il s'agit de variable qui permet de sauvegarder les informations jusqu'à ce que l'utilisateur quitte le navigateur.
Ainsi, à chaque fois que l'utilise rentre des informations nécessaire au fichier JSON elles sont stockées dans la variable $_SESSION exemple :

```
$_SESSION['json']['images']=images/;
$_SESSION['json']['colonne'] = $_POST['ligne'];
$_SESSION['json']['ligne'] = $_POST['colonne'];
```

Ici, après avoir saisi les lignes et colonnes, les informations sont stockées dans la variable $_POST grâce à la méthode `post`

3.3.2 Comment remplir le fichier JSON temporaire

- **Conversion de tableau vers format JSON**
  La conversion de simple tableau au format JSON est très simplifié avec le langage PHP. En effet, il existe une fonction PHP qui permet cette conversion, la fonction est la suivante : `json_encode()`
  Ainsi, dans notre cas l'utilisation de cette fonction se fera de la manière suivante : `json_encode($_SESSION['json'])` On convertit donc l'ensemble du sous-tableau `$_SESSION['json']`.

- **Ecriture dans le fichier JSON temporaire**
  Même si nous arrivons à remplir la variable $_SESSION correctement et à convertir le tableau en format JSON, reste à savoir comment mettre ses informations dans un fichier JSON.
  Viens ici l'utilisation d'une fonction PHP qui est vraiment très utile `file_put_contents()` cette fonction permet de créer un fichier si il ne l'est pas déjà ainsi que d'écrire dans le fichier.
  Elle prend un premier paramètre une variable de type string qui sera du format `nom\_fichier.extension` et en deuxième paramètre les informations à mettre dans le fichier. Dans notre cas, l'utilisation de cette fonction se fait en simultané avec la fonction vu précédemment à chaque fois que l'on doit modifier la variable $_SESSION['json'] son utilisation devient alors :

```
file_put_contents('temp_fichier.json',json_encode($_SESSION['json']));
```

3.3.3 Cas particulier : insertion des images

Dans un premier temps nous avions opté pour ne mettre seulement que les filenames des images que l'utilisateur utilise pour les personnages dans le fichier JSON.

Problème : Cette façon de faire n'était pas adéquate pour une raison très simple. Nous ne pouvions pas afficher les images qui ne sont pas directement dans le répertoire du générateur. Or, cela implique que l'utilisateur sera à la fois le client et le serveur. Cette manière d'effectuer n'est pas du tout idéal, voir n'est pas viable du tout. Alors comment résoudre ce problème ?

Il nous a fallut donc, pouvoir importer les images directement dans notre serveur afin de pouvoir les afficher à l'endroit voulu. Pour cela, nous avons utilisé la variable spécial disponible sur PHP $_FILES qui permet d'obtenir les informations nécessaires (taille,type,nom,etc) à l'importation de l'image par l'intérmediaire de la méthode POST.

De plus, il nous faut également importer les images sinon le problème n'est pas totalement résolu. Nous avons donc pour cela utilisé la fonction :
`move_uploaded_file();`

3.3.4 Téléchargement du fichier JSON

Problème : Lorsque nous avons entamé le script permettant le téléchargement du fichier JSON un enseignant nous a signalé que notre façon de faire n'était pas intuitive. Effectivement, lors du téléchargement seulement le fichier était téléchargé ainsi, il n'était pas très facile pour un utilisateur t'utiliser son propre fichier JSON dans notre jeu, car il devait absolument importer lui-même chaque photo qu'il a auparavant utilisé dans notre générateur.

De ce fait, l'utilisation d'un fichier était devenu evidente, nous avons alors changer le script de téléchargment disponible sur le fichier `telechargement.php` afin de faire télécharger à l'utilisateur un zip contenant un fichier JSON et un dossier "images" contenant l'ensemble des images utilisé dans le générateur.

4 Extension : Jouer à deux

4.1 Présentation de l'extension

Cette extension est faite afin de permettre de jouer en ligne. Pour cela il faut saisir un pseudo et créer un salon enfin il faudra envoyer le lien afin qu'il puisse rejoindre l'un des salons en ayant au préalable saisi son pseudo (chaque joueur peut créer son propre salon, et chaque joueur peut rejoindre n'importe quel salon créer). Le salon peut accueillir au plus 2 joueur afin de pouvoir jouer.

Régle du jeu :

- Les joueurs vont jouer chacun leur tour en commançant par le créateur du salon.
- Les joueurs ne peuvent poser qu'une question par tour, ensuite ils peuvent éliminer autant de personnages qu'ils veulent.
- A la fin du tour le jouer est obliger d'indiquer qu'il a fini par l'intermédiaire du bouton FinTurn afin que l'autre joueur puisse jouer. - Si un joueur élimine le personnage cherché il perd, et l'autre joueur gagne. - si un joueur trouve le personnage il doit choisir son prénom dans les questions, si le personnage qu'il a choisit été le personnage choisit le joueur gagne, sinon il perd.
- Quand le parie termine, chaque joueur à le droit de demander une nouvelle partie, l'autre joueur est libre d'accepter ou d'ignorer la demande.
- pour chaque partie, le serveur choisit un nouveau personnage, et c'est le même personnage pour les deux joueurs.

4.2 Environnement utilisé

Nous avons décider d'utiliser l'environnement d'exécution "node.js".
Node.js est un système single thread non bloquant pour l'exécution de serveurs Web développés en JavaScript. Il utilise le même moteur JavaScript que Chrome et Edge. Il s'agit d'un moteur V8 développé par Google et très puissant. Node.js utilise JavaScript, mais l'environnement est très différent du frontend. Il n'a pas de DOM ou d'API Web, mais il utilise de nombreuses bibliothèques développées en C et est très puissant comme libuv pour la gestion des événements et des modules pour le chiffrement(cryptographie) et l'accès au système de fichiers.

Nodejs utilise le framework Express. ExpressJS est un framework qui se veut minimaliste. Il est très léger et fournit quelques superpositions pour maintenir des performances optimales et une exécution rapide. Express ne fournit que des fonctionnalités d'application Web (et mobile) de base, mais celles-ci sont extrêmement robustes et ne remplacent pas la fonctionnalité NodeJS native.
Node Express JS est également très flexible. Si vous avez moins de fonctionnalités, vous pouvez le compléter avec les nombreuses bibliothèques disponibles

sur npm. Vous êtes libre de choisir la bibliothèque et l'architecture backend qui vous conviennent le mieux.
Au sein de cette extension nous avons également utilisé la bibliothèque socket.io. Socket.io est un module Node.js qui permet de créer des sockets web. C'est à dire des connexions bidirectionnelles entre des clients et un serveur qui permet une communication en temps réel à l'aide d'un protocole différent du protocole HTTP normalement utilisé pour les pages Web. Ce type de technique est utilisé pour créer des applications telles que des systèmes de communication en temps réel (comme les Tchat), des jeux multi-utilisateurs et des applications de collaboration, et bien plus encore.

4.3 protocole utilisé

Le protocole utilisé par cette extension est WebSocket. Il établit une véritable connexion entre le client et le serveur et permet une communication bidirectionnelle en temps réel. Le client peut envoyer un message au serveur, et le serveur peut envoyer un message au client sans avoir besoin d'une requête/réponse via http.
Ce nouveau type de connexion permet également de créer un groupe de communication entre tous les clients ayant établi une connexion WebSocket avec le même serveur (Figure 2). Le principe est très simple.
Le client envoie un message au serveur Le serveur reçoit le message Le serveur envoie le même message (ou message modifié) à tous les clients connectés

Cependant, gardez à l'esprit que le protocole WebSocket nécessite le protocole http pour établir la première connexion. En fait, le protocole WebSocket est une sorte de "mise à jour" du protocole http. Deux conditions préalables doivent être remplies pour effectuer cette mise à niveau :

Le serveur et le client doivent être capables de gérer le protocole WebSocket. Côté serveur, cela est rendu possible par la combinaison Express.js / Socket.io. Les clients peuvent le faire via l'API WebSockets HTML5, qui est compatible avec presque tous les navigateurs actuels.

4.4 Server.js

Pour créer un server avec node.js la structure est simple, comme nous avons besoin de express et socket.io il faut les télécharger grâce aux commandes suivantes:
pour télécharger express: `$ npm install --save express`
pour téléchager socket.io: `$ npm install --save socket.io`
maintenant nous remarquons les deux frameworks dans l'objet "dependencies" du fichier package.json

Nous commencons par écrire dans notre fichier server.js :

-`require('express')` est pour inclure express dans notre serveur.
-la fonction `express()` est utilisé pour créer une application express dans le serveur.
-`require('path')` est pour inclure le path module dans notre app.
-`require('http')` est pour inclure le http module, cela permet node.js de transemttre les data sur HTTP
-`const PORT` est pour indiquer le port que le serveur va écouter, dans notre jeu, le serveur écoute un port(process.env.PORT) libre quand nous sommes sur le cloud ou bien il écoute le port 3000 sur localhost.
-`http.creatServer(app)`, sert à créer le serveur HTTP qui va accepter les requêtes http.
-le serveur commence à écouter le port grâce à `server.listen(())`.

Pour indiquer la route menant au jeu nous avons utilisé la methode GET

pour commencer la connexion avec les clients, il suffit de créer un événement `io.on('connection', (socket)=>{.....}`

-les deux principales fonctions pour envoyer et recevoir des requêtes de ou vers le serveur sont `socket.emit()` et `socket.on()`
-`io.ot(id).emit()` sert à envoyer un message à un client ou un salon particulier

4.5 Le fichier "client.js"

dans le fichier client il suffit de créer un `const socket = io()\verb`; puis nous pouvons tranquillement envoyer et recevoir des requêtes au ou de serveur avec les fonctions `emit()` et `on()`

4.6 Les événements

Connection: Cet événement est déclenché lorsque le socket tente de se connecter.

PlayerData: L'événement playerData peut recevoir des données de joueur grâce au paramètre "player" envoyé par le client. De plus, lors de cet événement, le joueur va créé une salle grâce à la fonction "CreatRoom (player)", la salle a donc été créée à partir d'ici.

Rejoindre le salon: Dans notre jeu, chaque joueur a le droit de créer et de participer à une salle. Chaque salle a la capacité de deux joueurs. Une fois la salle créée, elle sera visible par tous les clients connectés au serveur. Pour rejoindre une salle, entrez un nom et cliquez sur le bouton pour rejoindre la salle sélectionnée.

start game: Lorsque deux joueurs se connectent à la même salle, l'événement de début de jeu démarre et la div html pour créer et rejoindre la salle est remplacée par la div du jeu, grâce au `roomsCard.classList.add('d-none');` `waitingArea.classList.add('d-none');` et pour faire apparaitre le jeu on utilise `gameArea.classList.remove('d-none');`

L'événement Win: grâce a une fonction GameWin() un événement "win" sera lancé par le serveur, le serveur va envoyer un événement "TouWin" pour le joueur gagnant, et également un autre événement au joueur perdant

L'événement Lose: comme l'événement win, ce événement envoit une message au jouer perdant, puis une message au joueur gagnant

Les événements de rejouer: après avoir terminé la partie, nous remarquons un bouton "nouvelle partie" ce bouton va permettre au serveur d'envoyer un demande a l'autre joueur et celui-ci aura le droit de choisir entre rejouer ou pas. Si l'autre joueur accepte, une nouvelle partie sera démarrer.

4.7 Problèmes rencontrés

Lors de la création du serveur et du client, nous avons rencontré divers problèmes que nous avons pu résoude en ligne et avec l'aide de nos professeurs.

Problème n°1
websocket.js:54 WebSocket connection to 'ws://localhost:3000/socket.io/?EIO=4transport=websocket' failed: l'extension McAfee® Web Boost empêche la connexion

nous avons remarqué que si les extensions de google chrome ou n'importe quel navigateur fonctionnent, cela crée parfois des problèmes côté serveur donc c'est très conseillé de désactiver toutes les extensions en particulier les extensions adblock et de protection.

Problème n°2 Failed to load resource: the server responded with a status of 404 (Not Found) http://127.0.0.1:5500/favicon.ico".
notre navigateur cherche toujour au favicon.ico même si nous l'inclue pas sur notre code HTML donc pour resudre ce problème il faut inclure dans la tête de notre html la ligne `<link rel='shortcut icon' href='#'>;`
cela va initaliser notre icon a vide!

Problème n°3 Nous souhaitons utiliser les alerts pour gérer certains événments comme, un joueur gagne, perd, demande une nouvelle partie, joueur déconnecté,etc..
les alerts par default de javascript n'étaient pas très interessants pour nous, donc nous avons cherché un alternative. les sweetalerts2 était notre choix préférée donc nous les avons utilisé dans notre jeu

pour utiliser les sweetalerts2 de javascript il faut les télécharger et inclure une ligne dans la tête du code html
pout téléchager les sweetalerts2 il faut utiliser la commande suivante :
`npm install sweetalert2`
la ligne html:
`<script src='//cdn.jsdelivr.net/npm/sweetalert2@11'></script>`

pour plus d'informations sur les Sweetalerts2: https://sweetalert2.github.io/

4.8 Utilisation de notre extension

Nous avons mit le jeu en ligne dans le cloud Heroku, pour que le jeu soit disponible en ligne donc il y a deux possibilités pour tester le jeu :

-Tester le jeu directement en ligne, lien du jeu hebérgé dans le cloud : https://jouer-a-deux.herokuapp.com/ -Tester le jeu en localhost pour cela : -Vous devez téléchager node.js
-Installer les sweetalert2 via la commande : `npm install sweetalert2` (windows)
-Localisez vous dans le repertoire du jeu et tapez `node server`
-vous allez remarquer dans le terminale que le serveur écoute le port (lestning

on) 3000, donc tout fonctionne correctement.
-ouvrez un navigateur et tapez `http://127.0.0.1:3000/`

5 Bilan et Conclusions

Nous voici donc à la fin de notre aventure, ce projet qui avait pour but de recréer le jeu "Qui-est-ce?" en Javascript est maintenant terminé.

Ce projet nous a permis de mettre en pratique nos connaissances durement au cours de cette année. De manière abrégée nous pourrions parfaitement dire que nos connaissances acquises en programmation web ont été les connaissances les plus utilisées durant ce projet étant donné nos choix de langage en début d'année. Même si le reste des UEs ont été eux aussi utiles, comme l'UE "HAI404I : IP Protocoles et communications réseau" qui nous a permis d'un petit mieux comprendre et d'aborder l'extension en multi-joueur.

Tout au long de ce projet, nous étions guidés par nos encadrants pendant 10 semaine afin d'assurer l'avancement de notre projet, des réunions régulières dans une salle de TP et en dehors de ça a on crée un serveur sur l'application Discord pour facilité la communication entre nous.À la fin de chaque rencontre "chaque samedi" dans une salle de notre résidence Triolet, nous établissions un compte-rendu résumant les points abordés ainsi que la date prévue pour la réunion suivante et la mission que chacun devait réaliser.

Enfin, nos avons pu regoûter au travail de groupe mais cette fois-ci de manière bien plus intense que les précédents projets que nous avons pu mener. Cela nous a permis de ressentir et de voir comment le travail de groupe s'organise, chose qui est loin d'être simple. Mais ensemble nous avons réussi à surmonter les obstacles et au fil des jours trouvés une véritable unité de groupe.

Lien utile :

Le dépôt GitHub du projet (jeu,generateur) https://github.com/AbdulllahObad/Projet-infoL2.
Le lien en ligne du jeu https://abdulllahobad.github.io/Projet-infoL2/
Le lien en ligne du générateur https://generateurjson.000webhostapp.com/
Le lien en ligne de l'extension jouer à deux : https://jouer-a-deux.herokuapp.com/