PROJET WEB

« Les derniers survivants »

Tom CHANG

Abdelhak BEKHOUCHE

Guillaume THIBAUT

Ariel BITON 2020-2021

Sommaire

Mise en place du dossier 3

Mise en place de la base de données 4

Introduction 6

Les pages 7

- Accueil 7
- Authentification 9
- Profile 10
- Armurerie 11
- Combat 14
  - Zombie 15
  - Gameplay 16

Conclusion 19

Mise en place du dossier

Après s'être muni du dossier nécessaire contenant : bin, client, server et les packages, il est nécessaire d'exécuter ***npm install*** dans l'invite de commande sur le chemin où se trouve le dossier contenant les packages. On obtiendra alors les composants nécessaires au serveur Node.js.

Ensuite, effectuer ***npm install pg bcrypt*** pour les composants requis aux bases de données (PostgreSQL).

La dernière commande est ***npm install email-validator*** car nous voulons valider les emails un minimum, bien évidemment.

Dans ***./server/routes/api.js***, remplacer le password par le mot de passe de votre postgreSQL.

Le serveur local est sur le port ***3000.***

Très important :

Après avoir implémenté la base de données, lien doit être **http://localhost:3000/?#/**

Et non pas **http://localhost:3000/#/,** sans quoi, le site ne semblera pas fonctionner.

Mise en place de la base de données

Avant tout, il faut mettre en place notre base de données. Pour cela, il faut tout d'abord que vous ayez le fichier nécessaire : **items.csv**. Sur **pgAdmin-4**, créer une base de données appelée ***Survivors***. Dedans, descendre jusqu'aux tables dans l'arborescence, clic droit sur **Tables** et sélectionner **Query Tool** comme-ci :

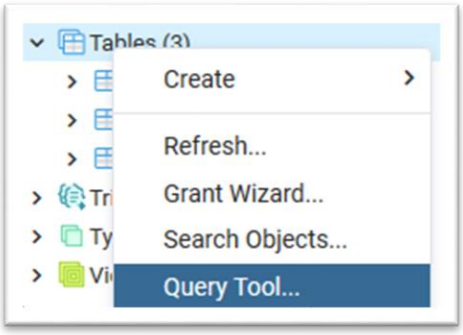

Figure 1: Query Tool

Il devrait s'afficher une page pour entrer des commandes. Il va falloir entrer les 3 commandes une par une ci-dessous :

```
CREATE TABLE items
(
        id integer,
        name text,
        description text,
        type integer,
        level integer,
        properties integer,
        image text,
        price integer
);
```

```
CREATE TABLE users
(
        id serial PRIMARY KEY NOT NULL,
        email text,
        password text,
        items integer[],
        equipped integer[],
        money integer,
        hp integer,
        stats integer[],
        date date
);
```

```
CREATE TABLE zombies
(
        id serial PRIMARY KEY NOT NULL,
        hp integer,
        def integer,
        type integer,
        prize integer,
        atk integer
);
```

Une fois fait, nous n'avons plus qu'à importer le fichier ***items.csv*** pour la table items. Pour cela, clic droit sur la table ***items*** puis sur ***Import/Export :***

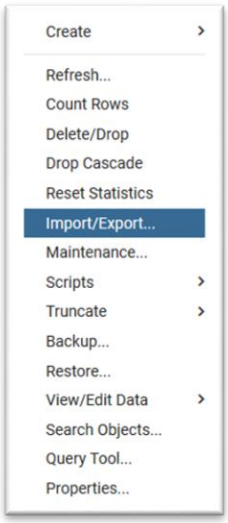

Figure 2: Import/Export

Enfin, il faut absolument que les trois éléments entourés soient exactement identiques à :

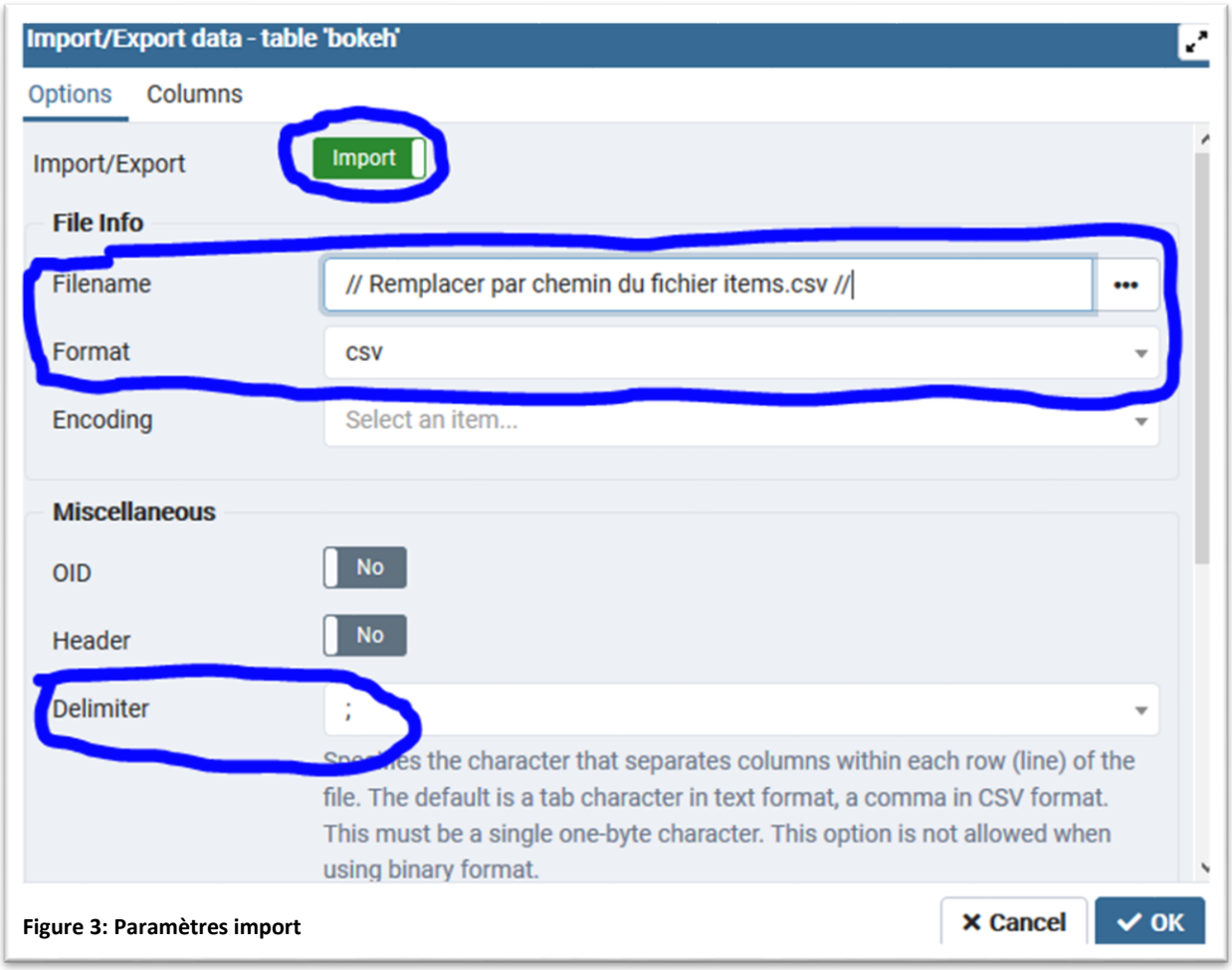

Figure 3: Paramètres import

En cas de doute, c'est bien **un point-virgule** (et non pas deux points) dans ***Delimiter.***

Si tout s'est bien passé, vous devriez avoir :

Figure 4: Import succès

et la table ***items*** complètement remplies. **La base de données est maintenant prête !**

Introduction

Pour ce projet web, nous sommes heureux de vous présenter notre site qui portera sur le thème de **zombies**.

C'est quoi « *Les derniers survivants* » ?

C'est un site-jeu où l'utilisateur incarne un survivant d'une apocalypse. Le but est de tuer des zombies pour amasser de l'argent.

Que peut-on y faire ?

Il y a un système d'équipements où l'utilisateur pourra équiper différents objets incluant 4 catégories : casque, armure, couteau et arme à feu qui changeront ses attributs comme l'attaque ou la défense. Nous avons donc implémenté un magasin où l'utilisateur pourra acheter et équiper des objets. Chaque joueur aura accès à sa page de profile contenant diverses informations. Pour avoir accès à tout cela, l'utilisateur devra bien sûr s'authentifier avec une adresse email de forme valide, sans quoi il sera constamment redirigé vers l'accueil.

Comment y jouer ?

C'est ce que nous découvrirons à travers la présentation des différentes pages de notre site. Pour ce qui est de la navigation entre les pages, l'utilisateur aura une barre de navigation contenant les boutons.

Pour rappel, l'objectif de base est de créer un site contenant différentes pages et un server avec une API. Un système d'authentification est obligatoire et la persistance des données se fait au moyen de la BDD (**PostgresSQL**) qu'on communique avec grâce au langage SQL. Notre site sera en effet une Single Page Application avec un serveur web **Node.js**.

Les pages

– Accueil –

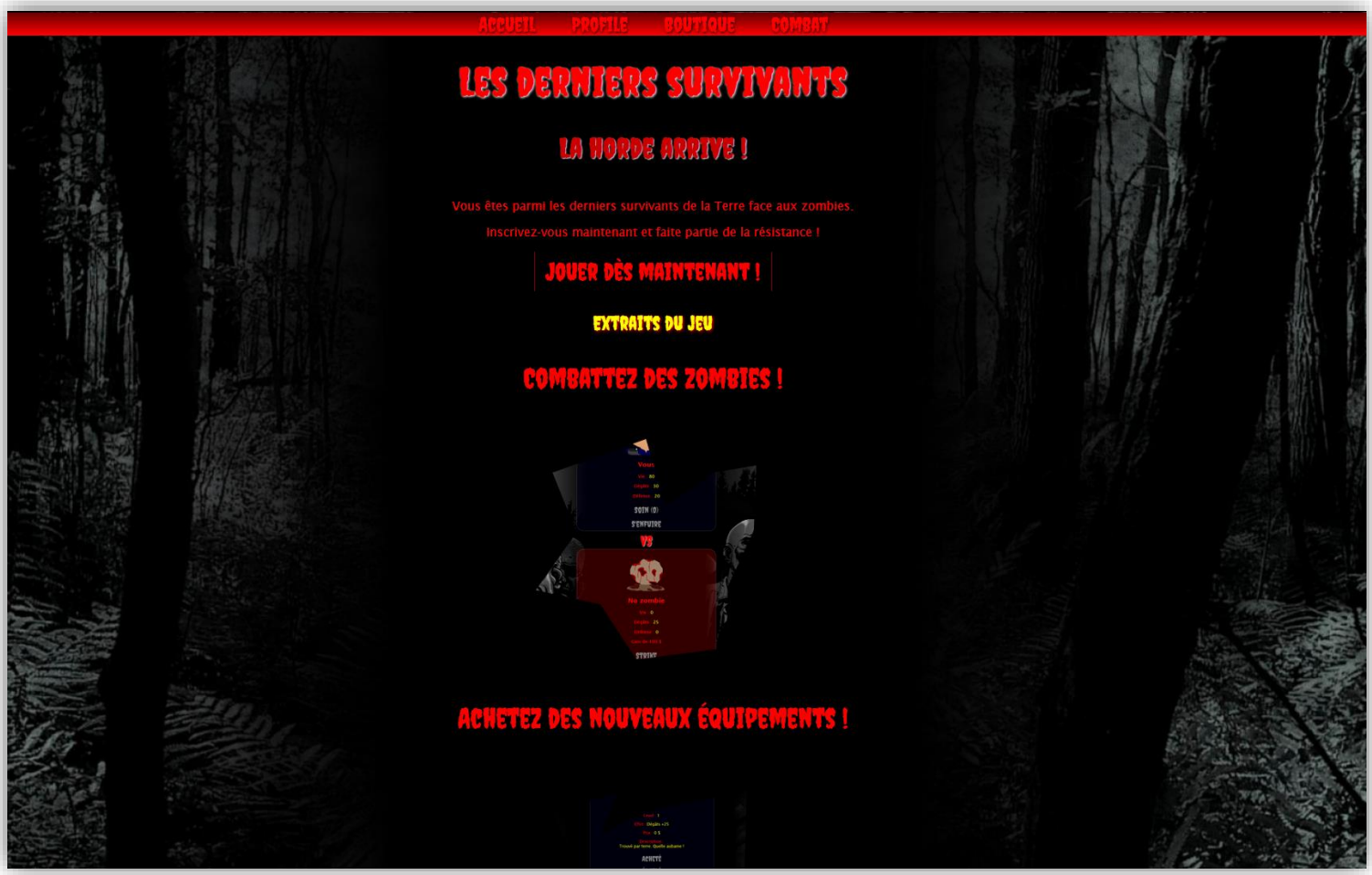

Figure 5: Interface de l'accueil

Pour notre première page, qui s'appellera ***Accueil.vue,*** on donnera le thème du site dès le moment où l'utilisateur charge la page (ce qui est évident) : le choix du fond est une forêt inquiétante. Le gradient noir au centre permet de mieux visualiser le texte : le nom du site, les petits slogans comme « **La horde arrive !** » etc ainsi que le bouton **Jouer dès maintenant**, bouton omniprésente dans tous les jeux de navigateur, on a alors le nôtre désormais.

Nous avons inséré des images, des captures du site pour montrer à quoi l'utilisateur peut s'attendre comme contenus. En passant la souris dessus, l'image reprend sa forme normale.

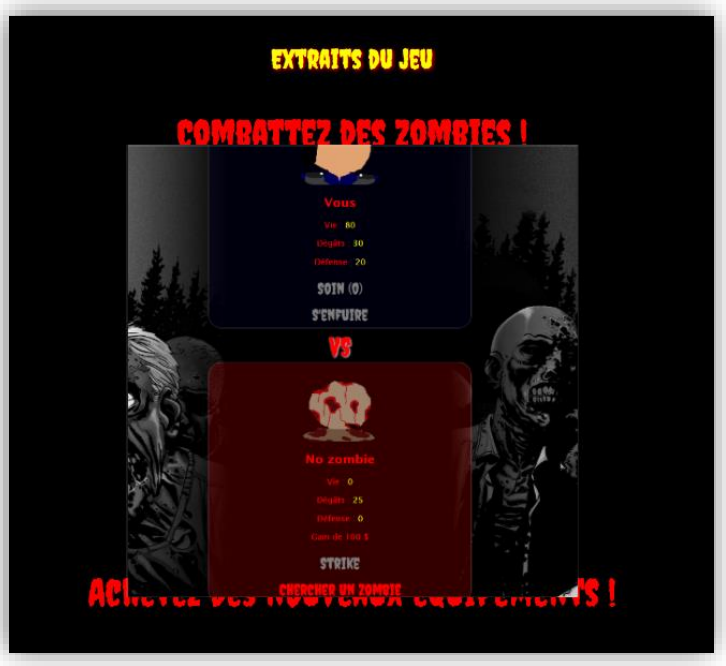

Figure 6: Zoom sur le gameplay

Au niveau des données, ce qui intéresse cette page c'est seulement si l'utilisateur est connecté ou non, afin d'afficher correctement la barre des tâches. Pour cela, il suffira que le server nous renvoie l'id de l'utilisateur :

L'utilisateur possède un id et est donc **connecté** :

Figure 7: Barre de navigation (connecté)

Ou l'utilisateur ne possède pas d'id et est donc **non connecté** :

Figure 8: Barre de navigation (déconnecté)

L'accès à une page qui ne se trouverait pas dans la barre de navigation provoque une redirection automatique vers l'accueil.

– Authentification –

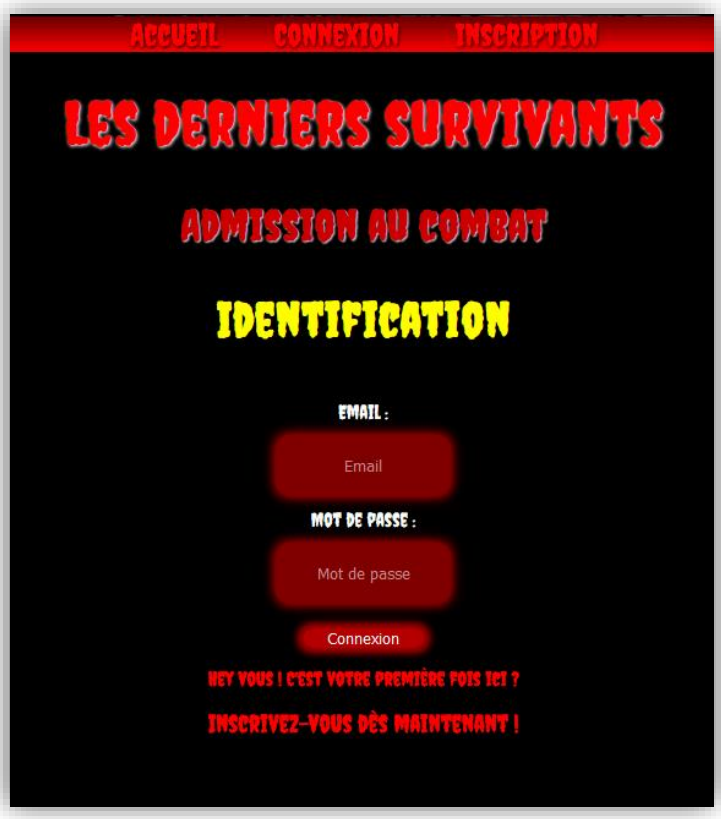

Figure 10: Login

Figure 9: Register

Voilà les deux pages sœurs : « **Login.vue** » et « **Register.vue** ». Le fonctionnement est simple, l'utilisateur entre des données qui seront traité dans le serveur : une requête pour s'inscrire vérifie d'abord la validité du mail et pour cela nous avons simplement besoin d'effectuer des requêtes SQL afin de savoir si le mail est déjà pris ou non. Dans ce cas, la page le montrerait : redirection vers la page de login si l'inscription s'est bien effectuée, sinon :

Figure 11: Email Invalide

On présente bien qu'on a un cas invalide.

Idem pour le login, rien de se passe si l'utilisateur n'entre pas de bons identifiants, sinon il est redirigé vers la page d'accueil avec cette fois-ci de nouveaux boutons sur la barre de navigateur.

– Profile –

Nous commençons à nous approcher des parties intéressantes.

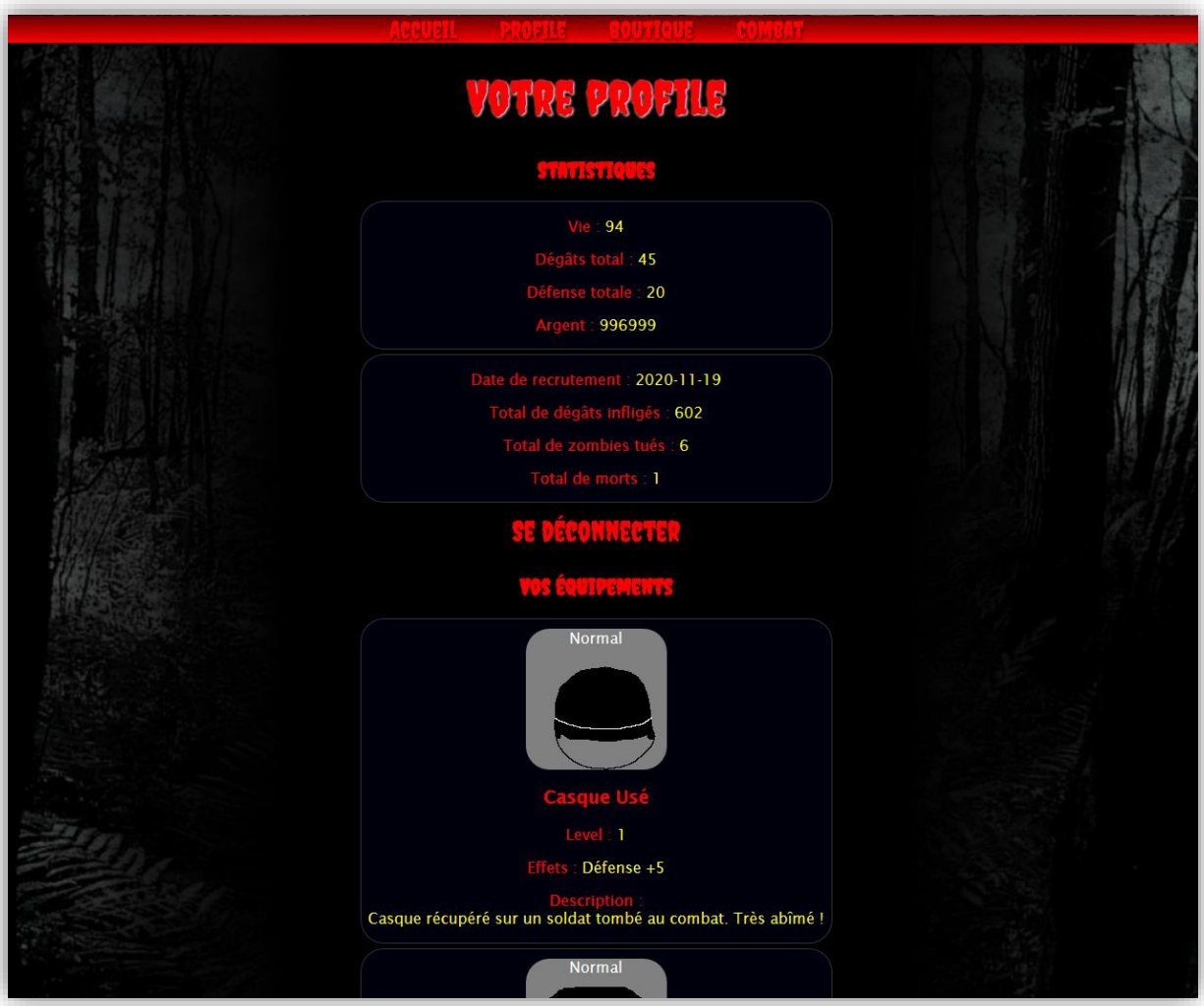

Figure 12: Interface du profil de l'utilisateur

Voici le profil du joueur, il s'agit de la page « **Profile.vue** », où toutes sortes d'informations y paraissent : vie, attaque, défense et votre argent. C'est un résumé des statistiques des équipements puisque casque et armure confèrent de la défense, et le total de défense est l'addition des deux attributs, idem pour l'attaque qui concerne le couteau et l'arme à feu.

En dessous, il y a d'autres informations diverses comme le total de dégâts infligés, de zombies tués etc. qui se mettent à jour au fur et à mesure que le joueur joue, ce sont vraiment des statistiques purs.

Enfin, il y a la liste des équipements portés et leurs descriptions, ce qui est utile pour vérifier ce que vous portez pour le combat et pourquoi pas, se plonger dans l'univers en lisant les descriptions etc. Sa dernière utilité est aussi de pouvoir se déconnecter grâce au bouton.

En commençant à jouer, c'est la première chose qu'on voit mais en réalité, il y a un peu plus de détail sur les objets. C'est ce que nous verrons dans la boutique.

– Armurerie –

Figure 14: L'armurerie (haut)

Figure 13: L'armurerie (bas)

La conception de cette page, « Shop.vue » a été passionnante, on a pu laisser notre imagination s'emporter. Il est très facile de reconnaitre que nous avons créé les icones des objets, c'était un plaisir de passer du temps à dessiner. Cependant, nous avions un design en tête pour refléter ce côté jeu vidéo et c'est bien le jeu ***Borderlands*** :

Figure 15: Description arme Borderlands

Figure 16: Description arme de notre site

Borderlands étant un jeu quand même plus développé, nous nous sommes inspirés de ce design de présentation : chaque objet possède un niveau de rareté allant de Commun, Rare, Epic et Légendaire tout comme sur ***Borderlands*** (common, uncommon, rare, very rare…) et c'est ce qui fait que la couleur du fond change ! Effectivement, chaque niveau de rareté, comme vous l'aurez remarqué, possède sa couleur de fond. Le gris par exemple, représente bien le côté pas impressionnant de l'objet et nous l'attribuons aux premiers objets de départ, c'est-à-dire Level 1 (le maximum étant le Level 4, tout cela pour bien distinguer facilement la progression du côté équipements).

Si vous êtes en train d'essayer notre site, vous pouvez tricher en modifiant votre argent dans la base de données afin d'avoir accès à l'achat de tous les objets.

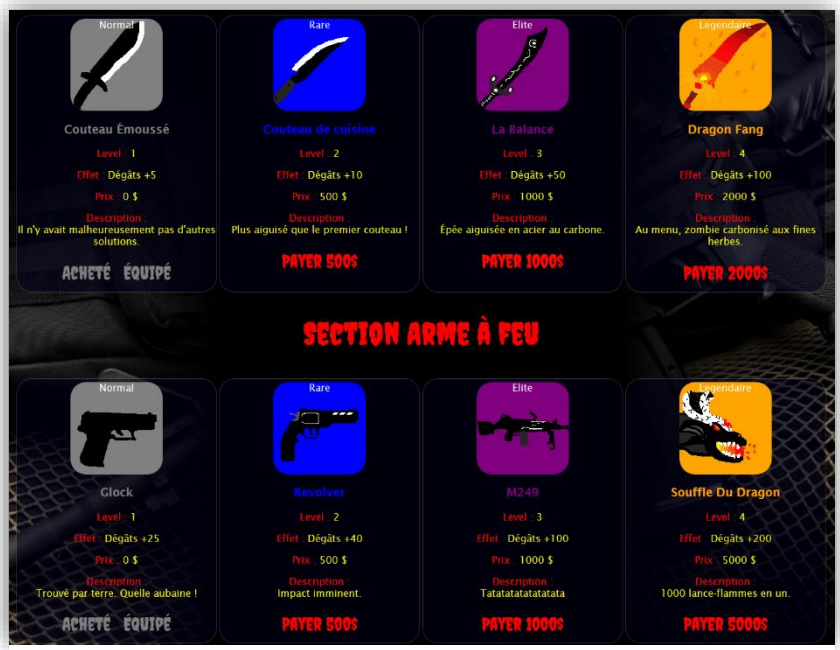

Figure 17: Bouton d'achat disponible

Avec assez d'argent, les possibilités d'achat se débloquent :

Comme vous pouvez le voir, en se donnant une quantité astronomique d'argent, nous pouvons commencer à effectuer nos achats. Pour cela, il suffit tout simplement de cliquer sur un bouton « Payer _ $ ».

Une fois que vous aurez acheté un objet, la base de données l'enregistrera en ajoutant l'*id* de l'objet en question dans votre inventaire, appelé « ***items*** » dans la table *users.*

Voilà l'objet débloqué. Pour implémenter cela, il suffira d'ajouter un vérificateur qui détermine la présence de l'*id* de l'objet dans l'inventaire du joueur et afficher les boutons en fonction de chaque résultat.

Nous avons désormais le choix de l'équiper et pour cela, un bouton a été mis à disposition. En réalité, c'est le seul endroit où l'on peut changer ses équipements. Démonstration :

Figure 18: Ancienne arme à feu équipée

Figure 19: Nouvelle arme à feu équipée

Le bouton pour ***équiper*** se réactive dans les autres objets de la même catégorie lorsqu'on équipe un nouvel objet. Cela se fait tout simplement par la liste d'*id* qui représente les équipements actuellement équipés (la colonne « ***equipped*** » de la table *users*), ce qui nous permet aussi de mettre à jour facilement le profil du joueur. Une fois équipée, les statistiques du joueur changeront directement.

Pour finir, il y a dans la dernière section, des soins pour regagner des points de vie. Pas si cher, mais très important pour la suite, surtout si le joueur ne fait pas attention à ses points de vie. Notons quand même la présence cette fois-ci d'un attribut : Quantité. En effet, le joueur peut acheter autant de soins qu'il désire. Pour implémenter ça, il y a pas mal de possibilités mais on a choisi d'incrémenter le 5^{ième} index de la liste des objets équipés (colonne « ***equipped*** ») qui représente la quantité de pack de soins.

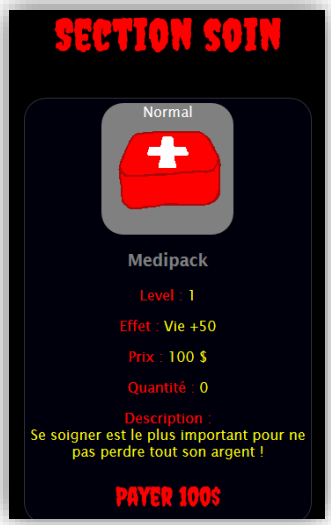

Figure 20: Médipack dans l'armurerie

– Combat –

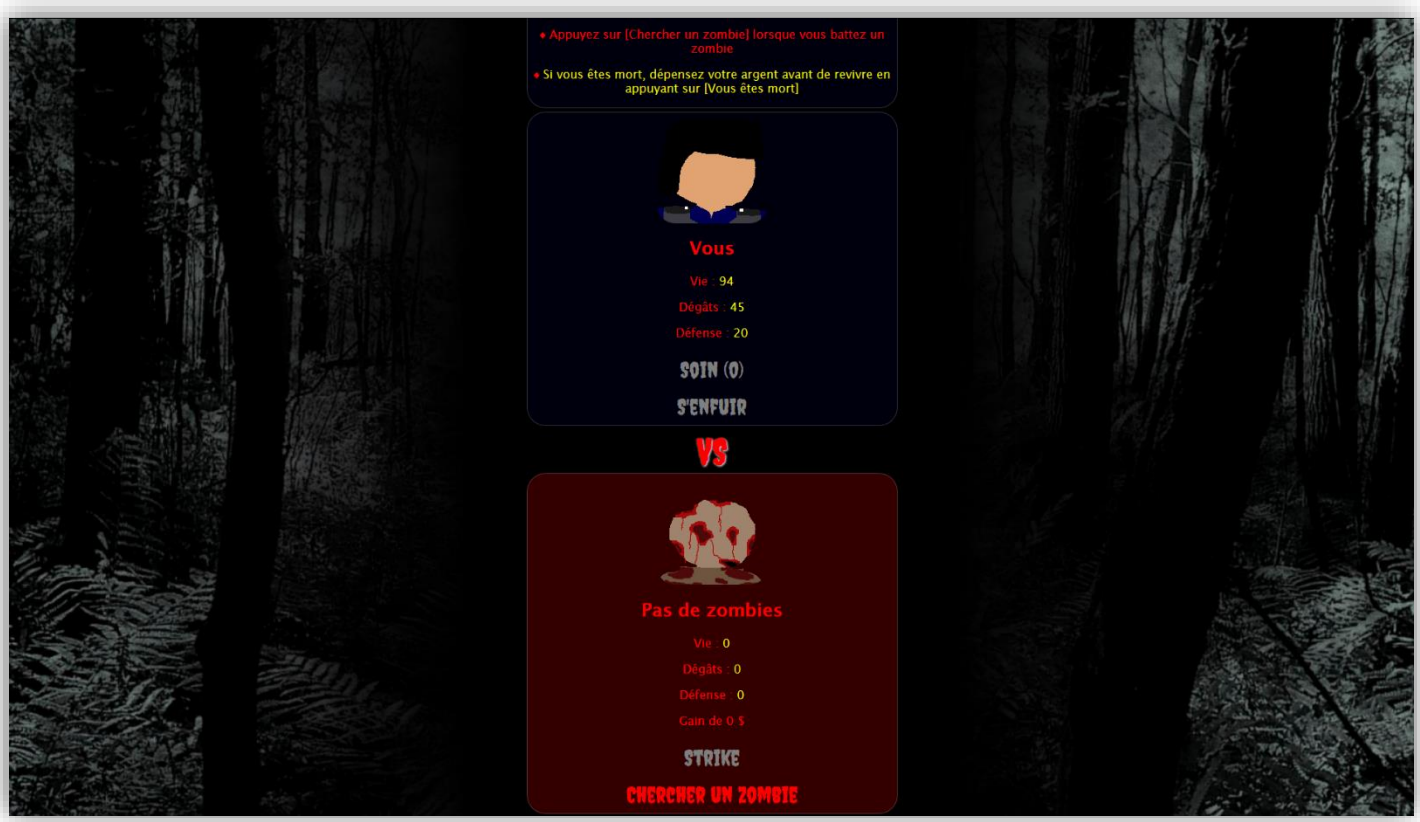

Figure 21: Interface de combat contre le zombie

Enfin, notre dernière page, « **Combat.vue** » est la plus importante car c'est ce qui fait que le site est un jeu et pour réaliser ça, plusieurs éléments vont être nécessaires. Nous allons séparer cela en deux parties : le zombie et le gameplay.

Zombie :

Figure 22: Infos du zombie

Nous allons créer une table de zombies dans la base de données. Le fond de forêt se justifie alors : nous cherchons des zombies sauvages pour les chasser. Pour cela, nous allons effectuer la création d'un zombie.

Tout d'abord, un zombie c'est quoi ? C'est tout simplement un objet possédant des attributs basiques comme vie, défense et attaque. Lors de son initialisation, le zombie possédera le même *id* que le joueur car il est unique et c'est d'ailleurs comme ça qu'on le retrouvera dans la table. Ce n'est pas tout, le zombie doit aussi contenir une récompense en guise de l'avoir vaincu : c'est le gain d'argent afin de permettre au joueur d'acheter de meilleurs équipements.

Ensuite, il y a plusieurs types de zombies : faible, commun, brutal et le boss. Ce sont en fait des difficultés qui vont influencer leurs propriétés comme les dégâts. Le type de zombies fait varier les gains (d'ailleurs, les gains sont aléatoires mais donnent au moins le nombre de vie qu'a le zombie. Par exemple, un zombie à 100 PV donne obligatoirement au moins 100$. Le type de zombie est toujours affiché comme son nom pour facilement repérer la difficulté du zombie.

Gameplay :

Les règles sont sur le site, explicitement expliquées.

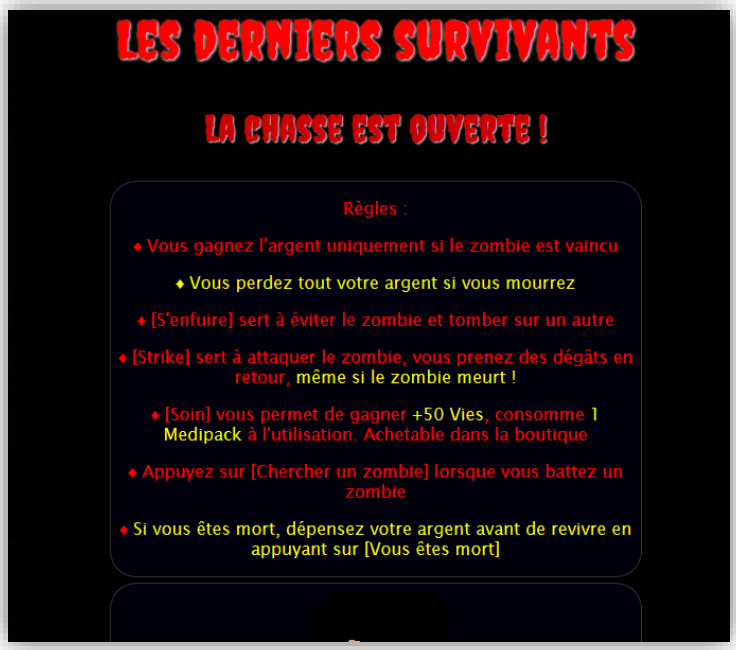

Figure 23: Les règles du jeu

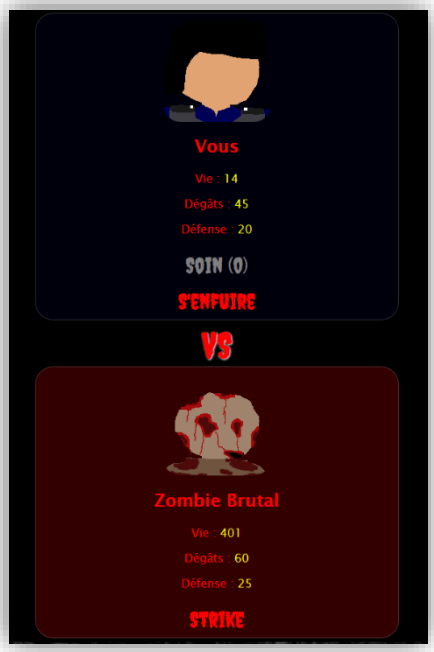

Figure 24: Interface avant combat

Figure 25: Après Strike

On avait donc dit précédemment que le joueur possède des attributs et que le zombie possède des attributs similaires. Pour réaliser le système de combat, nous allons implémenter plusieurs boutons afin de pouvoir combattre le zombie.

En tout premier, le bouton [Strike]. Ce qu'il fait, c'est échanger un coup contre le zombie. Vous perdez tous des points de vie en fonction de vos dégâts calculés incluant la défense de l'ennemi.

Comme on peut le voir, en attaquant le zombie, nous avons effectué 40 dégâts à cause des 5 défenses de l'ennemi. Quant à nous, c'est 3 points de vie qu'on perd car l'ennemi inflige 23 dégâts et nous avons 20 de défense. Ainsi, nous avons pris soin de mettre en rouge l'indice de dégâts subis.

Lorsqu'un zombie est abatu, ce dernier va lacher un butin d'argent.

Figure 26: Zombie mort, victoire et gain d'argent

En mourant, le zombie a lâché 64$. Le montant ne s'affiche que lorsque le zombie est vaincu. Il ne reste après qu'une chose à faire : [**Chercher un zombie**] qui respecte exactement son nom, va recréer un zombie dans la table car l'ancien, qui n'a plus de PV, est supprimé de la table.

Cependant, il arrivera que le joueur n'arrive pas à finir ou commencer à tuer un zombie sans mourir. Dans ce cas-là, nous avons mis à disposition un autre bouton qui n'est appuyable que lorsque le zombie n'est pas mort : [**S'enfuir**] qui permet de relancer les dés pour tomber sur un autre type de zombies aléatoire. Cette fois-ci, on ne supprime pas le zombie dans la table mais on le modifie.

Maintenant, voyons que faire lorsque les PV sont bas. Il n'y a qu'un seul remède et c'est le médipack qui restore 50 PV. Pour l'utiliser, il faut déjà en acheter. Dans le cas où on en a déjà un, le bouton devient disponible :

Figure 28: Avant soin

Figure 27: Après soin

L'utilisation est bien sûr immédiate, ainsi que l'effet. Le maximum de santé est 100 PV (ce qui veut dire que toute utilisation au-dessus de 50 PV entrainerait des pertes). Le médipack n'est pas donc pas utilisable (bouton grisé) si l'utilisateur est déjà à 100 PV. Il est très important de toujours regarder sa vie et ses packs de soin avant d'initier quelconques zombies.

Dans le cas où le joueur est voué à mourir, un bouton apparaîtra :

Figure 29: Mort, défaite du joueur

C'est vraiment le bouton que le joueur ne veut absolument pas voir. C'est la fin de la partie. En appuyant dessus, son argent est réinitialisé et son compteur de mort incrémente d'un. Par sympathie, il ne perdra pas tous ses objets (ce n'est pas une réinitialisation totale), tout regagner serait en effet très énervant.

D'ailleurs, on peut noter que l'utilisation du médipack ne permet pas de revivre. Son utilisation n'est plus possible une fois mort, ce qui est logique.

Conclusion

Finalement, ce projet nous aura coûté pas mal de temps, mais un temps sacrément utile pour bien comprendre une partie de la programmation Web, avec l'utilisation de Vue.js et de son interaction avec les serveurs comme Node.js ainsi que les bases de données en PostgreSQL. Cela nous permet de réaliser qu'il est bien plus simple de réaliser un site avec tous ces outils que traiter plusieurs pages html individuellement comme on a pu le faire l'an passé, sans oublier que le serveur et la base de données élargissent grandement le champ des possibilités au niveau du sujet, ce qui nous a permis donc d'effectuer ce projet.