Dyana DOUKAN

David ANATON

BattleShip

Manuel développeur

1. Table des matières

2. Historique des modifications

Name	*Date (dd/mm/yy)*	*Reason For Changes*	*Version*

3. Introduction

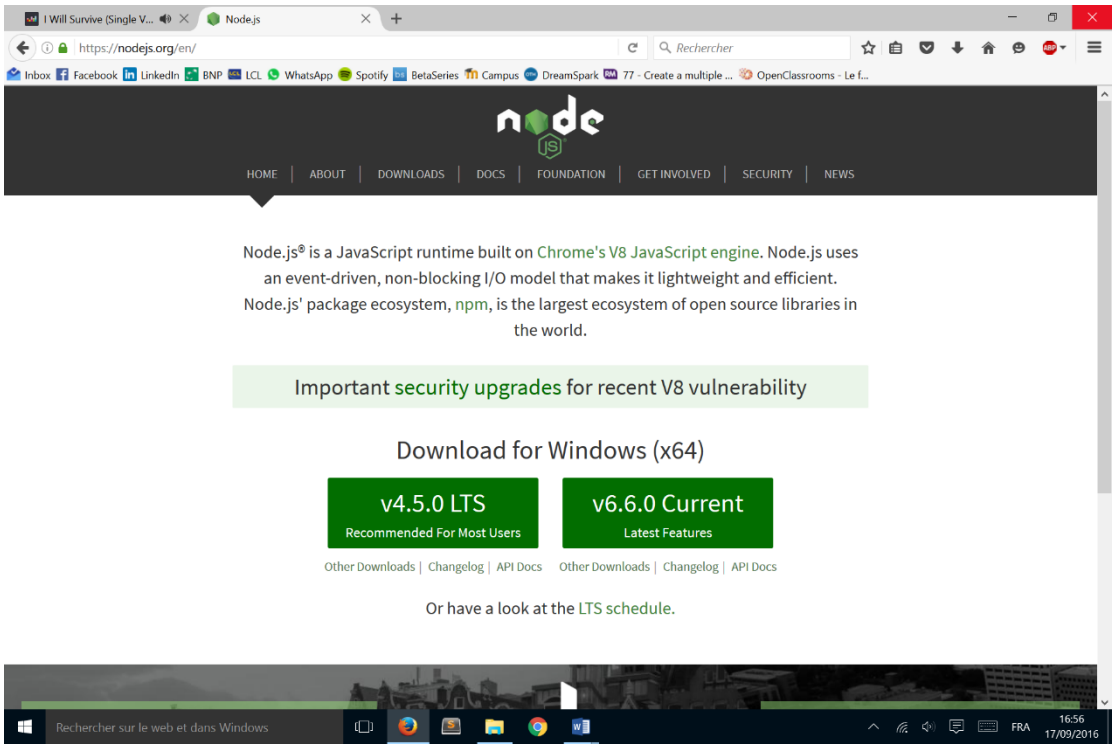
Ce manuel est là pour expliquer l'approche utilisée pour développer l'application BattleShip permettant de faire jouer simultanément 2 joueurs au jeu de bataille navale. Pour pouvoir jouer, ces 2 joueurs doivent être reliés par un serveur.

4. Serveur

Comme convenu, le serveur a été écrit en utilisant Node.js.

4.1. Installation de Node.js

Tout d'abord, aller sur http://nodejs.org/en/ et télécharger une des versions proposées.

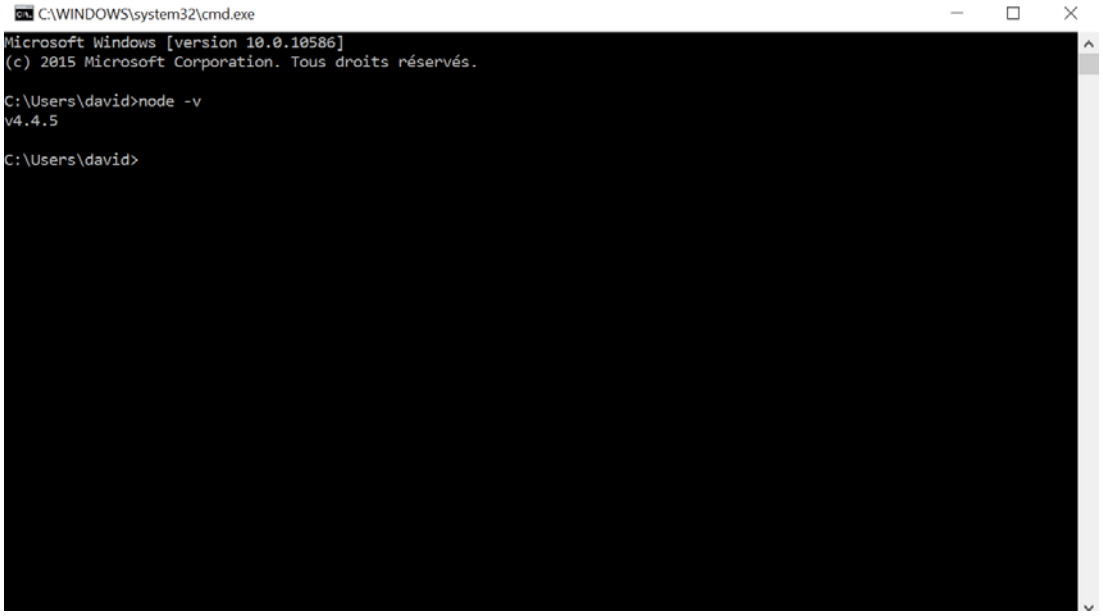


Lancer l'exécutable. Toujours cliquer sur "Next" et finir par "Finish" ; aucun logiciel tiers n'est installé par défaut.

4.2. S'assurer que l'installation a fonctionné

Sur Windows, taper Windows+R et lancer "CMD" pour ouvrir une nouvelle invite de commande.

Vous pouvez ensuite lancer "node -v". Vous devriez voir apparaitre le numéro de version de votre installation.

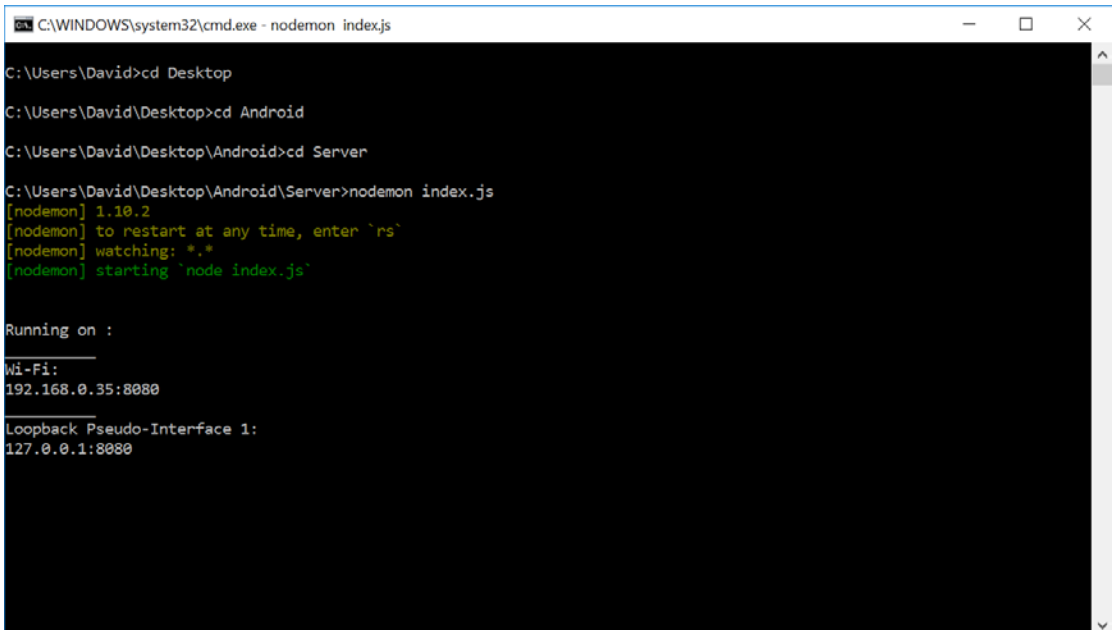


Sinon, vous devez ajouter node à votre variable d'environnement PATH (suivre ce tutoriel)

4.3. Lancer le serveur

Ouvrir une nouvelle invite de commande, naviguer jusqu'au dossier de l'application correspondant au serveur (appelé « Server »). Commencer par installer les dépendances nécessaires pour le serveur : lancer « npm install ». Cela va installer les dépendances déclarées dans le package.json.

Lancer « node index.js ». Vous devriez voir apparaitre les IP disponibles pour se connecter au serveur.



4.4. Travail effectué

Le serveur se constitue d'un seul fichier principal `index.js`. et d'un dossier models qui contient les implémentations des différentes « classes » utilisées dans notre projet. Nous avons, en effet, adopté une méthode de programmation permettant de se rapprocher d'un développement Java.

La majeure partie du code est commentée et parle d'elle-même. Nous tenions cependant à souligner quelques parties du code.

4.4.1. index.js

Ce dernier contient toute la logique du serveur.

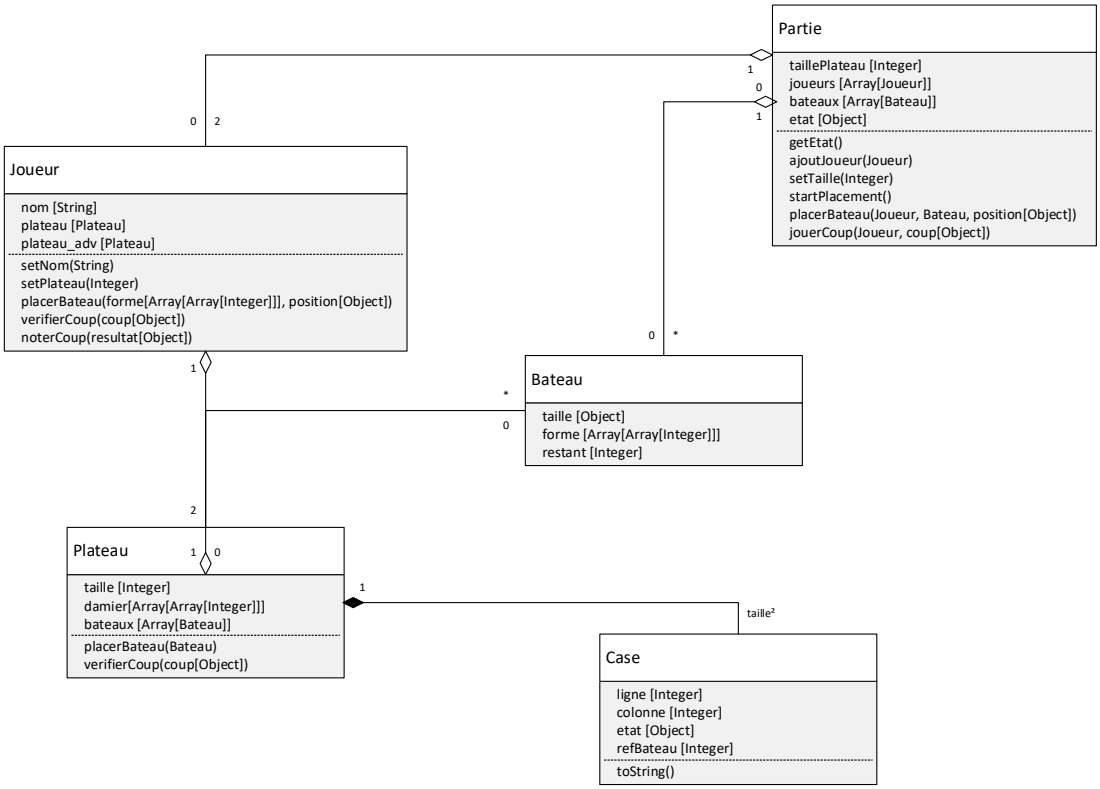
Pour effectuer le travail demandé, nous avons utilisé express. Express est un framework qui permet de s'affranchir des requêtes de bas niveau pour se concentrer uniquement sur le cœur applicatif et la communication par sockets. Il n'est utilisé que pour envoyer la page HTML lorsque le client demande la racine (« / ») de l'application web.

Nous avons également utilisé Socket.io qui permet de garder une socket ouverte entre le serveur et le client. Les paquets reçus du clients sont alors traités comme des évènements.

4.4.2. Dossier models

Comme expliqué précédemment, nous avons opté pour un développement orienté objet en javascript. Nous avons donc commencé par réaliser une étude relationnelle des différents objets de l'application.

4.4.2.1. Diagramme de classes UML



Nous avons ensuite implémenté les différentes classes.

5. Côté client

Le côté client se décompose de plusieurs fichiers.

5.1. views/index.ejs

Contient le code HTML permettant d'afficher des informations. Il est modifié depuis le code javascript et n'est pas intéressant à lui tout seul.

5.2. static/css/style.css

Fichier de mise en forme et de choix des couleurs. Inutile d'un point de vue algorithmique.

5.3. static/js/jquery.js

Importation de JQuery pour faciliter la manipulation de l'HTML en javascript. Téléchargé depuis https://jquery.com/

5.4. static/js/script.js

Ce fichier est celui qui représente le cœur applicatif côté client. Il est découpé en 3 blocs. Un premier qui s'exécute pour permettre le choix du nom d'utilisateur, un second pour les évènements reçus de la socket et, enfin, un dernier pour générer le code HTML représentant les données reçues du serveur.

6. Améliorations envisagées

6.1. Multijoueur

6.2. Meilleure interface graphique

6.3. Relancer une partie

6.4. Chat en ligne

7. Manuel Utilisateur

7.1. Serveur

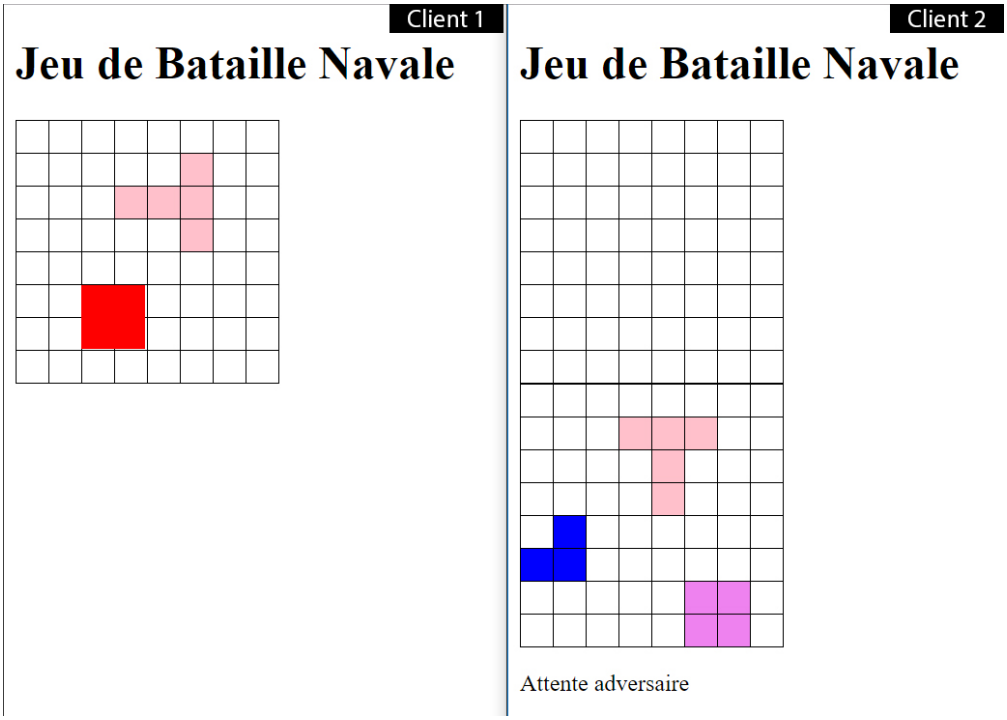
En tant que serveur, je dois simplement lancer la commande « `node index.js` ».

7.2. Client

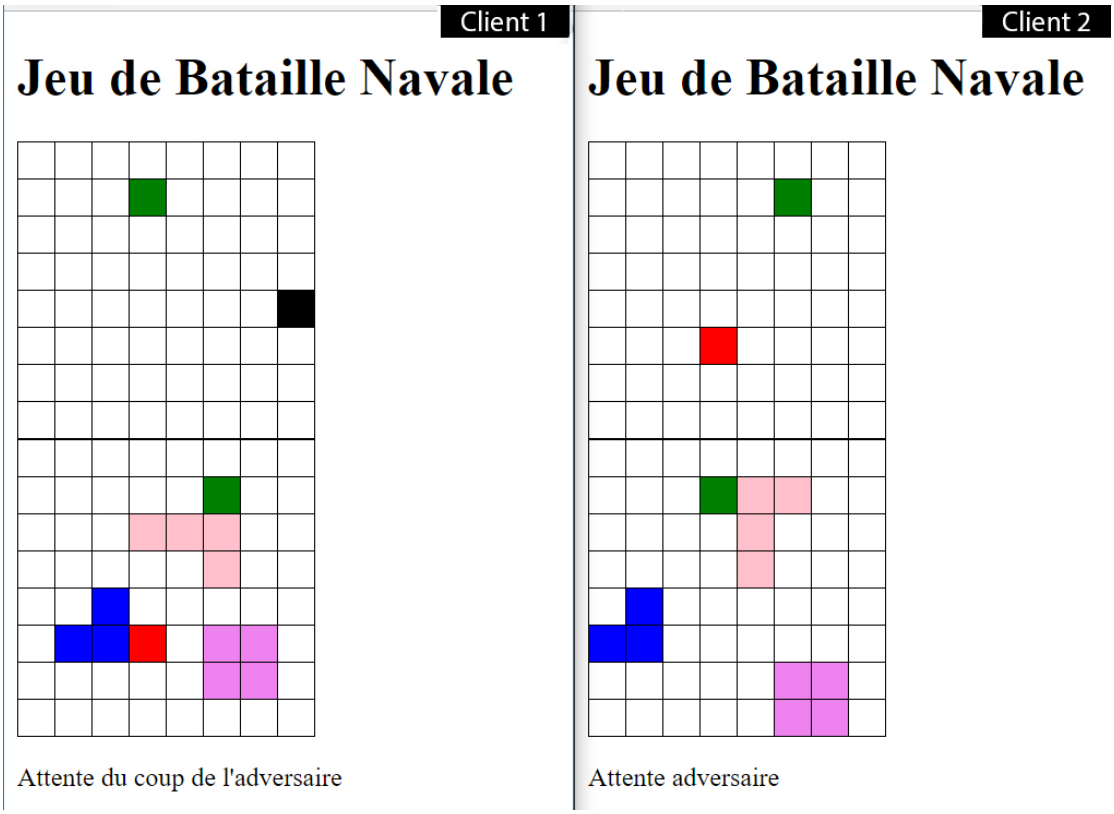
- Se connecter sur l'adresse du serveur. (Pensez à utiliser le bon port)

- Entrez un nom et envoyez

- Attendre un adversaire
- Quand ce dernier arrive, vous pouvez placer vos bateaux
  - Pour ce faire, utilisez les flèches du clavier, la barre d'espace pour tourner les bateaux et la touche Entrée pour valider votre choix.



- Quand ce sera votre tour, vous pourrez cliquer sur le plateau de votre adversaire.
  - Une couleur rouge signifie que vous avez raté votre coup ; Attendez celui de votre adversaire.
  - Une couleur verte montre que vous avez touché un bateau.
  - Lorsque ce dernier est coulé, il devient bleu.



Bon Jeu !