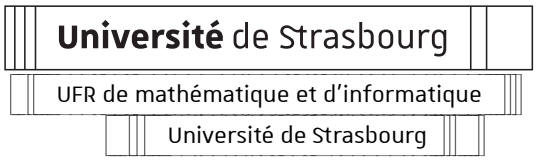

Master de Science, mention Informatique spécialité Science et Ingénierie des Réseaux, de l'Internet et des Systèmes

Mémoire de projet de Master, groupe nř2

Gabriel Poittevin
gabriel.poittevin@etu.unistra.fr

ChaTalK, application de télécommunications sécurisées et centralisées

Projet encadré et commandité par

Julien Montavont montavont@unistra.fr et Cristel Pelsser pelsser@unistra.fr

8 janvier 2020

Contents

List of Figures

List of Tables

Abstract

Ceci est le mémoire de projet du groupe nř2 de projet de Master de la promotion 2018-2020 du Master SIRIS de l'Université de Strasbourg.
L'objectif de ce document est de présenter le projet dans son ensemble, avec ses problématiques, les solutions qui leur ont été apportées et les difficultés rencontrées au cours de sa réalisation.

1 Introduction

1.1 Le projet

L'objectif de ce projet était de répondre à une série de problèmes rencontrés par l'organisation (fictive) Journalistes Sans Papiers. JSP souhaitait un moyen le plus sécurisé possible de permettre à ses membres de communiquer les uns avec les autres. La solution devait également supporter d'importants flux de données, et pouvoir fournir notamment un mode audio voire vidéo en plus du mode texte classique. Enfin, l'organisation souhaitait la possibilité de déployer à la volée le service sur des réseaux isolés, permettant à ses membres d'utiliser la solution de communication dans des situations coupées du réseau Internet et des serveurs centraux de l'organisation.

1.2 La solution

Notre équipe a conçu la solution ChaTalK afin de répondre aux besoin de l'entreprise. Il n'existait pas à notre connaissance de solution permettant les communication entièrement sécurisées de manière centralisée, dont un développement facile pour un néophyte puisse être possible. ChaTalK est une application Web 2.0, dont l'infrastructure repose sur des cluster Kubernetes sur des machines Ubuntu Server, le côté serveur de l'application est un ensemble de services Go et PostgresQL, et le côté client est une interface écrite en React JS pour la partie navigateur et en Kotlin pour l'application mobile.

2 Technique

2.1 Architecture de la solution

2.2 Infrastructure matérielle et logicielle

2.3 Protocoles de communication

2.4 Mesures et protocoles de sécurité

2.5 Base de données

2.6 Services logiciels, application côté serveur

2.7 Interface utilisateur, application côté client

3 Solution

La solution telle que nous l'avons conçue comprenait plusieurs éléments que nous allons rappeler ce-dessous.

Tolérance aux pannes et passage à l'échelle Grâce à son infrasctucture basée sur Kubernetes et à son architecture en micro-services, notre application est aisément capable de passer à l'échelle et de supporter la montée en charge, ainsi que de résister à la perte de machines dans les grappes Kubernetes. Dans les pires cas, les les fonctionnalités passent en mode dégradé. Ce mode dégradé permet la consultation des archives de conversations et donc de conserver une partie des activités des utilisateurs du service.

Sécurité Notre application devait garantir la sécurité des flux de données, en particulier d'un point de vue intégrité et confidentialité. Seuls les envoyeurs et récepteurs des messages devaient avoir accès aux données.

Nous avons partiellement rempli cet objectif, des contraintes de temps liées à nos retards nous ayant forcé à abandonner le chiffrement des flux audiovisuels. L'authentification des utilisateurs a néanmoins été réalisée. Une couche de sécurité minimale basée sur SSL/TLS assure un niveau de sécurité décent, tant que les utilisateurs du service font confiance à l'administrateur des serveurs sur lesquels fonctionne la solution.

Déploiement facile Notre application devait pouvoir être facilement déployée par un néophyte, sur n'importe quelle machine de type Linux, rapidement et sans connexion Internet. Cette tâche était d'une priorité maximale

Temps réel Notre application devait assurer des conversations en temps réel entre les utilisateurs.
Nos tests actuels ne sont pas représentatifs d'une utilisation réelle, mais montrent que la latence des communications textuelles sur notre solution est imperceptible à l'humain pour un petit nombre d'utilisateurs.

Interface utilisateur Notre solution devait proposer une interface utilisateur compatible avec un maximum d'appareils.
Notre application frontale Web est utilisable sur tous les navigateurs modernes sans modification et sans distinction de système d'expoitation. De plus elle s'adapte à des tailles d'écran diverses et à des appareils de puissance variée. Bien que cette interface Web puisse être utilisée en tant qu'application mobile, une application mobile pour Android en Kotlin est également disponible, plus adaptée à l'utilisation sur des téléphones intelligents.
Nous avons choisi une interface utilisateur épurée, en conservant les icônograpies habituelles, afin de permettre une meilleure expérience utilisateur et une plus grande ergonomie.

Modularité Notre solution devait permettre à un individu déployant un serveur de choisir les capacités qu'il souhaitait donner à celui-ci.
À cause des contraintes de temps, et puisque la tâche n'était pas prioritaire, elle a été la première abandonnée totalement. Les fonctionnalités supplémentaires audio et vidéo ont été incluses directement dans l'application d'arrière-plan principale et le concept de modules serveur a été laissé à l'abandon.

Compatibilité et standards Notre application devait être compatible avec un maximum de standards et ne pas utiliser de technologies trop peu orthodoxes.
Puisque l'interface utilisée est une interface Web, les quatre systèmes d'exploitation les plus courants, Microsoft Windows, Apple macOS, Apple iOS et Android, sont couverts tant qu'ils possèdent un navigateur Internet à jour. De plus, les systèmes d'exploitation GNU/Linux et BSD sont également couverts tant qu'ils possèdent un navigateur Internet à jour. Enfin, s'il n'existe pas d'application native pour iOS, il en existe une pour Android.
D'un point de vue des standards, notre application est accessible à des clients utilisant indifféremment de l'IPv4, de l'IPv6 ou les deux.

Internationalisation Notre application se devait de supporter les principaux alphabets du monde et d'être disponible à minima en Anglais.
Puisque tous les textes que nous utilisons sont encodés en UTF-8, les alphabets divers sont gérés dans l'application. L'interface que nous avons développée l'a été en Anglais et est donc utilisable par des utilisateurs anglophones.

4 Gestion de projet

4.1 Organisation et commentaire général du projet

Au départ du projet, nous avions prévu des binômes s'occupant des tâches. Au fil du déroulement du projet, nous nous sommes aperçus que certaines parties du projet avançaient plus vite que d'autres, et que l'idée de binômes a changé pour confier à chacun les tâches dans lesquelles il se sentait à l'aise.

Le suivi du projet était assuré par des réunions régulières avec les clients / professeurs, entre une réunion par semaine et une réunion pour deux semaines selon les périodes. Ces réunions étaient suivies d'un compte-rendu à l'équipe projet, durant lequel le nouveau planning était généralement présenté, et où les tâches à venir étaient réparties entre les membres de l'équipe.

Les membres de l'équipe étaient informés de leurs missions via trois moyens principaux :

- des *issues* GitLab, avec une date, une mission et un responsable
- une notification écrite sur l'application de communication Discord
- une notification orale, généralement à la fin de la réunion de répartition des tâches

Malgré les efforts mis par le chef de projet pour notifier son équipe des missions à réaliser, il est arrivé trop souvent que les issues soient créées tardivement et que la notification orale reste la seule information donnée à l'équipe.

La plus grosse difficulté pour l'organisation de ce projet a été de l'ordre de la communication. Gabriel a souvent été peu réactif à officialiser par écrit les tâches à réaliser d'une semaine sur l'autre. Le reste de l'équipe a été peu proactive, en particulier au début, pour donner des retours sur ses avancements et les tâches sur lesquelles elle travaillait et comment ces tâches avançaient.

4.2 Outils utilisés pour l'organisation et la gestion du projet

Discord L'outil principal utilisé pour l'organisation et la gestion du projet a été le logiciel de messagerie textuelle et vocale Discord. Nous l'avons choisi car nous l'utilisons tous quotidiennement et qu'il nous permet l'utilisation de certaines fonctionnalités pratiques comme la création de canaux thématiques pour ne pas mélanger les communication à propos de plusieurs tâches, ou l'épinglage de messages pour les retrouver facilement.

Réunions en présentiel Un outil très efficace que nous avons utilisé lors de l'organisation du projet a été les réunions en présentiel, avec autant de membres de l'équipe que possible. Rétrospectivement, nous aurions du en faire plus souvent. Lors de ces réunions, l'utilisation d'un tableau blanc nous permettait de schématiser et de planifier. Ces réunions étaient également propices à l'entraide et à la pédagogie, permettant des explications en face à face direct.

Git et GitLab Pour gérer le code source Git est devenu un outil indispensable. Toutefois, outre cette utilisation, la création de problèmes suivis a permis de communiquer sur les spécification des tâches à plusieurs reprises. Les historiques de *commit* permettent plus ou moins de voir qui a travaillé à quel moment et de détecter les passages à vide.

Gantt Project Pour créer et officialiser le planning, l'outil Gantt Project a été utilisé, permettant de créer simplement des diagrammes de Gantt. Ces diagrammes ont ensuite été partagés sur le dépôt GitLab pour que tous puissent les consulter.

4.3 Réactivité et gestion humaine

L'un des plus gros problèmes de ce projet a été la gestion de la communication. D'un côté certains membres de l'équipe on été peu proactifs en ce qui concerne le fait d'informer les autres de leur avancement, mais également de leurs difficultés. Certains membres de l'équipe ont également été peu réactifs à informer le chef du projet de leur impossibilité temporaire à remplir leurs fonctions (maladie, problèmes personnels grâves). D'un autre côté, le chef de projet n'a pas été s'enquérir rapidement de l'état de ses équipiers lorsque ceux-ci étaient absents ou ne donnaient pas de nouvelles pendant plusieurs jours.

Ce manque de communication a posé problème pour la redistribution des tâches auprès des autres membres de l'équipe. Si l'informations avait été obtenue plus tôt, les tâches non assurées auraient pu l'être par quelqu'un d'autre et le retard aurait été mitigé.

Toutefois, lors des cinq dernières semaines du projet, lorsqu'une tâche devait être réattribuée, elle l'a été sans délai, pendant la période nécessaire.

5 Diagrammes de Gantt, planning et évolutions

6 Emplois du temps et répartition du travail

7 Conclusion

8 References