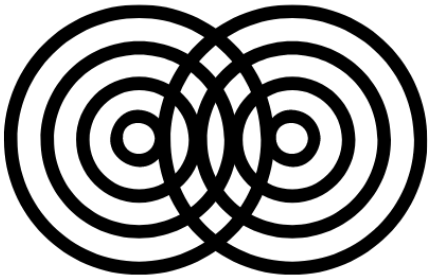
Rapport de soutenance 2 - Projet S2

Maxime TRIMBOLI Mickael RAZZOUK Rayan MAZOUZ

Yann THEVENIN



VERSINE

By K.I.S.S

Table des matières

Introduction

Dans le cadre de notre projet informatique en InfoSup, nous avons choisi de développer un réseau social. Notre réseau s'appelle Versine et a pour but de rapprocher les gens tout en garantissant le respect de leurs vies privées, et de leur offrir plus de contrôle sur ce qu'ils souhaitent montrer à leurs proches.
Ce rapport va vous permettre de découvrir les idées sur lesquelles se base Versine et la réalisation de ce projet.

Le projet Versine

2.1 Présentation de Versine

2.1.1 Notre idée de Versine

Versine nommé ainsi après la fonction trigonométrique du même nom, est un réseau social qui a pour ambition de retirer les aspects malsains de ses concurrents plus connus, le but est ici de créer une plateforme d'échange saine où les utilisateurs profiteront d'un environnement de partage virtuel confortable. Pour atteindre ces objectifs, nous avons décidé de nous focaliser sur trois points.
Premièrement le contrôle des informations diffusées sur notre site, il n'est pas question d'inciter à la sur-exposition de notre vie privée et les Circles sont là pour répondre à ce problème. Deuxièmement, la garantie de protections des données utilisateurs, en effet nos services étant hébergés en Europe, ils respectent le RGPD. Et enfin, la sécurité est garantie à nos utilisateurs par des algorithmes de chiffrement qui chiffrent de bout en bout le trafic, les mots de passe sont conservés dans la base de données sous leur forme hashée et les requêtes utilisent le protocole HTTPS.

2.1.2 Présentation de l'interface

Versine se présentera comme une application aux interactions simples. L'applications se divisera en des espaces propres aux différentes fonctionnalités de l'application.
Une page d'accueil avec un fil de posts, où l'utilisateur pour scroller et interagir avec le contenu.
Un espace où il pourra écrire et partager du contenu au sein de ses cercles.
Un espace pour gérer ces derniers, dédié à la gestion des groupes qui composent son entourage dans Versine.
Une zone pour voir ses demandes d'amis et pouvoir se mettre en relation avec de nouveaux utilisateurs de Versine.
Une partie messagerie où il pourra converser avec ses amis de manière sécurisée.
Une zone avec une barre de recherche pour découvrir de nouveaux profils ou juste voir le profil de ses amis.
Pour finir, une partie pour personnaliser son profil utilisateur.

2.1.3 Fonctionnalités

Invitation

Versine est une application dite "Invite Only". C'est une caractéristique notable du projet puisqu'elle définit un système de propagation des nouveaux utilisateurs différents des systèmes traditionnels. En effet, le système "Invite Only" consiste à uniquement accepter les nouveaux utilisateurs ayant été invités par un utilisateur de Versine grâce à un code d'invitation. Chaque utilisateur à la création de son compte génère un certain nombre de tickets d'invitation à usage unique qui lui permettra d'inviter ses proches sur Versine. Au moment de la création d'un compte, un ticket valide sera demandé pour pouvoir s'enregistrer. Ce système qui à première vue peut sembler désavantageux possède de nombreux avantages. Il permet premièrement de s'assurer que l'utilisateur aura des pairs avec qui converser et partager sur Versine. De

plus, il renforce l'engagement de l'utilisateur en créant un sentiment d'appartenance à un cercle restreint des personnes ayant accès à Versine.

Circles

Nous avons voulu créer une fonctionnalité innovante : les Circles, qui sont des groupes de diffusion. L'utilisateur crée lui-même ses Circles, qui correspondent à des cercles sociaux (amis proches, cercle familial, cercle professionnel) mais qui peuvent aussi correspondre à une passion, un hobby, par exemple, si Alice est passionnée par la pétanque, mais qu'elle ne souhaite pas que tout le monde le sache, elle peut réserver ses posts à propos de ce magnifique loisir à ses amis qui pratiquent la pétanque. L'utilisateur peut entièrement personnaliser les circles, en créer de nouveaux, ajouter ou supprimer des utilisateurs... Les amis de l'utilisateur ne savent pas dans quel cercle ils ont été placés (il serait maladroit d'exposer à nos connaissances comment ils sont catégorisés). Cette fonctionnalité permet de garder le même compte pour ses activités professionnelles, le contact avec la famille ou les amis, car les publications vues par les membres sont complètement différentes en fonction du circle dans lequel ils sont placés. De plus, l'aspect invite only renforce la fonctionnalité des circles, car un utilisateur en invitant un autre a de grandes chances d'avoir des passions en commun avec l'autre utilisateur.

Posts/IA de recommandation

Versine prend en charge plusieurs formes de média, du texte, des photos et même des vidéos. Pour chaque utilisateur, une timeline est générée, elle comprend tous les posts des amis de l'utilisateur et dont il fait partie des circles. La timeline est donc une liste de posts qui s'afficheront sur le client Versine. L'ordre des posts dans la timeline est gérée par une IA de recommandation. Elle prendra en compte l'appréciation du post et sa date

de publication.

2.2 Présentation du groupe KISS

Nous sommes 4 élèves qui ne se connaissaient pas avant d'entrer à EPITA, le projet Versine est notre premier projet de groupe de si grande ampleur, après 6 mois à travailler ensemble sur ce projet nous nous connaissons bien mieux et formons une bonne équipe. Nous avons beaucoup appris en travaillant sur ce projet et c'est une première étape vers la concrétisation de nos idées dans le futur. Malgré le départ de deux honorables membres de l'école, ils ont vaillamment continué à participer activement au développement de ce projet.

2.2.1 Maxime Trimboli

Comme beaucoup d'autres, j'ai réalisé au lycée quelques projets de site web et de jeux-vidéos, mais Versine est ma première expérience de travail en groupe pour mener un projet si abouti. Utilisateur quotidien de plusieurs réseaux sociaux, tels que Reddit ou Twitter, j'étais amusé à l'idée de créer un réseau social ayant pour but de virtuellement faire concurrence à ces plateformes. Connaissant ces plateformes, je savais à quel point les données privées étaient parfois exposées et comment les algorithmes de recommandation mettent parfois en avant des contenus malfaisants.
J'avais moins d'expérience que mes trois autres camarades en programmation et ce projet de réseau social m'a permis de m'essayer à de nombreuses technologies différentes, et j'ai pu apprendre de nouvelles compétences en programmation et apprendre à m'organiser en groupe.

2.2.2 Mickael Razzouk

Je suis un élève en première année à EPITA, j'ai adoré ce projet, car il m'a permis d'approfondir mes connaissances sur deux points, le premier étant la mise en place d'un système comme celui que représente Versine en production avec des outils comme Docker, nginx et Cloudflare. Le second étant l'utilisation concrète des programmes réalisés en C en dehors du terminal, aujourd'hui Versine est un ensemble de plus de 10 conteneurs, 5 micros services, 3 réseaux, 2 serveurs VPS et dédiés et tout cela en UNE API fonctionnelle et je suis fier de pouvoir dire que j'ai participé à cela.

2.2.3 Rayan Mazouz

Connu pour ma ponctualité hors du commun et mon sérieux extrême, je suis un ancien élève a EPITA sorti d'une Terminale options maths, physique et ISN. La vie privée est un sujet important pour moi, et c'est pourquoi ce projet me touche. Passionné d'informatique, j'ai commencé à programmer il y a 6 ans, avec du Basic puis du Java pour la création de mods Minecraft, de logiciels de bureau et d'applications Android, j'ai ensuite appris le C# pour coder sur Unity, et plus récemment, j'ai appris le langage Rust. en classe de NTS j'ai beaucoup codé en python, puis j'ai continué avec ce langage dans des projets personnels. J'ai donc très peu d'expérience en développement web, et ce projet me permettra de changer cela ; mon principal atout est en effet mon enthousiasme d'apprendre de nouvelles technologies, et mon plus grand défaut est mon manque de ponctualité

2.2.4 Yann Thévenin

Étudiant en S2 à Epita et développeur passionné, j'ai pu, grâce à la réalisation de nombreux projets, développer des compétences en termes de gestion de projet informatique. Je pense subséquemment être capable d'endosser le rôle de chef de projet. Ces expériences m'ont permis de travailler aussi bien sur les aspects techniques ou graphiques de la création d'un projet informatique, deux aspects que j'affectionne tout particulièrement. Ce projet soulève des challenges, des problématiques et des dimensions créatives que j'ai hâte d'explorer avec mon équipe !

Réalisation du projet

3.1 Rappels

3.1.1 Architecture

Afin d'organiser notre API du côté backend, nous avons choisi d'utiliser une architecture dite en "micro-services". L'architecture en micro-services propose de nombreux avantages qui nous l'ont fait adopter. L'architecture en micro-Services permet de décentraliser les différents endpoints de notre serveur sur des processus différents, ce qui nous met en capacité de déployer indépendamment chacune de ces dernières sur un serveur différent, et surtout de limiter un cas de panne générale en cas de crash d'un processus. En effet, si un processus vient à se suspendre pour différentes raisons, seule la partie du projet concernée ne sera plus accessible, mais laisse tourner les autres processus sans impacter l'intégralité du service. Cela nous permet grâce à Docker de multiplier ces 2 instances afin de pour réagir plus rapidement en cas de panne et permettre une meilleure disponibilité du service ainsi qu'une capacité à allouer plus de performances à une partie d'un projet (plus gourmande en ressources par exemple).
L'architecture en micro-services ne propose pas seulement des avantages techniques, mais aussi une meilleure organisation du code, en effet, chacun des différents micro-services du projet auraient leur repository associé, rendant

le travail sur une partie du projet plus facile et rapide d'accès. Cela a pour incidence une meilleure productivité et un traitement de plus petites parties de code.
Ceci permet de mieux répartir les tâches, et de se concentrer sur l'essentiel, contrairement à une architecture monolithique.
Toutefois, cette architecture nécessite une mise en place plus rigoureuse, en effet, la mise en place de cette architecture est plus longue, mais une fois cet effort réalisé, le projet peut avancer plus efficacement et rapidement grâce à de plus petites portions de code traitées pour chaque endpoint. Le langage qui sera principalement utilisé pour la réalisation de cette partie du projet sera le C#.

3.1.2 Termes employés

Micro-service

Un micro service est un service hébergé à part qui effectue une tâche bien précise, Dans le cadre de l'architecture par micro service que nous avons choisi d'implémenter, chaque service est hébergé sur un serveur virtuellement différent et écoute en permanence les demandes des autres micro services, ensemble ils permettent d'assurer les fonctionnalités de l'application.

Gateway

Le gateway reçois les requêtes envoyées par les utilisateurs et les redirige vers le micro service cible pour qu'elle soit traiter. Nous voulions coder notre propre gateway mais pour des raisons de limite de temps, nous avons décidé d'utiliser krakend. exemple : https ://api.versine.fr/users/health le gateway parse les informations dans le lien de la requête détermine que la destination de la requête est le micro service users, la requête est donc transmise au conteneur users sur son port d'écoute.

Endpoint

Un endpoint est la destination d'une requête dans un micro service, chaque endpoint a des fonctionnalités d'afférentes et permet de réaliser d'efférentes taches. exemple : L'endpoint /health du micro-service door donne l'état de santé de door
L'endpoint /login de door permet d'authentifier un utilisateur

CDN

L'acronyme CDN veux dire "content delivery network" c'est le réseau chargé de stocker, supprimer et distribuer les médias (images, vidéos). Nous avions prévu d'utiliser pictshare un outil open-source par soucis d'optimisation, mais l'outil s'est avéré complexe à utiliser, c'est pour cela que nous avons créé notre propre version simplifiée d'un CDN.

Requête

Nous utilisons des requêtes HTTPS pour "Hypertext Transfer Protocol Secure" Elle est composée de l'URL et du body, l'URL api.versine.fr/door/login est composée du sous domaine "api", du domaine "versine.fr", et de l'endpoint "/door/login". Le body est un JSON qui contient les informations que transporte la requête.

3.2 Technologies utilisées

3.2.1 Electron.NET

Nous avons décidé de faire un réseau social, mais pour rester dans les temps, nous avons utilisé une technologie qui nous a paru adaptée. Electron permet de coder des logiciels cross-platform (GNU/Linux, macOS, Windows) tout en utilisant des technologies du web. Electron fait tourner le contenu du logiciel comme une page web en faisant tourner une instance minimale Chromium. Nous pouvions donc utiliser un framework de web en C (Asp.NET) pour créer le logiciel comme une page web. Cela nous a permis de simplifier le développement. D'autant plus qu'il nous a suffi de modifier les règles de lancement de notre application pour la mettre en ligne rapidement dans un web client. Electron nous a permis d'avancer rapidement sans trop nous dépayser puisque certains d'entre nous avaient déjà eu l'occasion de développer des sites web.

3.2.2 Insomnia

Nous avons commencé le projet par le back-end, nous n'avions donc aucune interface sous la main pour pouvoir tester notre projet, nous n'avions que le squelette de l'application sans le reste des organes pour lui donner vie. Cependant, c'était une partie essentielle, nous avons donc commencé par le back-end. Nous avons utilisé Insomnia pour travailler sur le backend sans nous soucier de ne pas encore avoir d'interface. En effet, ce logiciel nous permettait de designer à la main les requêtes que nous aurions effectuées dans le client s'il était présent. Il nous a suivis tout au long du projet, il nous a même suivis pour certaines parties du client. C'est un outil gratuit et open-source qui est utilisé par beaucoup de développeurs aujourd'hui. Sans ça, Versine n'aurait pas pu voir le jour.

3.2.3 Docker

Une infrastructure en micro service comme la nôtre va de pair avec de la conteneurisation, pour faire cela nous avons décidé d'utiliser docker en raison du support disponible en ligne, mais également d'un grand nombre de conteneurs existant sur le docker hub. Cet outil nous permet de déployer dans des conteneurs isoler chaque micro-service tout en les connectant à un réseau virtuel en commun pour leur permettre de communiquer. Nous pouvons également compiler et déployer en un clic la dernière version de nos micro-services grâce à des Dockerfile qui automatise de processus de compilation et de réinstallation.

3.3 Chronologie du travail efféctué

3.3.1 Maxime Trimboli

Soutenance 1

Site Web : Au début du projet, je ne me sentais pas assez familier avec l'architecture de micro-services, de plus, Mickaël mettait en place le serveur et Yann et Rayan étaient en train de développer les outils pour gérer les bases de données. C'est donc tout naturellement que je me suis orienté sur la partie design de l'application et de réflexion sur les fonctionnalités. J'avais donc réfléchi avec Yann à des éléments de design et puis j'ai commencé à réaliser le prototype du site internet qui allait nous permettre de mettre en avant le réseau social Versine.

Front-End : Avec Yann, nous avions développé un prototype de design et nous avions commencé à développer le client, nous avions mis en place une page d'accueil qui permettait de s'inscrire ou de se connecter à Versine grâce aux end-points registre et login.

Post : Avec l'aide de Rayan, j'avais développé l'endpoint addPost du micro-service post, qui permettait de mettre en ligne un post. Cela m'a permis de commencer à me familiariser avec le back-end et la façon dont fonctionnent les endpoints, les requêtes et les bases de données.

Soutenance 2

Site Web : J'avais rectifié des détails esthétiques du site web, ainsi que rajouté une page de téléchargement d'une première version du client desktop.

micro-service Timeline : Complètement galvanisé par la découverte que j'avais faite en travaillant sur le micro-service post, je me suis attaqué au

développement du micro-service Timeline. J'avais développé l'endpoint "getTimeline", il permettait de renvoyer la timeline, c'est-à-dire l'ensemble des posts qui avaient étés postés par les amis de l'user et que l'User est habilité à voir.

Soutenance Finale

Site Web : Notre site web sera la façade de Versine, et doit donner envie aux invités de rejoindre et de faire partie de cette incroyable communauté qu'est Versine. Nous avons donc totalement refondu le site. Le site et l'hébergement des différentes pages a été réalisé par Mickaël sur son serveur. Le site comportait la page d'accueil et la page de téléchargement, celles-ci ont été changées pour les rendre visuellement plus plaisantes. Une page distincte de présentation du groupe KISS qui rappelle l'avancement du projet sous la forme d'un carnet de bord, on y trouvera aussi le lien GitHub du projet Versine. Une page de sources par rapport aux différentes technologies et bibliothèques que nous avons pu utiliser. La page de téléchargement permet de télécharger l'exécutable d'installation (Windows) de Versine, mais elle permet aussi d'accéder au web client de Versine. La procédure d'installation est également renseignée sur cette page.

micro-service Circles et Post : J'ai commencé par rajouter l'endpoint getPostsfromFriend, cet endpoint permet de répondre à une demande de Yann, en effet quand on recherche un utilisateur, on doit pouvoir voir ses posts que si on est son ami sur Versine et ne voir que les posts qu'il nous a autorisé à voir. Cet endpoint va donc retourner tous les posts de l'ami que l'utilisateur a le droit de voir. J'ai aussi dû changer une méthode de la classe Timeline car l'objet Timeline n'était pas sérialisé correctement par la méthode de base. De plus, j'ai dû modifier le micro-service Post pour pouvoir développer un algorithme de recommandation.

Algorithme de recomandation : Comme la classe Post a été modifiée pour avoir la date de publication du post et pouvoir avoir un score d'appréciation, on va pouvoir générer un score pour chaque post. À partir de ce score, on va trier la liste de posts et ainsi retourner la timeline la plus pertinente pour l'utilisateur.

3.3.2 Mickael Razzouk

Soutenance 1

Mise en place du serveur : Dans un premier temps, il a fallu installer le hardware sur le lieu d'hébergement, ensuite installer l'os, nous avons choisi Ubuntu server LTS. puis configurer le routeur et le par-feu pour permettre l'administration du système à distance. Pour la connexion à distance, nous utilisons OpenSSH avec une authentification par clé RSA.

Préparation de l'environnement de production : Il s'agit de préparer l'environnement à accueillir une production, pour cela il a fallu télécharger des outils d'administration comme portainer qui permet de gérer les conteneurs et réseaux docker avec une interface graphique. Un docker file et une crontab nous permettent de mettre à jour l'instance d'un micro service lors d'un push sur la main branch git.

Hébergement du site : Un simple conteneur nginx sert à héberger le site web versine.fr. Nous utilisons Cloudflare pour gérer les DNS en raison de leur offre de qualité assurant une grande vitesse et fournissant des proxys qui protègent l'IP du serveur.

Soutenance 2

Micro service profile : Le micro service "profile" sert à récupérer les informations sur le profil utilisateur ciblé depuis la base de données. Pour

implémenter ce micro service, j'ai utilisé la librairie EasyMango codée par Rayan qui permet de manipuler facilement une base de données, ainsi que le module Newtonsoft.json qui permet d'interagir facilement avec une requête web. Le fonctionnement du service est le suivant : Pour beaucoup de raisons différentes, certains services seront amenés à avoir besoin d'informations d'un utilisateur (Afficher les informations à un utilisateur/vérifier qu'un compte avec ce nom n'existe pas déjà/rechercher un utilisateur sur le réseau/ lancer une discussion priver entre 2 utilisateurs, etc.). C'est à ce moment qu'une requête avec la méthode GET contenant des informations sur la cible de la recherche et l'utilisateur qui l'effectue sera envoyée au micro service profile la classe profile vas ensuite vérifier la validité de la requête, vérifier que l'utilisateur a les permissions de faire cette requête et enfin recherche une correspondance dans la base de donnée crée la réponse et la renvoyer au micro service qui en a besoin.

Micro service users Le profil d'un utilisateur sera amené à changer au cours de son utilisation de Versine, ce service est là pour effectuer ces modifications directement dans la base de donnée, voire une suppression. La requête reçue contient donc le token de l'utilisateur cible qui permet de l'authentifier pour vérifier que la demande de modification est bien fait pas le propriétaire du compte. Elle contient également les modifications à effectuer, le tout dans le body. Le micro service fera donc toutes les modifications demander dans la Base de donnée et renverront un différend message en fonction du déroulement de l'opération (réussite/erreur d'authentification/etc). Mais cette implémentation pose un problème quand nous changeons le nom d'utilisateur qui sert aussi de salt rendant impossible de se connecter par lac suite, ce problème sera résolu pour la soutenance finale.

Soutenance Finale

Outils de Monitoring : Le nombre important d'endpoints rend impossible de tout surveiller manuellement, si un micro service plante la veille de la soutenance il serait problématique que personne ne s'en rende compte, pour palier à cela je me suis inspiré du CRI en implémentant un écran de status qui suit chaque micro service et vérifie qu'il est en ligne, si ce n'est pas le cas des notifications sont envoyées aux membres du groupe. L'outil utilisé est Gatus, pour lui permettre de faire son travail, j'ai dû rajouter un endpoint /health dans chaque micro-service. Bien entendu, pour que Gatus soit utile même en cas de panne du serveur, il doit être hébergé sur un autre serveur, nous utilisons le serveur de Rayan pour faire cela. Gatus fait toutes les 30 secondes une requête de type GET sur l'endpoint /health de chaque micro-service, ex : https ://api.versine.fr/users/health ou /circles/health. Les résultats sont organisés sous forme de graphe disponible sur https ://status.versine.fr.

Déploiement final en production : Pour mettre en place l'API de Versine nous avons utilisé Krakend un gateway qui redirige les requêtes sur le bon micro-service, ex : https ://api.versine.fr/users/xxxxx est redirigé sur l'IP dans le réseau local correspondant au micro service users sur le port 8000 (convention décidée par le groupe au début du projet). Un déploiement se passe rarement sans difficultés, et celui-ci ne fut pas une exception, il a fallu débeuger pendant de nombreuses heures les requêtes qui n'arrivaient pas à destination ou les micro-services qui ne répondaient pas. Une fois cette tache finie, l'API était opérationnelle avec tout ses endpoint (cf. Documentation/API).

Micro Service Posts : Le micro service post n'étant pas terminé, il a fallu implémenter les endpoints manquants et le lien entre le micro-service et le CDN. L'endpoint addPost était à revoir, il n'était plus adapté après les modifications sur d'autres micro services, cet endpoint doit permettre

d'ajouter, supprimer et modifier les posts des utilisateurs. Pour ajouter un post, le micro service envoie le média au CDN et attend en réponse le path du lieu où le média est stocké, le path du média sera donc enregistrer dans la base de donne avec les informations sur le post comme son ID, message associer, propriétaire du post, date et heure, etc. Pour retirer un post, le micro-service demande au CDN la suppression du média et supprime la référence au post dans la base de donnée. La modification d'un post correspond à une simple modification des informations dans la base des donnes via la librairie de Rayan EasyMango seul le cercle de diffusion pourra être modifié.

3.3.3 Rayan Mazouz

Soutenance 1

Web token : J'ai codé une classe permettant de gérer les tokens utilisateurs (WebToken.cs). Un token est une clé utilisée comme mot de passe après la première connexion de l'utilisateur sur un appareil, afin qu'il n'ait pas à retaper son mot de passe à chaque fois.

La classe Webtoken.cs utilise le format JSON Web Token, qui est constitué de 3 parties :

-Le Header qui informe de l'algorithme de chiffrement, ainsi que du type de token. le Header est encodé en base 64

-Le Payload qui contient l'id de l'utilisateur et la date d'expiration du token. le Payload est encodé en base 64

-La Signature qui permet de vérifier que le token n'a pas été modifié. Celle-ci encode le header et le payload en sha256 avec une clé privée gardée sur le serveur. Cela permet de vérifier que le token n'a pas été modifié, car s'il l'a

été, la signature n'y correspondra plus, à moins que l'utilisateur aie connaissance de la clé privée du serveur. la Signature est encodée en base 64

Interaction base de données : J'ai aussi codé librairie qui facilite les interactions avec nos bases de données.
Je l'ai appelée EasyMango, et l'ai publiée sur le package manager de C# (NuGet).

Voici un exemple de fonction venant de cette librairie :

Door : J'ai implémenté le micro-service door avec Yann. Le micro-service door est le micro-service qui se charge de la connexion et de l'ajout de nouveaux utilisateurs J'ai aussi travaillé sur le micro-service Users, qui gère les utilisateurs

Soutenance 2

Micro service post : le micro-service post permet aux utilisateurs de poster des posts texte et images, ainsi que de supprimer leurs posts. Le stockage d'images et vidéos ne se fait pour l'instant pas sur notre serveur, mais sur un site de stockage d'images et de vidéos appelle imgur.com. L'API de Imgur est donc utilisée. Une API ou interface de programmation d'application est une librairie permettant de communiquer avec d'autres services (en l'occurrence Imgur).

Micro service circles : les circles permettent aux utilisateurs de poster des posts pour un groupe d'amis, ou pour l'intégralité de leurs amis. Le micro service circles permet donc de créer, modifier et supprimer des cirlces, afin que ceux-ci soient utilisés par d'autres micro-services tels que le micro-service posts ou le micro-service timeline.

Soutenance Finale

Review du code : notre code comprenais maintes duplications de code, et de classes, ce qui rendait compliqué la maintenance ; nous devions en effet modifier le code a plusieurs endroits, j'ai donc remplacé des classes dupliquées par des librairies.

Cdn : J'ai implémenté le CDN, ou content delivery network, qui est concrètement un serveur de stockage de fichiers. le CDN vérifie que les images sont valides, les compresse, puis les convertit sous format WebP, un format d'images adapté pour le web. Le CDN est aussi capable de supprimer et de remplacer des images. Les images sont stockées dans un dossier sur notre serveur (hosté par Mickael) et le nom de fichier des images sont stockés sur les bases de données afin d'y accéder.

3.3.4 Yann Thévenin

Soutenance 1

En tant que chef de projet, la première soutenance m'a permis de mettre en place l'organisation du groupe, les différents outils que nous allions utiliser pour communiquer, répartir les tâches et comment travailler avec GitHub.
J'ai pu chercher les différents outils que nous allions utiliser pour que tout le monde puisse aisément travailler sur le projet (logiciels, frameworks, etc.).
J'ai mis en place les bases des micro-services qui seront réutilisées dans tout Versine et j'ai travaillé sur le micro-service Door, responsable d'authentifier les utilisateurs.
C'était un moment important du projet, car il fallait donner les directions à suivre pour toute la suite du projet, c'est pour ça que j'y ai donné grande attention afin d'aboutir sur un résultat stable.

Soutenance 2

Tout le monde a vite commencé à prendre la main sur le projet et une fois l'organisation en place, le projet avançait sans trop de problèmes, la méthode fonctionnait bien.

Dans la continuité de door, pour finaliser la porte d'entrée de Versine, j'ai implémenté le système de tickets dans le micro-service door.
Cette partie de door permet aux utilisateurs de ne s'inscrire seulement s'ils possèdent une invitation pour rejoindre Versine.
Une difficulté que j'ai pu rencontrer était de créer un compte pour faire mes tests puisque pour créer un compte il fallait une invitation, cependant la base de données était vide. J'ai donc dû insérer manuellement un premier utilisateur, capable de lancer la chaîne des invitations.
J'ai travaillé sur le micro-service Posts pour rendre les utilisateurs capables de poster sur Versine. La première Version ne supportait qu'uniquement le partage de texte, car nous n'avions pas encore implémenté la solution pour stocker les images.

J'ai commencé à travailler sur le Client Desktop, l'application de bureau que les utilisateurs auront en main pour interagir avec Versine. Au départ, je ne me suis contenté que de relier le backend au Client, avec une interface rudimentaire sans design. Il s'agissait de vérifier comment marchaient les différents endpoints en communication avec le front-end. Nous avions pensé le backend sans avoir le front-end sous la main, donc il s'est avéré que certains micro-services devaient subir des modifications pour mieux s'intégrer au front-end. Nous avons donc dû changer des parties de code dans quelque micro-services.

J'ai pu implémenter le système d'authentification connecté au micro-service door. Il fallait connecter les champs de texte à une fonction qui était amenée à faire une requête HTTPS vers notre api en ligne, et en fonction de la réponse, autoriser l'authentification ou non. Une fois l'utilisateur authentifié, l'application bureau gardait en mémoire locale son token pour authentifier ses prochaines actions avec l'api.
J'ai affiché le profil de l'utilisateur grâce au micro-service user. J'ai utilisé l'endpoint user pour donner des informations sur un utilisateur en renseignant mon token et la personne recherchée.
J'ai fait une interface qui permet d'écrire des posts et de les publier, connectée au micro-service posts. Une fonction prend le contenu du champ de texte et fait une requête HTTPS en renseignant le token sur l'endpoint de publication.

Soutenance Finale

J'ai travaillé sur le client desktop, le site web, effectué des retouches mineures sur quelques micro-services et ai mis en place le websocket server.

Client Desktop : J'ai terminé le Client Desktop, il s'agissait d'implémenter les dernières fonctionnalités et leur(s) micro-service(s).
J'ai pu coder le système de demande d'amis, avec une boîte aux lettres pour voir les demandes d'amis qu'on a reçus et les accepter ou non. Cette partie était connectée au micro-service users. Il s'agissait de récupérer les informations de l'utilisateur pour récupérer ses requêtes d'amies entrantes, et les associer à des utilisateurs. En acceptant ces dernières, une requête était envoyée au micro-service user. Une fois que deux utilisateurs se demandent mutuellement à être amis, leur relation était stockée dans la base de données. Ce qui leur permet de voir leurs posts ou encore échanger des messages
J'ai mis en place la timeline sur la page d'accueil pour voir les posts de nos amis qui était connectée au micro-service timeline. En faisant une requête sur l'endpoint getTimeline, je pouvais, en renseignant mon token, voir tous

les posts des personnes avec qui je suis en relation. Je devais générer dynamiquement le contenu de la page avec l'array de posts que j'avais en réponse de l'api.
J'ai mis en place une barre de recherche pour rechercher des profils d'autres utilisateurs. Une fois la recherche effectuée, si l'utilisateur entré existe, on peut y voir son profil, sinon, une page affichant "User no found" apparaît.
J'ai créé des pages de profils pour les utilisateurs, affichant leurs photos de profil, leurs bannières, leurs pseudos, leurs bios ainsi que la couleur qu'ils auront choisis. Tout cela était lié au micro-service users. En effectuant une requête sur l'endpoint user, je pouvais récupérer les informations publiques d'un utilisateur, à partir de cela, je n'avais qu'à remplir les éléments vides de la page de profil de manière dynamique avec les informations de l'utilisateur.
J'ai créé une interface pour avoir des informations personnelles sur son compte ainsi que de pouvoir éditer son compte. On peut notamment y voir son ticket d'invitation pour inviter des proches et son nombre de tickets restants. Cette partie de Versine utilise le micro-service users, cette fois-ci avec l'endpoint profile, pour afficher des informations confidentielles sur son profil. Il y avait également l'endpoint editUser qui modifiait l'utilisateur en fonction des changements qu'il aurait renseignés sur la page de modification de profil.
J'ai codé l'interface qui permet de gérer ses circles, y ajouter des amis ou en retirer pour intégrer une des fonctionnalités essentielles de Versine. Les Circles permettent de trier son audience. Ils permettent de rendre le contenu que vous postez uniquement accessible par des cercles précis que vous créez. J'ai donc utilisé le micro-service circles et users. Il fallait d'abord récupérer la liste des amis de l'utilisateur. Il fallait aussi récupérer la liste des circles déjà existante depuis le micro-service circles. Cette partie était intéressante puisqu'elle reposait sur beaucoup de requêtes à travers l'api.
J'ai rajouté la possibilité dans la page de publication de sélectionner des circles pour partager un post.
J'ai rajouté une partie de messagerie instantanée pour discuter avec d'autres utilisateurs de façon privée. Cette partie a fait recours aux websockets pour

permettre de communiquer instantanément sans effectuer trop de requêtes sur le serveur.
J'ai fait une refonte du design pour rendre l'application agréable à l'utilisation.

Un des points forts de notre application est sa polyvalence en terme d'accessibilité, grâce à electron.net, nous pouvons déployer notre application directement depuis notre site web ou encore la télécharger sous forme d'application bureau.

Site Web : J'ai fait une refonte du design du site web pour apporter de la cohérence entre l'application desktop et le site web.

Retouches sur les micro-services : Étant donné que j'ai été amené à travailler sur beaucoup d'implémentations dans le client desktop, j'ai été amené à changer des micro-services, voire implémenter de nouvelles fonctionnalités pour palier à des manquements dans le back-end. Cela relevait de modifications mineures, mais elles étaient nécessaires pour atteindre le bon résultat.

Serveur Websocket : Le serveur Websocket permet à la messagerie instantanée de fonctionner. La plupart des micro-services sont des serveurs http, mais celui-là utilise un autre protocole pour communiquer en temps réel. Au lieu de gérer des requêtes et des réponses, il crée un tunnel de communication pour maintenir une communication avec les clients.
J'ai donc dû mettre en place le serveur qui allait être chargé de relayer des messages entre utilisateurs.
Il fallait d'abord identifier les clients auprès du serveur websocket pour les mettre en position d'écoute et d'émission de messages de façon sécurisée.
Une autre difficulté était d'isoler les messages pour ne pas que tous les utili-

sateurs de Versine reçoivent des événements de messages qui ne leur sont pas adressés. Il a fallu alors créer un système de rooms pour mettre en relation les utilisateurs de façon isolé.
J'ai pu implémenter cela grâce à une librairie que j'ai trouvé en ligne qui facilite grandement la façon de gérer les communications (https ://github.com/sta/websocket-sharp).

3.4 Reprise du cahier des charges

3.4.1 Répartitions des tâches

Tâches	Yann THEVENIN	Rayan MAZOUZ	Maxime TRIMBOLI	Mickael RAZZOUK
Gateways	••			•
Design	•		••	
Site web			••	•
Micro Services	•	••	•	•
Conteneurs				••
Front end	••	•		
Serveur		•	•	••

Légende :

•• = Responsable de la tâche • = Travaille sur la tâche

Nous avions prévu des responsables pour chaque tâche et une répartition équilibrée de chacun sur différentes parties du projet. Au final, tout en gardant une bonne organisation, tout le monde a pu toucher à des parties plus larges du projet, car nous y avons été naturellement amenés puisque beaucoup de parties sont reliées entre elles. Finalement, cette organisation nous a permis au début de se cadrer chacun sur une partie et ses objectifs. Pour finir, nous nous sommes un peu affranchis de ces rôles, mais nous pensons que c'était pour le mieux.

3.4.2 Avancement

Les objectifs que nous nous étions fixés dans le cahier des charges :

Tâches	$1^{ère}$ soutenance	$2^{ème}$ soutenance	$3^{ème}$ soutenance
Gateways	•	••	•••
Design	••	••	•••
Site web	••	••	•••
Micro Services	•	••	•••
Conteneurs	•	••	•••
Front end		•	•••
Serveur	••	•••	•••

Les objectifs que nous avons atteints :

Tâches	$1^{ère}$ soutenance	$2^{ème}$ soutenance	$3^{ème}$ soutenance
Gateways	•••	•••	•••
Design	••	••	•••
Site web	••	••	•••
Micro Services	•	••	•••
Conteneurs	•••	•••	•••
Front end		•	•••
Serveur	•••	•••	•••

Légende :

- • = Tâche commencée
- •• = Tâche avancée
- ••• = Tâche terminée

1ère soutenance

Comme nous l'avions dit lors de la 1ère soutenance, nous avons finalement utilisés un gateway open source. Même si lors de cette soutenance il a pu sembler que nous n'avions pas assez de code, nous en avions déjà fait assez, car il fallait développer les outils pour gérer l'architecture de micro-services. Le serveur et les conteneurs étaient complètement prêts. Une ébauche de site web et de front-end avait aussi été développée.

2ème soutenance

Lors de la deuxième soutenance, nos micro-services étaient déjà tous au complet et fonctionnels, et nous avions présenté un prototype du client desktop. Le site web quant à lui présentait un bouton de téléchargement du projet.

Soutenance finale

Pour cette dernière soutenance, nous allons présenter la version finale de Versine. Le projet comporte le site web qui comprend les informations sur le projet et le groupe, et surtout l'accès au téléchargement de l'application Versine ou l'accès à la Webapp. Versine comporte toutes les fonctionnalités que nous avions initialement prévues et le projet s'est aussi enrichi de nouvelles fonctionnalités. Nous sommes aussi très satisfaits du rendu esthétique du projet et des aspects de design du projet.

Beta test

4.1 Mise en place

Nous avons lancé une bêta test de Versine, en utilisant des bases de données temporaires qui vont être supprimés au lancement de Versine. Les bêta-testeurs sont des connaissances de la famille et des amis, nous avons essayé d'avoir des profils varier pour avoir un échantillon représentatif de la population.

4.2 Ressenti

La plupart des utilisateurs de Versine étaient réticents à installer le client Desktop, et ont préféré utiliser la version Web. Les utilisateurs ont été impressionnés par le design de Versine, ainsi que la facilité d'installation et de désinstallation du client Desktop. Peu d'utilisateurs ont utilisé la fonctionnalité de chat

4.3 Bugs et améliorations proposées

Nous avons mis en place une plateforme de report de bug sur https://bugs.versine.fr ou nos utilisateurs peuvent reporter des bugs ou proposer des fonctionnalités pour nous aider à améliorer Versine.

Récit de la réalisation

5.1 Ressenti individuel

5.1.1 Maxime

J'étais intimidé au début du projet par l'idée de développer un réseau social. En effet, je n'avais aucune expérience en développement web et je me sentais moins à l'aise que les autres membres du groupe en programmation. Cependant, au cours de ce projet, j'ai pu apprendre de mes camarades, qui m'ont beaucoup aidé à prendre en main les outils que nous utilisions. Déjà au niveau de la gestion du travail collaboratif, j'ai dû apprendre à bien gérer git et à ne pas faire n'importe quoi avec les merge. J'ai aussi appris sur le web, les serveurs et la gestion des requêtes. Cependant, il y a hic, car dans la vie rien n'est parfait. En effet, j'utilise Windows, et j'ai dû subir nombreux quolibets sur le choix de mon OS, et j'ai dû servir de cobaye pour tester l'installation du client de Versine pour Windows. Sur un ton plus sérieux, j'ai rencontré quelques problèmes de pare-feus avec Windows quand nous testions les micro-services.
Au final, cette expérience m'a permis d'apprendre beaucoup sur les technologies utilisées et sur l'organisation du travail en groupe.

5.1.2 Mickaël

Au début du projet mon rôle restait confiné dans ce que je savais déjà faire, à savoir la gestion du serveur et son infrastructure, puis au fur et à mesure j'ai de plus en plus explorer des aspects que je ne connaissais pas comme la programmation backend qui m'a apporté de nouvelles connaissances, j'ai pu constater l'importance de la communication dans entre le backend et le frontend mais aussi entre micro service, car ces derniers dépendent les uns des autres et un changement de format de requête peux entrainer des dysfonctionnements.

En parlant de dysfonctionnement, en tant que responsable de l'infra, mon rôle était d'assister tout le monde dans les problèmes rencontrer et nous en avons eu de nombreux, tous inattendus, par exemple une semaine avant la première soutenance un arbre tombe à 3 mètre du lieu d'hébergement coupant le câble de fibre rendant le serveur injoignable pour une durée de 5 jours. Un autre exemple, 1 semaine avant la soutenance finale, un arbre brule à 250 mètres du lieu d'hébergement, coupant les lignes électriques, rendant le serveur injoignable pendant 3 heures pour éviter des coupures de service dans le futur, j'envisage de mettre en place un système de redondance des données savoir faire cela serait une compétence intéressante à avoir. Morale de l'histoire : "il faut se méfier des arbres"

5.1.3 Rayan

Versine est mon premier projet web. J'étais donc submergé par l'idée de coder un réseau social, une tache qui me paraissait inatteignable. D'autant plus que je n'avais jamais utilisé de bases de données. Au dépit de ma nouveauté dans le domaine et des difficultés que j'ai rencontrées, participer au développement de Versine m'a permis de réaliser que développer un réseau social n'est pas une tache insurmontable, bien que difficile. Une des difficultés que j'ai rencontrées est l'utilisation d'un content delivery network (pictshare)

qui ne documentait pas de requêtes API en C. après maint essai, j'ai donc décidé, avec le reste du groupe, de développer notre propre CDN. Verisne est un projet qui m'a appris beaucoup de technologies, et, de façon tout aussi importante, Versine m'a appris à travailler en groupe, de manière organisée et à faire confiance à mes camarades.

5.1.4 Yann

Au début du projet, j'appréhendais la complexité de l'architecture en micro-services, j'avais peur que ça impacte plus le projet de façon négative qu'autre chose. J'ai souvent pensé à changer l'architecture pour repasser vers une architecture monolithique. Mais le temps passant, tout le monde s'est rapidement fait à la chose, ils ont su trouver leurs propres réponses et avancer de manière autonome. Finalement, je suis fier de ce que nous avons produit. Tout marche comme prévu. Au début, j'avais peur, je voulais tout superviser, mais avec le temps, j'ai appris à lâcher prise et faire confiance à mes camarades. Aujourd'hui, je peux leur laisser des tâches sans-soucis. Ça m'a beaucoup aidé pour alléger ce à quoi je devais penser pendant le projet pour me concentrer uniquement sur l'essentiel et sur les tâches qui m'étaient données. Le projet avance beaucoup plus vite maintenant, qu'au début du projet, au début, il y avait des problèmes de compréhension peut être, mais maintenant qu'on a appris à se faire confiance pour les tâches du projet, on a vraiment monté en productivité. Le rôle de chef de groupe s'est aussi laissé aller le groupe, se laisser aller et reposer mutuellement sur ses camarades. J'ai beaucoup appris, aussi bien en compétence technique qu'humainement. J'ai pu découvrir de nouvelles technologies qui pourraient m'être utiles dans de futurs projets, et j'ai trouvé des solutions à des problèmes auxquels je pourrais potentiellement être amené à faire face à nouveau. J'ai vraiment apprécié cette expérience de travail de groupe. J'ai aussi appris de certaines erreurs, en trouvant ce qu'on aurait pu mieux faire à certains moments, je pourrais éviter de reproduire les mêmes erreurs dans de futurs projets.

5.2 Conclusion

Lorsque le groupe s'est formé, nous étions sûrs d'une chose, nous ne voulions pas faire de jeu vidéo. Nous avons donc réfléchi à une application à notre portée sur laquelle nous pourrions découvrir de nouvelles technologies : les réseaux sociaux. Nous avons observé les aspects négatifs des réseaux sociaux et réfléchis de manière expérimentale à des façons dont ceux-cis pouvaient être résolus. Grâce à la mise en place de l'environnement de programmation, des outils de travail collaboratifs nous avons vite pu être efficaces et avancer rapidement sur le projet. Nous y avons tous appris des choses et nous nous sommes souvent entraidés sur beaucoup d'aspects. L'ambiance de groupe a toujours été agréable et nous avons pu apprendre tout en nous amusant, et la satisfaction et la fierté d'avoir créé une application au design original et qui offre des fonctionnalités innovantes nous réjouit tous.

Documentation

6.0.1 Domaines et sous-domaines

- https ://versine.fr
  C'est le domaine de Versine qui redirige vers la page d'accueil du site ; le site sert à présenter le projet et notre équipe ainsi qu'à télécharger le client ou accéder à la webapp il y a aussi un bouton vers la page de status.
- https ://app.versine.fr
  C'est la webapp de Versine, en plus du client desktop obligatoire, nous avons mis a disposition de tous une webapp pour utiliser Versine en ligne !
- https ://status.versine.fr
  C'est la page de status du réseau Versine il permet de suivre l'état du réseau et peut aider les administrateurs comme les utilisateurs en cas de problème avec certaines fonctionnalités.
- https ://api.versine.fr
  C'est le cœur de Versine, le lien entre les 5 micro-services en backend et l'interface graphique, toutes les requêtes passent par là. Pour plus de détails, veuillez vous référer à la documentation détailler de l'API
- https ://portainer.versine.fr
  C'est l'interface d'administration de docker durant la phase de développement,

elle est accessible sur ce sous domaine, elle peut être cachée derrière un VPN a tout moment. Cette interface permet de visualiser les réseaux docker et les nombreux conteneurs.

- https ://beugs.versine.fr
  C'est le trackeur de bugs de versine, ou les utilisateurs peuvent signaler des bugs ou proposer des améliorations a Versine.

6.0.2 Hébergement

L'ensemble de ces sites sont auto-hébergés, c'est-à-dire qu'aucune entreprise n'a accès aux données qui sont intégralement hébergées par des membres de Versine. (L'ensemble des sites sont hostés par Mickaël, sauf status.versine.fr qui est hosté par Rayan, afin de pouvoir voir le status des autres sites en cas de coupure d'électricité ou d'internet chez Mickaël)

6.0.3 API

door

- https ://api.versine.fr/door/login
  Méthode : POST
  body : username, password
  Description : Sert à obtenir le token d'un utilisateur avec son mot de passe.
- https ://api.versine.fr/door/register
  Méthode : POST
  body : username, password, ticket
  Description : Sert à créer un compte Versine.
- https ://api.versine.fr/door/health
  Méthode : GET
  body : null
  Description : Sert à obtenir l'état du micro-service.

users

- https ://api.versine.fr/users/user/username
  Méthode : GET
  body : null
  Description : Sert à obtenir les informations publiques d'un utilisateur username.
- https ://api.versine.fr/users/profile
  Méthode : GET
  body : token
  Description : Sert à obtenir les informations publiques et privées du propriétaire du token.
- https ://api.versine.fr/users/deleteUser
  Méthode : POST
  body : username, password,token
  Description : Sert à supprimer le compte d'un utilisateur.
- https ://api.versine.fr/users/editUser
  Méthode : POST
  body : token, modifications
  Description : Sert à modifier le profil du compte lier au token .
- https ://api.versine.fr/users/editUsername
  Méthode : POST
  body : username, password, token
  Description : Sert à modifier le username d'un profil.
- https ://api.versine.fr/users/editPassword
  Méthode : POST
  body : username, password, token, newpassword
  Description : Sert à changer le mot de passe d'un utilisateur.
- https ://api.versine.fr/users/requestFriend
  Méthode : POST
  body : id, token

Description : Sert à envoyer une demande d'ami.

- https ://api.versine.fr/users/deleteRequest
  Méthode : POST
  body : id, token
  Description : Sert à retirer une demande d'ami.
- https ://api.versine.fr/users/health
  Methode : GET
  body : null
  Description : Sert à obtenir l'état du micro-service.

circles

- https ://api.versine.fr/circles/userCircles
  Methode : GET
  body : token
  Description : Sert à obtenir les informations sur les cercles d'un utilisateur.
- https ://api.versine.fr/circles/addToCircle
  Methode : POST
  body : token, circle, id
  Description : Sert à ajouter l'utilisateur "id" au cercle "circle" de l'utilisateur lié au token.
- https ://api.versine.fr/circles/removeFromCircle
  Méthode : POST
  body : token, circle, id
  Description : Sert à retirer l'utilisateur "id" du cercle "circle" de l'utilisateur lié au token.
- https ://api.versine.fr/circles/health
  Méthode : GET
  body : null
  Description : Sert à obtenir l'état du micro-service.

posts

- https ://api.versine.fr/posts/addPost
  Méthode : GET
  body : token, circle, post
  Description : Sert à ajouter un post dans un circle.
- https ://api.versine.fr/posts/removePost Méthode : POST
  body : Not implemented
  Description : Sert à retirer un post de la base de donnée.
- https ://api.versine.fr/posts/health Méthode : GET
  body : null
  Description : Sert à obtenir l'état du micro-service.

timeline

- https ://api.versine.fr/timeline/getTimeline Méthode : GET
  body : token
  Description : Sert à générer une timeline pour l'utilisateur lié au token.
- https ://api.versine.fr/timeline//getPostsfromFriend Méthode : GET
  body : token, id(id du friend)
  Description : Vérifie que l'utilisateur est bien friend et renvoie tous les posts du friend.
- https ://api.versine.fr/timeline/health Méthode : GET
  body : null
  Description : Sert à obtenir l'état du micro-service.