Rapport de stage

Développement du backend et gestion des logs

Photo by Stephan Louis on Unsplash

Auteur
Rioja Kenneth, Master in learning and Teaching Technologies (MALTT)

Institution
Laboratoire d'Innovation Pédagogique (LIP), TECFA, UNIGE

Responsable terrain
Jaouadi Mariem

Date
18 Mars 2024 – 5 Juillet 2024
(16 semaines à 50%)

Date de remise du rapport
20 août 2024

TABLE DES MATIÈRES

Introduction

En tant que chercheur·euse, la collecte de données est une étape obligatoire dans le cadre de recherches scientifiques. Or, lorsque l'objet de recherche est un jeu vidéo distribué hors des murs du laboratoire, il est nécessaire de collecter des données de manière sécurisée sur le serveur de l'institution encadrant le projet de recherche. En l'occurrence, les données d'un jeu vidéo sont souvent représentées sous la forme de *logs*,

c’est-à-dire toute action *X* des joueur·euse·s à un temps donné *T*. La récolte des données est donc dite « event-based » (Serrano-Laguna et al., 2017).

À ce jour, il n’existe aucun système ou application externe permettant de tracer ces actions dans des jeux vidéo, à l’instar de Qualtrics/Limesurvey pour les questionnaires en ligne. Une autre couche de difficulté est présente lorsque les actions *X* sont choisies *à priori* par les chercheur·euse·s. Un moyen extrêmement bas niveau serait de récupérer chaque touche au moyen d’un *keylogger*. Cependant les données ne seraient pas directement lisibles du côté des chercheur·euse·s ce qui nécessiterait un travail important en aval pour les rendre lisible. Un autre moyen serait d’enregistrer les parties sous forme vidéo et d’analyser les actions dans le jeu. Mais nous nous retrouvons dans la même situation que pour le *keylogger*. En outre, ces deux techniques (i) sont coûteuses en temps ; (ii) nécessitent de traduire une série d’actions en objet.

Le moyen choisi ici est pour permettre de récupérer les logs d’un jeu vidéo est de créer une RESTful API (*Interface de Programmation d’Application respectant les contraintes du style d’architecture Representational State Transfer*) en même temps que la conception du jeu vidéo. Cet outil permet (i) de récupérer en temps réel les actions des joueur·euse sans action de la part des chercheur·euse·s, (ii) de stocker les données sous forme lisible et utilisable.

Moi-même chercheur et intéressé à récupérer des logs d’un jeu vidéo joué en ligne, j’ai souhaité m’adonner au challenge de créer une RESTful API. Ce rapport de stage vous montrera les objectifs du stage ainsi que les tâches que j’ai effectuées, pour ensuite voir les résultats obtenus et nous finirons sur un bilan personnel.

LIP – Laboratoire d'innovation pédagogique

Le Laboratoire d'Innovation Pédagogique (LIP) est une équipe dont les travaux relèvent de la recherche orientée par la conception (*design-based research*, Sanchez & Monod-Ansaldi, 2015). Initialement mis en place à l'Université de Fribourg, le laboratoire a aujourd'hui rejoint le TECFA à l'Université de Genève où il mène des projets qui consistent dans la conception de dispositifs numériques d'apprentissage innovants, à leur expérimentation en conditions naturelles et à l'analyse des données collectées lors de ces expérimentations. (source)

Historique – conception du système de traçage dans TSADK (V2)

Le jeu TSADK concerne l'apprentissage de la programmation en Java, et se déroule dans un château à l'intérieur d'un monde médiéval fantastique. Le joueur est piégé dans le jeu et il doit réparer un code en résolvant des énigmes proposées par des personnages-non joueurs (PNJs).

Effectifs

La conception du jeu TSADK se fait par une équipe pluridisciplinaire dont les acteur·ice·s sont issu·e·s de trois institutions : IMT Nord Europe, Université Grenoble Alpes et Université de Genève. Sont présent·e·s :

1 responsable du projet (IMT Nord Europe), 2 doctorantes, et 1 chercheur sur la conception et l'analyse des usages du jeu (TECFA-LIP), 1 chercheure en méthodes de recherche et 1 chercheur en didactique de l'informatique (LIG- Université de Grenoble), 2 enseignants-chercheurs en informatique (IMT Nord Europe), 1 ingénieure techno-pédagogique (IMT Nord Europe) et 3 autres concepteurs rejoignent l'équipe (IMT Nord Europe)

J’ai été en étroite collaboration avec Mariem Jaouadi, ma référante pour ce stage, doctorante à l’Université de Genève, Ouiam Kourchi, la développeuse du jeu et étudiante à l’IMT Nord Europe et Stéphane Morand, ingénieur système de l’équipe TECFA. Je les remercie d’avance tous et toutes pour les échanges fructueux !

Lieu du stage & Position

Le stage s’est effectué à hauteur de deux jours et demi par semaine pendant 16 semaines. Il s’est déroulé à la villa Battelle, dans le campus Battelle à Carouge, Genève. La position occupée est *stagiaire en spécification collaborative des traces numériques d'interaction*.

Mission – Développement du backend et gestion des logs

Objectifs de la mission

Planification initiale des besoins spécifiques dans le jeu TSADK

(source)

- Comprendre les besoins spécifiques en matière de logs (types de données, fréquence, volume) et définir les spécifications du module des traces à mettre en place
- Choix technologiques :
    - Sélectionner le langage de programmation et le framework pour le développement du backend en fonction des compétences de l'étudiant et des besoins du projet : Java, C, etc.
    - Choisir le système de gestion de base de données (SGBD) adapté au stockage et à la gestion des logs de joueurs.
    - Établir un plan du travail avec des objectifs et deadlines.

Développement technique du Backend de la collecte des traces

- Configurer l'environnement de développement (local setup), y compris les outils nécessaires, et les bibliothèques.
- Concevoir et implémenter l'architecture de l'API, incluant l'authentification, la gestion des erreurs, et la pagination si nécessaire.
- Développer les *endpoints* nécessaires pour la réception, le traitement et la récupération des logs de joueurs.

- Se synchroniser avec le développeur pour mettre en place une infrastructure pour gérer les logs
- Tester l'API pour s'assurer de sa fiabilité, de sa performance, et de sa sécurité
- Travailler en étroite collaboration avec la game developer Unity pour intégrer l'API avec le client de jeu
    - Tester l'intégration pour s'assurer que les logs sont correctement capturés, transmis, et stockés.
    - Réaliser des tests de charge pour valider la scalabilité et la performance de la solution.

Conception et gestion de base de données

- Création de la structure de la base de données qui stockera les logs (MySql ou postgreSQL). La modéliser la structure de la base de données se fera en collaboration étroite avec Mariem
- Mettre en place des procédures pour l'insertion, la mise à jour, et la récupération sécurisée des données.
- Conformité et Sécurité des Données :
    - Élaborer une liste de règles pour le traitement des données personnelles (nom, prénom, email) et des données sensibles, en conformité avec le GDPR.
    - Implémenter des mécanismes pour l'anonymisation/pseudonymisation des données personnelles (et sensibles) conformément aux règles du GDPR. Réglementations comme le RGPD (Data masking)
- Préparation des données pour l'analyse : nettoyage, la structuration et éventuellement

Descriptif des tâches et calendrier

Voir liste des tâches et calendrier ici : https://unigech-my.sharepoint.com/:x:/r/personal/kenneth_rioja_unige_ch/Documents/LIP_stageDev_KR_2024/calendrier_lip_kr_202403.xlsx?d=w225713eb8f0f40adbaef3b18c035dfd2&csf=1&web=1&e=cJkMGM

En général, les tâches durant ce stage se résument à : auto-formation sur les concepts et/ou langages de programmation, mise en place effective

des apprentissages sur l'environnement de développement (en local), discussion avec Mariem et/ou Ouiam sur les choix d'architecture. Ces tâches en itération puis à la fin, la tâche principale était le déploiement sur le serveur *tecfacl3* avec l'aide précieuse de Stéphane.

1er mois – 18.03/12.04 – Plannification

Les premières semaines ont été consacrées à « comprendre le contexte » selon Mariem et décider d'un framework pour la RESTful API. J'ai donc lu en détail la proposition de stage, les détails du stage, le cahier des charges du jeu TSADK, ainsi que les slides de la conception de l'architecture back-end. Pour mettre en place mon environnement de développement je souhaitais être assisté par IA, c'est-à-dire avoir une sorte de *copilot* (hors GitHub pour plus de confidentialité). J'ai donc téléchargé Tabby, que j'ai utilisé les quelques premiers jours d'apprentissage mais que j'ai laissé tomber par la suite du fait que je suivais les tutoriels et que je ne ressentais pas le besoin d'être efficient au niveau de l'écriture du code. De plus, je sentais l'outil un peu lent. J'ai suivi les vidéos que Mariem m'avait proposées, entre autres, sur les concepts de base pour le web service, adopter les API REST pour vos projets web, base de données MySQL, lu les documents attraits à la RGPD, REST API Mistakes Every Junior Developer should Avoid | clean-code (lien vers mes notes). Le choix pour le langage de programmation a utiliser pour la création de la RESTful API s'est décidé selon plusieurs critères : 1) quel langage je souhaitais approfondir ; 2) l'open-source ; 3) la communauté ; 4) l'accessibilité. Mon choix s'est donc porté sur Python.

2è mois – 15.04/09.05 – Apprendre Python

Le deuxième mois a été consacré à se former au langage Python puis à trouver et travailler sur un framework. J'ai suivi entre autres les cours suivants : Ecrivez du code Python maintenable. Une tâche initiale était de trouver le framework correspondant à la demande, c'est-à-dire presque uniquement recevoir des données d'un client.

"Look for frameworks that support features like load balancing and database connection pooling, which enable your application to handle high traffic and perform well under heavy loads (...) Community of support and documentation (...) Security (...) " (source)

J'ai donc commencé par me pencher sur Django et sur les cours suivants : Débutez avec le framework Django, Mettez en place une API avec Django Rest framework.

3è et 4è mois – 13.05/05.07 : Développement et déploiement

Une fois avoir compris ce qu'était concrètement Django via le suivi des cours, je m'aperçois que ce framework est un service 'tout en un'/*'batteries included'*. En suivant les tutoriels liés à ce framework, l'architecture créée est non pas qu'un backend – comme souhaité – mais il s'agissait aussi d'un frontend avec de multiples dépendances, or cela ne correspondait pas à la simplicité (en termes de quantité ici) de ce qui était demandé : avoir un backend qui puisse recevoir une requête 'POST' et sauvegarder les logs. Je décide donc, à mi-chemin du stage, de changer de framework et me concentre sur le 'micro-framework' Flask (source).

Je suis donc ce premier article Designing a RESTful API with Python and Flask, puis je pioche dans les quelques très bons articles de Miguel Grinberg et son Flask Mega-Tutorial. Ce qui me permettra a la fin de créer l'architecture actuelle, présente sur GitHub. Je mets en place l'architecture globale, puis j'y ajoute des améliorations comme l'utilisation de *blueprints* ou d'un *application factory pattern*, de décorateurs, la gestion des erreurs, une route pour exporter les logs sous forme d'un csv, une liste de contrôle d'accès (ACL), une authentification par jeton (JWT).

Je m'arrange avec Stéphane afin qu'il puisse me donner accès au serveur *tecfacl3.unige.ch* ainsi que les commandes nécessaires pour permettre le déploiement de l'API sur le serveur. Il a été difficile de trouver les commandes nécessaires au déploiement alors que je finissais mon architecture, il y avait notamment des soucis au niveau du système avec

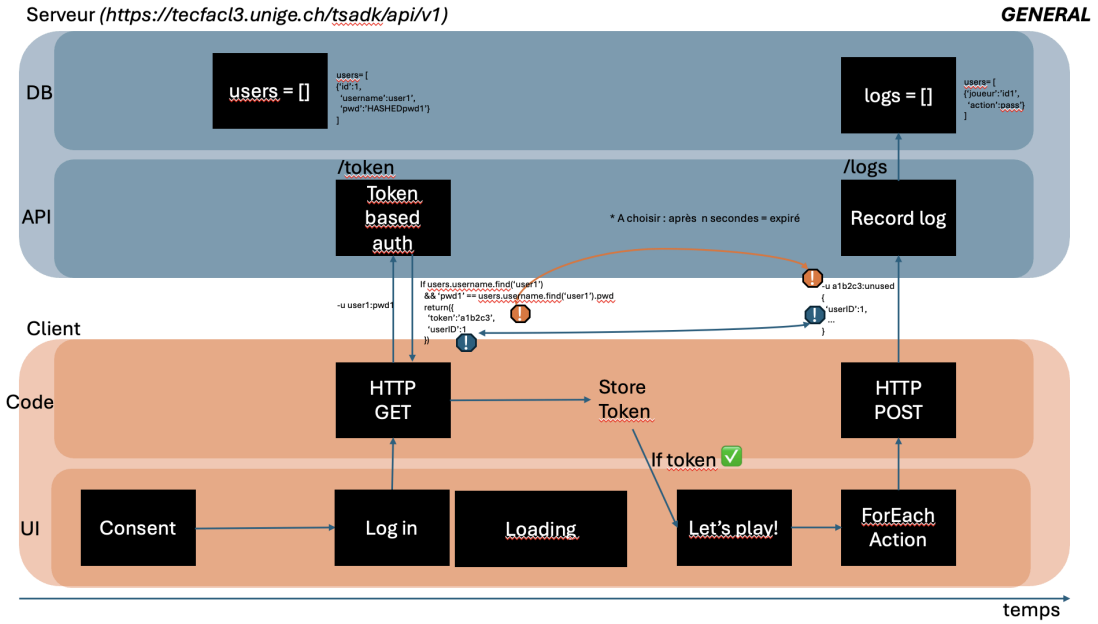
les commandes *git clone*, *python3*, et *python-venv* (= je ne pouvais pas les exécuter dès le début).

Une fois mon architecture en place en local, j'ai créé des slides que j'ai partagées et présentées à Mariem et Ouiam. La réunion avec Ouiam m'a permis de venir avec des propositions sur l'authentification, le schéma de la base de données et les variables des logs. Des discussions concernant l'authentification nous a permis de revoir cette partie de manière plus simple : la table *users* sera préremplie et les expérimentateur·trice·s auront accès à *n* paires d'identifiants – mot de passe à distribuer aux joueur·euse·s.

Les deux dernières semaines ont été allouées au déploiement avec des échanges avec Stéphane. Ici, ma tâche était de s'assurer de rendre une version finale de la RESTful API avec les changements proposés par l'équipe et de produire le nécessaire pour pouvoir donner la main aux prochaines personnes qui seront sur le projet. Il y a notamment une documentation exhaustive couvrant le code ainsi qu'une vidéo explicative sur le test de la RESTful API en local.

Tout au long du stage j'ai pu suivre les meilleures pratiques de développement, comme le code propre (suivi la guideline de style PEP8) et l'utilisation de contrôle de version (GitHub).

Réalisations et Résultats obtenus



GitHub : https://github.com/kennethrioja/flask-restfulapi

Vidéo Youtube : https://www.youtube.com/watch?v=yZxAamS0hrY

Présentation du travail à l'équipe de recherche TECFA le 24 juin :

lien zoom, mdp : uSTm3!XX

Évaluation des réalisations par la responsable du stage (Mariem Jaouadi)

Kenneth Rioja a démarré le projet en prenant le temps nécessaire pour comprendre les exigences et a démontré un engagement fort pour atteindre les objectifs de l'équipe TSADK, en matière de développement de l'API RESTful et de communication avec la développeuse du jeu TSADK.

Durant les choix techniques et la mise en place du système de collecte des traces, Kenneth a fait preuve d'une capacité à écouter les retours et à les utiliser de manière constructive. La conception de l'authentification a connu plusieurs itérations, avec une forte contrainte de simplifier le processus de développement. Cet effort avait pour double objectif de faciliter le travail des développeurs et de respecter les délais imposés par la durée limitée du stage (16 semaines à 50%).

La mise en place de la BD est effectuée sans incidents techniques, et en respectant le schéma discuté lors des échanges. La BD "tsadk" est considérée comme une boîte noire, via des requêtes API. Pour visualiser ou modifier les utilisateurs, nous utilisons des commandes curl spécifiques. Cela peut être courant dans les premières phases de développement. Kenneth suggère de réinitialiser la base de données si nous voulons enlever cet utilisateur et en ajouter d'autres.

L'architecture mise en place intègre plusieurs pratiques de sécurité et une base solide et suffisante pour les besoins du stage, incluant des mécanismes comme des tokens de session de durée limitée (24h) et la protection des logs. La fonctionnalité d'export des données au format CSV est également implémentée, permettant ensuite une analyse facile des logs.

Le déploiement de l'application s'est réalisé sans encombre. Kenneth a efficacement mobilisé les ressources du laboratoire pour assurer l'hébergement sur le serveur **tecfacl3.unige.ch,** démontrant sa capacité à planifier, exécuter et documenter des tâches techniques avancées.

Compte tenu de la durée limitée du stage, certains aspects, tels que la conformité RGPD et LIPAD, nécessiteront des améliorations et des raffinements futurs. Cela inclut la possibilité pour les utilisateurs de voir, modifier ou supprimer leurs informations personnelles. Il est également possible aussi d'envisager les mesures avancées telles que le rate limiting, le monitoring. Finalement, Kenneth a efficacement mis en place l'endpoint GET /v1/logs, accessible uniquement par les administrateurs. Il pourrait être bénéfique d'introduire une fonctionnalité de pagination pour cet endpoint. Cela permettrait d'optimiser l'utilisation de la mémoire, surtout si nous travaillons avec un volume potentiellement élevé de logs.

Bilan personnel d'apprentissage

Quels étaient les objectifs personnels du/de la stagiaire dans le contexte du stage ? Ces objectifs ont-ils été atteints, ont-ils évolué au cours du stage ?

Avant de commencer le stage, je souhaitais être capable de :

- *Mettre en place une **architecture de type RESTful API** pour récolter des logs d'un jeu vidéo* (résultat : ✅)
    - *Tout en suivant des documentations officielles et ne pas suivre de tutoriels sur Youtube* (résultat : ⚠)
- *Mettre en place un environnement de développement sur **Docker*** (résulat : ❌)
- *Faire des **tests unitaires*** (résulat : ❌)
- *Consigner mon avancement sur **GitHub*** (résultat : ✅✅✅)
- *Mettre ne place un **pipeline CI/CD*** (résulat : ❌)

Quelles connaissances et compétences acquises au sein du MALTT ont été mobilisées au cours du stage ?

La gestion de projet est une compétence que j'ai pu mobiliser au cours de ce stage ainsi que la documentation et la présentation écrite des savoirs afin de les rendre accessibles et digestes (= utilisables) pour une future personne reprenant le flambeau.

Quels ont été les apports de l'institution du stage dans la formation en matière d'organisation du travail, d'acquisition de techniques, d'outils numériques, de connaissances, de compétences, de savoir-être ?

J'ai appris à utiliser Python, le langage de programmation le plus accessible et ayant une large communauté. Ayant déjà de l'expérience en programmation avec les langages C, C++ et JavaScript, apprendre Python n'était pas une tâche insurmontable. J'y ai pris du plaisir et je comprends désormais que sa notoriété vient de sa facilité d'écriture. Entre autres, pas

de parenthèses si points virgules ‘superflus’, des frameworks faciles à utiliser pour des tâches complexes, comme lancer un serveur, etc.

J’ai suivi la convention de style PEP8 qui permet d’écrire du code Python facile à lire, (qui j’espère sera) maintenable. Il s’agit là d’un pas en plus vers un style d’écriture plus propre (clean-code) qui s’est ajouté sur mes bases solides. Bases que j’avais déjà acquis grâce à l’école 42 où la *norminette* était un programme qui vérifiait notre syntaxe jusqu’à l’utilisation d’un Tab vs. 4 espaces.

Pour écrire du code Python maintenable, j’ai été sensibilisé à l’architecture MVC (Model-Vue-Controller) qui permet d’avoir des fonctionnalités du code dans des endroits bien distincts. Le modèle retenant toutes les infos relatives à l'état du système, c’est-à-dire les fonctionnalités brutes de l'application. La vue étant l’interface visuelle et/ou sonore pour l’utilisatrice. Le contrôleur qui garantit que les commandes sont exécutées correctement, modifie les objets, met à jour l'application, en somme, s’occupe des rouages de l'application.

Ensuite, les petits ajustements à la suite de mes demandes à l’ingénieur système afin de me donner les droits et commandes nécessaires ont été les moments les plus incertains du stage. Stéphane étant très efficace, a pu en un jour en moyenne, débloquer la situation. J’ai appris ici que si j’ai besoin de tester un environnement autre que celui que j’ai en local, il me faut déjà créer et tester le strict minimum dans le serveur de production. Je saurais maintenant faire attention à cela dans mes projets futurs.

Conclusion

Forces et faiblesses du stage par rapport au projet professionnel

Une énorme force, malgré le fait de ne pas avoir pas tant mobilisé de compétences du MALTT. J'ai pu développer ici des compétences complémentaires à celles enseignées au MALTT. Il s'agissait d'un pari risqué, et je suis heureux de l'avoir mené à bien. Pas de faiblesse du tout, il n'y a eu aucune redondance dans les apprentissages, que des choses nouvelles et qui s'articulent parfaitement avec ma perspective personnelle.

Quelles perspectives professionnelles et personnelles ?

Le but personnel quant à ce stage était d'être capable de récolter des données d'un jeu vidéo. Je me répète avec le but du stage, mais il s'agissait réellement pour moi de l'objectif. Il y a un an que je développe des jeux avec deux amis, et la récolte de données en temps réel est une nette amélioration dans la quête d'expertise du développement de jeux.

Bibliographie

Sanchez, É., & Monod-Ansaldi, R. (2015). Recherche collaborative orientée par la conception. Un paradigme méthodologique pour prendre en compte la complexité des situations d'enseignement-apprentissage. *Éducation & didactique*, *9*(2), 73–94. https://doi.org/10.4000/educationdidactique.2288

Serrano-Laguna, Á., Martínez-Ortiz, I., Haag, J., Regan, D., Johnson, A., & Fernández-Manjón, B. (2017). Applying standards to systematize learning analytics in serious games. *Computer Standards & Interfaces*, *50*, 116–123. https://doi.org/10.1016/j.csi.2016.09.014

INDEX

Backend : Code qui est littéralement « à l'arrière ». Désigne tout code/programme/application permettant des manipulations que l'utilisateur·trice ne voit pas à l'écran. Les serveurs sont souvent désignés comme faisant partie du backend. Peut s'employer de cette manière : « *Tu fais quoi là ? – Je fais du backend.* ». Contraire de Frontend / Interface utilisateur·trice. Javascript n'est en revanche pas considéré comme du « back », sa résultante se voit à l'écran, il s'agit d'un langage « front ».

Framework : Sous-catégorie d'un langage de programmation qui va permettre d'offrir des fonctions ou manipulations spécifiques au framework, par exemple pour lancer un serveur, le framework Flask du langage Python propose une commande spécifique, si nous n'utilisions pas Flask, il faudrait le faire « à la main ». Similaire à un voyage que l'on aurait commandé via une agence qui vous offre un paquet, comparativement à si vous deviez choisir les hôtels, les visites, les transports « à la main ».

Logs : Historiquement, l'expression "log in" vient de l'usage informatique introduit au début des années 1960, notamment avec le système de partage de temps CTSS (Compatible Time-Sharing System) du MIT. Avant les systèmes de partage de temps, l'interaction avec les ordinateurs se faisait principalement par traitement par lots, sans nécessité de se "connecter". Le terme "log" trouve son origine dans les journaux de bord des navires (logbooks), où les marins enregistraient les détails de la navigation. Cette pratique elle-même dérive de l'utilisation d'un "log" (un morceau de bois attaché à une corde) pour mesurer la vitesse d'un navire en mer. Ainsi, "log in" signifie littéralement enregistrer son accès, tout comme on enregistrait la vitesse d'un navire dans un journal de bord. (Résumé ChatGPT à partir de https://www.designcult.org/2011/08/why-do-we-call-in-logging-in.html). Un « Log » signifie toute donnée récoltée à la suite de chaque action d'un·e utilisateur·trice.

RESTful API : *Interface de Programmation d'Application respectant les contraintes du style d'architecture Representational State Transfer*.

Interface permettant l'échange d'information entre deux ordinateurs sur Internet. L'échange d'information suit les opérations de base pour la persistance des données: la création, la lecture, la mise à jour et la suppression (CRUD, create-read-update-delete).