

RAPPORT DE PROJET



Réalisé par :

Maxence BOISEDU

Simon VENNAT

Kévin AVOT

Axel BLEUSE

Kévin KERKHOVE

SOMMAIRE

Partie 1 : Introduction	p.2
Partie 2 : Développement préliminaire	
2.1 : Démarche adoptée	p.3
2.2 : Choix opérés	p.4
2.3 : Éléments mis en œuvre par rapport à la demande initiale	p.5
2.4 : Répartition des tâches	p.6
2.5 : Difficultés rencontrées et comment elles ont été réglées	p.7
Partie 3 : Rapport technique	
3.1 : Détail des fonctionnalités	p.9
3.2 : Descriptif de la démarche adoptée	p.10
3.3 : Diagramme de cas d'utilisations	p.11
3.4 : Description des méthodes les plus utiles	p.12
3.5 : Description des données (et structures de données)	p.14
3.6 : Validation des programmes (tests etc...)	p.15
Partie 4 : Conclusion	
4.1 : Bilans personnels	p.16
4.2 : Bilan Global	p.18
Partie 5 : Annexes	
5.1 : Liste des abréviations du rapport	p.19
5.2 : Charte graphique et moodboard	p.20
5.3 : Planning prévisionnel	p.21
5.4 : Bibliographie	p.22
5.5 : Autres annexes	p.23

LENS

Partie 1 : Introduction

Notre jeu (“Chronicles of Naruberaria”) est un side-scrolling shooter se voulant un gameplay dynamique et qui met les capacités du joueur au défi. Notre intention autour de ce jeu, est de retrouver une ambiance de jeu d’arcade tout en proposant une expérience innovante et captivante.

Ce jeu est alors programmé à l'aide du framework Angular (pour Javascript), du framework Laravel ainsi que de Phaser, une librairie permettant la création même du jeu.

L'objectif du jeu est alors de vaincre tous les ennemis afin de sauver le monde d'une destruction irréfutable. Dans le cas contraire, seules la mort et la destruction feront place sur terre. Afin de lutter contre cela, notre héros devra voyager à travers les univers pour lutter contre les envahisseurs.

Partie 2 : Développement préliminaire

Avant de commencer à programmer, nous avons réfléchi à la ligne directrice que l'on souhaitait suivre pour la réalisation de notre jeu. Nous avons d'abord choisi le type de jeu que nous voulions réaliser, puis nous avons réfléchi à un univers (un scénario, une époque) dans lequel nous aventurer.

2.1 : Démarche adoptée

Nous avons d'abord commencé par réaliser un Brainstorming afin de trouver quel type de jeu nous allions réaliser, chacun a alors donné des idées ou exemples de jeux, puis chacun a fait son choix parmi la liste des jeux proposés.

Concernant le titre, chacun a fait fonctionner son imagination, car en effet, il n'est pas toujours facile de trouver un titre à sa création. Le choix s'est porté sur "Chronicle of Naruberaria".

Ce titre est inspiré de l'animation Japonaise, il fait référence à un personnage du manga "*Overlord*" et précisément au personnage nommé "*Narberal*". Comme nous apprécions tous cette culture, le choix de ce nom semblait tout simplement normal.

LENS

2.2 : Choix Opérés

Nous avons recherché un jeu différent des standards qui nous avaient été présentés, c'est à dire que nous ne voulions pas coder des jeux "type Mario". Le plateformer classique ne nous donnait vraiment pas envie et ne nous plaisait pas vraiment. Nous avons alors choisi de partir sur un jeu ressemblant au légendaire jeu "*Space Invaders*".

"*Space Invaders*" est un **shoot them up fixe** dans lequel nous incarnons un vaisseau spatial devant se débarrasser des vagues d'aliens voulant détruire la terre. La spécificité de ce jeu est que le personnage est fixe, en effet son seul déplacement se faisait de manière horizontale.

Notre jeu veut alors garder le côté classique du jeu sur le concept, mais intégrer de nouvelles fonctionnalités afin que le jeu ne soit plus fixe notamment. De plus, nous avons choisi d'utiliser un décor rétro-futuriste (Cyber-Punk) pour notre premier niveau, car c'est un thème que chaque personne de notre groupe apprécie. Puis dans les niveaux suivants nous avions la volonté de changer les décors et les ambiances afin de rendre le jeu plus dynamique et plus attrayant pour les joueurs. Un ajout de taille a également été réalisé, car en effet nous avons ajouté des obstacles correspondants aux différents thèmes de nos niveaux.

Pour donner un aperçu des diverses zones que nous pouvons retrouver sur le jeu, nous pouvons citer les thèmes Cyber-Punk, Steam-Punk, Apocalyptique et Aquatique.

LENS

2.3 : Éléments mis en œuvre par rapport à la demande initiale

Pour contribuer au bon fonctionnement du travail en groupe, nous avons utilisé quelques outils collaboratifs. Leur mise en place fût rapide et nous a grandement été utile.

Nous avons premièrement créé un serveur **Discord** dédié au projet, dans lequel les salons textuels étaient dédiés aux différentes tâches. Discord nous a également été fort utile pour la communication vocale ainsi que pour sa fonction “*Partage d’écran*” nous permettant de nous déboguer l’un l’autre sans que cela devienne difficile à gérer.

Nous avons également créé un groupe **Messenger** afin de faciliter la communication textuelle entre tous les membres du groupe. Il était alors principalement utilisé pour donner les informations principales concernant le projet (Avancement de chacun, dates importantes, Indication sur la réception des mails etc...).

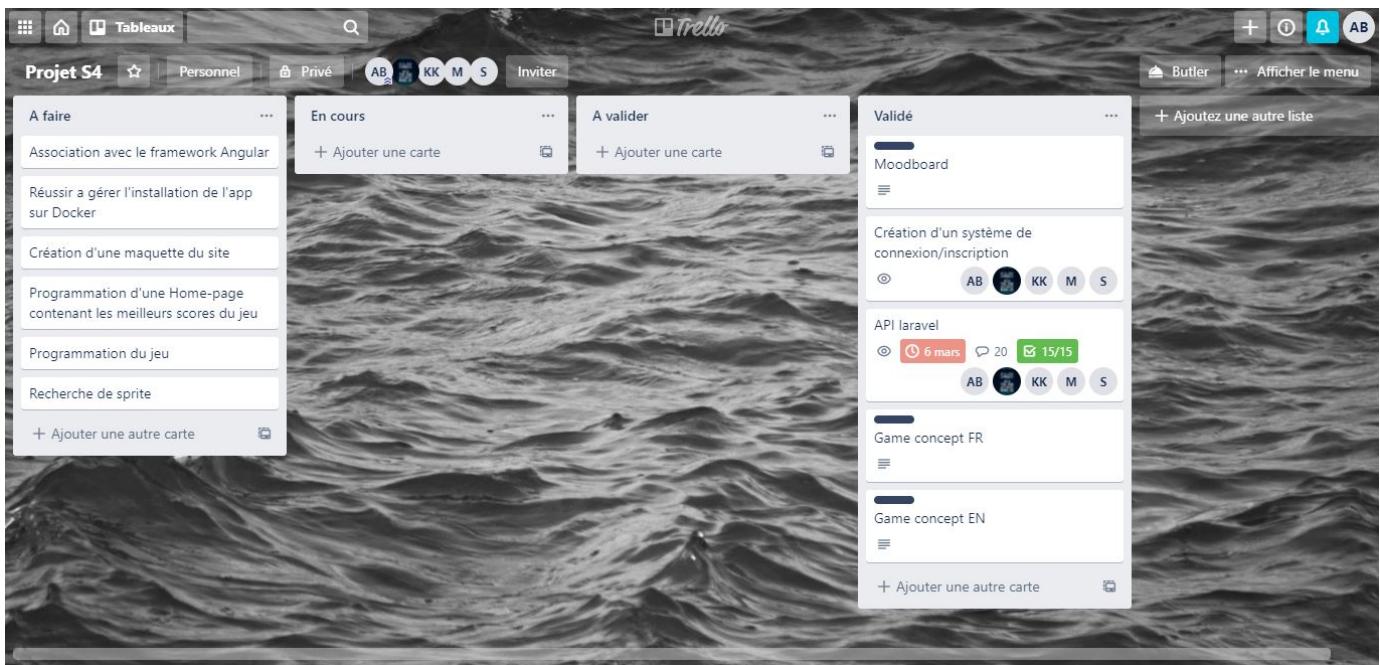
Nous avons utilisé **GitLab** pour faciliter le partage du code lorsqu'il était fonctionnel et également pour gérer le versionnage du projet.

Pour finir, la répartition des tâches s'est effectuée (au début seulement) via un board **Trello**. Nous l'avions utilisé lors de la conception de l'API Laravel, puis par la suite chacun s'est attribué une tâche de façon automatique (naturellement).

LENS

2.4 : Répartition des tâches

Comme dit précédemment, la répartition des tâches s'est faite premièrement via un board **Trello**.



Par la suite, la répartition s'est faite naturellement. En effet, Nous avions déjà travaillé ensemble par le passé (Marathon du WEB, Projets tuteurés des semestres précédents). Nous connaissions alors bien les points forts et faibles de chacun.

Chacun a alors pu se trouver une tâche propre, dont il est dur de voir un résultat correct sans que quelqu'un d'autre ait à "repasser derrière" pour la correction des bugs.

LENS

2.5 : Difficultés rencontrées et comment elles ont été réglées

Au niveau de l'organisation, aucun problème n'a été recensé, de même concernant les relations entre les membres du groupe. Les problèmes que nous avons eus furent des problèmes de code qui ont tous été résolus.

Nous avons eu quelques gros problèmes avec les API :

Premièrement nous avons eu un premier problème lors de la liaison entre Angular et Laravel, malgré nos formulaires de connexion et d'enregistrement, aucune action ne se réalisait (Indiquant alors "Unknown error"). Nous avons résolu ce problème en changeant la valeur de la variable "apiUrl" au sein du fichier *environnement.ts*.

Secondement, nous avions un problème avec notre formulaire d'enregistrement d'un nouvel utilisateur. En effet, la validation de celui-ci nous redirigeait directement vers la page '/game' et donc n'enregistrait pas les informations indiquées. Pour corriger cela, Nous n'avons eu qu'à corriger le *<input>* du bouton qui contenait un "*routerLink="/game"*" qui n'envoyait alors pas les données au serveur.

LENS

Pour finir, nous avions un problème pour l'édition du profil. D'une part, les utilisateurs n'avaient pas accès à cette page ("scope error"). D'une autre part, il était impossible de valider le changement de données liées à l'utilisateur. Pour résoudre cela, nous avons alors dû, d'une part, modifier les rôles contenus dans les routes du fichier `api.php` (de l'api Laravel). Et concernant la validation de l'édition du profil nous n'avons eu qu'à modifier le type de la variable qui posait problème (de String vers Int).

Nous avons eu un seul problème majeur concernant le jeu :

En effet, lors de la programmation du jeu, nous avons eu un problème concernant les ennemis. Lorsque notre personnage tirait sur les ennemis, ceux-ci disparaissaient, mais leurs entités étaient encore présentes, ce qui fait qu'ils n'étaient "*pas vraiment morts*". Nous avons pu résoudre ce problème en reprogrammant les ennemis, devenant alors des groupes de sprites. Ceux-ci sont plus simples à retirer du jeu, et limitent donc le risque de problèmes.

LENS

Partie 3 : Rapport technique

3.1 : Détail des fonctionnalités

Le site et le jeu créés ont de nombreuses fonctionnalités. Lors de notre arrivée sur le site, nous avons alors vue sur notre moodboard et sur une description du jeu contenu dans le site.

A partir de cette même page, il est alors possible de se connecter ou de s'enregistrer via des formulaires (On soulignera que nous ne nécessitons aucune actualisation de la page pour accéder à ces formulaires).

Suite à la connexion sur le site, il est alors possible d'accéder à la page « Mon profil », à partir de laquelle nous pouvons voir de nombreuses informations, telles que les informations personnelles, et d'autres informations telles que l'image de profil ou le temps de jeu notamment. Aussi, dans la barre de navigation, le bouton « Joueurs » apparaît. Ce bouton permet d'afficher la liste des joueurs enregistrés. Il est possible de retourner à la page d'accueil à tout moment.

Il est également possible d'accéder au jeu lorsque l'on est connecté. Nous arrivons alors sur la page de menu du jeu. Lorsque l'on clique sur « Jouer », le jeu commence, et tant que le joueur ne meurt pas, il évolue au sein des quatre niveaux. Il est alors affiché le score actuel, 500 points par ennemi tué et 1500 par niveau gagné. Lorsque l'on atteint la fin du niveau, un portail apparaît. Si nous sommes encore vivants à la fin, nous avons un écran de victoire proposant de passer au niveau suivant, dans l'autre cas, si nous mourons, il y a une animation (caméra secouée) et le niveau

recommence. La mort du joueur est causée par une collision avec un adversaire.

3.2 : Descriptif de la démarche adoptée

La démarche que nous avons adoptée a toujours été la même tout le long du projet. En effet, dès lors qu'un travail était rendu, nous faisions la même répartition des tâches pour le travail suivant.

Lors de la programmation de chaque API (et du jeu) dès lors qu'une fonction était programmée, elle était testée afin de pouvoir corriger au plus vite (et au plus tôt) les problèmes qu'elle aurait pu causer.

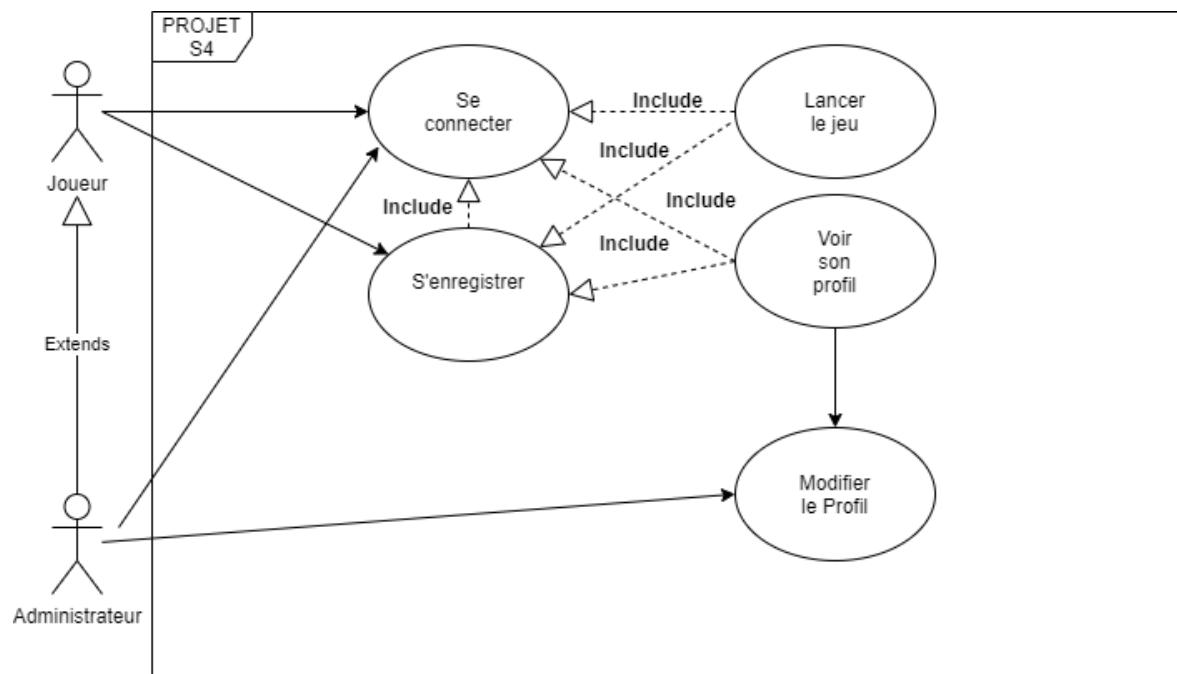
Lors de l'association entre Laravel et Angular, dès lors que nous réalisions un lien pour n'importe quel élément, nous lancions la page afin de savoir si tout fonctionnait comme il fallait. (Lors de cette liaison, si un problème survient, il peut être dû à Angular comme à Laravel et peut être fastidieux à corriger).

Pour le fonctionnement du jeu au sein de l'API Angular, nous n'avions qu'à tester en jouant au jeu. D'autres tests ont par la suite été réalisés. Ils seront tous détaillés dans la partie 3.7 de ce rapport.

LENS

3.3 : Diagramme de cas d'utilisation

Voici le diagramme de cas d'utilisation associé à notre projet :



3.4 : Description des méthodes les plus importantes

En ce qui concerne le **client Angular**, plusieurs méthodes permettent le bon fonctionnement de l'application, notamment celles permettant de s'enregistrer et de se connecter ou encore de modifier un joueur.

La méthode *onSubmit* de la classe *RegisterComponent* permet d'envoyer les informations entrées dans le formulaire d'enregistrement par le nouvel utilisateur au serveur. Ainsi, l'utilisateur est ajouté à la base de données et peut donc se connecter.

```
onSubmit() {
    this.player.name = this.name.value;
    this.player.user.email = this.email.value;
    this.player.user.name = `${this.player.name}`;
    const pwd = this.password.value;
    this.authService.onRegister( valeur: {player: this.player, pwd: this.password.value})
        .subscribe( next: data => {
            this.router.navigate( commands: ['/']);
        },
        error: error => {
            console.log('erreur en retour : ', error);
            this.error = error;
            this.loading = false;
        });
}
```

Implémentation de la méthode *onSubmit* dans la classe *RegisterComponent*

LENS

La méthode *updatePlayer* (se trouvant dans la classe *PlayersService*) permet quant à elle d'envoyer au serveur les nouvelles informations entrées par l'utilisateur dans le formulaire.

```
updatePlayer(player: Player, file: FileInput, pwd: string): Observable<Player> {
    const url = `${this.playerUrl}/${player.id}`;
    const formData: FormData = new FormData();
    formData.append('name', player.name);
    formData.append('bio', player.bio);
    formData.append('bestScore', player.bestScore.toString());
    formData.append('email', player.user.email);
    if (pwd) {
        formData.append('password', pwd);
    }
    formData.append('_method', 'PUT');
    if (file) {
        console.log(`fichier avatar : ${file.fileNames}`);
        formData.append('avatar', file.files[0], file.fileNames);
    }
    return this.http.post<Observable<Player>>(url, formData)
        .pipe(
            tap(next: (rep: any) => console.log(rep)),
            map(project: p => Player.parse(p.data)),
        );
}
```

Implémentation de la méthode *updatePlayer* dans la classe *PlayersService*

Pour la partie **Phaser** du projet, la fonctionnalité la plus importante est la création de scènes Phaser personnalisées.

Cela est géré avec une classe *GameCreator* contenant 13 méthodes statiques. En fonction des paramètres que l'on donnera à ces méthodes, nous obtiendrons une scène Phaser différente.

Cette façon de créer des scènes est très utile pour faire à la fois des niveaux qui se ressemblent mais qui diffèrent sur certains points.

A chaque fois que l'on veut faire un nouveau niveau, il suffit d'appeler les méthodes adéquates dans la classe *GameCreator* en mettant la scène du niveau en paramètre.

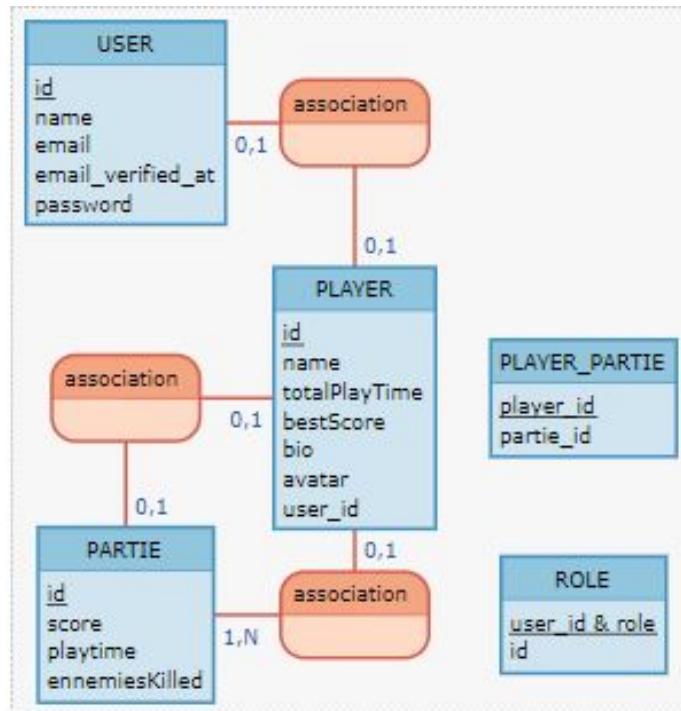
```
create() {
    GameCreator.globalScore = 0;
    GameCreator.create(this, 'map1');
    GameCreator.createEnemies(this, 'enemy1');
    GameCreator.generateObstacle(this, 'obstacle1.1', 'obstacle1.2');
}
```

Exemple d'utilisation de la classe *GameCreator*

LENS

3.5 : Description des données (et structures de données)

Nous utilisons une base de données afin de stocker toutes nos données. Celle-ci est composée de 5 tables dont 1 intermédiaire. Elles se nomment *User*, *Partie*, *Player*, *Roles* et *Player_Partie*. Elles permettent la création, modification et suppression d'un joueur contenant les droits de la table Rôles.



LENS

3.6 : Validation des programmes (Test, etc...)

Les programmes que nous avons réalisés pendant ce projet ont chacun été testés, dans certains cas, seule une ligne de commande suffisait et dans d'autres, il a fallu réellement lancer l'application afin de tester le bon fonctionnement.

Dans le cas de l'API Laravel, les tests ont été faits premièrement par exécution des commandes de migration et de seed. Si ces deux tests étaient concluants, nous réalisions un affichage des valeurs dans une page, afin de bien voir s'il n'y avait pas de problème.

Dans le cas de l'API Angular et de la connexion Angular-Laravel, les tests ont été faits via le logiciel "Insomnia" ainsi que par exécution de l'API Laravel. La vérification des bons résultats se faisait alors par des affichages sur la page ou via la console.

Pour le jeu, il a simplement fallu tester le jeu, et vérifier le bon fonctionnement de la méthode implémentée.

Partie 4 : Conclusions

4.1 : Bilans personnels

Axel :

“Ce projet a encore une fois été très instructif, il nous a permis d’apprendre de nombreuses notions utilisées en WEB (telles que la création d’une API Rest par exemple) et nous a permis de réaliser un projet WEB concret et complet dans l’ambiance du travail d’équipe.”

Simon :

“Ce projet m’a permis de développer mes compétences, tant en développement web qu’en la maîtrise des outils collaboratifs tels que GitLab. De plus, cela m’a à nouveau permis de travailler en équipe, ce qui est, à mon sens, essentiel dans le développement d’un tel projet.

Le fait de travailler sur un cas concret comme celui-ci m’a permis de lier les compétences acquises au cours des semestres 3 et 4 ainsi que d’appliquer les notions vues au cours des TPs de cette année, plus particulièrement en PHP avec Laravel et en JavaScript avec Angular.”

Maxence :

“Durant ce projet, j’ai pu constater que développer un simple site pour jouer à un jeu n’est pas aussi simple que je le pensais. Surtout le lien entre Angular et Laravel. Ce fut un projet instructif grâce auquel j’ai découvert Phaser, un outil qui m’a fortement intéressé. De plus, ce dernier projet en “grand” groupe nous a permis d’améliorer nos compétences de travail d’équipe notamment avec les outils collaboratifs.”

LENS

Kévin A. :

“Ce projet nous a permis de grandement agrandir nos connaissances en Angular et en débogage, on a eu également l’occasion de découvrir Phaser un outil intéressant et pratique. On a aussi pu appliquer nos connaissances à des cas plus concrets que ceux vu en TP.”

Kévin K. :

“Ce projet S4 est un projet qui a une fois de plus été très instructif. Il nous a permis d’appliquer les connaissances acquises en TP de PWEB, de développer nos connaissances en développement web en PHP avec Laravel et en JavaScript avec Angular puis de découvrir Phaser. Grâce à ce projet on a eu l’occasion de travailler en équipe une fois de plus et de nous familiariser encore plus avec les outils collaboratifs. Puis pour ma part j’ai pu développer mes compétences en graphisme, notamment sur Photoshop.”

LENS

4.2 : Bilan Global

De manière globale, ce projet est un projet intéressant, qui nous a nécessité beaucoup de travail et d'investissement personnel.

Ce projet est notre premier “vrai gros projet” WEB nécessitant toutes nos compétences acquises lors de cette deuxième année. De plus, nous avons pu tous acquérir de nouvelles compétences dans la programmation Laravel avec la production des API REST et en Angular avec Phaser ou encore la connexion entre Angular et Laravel.

Pour conclure, c'est un projet qui a beaucoup plu, notamment par la conception d'un jeu avec Phaser. Et nous sommes fiers du résultat auquel nous avons abouti après les évènements exceptionnels auxquels nous faisons face.

Partie 5 : Annexes

5.1 : Liste des abréviations du rapport

API : Application Programming Interface = Interface de programmation d'applications

REST : Representational state transfer = Transfert d'état représentatif

WEB : World Wide Web = Internet

Migration : Création des tables contenues dans la base de données

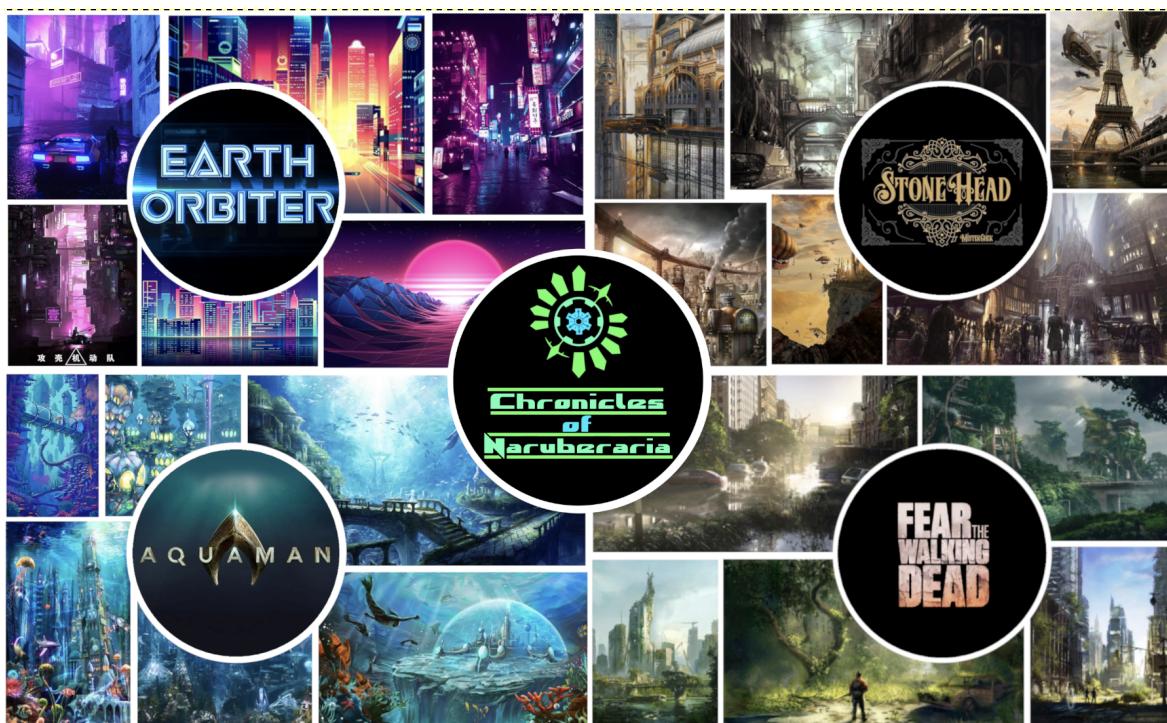
Seed : implémentation des bases de données au code.

LENS

5.2 : Charte graphique et moodboard

Notre jeu représente le voyage à travers les univers, de ce point de vue, on peut alors penser à l'espace et à ses couleurs sobres. Nous avons alors utilisé des couleurs sobres faisant référence à l'espace, des teintes violettes et des néons pour garder un côté rétro.

Le moodboard que nous avons mis en place permet d'apercevoir les différents univers dans lesquels notre héros se battra.



LENS

5.3 : Planning prévisionnel :

Le rendu de l'API Laravel devait se faire le 6 mars, nous avions pu rendre le travail fait, mais pas totalement terminé, un bug faisait qu'il n'était pas possible d'utiliser l'API comme tel. Nous avons alors rendu le travail le 5 mars mais nous avons finalisé sa réalisation dans les semaines qui ont suivi.

Le Rendu de l'API Angular ainsi que du jeu programmé à l'aide de Phaser devait être rendu le 30 Mars, nous avons pu rendre la totalité du travail fonctionnel.

La présentation pour la soutenance était, quant à elle prête dès le 30 Mars (soit 3 jours en avance).

Le rapport de projet devait être rendu le 31 Mars et a été rendu à la bonne date, complet.

Les timings ont alors été généralement respectés. Bien qu'au début le travail sous Laravel fut difficile et que le retard ait été accumulé, nous avons tout de même pu rattraper ce retard et programmer un site et un jeu complet au final.

LENS

5.4 : Bibliographie

Utilisation de la documentation de Phaser pour la réalisation du jeu :

<https://photonstorm.github.io/phaser3-docs/>

Utilisation de la documentation Laravel concernant les instructions http :

<https://angular.io/guide/http>

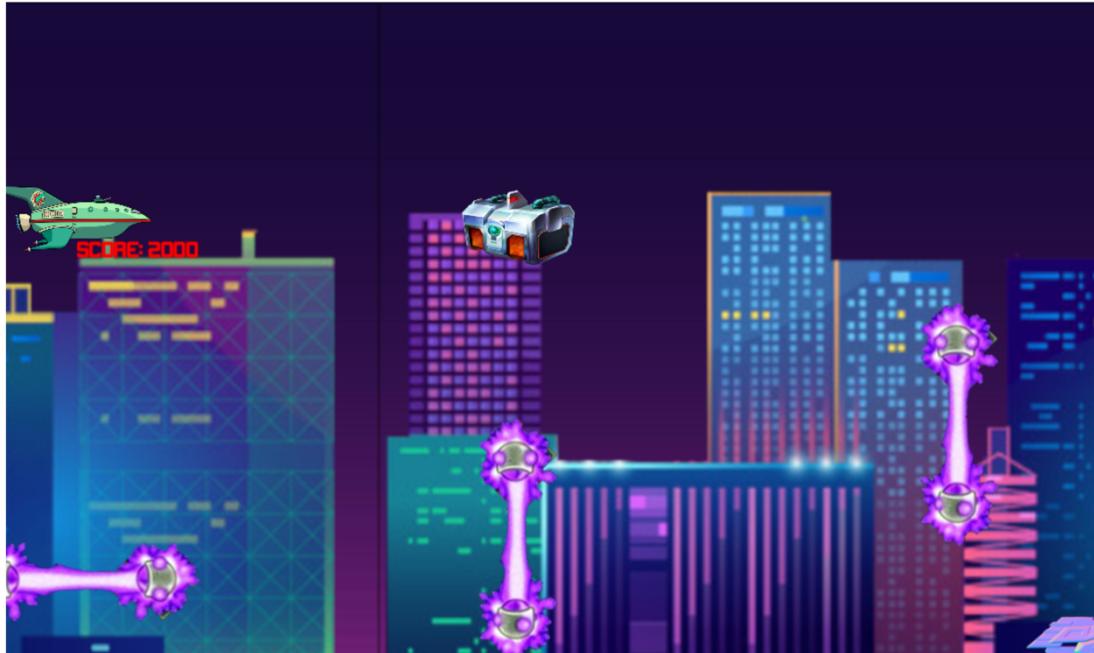
Utilisation d'un site apprenant la création d'une API REST avec Laravel :

<https://www.toptal.com/laravel/restful-laravel-api-tutorial>

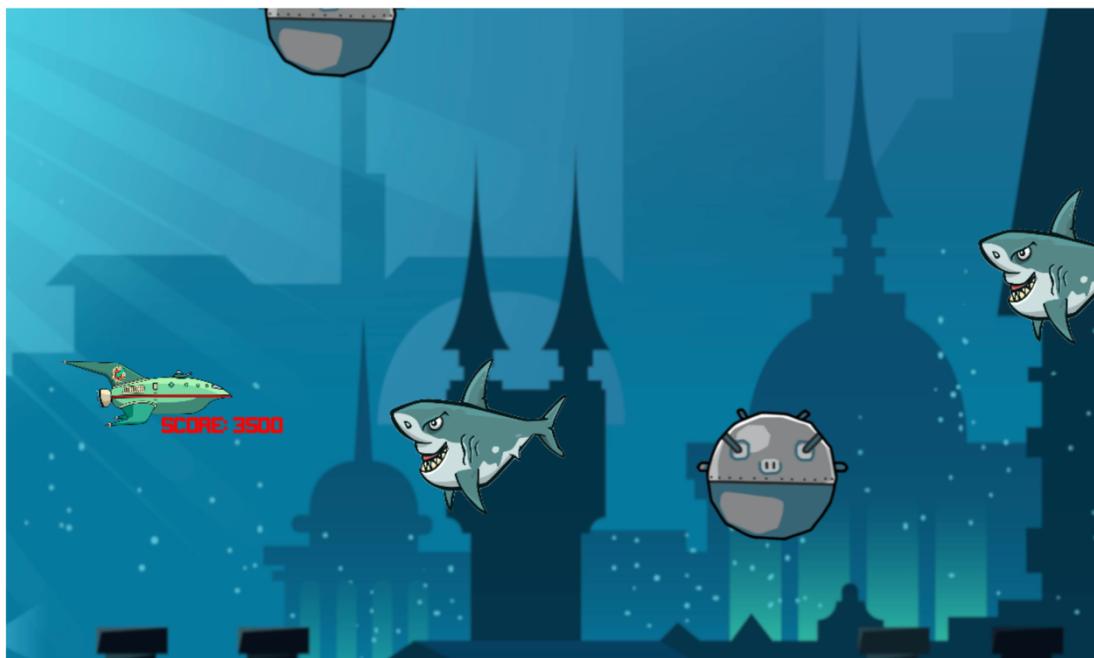
LENS

5.5 : Autre Annexes

Captures d'écran :

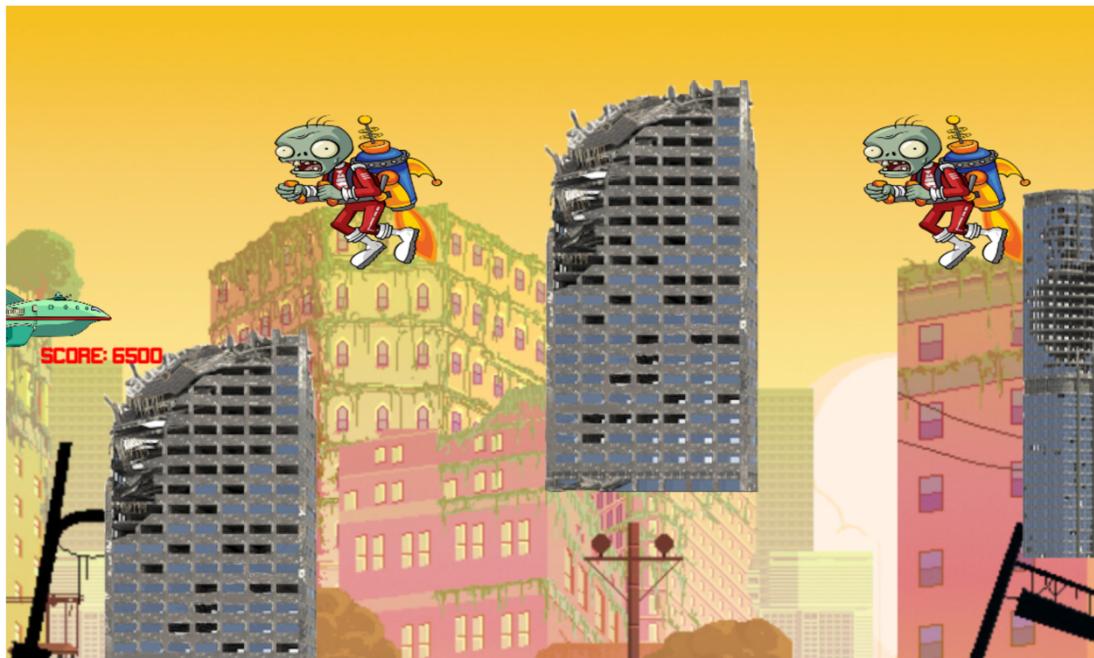


Niveau 1 « Cyber-Punk » et un bonus

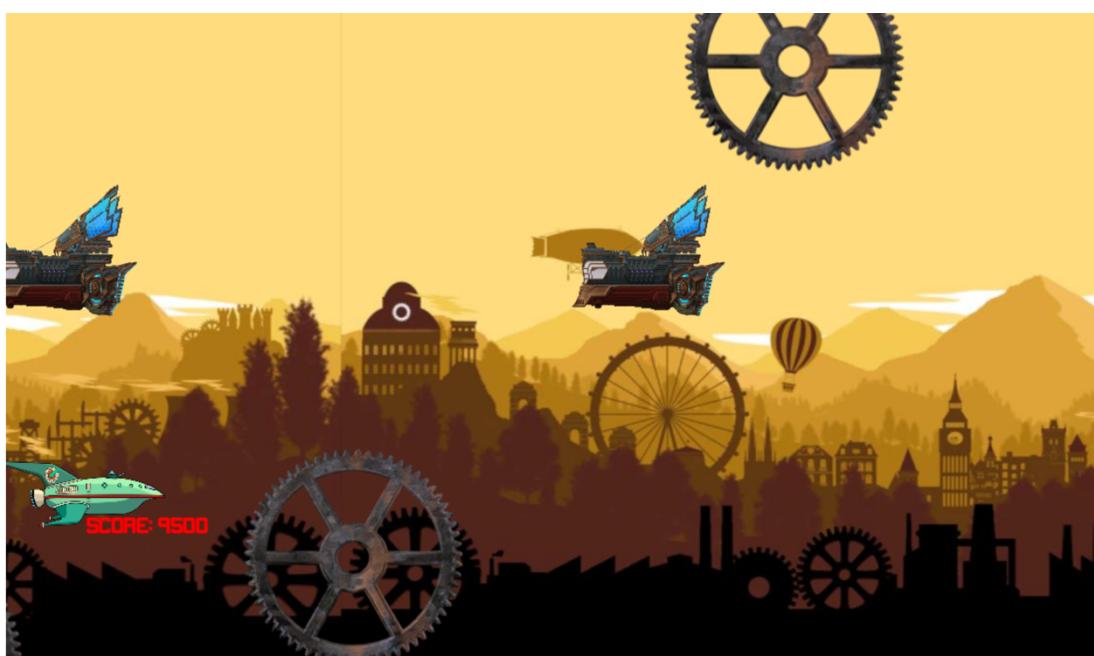


Niveau 2 « Aquatique »

LENS



Niveau 3 « Apocalyptique »



Niveau 4 « Steam-Punk »

Gestion des joueurs		
No.Nom	Présentation	Temps de jeu Meilleur score
1 Miss Jewell Botsford	Dolor praesentium rerum qui quia. Veniam voluptatem qui sapiente magnam. Omnis est deleniti repellat placeat aspernatur id.	22:18:43 583
2 Miss Chasity Pollich	Aut ut labore ducimus est ducimus reiciendis fugiat ipsum. Et perferendis quisquam cum dolorum alias laborum. Fuga nostrum minus assumenda ut quis aut ut cupiditate.	10:57:17 9
3 Haven Grady	Facere quis debitis et reprehenderit iste harum. Et alias nesciunt alias veniam consequatur. Voluptatem modi ut ullam soluta ullam.	01:25:44 6046695
4 Norberto Grant	Hic labore consequatur voluptatem eum. Qui aperiam eligendi corporis repellat sed optio. Reprehenderit vero quidem repudiandae suscipit voluptatem ut.	19:02:38 3405
5 Jessika Feeney	Corporis vel nobis molestiae cupiditate eum distinctio. Cumque debitis a ut sint officia sapiente qui. Temporibus amet ut sit officiis ut.	02:26:49 180250
6 Meaghan Luehwitz	Dolores aspernatur quam natus tempore. Nam assumenda quia ut rerum consequatur tempore. Vero repellat esse quibusdam reprehenderit fugiat.	00:17:15 23415
7 Miss Citalli Becker II	Ipsam officia saepe aut eligendi sunt quo. Qui inventore eaque temporibus aliquid nobis. Quo ex voluptatem itaque ad voluptatem quia autem est.	13:13:18 966345
8 Kaylah Gorczany	Ullam in dolore illo voluptas eaque. Et error quibusdam corrupti excepturi sit nostrum excepturi corporis.	12:22:20 584916391
9 Shayne Bergnaum	Illum omnis dolorum aut omnis dolorem suscipit architecto. Qui earum aut voluptatem. Mollitia ab nisi ut. Qui quia odit voluptatum ex error. Animi explicabo eum omnis quidem esse esse at.	12:19:18 1056418
10 Rosalinda Gutmann	In voluptatem repellat dolor quos. Ut quia tempora sed fugiat. Et cum magnam veritatis excepturi sint iure corrupti illo. Totam unde facilis corporis ipsam. Nihil commodi officia necessitatibus.	09:18:37 490770788
11 Duchbbmol HBJHJ	Bonjour, j'aime jouer aux jeux-videos player	06:53:37 89778
12 Smegol	Bonjour, j'adore jouer aux jeux-videos admin	16:47:51 8278
13 Martin Pecheur	Bonjour, j'aime jouer aux jeux-videos auteur	29:12:32 5675
16 AZERTYUIO		

Liste des joueurs enregistrés avec statistiques (modifiables par un administrateur)

LENS

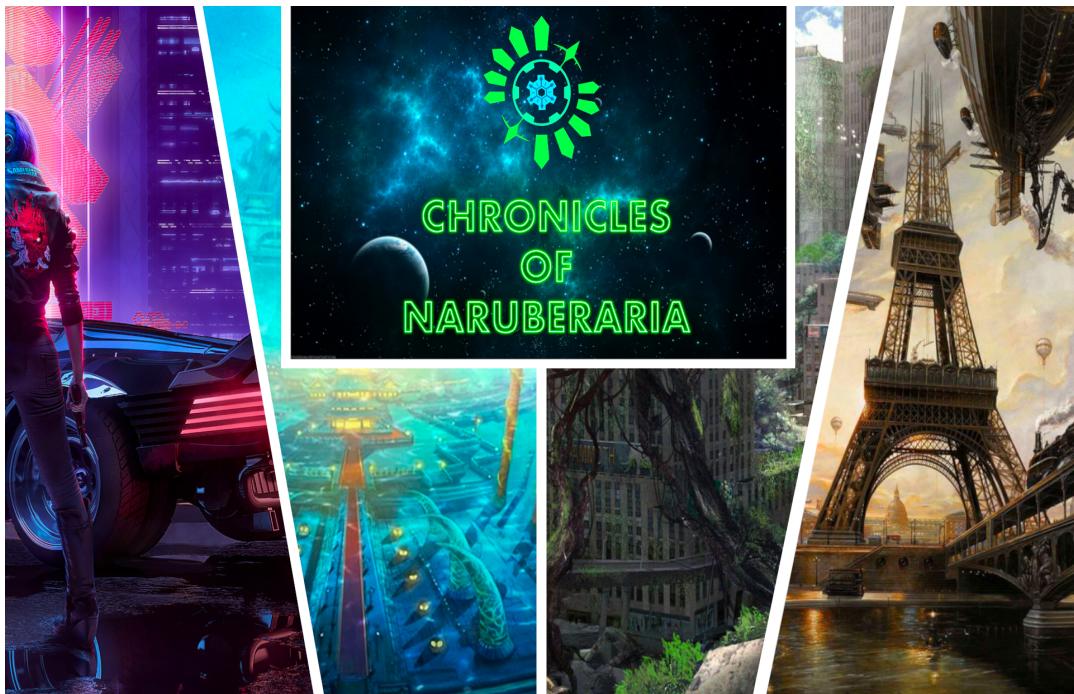
Création de quelques éléments graphiques :

- Pages de victoire adaptées au niveau :



LENS

- Création d'une page d'accueil pour le jeu :



- Création d'une page 404 pour le site :

