UFR de mathématique et d'informatique
Université de Strasbourg

MASTER 2 D'INFORMATIQUE
SCIENCES ET INGÉNIERIE DES RÉSEAUX, DE L'INTERNET ET DES SYSTÈMES

Rapport de projet master

Groupe 1

Ismail ACAR
Olga KRUPOVITCH
Lucas LETT
Dorian SCHWAMBACH

# CTI'XPLORE

# SUIVI DES VÉHICULES ET DES USAGERS DE LA CTI

7 janvier 2022

Encadré par

Stéphane CATELOIN
Pascal MERINDOL

ENTREPRISE KLASCORP

KLAS CORP

# Table des matières

**Table des matières** **3**

**1 Présentation générale du projet** **5**
1.1 Contexte et problématique du projet . . . . . 5
1.2 La solution proposée par KLAS Corp . . . . . 5
1.3 Exemples de scénarios . . . . . 6

**2 Réalisation du projet** **9**
2.1 Justification des choix techniques . . . . . 9
2.2 Architecture du projet . . . . . 10
2.3 Le résultat final . . . . . 11
2.4 Les 13 points clés . . . . . 13
2.4.1 Entre 30 et 50% de développement logiciel . . . . . 13
2.4.2 Gestion de flux de données très importants . . . . . 13
2.4.3 Tolérance aux pannes . . . . . 13
2.4.4 Tolérance à la charge, redimensionnement automatique . . . . . 14
2.4.5 Temps de réponse garantis . . . . . 15
2.4.6 Hébergement sur deux sites . . . . . 16
2.4.7 Application web/mobile . . . . . 17
2.4.8 Sécurisation et analyse de risques . . . . . 18
2.4.9 Supervision . . . . . 18
2.4.10 Double pile v4 et v6 + v6 only . . . . . 19
2.4.11 Démarche d'évaluation des critères . . . . . 19
2.4.12 Organisation du projet . . . . . 19
2.4.13 Supports de communication . . . . . 19
2.5 Perspectives d'amélioration . . . . . 20

**3 Gestion du projet** **21**
3.1 Méthode de travail . . . . . 21
3.2 Gestion du temps et partage du travail . . . . . 21
3.3 Réactivité face aux difficultés d'ordre humain . . . . . 22

# Chapitre 1

# Présentation générale du projet

## 1.1 Contexte et problématique du projet

Dans une problématique d'amélioration de la desserte de la ville d'Illkirch Graffenstaden, la compagnie des transports illkirchois (CTI) souhaite connaître en temps réel la position de ses véhicules bus et tramways ainsi que la position de ses usagers.
La compagnie rencontre aujourd'hui des incohérences de répartition du mobilier urbain sur son réseau et se doit d'améliorer la desserte de certains quartiers en heure de pointe.
Afin d'inciter les utilisateurs à collaborer et à partager leur position, la CTI souhaite proposer aux usagers un service ludique, culturel ou encore d'aide aux déplacements en contrepartie de leur participation.

## 1.2 La solution proposée par KLAS Corp

La solution portée par notre entreprise propose un suivi de la position des véhicules et des usagers permettant de récompenser la fidélité des usagers.

Notre solution, baptisée "CTI'Xplore", est composée de divers éléments à destination de l'entreprise cliente et des usagers.

Pour l'entreprise cliente :

— Une solution d'administration et de consultation des données par le biais d'une plateforme web. Cette interface permet la consultation de statistiques issues de l'étude de la position des véhicules et des usagers sur une période donnée, ainsi que la visualisation en direct et dans le passé de ces positions de manière anonyme.L'interface permet aussi l'édition de la topologie du réseau et l'édition des fiches de stations pour faciliter leur mise à jour.

— Le nécessaire technique de traitement et de stockage des données. Cela comprend la répartition des services sur deux sites distincts, l'équilibrage de charge et la duplication des services dans l'optique d'une haute disponibilité. Le traitement sera réalisé par un serveur principal communiquant via une API et stockant ses données dans une base de données. La base de données conserve les

positions récupérées ainsi que la progression des utilisateurs. Les échanges sont authentifiés et chiffrés.

— Un dispositif électronique embarqué dans les véhicules bus et tramways de la compagnie. Équipés d'une technologie GPS, ces dispositifs renvoient régulièrement la position géographique du véhicule à l'API du serveur par le biais d'internet. Il est aussi possible de détecter les usagers présents dans le véhicule avec un scan wifi régulier.

Pour l'usager :

— Une application mobile qui remonte régulièrement la position géographique d'usagers de manière anonyme et qui propose en contrepartie un jeu d'exploration qui récompense la fidélité des usagers.
L'application relève fréquemment la position GPS du téléphone et l'envoie au serveur de CTI'Xplore. L'utilisateur est en mesure d'activer ou non l'envoi de la position dans un souci de respect de la vie privée. Lorsque la géolocalisation est activée, un solde de points, nommés Oli, s'incrémente plus ou moins rapidement. L'utilisateur peut utiliser ce solde pour obtenir des réductions auprès de la CTI ou de commerçants partenaires.
L'aspect ludique de l'application se traduit par un taux d'exploration du réseau de transport par l'usager. À l'installation, une carte du réseau se compose de stations et d'arrêts verrouillés. En fonction des déplacements de l'utilisateur, les stations vont peu à peu se déverrouiller, débloquant progressivement des succès et augmentant ainsi les points Oli.
Chaque station déverrouillée propose une fiche descriptive dans laquelle se trouvent des informations techniques, historiques, anecdotiques ou pratiques sur la station et ses alentours. On y trouve notamment les prochains passages en temps réel, les perturbations sur le réseau ou les horaires théoriques. Chaque station dispose aussi d'une fonction "Téléportation" qui va calculer l'itinéraire et le coût pour s'y rendre depuis la position de l'usager.
L'esprit de compétitivité entre les différents usagers est présente avec la mise en place d'un tableau des scores

## 1.3 Exemples de scénarios

**Scénario type pour un travailleur**

Jérôme est un illkirchois du sud qui dispose d'un smartphone Android et il utilise les transports en commun chaque jour pour se rendre dans l'Illkirch du nord. Ce matin, Jérôme allume CTI'Xplore et active la localisation GPS. Son solde de points Oli va augmenter et de nouvelles stations peuvent être déverrouillées en fonction de son trajet. Jérôme est détendu car il sait que son solde Oli grimpe et qu'il va pouvoir troquer ses points contre une baguette gratuite à la boulangerie partenaire en rentrant ce soir.

**Scénario type pour un supporter du RCIA**

David passe sa soirée au stade Friedel pour supporter son équipe préférée. Pendant tout le match, il pense à la difficulté qu'il rencontrera à la sortie du match pour rentrer chez lui. Mais pas de panique, il a installé CTI'Xplore, comme un bon nombre des autres supporters du match. La CTI sait donc où se trouvent les foules et peut alors injecter des tramways supplémentaires et augmenter la fréquence de ses véhicules à la sortie du match. David est rassuré ! Il peut rentrer sereinement chez lui !

# Chapitre 2

# Réalisation du projet

## 2.1 Justification des choix techniques

Pour réaliser ce projet, nous avons fait le choix de plusieurs outils ou langages en fonction de nos besoins.

**Choix de Python Kivy** Pour réaliser l'application mobile, nous avons longtemps cherché la technologie à utiliser car beaucoup de solutions ne fonctionnaient pas sur nos machines. Android Studio est une solution trop lourde et l'univers node.JS plutôt complexe. Le choix s'est alors porté sur le langage Python et son extension Kivy qui permet de générer des applications mobiles multiplateformes. Python ne demande pas d'installation d'émulateurs ou d'un IDE lourd. Ce langage simple à mettre en œuvre permet de tester immédiatement notre application sur un smartphone Android relié à la machine et d'accéder aux logs d'erreurs de l'application depuis l'ordinateur.

**Choix de Kubernetes** Nous avons choisi de déployer notre projet sur une architecture basée sur une grappe Kubernetes pour différentes raisons.
Kubernetes est une technologie de plus en plus répandue et approuvée, utilisée aussi bien dans des contextes de petite entreprise que dans des architectures à très grande échelle. À l'heure actuelle, Kubernetes est une référence incontournable en termes de fiabilité pour mettre en place une infrastructure qui doit répondre à des problématiques de redondance, de tolérance aux pannes et à la charge, de redimensionnement, etc. Ces problématiques correspondent justement aux différents critères exposés par notre client.
Le fait de choisir Kubernetes comme base d'infrastructure nous a automatiquement poussés à découper notre projet en micro-services, de sorte à permettre la meilleure modularité possible. De ce fait, chaque service est testé et construit avant d'être conteneurisé automatiquement à l'aide de Docker. Il ne reste ainsi plus qu'à les déployer sur une infrastructure Kubernetes.
Nous avons également mis en place une instance de développement locale en plus de l'instance de production Kubernetes afin de pouvoir tester des changements applicatifs très rapidement directement en local.

**Choix de React JS** Pour réaliser l'interface web d'administration, nous avons décidé d'utiliser la technologie React JS car le Javascript est un langage connu de chacun des membres de l'équipe. Nous étudions React en cours de programmation mobile et nous trouvons son usage correct. Cependant, nous aurions pu utiliser sans difficulté d'autres solutions web.

**Choix de Mongodb** Le projet nécessite un système de base de données pour stocker les positions et la progression des utilisateurs. Il est aussi important de noter que les requêtes entre l'application mobile et la base de données passent par l'API du serveur principal. C'est alors Mongodb qui a été choisi car il présente l'avantage de stocker ses données dans un format texte json. C'est un format nativement présent dans les requêtes GET et POST de l'API et ce format est très facilement utilisable en Python.

## 2.2 Architecture du projet

Notre solution est composée des éléments techniques suivants :

Nous trouvons divers services répartis sur le cluster Kubernetes.
Parmi ces services, nous pouvons trouver un système de queue chargé de récupérer l'ensemble des positions émises par le simulateur ou par les clients mobiles. L'objectif est d'éviter la perte de données. Nous avions le choix entre plusieurs solutions existantes telles que Apache Kafka, RabbitMQ, Nats ou encore KubeMQ. Notre choix s'est porté sur KubeMQ après étude et comparatifs de l'ensemble de ces solutions car cette dernière est légère, compatible avec Kubernetes, bien documentée et intègre de multiples langages comme Python, Go ou Java et cela en fournissant directement un mini client léger.

Pour récupérer les positions réceptionnées par KubeMQ, un service nommé Pooler a été développé en Python. Ce dernier interroge périodiquement le système de queue pour récupérer les positions de la file d'attente et les stocker dans la base de données MongoDB.

La base de données Mongodb permet à la fois de stocker de manière persistante les données utilisateurs et les données GPS. Les données transitent toutes par l'API avant d'accéder à la base de données.

Le service d'API REST sert à fournir un accès aux données de notre projet depuis l'ensemble des micro-services. L'API est également écrite en Python et se repose sur la librairie Flask.

Nous disposons également d'un générateur de position qui va envoyer à notre serveur des coordonnées géographiques pour simuler des utilisateurs réels.

Côté serveur se trouve aussi le site web d'administration en React. Il propose de consulter en temps réel la position des usagers, mais aussi l'historique de leur déplacement. Toutes ces données sont anonymisées pour éviter une atteinte à la vie privée. Le site propose également une série de statistiques utiles pour la CTI afin de déterminer les stations du réseau de transport les plus sollicitées.

Le schéma 2.1 récapitule les différents services évoqués dans cette section.

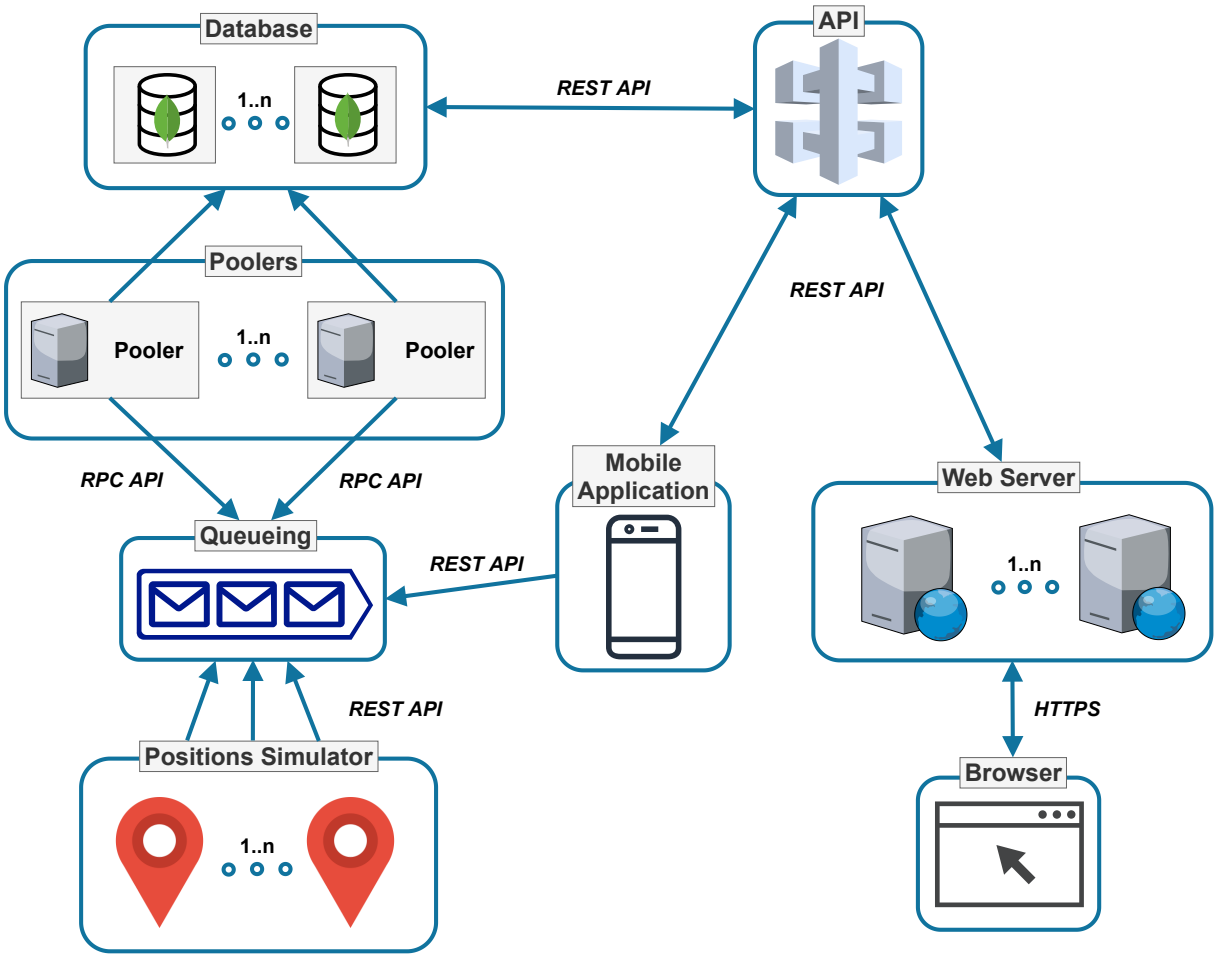


FIGURE 2.1 – Articulation des différents composants de notre solution

## 2.3 Le résultat final

Après la réalisation de ce projet, toutes les fonctionnalités annoncées n'ont pas été achevées.

**Application mobile** Elle est actuellement en mesure de géolocaliser de manière précise le téléphone et d'envoyer la position au serveur CTI'Xplore. L'utilisateur peut activer ou non cette fonction s'il le désire.
Avec la position, l'application trouve la station la plus proche ainsi que la distance jusqu'à celle-ci et peut alors débloquer la station si l'utilisateur s'y trouve.
La carte du réseau s'affiche (cf. figure 2.2a) et les stations sont indiquées verrouillées ou non selon la progression. Chaque station déverrouillée dispose d'une fiche qui affiche uniquement le temps d'attente réel en station. Le solde Oli est fonctionnel et s'incrémente (cf. figure 2.2b). Il est possible de créer sur l'application des comptes utilisateurs et de s'y connecter. De plus, la progression de l'utilisateur est enregistrée sur le serveur et chargée au démarrage.

En revanche, par manque de temps, l'évolution du pourcentage de progression, la présence des succès, l'échange d'Oli contre des réductions, le bouton téléportation et les


(a) Plan du réseau (b) Page d'accueil

FIGURE 2.2 – Captures d'écran de l'application

informations culturelles n'ont pas été réalisés. Nous avons aussi rencontré des difficultés avec la mise en page de la carte sur l'application et la fonction de zoom. La précision de la position GPS a aussi été compliquée à obtenir et nécessite que l'application reste au premier plan en permanence.

**Interface web** Le site est en place à l'adresse http://admin.klascorp.space
Nous pouvons y trouver une page d'accueil contenant des statistiques pour la CTI, mais aussi une carte où il est possible de consulter en temps réel la position des utilisateurs. Une autre carte permet de sélectionner un utilisateur et de consulter l'historique de ses trajets.
Nous avons eu des difficultés avec l'apprentissage de React. La compréhension de ce framework a affecté le temps que nous pouvions consacrer au développement. Par ce manque de temps, l'édition de la topologie du réseau, des contenus de présentation des stations et la gestion des comptes n'ont pas été développées.

**Infrastructure** L'usage de Kubernetes a permis la mise en place des serveurs sur deux sites distincts ainsi que la réplication et l'équilibrage de charge. Un outil de supervision a été mis en place pour surveiller les différents éléments. Le serveur de traitement et son API fonctionnent et sont accessibles par les autres composants. La base de données MongoDB permet bien le stockage de données utilisateurs. Le système de queue KubeMQ fonctionne et les paquets ne sont pas perdus. De plus, tout le système fonctionne en IPv4 et IPv6

De plus, l'intégralité du dispositif électronique devant équiper les véhicules a été mis de côté lors de la réalisation du projet.

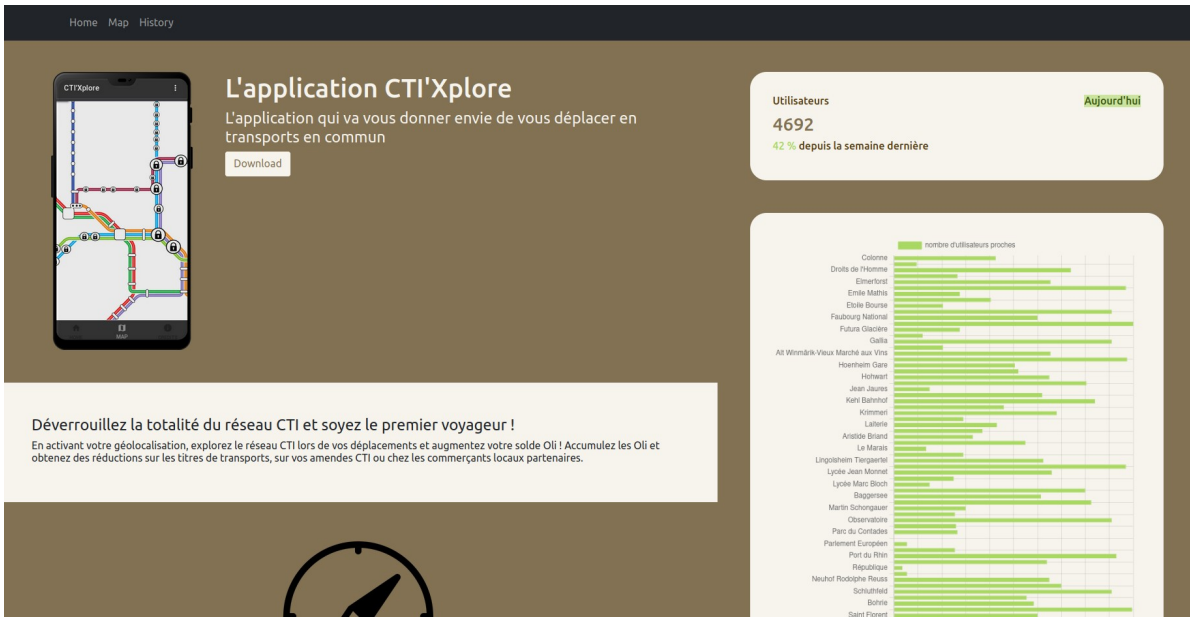


FIGURE 2.3 – Page d'administration avec les statistiques

## 2.4 Les 13 points clés

### 2.4.1 Entre 30 et 50% de développement logiciel

Le projet est composé d'une majorité de développement pour chacun des membres de l'équipe. En plus du développement, nous trouvons des moments de rédaction de documents comme le cahier des charge, l'appel d'offres ou le rapport. Nous trouvons aussi des moments de mesures de performance, de tests réels, d'états de l'art pour le choix des solutions, de réunions et d'auto-formation.

### 2.4.2 Gestion de flux de données très importants

Lors de la définition du projet, nous pensions pouvoir mettre en pratique ou simuler une grande quantité de données. Des essais ont été réalisés avec la simulation d'un très grand nombre de générateurs de positions GPS (cf. point 2.4.5) et le système a résisté. Comme certaines fonctionnalités devant envoyer des données n'ont pas été développées par manque de temps, nous n'avons pas pu tester plus loin.

### 2.4.3 Tolérance aux pannes

Les grappes Kubernetes sont formées de trois nœuds de sorte à mutualiser et regrouper l'ensemble des ressources au sein d'une même grappe. Ces trois nœuds jouent tous les mêmes rôles dans le cluster : contrôleur, travailleur et responsable de sauvegarde d'état du cluster.
En cas d'une panne d'un nœud de la grappe, il en restera toujours deux opérationnels et qui peuvent permettre un fonctionnement normal à Kubernetes car il est toujours

possible d'instancier de nouveaux services. En revanche, si nous perdons deux nœuds au sein d'une grappe, les services actuellement en cours dans le dernier nœud resteront toujours accessibles mais l'ensemble des autres services se trouvant sur les autres nœuds tombés en panne ne pourront pas être migrés sur ce seul nœud.

Si un service tombe en panne, ceci est complètement invisible pour l'utilisateur car l'ensemble de nos services sont répliqués. Cette redondance offre un basculement instantané et implicite lorsqu'une panne survient.

Concernant la base de données, toutes les données de notre application sont persistées sur le disque. Cela permet une reprise implicite en cas de panne sur la base.

### 2.4.4 Tolérance à la charge, redimensionnement automatique

Suite à notre choix d'utiliser une infrastructure basée sur Kubernetes, nous sommes capables de configurer chaque service indépendamment de sorte à définir plusieurs instances d'un même service. Il est ainsi possible de mettre en place des règles personnalisables afin que ces derniers soient par exemple lancés sur des nœuds différents.

Il est également possible d'ajuster le nombre d'instances des services en fonction de certaines métriques, comme les ressources utilisées telle que le CPU, la RAM, le disque ou encore des métriques personnalisées, qui seront récupérées par des services de surveillance du cluster.

Ainsi, dans la mesure où notre application prendrait une plus grande ampleur que la seule ville d'Illkirch, il serait tout à fait possible de rajouter de nouvelles machines aux grappes existantes pour augmenter leurs capacités.

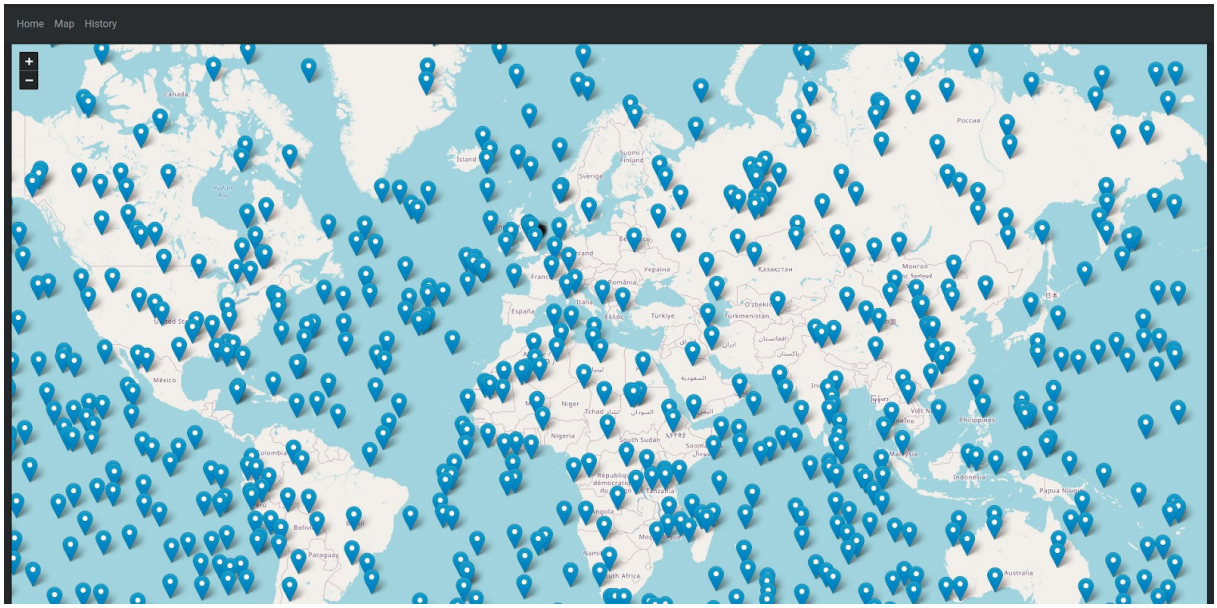


FIGURE 2.4 – Page d'administration avec 10.000 émetteurs simultanés

### 2.4.5 Temps de réponse garantis

Pour pouvoir évaluer les performances de notre application, nous avons dû solliciter notre simulateur pour pouvoir représenter un pic d'utilisation conséquent. Pour cela, le simulateur a été conçu de sorte à pouvoir simuler plusieurs dizaines de millier d'émetteur de position en parallèle. En couplant ceci à une grande instanciation de ce service sur le cluster, nous avons pu obtenir un groupe de simulateur assez important. Cela nous a permis de récolter des données statistiques sur les performances de notre application.

La figure 2.4 représente une capture d'écran de la carte de temps réel du site d'administration lorsque nous étions à 10.000 émetteurs simultanés.

Pour pouvoir garantir les temps de réponse de notre applications, nous nous sommes intéressé à deux différents temps :

— Le temps de réponse de notre API avec la requête la plus gourmande possible : la dernière position GPS de l'ensemble de tous les émetteurs
— Le temps maximum de remonté des positions envoyées par les émetteurs

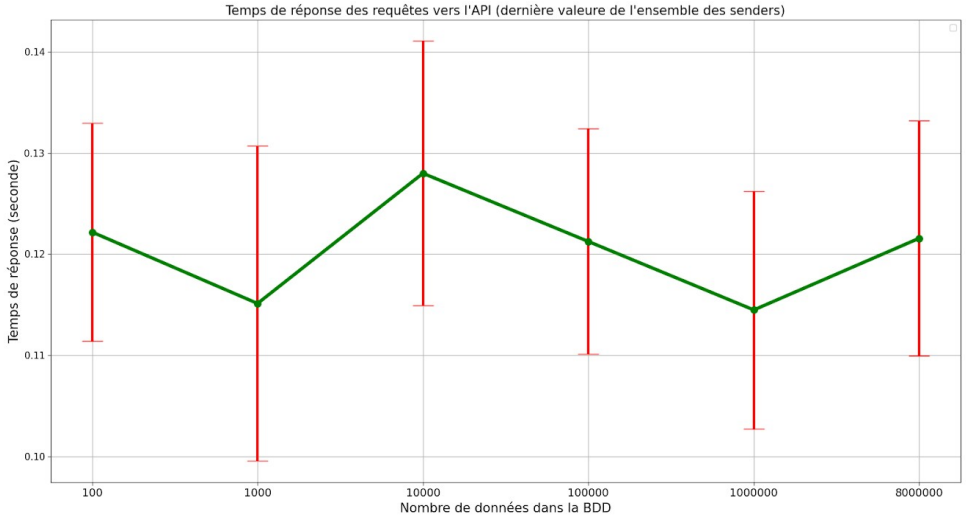


FIGURE 2.5 – Temps de réponse de notre API sur la requête la plus gourmande

Les deux graphiques 2.5 et 2.6 représentent ces temps en se basant sur des abscisses différentes, respectivement le nombre de données dans la BDD et le nombre d'émetteurs simultanés.

Nous pouvons donc voir que le temps de réponse de notre API se fait en temps constant, quel que soit le nombre de données présentes dans cette dernière. Ceci a été rendu possible en utilisant en réalité deux tables distinctes : une table qui sauvegarde l'ensemble des positions émises depuis le début et une autre table, hashé cette fois-ci sur sa clé primaire qui va mapper sur les dernières positions pour chaque émetteur.

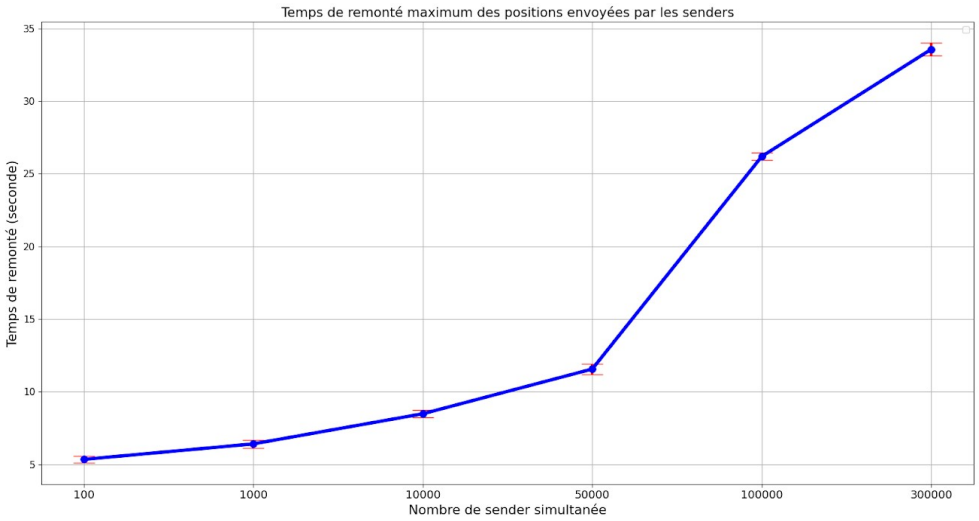


FIGURE 2.6 – temps maximum de remontée des positions envoyées par les émetteurs

Concernant le temps maximum de remontée, nous pouvons voir que cette fois-ci le temps n'est plus constant, mais quasi linéaire. Ceci s'explique simplement par des limites logicielles, qui peinent à supporter des traitements hautement parallèles. Cependant, ces temps relevés sont tout de même largement satisfaisants pour notre application, qui n'a pas besoin de plus de précision.

### 2.4.6 Hébergement sur deux sites

Nous avons rencontrés beaucoup de problèmes avec l'hébergement sur deux sites. En effet, nous sommes d'abord partis avec l'idée de créer deux clusters Kubernetes sur les deux sites, en spécifiant l'un d'entre eux comme maître pour répliquer automatiquement tout déploiement applicatif sur le second cluster. Cela a pu se faire en utilisant un outil appelé Kubefed.

Cependant, la mise en place d'un équilibrage de charge entre ces différents sites était trop complexe. Nous voulions partir sur un équilibrage de charge au niveau DNS, en doublant simplement les entrées dans les résolutions de noms. Mais cette solution engendre un problème de tolérance aux pannes car si l'un des clusters est amené à tomber, 50% des requêtes vers notre application seraient perdues.

Nous avons trouvé deux autres solutions qui devraient marcher dans notre cas, mais que nous n'avons malheureusement pas pu mettre en place par manque de moyen.

La première solution est l'usage d'anycast pour utiliser les mêmes adresses (IPv4 et IPv6) sur les deux clusters. Ceci permettrait d'obtenir un équilibrage de charge en contournant le problème du doublement des entrées DNS. Cependant cette solution n'est pas exploitable dans notre cas car nous n'avons pas la possibilité de configurer de l'anycast sur nos deux sites.

La seconde solution est l'usage d'une autre machine située sur un site différent de nos sites d'Illkirch et de l'Esplanade. En y installant un reverse proxy, nous pourrions mettre en place un équilibreur de charge physique, comme peuvent le faire des clouds providers bien connus. Cependant, n'en n'ayant pas la possibilité, cette solution n'a également pas pu être déployée.
Nous avons finalement opté pour une infrastructure comme illustrée sur la figure 2.7 : deux clusters distincts faisant tourner chacun notre application avec des noms de domaines différents. De cette manière, nous arrivons à assurer la répartition sur au moins deux sites, mais sans fournir un équilibrage de charge entre eux.

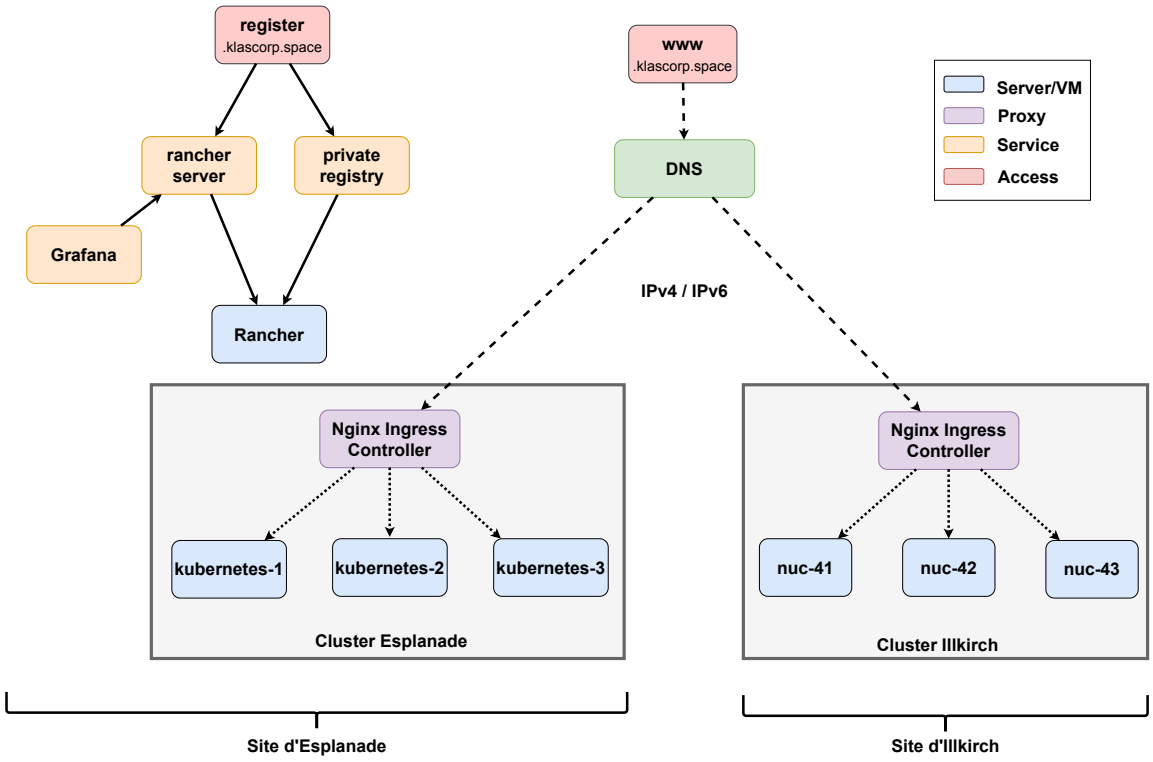


FIGURE 2.7 – Découpage géographique des services

### 2.4.7 Application web/mobile

Ce projet comprend bien un développement d'application mobile Android en Python Kivy et une interface web en React.JS.

### 2.4.8 Sécurisation et analyse de risques

Dans notre architecture, seuls le serveur web, l'API et le système de queue sont exposés vers l'extérieur à travers le reverse proxy Nginx. Le serveur web et l'API sont uniquement accessibles de manière sécurisée avec un certificat HTTPS.

Le serveur web ne prend en compte aucune entrée utilisateur, elle offre simplement une interface pour notre application.

L'API est cependant publique en lecture mais cela ne pose pas de problème de sécurité car les temps d'accès sont constants. Il n'y pas de risque d'attaque de type amplification.

Enfin, le système de queue est uniquement accessible par authentification préalable et en utilisant un chiffrement de bout en bout.

L'application mobile offre également une sécurité vis-à-vis de la connexion de l'utilisateur. Lors de la connexion, un token de session est généré et conservé dans l'application. Il est alors utilisé pour chaque requête afin de les authentifier. La connexion et le token ont une durée de vie d'une semaine.

### 2.4.9 Supervision

Nous avons pu mettre en place une surveillance au niveau des grappes Kubernetes en nous basant sur le projet kube-prometheus. De ce fait, nous avons déployé une pile de surveillance qui comporte trois principaux éléments :

— l'Alert Manager, qui permet la configuration d'un certain nombre d'alertes
— Prometheus, qui permet l'affichage de ces alertes et la récupération des métriques exposées par Kubernetes (ou des métriques personnalisées)
— Grafana, qui permet d'afficher un très grand nombre de tableaux de bords illustrés avec des graphiques temps réel.

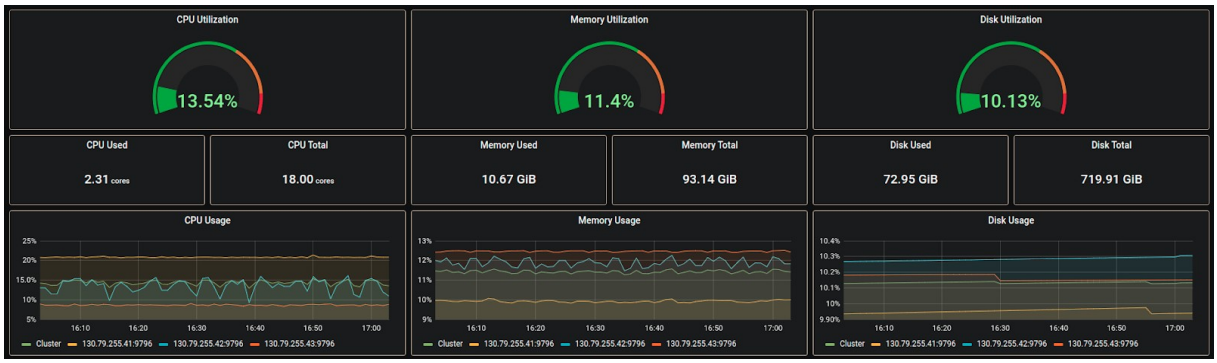


FIGURE 2.8 – Capture d'écran de l'outil de supervision Grafana

La figure 2.8 illustre la page d'accueil de Grafana avec l'utilisation des métriques principales du cluster Kubernetes, à savoir l'utilisation CPU, RAM et disque.

### 2.4.10 Double pile v4 et v6 + v6 only

Nos clusters Kubernetes sont configurés pour travailler en IPv4 seulement en interne. En effet, Kubernetes offre une abstraction assez importante des applications en les déployant directement dans des réseaux locaux privés à chaque grappe. Ces réseaux étant complètement internes aux clusters, il n'y a pas d'intérêt à utiliser de l'IPv6 ici. Notons aussi que seule la version bêta de Kubernetes propose cette option aujourd'hui.

Pour pouvoir fournir une double pile v4 et v6, ainsi que du v6 only, nous avons configuré notre Reverse Proxy / Nginx Ingress Controller (cf : figure 2.7) pour qu'il permette les entrées aussi bien en IPv4 qu'en IPv6. Ainsi, en configurant en plus nos entrées DNS avec des entrées v4 et v6, nous arrivons à offrir cette connectivité en double pile ainsi qu'en IPv6 only.

La figure 2.9 représente les entrées DNS pour notre nom de domaine api.klascorp.space qui pointe vers notre serveur. Nous pouvons donc y voir la présence d'IPv4 et d'IPv6.

| | | | |
|---|---|---|---|
| api.klascorp.space. | 0 | A | 130.79.255.44 |
| api.klascorp.space. | 0 | AAAA | 2001:660:4701:2014:ca:fe:4:0 |

FIGURE 2.9 – Capture d'écran du DNS avec une entrée A et AAAA

### 2.4.11 Démarche d'évaluation des critères

Certains critères ont été fixés en début de projet afin de noter les performances de notre solution.

### 2.4.12 Organisation du projet

L'organisation du projet est détaillée dans le chapitre 3.

### 2.4.13 Supports de communication

Nous avons mis en place plusieurs éléments de communication pour faire la promotion de CTI'Xplore.

Tout d'abord, vous pouvez trouver un site vitrine sur internet à l'adresse http://klascorp.space
Des logos d'application ont été dessinés et des templates de document ont été réalisés pour les comptes rendus de réunions et les échanges avec le client.
Nous avons contacté la CTS à Strasbourg pour obtenir le template de leurs affiches de communication du lancement de leur application. Nous espérons finir notre affiche et éventuellement une vidéo d'ici la magnifique présentation de notre projet.

Les éléments graphiques ont été réalisés grâce à Lunacy et Canva. Lunacy est une application graphique vectorielle pour la conception d'interfaces utilisateur et de sites Web. Canva est une plateforme de conception graphique, utilisée pour créer des designs pour les médias sociaux, les affiches et tout autre contenu visuel.

## 2.5 Perspectives d'amélioration

Dans l'optique d'améliorer le projet, il serait intéressant de mettre en place toutes les fonctionnalités qui n'ont pas été réalisées. Elles ont été listées à la section 2.3.

Parmi les fonctionnalités qui ont été développées, certaines peuvent être améliorées. Le code source de l'application n'est pas optimisé et cela s'explique par la prise en main de l'outil Kivy durant le développement de l'application. Il serait intéressant de recommencer l'outil en partant sur de bonnes bases.
De plus, l'application a été conçue avec un processus en arrière plan capable de récupérer des localisations GPS en permanence afin de permettre d'utiliser son smartphone. Malheureusement, nous n'avons pas réussi à obtenir des données GPS assez précises avec une localisation issue du processus en arrière plan. Cela explique pourquoi notre application doit absolument rester au premier plan et être ouverte en permanence pour fonctionner et ainsi avoir des données GPS très précises.
Concernant l'affichage de la carte du réseau, il faudrait continuer le développement pour obtenir un affichage différent entre une station terminus et une station classique.

De plus, la sensibilité de la carte est trop élevée car la fiche descriptive d'une station s'ouvre alors qu'un zoom est effectué et non un clic.

Concernant l'interface web, le plus grand axe d'amélioration est l'optimisation de l'affichage. Le regroupement des marqueurs qui sont proches les uns des autres lorsque l'utilisateur dézoome par exemple. De plus, la création d'un style homogène CSS pour la totalité de l'interface utilisateur est nécessaire afin d'offrir une meilleure expérience et de donner envie à l'utilisateur de continuer à utiliser CTI'Xplore.

Concernant la partie technique, le développement d'application web en React JS nécessite l'utilisation de npm (un gestionnaire de paquets pour le langage de programmation JavaScript). Certaines dépendances de npm sont devenues obsolètes et nécessitent une intervention manuelle.

# Chapitre 3

# Gestion du projet

## 3.1 Méthode de travail

La réalisation du projet a rassemblé quatre étudiants de master SIRIS. Nous avons organisé, en début de projet, de nombreuses réunions en présentiel dans l'objectif de définir le sujet, le cahier des charges et l'appel d'offres. À partir du début de la réalisation concrète du projet, ces réunions sont devenues virtuelles.
Chaque réunion se conclut avec la réalisation d'un compte rendu et le chef de projet recontacte chaque membre du groupe pour lui rappeler ce qu'il doit faire.
Il en est de même avec les réunions client toutes les deux semaines, un compte rendu est rédigé et un message résumé est transmis aux membres du groupe.

Pour communiquer, nous utilisons avant tout un serveur de discussion Discord, avec des salons de discussion séparés par sujets. De plus, le partage et stockage de documents s'est fait à travers un Drive Google et chaque membre s'est vu attribuer une adresse mail dans le domaine "klascorp.space", dans un souci de communication professionnelle avec le client.
Si un membre du groupe se retrouve bloqué, nous passons également par voie téléphonique.

Le code développé a été partagé sur un dépôt git. Nous y disposons de différents répertoires pour séparer le code du site, de l'infrastructure et de l'application. Nous utilisons également git pour son wiki où nous y centralisons les liens vers différents outils comme le Drive ou les comptes-rendus de réunions.

## 3.2 Gestion du temps et partage du travail

Concernant la répartition des tâches, nous nous sommes basés sur les compétences de chacun. Nous avons remarqué que les étudiants CMI ont eu des expériences en stage qui ont pu servir. C'est ainsi qu'Ismail s'est occupé de la partie infrastructure suite à son stage d'été 2021 autour de Kubernetes. Olga s'est rapidement présentée comme bonne graphiste et a dès le début créé les logos et les templates utilisés pour l'application. Elle s'est ensuite dirigée vers le développement web car elle maîtrisait bien le sujet.
Dorian s'est alors vu attribué la mise en place du reverse-proxy en binôme avec Ismail,

mais était bloqué. Lucas s'est chargé de l'application mobile car personne n'en avait développé jusqu'à présent. Dorian a ensuite rejoint Lucas car la charge de travail pour l'application était plus grosse que prévu.

Nous avons mis en place un tableur GSheet sur le Drive commun pour que chaque membre saisisse ses heures de travail et pour disposer d'un repère pour jauger l'implication de chacun. Tous les membres ont dépassé la barre des 100 heures de travail plus ou moins rapidement selon les circonstances. Le gsheet est visible sur notre drive : https://tinyurl.com/bdhebe8f

Afin d'avoir une visibilité sur l'accomplissement global du projet, un diagramme de Gantt a été réalisé et mis à jour toutes les deux semaines. Tout d'abord assez mal conçu, il a été repris début octobre mais n'a pas toujours été respecté. Du retard s'accumule en permanence suite à la présence de projets d'autres cours ou au taux de procrastination de chacun. Mi-décembre, certaines fonctionnalités non vitales pour le projet ont été mises à l'écart pour centrer le travail sur le cœur de l'application. Le diagramme de gantt est visible sur notre dépot git : https://tinyurl.com/2p8vxkhd

## 3.3 Réactivité face aux difficultés d'ordre humain

Pendant le projet, nous avons rencontré plusieurs difficultés qui ont entraîné des retards, des reports ou des modifications dans la répartition des tâches.
Au début, les membres se sont vu attribuer des tâches en fonction de leurs compétences ou de leurs envies. Mais des aléas peuvent survenir. D'autres projets à rendre, des problèmes familiaux ou des maladies sont des raisons à ces retards. Des blocages dans le développement suite à des soucis matériels ou des soucis de documentation peuvent aussi devenir des imprévus.
Ces aléas sont survenus durant la réalisation de notre projet.
L'objectif était de débloquer au plus vite la situation.
Dans un premier temps, contacter le membre par message ou même par téléphone. Les appels téléphoniques sont plus efficaces dans la résolution des problèmes qu'un échange de mail. Aider le membre s'il a besoin d'aide et s'il est bloqué ou encore redistribuer les rôles et les tâches si un membre ne peut pas avancer sont des solutions possibles pour continuer le projet en mode dégradé.
Le diagramme de Gantt est mis à jour pour y faire refléter les retards et définit qui est en mesure d'accueillir un tâche supplémentaire.