Rapport Universitus

https://gitlab.com/Adri-/universitus

Introduction

Ce projet s'inscrit dans le cours de Projet de Communication Transdisciplinaire. Nous devons mettre en place un guide pour les nouveaux étudiants en informatique, leur permettant de faire connaissance avec les services disponibles à l'université, et leur permettant d'apprendre les commandes de base de Linux.

Implémentation

Choix techniques

Le jeu sera une application web, permettant ainsi de faciliter son accès aux utilisateurs. Nous avons donc du passer par une phase d'apprentissage des technologies liées au web, car la majorité de notre groupe n'avait que quelques bases dans ce domaine.

Front-End

Pour le côté utilisateur, nous avions dans un premier temps cherché à utilisé des émulations de terminaux pré-fait. Mais ceux que nous avions trouvé n'étaient pas suffisament modulables, étaient "lourds", et ajoutaient de nombreuses technologies à apprendre (ReactJs, Less, ...).

Nous avons donc décidé de réaliser le terminal nous même, inspiré de divers projet libre sur github. L'émulation du terminal à donc été réalisée en Javascript, sans utiliser de framework. Cela nous a permi de garder une application légère que nous maîtrisions.

Nous avons donc utilisé du HTML 5, CSS 3 et Javascript ES6. Notre application est donc compatible avec une large majorité de navigateur web.

Le terminal permet d'avoir une coloration basique, et permet l'utilisation d'un éditeur de texte, ace, pour que le joueur puisse facilement modifier des fichiers.

Pour la communication avec le serveur, nous avons utilisé dans un premier temps des appels Ajax. Puis en testant différentes technologies, nous avons décidés d'utiliser les WebSocket. Permettant de grandement simplifier la communication avec le serveur, toujours en gardant une compatibilité avec les navigateurs web les plus utilisés.

Back-End

Côté serveur, nous avons décidé d'utiliser NodeJs. Ce choix nous permettait d'utiliser nos connaissances nouvellement apprises en javascript côté serveur. De plus, l'utilisation de NodeJs nous permettait d'utiliser le package Dockerode, qui nous permet de simplifier la gestion des conteneurs.

Conteneurs

Nous avons décidé d'utiliser Docker, qui permet de fournir à l'utilisateur un univers *bac-à-sable* dans lequel il pourra réaliser des actions qui seraient normalement *dangereuse*s. Ainsi, même si l'utilisateur parvient à supprimer des éléments importants du système entrainant un crash du conteneur, on pourra facilement le relancer. Et cela sans impacter les autres utilisateurs.

Les conteneurs sont géré par le serveur, grâce à Dockerode. Le conteneur est créé à la connexion de l'utilisateur, et est fermé lorsqu'il quitte l'application.

Jeu

Pour le jeu, nous pouvions utiliser n'importe quels langages grâce à l'utilisation des conteneurs. Nous devions utiliser un langage nous permettant de faire des appels aux commandes Linux.

Nous avons choisi Python 3, car c'est un langage que nous avons tous vu en cours, et notre application ne demande pas des performances très élevées. De plus, Python est un langage de haut niveau, qui nous permet de gagner du temps de développement et de traiter facilement les chaines de caractères.

Notre jeu a été réalisé en gardant à l'esprit qu'un utilisateur puisse facilement ajouter du contenu. Ainsi l'ajout de quêtes et de personnages se font à l'aide de fichiers de configuration.

Notre jeu est une surcouche du système, dans lequel le joueur est bloqué. Le joueur évolue donc dans l'arborescence du système, dans lequel le jeu ajoute du contenu.

Gestion du projet

Pour la gestion de notre projet, nous avons utilisé les fonctionnalités fournies par GitLab. Nous avons ainsi les taches planifiées dans un ScrumBoard, nous permettant de générer automatiquement le diagramme de Gantt.

La première partie du projet à été la réalisation du cahier des charges. Nous nous sommes basés sur le cahier des charges que nous avions réalisé le semestre précédent. Ce qu'il nous manquait alors était la partie sur les technologies à utiliser.

Nous avons donc pris du temps pour apprendre les différentes technologies liées au web, afin de choisir celle qui nous permettais de mener à bien le projet.

Nous nous sommes ensuite partagé le travail, Yann et Romain travaillant sur le Game Design, Floriant et Isaak sur le front-end, Adrien sur la base de données et le jeu python, et Robin sur le serveur NodeJs et Docker.

Cette séparation des taches nous a permis de nous spécialiser chacun dans un domaine du projet, et de gagner ainsi du temps.

Futur du projet

Les quêtes actuellement implémentées ne permettent pas au joueur d'aller très loin, mais il est facile de rajouter du contenu comme les Quêtes grâce au système de fichiers de configurations en .quest, ainsi que des personnages avec des dialogues poussés et évidemment de modifier à souhait le comportement des commandes ainsi que la structure du monde.

Mise en production

Le projet est publiable automatiquement grâce à une image Docker, et il est possible de le mettre à jour directement depuis la branche master grâce à un script. La base de données, relativement légère, est importable depuis un fichier .sql.

Conclusion

Ce projet nous a permis de découvrir la programmation web, et l'organisation d'un projet sur le long terme.

Il nous a permis de découvrir des outils tels que le ScrumBoard et le diagramme de Gantt, ainsi que de prendre en compte les besoins non-fonctionnels d'un client, puis d'adapter notre vision du projet à la sienne.