Rapport S6

Pas besoin de mettre les noms sur qui a rédigé quoi

# Introduction

Expliquer la SAE S6

# Rappel du sujet

## Objectif

Pour la SAE5, notre objectif est la réalisation d’un jeu de plateforme qui adoptera un style 2D en pixel art, accessible sur un navigateur. Au début d’une nouvelle partie, le joueur aura le choix entre deux classes : l’archer et le guerrier, le joueur pourra ensuite jouer aux différents niveaux proposés par le jeu en affrontant les différents ennemis et en résolvant les différentes énigmes du jeu qui nécessite compétence et stratégie du joueur pour pouvoir progresser aux niveaux suivants.

Nous avons décidé de nous orienter vers le domaine du jeu vidéo, car ce secteur représente à lui seul une grande partie du marché, nous avons donc opté pour le style emblématique d’un jeu de plateforme avec un style 2D en pixel art comme Mario ou Sonic par exemple. De plus, lors de notre enfance, nous avons tous joué à ce type de jeu, ce serait un rêve devenu réalité de pouvoir réalisé un jeu de ce type. Nous avons décidé d’héberger notre jeu sur un navigateur afin qu’il puisse être facilement accessible à tous sans téléchargement préalable. Pour finir, les langages informatiques abordés sur ce projet ne sont pas entièrement maîtrisés par l’ensemble de l’équipe. C’est donc également une occasion idéale pour enrichir notre CV.

## Diagramme fonctionnelle

### Diagramme de cas d’utilisation de l’application

Dans le diagramme de cas d’utilisation sur l’image ci-dessous, nous pouvons apercevoir les différents éléments de notre application. On peut prendre le cas, d’un acteur n’ayant pas de compte, au départ, il sera positionné sur notre page d’accueil, il peut ensuite accéder à deux différentes pages sans comptes, la page à propos permettant d’avoir plus d’information à propos de notre site web ainsi que de notre jeu et la page permettant de nous contacter.

Si le joueur souhaite se connecter, ce ne sera pas possible, car il n’a pas de compte, il devra donc se rediriger vers la page d’inscription ou il devra remplir un formulaire en indiquant différentes informations telles que le mail, son nom, son pseudo…, et il n’est pas possible de pouvoir se créer un compte si ce compte est déjà existant.

Après que l’utilisateur soit inscrit, il peut désormais accéder à différentes pages comme la page permettant de modifier ses données utilisateur (nom, prénom, email…). Il peut également accéder à la page jouée qui contient le jeu Unity sous format WebGL, s’il a acheté le jeu sinon il sera redirigé vers la page d’achat du jeu, le joueur doit acheter le jeu avec le moyen de paiement qu’il souhaite par PayPal ou par carte bancaire et après ces différentes manipulations, il pourra jouer au jeu.

Si le joueur souhaite se déconnecter puis se reconnecter, mais qu’il a oublié son mot de passe alors il devra se diriger vers la page d’oubli de mot de passe, grâce à un lien qui se trouve sur la page de connexion, il devra alors choisir un nouveau mot de passe après avoir passé l’ensemble des étapes de sécurité. Après avoir réalisé cette étape, il pourra alors se connecter avec son nouveau compte.

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

Figure 1 : Diagramme de cas d'utilisation de notre application

## Diagrammes techniques

### Architecture globale du projet

Dans le diagramme technique ci-dessous, nous pouvons apercevoir l’architecture globale du projet.

Au niveau, du site web, nous avons utilisé pour le frontend React Native, pour le backend Node.js et pour la base de données MongoDB. Nous avons utilisé l’ensemble de ces technologies, car ils ne sont pas maîtrisés par l’ensemble des membres de l’équipe donc il s’agit de l’occasion idéal pour pouvoir apprendre l’utilisation de nouvelles technologies et d’avoir de nouvelles références sur nos CV.

Pour NodeJS, nous avons utilisé la librairie Express pour pouvoir réaliser des requêtes HTTP très facilement et utilisé la session proposée par Express permettant de sauvegarder des données dans les cookies de manière très sécurisé et pratique. Nous avons également utilisé la librairie Mongoose pour interagir avec notre base de données MongoDB.

Pour les API de paiement, nous avons choisi de travailler avec Stripe et PayPal, car les deux différents API propose une documentation assez bien fournie qui permettent donc une utilisation simple et efficace.

Pour le jeu, nous avons opté pour le moteur de jeu Unity avec le langage C#, Unity propose une documentation très détaillée, de plus, la communauté Unity est assez grande, cela signifie que si on rencontre un problème alors on pourra facilement le régler grâce à sa documentation et à sa communauté.

Unity permet également de transformer notre jeu sous format web (WebGL), cela permet donc, de pouvoir insérer le jeu dans une iframe sur notre site web.

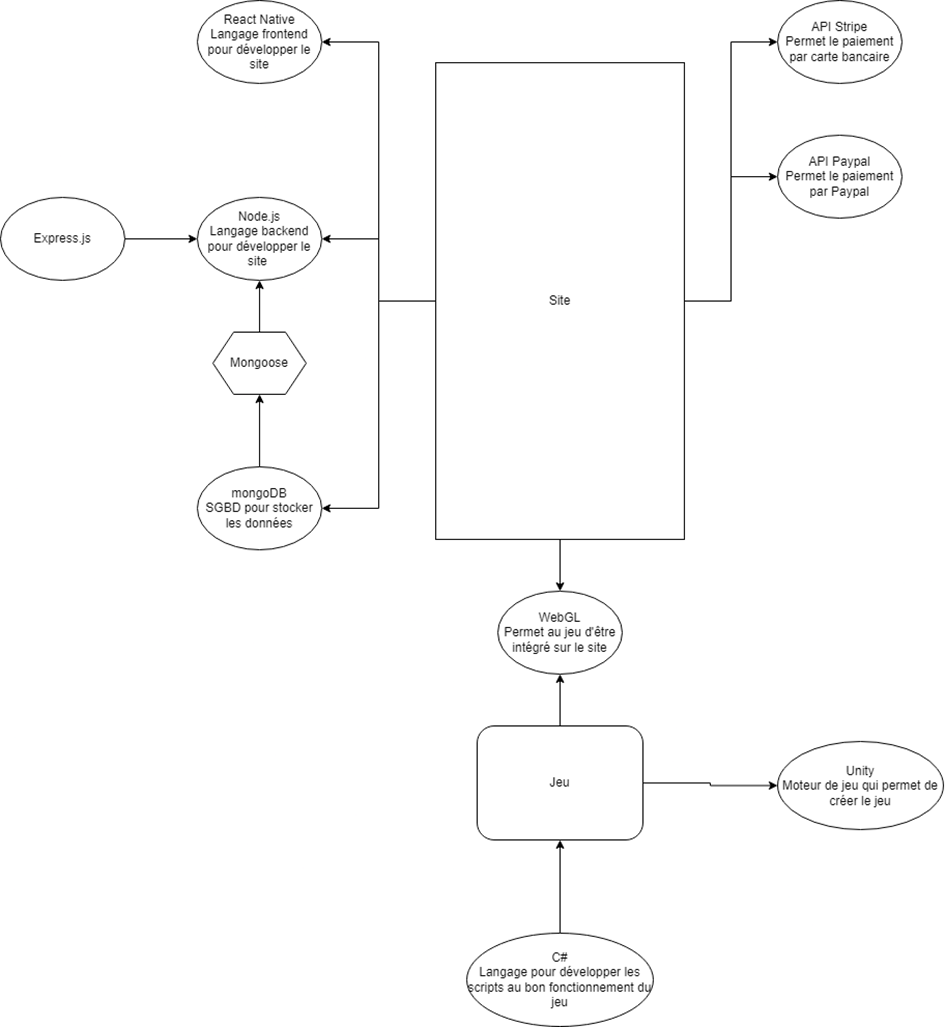


Figure 2 : Architecture technique globale du projet

Diagramme de séquence de la page oublie de mot de passe

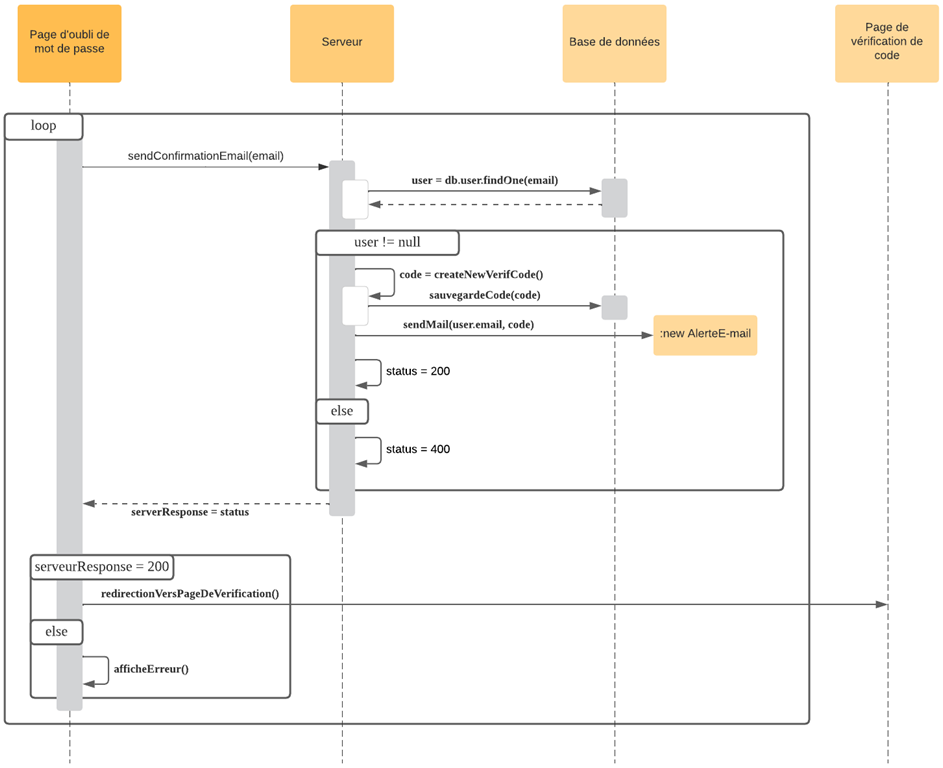


Figure 3 : Diagramme de séquence de la page oublie de mot de passe

### Diagramme de séquence de la page inscription

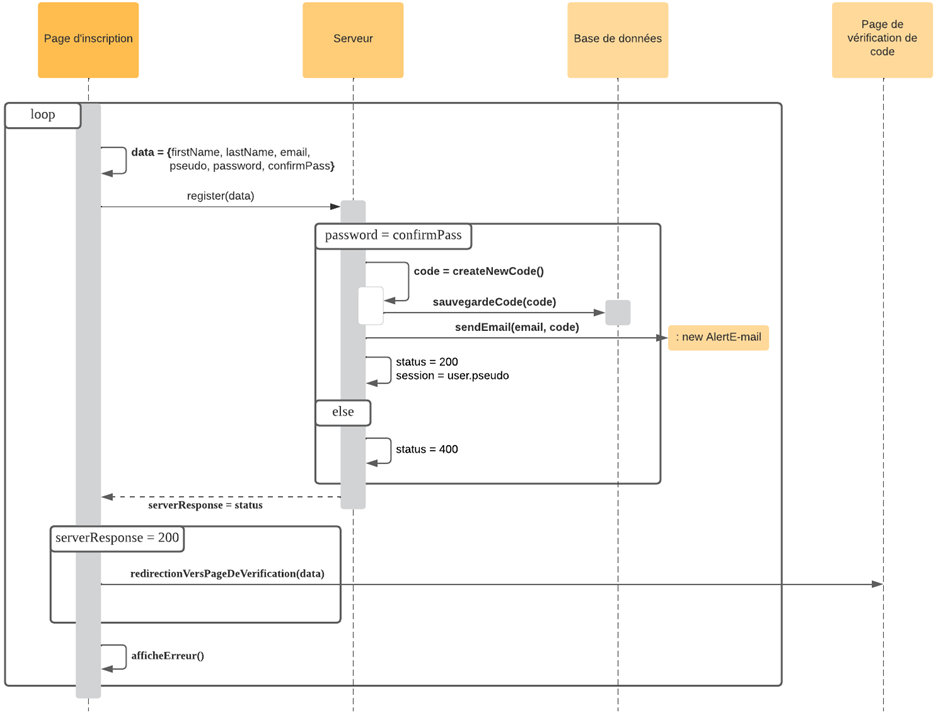


Figure 4 : Diagramme de séquence de la page inscription

### Diagramme de séquence de la page vérification de code

A screenshot of a computer

Description automatically generated

Figure 5 : Diagramme de séquence de la page de vérification de code

### Diagramme de séquence de la page de modification de mot de passe

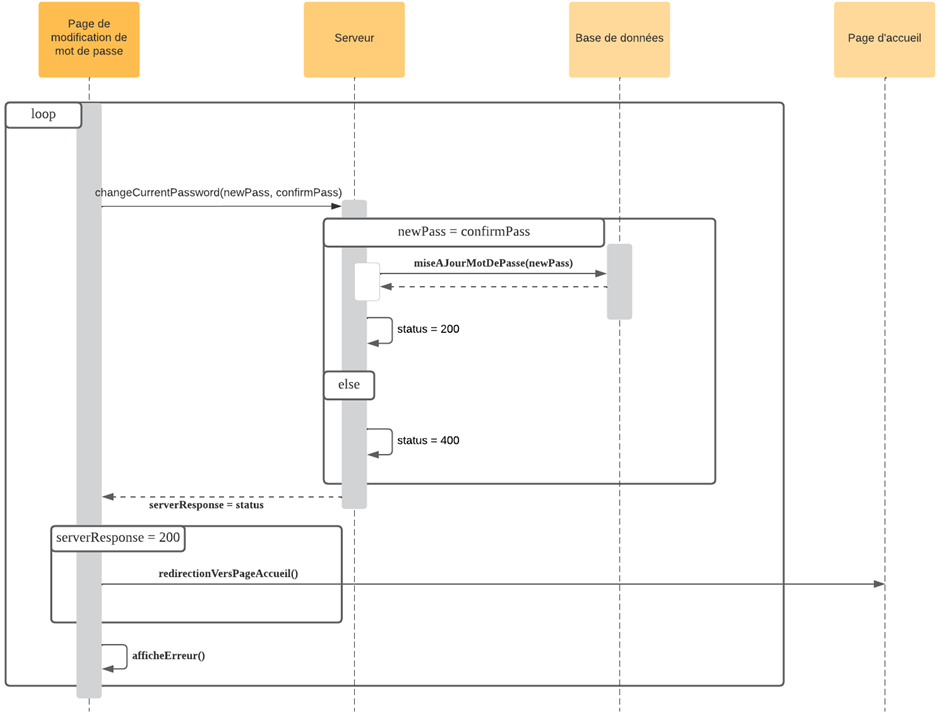


Figure 6 : Diagramme de séquence de la page de modification de mot de passe

### Diagramme de séquence de la page de modification des données utilisateur

图形用户界面

中度可信度描述已自动生成

Figure 7 : Digramme de séquence de la page de modification des données utilisateur

### Diagramme de séquence de la page de connexion

A screenshot of a computer

Description automatically generated

Figure 8 : Digramme de séquence de la page de connexion

### Diagramme de séquence de la page de contact

A diagram of a chat

Description automatically generated

Figure 9 : Diagramme de séquence de la page de contact

### 

### Les états et transitions du joueur

A diagram of a diagram

Description automatically generated

Figure 10 : Diagramme d'état-transition du joueur

### Les états et transitions d’un ennemi

A diagram of a diagram

Description automatically generated

Figure 11 : Diagramme d’état-transition des ennemies du jeu

# Organisation

Pour la SAE S6, nous avons décidé d'adopter le cycle en V (cf. image ci-dessous). En effet, après avoir mis en place et testé le SCRUM pour la SAE S5, nous nous sommes rendu compte qu'à chaque réunion hebdomadaire, visant à valider le sprint mais aussi dans le but de faire part des difficultés rencontrées de chacun(e), certaines personnes n'étaient pas présentes. Cependant, cela n'a pas été le cas pour la dernière réunion. Cette fois-ci, le cycle en V nous garantit que tous les membres de l'équipe seront présents pour la réunion finale.

A diagram of a variety of colors

Description automatically generated with medium confidence

Figure 12 schéma du cycle en V

Durant la phase de développement, après avoir développé une fonctionnalité, nous la testons puis l'intégrons directement dans le projet. Nous avons également réalisé un diagramme de Gantt (cf. image ci-dessous) illustrant toutes les tâches à effectuer.

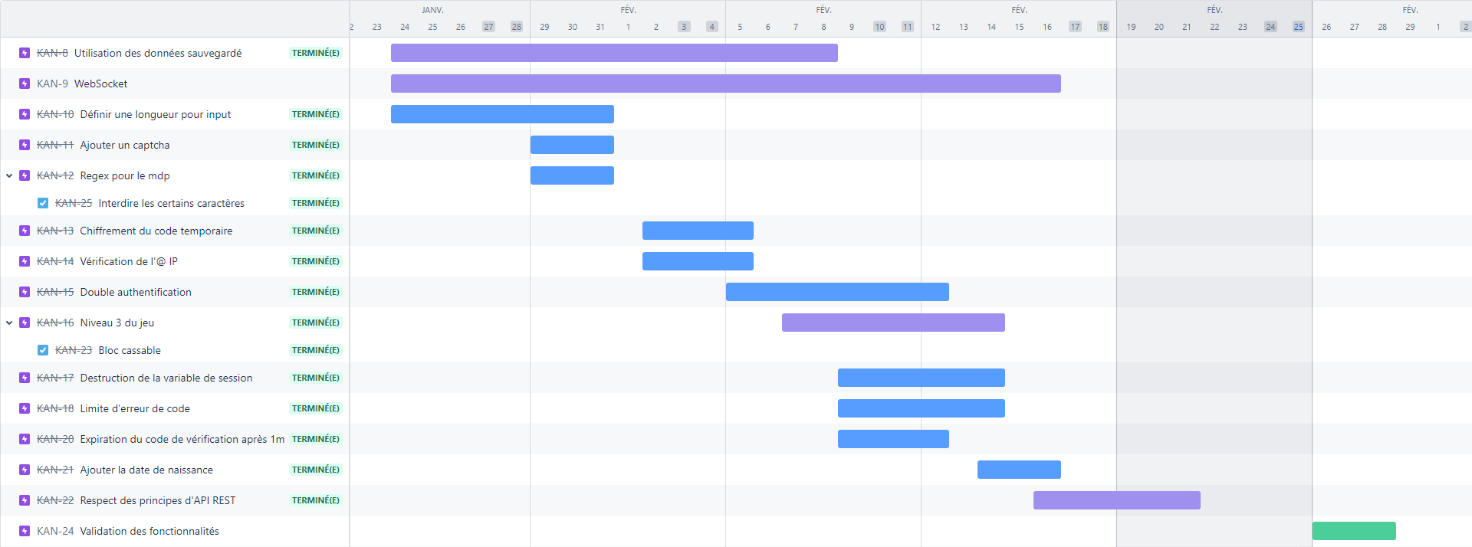


Figure 13 calendrier de réalisation des maintenances

# Maintenances évolutives

## Fonctionnelles

Dans la précédente SAE, la SAES5, nous avions mis en place un système qui sauvegardait les données des utilisateurs, y compris leur progression dans le jeu. Cependant, nous ne l'utilisions pas effectivement. À présent, nous avons réussi à l'implémenter. Les utilisateurs sont maintenant en mesure de reprendre là où ils s'étaient arrêtés lors de leur précédente partie.

La deuxième évolution fonctionnelle concerne l’ajout d’un troisième niveau, avec une nouvelle mécanique de jeu, qui est la possibilité de détruire un certain type de bloque afin de de progresser dans le niveau et de récupérer des récompenses.

Ce développement nous a offert l'opportunité d'approfondir nos connaissances aussi bien en design qu'en implémentation de fonctionnalités inédites.

## Techniques

L'aspect sécurité est l'un des plus importants dans une application, surtout pour celles qui communiquent avec une base de données. C'est pourquoi nous avons corriger deux vulnérabilités qui nous ont semblé être les plus importantes parmi celles répertoriées dans le top dix du projet OWASP (Open Web Application Security Project).

Notre application comportait de nombreux formulaires qui interagissent avec notre base de données. Cependant, étant concentrés sur les aspects fonctionnels de l’application durant la précédente SAE, nous n'avons pas immédiatement donné la priorité à la sécurité. Lors de notre analyse pour une maintenance évolutive, nous avons identifié deux failles qui nous ont paru importantes à sécuriser.

### Identification et authentification de mauvaise qualité (top 7)

Nos faiblesses d’authentification et nos solution apporté :

* Autorise les attaques automatisées telles que le bourrage d’informations d’identification, où l’attaquant dispose d’une liste de noms d’utilisateurs et de mots de passe valides.

Nous avons mis en place une limite d’erreurs : après cinq tentatives, le compte se bloquera. Un email contenant un code sera envoyé à l'utilisateur, qui devra l'utiliser dans les 30 secondes ; sinon, le compte restera bloqué pendant une dizaine de minute. Nous avons également mis en place une fonctionnalité qui limite l'inscription d'une même personne utilisant la même adresse IP à une fois par jour, afin d'éviter la surcharge de notre base de données.

* Autorise les mots de passe par défaut, faibles ou bien connus, tels que "Password1" ou "admin/admin".

Pour remédier à ce problème nous avons implémenté une expression régulière (regex) qui oblige l’utilisateur à entrer un mot de passe fort. Le code doit contenir : une lettre majuscule et une lettre minuscule, au moins un chiffre, un symbole parmi @ ! % \* ? & et avoir une longueur d'au moins 8 caractères.

* Utilisez des processus de récupération d'informations d'identification et de réinitialisation de mot de passe faibles ou inefficaces, tels que les « questions secrètes », qui ne peuvent pas être sécurisés.

La précédente version de notre application utilisait un code généré aléatoirement de 6 chiffres, compris entre 100000 et 999999. Cependant, cette méthode présente une faille, car rien n'empêche un malfaiteur de tester toutes les combinaisons possibles dans cet intervalle. Nous avons donc réduit la durée de validité de notre code de vérification, la faisant passer d'une heure à 30 secondes. De plus, nous avons aussi ajouter une donnée supplémentaire la date de naissance de l'utilisateur, afin de confirmer l’identité de l’utilisateur.

* Utilisez des mots de passe en texte brut, chiffrés ou faiblement hachés (voir A02:2021 – Défaillances cryptographiques).

Nous utilisons bcrypt comme algorithme de hachage pour crypter les mots de passe des utilisateurs. Cependant, nous ne l'appliquons pas aux codes de vérification. Maintenant, même le cote temporaire est crypté.

* Absence ou utilisation inefficace de l’authentification multi-facteurs.

La solution que nous avons apporté pour ce problème c’est l'implémentation un CAPTCHA et d'ajouter une authentification à deux facteurs.

* Ne pas invalider correctement les identifiants de session. Les sessions utilisateurs ou les jetons d'authentification (en particulier les jetons SSO) ne sont pas correctement invalidés lors de la déconnexion ou après une période d'inactivité.

Auparavant nous ne détruisons pas la variable de session d’un utilisateur après sa déconnexion ; nous nous contentons de lui attribuer une valeur nulle. Maintenant, nous la détruisons bien dans les règles de l’art.

### Injection (top 3)

* Utilisation de données hostiles.

Il est essentiel que nous nous protégions contre les injections NoSQL. Par conséquent, nous avons interdit tout caractères permettant de réaliser des opérations CRUD dans MongoDB ({ } $ [ ] \" ' : ; < >). Bien sûr pour éviter des problèmes en interne nous avons aussi interdit le mot « admin ».

## Technologiques

Nous avons transformé nos opérations CRUD en une API REST, afin de garantir une simplicité et une légèreté au niveau du serveur. L’API REST est aussi très évolutive, qui nous permettra par la suite d’ajouter davantage de fonctionnalités.

# Description technique de la nouvelle version de l’application

Cette deuxième version du digramme technique comporte deux nouvelles dépendances (couleur marron), l’API Speakeasy pour l’utilisation de la double authentification de google et l’API ReCAPTACH pour l’implémentation du CAPTCHA.

A diagram of a website

Description automatically generated

Figure 14 nouvelle version de l’architecture technique globale du projet

# Bilan

Nous avons réussi à atteindre tous les objectifs définis dans notre cahier des charges. Tout d'abord, on a utilisé les données sauvegarde de Progression des Joueurs, permettant aux joueurs de conserver leur progression, y compris les objets dans l'inventaire et le nombre de pièces collectées. Après, on a créé le 3ème niveau, introduisant des blocs cassables et enrichissant ainsi l'expérience du jeu. On a également renforcé la sécurité de l'authentification en ajoutant des fonctionnalités telles que la limitation d'erreur, le CAPTCHA, la destruction de session, etc. Enfin, on a transformé nos opérations CRUD en un API REST.

Même si on a réussi pas mal de choses, on a quand même rencontré un obstacle, WebSocket, qui marchait sur Unity mais pas sur le webGL. Les codes restent disponibles sur git, attendant une solution future.

Pour continuer à élever notre projet, on veut aller plus loin dans le terme de sécurité. Par exemple, rediriger nos sites de http vers https pour une communication plus sécurisée, aussi pour gagner la confiance au client, augmenter leur intention de visiter notre site. Un autre exemple est l’utilisation du schéma JSON, qui peut intégrer des validations pour les données entrées et les données sorties, garantir ainsi la purification des données.

# Conclusion

Chaque membre du groupe a pu en apprendre plus sur la sécurité d’un site, que ce soit découvrir les défauts d’un site à l’aide du projet OWASP. Dans le cas d’une attaque par force brute, un type d’attaque qui consiste simplement à rentrer un grand nombre de mots de passe jusqu’à trouver le bon, il faut implémenter dans le site un petit délai d'une durée très courte comme 0.1 seconde, qui peut réduire considérablement une attaque dans ce genre. Une autre solution est de bloquer l'accès au compte quand plusieurs erreurs sont réalisées pendant une certaine durée, c'est cette solution qu'on a implémentée dans notre projet. Une autre attaque populaire est l'injection consistant à essayer d'ajouter une commande sql dans un champ de caractères afin que la commande se lance. La solution contre cette attaque est simplement d'interdire simplement les caractères qui pourraient être utilisés dans des lignes de codes, ce qu'on a rajouté aussi sut notre site. On a aussi été déçu de ne pas avoir pu implémenter les websockets, malgré les efforts réalisés pour implémenter ce dernier, il y avait de nombreux problèmes de compatibilité de la bibliothèque qu'on a utilisé avec la compilation en webGL. Dans un dernier espoir de faire fonctionner cette fonctionnalité, on a trouvé un plugin de notre bibliothèque qui était censée la rendre compatible avec le webGL. Cependant ce plugin n'a rien changé à notre problème, on a dû alors abandonner cette fonctionnalité contre notre gré.

Annexes