4IRC - Projet Scientifique

Cahier des charges

Système de gestion et suivi des incidents pour les SDMIS de Lyon

1. Contexte et objectifs du projet

Le Service Départemental-Métropolitain d'Incendie et de Secours (SDMIS) de Lyon/Villeurbanne assure la gestion et la coordination des interventions d'urgence sur le territoire lyonnais. La densité urbaine, la multiplicité des risques et la dispersion des casernes imposent une coordination précise et efficace des ressources tant humaines que matérielles.

L'objectif est de développer un nouveau système intégré permettant :

- le **suivi en temps "réel"** des incidents et interventions ;
- une **coordination optimisée** des équipes et véhicules.

2. Besoins métier

Ce **Système Intégré de Gestion et Suivi des Incidents** devra répondre aux besoins suivants et réaliser les fonctionnalités décrites dans ce document.

2.1. QG SDMIS

Le SDMIS a besoin d'une application principale utilisée depuis la quartier général (QG) s'occupant de la gestion et de la supervision. Nommons-la "**App QG**" :

2.1.1. Supervision des interventions

Les opérateurs du QG doivent pouvoir être informés en temps "réel" des éléments suivants sur une **carte interactive et dynamique** :

- la localisation, la survenance et l'évolution des incidents ;
- la localisation, l'état et la disponibilité des ressources (véhicules et personnel) ;
- l'état de chaque intervention et quelles ressources leur sont affectées.

Temps réel : en cas de *pulling*, prévoir une fréquence de rafraîchissement adaptée.

2.1.2. Gestion des ressources et interventions

Ils doivent aussi pouvoir effectuer les tâches suivantes :

- déclarer eux-même la survenance d'une intervention ;
- déclencher une intervention sur un incident ;
- affecter des ressources à cette intervention.

Cependant, le système a pour rôle de faciliter le déploiement efficace des ressources (rapidité de déploiement, équilibrage des ressources affectées). Pour cela, dès la survenance d'un incident, le système doit :

- proposer la création automatique de l'intervention en un clic, avec pré-affectation des ressources adaptées selon :
    - le **type d'incident** ;
    - la **distance** au lieu de l'événement ;
    - le **statut opérationnel** (disponibilité, état fonctionnel, jauges de ressources, etc.).
- laisser la décision finale à l'utilisateur.

2.2. Équipes sur le terrain

Les équipes en intervention s'y rendent avec un véhicule. Elles doivent pouvoir :

- Être localisables à tout instant ;
- Signaler la fin d'une intervention (signifiant la libération et le retour des équipes et ressources) ;

- Ainsi que toute autre information que vous jugerez pertinente.

Pour ce faire, chaque véhicule (voir chaque agent aussi) est équipé d'un **dispositif permettant de saisir et transmettre** des informations du terrain via radiofréquences (voir "*IoT*" plus bas). Nommons-la "**App Terrain**".

2.3. Simulation d'incidents

Attention : cette partie n'est pas approfondie. **C'est à vous** de concevoir cette simulation et de définir l'ensemble de ses responsabilités.

Afin de pouvoir garantir le fonctionnement de l'application, elle devra être fournie avec un outil de simulation. Son rôle est de représenter (prendre en charge) les états et les changements d'états (événements) **de tous les éléments externes au système**, afin de tester et valider son bon fonctionnement (captation, traitement / décision, restitution de ces états externes.

Cela devra inclure (sans que cela soit exhaustif) :

- Incidents :
  - Générer des **incidents simulés** : type d'incident, localisation, gravité, heure.
  - Faire **évoluer** ces incidents.
  - Les transmettre de façon cohérente à l'App QG.
- Véhicules : voir la section IoT et simulation radiofréquence plus bas
  - Simuler les véhicules, leur localisation, leurs déplacements, etc. ;
  - Saisie d'information via le terminal du véhicule : simuler ces informations ou fournir une interface de simulation pour permettre à un testeur de les saisir.

2.4. Intégration de données externes

Voici des ouvertures possibles pour enrichir le système. Leur implémentation n'est pas obligatoire, mais peut constituer une amélioration fonctionnelle :

- météo (événements climatiques critique)
- trafic routier (affectation des ressources, délais)
- historique des incidents pour analyse postérieure

3. Fonctionnalités attendues

Cette section vise à détailler certaines fonctionnalités sans répéter l'ensemble des besoins métiers déjà évoqués plus haut. Elle précise uniquement les fonctionnalités qui nécessitent des règles, formats ou comportements explicites pour être correctement implémentées.

3.1. Remontée d'un nouvel incident

En conditions réelles, les incidents sont généralement remontés par un signalement humain : appel téléphonique aux services d'urgences, alarme déclenchée et entendue, etc. Lorsque le SDMIS reçoit ces alertes, un opérateur saisit l'incident depuis l'App QG.

Dans le cadre d'une simulation, voici deux pistes différentes possibles, imaginées pour remonter ces nouvelles informations dans le système :

- soit une **interface de simulation** dédiée, consultée par l'opérateur en temps réel : lorsqu'il y constate un événement, il peut le saisir dans l'App QG comme il l'aurait fait après un appel téléphonique.
- soit un **accès automatique et direct** de la simulation à l'API de l'App QG. Ce faisant, il convient de délimiter clairement les droits donnés à la simulation sur cette API.

Cette liste n'est pas exhaustive : l'objectif est de choisir une solution propre qui permet d'avancer dans les développements, mais en aucun cas, les choix côté simulation ne doivent avoir d'impact sur la façon de développer l'App QG.

3.2. Nature des incidents

Les incendies ne sont qu'une fraction des différents types d'incidents auxquels font face les pompiers, de même qu'ils possèdent différents types de véhicules dédiés en fonction. Le système doit permettre de **gérer cette diversité** et être **ouvert à l'évolution** sur ces aspects.

4. Besoins techniques

4.1. Architecture système

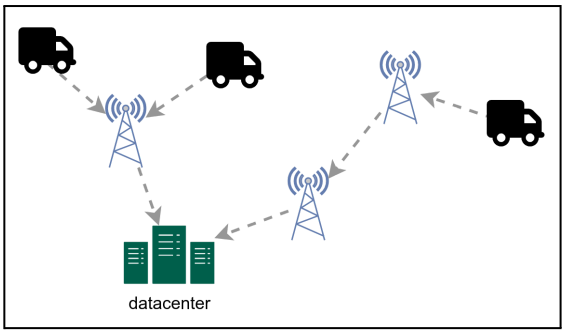
Architecture globale du système à définir avec un découpage en composants cohérent et dans le respect des standards de développement. Il convient de définir :

- définir tous les composants du système (rôles, technologies, y compris l'IoT) ;
- définir les interfaces entre eux (protocoles, types d'information transmises, sens de communication) ;
- définir les modes de stockages des données et leurs emplacements ;
- prévoir une architecture en concordance avec les attentes en termes d'Infrastructure et déploiement ;
- prévoir aussi une architecture modulaire et découplée pour faciliter les évolutions et substitutions de composants (en particulier pour la bascule entre simulation et réalité).

4.2. IoT et simulation radiofréquence

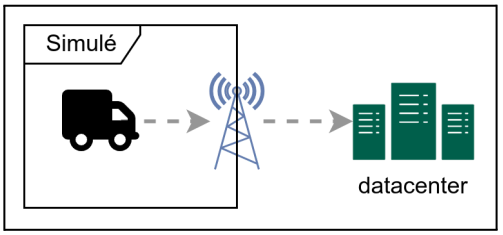
4.2.1. Infrastructure du réseau

En conditions réelles, le réseau de communication par radiofréquences utilisé par le SDMIS reposera sur 3 types d'appareils :



1. **Émetteurs** : dispositif à bord de chaque véhicule, associé à un capteur GPS et un terminal (tablette ou autre) ;
2. **Passerelles** : maillage sur tout le secteur Lyon/Villeurbanne d'émetteurs-récepteurs radiofréquence (microbits) ;
3. **Récepteur** : en fin de course, un unique microbit récepteur et centralisateur est relié en liaison série à une machine connectée à Internet. Celle-ci pourra ensuite transmettre les données au datacenter hébergeant la solution.

Mais dans le cadre d'une simulation, celle-ci sera probablement aussi située aussi dans le datacenter. Aucun véhicule réel n'étant disponible, seuls deux dispositifs seront utilisés :



1. **Une seule passerelle** microbit, connectée en liaison série à la simulation qui l'alimente avec les informations issues de véhicules et d'interventions simulées. Ce microbit fonctionnera donc en **émetteur** radiofréquences ;
2. **Le récepteur** qui jouera son rôle comme prévu.

4.2.2. Protocole et sécurité des communications

Le protocole de communication à concevoir doit répondre aux exigences suivantes :

- **Données et modes d'envoi** :
  - Transmettre des **messages de natures variées** (coordonnées GPS, fin d'intervention, etc.) ;
  - Supporter des **transmissions automatiques périodiques** (ex : coordonnées GPS) et des **transmissions à la demande** (ex : fin d'intervention) ;
- **Fiabilité et robustesse** des communications (faible sensibilité aux pertes, cohérence des données, etc.) ;
- **Sécurité des données** transmises (intégrité, authenticité, confidentialité).

4.3. Infrastructure et déploiement

Le développement doit s'appuyer sur une forge logicielle afin d'assurer la mise en place d'une intégration et un déploiement continus (CI/CD).

La chaîne CI/CD doit permettre la compilation, les tests unitaires et d'intégration, le déploiement automatique de la solution dans un environnement exécutable.

Vous pouvez utiliser Gitlab.com comme forge. Vous pouvez utiliser les runners partagés de Gitlab ou configurer vos propres runners locaux. Il n'est pas attendu – dans le cadre du projet – qu'un déploiement dans le cloud soit effectif, juste que tout soit préparé en ce sens.

4.4. Technologies à utiliser

- Requis :
  - Java (sans *framework* ou au moins dans son propre *process*, pas en tant que serveur web !)
    - Simulation
    - App QG, pour la partie "prise de décision" (quelles ressources attribuer à quelles interventions) :
- Conseillé :
  - Interface QG : web (technologies : libres, suggestion : PHP ou Python Flask)
  - Affichage carte : web (conseillé Leaflet, possibilités OSM, MapBox, Google Map, etc.)
  - BDD : selon besoins
  - Technologiques de communication : selon besoin (HTTP REST, SSE, Websocket, etc.)

5. Exigences de qualité et de sécurité

5.1. Qualité logicielle

- Documentation d'installation
- Documentation d'architecture

- Documentation d'utilisation
- Tests Unitaires API/Backend
- Tableau des charges (licences, serveurs, coûts APIs, etc.)

5.2. Sécurité informatique

- accès restreints : authentification (mise en place d'un serveur **Keycloak**)
- respecter les bonnes pratique de l'**OWASP top 10**
- échanges sécurisés/chiffres (API, réseau et éventuellement radio)
- fiabilité / résistance aux pannes, reprise sur incident

Pour aller plus loin :

- mettre en place un SAST (SonarQube)
- chiffrement des communications sur le réseau (pour l'IoT : voir avec les enseignants)
- souveraineté et hébergement des données