Compte rendu du projet Application Design JavaScript

Réalisé par :
PORETTI Isidore
SOW Ibrahima

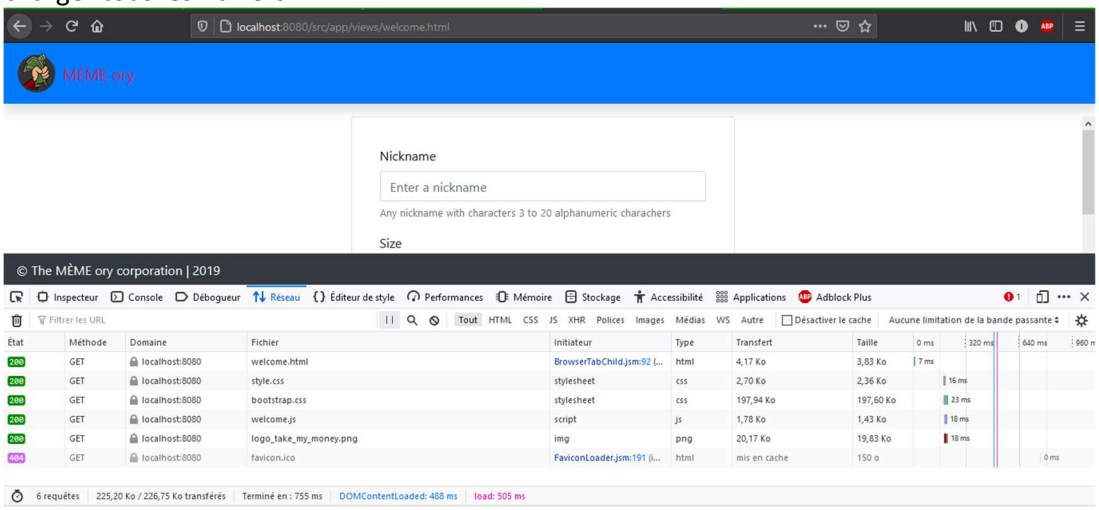
Encadré par :
M. PURET Alexis
M. THIERION Nicolas

Step 0 - Setup

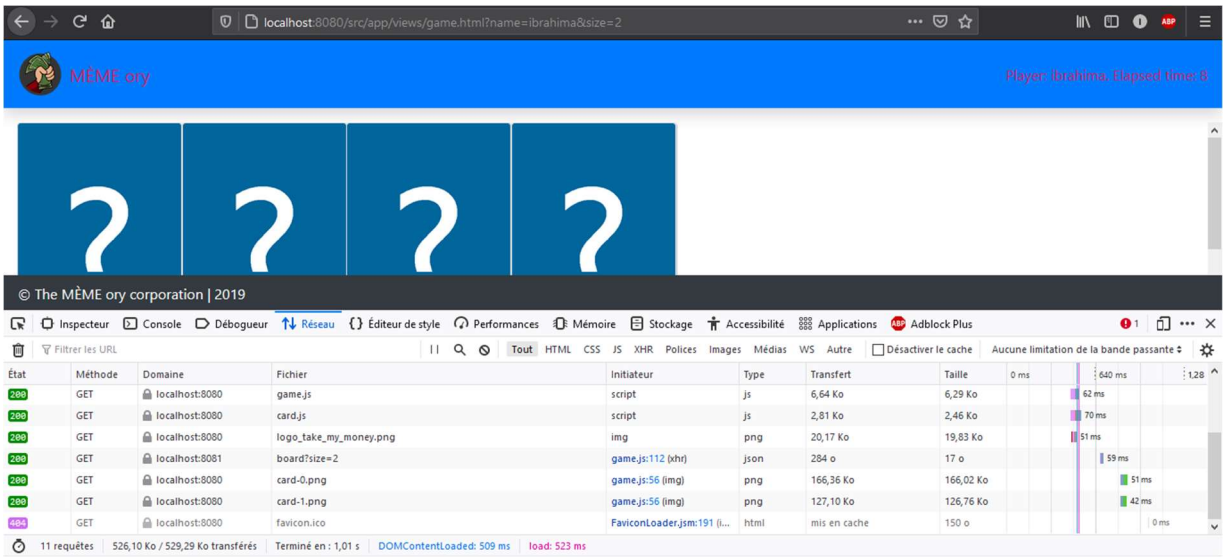
Q : While going through the 3 views of the application , how many files did your browser download overall ? How many times did it took to load them all?

A : En parcourant les 3 vues de l'application :

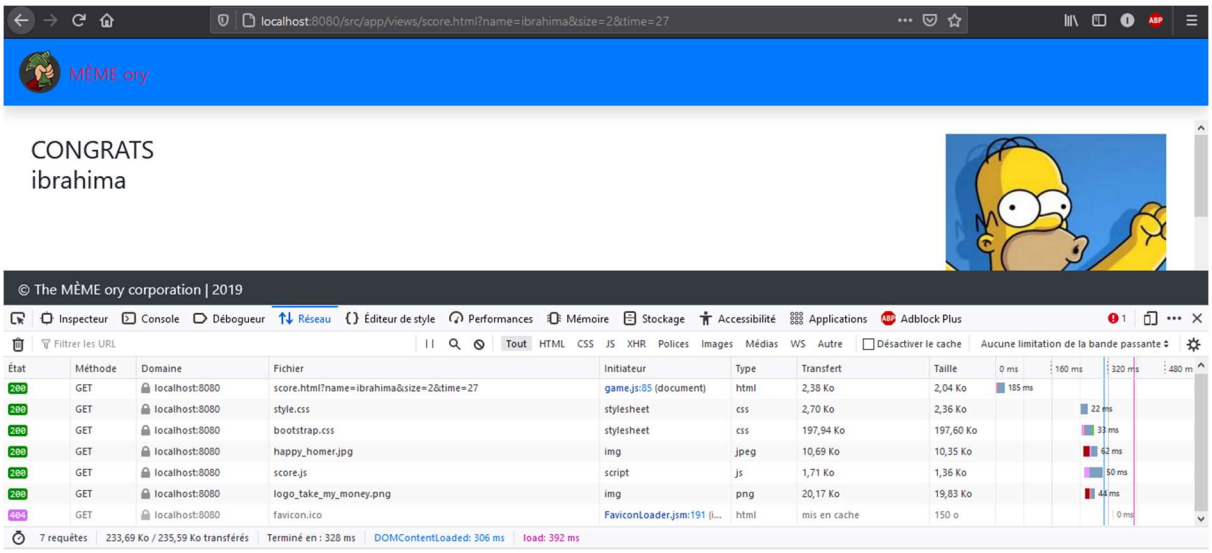
- Page d'accueil : le navigateur télécharge un total de 6 fichiers. Il lui a fallu 505 ms pour charger tous les fichiers.



- Page de jeu : télécharge 11 fichiers pour une durée de 523 ms



- Page des scores : télécharge 7 fichiers pour une durée de 392 ms



Step 1 – The Component architecture

Q : Component-oriented programming for the web is considered more maintainable. Why ?
A : En web, la programmation orientée objet est plus recommandée parce qu'elle a plusieurs avantages.

Q : If you look at the source code, every JS file wraps its code into a closure:
Try to remove the 2 closures from both card.component.js & game.component.js. What happens? Why?
Once figured out, remove the extra variable that makes the code to crash.

A : Les closures permettent à une fonction accéder à d'autres variables que celle directement à sa portée, alors qu'ici nous n'en avons pas besoin car on utilise des variables hors de la classe et des fonctions, raison pour laquelle il ne se passe rien quand on supprime les 2 closures dans card.component.js & game.component.js

Step 2 – NPM

Q : As you can see, npm install command also generated a package-lock.json file along with package.json. What is the purpose of this file?

A : le fichier package-lock.json est un fichier qui est généré lorsque vous apportez des modifications dans le fichier package.json ou dans l'architecture du dossier node_modules. Ce fichier stocke la structure exacte de votre arborescence de dépendances à chaque installation.
Ce fichier conçu spécifiquement pour les projets collaboratifs permet de gérer le fait que le dossier node_modules ne soit pas dans les sources comités mais également que les utilisateurs du projet puissent revenir en arrière et trouver les dépendances qu'ils avaient à un instant T.

Q : By Convention, all NPM dependencies use a 3-digit format for version numbers. How is it called? Can you explain the meaning of the ^ symbol next to bootstrap version?

A :
- Ces 3 digits sont appelés major, mineure et release.
- Le symbole ^ permet de dire que la version boostrap installée est une version compatible avec la version suivant le ^.

Q : What is a devDependency exactly? What are the differences with a depencency?

A : Les devDependencies sont des modules utilisés que pendant le développement alors que les dépendencies sont nécessaires à la mise en production.

Step 3 – ESNext

Q : Can you think of at least 2 things that are possible with Java classes, but cannot be done with ES6 classes?

A : ES6 apporte un nouveau style d'écriture avec les classes et comporte des différences au niveau de l'héritage.

Step 3.2 - ES6 Arrow functions

Q : What are the differences between var and let

A :
- Var est utilisé pour déclarer une variable.
- Let est utilisé pour déclarer une variable dans la portée est limitée à un bloc.

Q : What is the .bind(this) stuff? What does happen if you remove it? Is it needed when using an *arrow* *function* ?

A : .bind(this) permet d'exécuter la fonction à laquelle elle est associée. En retirant le .bind(this) d'une fonction, cette dernière n'est plus exécutée.

Step 4 – Promises -Async/await

Q : What are the advantages of Promises ?

A : les avantages du Promises sont :
- L' amélioration de la lisibilité du code.
- La bonne gestion des opérations asynchrone.
- Un meilleur flux de définition de contrôle en logique asynchrone.
- Une meilleure gestion des erreurs.

Q : Inside wich version of ECMAScript async/await has been released ?

A : Async/await a été publié dans la version ES6 d' ECMAScript.

Step 5 – Babel, transpilation

Q : What does the @ symbol mean in @babel/*?**

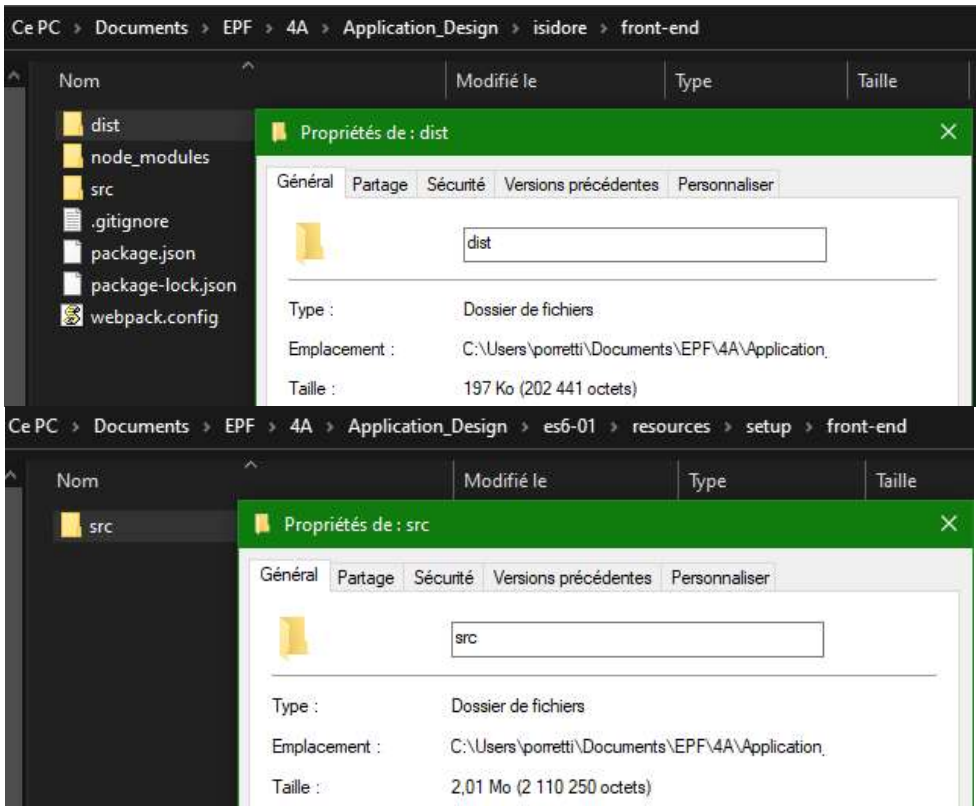
A : Le symbole @ dans @babel/*** permet de télécharger la dernière version enregistrée sur le registre npm.

Q : Look for files produced within dist/ folder. How did babel transpile your class WelcomeComponent?

A : Babel transpile la classe WelcomeComponent en utilisant des require, imports.

Q : What is the weight of the transpiled sources compared to your original sources?

A : Le fichier dist pèse 197 Ko alors que le dossier source original pèse 2.01 Mo.



Step 6 – Webpack & imports

Q : What is the difference between import * from './utils' and import { parseUrl } from './utils'?

A :
- import * from './utils' permet d'importer l'ensemble des des éléments présents dans utils
- import { parseUrl } from './utils' importe seulement les éléments parseUrl de utils

Step 6.2 – The bundler

Q : Why the utils.js will also be transpiled?

A : Le fichier utils.js sera egalement transpile pour pouvoir l'utiliser parceque GameComponent et WelcomeComponent utilisent tous les deux une methode parseUrl qui est importee de utils.js

Q : What does the webpack --config webpack.config.js do ?

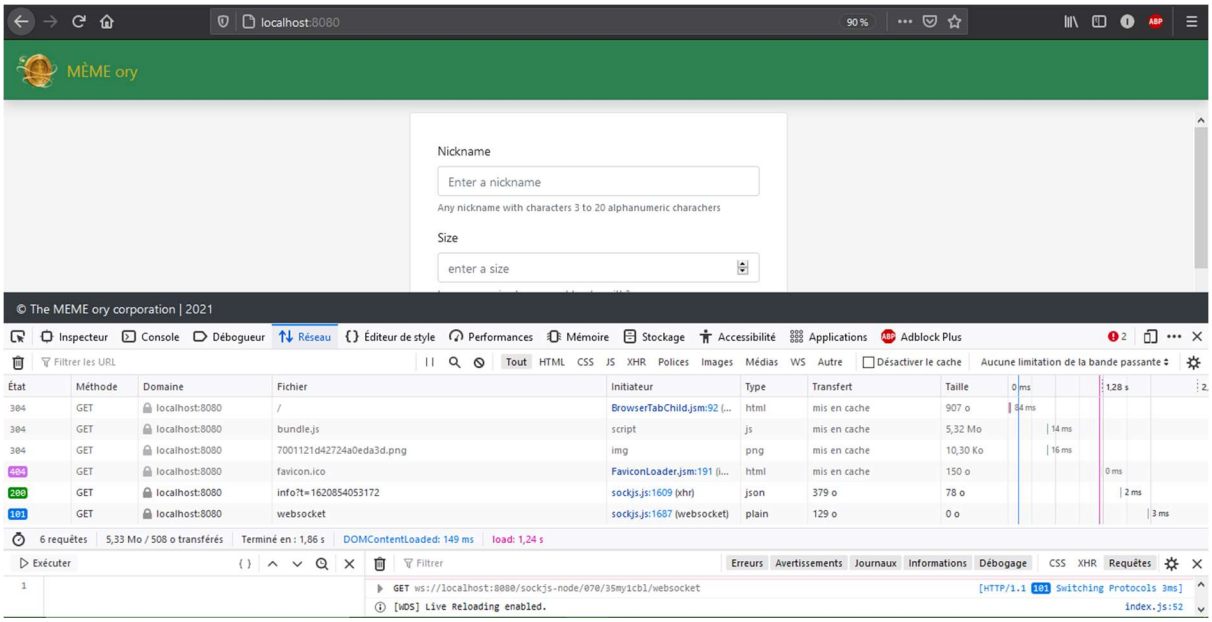
A : webpack –config webpack.config.js permet de créer un fichier de configuration

Step 7 - SPA

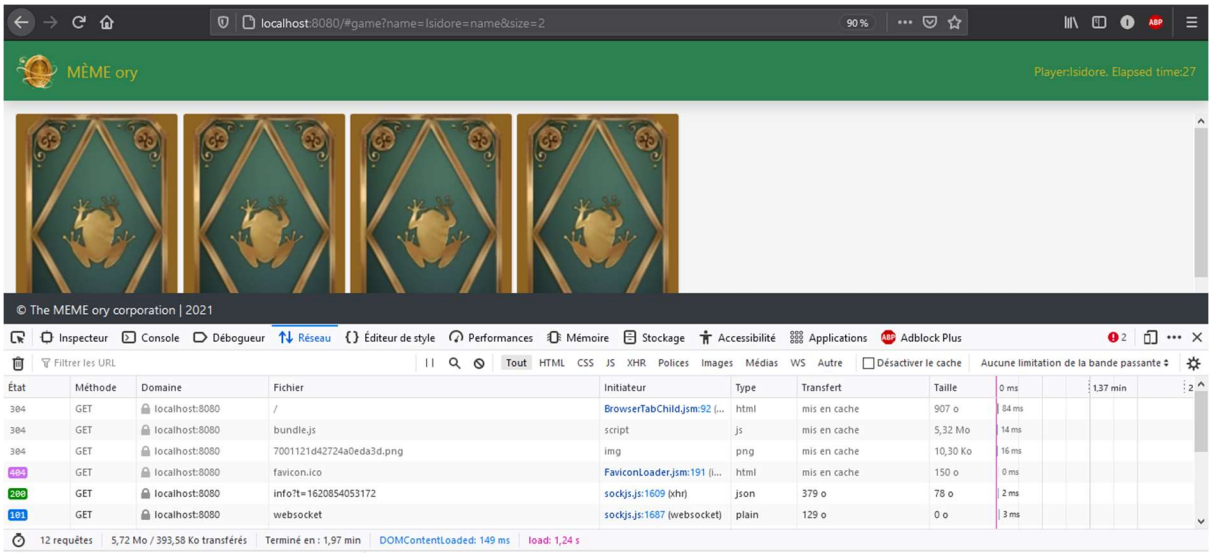
Q : Play the whole game with size=2. By browsing the 3 views of the application, how many files did your browser download in total? How many time did it took to load them all?

A :

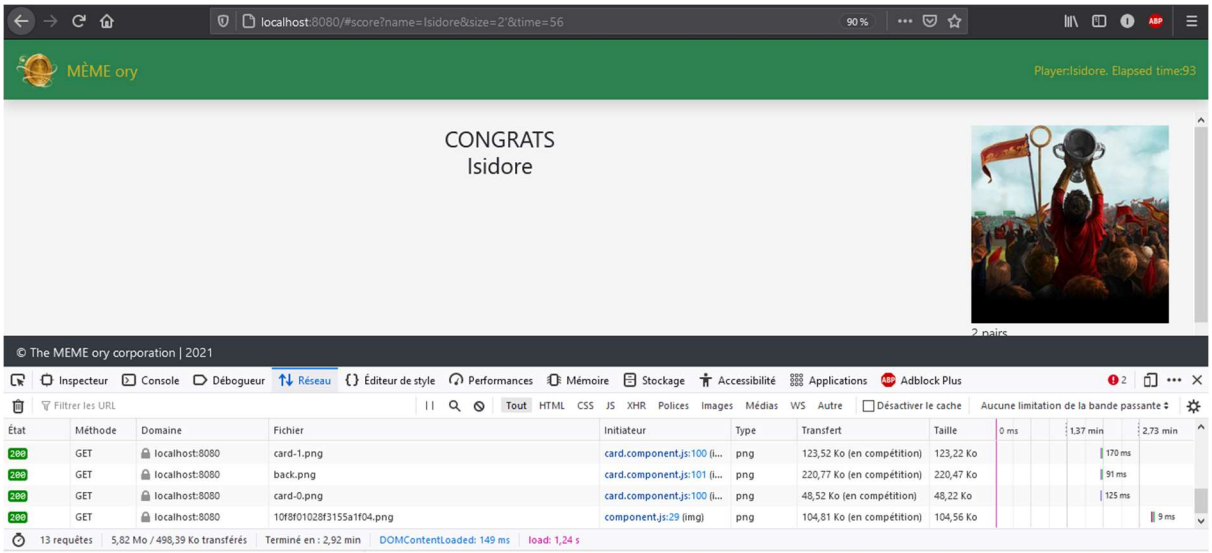
- Page d'accueil :



- Page de jeu :



- Page des scores :



- Au total, le navigateur télécharge un total de 13 fichiers en 1,24s pour le jeu avec 2 cartes.

A titre de comparaison le navigateur charge pendant 19 fichiers dans le même temps pour 9 cartes.

Step 8 – Style the application

Q : Can you guess how style-loader works exactly?

A : Un loader, est un programme qui va permettre à Webpack de pouvoir gérer un certain type de fichier**. W**ebpack ne comprend pas nativement le css. Avec le style-loader, le css est désormais compris et géré par Webpack.

Q : What does the _ prefix means on a sass file?

A : Lorsque l'on ajoute le préfixe _ devant le nom du fichier, il ne sera pas généré dans le CSS à moins d'être importé dans un fichier sass prinicpal. C'est un excellent moyen de modulariser votre CSS et de faciliter la maintenance.

Problèmes rencontrés :

- Problème de version sur mac, notamment au niveau des dependencies, avec une version du core-js ultérieur à ^3.
- Difficultés rencontrées au niveau d'installation de modules en raison d'un autre projet en javascript que l'on avance en parallèle.
- Step 3 : Problème avec les nested blocks, lorsque nous avons mis les card-wrapper dans les card-cmp cela fonctionne mais lorsque j'ai voulu mettre tout ce qui appartient aux card-wrapper, cela pose des problèmes. Nous avons dû annuler la plupart des modifications afin de retrouver un projet jouable.