



DEPARTMENT INFORMATIQUE

MIN17217 - APPLICATION WEB ET SÉCURITÉ

---

## SlayerJS : A Nordic Card Game

---

*Authors :*

RAKOTOMANGA Andrianina  
CHIKAR Soufiane  
HU Xunyue  
ALI AMEDHI Mycipssa

MAY, 2021

---

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Jouabilité</b>	<b>1</b>
<b>3</b>	<b>Technologies et Concept</b>	<b>2</b>
3.1	Structure de l'application (M.E.R.N) . . . . .	2
3.1.1	REACT.JS . . . . .	2
3.1.2	MongoDB/NoSQL . . . . .	2
3.1.3	NODE.JS/Express . . . . .	3
3.1.4	NVM/NPM . . . . .	3
3.2	Bibliothèque et Extension supplémentaire . . . . .	3
3.2.1	Redux/Konva . . . . .	3
3.2.2	Sass/SCSS . . . . .	3
3.2.3	ESLint . . . . .	3
3.2.4	Sinter/Prettier . . . . .	4
3.3	Déploiement . . . . .	4
3.3.1	Ngrok . . . . .	4
<b>4</b>	<b>Contrainte et Difficultés</b>	<b>4</b>
<b>5</b>	<b>Rendu Finale</b>	<b>5</b>
5.1	Choix du personnage . . . . .	5
5.2	Choix du chemin . . . . .	5
5.3	Ecran de combat . . . . .	6
5.4	Ecran de trial . . . . .	6
5.5	Ecran de bonefire . . . . .	7
5.6	Ecran de Shop . . . . .	7
<b>6</b>	<b>Conclusion et Nouveau Horizon</b>	<b>7</b>
	<b>Bibliography</b>	<b>9</b>
	<b>Table des figures</b>	<b>9</b>

---

## 1 Introduction

Notre projet web est un jeu de style rogue-like au tour par tour basé sur des jeux de cartes style hearthstone, magic.

Le jeu sera jouable en Solo où l'on peut choisir entre trois personnages. Chaque personnage aura un ensemble de cartes différents stockées dans son deck. Le but sera d'atteindre et battre le boss pour finir le jeu.

Le jeu sera disponible sur la page web lié à github et hébergé par Heroku/Netlify ainsi que ngrok.

## 2 Jouabilité

Le jeu est donc un rogue-like avec un système de carte. L'univers est basé sur la mythologie Nordique, et les héros ainsi que les monstres s'inspirent de celle-ci.

Chaque personnage (héros/monstres) possède une barre de vie, de l'armure ainsi qu'un taux de pièce qui servent à acheter des nouvelles cartes pour les héros. Les héros récupèrent des pièces quand un monstre est battu.

Les héros possèdent de l'énergie, ainsi qu'un deck, indispensable pour utiliser les cartes.

Les Héros sont au nombre de 3 (Siegfried, Owl et Brunehilde) chacun spécialisé dans un domaine, l'attaque (le personnage possède plus de carte attaque mais moins de carte défense), la défense, et les effets.

Les cartes sont des actions utilisables par le joueur. Elles consomment de l'énergie. Il y a 4 grands types de cartes.

Les cartes Attaque qui infligent des dégâts.

Les cartes de Défense qui augmentent les points de bouclier (ou utilisent ses points de bouclier).

Les cartes Soin/Drain qui font récupérer des points de vie ou voler des points de vie.

Les types de cartes peuvent être combinés ce qui donne un grand nombre de possibilités malgré le peu de type de cartes actuellement implémentées.

Pour atteindre le boss il existe plusieurs chemins possibles, l'utilisateur doit passer par des salles qui sont de 6 types.

Les salles de type "fight" sont des combats contre des monstres faibles.

Les salles de type "bonfire" sont des lieux de repos où l'on peut soit regagner de la vie ou se purger des effets sur le héros (étourdissement, affaiblissement)

Les salles de type "trial" sont des salles à choix multiples qui nous demandent de choisir entre deux options qui peuvent soit avantagé soit désavantagé le joueur pour le reste de la partie (augmentation de points de vie ou paralysie des joueurs)

Les salles de type "shop" permettent d'acheter des cartes contre des pièces

Les salles de type "élites" sont des combats contre des monstres intermédiaires

Enfin La salle de type "boss" qui signifie la dernière salle de la session du jeu.

Les combats se déroulent au tour par tour. Les joueurs utilisent des cartes selon l'énergie qu'ils possèdent. Un tour est terminé quand le joueur n'a plus d'énergie ou de cartes ou si le joueur décide de terminer le tour.

Les mécaniques ainsi que les données du jeu étant assez importantes, les explications détaillées de

---

l'histoire des personnages, cartes et monstres sont détaillés sur le github de l'application(mettre lien).

### 3 Technologies et Concept

#### 3.1 Structure de l'application (M.E.R.N)

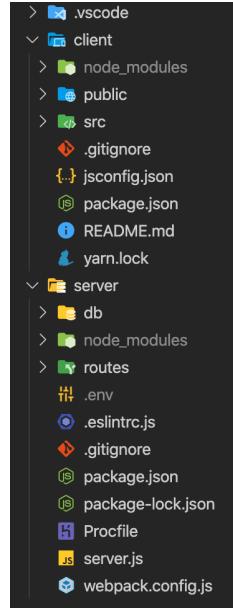


FIGURE 1: Arborescence du projet.

Source:

Le projet suit le paradigme de la pile de logiciel MERN 2021 (MongoDB, Express, React, Node), une structure permettant de gerer efficacement l'interaction client/server/Base de données avec Express et MongoDB en NoSQL ainsi que REACT.JS, un framework client efficace qui permet une gestion de site web monopage en actualisant cette seule page à l'aide de changement d'état.

##### 3.1.1 REACT.JS

REACT 2021 est un framework en javascript qui intègre du html contrairement à la plupart de ses concurrents, ce qui permet une meilleur optimisation des calculs et lisibilité du code en général. La possibilité d'intégrer du HTML directement dans du Javascript permet un simplicité et une efficacité dans l'écriture, la lecture et l'interprétation du code. La conception en "Components" permet également d'avoir un rendu graphique clair et indépendant ainsi que extensible pour chaque composants du graphique de la page.

Comme React est un framework rapide en execution par rapport à AngularJS ou VueJS il était adapté à la conception d'un jeu qui demande une efficacité algorithmique et temporelles conséquentes.

##### 3.1.2 MongoDB/NoSQL

MongoDB est un logiciel de base de données basé sur le paradigme NoSQL ORACLE 2021. Cet ensemble est particulièrement adapté à nos besoins car MongoDB traite principalement des fichiers

---

BJSON (JSON Binaire) donc des fichiers de types de documents complexes qui nécessiteraient plusieurs tables et opérations pour recréer dans une base de données relationnelle. Nos données de jeu (Cartes, Personnage, Musique, Monstres) étant stockés comme des structures en tableau NoSQL permet une simplification du stockage de celle-ci car le NoSQL n'a pas besoin de structure pour stocker ses données.

### **3.1.3 NODE.JS/Express**

NODE 2021 est le framework côté serveur de javascript, il permet de faire interagir l'application avec son serveur et sa base de données. Nous utiliserons le framework EXPRESS 2021 de Node.js pour connecter notre base de données Mongo à l'application.

L'application est donc intégralement codé en Javascript (React/Node) ce qui permet une continuité du code tout au long du projet ainsi que d'une meilleure intellegibilité et d'une clarté continue.

### **3.1.4 NVM/NPM**

Le gestionnaire de version de node.js NVM ainsi que le gestionnaire de paquet pour node npm NPM 2021 nous ont permis de construire efficacement une architecture solide et viable de notre application en installant toutes les dépendances requises durant notre projet via create-new-app. Son fonctionnement est clair et simple, ce qui permet de tester ainsi que de debugger rapidement le code grâce notamment au tracage d'erreur javascript de node/react.

## **3.2 Bibliothèque et Extension supplémentaire**

### **3.2.1 Redux/Konva**

Afin d'utiliser pleinement React, nous avons choisis d'utiliser ses extensions pour améliorer le rendu graphique ainsi que gerer et manipuler des actions et des données rapidement.

Konva permet de creer des canvas en ReactJS, ce qui nous permet de creer des animations, dessiner des formes ou manipuler des images.

Redux permet de gerer de manière centralisé les données ainsi que les fonctions utiles au fonctionnement du jeu. Les fonctions "dispatch" permet d'expédier une action qui lui, est une fonction qui est un objet javascript qui décrit ce qui doit se passer dans le code.

### **3.2.2 Sass/SCSS**

SASS 2021 est un préprocesseur de CSS qui permet d'améliorer le style d'une application web notamment en simplifiant la création d'animation, du déploiement de css avec l'ajout de variable utilisable en grande portée (import styles), mais aussi en rajoutant des opérations algorithmes de base pour générer des styles plus complexes avec des conditions, boucles, branchement ou fonction.

Avec Sass nous avons pu concevoir des animations ainsi que certains de nos personnages entièrement en SCSS (aucun sprite utilisé) de manière rapide et simple.

### **3.2.3 ESLint**

Dans notre opération pour un code intelligible et propre, la première étape fut de constituer une pile de logiciel/bibliothèque testant la syntaxe ainsi que la continuité du code durant le cheminement du projet.

---

Notre premier outil principal a été ESLINT 2021, un analyseur syntaxique personnalisable qui permet d'éviter des erreurs triviales de code ainsi que de soigner ce dernier en signalant toutes syntaxes peu lisibles. ESLint permet également de reduire des algorithmes, fonctions ou lignes de code quelconque à son plus simple et comprehensible apparat. La fonction la plus intéressante reste l'uniformisation du code selon des règles prédefinis (exemple : utilisation de lambda, accolade à la ligne etc) ce qui permet à une équipe de se retrouver avec un code suivant les même règles donc plus facile à prendre ou reprendre en main.

### 3.2.4 Sinter/Prettier

Sinter et PRETTIER 2021 sont les deux dernières bibliothèque d'analyse statique de code. Elle permettent de vérifier la syntaxe et notamment réduire la longueur de ligne automatiquement. Ces deux bibliothèques sont simple et pratique à utiliser car elle marche en plug and play, il suffit donc de les lancer une fois pour qu'elles s'appliquent à chaque compilation.

## 3.3 Déploiement

Afin que l'application soit disponible pour tous, il est nécessaire de déployer celle-ci en adéquation avec le projet et la portée de ce dernier. Nous avons choisi l'alternative gratuite Ngrok, qui prend avantage du service web de déploiement de github pour stocker le code sur le cloud et également capable de faire interagir le code client et serveur.

### 3.3.1 Ngrok

NGROK 2021 est un service gratuit de hosting temporaire. Il est initialement utiles pour tester ou montrer une version de l'application au client, mais c'est également un moyen efficace et sécurisé d'héberger un site web comprenant des interactions client-serveur.

Ngrok est déjà disponible dans le package npm. Le principe est de fournir un tunnel tcp sécurisé vers l'extérieur (internet) de son adresse local (localhost) d'hébergement de son application. Grâce à cela tout les utilisateurs du web peuvent accéder gratuitement à une application héberger localement.

Ce n'est pas une alternative définitive à l'hébergement et au déploiement d'une application, mais cela permet d'avoir un moyen sécurisé et gratuit de déploiement d'une application sur l'internet.

## 4 Contrainte et Difficultés

Tout projets ou développements rencontrent des difficultés à tous les étages du cheminement. Le projet devait se faire sur le web avec des technologies serveur/client ainsi qu'une possibilité de déploiement public sur internet. Nous avons exposé nos solutions dans la section précédente pour ces différentes contraintes ainsi que l'explication de nos choix.

Le choix du type d'application fut également une source de réflexion, sachant qu'il fallait avoir une vraie interaction client serveur dans notre application. Le choix d'une application de type jeux de carte était bien adapté à la programmation web serveur/client, car nous pouvons exploiter tous les aspects de la programmation web, l'importance du rendu graphique attrayant, la communication et le stockage avec une base de données adapté, de l'inventivité ainsi que l'implémentation de nombreuses fonctionnalités possibles, ce qui permet une durée de vie de l'application ainsi que des possibilités d'extensions importantes.

Nous avons choisi d'utiliser MERN Stack car la pile logiciel complétait parfaitement les conditions de l'implémentation du projet.

---

En ce qui concerne les difficultés techniques, nous avons regrouper beaucoup de technologies prévues pour l'application qui nous ont été utile mais certaines ne nous ont pas servi pour la portée actuelle du jeu. Notamment les recherches du multijoueurs utiles mais non implémenté actuellement. Cependant d'autres technologie telles que Sass, Redux, ESLint, Ant Design ont été particulièrement utiles à notre avancement du projet notamment niveau rendu graphique.

En ce qui concerne les difficultés organisationnelle, le plus important était de clairement définir les fonctionnalités ainsi que la charte graphique du jeu, ce qui était faisable dans le laps de temps données et ce qui était améliorable ou à prévoir pour un développement externe de l'application. Le manuel du gameplay du jeu complet est disponible directement sur le github de l'application pour voir toutes les fonctionnalités, technologies et extensions possibles du jeu.

## 5 Rendu Finale

### 5.1 Choix du personnage



FIGURE 2: Choix du personnage.

Nous avons la possibilité de choisir entre trois personnage, le combattant, le mage, ou le défenseur.

### 5.2 Choix du chemin

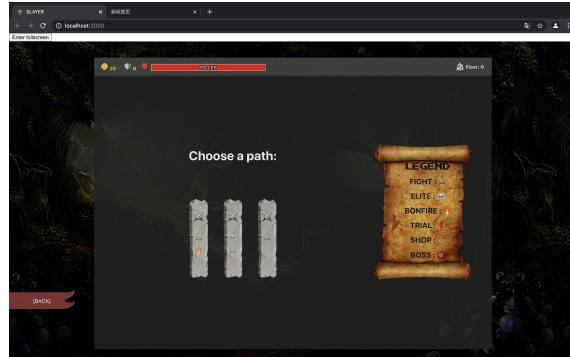


FIGURE 3: Choix du chemin.

Les chemins sont représentés par des lignes de chemin possibles (maximum 3). La première icône de chaque ligne représente le prochain tableau, les autres icônes plus bas représentent les chemins suivants après le prochain tableau.

La légende permet de se repérer sur quel type de niveau sont disponibles.

La barre d'information est disponible dans tous les écrans. Les icônes permettent de repérer facilement les significations des informations disponibles.

---

### 5.3 Ecran de combat

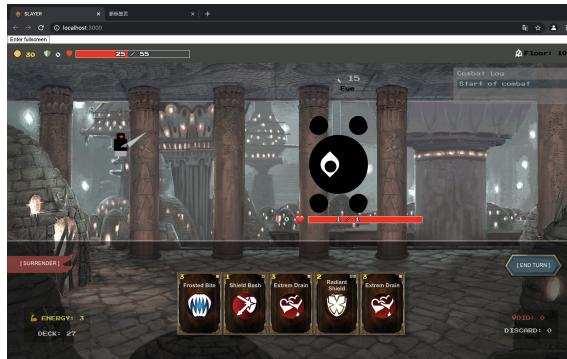


FIGURE 4: Ecran de combat.

L'écran se constitue simplement de trois zones. La zone du joueurs, la zone des monstres avec points de vie etat (paralysé, etc) ainsi que la zone des cartes, où on affiche l'energie restante, le nombre de cartes dans le deck, les cartes défaussé et les cartes non utilisés.

### 5.4 Ecran de trial

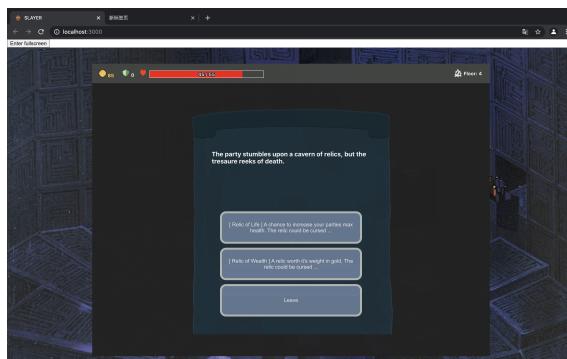


FIGURE 5: Ecran d'histoire.

L'écran d'histoire (Trial) est un choix de chemin qui peut soit donner un avantage ou un désavantage au personnage.

Il y a deux choix de chemin possible, chacun donne un bonus ou malus different en fonction du hasard.

---

## 5.5 Ecran de bonefire

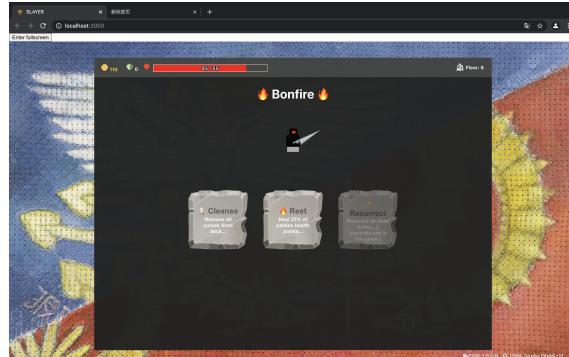


FIGURE 6: Ecran du bonefire.

Source:

L'écran de feu de camp permet de reposer son personnage sans combattre.

Cleanse : enlève tout les effets indésirable

Rest : récupère 25% de ses points de vie.

## 5.6 Ecran de Shop

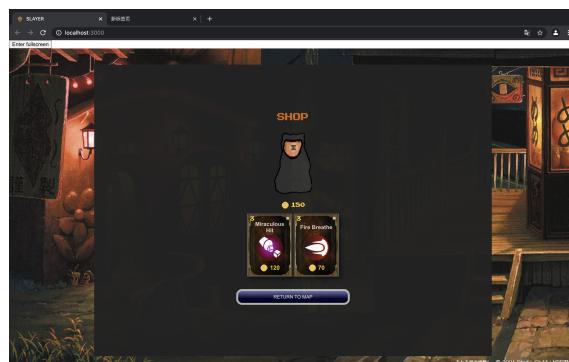


FIGURE 7: Ecran de shop.

Source:

Le shop donne la possibilité d'acheter des cartes contre des pièces.

## 6 Conclusion et Nouveau Horizon

Pour résumer, nous avons parfaitement rempli et réussi l'objectif de notre projet : produire puis déployer une application web stable avec des interactions client/serveur possédant de nombreuses fonctionnalités et extensions possibles.

1 héro ajouté, 38 cartes disponibles, 7 monstres codés, 6 types de niveaux jouables, 4 types de cartes utilisables (sans compter les cartes au propriétés multiples). Le jeu est donc jouable et attrayant même si l'horizon des possibles est encore important.

---

Il a fallu apprendre, apprivoiser et comprendre des technologies récentes, mais déjà correctement documenté, ce qui nous a facilité la tâches d'implementation de ces dernières.

A présent, nous connaissons mieux l'univers du développement web dans son ensemble, de l'affichage à la base de données.

Même si l'application est complète ainsi que fonctionnelle, les horizons d'extensions sont nombreux : ajout d'un mode multijoueur ainsi que d'une base d'utilisateur à enregistrer, ajouter de la complexité dans les mécanismes du jeu ; notamment ajout d'effet de carte, ajout de monde et de monstres avec des mécaniques de jeu différentes, mode multijoueurs local puis en ligne, possibilités d'interaction diverse entre joueurs, amélioration du rendu graphique, ajout de nouveau héros... Ces améliorations possibles sont déjà détaillées de manière non exhaustive sur le github du jeu.

---

## Bibliography

- ESLINT (2021). *ESLINT : Javascript LINTER*. URL : <https://eslint.org> (visité le 21 avr. 2021).
- EXPRESS (2021). *Express : A Node Web Manager*. URL : <https://expressjs.com/fr/> (visité le 21 avr. 2021).
- MERN (2021). *What Is MERN Stack*. URL : <https://www.mongodb.com/mern-stack> (visité le 21 avr. 2021).
- NGROK (2021). *Secure Introspectable tunnels to localhost*. URL : <https://ngrok.com> (visité le 21 avr. 2021).
- NODE (2021). *NodeJS*. URL : <https://nodejs.org/en> (visité le 21 avr. 2021).
- NPM (2021). *NPM/NVM : Version/Package Manager*. URL : <https://www.npmjs.com/package/nvm> (visité le 21 avr. 2021).
- ORACLE (2021). *NoSQL Database*. URL : <https://www.oracle.com/fr/database/nosql-cloud.html> (visité le 21 avr. 2021).
- PRETTIER (2021). *Prettier : Optiniated Code Formater*. URL : <https://prettier.io> (visité le 21 avr. 2021).
- REACT (2021). *ReactJS*. URL : <https://fr.reactjs.org> (visité le 21 avr. 2021).
- SASS (2021). *SASS/SCSS : Syntactically Awesome Style Cheet*. URL : <https://sass-lang.com> (visité le 21 avr. 2021).

## Table des figures

1	Arborescence . . . . .	2
2	Personnages . . . . .	5
3	Choix . . . . .	5
4	Combat . . . . .	6
5	Histoire . . . . .	6
6	Bonfire . . . . .	7
7	Shop . . . . .	7