Titre: Title:	Modèle IoT de gestion du feu en temps réel dans les villes intelligentes
Auteur: Author:	Armel Kemzanga Landry Sanou
Date:	2022
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Sanou, A. K. L. (2022). Modèle IoT de gestion du feu en temps réel dans les villes intelligentes [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10310/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/10310/
Directeurs de recherche: Advisors:	Samuel Pierre
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Modèle IoT de gestion du feu en temps réel dans les villes intelligentes

ARMEL KEMZANGA LANDRY SANOU

Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

Génie informatique

Avril 2022

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

Modèle IoT de gestion du feu en temps réel dans les villes intelligentes

présenté par **Armel Kemzamga Landry SANOU**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Alejandro QUINTERO, président
Samuel PIERRE, membre et directeur recherche
Abdellah ZYANE, membre externe

DEDICACE

Je dédie ce travail à ma famille, ma mère Ouédraogo Léontine, mon père Sanou Frédéric, mes frères, et toutes les personnes qui m'ont soutenu en particulier Abdellah Zyane, Congo Pengwendé Ismaël Armel, Dabo Abdoul Djallil et Ouédrago Damien.

REMERCIEMENTS

Je tiens à remercier toutes les personnes qui ont contribué au succès de ma maîtrise recherche et qui m'ont aidé lors de la rédaction de ce mémoire.

Je remercie Pr Samuel Pierre, mon directeur de recherche, professeur à l'École Polytechnique de Montréal et directeur du Laboratoire de recherche en Réseautique et Informatique Mobile (LARIM), pour m'avoir offert l'opportunité de faire ma maîtrise dans son laboratoire de recherche.

Je remercie Dre Franjieh El Khoury, coordonnatrice du LARIM et superviseure du projet APSEC, pour son encadrement tout au long du mémoire, ainsi que pour son aide précieuse dans la relecture et la correction de ce mémoire.

Je remercie la société Flex Group qui soutient ce projet ASPEC dont fait partie mon mémoire de recherche.

Enfin, je remercie Polytechnique Montréal et l'ensemble de son personnel et mes collèges du LARIM pour leur support.

RÉSUMÉ

Chaque année, nous observons des milliers d'incendies résidentiels dans nos villes. Ces incendies causent de nombreux décès et dommages matériels. Au Québec, il y a en moyenne 16 500 incendies par année dont au moins 5 600 sont des incendies résidentiels. Entre 2015 et 2017, les incendies ont causé 131 décès et 540 millions de dollars CAD de dommage en moyenne par année. En 2018, il y avait 11 046 incendies avec 91 décès et 836.1 millions de dollars CAD de dommages en Ontario. Les personnes chargées d'intervenir en cas d'incendies sont les sapeurs-pompiers. Ces derniers disposent d'un temps très étroit (i.e., 5 à 10 minutes) pour sauver le maximum de personnes et arrêter la propagation des incendies. Pour aider les sapeurs-pompiers dans leurs tâches, de nombreuses solutions utilisant les *Technologies de l'Information et de la Communication » (TIC)* ont été mises en place. L'avènement des *villes intelligentes* et des nouvelles technologies, tels que le *réseau 5G*, les *objets connectés « Internet of Things » (IoT)* et la technologie *« Building Information Model » (BIM)*, ont permis la création de solutions innovantes. Cependant, ces solutions présentent quelques lacunes comme l'incapacité de fonctionner en temps réel et de s'adapter à la plupart des situations d'incendie. Dans ce mémoire, nous proposons un nouveau modèle de gestion des incendies en temps réel. Ce modèle exploite les systèmes distribués et l'architecture microservice pour fournir un système robuste, adaptable et évolutif. Le modèle proposé est composé de quatre modules : le réseau de capteurs sans fil « *Fire Emergency Sensor Network* » *(FESNet)*, l'*application distribuée*, les *applications clients* et un *serveur Web* qui agit comme intermédiaire entre l'*application distribuée* et les *applications clients*. Afin de valider notre modèle, un prototype capable du traitement de l'information en temps réel et de l'affichage des bâtiments en 3D a été implémenté. Nous avons effectué trois simulations en utilisant l'algorithme « *Parallel Merge Sort* » pour le triage des données. Dans chaque simulation, nous avons utilisé un nombre différent de processeurs pour trier une quantité de données variant de 100 mille à 1 million. Comme résultats de ces trois simulations, nous concluons que plus nous augmentons le nombre de processeurs, plus le temps de calcul diminue. De plus, lorsque les ressources nécessaires au fonctionnement en temps réel d'un algorithme sont disponibles, le modèle proposé est capable d'allouer de manière précise les ressources demandées. Ceci valide que notre modèle proposé est pertinent du point de vue précision et efficacité dans la répartition des tâches d'évacuation d'un bâtiment en cas d'incendie par des sapeurs-pompiers.

ABSTRACT

Every year, we see thousands of residential fires in our cities. These fires cause many deaths and property damage. In Quebec, there are an average of 16,500 fires per year of which at least 5,600 are residential fires. Between 2015 and 2017, fires caused 131 deaths and CAD 540 million in damage on average per year. In 2018, there were 11,046 fires with 91 deaths and CAD 836.1 million in damage in Ontario. The people responsible for responding to fires are the firefighters. Firefighters have a very limited amount of time (i.e., 5 to 10 minutes) to save as many people as possible and to stop the fire prorogation. To help firefighters in their tasks, many solutions using *Information and Communication Technologies (ICT)* have been implemented. The advent of smart cities and new technologies, such as the *5G network*, the *Internet of Things (IoT)* and the *Building Information Model (BIM)*, have allowed the creation of innovative solutions. However, these solutions have some shortcomings such as the inability to operate in real time and to adapt to most fire situations. In this dissertation, we propose a new model for real-time fire management. This model exploits distributed systems and microservices architecture to provide a robust, adaptable and scalable system. The proposed model is composed of four modules: The *Fire Emergency Sensor Network (FESNet)*, the *distributed application*, the *client applications* and a *Web server* that acts as an intermediary between the *distributed application* and the *client applications*. To validate our model, a prototype able to process the information in real-time and to display the buildings in 3D, was implemented. Moreover, we perform three simulations using the Parallel Merge Sort algorithm. In each simulation, we use a different number of processors to sort an amount of data ranging from 100 thousand to 1 million. As results of these three simulations, we conclude that more than we increase the number of processors more than the processing time is reduced. In addition, when the required resources for the real-time processing of an algorithm are available, the proposed model can accurately allocate the requested resources. This validates that our proposed model is relevant in terms of accuracy and efficiency in the allocation of tasks for the evacuation of a building in case of fire by the fire fighters.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

2D	2 Dimensions
3D	3 Dimensions
ABM	Agent Based Modeling
AR	Augmented Reality
ASET	Available Safe Egress Time
BIM	Building Information Model
CCTV	Closed Circuit Television
CO	Carbone Monoxide
CO2	Carbone Dioxide
COHb	Carboxyhemoglobin
DFAFR	Data Forwarding Approach For Firerescue
FDS	Fire Dynamic Simulation
FESNet	Fire Emergency Sensor Network
GIS	Geographic Information System
HTTP	Hypertext Transfer Protocol
IFC	Industry Foundation Classes
IP	Internet Protocol
IoT	Internet of Things
NFPA	National Fire Protection Association
NIST	National Institute of Standards and Technology
OLED	Organic Light Emitting Diode
OSHA	Occupational Safety and Health Administration

PC	Personnal Computer
RSET	Required Safe Egress Time
RTC	Real Time Computing
RTSJ	Real Time Specification for Java
SIM	Service de Sécurité Incendie de Montréal
TCP	Transmission Control Protocol
TIC	Techniques de l'Information et de la Communication
VR	Virtual Reality

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Les incendies urbains sont des désastres qui se produisent très fréquemment dans nos villes. Les deux principaux problèmes causés par ces incendies sont la perte de vies humaines et les dommages économiques dus à la destruction des bâtiments et à l'arrêt des services commerciaux dépendant de ces bâtiments. Les premiers acteurs à intervenir dans la lutte contre ces incendies sont les sapeurs-pompiers. La mission principale des sapeurs-pompiers est de contrôler et éteindre les incendies. Cependant, leur rôle ne se limite pas à ce niveau. De manière générale, un pompier a pour mission de protéger et secourir les personnes ainsi que l'environnement. De ce fait, en cas d'accident de route ou d'effondrement d'un immeuble, les sapeurs-pompiers seront envoyés sur place même s'il n'y a pas d'incendie. Le présent mémoire a pour objectif de fournir une solution qui facilitera le travail des sapeurs-pompiers, spécialement dans le cas des incendies. Dans ce chapitre, nous présenterons d'abord les définitions et concepts de base pour la compréhension du mémoire, suivis des éléments de la problématique que nous chercherons à traiter et les objectifs ainsi définis, pour terminer avec le plan du mémoire dans son ensemble.

1.1 Définitions et concepts de base

Dans cette section, nous définissons les concepts de gestion des incendies, de villes intelligentes et de bâtiments intelligents, ainsi que la technologie *« Building Information Model » (BIM)*, les objets connectés et le calcul en temps réel.

Gestion des incendies

L'opération de *gestion des incendies* est menée par les sapeurs-pompiers et vise de manière générale deux grands objectifs [1] : le premier est l'*évacuation* de toutes les personnes du bâtiment en feu et le second est de contrôler et éteindre le feu. L'*évacuation* des occupants est la phase la plus importante. Elle consiste à trouver ou localiser les personnes dans le bâtiment puis à les guider à travers un chemin d'évacuation. Ceci peut paraître simple, mais en situation de feu, certains chemins sont inaccessibles dus à la haute température ou à l'écroulement du bâtiment, ainsi qu'à la fumée qui réduit fortement la visibilité [2], [3]. Il existe plusieurs stratégies d'évacuation comme l'*évacuation simultanée* et l'*évacuation graduelle verticale* [4]. L'un des principaux défis rencontrés par les sapeurs-pompiers est le manque d'information dont ils disposent en arrivant sur la scène de

feu [5]. Cependant, avec l'avènement des *villes intelligentes* et des *objets connectés* une grande quantité d'information devient accessible aux sapeurs-pompiers.

Les villes intelligentes et les bâtiments intelligents

Les *villes intelligentes* sont des villes où d'énormes quantités d'informations sont collectées et analysées, afin de permettre un fonctionnement plus efficace de la ville. Ceci est possible par l'utilisation combinée des *Technologies de l'Information et de la Communication (TIC)* ainsi que l'Internet des objets (IoT pour Internet of Things) [6]. Un sous-ensemble des villes intelligentes est le *bâtiment intelligent* appelé en anglais *« Smart Building »*. Le concept du *bâtiment intelligent* est le même que celui des *villes intelligentes,* mais appliqué à l'échelle d'un immeuble [7]. En combinant les TICs et l'IoT, on collecte de nombreuses informations sur le bâtiment telles que la température, la qualité de l'air et la consommation électrique. Ces données sont ensuite analysées pour améliorer le fonctionnement du bâtiment, plus spécifiquement pour réduire la consommation d'énergie.

La technologie « Building Information Model » (BIM)

En plus d'avoir des *bâtiments intelligents*, il est aujourd'hui possible de créer une représentation numérique des bâtiments grâce à la technologie *« Building Information Model » (BIM)* [7]. Les données BIM sont représentées sous le format *« Industry Foundation Classes » (IFC)* [8]. Ce format a été mis en place pour la construction et la gestion de structures telles que les bâtiments et les ponts [9]. Cependant, certains auteurs [10] ont proposé de l'étendre afin qu'il soit plus utile aux situations de feu. Grâce au BIM, il est possible d'afficher les bâtiments en 3 dimensions (3D) et les manipuler à l'aide de logiciels.

La Figure 1.1 montre l'utilisation du BIM dans la phase de conception et de post-construction d'un bâtiment intelligent [7]. Durant la phase de conception, le BIM fournit une visualisation 3D du bâtiment et joue le rôle d'un moyen de communication entre les différents acteurs du projet (e.g., Client/Utilisateur et Concepteur/Ingénieur). Dans la phase de post-construction, qui vient après que le bâtiment eut été achevé et mis en service, le BIM est utilisé lors des opérations de gestion (e.g., restauration, amélioration, incendie, etc.).

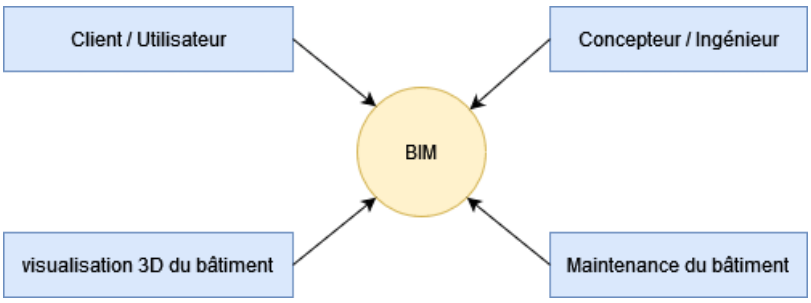

Figure 1.1 Rôle du BIM dans la conception et la gestion des bâtiments intelligents

Les objets connectés

Les objets connectés *« Internet of Things » (IoT)* consistent à interconnecter de multiples dispositifs électroniques analogiques et numériques, homogènes ou hétérogènes par nature, mais dont la portée de transmission se chevauche, afin qu'ils puissent communiquer efficacement des informations [11]. Grâce à ces objets, il est possible de placer des capteurs à l'intérieur des bâtiments en feu pour collecter de nombreuses données qui seront utiles aux sapeurs-pompiers.

Le calcul en temps réel

La gestion en *temps réel* des incendies consiste à assister les sapeurs-pompiers de manière continue durant leur intervention. Ceci nécessite une collecte et une analyse des données en temps réel. L'analyse des données en temps réel est appelée calcul en temps réel *« Real Time Computing » (RTC)*. De manière spécifique, ceci consiste à collecter les données, les traiter et les afficher dans un temps très restreint qui est de l'ordre de la milliseconde ou de la microseconde [12], [13].

1.2 Éléments de la problématique

Chaque année, nous observons des milliers d'incendies résidentiels et industriels. Ces incendies causent de nombreux décès et dommages matériels. Par exemple, selon le Ministère de la Sécurité Publique du Québec, il y a en moyenne 16 500 incendies par année au Québec, dont au moins 5 600 sont des incendies résidentiels. Entre 2015 et 2017, les incendies ont causé 131 décès et 540 millions CAD de dommage en moyenne par année [14]. En Ontario, il y avait 11 046 incendies avec 91 décès et 836.1 millions de dollars CAD de dommages en 2018, et 10 645 incendies avec

67 décès et 968.9 millions de dollars CAD de dommage en 2019 [15]. Les sapeurs-pompiers sont les premiers à intervenir en cas d'incendies ou de situations d'urgence. Leur rôle est de prévenir les pertes en vies humaines et en biens matériels. Avec l'avènement de la ville intelligente et des bâtiments intelligents, de nouvelles technologies et opportunités s'offrent aux sapeurs-pompiers.

L'émergence de la ville intelligente signifie l'accès à l'*Internet haut débit (5G)* et aussi la possibilité de connecter des milliards d'objets électroniques à Internet, créant ainsi les *objets connectés « Internet of Things » (IoT)* [3], [16]-[20]. L'émergence des *bâtiments intelligents « Smart Building »* signifie la vulgarisation de la technologie *« Building Information Model » (BIM)* [7]. Le *BIM* consiste à la représentation numérique complète d'un bâtiment. Il fournit un modèle 3D du bâtiment avec des informations (e.g., matériaux, fabricants, etc.) sur chaque partie du bâtiment. Un autre aspect des bâtiments intelligents est la présence de capteurs (e.g., détecteur d'incendie, température, qualité de l'air, etc.) collectant de nombreuses données sur le bâtiment en temps réel. Ces données sont utilisées pour automatiser certains aspects du bâtiment comme la consommation d'énergie et les alertes d'urgence. Enfin, des recherches récentes sur la simulation d'incendie par le *« National Institute of Standards and Technology » (NIST)* ont abouti à la création du simulateur *« Fire Dynamic Simulation » (FDS)*. La simulation *FDS* utilise un modèle mathématique qui permet une simulation de haute précision des événements durant un incendie à l'intérieur d'un bâtiment [21]-[31]. Avec toutes ces technologies prometteuses - *Smart Building*, *5G*, *IoT*, *FDS* et *BIM* – nous pensons qu'il est possible d'améliorer les modèles de gestion des incendies dans les villes intelligentes notamment au niveau de la détection des incendies, de l'évacuation des occupants à l'intérieur des bâtiments, de la navigation des sapeurs-pompiers à l'intérieur des bâtiments, de la simulation des incendies et de la collecte de données des incendies.

Tenant compte de ces éléments de problématique, notre question principale de recherche est la suivante : Comment pouvons-nous améliorer la gestion des incendies dans les villes intelligentes afin de réduire les pertes de vie humaine et les dommages économiques? De cette question principale découlent les deux questions secondaires suivantes :

1. Comment assister les sapeurs-pompiers en temps réel en leur fournissant des informations utiles et des plans d'actions efficaces?
2. Comment garantir et maintenir un fonctionnement de l'assistance en temps réel malgré la complexité des calculs effectués?

1.3 Objectifs de recherche

L'objectif principal de ce mémoire est de proposer un modèle de gestion des incendies en temps réel qui utilise les technologies de la ville intelligente afin de réduire les pertes de vie humaine et les dommages économiques. De manière plus spécifique, ce mémoire vise à :

- Analyser les modèles de gestion des incendies existants pour en déceler les lacunes et définir les requis du nouveau modèle proposé;
- Concevoir un modèle de gestion des incendies basé sur l'IoT pour fournir de l'information en temps réel aux sapeurs-pompiers qui sont dans le bâtiment en feu ou au centre de commande;
- Implémenter le modèle de gestion des incendies proposé;
- Évaluer la performance du modèle proposé pour en démontrer la robustesse et la conformité par rapport à un ensemble de requis.

1.4 Plan du mémoire

Ce mémoire comprend cinq chapitres. Suivant ce premier chapitre d'introduction, le deuxième chapitre présente une revue de littérature approfondie sur les solutions existantes pour la gestion des incendies dans les villes intelligentes et souligne l'apport de certaines solutions dans les bâtiments intelligents avec leurs limites. Le troisième chapitre définit le modèle proposé de la gestion des incendies en temps réel dans les villes intelligentes avec les différentes technologies utilisées : le calcul en temps réel, les systèmes distribués, les microservices et les bases de données en temps réel. Le quatrième chapitre fournit les détails sur l'implémentation de la partie distribuée du modèle proposé et de l'interface Web utilisée par ce modèle, ainsi que l'évaluation de performance de ce modèle. Le cinquième chapitre, en guise de conclusion, fait la synthèse des travaux, en expose les limitations et offre quelques indications de recherche future.

CHAPITRE 2 GESTION DES INCENDIES DANS LES VILLES INTELLIGENTES

Dans ce chapitre, nous ferons une revue détaillée de la gestion des incendies dans *les villes intelligentes*. Dans un premier temps, nous présenterons les incendies urbains et le rôle des sapeurs-pompiers. Puis, nous illustrons les défis rencontrés par les sapeurs-pompiers. Ensuite, nous détaillerons les techniques et les méthodes existantes utilisées dans la gestion des incendies. Enfin, nous présenterons les limitations de ces techniques et de ces méthodes.

2.1 Les incendies urbains et les sapeurs-pompiers

Les incendies urbains sont des incendies qui se produisent en pleine ville. Selon le rapport 2020 du Service de Sécurité Incendie de Montréal (SIM), le nombre d'incendies de bâtiments à Montréal était comme suit : 1 172 en 2018, 1 208 en 2019 et 1 385 en 2020 [32]. La raison des incendies est variée. Ces incendies peuvent être causés par les matériaux inflammables tels que le bois, mais le plus souvent il s'agit du feu de cuisson [33]. Selon le SIM, un tiers (1/3) des incendies résidentiels à Montréal proviennent de la cuisine [28]. Il y a plusieurs types d'incendies urbains. Les incendies de bâtiments ou incendies structurels sont des incendies qui se produisent à l'intérieur d'un bâtiment. En fonction de l'usage du bâtiment, un incendie structurel peut être classifié comme résidentiel, commercial ou industriel. Un incendie résidentiel peut concerner plus d'un bâtiment à la fois, et nécessiter l'intervention de plus d'une centaine de sapeurs-pompiers [9]. En plus des incendies structurels, il y a les incendies de voitures et les incendies extérieurs qui se produisent à l'extérieur de bâtiments (i.e., dans un coin de rue ou dans un parc).

Les principaux dommages causés par les incendies sont la perte de vies humaines et les pertes financières. Selon le Ministère de la Sécurité publique du Québec, il y a en moyenne 16 500 incendies par années au Québec dont 5600 sont des incendies résidentiels. Entre 2015 et 2017, les incendies ont causé 131 décès et 540 millions CAD de dommage par ans [14]. En Ontario, en 2018 il y avait 11 046 incendies avec 91 décès et 836.1 millions CAD de dommages, et en 2019 il y avait 10 645 incendies avec 67 décès et 968.9 millions CAD de dommage [15].

Les principaux acteurs de la lutte contre le feu sont les sapeurs-pompiers aussi appelés premiers intervenants. Les sapeurs-pompiers disposent de plusieurs stations ou casernes à travers la ville. A Montréal par exemple, il y a 61 casernes couvrant toute la ville [32]. Lorsqu'un incendie se déclare, les sapeurs-pompiers les plus proches sont envoyés sur la scène. Le rôle des sapeurs-pompiers est de sauver des vies en évacuant les bâtiments en feu, puis de contrôler et éteindre le feu. De plus, les sapeurs-pompiers s'occupent de la prévention contre les incendies et de l'analyse des origines des feux. En outre, ils interviennent dans des incidents où il n'y a pas de feu, tels que les accidents de voitures ou les effrondrements de bâtiments.

2.2 Concepts reliés aux villes et bâtiments intelligents

Les *bâtiments intelligents* sont un sous-composant des *villes intelligentes* [36]. Ces dernières se composent de la maison intelligente « Smart Home », du *bâtiment intelligent « Smart Building »*, de la mobilité intelligente « Smart Mobility », du parking intelligent « Smart Parking », du transport intelligent « Smart Transport », de la santé intelligente « Smart Health », de la gouvernance intelligente « Smart Governance » et du réseau électrique intelligent « Smart Metering/Grid ». Dans cette section, nous détaillerons les spécificités des *bâtiments intelligents*.

L'avènement des *villes intelligentes* et des *bâtiments intelligents* s'accompagnent de l'émergence de nouvelles technologies tels que [6] :

- la numérisation du bâtiment;
- la modélisation des données du bâtiment avec le BIM;
- l'Internet à très haut débit avec la technologie 5G;
- les objets connectés (IoT);
- le cloud et l'edge computing;
- l'intelligence artificielle et l'apprentissage automatique.

Les *bâtiments intelligents* utilisent les *objets connectés* pour surveiller divers éléments du bâtiment, analyser les données et générer des informations qui peuvent être utilisées pour optimiser les opérations du bâtiment [37]. Les *bâtiments intelligents* utilisent les 7 technologies suivantes [37] : Systèmes de gestion des bâtiments, capteurs intelligents, système de positionnement intérieur, cartographie numérique intérieur, éclairage intelligent, système de chauffage, ventilation et climatisation, et enfin système de gestion des incendies.

L'objectif d'un système de gestion des bâtiments « Building Management Systems » (BMS) traditionnels est de fournir assez de données afin d'automatiser les principales opérations d'un bâtiment. Aujourd'hui, l'IoT permet aux bâtiments d'aller plus loin dans ce processus pour collecter des données plus détaillées et fournir des rapports analytiques. Un énorme avantage d'un BMS moderne est la capacité de collecter des données en temps réel. Ceci permet aux gestionnaires d'installations de prendre des mesures dès qu'un problème ou une opportunité est signalé.

Les capteurs intelligents sont au cœur de nombreux *bâtiments intelligents* et sont utilisés pour déclencher une action ou pour collecter des données sur les conditions actuelles d'une pièce. Il existe de nombreux types de capteurs intelligents. Nous pouvons citer :

- les capteurs de température;
- les capteurs d'humidité;
- les capteurs de qualité de l'air;
- les capteurs de qualité de l'eau;
- les capteurs de mouvement;
- les capteurs de proximité;
- les capteurs optiques.

Un système de positionnement intérieur « Indoor Positioning Systems » (IPS) offre deux avantages essentiels. Le premier avantage est l’expérience améliorée pour les occupants et les visiteurs, et le deuxième est la collecte de données approfondies et précises sur le trafic, l'occupation et l'utilisation du bâtiment. L'IPS, comme les capteurs intelligents, peut être utilisé pour collecter des données sur le mouvement des appareils dans le bâtiment, ainsi que pour fournir des services supplémentaires telles que des indications détaillées. Un IPS peut être mis en place par l'installation de matériel (i.e., des balises) ou par l'utilisation des téléphones iOS ou Android.

Une cartographie numérique intérieure fournit des informations utiles aux visiteurs, aux employés et aux équipes d'entretien, ainsi qu'une orientation intuitive et actualisée vers les différents points d'intérêt du bâtiment. Elle est utilisée en combinaison avec le positionnement intérieur ou d'autres technologies de *bâtiments intelligents* pour permettre aux locataires et aux gestionnaires de bâtiments de gagner en efficacité et en expérience.

L'éclairage intelligent peut générer d'énormes économies d'énergie et réduire les coûts d'exploitation. Des capteurs optiques peuvent être utilisés pour déterminer le niveau d'éclairage nécessaire dans une pièce lorsque le soleil commence à se coucher, ou pour éteindre complètement les lumières si une salle de réunion est inoccupée.

Le système de chauffage, ventilation et climatisation (CVC) d'un *bâtiment intelligent* peut réduire les coûts d'exploitation, améliorer le contrôle des différents systèmes du bâtiment et créer des automatismes plus intelligents. Des capteurs peuvent être utilisés pour détecter les niveaux variables de plusieurs facteurs, notamment l'occupation et le CO2, dans l'ensemble du bâtiment, et pour régler avec précision la température, l'humidité et le débit d'air en conséquence. En surveillant la consommation et l'utilisation de l'énergie, les propriétaires d'immeubles peuvent être plus stratégiques en matière de chauffage et de refroidissement des différents espaces de travail.

Les capteurs et les dispositifs IoT peuvent mieux surveiller les *bâtiments intelligents* pour détecter les incendies plus rapidement, fournir aux centres de commandement des incidents davantage d'informations, améliorer la répartition assistée par ordinateur, améliorer la connaissance de la situation pour les sapeurs-pompiers une fois qu'ils sont sur les lieux d'un incendie et aider à l'extinction des incendies sous la forme de sprinklers intelligents.

2.3 Gestion des incendies

L’objectif de la gestion des incendies est de sauver des vies ainsi que de réduire les pertes financières dues aux dommages causés par le feu. Pour ce faire, les sapeurs-pompiers doivent prendre les décisions appropriées très rapidement. Afin de les aider dans leur prise de décision, les systèmes de gestion des incendies collectent, analysent et présentent l’information utile aux sapeurs-pompiers. La Figure 2.1 montre le processus en quatre étapes de la gestion des incendies qui seront détaillées plus tard dans ce chapitre :

- Étape 1 : La collecte des données;
- Étape 2 : L’analyse des données;
- Étape 3 : La présentation de l’information;
- Étape 4 : La prise de décision.

Dans la suite, nous soulignerons l'importance d'avoir les bonnes informations, puis nous illustrerons comment les solutions existantes abordent les 3 premières étapes. L'Annexe B résume les travaux les plus importants qui ont été réalisés.

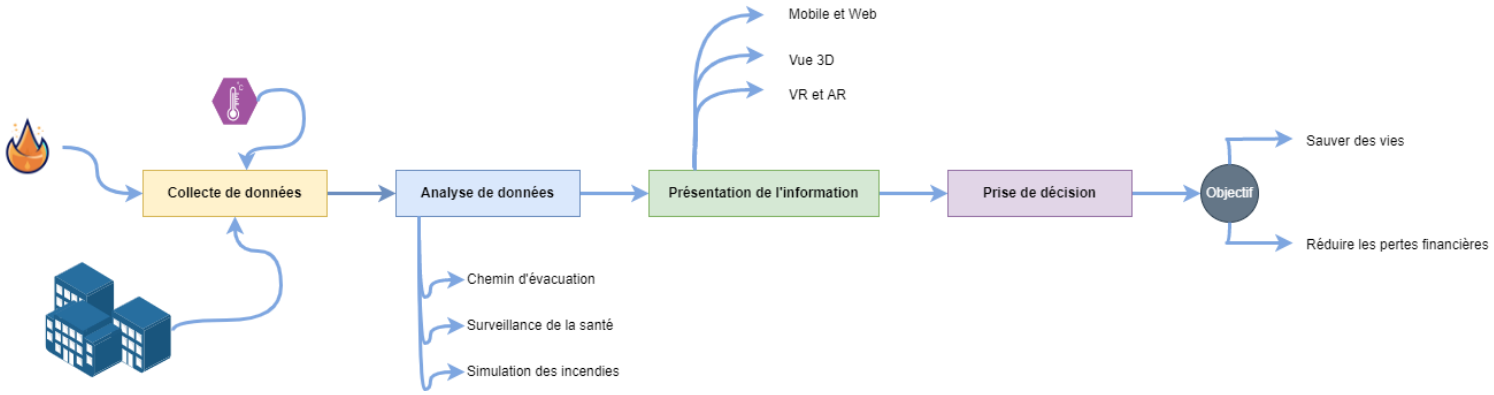

Figure 2.1 Processus de gestion d'incendie

2.3.1 La contrainte de temps et l'importance de l'information

En cas d'incendie, selon le « National Fire Protection Association » (NFPA) [38], les deux principales causes de décès sont la haute température et la fumée. De plus, l'exposition à la fumée peut tuer en 10 minutes, ainsi que le corps humain peut supporter jusqu'à 100 degrés Celsius. Cependant, en cas d'incendie, la température peut atteindre 260 degré Celsius en 3 ou 4 minutes et 593.33 au-delà de 5 minutes. Les études sur le déroulement des incendies [38], [33] montrent que les sapeurs-pompiers disposent d'environ 10 minutes pour sauver le maximum de personnes. La Figure 2.2 illustre la progression d'un incendie dans le temps à partir du moment où l'alarme d'incendie est déclenchée [38]. De plus, les sapeurs-pompiers arrivent sur la scène entre 4 et 6 minutes et le feu atteint son paroxysme ou « Embrasement Généralisé Éclair » (EGE) (i.e., Flashover en anglais) en 10 minutes. Au-delà de 10 minutes, les chances de survie sont presque nulles et le bâtiment aura subi au moins 50 % de dommages.

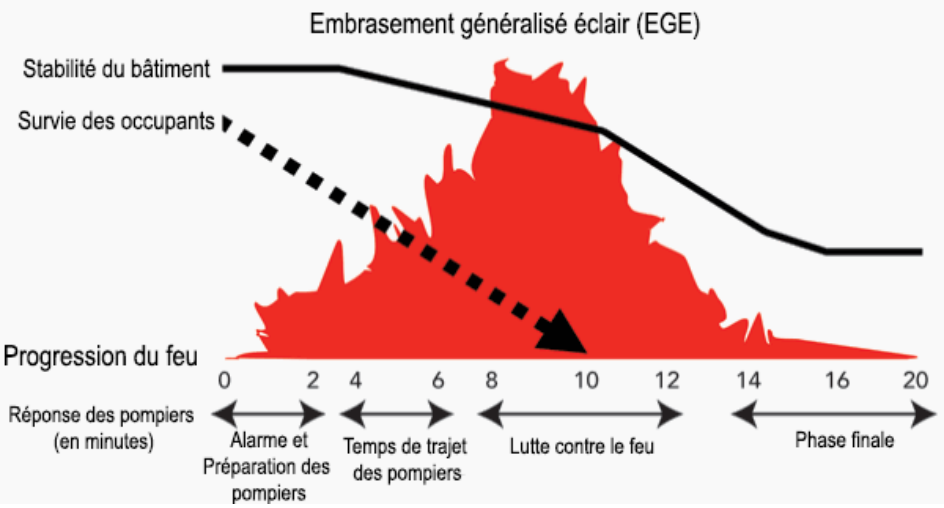

Figure 2.2 Progression d'un incendie dans le temps

La Figure 2.2 ne montre que des valeurs moyennes, dans certaines situations le « Flashover » peut intervenir en très peu de temps comme le cas lors de l'incendie du club « The Station » [39]. Le Tableau 2.1 résume l'incendie tel que décrit dans le rapport du NIST [39].

Tableau 2.1 Incendie du club The Station

Lieu	West Warwick, Rhode Island, United Station
Date	20 février 2003
Flashover	1 minute
Black smoke	2 minutes
Appel au 911	Moins de 40 secondes après l'alarme
Arrivée des sapeurs-pompiers	Moins de 5 minutes après l'appel au 911
Nombre de morts	100

La monté de la fumée (CO et CO2) et l'absence d'oxygène peut entraîner la mort en quelques minutes, comme le montre les tableaux 2.2, 2.3 et 2.4.

Tableau 2.2 Effet de l'absence d'oxygène sur le corps humain

Taux d'oxygène dans l'air (%)	Symptômes
21	Conditions normales et aucun effet
19	Atmosphère déficient en oxygène selon le OSHA (Occupational Safety and Health Administration)
17.5	Affaiblissement musculaire et respiration rapide
12	Vertiges, maux de tête et fatigue rapide
9	Inconscience
7 à 6	Mort en quelques minutes

Tableau 2.3 Effet de l'augmentation du taux de carbone dans l'air

Taux de CO dans l'air (%)	Symptômes
0.2	Maux de tête après 10 minutes, effondrement après 20 minutes et mort après 45 minutes
0.3	Peu d'effets sur 5 minutes et danger d'effondrement à 10 minutes
0.6	Maux de tête et vertiges en 1 ou 2 minutes et danger de mort entre 10 à 45 minutes
1.28	Effet immédiat, inconscience après 2 ou 3 respirations, danger de mort entre 1 et 3 minutes

Tableau 2.4 Effet de l'augmentation du CO dans le sang

Taux de CO dans le sang (COHb - Carboxyhemoglobin) (%)	Symptômes
10	Pas de symptômes
15	Léger maux de tête
25	Nausée et sérieux maux de tête
30	Intensification des symptômes précédents
45	Inconscience
50	Mort

Toutes les données ci-dessus nous montrent que la survie des occupants dans un bâtiment en feu dépend de la rapidité d'intervention des sapeurs-pompiers. Cependant, arrivés sur les lieux, les sapeurs-pompiers disposent de très peu de temps pour mettre en place un plan d'action et agir. Le principal défi dans cette situation est de prendre les bonnes décisions dans un délai de temps très court. La prise de bonnes décisions dépend de l'information mise à disposition des sapeurs-pompiers. Plus ils disposent d'informations utiles sur la situation, plus ils seront en mesure d'agir le plus efficacement possible. Le fait de disposer des informations utiles sur la situation de feu est reconnu selon le modèle « Situational awareness » qui est défini par Endsley [40], et dont il y a trois composantes à prendre en compte : la perception des éléments de la situation actuelle, la compréhension de la situation actuelle et la projection de la situation future. En 2004, le NIST a organisé un atelier pour identifier les besoins en information lors d'intervention d'urgence dans un bâtiment [5]. Les informations dont les sapeurs-pompiers ont besoin le plus sont résumées dans l'Annexe A.

Plusieurs études ont été menées pour comprendre l'impact de l'information sur la décision en cas d'incendie [41]-[43]. Lim *et al.* [42] ont mené une expérience dans laquelle une grande quantité d'informations sur un incendie a été fournie aux sapeurs-pompiers pour qu'ils puissent bien mener leurs activités de sauvetage. L'expérience montrée qu'avec la bonne information, le temps d'évacuation était réduit. Peeters *et al.* [43] ont mené une expérience pour mesurer l'impact de l'information sur le choix de sortie des victimes d'un bâtiment en feu. Sept étages identiques d'un bâtiment ont été utilisés avec des configurations différentes pour voir si le choix de la sortie est influencé par les informations fournies au moment de l'alarme. Les auteurs ont constaté que l'information a un impact significatif sur le choix de la sortie, la vitesse de fuite et la distance parcourue. En outre, ils ont démontré que les fausses informations peuvent augmenter le temps nécessaire pour quitter le bâtiment et la distance parcourue, ce qui a un impact sur le taux de survie.

2.3.2 Les types des données collectées

La collection des données constitue la première opération des modèles de gestion d'incendies. Dans cette section, nous verrons en détail les types de données collectées par les modèles existants dans la littérature comme suit :

- les données sur la structure du bâtiment;
- les données médicales des sapeurs-pompiers;

- les données image et vidéo;
- les données IoT.

Ces données sont des données brutes collectées sur la scène de feu, dont le but étant de les analyser et d'en extraire des informations plus utiles aux sapeurs-pompiers.

a) Les données sur la structure du bâtiment

Les données sur la structure du bâtiment consistent en général sur les plans du bâtiment. Ces plans peuvent être en deux dimensions (plan 2D) ou en trois dimensions (plan 3D). Les modèles dans la littérature se focalisent plus sur l'utilisation des données 3D aussi appelées données *« Building Information Modèle » (BIM)*. Les données BIM sont sous le format « *Industry Foundation Classes* » *(IFC)* [8], [31]. Ce format a été mis en place pour la construction et la gestion de structures telles que les bâtiments et les ponts [9]. Cependant, certains auteurs [10] ont proposé de l'étendre pour qu'il soit plus utile aux situations de feu. En combinant les données BIM aux données « Geographic Information System » (GIS), on peut obtenir une vue numérique d'un bâtiment ainsi que tout le quartier qui l'entoure [44]. Les données BIM sont utiles à tous les stages de construction et de gestion d'un bâtiment (i.e., planification, désigne, construction, opération) [45]. Dans le cas de la gestion de feu dans les bâtiments, les données BIM peuvent être utilisées durant le design des mesures de sécurité, pour l'entraînement des sapeurs-pompiers et l'évacuation des personnes en cas d'incendies [46]. Sun *et al*. [47] ont utilisé les données BIM pour simuler des scénarios de feu et calculer la meilleure stratégie d'évacuation. Les occupants du bâtiment sont considérés comme des agents et leur comportement est analysé à l'aide de la technique *« Agent Based Modeling (ABM) »*. Liu *et al*. [48] ont utilisé les données BIM pour détecter les dangers tel que le feu dans un bâtiment. Kanak *et al*. [49] ont profité de l'utilisation efficace des données BIM et des données GIS dans les villes, afin de gérer des catastrophes comme le feu. Peijun *et al.* [50] ont démonté l'importance de la technologie BIM dans la phase de conception et dans la phase de gestion de l'exploitation et de la maintenance des bâtiments. Dans la phase de conception, le BIM est utilisé pour la simulation, afin de fournir des références et des suggestions pour la conception du bâtiment du point de vue de la sécurité contre les incendies. Dans la phase de gestion de l'exploitation et de la maintenance, le BIM permet la simulation des évacuations d'urgence en cas d'incendie.

b) Les données médicales des sapeurs-pompiers

Les sapeurs-pompiers sont au cœur de la lutte contre les incendies, faire contrôler leur santé durant l'incendie est donc très important. Raj *et al.* [51] ont proposé un système pour mesurer en temps réel le stress des sapeurs-pompiers. Pour cela, ils utilisent des capteurs placés dans les gants des sapeurs-pompiers, tels que le capteur de battement de cœur et le capteur « Galvanic Skin Response » (GSR).

Le capteur GSR mesure les changements dans l'activité des glandes sudoripares qui reflètent l'intensité de notre état émotionnel. En combinant ces données avec celles du battement du cœur les auteurs ont évalué le niveau de stress des sapeurs-pompiers en temps réel.

c) Les données images et vidéos

La plupart des bâtiments disposent de caméras de surveillance vidéo appelée « Closed Circuit Television » (CCTV) camera. Bien que ces caméras soient utilisées principalement pour la sécurité, elles peuvent aussi servir en cas d'incendie. Plusieurs auteurs [3], [52] ont essayé d'exploiter ces données en utilisant des techniques de traitement d'image et de vidéos. Par exemple, Jack *et al*. [52] ont utilisé les flux vidéos provenant des caméras en temps réel et des algorithmes d'apprentissage profond « Deep-Learning » pour détecter et compter le nombre de personnes dans le bâtiment en feu.

d) Les données IoT

Les données IoT sont très utilisées dans les modèles existants [16]-[19], [53]. Ces données proviennent de capteurs placés de manière stratégique à l'intérieur du bâtiment. Les données collectées sont des informations comme la température et le taux de Co2 dans l'air. Les principaux capteurs utilisés dans la littérature sont les suivants :

- capteur de fumée;
- capteur de feu;
- capteur de CO;
- capteur de CO2;
- capteur de qualité de l'air;
- capteur de température;
- capteur d'humidité.

Kodali *et al*. [54] ont proposé un modèle de gestion d'incendie qui utilise un capteur de détection de flamme, un capteur de détection de fumée (MQ-5) et un capteur de détection de gaz inflammable, afin de détecter le danger lier au feu. À la détection du danger, le modèle alerte la police et les sapeurs-pompiers. Fong *et al.* [53] ont défini le concept d'« Internet of Breath » (IoB). Le modèle IoB utilise un ensemble de capteur IoT pour détecter l'occupation humaine dans une zone confinée, tout en mesurant la concentration de CO2 dans la zone concernée. Le modèle permet ainsi de localiser la concentration de personnes dans un bâtiment en feu et d'en aviser les sapeurs-pompiers. Cavalera *et al.* [55] ont créé leur propre conteneur IoT appelé « Smart Box ». Le Smart Box est un dispositif intelligent capable de gérer plusieurs capteurs environnementaux qui lui sont connectés. Il comprend six capteurs différents : humidité, température, monoxyde de carbone, fumée, présence (i.e., capteur Infrarouge Passif (IRP)) et pression (capteur GPL).

2.3.3 Traitement des données et services offerts

Une fois les données collectées, elles sont analysées pour extraire l'information utile et fournir certains services aux sapeurs-pompiers. Dans cette section, nous détaillerons les six types : d'analyses effectuées par les modèles existants dans la littérature :

- le calcul des chemins d'évacuation;
- la surveillance de la santé des sapeurs-pompiers;
- la détection du point d'origine du feu;
- la détection et la classification des dangers;
- la collection et la distribution de l'information en temps réel;
- la simulation d'incendie.

a) Calcul des chemins d'évacuation

La principale mission des sapeurs-pompiers est de sauver les personnes emprisonnées dans les bâtiments en feu, pour cela ils doivent les évacuer du bâtiment. Afin de faciliter cette tâche, certains auteurs [2], [3], [52] ont proposé des modèles qui calculent le meilleur chemin vers la sortie du bâtiment. La performance des systèmes d'évacuation dans un bâtiment peut être mesurée avec les valeurs « Available Safe Egress Time » (ASET) et « Required Safe Egress Time » (RSET) [56]. L'ASET ou temps d'évacuation disponible est le temps qui s'écoule entre l'allumage du feu et le développement des conditions intenables pour les occupants (e.g., très haute température). Le

RSET ou temps d'évacuation requis est le temps, à partir de l'allumage du feu, nécessaire aux occupants pour évacuer le bâtiment et atteindre l'extérieur du bâtiment ou une enceinte de sortie protégée. Cependant, les conditions sont favorables lorsque l'ASET est supérieur au RSET. Il est important de noter que les RSET changent en fonction du nombre de sorties, de la largeur des voies et de la capacité physique des occupants à se déplacer. Les méthodes de calcul existantes pour les chemins d'évacuation peuvent se classer en trois catégories :

- ceux calculant les chemins les plus rapides (A* et Dijkstra) [2], [57], [58];
- ceux tenant en compte du comportement social de l'être humain (ABM, comportement de foule) [3], [27], [30], [59];
- ceux pouvant ajuster le chemin en temps réel pour éviter les obstacles [26], [60], [61].

Chemins les plus rapides

Wang *et al.* [2] ont proposé une solution utilisant l'algorithme Dijkstra pour calculer les différents chemins possibles pour l'évacuation dans le cas d'une seule personne à évacuer et dans le cas de plusieurs personnes à évacuer. Leur modèle permet l'ajustement en temps réel du chemin d'évacuation en fonction de la propagation du feu. Le modèle est vérifié à l'aide de trois scénarios de feu. Le premier scénario consiste en une source de feu et une seule personne à évacuer. Le deuxième scénario consiste à plusieurs sources de feu et une seule personne à évacuer. Le troisième scénario consiste en plusieurs sources de feu et plusieurs personnes à évacuer. Dans le cas de l'évacuation de plusieurs personnes, le modèle est capable de prédire les points de congestion. Jack *et al.* [52] ont proposé de calculer un chemin d'évacuation en utilisant un réseau à base de graphes intégrant « Medial Axis Transform » (MAT) et « Visibility Graphs » (VG). Leurs résultats montrent que la combinaison de MAT et VG donne de meilleurs résultats que les utiliser séparément. Cependant, l'utilisation de MAT implique un nombre de nœuds=15, un temps d'exécution=0.79ms, et une distance totale=148, l'utilisation de VG donne un nombre de nœuds=24, un temps d'exécution=1.43ms, et une distance totale=122, et la combinaison MAT+VG résulte un nombre de nœuds=20, un temps d'exécution=1.08ms, et une distance totale=128. Les chemins les plus rapides ne sont pas très efficaces car les occupants du bâtiment ont tendance à avoir un comportement de groupe. Ceci mène à la mise en œuvre des solutions prenant en compte le comprenant social.

Prise en compte du comportement social

Dugdale *et al.* [3] ont proposé un système d'évacuation qui prend en compte le comportement social des occupants du bâtiment. En conséquence, ils ont combiné un algorithme d'optimisation de flux de réseau avec une simulation sociale basée sur des agents pour fournir une évacuation plus réaliste. Les occupants individuels sont modélisés comme des agents informatiques dans le simulateur. Les agents sont hétérogènes et présentent un attachement social aux autres et des variations dans la vitesse de déplacement. Une architecture d'agent Belief-Desire-Intention (BDI) est utilisée pour modéliser le raisonnement cognitif des agents individuels. Les résultats montrent que l'attachement social ralentit l'évacuation et que les portes doubles internes placées adjacentes sont plus efficaces qu'une seule large porte double.

Chemins d'évacuation en temps réel

Adriana *et al.* [62] ont proposé un système intelligent de guidage de l'évacuation pour les bâtiments complexes. Cependant, ils ont calculé les itinéraires d'évacuation optimaux à partir de simulations en temps réel pour guider les évacués par une signalisation dynamique à travers les chemins disponibles les plus sûrs et les plus rapides. Le modèle proposé permet de réduire le temps d'évacuation de de 28,41 % à 59,79 %. Hong-Hsu *et al.* [63] ont développé un algorithme intelligent permettant d'identifier des chemins d'évacuation en fonction du temps et de la température. Les capteurs de température dotés d'une capacité de communication peuvent aider à surveiller et à communiquer les valeurs de température à une station de contrôle. Ceci permet d'établir des voies d'évacuation dynamiques et en temps réel en cas d'incendie. Six algorithmes d'évacuation (i.e., « Best-First Search» (BFS), « Shortest Path » (SP), « Dynamic Best-First Search » (DBFS), « Temperature Aware Best-First Search » (TABFS), « Temperature Aware Shortest Path » (TASP) et « Temperature Aware Dynamic Best-First Search » (TADBFS)) sont proposés pour identifier le chemin d'évacuation le plus efficace. Les trois premiers algorithmes, qui ne prennent pas en compte les contraintes de température humaine (i.e., BFS, SP, DBFS), peuvent être utilisés par les services de secours. Les trois derniers algorithmes qui prennent en compte les contraintes de température humaine (i.e., TABFS, TASP, TADBFS) peuvent être utilisés par les personnes à évacuer.

b) **La surveillance de la santé des sapeurs-pompiers**

Afin de veiller à ce que les sapeurs-pompiers qui entre dans le bâtiment en feu puissent revenir sain et sauf, des modèles ont été mis en place pour faire le suivi de leur position et mesurer leur état physique. Li *et al.* [64] ont proposé un modèle permettant de faire le suivi des sapeurs-pompiers. Leur solution utilise la recherche taboue et atteint une précision moyenne au niveau de la pièce de 87,1 %. Raj *et al.* [51] ont proposé un système pour mesurer en temps réel le stress des sapeurs-pompiers. Leur solution utilise les données de battement de cœur et des glandes sudoripares pour calculer le stress des sapeurs-pompiers et leur fournir l'information. Ceci permet de détecter les sapeurs-pompiers en détresse pour leur venir en aide.

c) La détection du point d'origine du feu

Savoir le point d'origine du feu est très important pour les sapeurs-pompiers, ceci leur permet de mieux planifier l'évacuation des occupants du bâtiment et de contrer la prorogation du feu. Wu *et al.* [17] ont mis en place un modèle pouvant détecter l'origine et la tailler dans un milieu étroit à savoir les tunnels. Pour ce faire, ils ont construit une grande base de données d'incendies de tunnel avec des emplacements des incendies, des tailles des incendies et des conditions de ventilation. En combinant ces données avec celles du capteur de températures distribuées dans le tunnel, ils ont construit un modèle intelligent de réseaux de neurones utilisant « Long-Short Term Memory Recurrent Neural Network » (LSTM-RN). Leur modèle prédit l'emplacement et la taille de l'incendie dans le tunnel et la vitesse du vent de ventilation avec une précision de 90 %.

d) La détection et la classification des dangers

Zhansheng *et al.* [48] ont proposé un système de détection et de classification des dangers dans le bâtiment en feu. Cependant, ce système est basé sur la technique de l'apprentissage automatique « Support Vector Machine » (SVM) avec « Radial Basis Function » (RBF) et « K-fold Cross-Validation ». Ils ont obtenu de bon résultats au niveau de la détection du feu (accuracy=100, precision=100, recall=100), la détection de surpopulation (accuracy=99.25, precision=97.07, recall=99.38), et la détection des entrées illégales (accuracy=97.57, precision=98.15, recall=99.25).

e) La collection et la distribution de l'information en temps réel

Wang *et al.* [65] ont proposé un modèle de collection et de distribution en temps réel de l'information en utilisant la technologie « Narrowband IoT » (NB-IoT). L'information collectée provient d'un capteur intelligent basé sur le contrôleur STM32. Le NB-IoT est une technologie « Low Power Wide Area » (LPWA) et permet d'avoir une meilleure consommation de l'énergie des appareils utilisateurs. Une autonomie de batterie de plus de 10 ans peut être prise en charge pour un large éventail de cas d'utilisation [66]. Liu *et al.* [67] ont proposé un modèle de transport des données appelé « Data Forwarding Approach For Firerescue » (DFAFR). Avec DFAFR, chaque détenteur de données transmet les paquets de données conservés aux nœuds voisins indépendamment, et les proportions optimales de détenteurs de données sont maintenues de manière approximative.

f) **La simulation d'incendie**

Un des simulateurs de feu et de fumée les plus avancé est le simulateur *« Fire Dyanamic Simulation » (FDS)* développé par le laboratoire *« National Institute of Standards and Technology » (NIST)* [23]. Le modèle FDS est un modèle « Computational Fluid Dynamics » (CFD) centré sur l'écoulement du feu. Ce modèle est capable de résoudre numériquement une forme des équations de Navier-Stokes appropriée pour les écoulements à faible vitesse et thermiquement entraînés, tout en mettant l'accent sur le transport de fumée et de chaleur à partir des incendies [68]. Plusieurs auteurs [21]-[24], [34], [69]-[74] ont essayé d'exploiter cette technologie dans leur modèle.
Gamaleldin *et al.* [69] ont présenté un modèle permettant d'émuler la fumée lors d'un incendie. Ce modèle possède deux composants fondamentaux : un réseau de capteurs IoT et un simulateur FDS. Grâce aux capteurs IoT, les événements en temps réel tels que l'ouverture des portes et le bris des fenêtres peuvent être détectés. Ces informations sont collectées et fournies au simulateur FDS, afin de garantir une précision plus élevée de la simulation. La technique de l'apprentissage automatique « Long-Short Term Memory » (LSTM) est ensuite utilisée pour évaluer les valeurs de tout capteur dysfonctionnant temporairement. Un algorithme de routage à deux niveaux de l'algorithme A* est utilisé pour optimiser l'itinéraire de sortie des personnes évacuées dans le but de minimiser le temps d'exposition à la fumée. Mingbiao *et al.* [21] ont développé un modèle en utilisant les simulateurs PyroSim et Pathfinder. PyroSim est un logiciel de simulation basé sur le FDS. Grâce à cette simulation, il est possible de prédire la propagation du feu. Xinzheng *et al.* [22] ont proposé un

système de simulation d'incendies post-séisme. Le modèle proposé a été testé dans un un hôpital de 19 étages et a montré une prédiction réaliste de la propagation de l'incendie. Les collisions physiques entre occupants influencent les évacuations d'incendie en intérieur. Pour limiter cela, Xu *et al.* [23] ont proposé une méthode d'exercice virtuel pour les évacuations lors de l'incendie. Leur étude a montré que la simulation améliore les prises de décision lors de l'évacuation. Farid *et al.* [24] ont développé le modèle EvacuSafe. Ce modèle permet d'analyser et d'optimiser l'évacuation dans un bâtiment durant sa phase de construction. Pour ce faire, EvacuSafe est doté de divers scénario de feu pour détecter les failles dans le bâtiment.

2.3.1 Présentation de l'information

Une fois les données collectées et analysées, l'information utile peut être partagée de plusieurs manières. Dans cette section, nous développerons les cinq méthodes utilisées dans la littérature : la *vue 3D*, l'*application mobile*, l'*application Web*, la *réalité augmentée « Augmented Reality » (AR)* et la *réalité virtuelle « Virtual Reality » (VR)*.

a) **Vue 3D**

La *vue 3D* consiste en une vue numérique du bâtiment. La Figure 2.3 montre un exemple de vue 3D utilisée par Wang *et al.* [2]. La *vue 3D* permet un affichage simplifié des plans du bâtiment. Les données utiles (i.e., positionnent, chemin d'évacuation) sont ensuite rajoutées à la vue à l'aide d'icônes. Il est possible de se déplacer dans une *vue 3D* à l'aide d'une caméra virtuelle, ce qui permet une meilleure immersion.

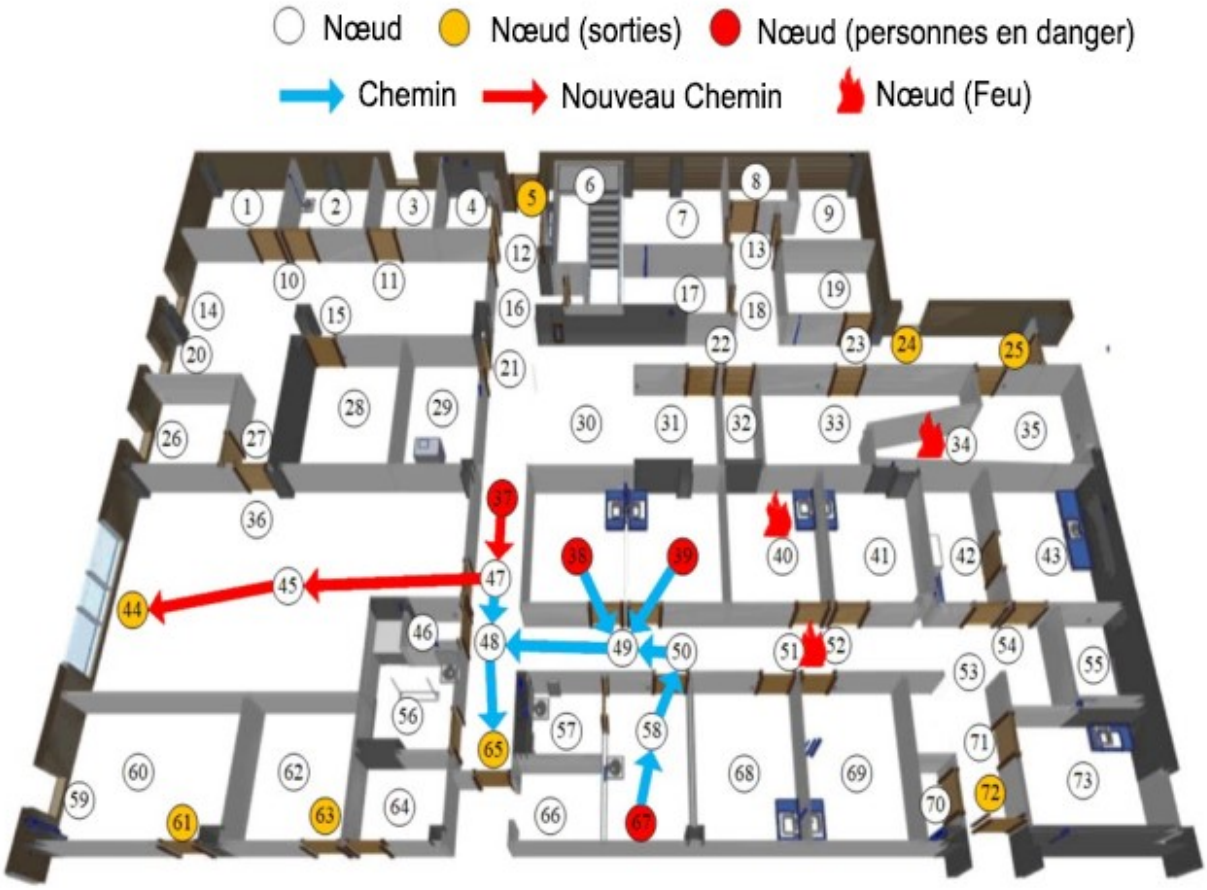

Figure 2.3 Vue 3D avec chemin d'évacuation et position du feu

b) Applications Mobile et Applications Web

Afin de présenter l'information, plusieurs auteurs [35], [75] ont choisi l'utilisation de l'*application mobile* ou de l'*application Web*. Lee *et al.* [75]. Les auteurs ont adopté la combinaison de l'*application mobile* et de l'*application Web* pour afficher le meilleur chemin d'évacuation à l'utilisateur. Les *applications mobiles* sont faites pour les personnes à l'intérieur du bâtiment et sont utilisées pour envoyer des notifications pour informer les occupants du feu qui se propage ou les chemins d'évacuation [76]. Certains auteurs [52], [75] combinent l'application mobile à la *réalité augmentée*, afin d'indiquer aux usages les meilleurs chemins purement évacués.

c) La réalité augmentée

La *réalité augmentée* (AR) consiste à la superposition de données 3D et de données réelles. Cette technique est utilisée par certains auteurs [52], [75] lors de la présentation des données. La Figure 2.4 montre un exemple de *réalité augmentée* mis en place par Jack *et al.* [52]. Dans cet exemple, les auteurs ont utilisé une flèche 3D pour indiquer le chemin d'évacuation aux occupants du bâtiment. Cependant, la flèche est visible en temps réel sur le flux vidéo du téléphone.

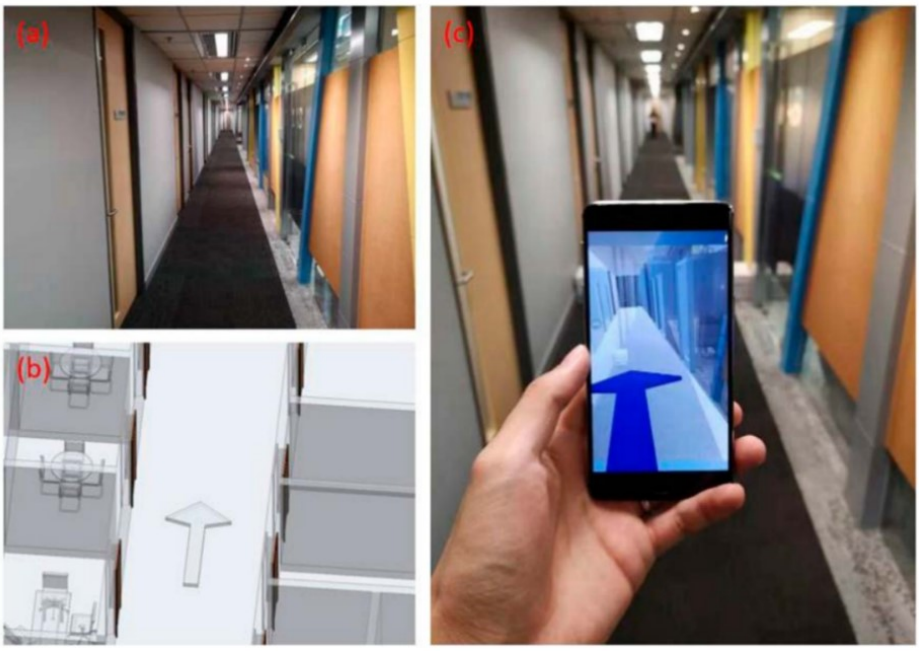

Figure 2.4 Exemple de réalité augmentée

d) La réalité virtuelle

La *réalité virtuelle* permet à une personne de s'immerger dans la vue 3D grâce à un casque de réalité virtuelle, comme le montre la Figure 2.5. Chen *et al.* [16] ont développé un système utilisant la *réalité virtuelle* et la *réalité augmentée*. Leurs travaux montrent que l'utilisation de VR et AR permet une meilleure immersion dans une situation de feu et que cette solution peut être utilisée pour des entraînements virtuels. Kanak *et al.* [49] ont proposé un environnement de réalité virtuelle qui s'appuie sur les données visuelles obtenues à partir du BIM et du GIS, ainsi que sur des données sensorielles.

Figure 2.5 Casque de réalité virtuelle - Samsung Gear VR SM-R320

2.4 Lacunes des méthodes existantes

Les modèles présents dans la littérature présentent de nombreuses limitations. Dans cette section, nous allons présenter les limitations portant sur la contrainte de temps et le calcul en temps réel, l'exploitation des données sonores, les conditions physiques des occupants et des personnes handicapées, ainsi que, l'adaptabilité et l'évolutivité les services limités.

2.4.1 Contrainte de temps et calcul en temps réel

Les modèles effectuant le calcul du meilleur chemin en temps réel [26], [60], [61] et les modèles réalisant la simulation FDS [21], [22], [69]-[71] ne respectent pas une contrainte de temps et ne sont donc pas des modèles en temps réel. Pour être en temps réel, il faudrait respecter deux contraintes [12]. La première est d'avoir un fonctionnement continu. En d'autres termes, il s'agit de collecter, traiter et afficher l'information de manière continue dans le temps. La deuxième est de respecter une contrainte de temps bien définie, généralement de l'ordre de microsecondes ou millisecondes. Les modèles en temps réel proposés, tels que ceux faisant du calcul du chemin en temps réel, respectent la première contrainte et non pas la seconde contrainte. Les modèles utilisant la simulation FDS ne garantissent pas non plus le fonctionnement en temps réel et cela pour deux raisons. La première est que la version actuelle du FDS ne permet pas la simulation du temps. Cependant, le NIST a lancé le projet « Advanced Fire Modeling » en 2018 [3] qui permettra la simulation du temps, mais ce projet est toujours en cours. La deuxième raison est que la simulation FDS consomme beaucoup de ressources informatiques et les modèles existants n'apportent aucune

solution à cette problématique. Les tableaux 2.5 et 2.6 montrent les résultats d'un benchmark sur la simulation FDS réalisée avec PyroSim [77]. La puissance de calcul nécessaire pour une simulation dépend du volume du bâtiment. Ce volume est mesuré en cellule (cell en Anglais). Un million de cellules vaut mille mètre cube, ou un bâtiment de dimension 20 m par 10 m par 5m. (10^6 cells = 20 m x 10 m x 5 m = 1000 m^3). Le Tableau 2.5 montre les machines utilisées pour le benchmark et le Tableau 2.6 présente les résultats moyens obtenus. Pour une simulation de 5 minutes d'un volume de 1 million de cellules ou (1000 m^3), un ordinateur aura besoin de 100 heures de calcul. Ces chiffres montrent que les modèles utilisant la simulation FDS ne peuvent pas être utilisées en temps réel.

Tableau 2.5 Ordinateurs Benchmark PyroSim (Simulation FDS)

Id	Processor	CPUs	RAM	OS
IntelD	Intel Pentium D @2.8 GHz	2	2 GB	XP Pro 32-bit
IntelC24	Intel Core2 Quad @2.4 GHz	4	8 GB	Vista Business 64-bit

Tableau 2.6 Performance Benchmark PyroSim (Simulation FDS)

Cells	Sim Time (min)	Run Tim (hrs)	Memory (GB)	Ouput (GB)
500 000	5	33	0.5	1.5
1 000 000	5	100	1	2.5
2 000 000	5	300	2	4.5

2.4.2 L'exploitation des données sonores

Aucun des modèles existants n'exploitent les données sonores. Ils utilisent des données BIM [8], [31], des données IoT (e.g., température, fumée, feu, etc.) [16]-[18] ou des données de caméra (i.e., vidéos et images) [3], [52]. Il est important de diversifier les sources d'information au niveau du bâtiment en feu, car les appareils utilisés peuvent être endommagé ou rendu inutilisable. Par exemple, les capteurs IoT peuvent fondre avec le feu, la fumée noire ou l'absence d'électricité peut rendre les caméras inutilisables. Les données sonores sont aussi importantes que les données visuelles et les autres données IoT, Elles peuvent être utilisées pour comprendre ce qui se passe dans les bâtiments en feu ainsi que pour localiser les personnes emprisonnées par le feu.

L’utilisation du son pour visualiser son environnement consiste au « Sound Event Detection » (SED) [78] et au « Acoustic Scene Classification » (ASC) [79]. Les données sonores sont aujourd'hui utilisées comme un système de sécurité dans les maisons. Ces systèmes permettent d'identifier des sons, tels que des bris de verre, des sonnettes de porte, des alarmes de détecteur de fumée, des alertes rouges, des cris humains, des pleurs de bébé, et d’autres.

2.4.3 Conditions physiques des occupants et personnes handicapées

La troisième limitation des modèles existants est que ces modèles ne prennent pas en compte la diversité physique des occupants du bâtiment. Ceci concerne plus les modèles proposant le calcul du chemin d'évacuation [2], [30], [57], [59]-[61]. Par exemple, pour un enfant, un adulte et une personne âgée, le meilleur chemin d'évacuation ne sera pas le même, due à leur capacité de déplacement. Les calculs utilisant les chemins les plus rapide tels que A* et Dijkstra [2], [57], [58] ne sont pas efficaces, puisqu’ils ne prennent pas en compte l'aspect psychologique des personnes dans le bâtiment. Pour pallier ces problèmes, des solutions ont été apportées [3], [27], [30], [59]. Cependant, aucune solution ne prend en compte la diversité des capacités physiques des occupants. De plus, les modèles existants ne prennent pas en charge les personnes handicapées ou de manière générale les personnes ayant des restrictions quelconque (i.e., mobilité limitée (chaise roulante), vue limitée ou malade couché au lit). La Figure 2.6 montre les problèmes de mobilité au Canada dans le recensement de 2012 [80] qui augmente avec l'âge. Le fait de ne pas considérer les aptitudes physiques des occupants du bâtiment en feu pose les problèmes suivants :

- une personne sourde n'entendra pas l'alarme incendiée sonnée et pourrait se rendre compte du feu en retard;
- une personne aveugle ne pourra pas suivre une carte sur son téléphone ou utiliser la réalité augmentée;
- une personne en chaise roulante ne pourra pas prendre les escaliers;
- une personne qui a une mobilité limitée ou une personne âgée se déplacera plus lentement qu'une personne en bonne santé. Elle est plus à risque de ne pas évacuer à temps, et on ne peut pas lui fournir les mêmes chemins que des individus bien portant.

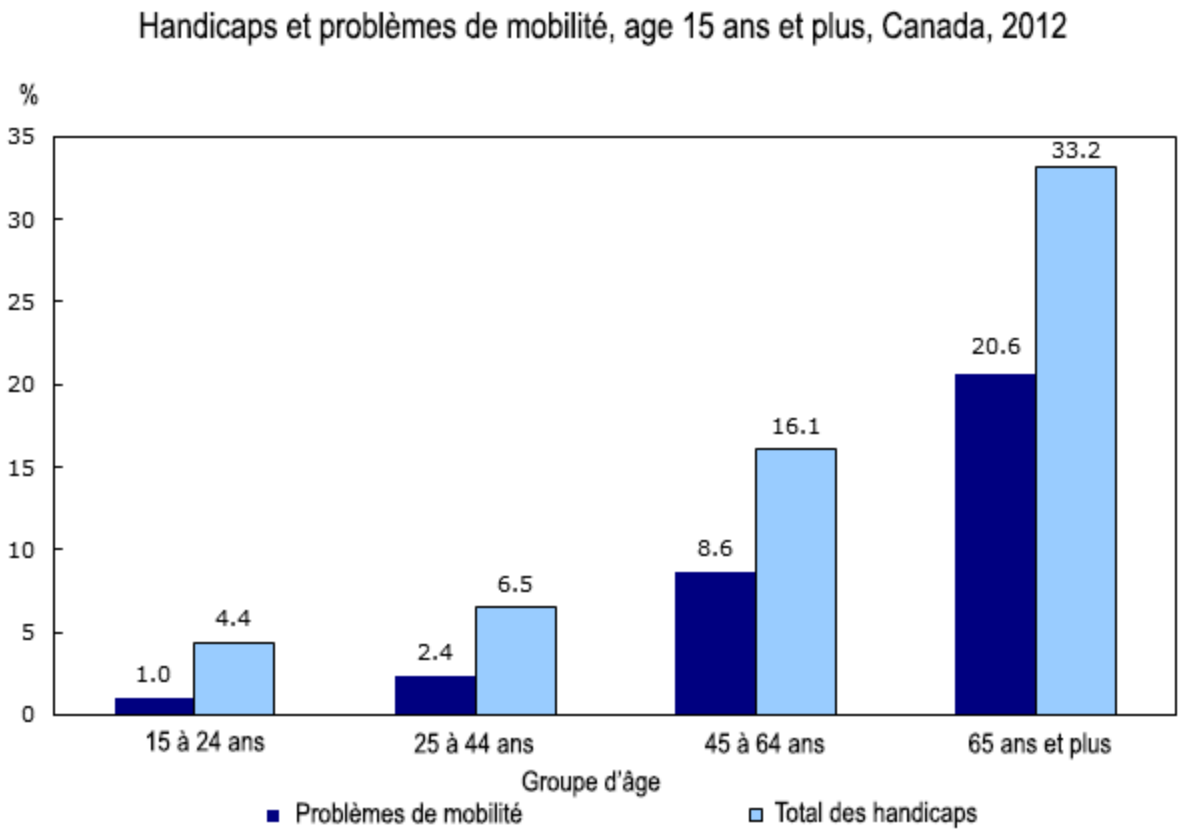

Figure 2.6 Problème de mobilité avec l'âge

2.4.4 Évolutivité, adaptabilité et services limités

De manière générale, les modèles existants proposent des services limités et sont peu évolutifs. Certains modèles consistent à calculer les meilleurs chemins d'évacuation, d'autres à contrôler la santé des sapeurs-pompiers, et d'autres à détecter l'origine des incendies, la position des occupants du bâtiment, etc. Tous ces services sont très utiles mais utilisés de manière séparée. Ce qui réduit leur efficacité. Les incendies sont une situation complexe, les modèles rigides de la littérature [2] [3], [48], [52], [51] limiteront la capacité d'action de sapeurs-pompiers. Un modèle plus flexible permettant aux sapeurs-pompiers d'utiliser les services dont ils ont besoin au moment voulu serait plus efficace. Un tel modèle n'aurait pas de services prédéfinis, mais plutôt une poule de services qui sont déployés au besoin. De nouveaux services peuvent être créés et s'incorporés aux existants pour avoir un système évolutif. De plus, les modèles existants n'offrent pas un système pour adapter leur puissance de calcul en fonction de la situation d'incendie. Un incendie peut arriver

dans n'importe quel bâtiment, en fonction de sa taille, sa complexité et le nombre de personnes à l'intérieur. Cependant, différentes puissances de calcul seront requises. Par exemple, un modèle qui fonctionne parfaitement pour un bâtiment à 3 étages ne le sera forcément pour un bâtiment à 10 étages. La Figure 2.7 montre un exemple de variation de l'occupation d'un bâtiment commercial au cours du temps [81]. Si un incendie est déclenché à 11 heures, il nécessitera plus de ressources pour l'évacuation qu'un incendie qui est déclenché à 17 heures.

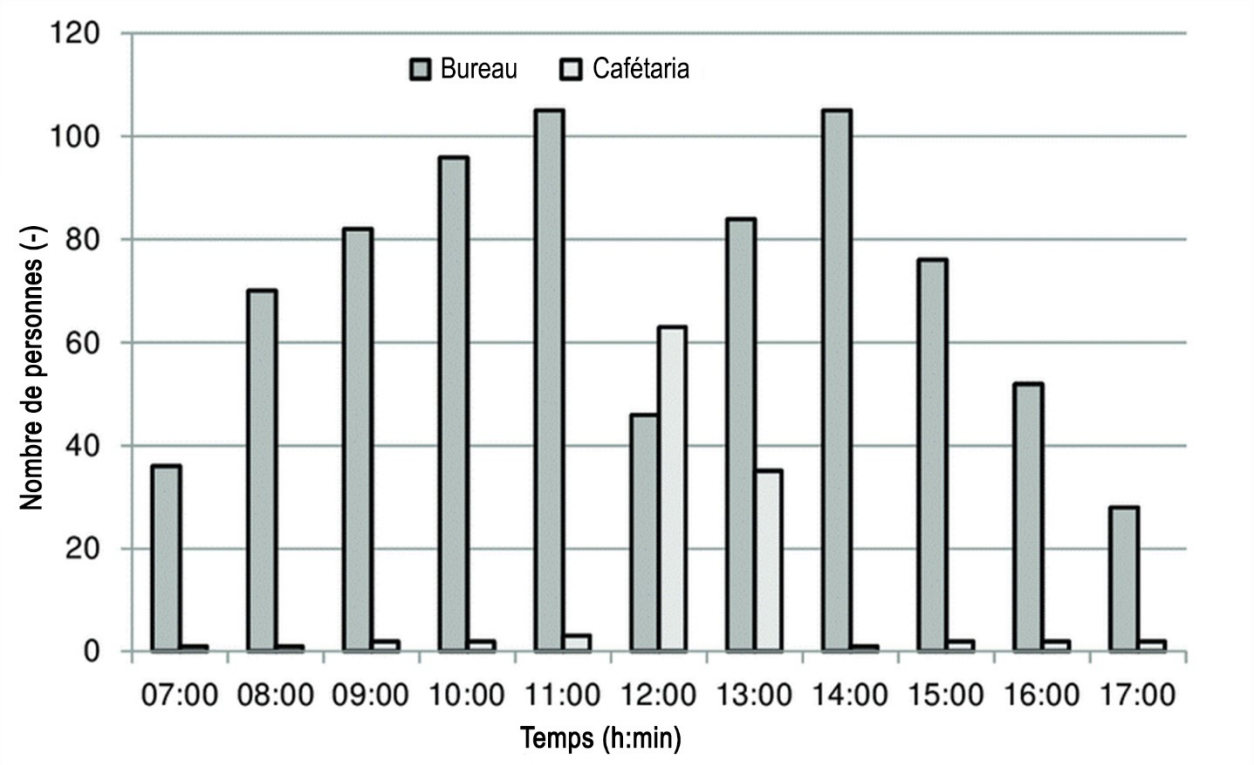

Figure 2.7 Occupation d'un bâtiment en fonction du temps

CHAPITRE 3 MODÈLE IOT DE GESTION D'INCENDIE EN TEMPS RÉEL

Dans ce chapitre, nous présenterons le nouveau modèle proposé pour la gestion des incendies en temps réel dans les villes intelligentes. Premièrement, nous élaborerons une vue globale de l'architecture du modèle proposé. Ensuite, nous détaillerons les quatre modules de l'architecture : le réseau de capteurs sans fil *« Fire Emergency Sensor Network » (FESNet)*, l'application distribuée, le serveur Web et les applications clients.

3.1 Architecture du modèle proposé

Dans cette section, nous présentons une vue globale de l'architecture du modèle proposé et les requis de ce modèle.

3.1.1 Vue globale de l'architecture

La Figure 3.1 montre l'architecture du modèle proposé pour la gestion d'incendie en temps réel. Cette nouvelle architecture est composée de quatre modules. Le premier module, en bleu, est fait du réseau *« Fire Emergency Sensor Network » (FESNet)*. Ce réseau assure la collection des données au niveau du bâtiment en feu ainsi que le contrôle à distance de certaines parties du bâtiment (i.e., portes, valves d'eau, etc.). Le second module, en jaune, est l'application distribuée qui assure le calcul de l'information en temps réel. Il est composé de trois sous-parties : une grappe « cluster » distribuée, trois bases de données et un pool de microservices. Le troisième module, en violet, est constitué d'un serveur Web qui joue le rôle de proxy entre la partie distribuée et les clients. Le quatrième module, en vert, est constitué des divers clients (e.g., application mobile, Web, etc.) pouvant se connecter au système distribué et afficher les résultats des divers calculs effectués. Ces différents modules échangent de l'information à travers le *réseau Internet haut débit* de la *ville intelligente*. Les microservcies sont en mesures de lire et d'écrire les données collectées ou calculées dans les bases de données. Les microservices de capture de données reçoivent les informations envoyées à partir de la passerelle du réseau FESNet. Le microservice de contrôle à distance envoie des données aux actionneurs situés au niveau du bâtiment en feu. Les appareils clients sont capables de se connecter à l'application distribuée par l'intermédiaire du serveur Web en utilisant le protocole *« Hypertext Transfer Protocol » (HTTP).*

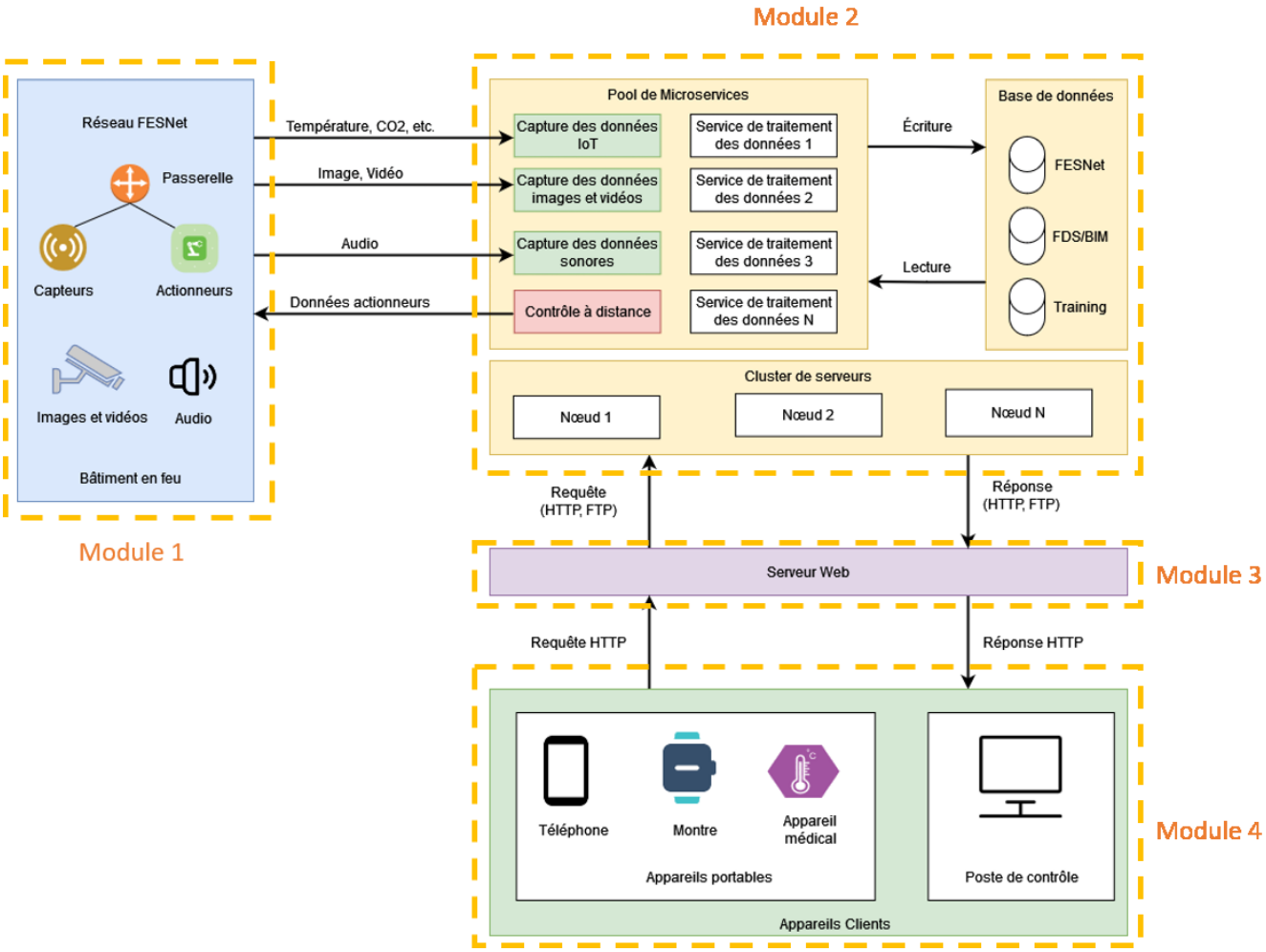

Figure 3.1 Architecture du modèle proposé

3.1.2 Requis du modèle

Afin de combler les limitations des modèles existants, notre modèle devra vérifier les cinq requis suivants : fonctionnement en temps réel, diversité dans la collection de l'information, diversité dans l'affichage de l'information, contrôle du bâtiment à distance, adaptabilité et évolutivité.

- **Fonctionnement en temps réel**

Le modèle proposé devra être capable de collecter les données, les traiter et les afficher en temps réel. Cela signifie que le modèle devra toujours respecter un délai de temps précis, peu importe la quantité de données à traiter.

- **Diversité dans la collecte de l'information**

Le modèle proposé ne devra imposer aucune limite sur le type de données pouvant être collectées. Ces données pourraient être de type : image, vidéo, son, données provenant des capteurs IoT (e.g., température, CO_2, humidité, etc.), données provenant des sources externes (i.e., base de données médicales) et d'autres. Il est important de diversifier la source de l'information, car certaines sources peuvent être rendues inutilisables durant les incendies. Par exemple, des capteurs peuvent fondre avec la chaleur, des images de caméra sont indisponibles à cause de la fumée, etc.

- **Diversité dans l'affichage de l'information**

Le modèle proposé ne devra imposer aucune limite sur les méthodes d'affichage de l'information. L'information devra pouvoir être distribuée sur une multitude de clients (e.g., application mobile, Web, etc.). Cela est important, car les personnes à l'intérieur du bâtiment n'auront pas tous accès aux mêmes clients. Par exemple, si l'on décidait d'utiliser uniquement une application mobile, cela exclurait les personnes handicapées qui ne peuvent pas utiliser un téléphone ou les personnes qui ne disposeraient pas de leur téléphone au moment de l'incendie.

- **Contrôle du bâtiment à distance**

Le modèle proposé devra pouvoir permettre le contrôle à distance du bâtiment, par exemple pour l'activation de porte ou de valve d'eau.

- **Adaptabilité et évolutivité**

Le modèle proposé devra pouvoir s'ajuster aux besoins de l'incendie, peu importe le bâtiment, sa structure et les personnes à l'intérieur du bâtiment.

3.2 Modules de l'architecture du modèle proposé

Dans cette section, nous présentons les quatre modules de l'architecture de notre modèle proposé, soit : le *réseau de capteurs sans fil « Fire Emergency Sensor Network » (FESNet)*, l'*application distribuée*, le *serveur Web* et les *applications clients*.

3.2.1 Module 1 - Réseau de capteurs sans fil FESNet

Le réseau *« Fire Emergency Sensor Network » (FESNet)* est un réseau de capteurs IoT dédié aux situations d'incendie. Dans cette section, nous ferons un rappel du réseau d'objets connectés, puis nous présenterons le réseau *FESNet*.

3.2.1.1 Réseau d'objets connectés

Le réseau d'objets connectés ou réseau IoT est constitué de trois couches, comme le montre la Figure 3.2 [82].

La **première couche** est composée de capteurs « sensors » sans fil et d'actionneurs « actuator » ou contrôleurs « controller ». Les capteurs sont chargés de collecter des données comme la température ou le taux de CO_2 dans l'air. Les actionneurs ou contrôleurs permettent de prendre des actions à distance. Par exemple, une valve d'eau connectée est un actionneur qui permettra de contrôler l'écoulement de l'eau à distance.

La **deuxième couche** est constituée de passerelles « gateways ». Les gateways sont responsables de collecter une multitude d'informations provenant de divers capteurs pour les transférer vers l'Internet ou le Cloud. En outre, lorsqu'une information est envoyée à un actionneur, la Gateway reçoit cette information et la transfère à l'actionneur correspondant.

La **troisième couche**, communément appelée Internet ou Cloud, est faite de diverses applications qui reçoivent les données collectées, afin de les analyser et en extraire de l'information utile.

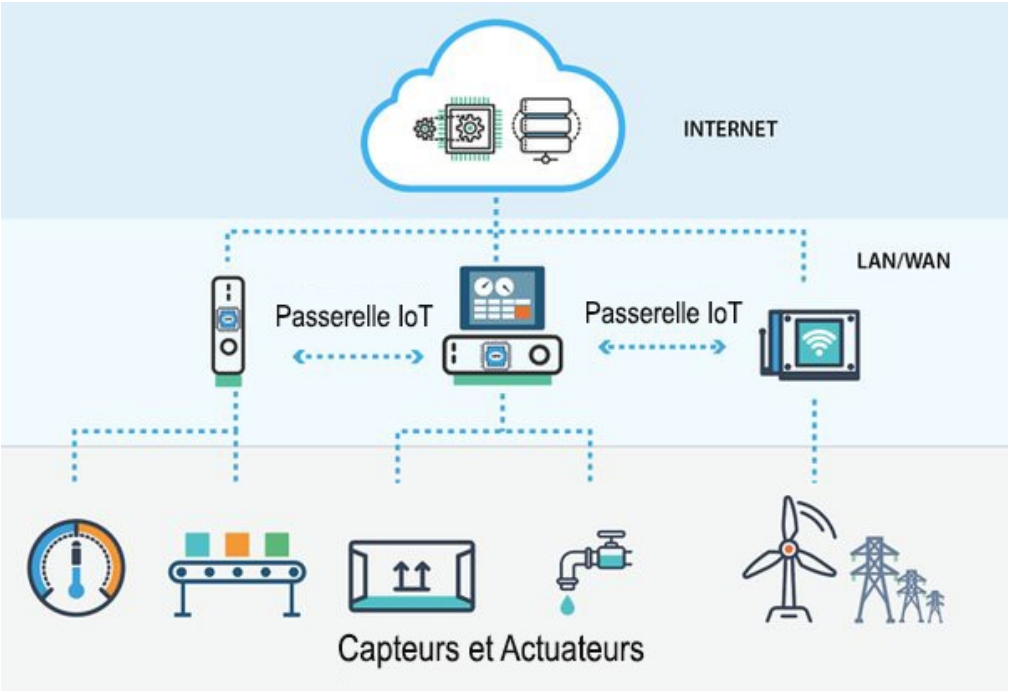

Figure 3.2 Réseau d'objets connectés

3.2.1.2 Réseau FESNet

Les réseaux de capteurs sans fil sont déjà utilisés pour la gestion de l'énergie dans les bâtiments dans un processus appelé *« Building Energy Management System » (BEMS)* [83]. Le BEMS utilise les capteurs IoT pour contrôler la ventilation, l'éclairage et la température des bâtiments. Le réseau *FESNet* est un réseau similaire, mais dédié aux situations d'incendie ou toute situation nécessitant l'intervention des sapeurs-pompiers. Le réseau *FESNet* est un réseau informatique de capteurs IoT chargé de capturer toutes les données utiles dans le bâtiment. En absence d'incendie, ce réseau permet la surveillance du bâtiment et la prédiction du potentiel danger de feu. En cas d'incendie, ce réseau collecte les données sur la situation de feu en temps réel (e.g., température, densité et du CO_2, etc.) et les envoie aux services de traitement et d'analyse des données. Le type de données collectées sont les données IoT, les données sonores et les données image et vidéo. Ce réseau est aussi composé d'actionneurs permettant ainsi le contrôle du bâtiment à distance (e.g., ouverture de valve, ouverture et fermeture de porte, etc.).

3.2.2 Module 2 - Application distribuée

Les données collectées au niveau du réseau *FESNet* en temps réel sont envoyées à l'application distribuée pour être analysées en temps réel. L'application distribuée est constituée de trois sous-parties : un système distribué, un pool de microservices et un ensemble de bases de données. Dans un premier temps, nous détaillerons le calcul en temps réel, les systèmes distribués, les microservcies et les bases de données en temps réel. Ensuite, nous présenterons la manière dont fonctionne l'application distribuée. Enfin, nous décrierons comment l'application garantit le calcul en temps et la modularité.

3.2.2.1 Calcul en temps réel

Le calcul en temps réel « Real-Time Computing » (RTC) consiste à collecter des données, les traiter et les afficher dans un temps très restreint. Ce dernier est de l'ordre de milliseconde (ms) ou de microseconde (µs) [84]. Nous pouvons citer quelques exemples de processus en temps réel :

- gestion du trafic des avions ou des voitures;
- voiture intelligent;
- surveillance médicale des battements du cœur;
- jeux vidéo en ligne;
- voix sur protocole Internet « Voice over Internet Protocol » (VoIP);
- vidéoconférence;
- opérations boursières.

Il n'existe pas de temps exact pur définir le calcul en temps réel. De manière générale, cela dépend du contexte et des capacités de perception de l'humain. Par exemple, en télécommunications, un délai bidirectionnel (i.e., round trip) inférieur à 300 ms pour la voie est considéré en temps réel [84]. Robert Miller [12] avait décrit trois ordres de grandeur différents de la réactivité des systèmes en temps réel :

1. le temps de réponse de 100 ms est perçu comme instantané;
2. les temps de réponse d'une seconde ou moins sont suffisamment rapides pour que les utilisateurs aient l'impression d'interagir librement avec l'information;

3. Les temps de réponse supérieurs à 10 secondes perdent complètement l'attention de l'utilisateur.

L'être humain a un temps de réaction moyen de 250 millisecondes [85]. Cette valeur peut donc être utilisée comme le maximum à ne pas dépasser. Cependant, les situations de feu sont très critiques, le feu et la fumée se propagent très rapidement, et les pompiers ont un temps très restreint pour agir. Ainsi, il faudrait considérer un délai plus court.

Dans notre modèle, le temps de calcul et d'affichage de l'information sera affecté par deux composantes :

- la vitesse du réseau (e.g., Internet, 5G, etc.) transportant l'information;
- la vitesse de traitement de l'information.

Dans notre modèle, nous supposons le contexte de la *ville intelligente* avec la présence des réseaux à haut débit tel que le réseau 5G. Nous admettons ainsi que notre réseau de données est capable de transporter l'information avec une vitesse appropriée. Le modèle proposé se concentrera sur le second point, c'est-à-dire la vitesse de traitement de l'information.

3.2.2.2 Systèmes distribués

Les systèmes distribués sont des systèmes où les calculs sont répartis sur plusieurs ordinateurs situés dans un réseau informatique. Ces ordinateurs, qui peuvent être ajoutés et retirés du système en tout temps, combinent leurs ressources *« Central Processing Unit » (CPU)*, *«Graphics Processing Unit » (GPU)* et mémoire *« Random Access Memory » (RAM)* pour former un système plus puissant [13]. La Figure 3.3 montre un exemple d'architecture distribuée [86].

Les systèmes distribués sont très utiles pour les systèmes en temps réel [13]. Par la suite, nous verrons leurs principaux avantages et inconvénients.

a) Avantages des systèmes distribués

Évolutivité : Le système distribué peut accroître ses ressources de calcul en fonction de la charge de travail. Ceci est fait en ajoutant des nœuds de calcul ou des nœuds supplémentaires au réseau selon les besoins.

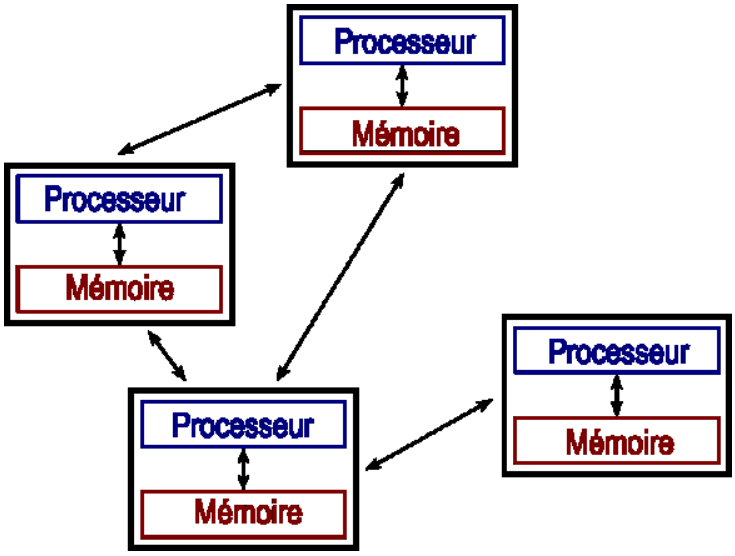

Figure 3.3 Exemple de système distribué avec 4 nœuds

Concurrence : Les composants des systèmes distribués fonctionnent simultanément en parallèle. Ils se caractérisent également par l'absence d'une horloge globale, ainsi que les tâches se produisent hors séquence et à des rythmes différents.

Disponibilité et tolérance aux pannes : Si un nœud tombe en panne, les autres nœuds peuvent continuer à fonctionner sans perturber le calcul global.

Transparence : Un programmeur externe ou un utilisateur final voit un système distribué comme une seule unité de calcul plutôt que comme ses parties sous-jacentes. Ceci permet aux utilisateurs d'interagir avec un seul dispositif logique plutôt que de s'intéresser à l'architecture du système.

Hétérogénéité : Dans la plupart des systèmes distribués, les nœuds et les composants sont souvent asynchrones, avec des matériels, des intergiciels « middleware », des logiciels et des systèmes d'exploitation différents. Ceci permet aux systèmes distribués d'être étendus par l'ajout de nouveaux composants.

Réplication : Les systèmes distribués permettent le partage d'informations et de messages, tout en assurant la cohérence entre des ressources redondantes (i.e., composants logiciels ou matériels), ainsi qu'en améliorant la tolérance aux pannes, la fiabilité et l'accessibilité.

b) Inconvénients des systèmes distribués

Sécurité : Les systèmes distribués sont aussi vulnérables aux attaques que tout autre système. Leur nature distribuée crée une grande surface d'attaque qui expose les organisations aux menaces.

Risque de défaillance du réseau : Les systèmes distribués sont tributaires des réseaux publics pour transmettre et recevoir des données. Si un segment de l'Internet devient indisponible ou surchargé, les performances du système distribué peuvent diminuer.

Contrôle des coûts : Contrairement aux systèmes centralisés, l'évolutivité des systèmes distribués permet aux administrateurs d'ajouter facilement des capacités supplémentaires selon les besoins, ce qui peut également augmenter les coûts. La tarification des systèmes informatiques distribués basés sur le cloud est basée sur l'utilisation, par exemple, le nombre de ressources mémoire RAM et la puissance CPU consommées au fil du temps.

3.2.2.3 Les microservices

Les microservices sont une approche architecturale dans laquelle une application unique est composée de nombreux petits composants, ou services, faiblement couplés et pouvant être déployés indépendamment [87]. Les microservices sont définis comme *Cloud Native*, c'est-à-dire qu'ils sont optimisés pour le *Cloud computing* (i.e., les systèmes de calcul distribués) [87]. Avec les microservices, il n'est pas nécessaire de redéployer l'ensemble de l'application, mais uniquement les services dont on a besoin. Dans le cas d'ajout ou de mise à fin d'un service, il est facile de le retirer, de l'ajouter, de le mettre à l'échelle et de le modifier pour l'adapter à l'évolution des besoins, rapidement et efficacement, sans avoir besoin de fermer l'ensemble de l'application. La Figure 3.4 montre une comparaison entre l'architecture traditionnelle (i.e., monolithique) et l'architecture microservice. Comme on peut le voir dans cette figure, le traitement de l'information se fait par un seul composant dans l'architecture monolithique, tandis que dans l'architecture microservice le traitement de l'information est réparti sur plusieurs microservcies.

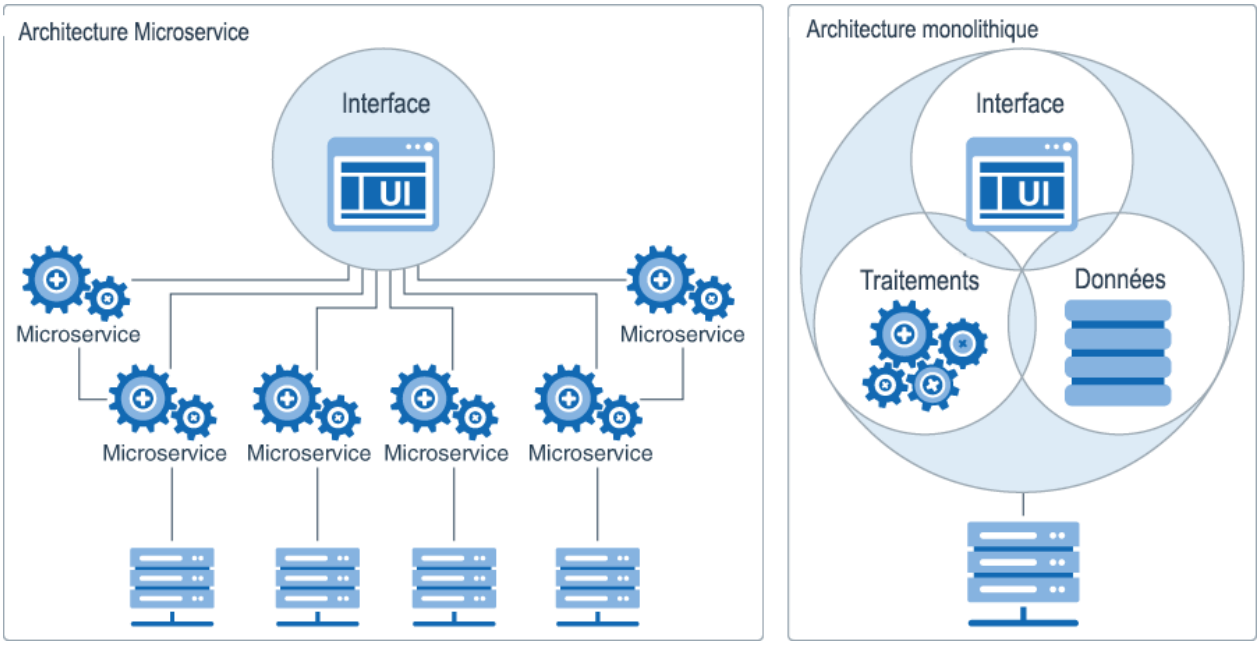

Figure 3.4 Architecture microservice et monolithique

Les microservices ont des avantages, mais aussi des inconvénients par rapport à l'architecture traditionnelle [88]. Les principaux avantages et inconvénients de l'architecture microservice sont présentés par la suite.

a) Avantages des microservices

- les microservices offrent des solutions polyvalentes, c'est-à-dire que plusieurs langages de programmation et technologies peuvent être utilisés;
- l'intégration et la mise à l'échelle sont faciles grâce à des applications tierces;
- les microservices se déploient facilement grâce à leur compatibilité avec les méthodologies flexibles;
- les solutions basées sur les microservices offrent des améliorations rapides et constantes liées à chaque fonction;
- il est devenu facile pour les développeurs de tirer parti de l'avantage des tiers et il n'est pas nécessaire de partir de zéro;

- les projets modulaires évoluent de manière plus organique, ce qui rend plus facile d'organiser des développements en parallèle, en utilisant des ressources multiples dans une période déterminée.

b) Inconvénients des microservices

Le principal inconvénient des microservices est leur complexité. Cependant, il est plus difficile de développer une architecture microservice qu'une architecture traditionnelle. Ceci signifie un temps de développement plus long et un coût financier plus élevé.

3.2.2.4 Les bases de données en temps réel

Une base de données en temps réel est un système de base de données capable de gérer des charges de travail de données en constante évolution et extrêmement sensible au facteur temps, contrairement aux bases de données traditionnelles, qui contiennent des données persistantes et qui changent beaucoup moins fréquemment [89]. Le traitement en temps réel signifie que les transactions sont traitées très rapidement, ce qui permet à l'organisation ou à d'autres systèmes intégrés d'agir immédiatement sur les données. Les bases de données en temps réel sont généralement utilisées pour la comptabilité, la banque, le droit, les dossiers médicaux, le multimédia, le contrôle des processus, les systèmes de réservation et l'analyse des données scientifiques [89]. Les bases de données relationnelles comme MySQL stockent leurs données dans des tables, en utilisant des lignes et des colonnes [89]. Les bases de données en temps réel (i.e., Firebase, RethinkDB, Backendless, MongoDB, Cassandra, OrientDB, Parse, Meteor) utilisent différentes architectures de base de données comme JavaScript « Object Notation » (JSON), ou clé-valeur [89].

Contrairement au modèle de base de données traditionnel, les bases de données en temps réel font du streaming de données en temps réel. Le streaming de données en temps réel utilise des requêtes multiples qui fonctionnent sur des fenêtres de temps et de tampon, en utilisant les données pendant qu'elles sont traitées dans le serveur [90].

3.2.2.5 Fonctionnement de l'application distribuée

L'application distribuée est composée d'un cluster distribué d'ordinateurs sur lequel sont installés les bases données et les différents services. Ce cluster fournit les ressources CPU, GPU et mémoire RAM nécessaire au traitement de l'information. Il est chargé d'allouer les ressources requises pour

chaque service, afin de maintenir un calcul en temps réel. Ce cluster peut être étendu ou réduit à tout moment en changeant le nombre d'ordinateurs connectés. La Figure 3.5 montre l'architecture de l'application distribuée.

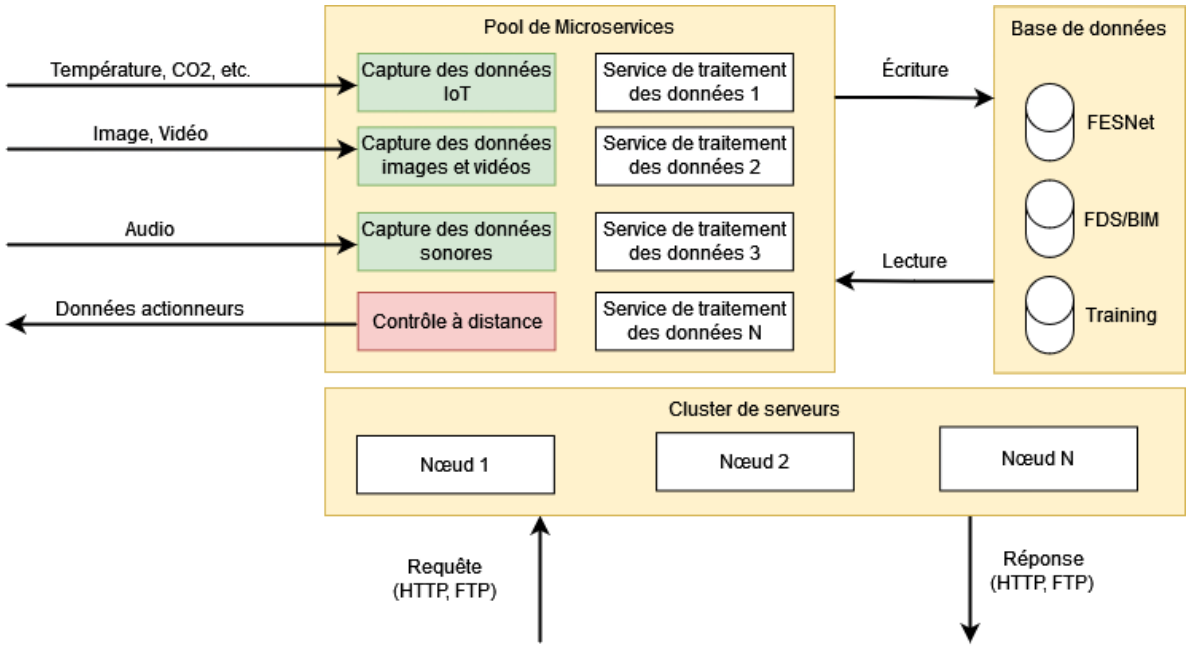

Figure 3.5 Architecture de l'application distribuée

Les microservices sont le cœur du traitement de l'information. Chaque microservice est chargé d'une tâche spécifique qu'il se doit de réaliser dans un délai précis. Pour ce faire, le microservice peut demander au cluster distribué de lui allouer les ressources nécessaires à son calcul. Il existe trois types de microservices :

- ceux chargés de collecter les données venant de réseau *FESNet*;
- ceux chargés du contrôle à distance du bâtiment;
- ceux chargés du traitement de données (i.e., calcul de chemin d'évacuation, détection des dangers, etc.).

Enfin, les bases de données permettent de stocker et partager l'information entre les différents services.

Le modèle proposé utilise trois bases de données :

- la base de données *BIM-FDS* qui contient les données 3D du bâtiment. Ces données sont sous forme de fichiers aux formats *«* Industry Foundation Classes *»* (IFC) et *«* Fire Dynamics Simulator *»* (FDS). Les microservices sont en mesure de lire et d'écrire les fichiers BIM et FDS à l'aide du protocole *«* File Transfer Protocol *»* (FTP);
- la base de données *Training* qui contient les résultats de différents scénarios de simulation. En faisant une analyse des scenarios de feu passés et des simulations de potentiels scenarios de feu, des données importantes sur la prédiction des futurs incendies peuvent être récoltées. Ces données sont stockées dans la base de données *Training*. Une fois qu'un feu se déclenche, il est possible de retrouver les scénarios de feu les plus similaires dans la base de données. Ceci aidera à accélérer les calculs. Les données de *Training* peuvent être stockées à l'aide de données *« Structured Query Language » (SQL)* ou NoSQL;
- la base de données *FESNet* qui stocke les données collectées en temps réel provenant du bâtiment et les données résultant du traitement de l'information par les différents services. La base de données *FESNet* est une base de données en temps réel. Elle est continuellement mise à jour et permet d'avoir un contrôle en temps réel sur le bâtiment.

3.2.2.6 Traitement de l'information en temps réel et adaptabilité du modèle

Afin de garantir que notre modèle pourra fonctionner en temps réel et respecter les délais de temps imposés, il nous faut réduire le délai de traitement de l'information. Ceci peut se faire de deux manières :

- optimiser nos algorithmes pour qu'ils prennent moins de temps;
- fournir les ressources nécessaires (i.e., CPU, GPU, mémoire RAM) aux algorithmes.

La première méthode pour optimiser les algorithmes n'est pas possible, puisque le temps de calcul d'un algorithme dépend de la quantité de données à traiter. En fonction des incendies (i.e., taille du bâtiment, quantité de personnes concernées), la quantité de données à traiter varie. Le temps que prend un algorithme est mesuré en utilisant la complexité temporelle. La complexité temporelle est généralement exprimée à l'aide de la notation grand O (i.e., O(n), O(nlogn), etc.) [91]. La Figure 3.6 montre la variation du nombre de calculs nécessaire (N) en fonction de la quantité des données à traiter (n). Pour un algorithme de complexité O(n), le temps de calcul augment de manière linéaire avec un coefficient de 1. Ceci signifie que pour deux fois plus de données il faut deux fois plus de

temps. Cependant, si l'on double les ressources CPU, le temps de calcul restera le même. Le modèle proposé permet d'allouer de manière précise les ressources nécessaires à chaque service pour maintenir la contrainte de temps, tout en garantissant ainsi un calcul en temps réel.

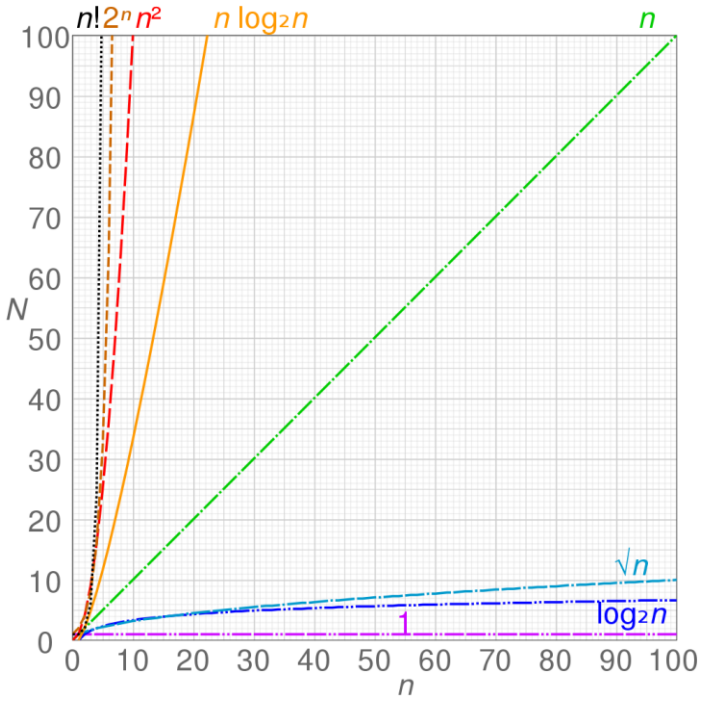

Figure 3.6 Complexité en temps des algorithmes

3.2.3 Module 3 - Serveur Web

Le serveur Web sert de proxy entre le système distribué et les clients. Cette configuration permet de simplifier le partage de l'information avec une diversité de clients en simultané. Du côté client, le système distribué n'est qu'un simple serveur Web et tout appareil connecté peut y soumettre de requête. La Figure 3.7 montre l'architecture d'un serveur web dynamique [92]. Comme on le voit sur la figure, un serveur Web utilise le protocole « Hypertext Transfer Protocol » (HTTP) pour communiquer avec les clients.

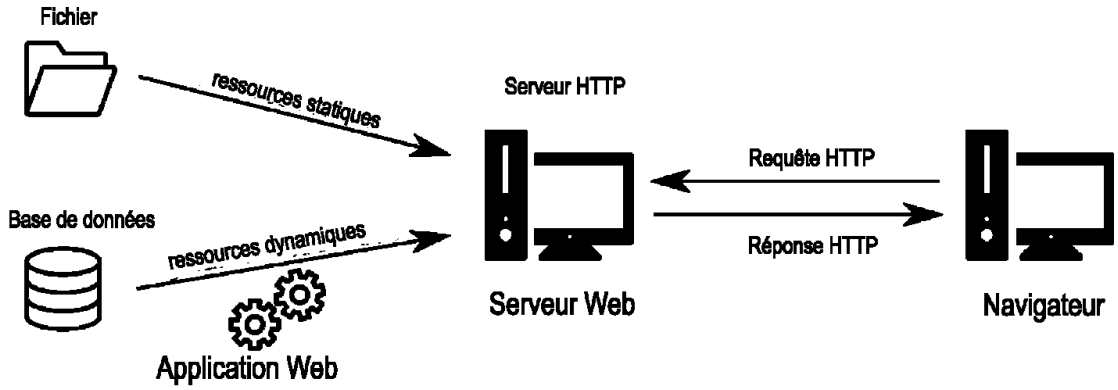

Figure 3.7 Architecture server Web

3.2.4 Module 4 - Applications clients

Le dernier composant de l'architecture est les applications clientes qui peuvent être fonctionnées sur un poste fixe ou mobile. Tout poste client peut se connecter à un serveur, comme le montre la Figure 3.8. Les clients peuvent être divisés en deux groupes :

- **les postes de contrôle dédiés aux pompiers** : ces types de client ont un accès total sur le système distribué. Ils peuvent déployer de nouveaux services. Ils ont un accès sur la vue 3D du bâtiment et peuvent contrôler le bâtiment à distance;
- **les périphériques dédiés aux occupants du bâtiment** : il s'agit des téléphones intelligents, des montres intelligentes et d'autres. Ces clients ne servent qu'à recevoir de l'information comme des notifications sur les dangers ou les directives à suivre pour évacuer.

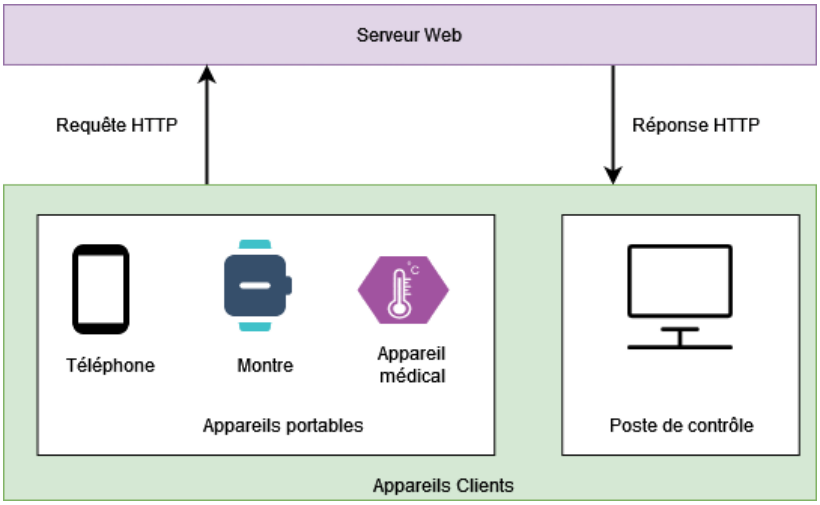

Figure 3.8 Serveur Web et clients

CHAPITRE 4 IMPLÉMENTATION ET ANALYSE DES RÉSULTATS

Dans le Chapitre 3, nous avons établi les quatre modules de l'architecture de notre modèle de gestion des incendies en temps réel. Afin de pouvoir évaluer la performance de cette architecture, il nous faut implémenter chacun de ces modules. Dans ce chapitre, nous présenterons tout d'abord une vue globale du prototypage du modèle proposé. Ensuite, nous détaillerons l'implémentation du deuxième module *Application distribuée*, du troisième module *Serveur Web* et du quatrième module *Applications clients*. Enfin, nous terminerons par une évaluation du modèle proposé et une discussion des résultats.

4.1 Prototypage du modèle proposé

Le prototype est constitué de deux applications principales qui fonctionnent de manière indépendante, mais qui collaborent pour former un système complet. La première application est le système distribué, il s'agit du cœur du modèle où le traitement des données se passe. La deuxième application est le serveur WWeb qui permet l'affichage de l'information via le navigateur Web ou via une application mobile. Afin de simplifier et accélérer le développement du prototype, nous avons choisi d'utiliser le langage de programmation Python. Pour une solution réelle, les langages C++ et « *Real-Time Specification for Java* » (RTSJ) offriront de meilleures performances. L'architecture du modèle proposé est constituée de quatre modules :

- module 1 : Réseau de capteurs sans fil « *Fire Emergency Sensor Network* » (FESNet);
- module 2 : *Application distribuée* dédiée au calcul en temps réel;
- module 3 : *Serveur Web* qui sert de liaison entre l'*application distribuée* et les clients;
- module 4 : *Applications clients* qui peuvent être des postes d'ordinateurs ou des téléphones intelligents.

L'implémentation du réseau *FESNet* nécessite l'utilisation du matériel « *Internet of Things* » (IoT). En d'autres termes, il s'agit d'une passerelle IoT « gateway », des nœuds IoT (i.e., carte arduino), des capteurs « sensor » et des actionneurs « actuator ou controller ». Le principal rôle du réseau *FESNet* est de collecter les données tels que la température et le taux de CO_2 au moment de l'incendie. Comme nous ne disposons pas du matériel IoT nécessaire, nous utiliserons des données générées aléatoirement pour simuler le réseau *FESNet*.

L'implémentation de l'*application distribuée* nécessite l'implémentation d'un cluster de serveurs, de microservices et de la base de données en temps réel. La base de données BIM-FDS et la base de données « Training » ne seront pas implémentées car nous ne disposons pas des données nécessaires.

L'implémentation du *serveur Web* permettra aux sapeurs-pompiers et aux occupants du bâtiment en feu d'avoir un accès en temps réel aux informations utiles calculées au niveau de l'application distribuée.

Enfin, nous utilisons deux types de clients. Le premier est le poste de contrôle dédié aux sapeurs-pompiers et qui est constitué d'un *navigateur Web* (e.g., Firefox, Chrome, etc.). Le second client est l'*application mobile Android FireAlert*.

4.1.1 Implémentation du module 2 – application distribuée

L'*application distribuée* (ou système distribué) est constituée d'un cluster distribué sur lequel tourne une multitude de microservices et leur base de données. Dans cette section, nous présenterons l'implémentation du cluster distribué, de la base de données en temps réel FESNet et de deux microservices. La Figure 4.1 montre une vue globale du prototype de notre système distribué. Le code source du système distribué est présenté à l'Annexe C.

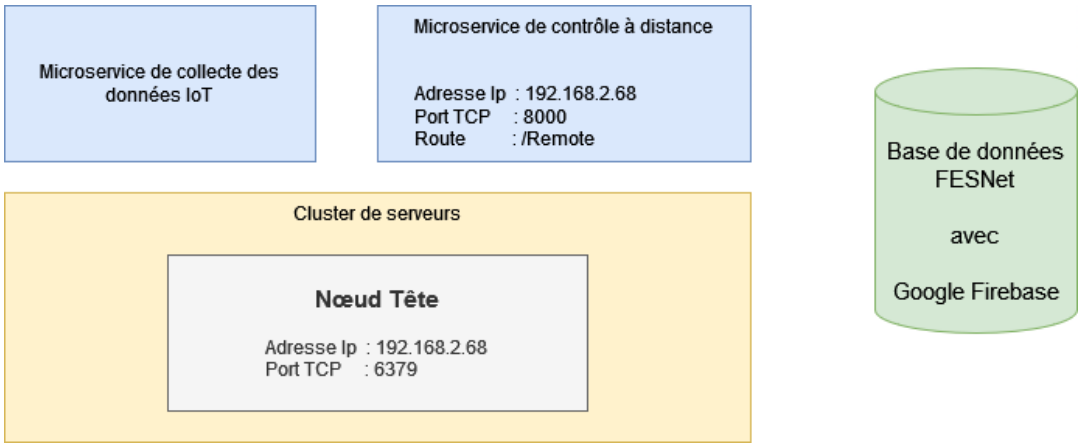

Figure 4.1 Vue générale – prototype du système distribué

4.1.1.1 Implémentation et déploiement du cluster distribué

Dans cette section, nous présenterons l'architecture de la librairie Ray [93]. Puis, nous détaillerons les étapes d'installation de la librairie Ray. Ensuite, nous décrierons le déploiement de l'*application*

distribuée concernant le cluster avec la librairie Ray. Enfin, nous présenterons le processus d'ajout ou de suppression de nœuds au cluster.

a) Architecture de la librairie Ray

L'architecture de la librairie Ray est présentée à la Figure 4.2. Afin de mettre en place notre cluster distribué, nous utilisons la librairie PPython Ray. Avec Ray, le cluster est constitué de plusieurs ordinateurs appelés nœuds « Node ». Le cluster est construit autour du nœud principal appelé « Head Node ». Une fois le « Head Node » déployé, il est possible d'agrandir ou de réduire le cluster à tout moment en rajoutant ou en retirant des ordinateurs appelés « Worker Node ». Les « Worker Node » se connectent au « Head Node » via le réseau Internet et l'adresse « Internet Protocol » (IP) du « Head Node ». Tous les ordonnateurs dans le cluster forment une seule entité en combinant le total des ressources *« Central Processing Unit » (CPU)*, *« Graphics Processing Unit » (GPU)* et mémoire *« Random Access Memory » (RAM)*. Lorsqu'un mircroservice est déployé sur le cluster, l'« Application Programming Interface » (API) de Ray choisit automatiquement le nœud « Node » le plus approprié pour le déploiement. Ce dernier reste totalement transparent envers l'utilisateur.

a) Installation de la librairie Ray

Afin d'installer la librairie Ray, nous devons d'abord installer Python et la librairie Python Pip [94]. Pour notre prototype, nous installons la version 3.9.6 de Python en suivant les directives sur le site officiel [95]. Une fois Python est installé, nous pouvons vérifier sa version avec la commande suivante :

```
py --version
```

Ensuite, nous installons ou mettons à jour Python Pip avec la commande suivante :

```
py -m ensurepip --upgrade
```

À ce stade, nous disposons de Python et Pip sur notre machine de test, alors nous pouvons installer Ray avec la commande suivante :

```
pip install ray
```

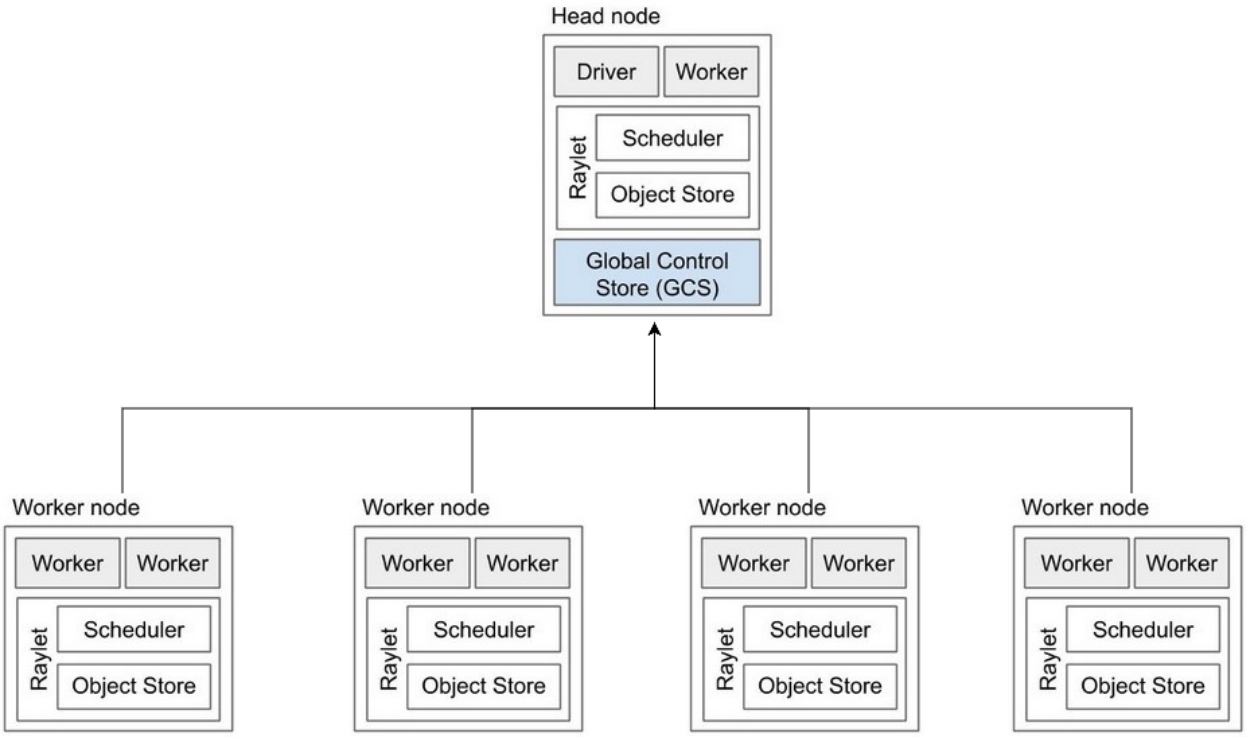

Figure 4.2 Cluster distribué avec Ray

La librairie Ray dispose de trois sous-librairies qui doivent être installées de manière indépendante. Ces trois sous-librairies sont les suivantes :

- *Serve* « Scalable and Programmable Serving »;
- *Tune* « Scalable Hyperparameter Tuning »;
- *RLLib* « Reinforcement Learning ».

Nous aurons besoin de la sous-librairie *Ray Serve* pour l'implémentation du microservice de contrôle à distance. Nous allons donc installer *Ray Serve* avec la commande suivante :

```
pip install "ray[serve]"
```

b) Déploiement de l'application distribuée

Après l'installation de la librairie Ray sur notre machine, nous pourrons déployer notre machine comme nœud principal « Head Node » de notre cluster. Ceci se fait avec la commande « ray start » suivie de quelques paramètres tel que le port « *Transmission Control Protocol* » (TCP) à utiliser. L'adresse IP du nœud principal sera celui de notre machine, soit 127.0. 0.1 pour le « localhost » ou

192.168.2.68 pour l'adresse WIFI. La Figure 4.3 montre le démarrage du cluster sur une console Windows.

Le déploiement se fait avec la commande suivante :

```
ray start --logging-level=debug --head --port=6379 --dashboard-port=8265 -v
```

Les paramètres utilisés sont les suivants :

- *logging-level* : active l'affichage de toute information de débogage;
- *head* : indique que le présent ordinateur sera le Head Node;
- *port* : indique le port TCP utilisé;
- *dashboard-port* : indique le port TCP du *Dashboard*. Le *Dashboard* est un serveur Web accessible à l'adresse « localhost:dashport-port »), dont ce dernier n'est disponible que sur les machines Linux;
- *v* : pour verbose, active l'affichage de plus d'information.

Après le démarrage du cluster, nous démarrons aussi le service *Ray Serve*, qui sera utilisé pour le contrôle à distance du bâtiment. La commande suivante démarre le service Serve :

```
serve start
```

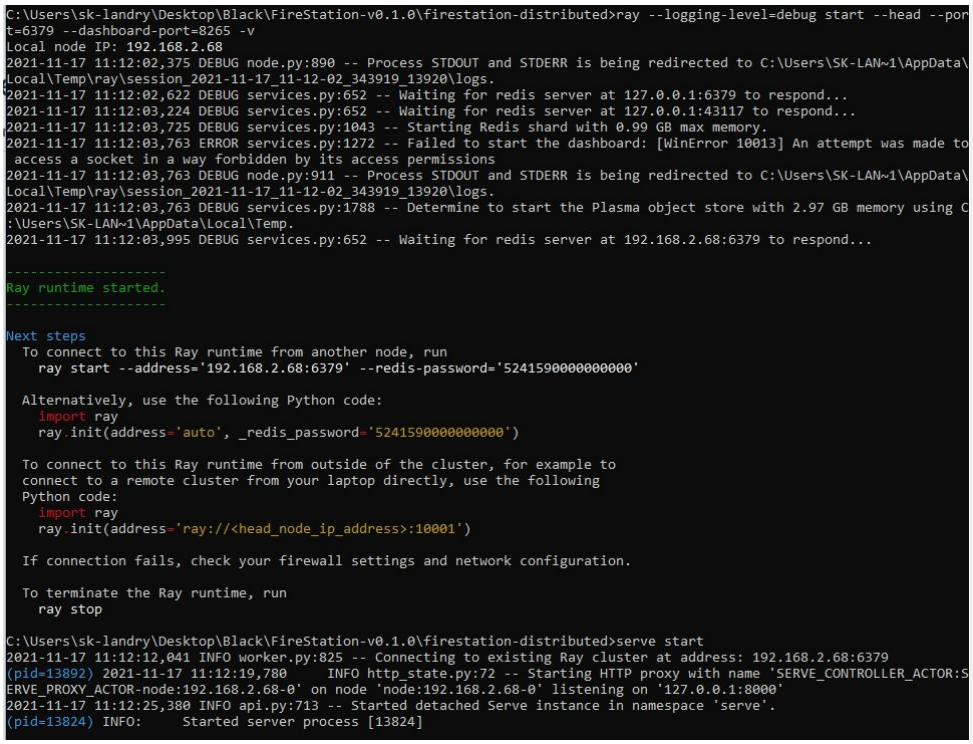

Figure 4.3 Démarrage du cluster distribué

c) Ajout et suppression de nœuds au cluster

Pour notre prototype, nous utilisons un seul ordinateur qui sera configuré comme le nœud principal « Head Node ». Cependant, il est possible de rajouter des nœuds supplémentaires « Worker Node ». De ce fait, il suffit d'installer la librairie Ray sur les nouvelles machines, puis la déployer avec la commande « ray start » suivante en indiquant l'adresse IP du nœud principal :

```
ray start --address=192.168.2.68
```

Afin de déconnecter les nœuds supplémentaires, nous utilisons la commande suivante :

```
ray stop
```

4.1.1.2 **Installation et déploiement de la base de données en temps réel FESNet**

Une fois le cluster déployé, nous aurons besoin de mettre en place la base de données en temps réel. Pour notre prototype, nous utiliserons la base de données Firestore de Google Firebase [96] car elle est facile à installer et à utiliser. La base de données Firestore est une base de données NoSQL en temps réel. Elle enregistre les données sous forme de Collections et de Documents.

Afin de créer notre base de données, nous aurons besoin d'un compte Google. Une fois le compte Google créé, nous suivrons la documentation officielle sur le site Google Firebase pour créer notre base de données. Après la création de la base de données, nous obtiendrons un ensemble de données d'authentification. Ces données d'authentification nous permettent de nous connecter à la base de données en temps réel à partir de notre application Web et notre application mobile, comme le montre la Figure 4.4.

La Figure 4.5 montre la structure de notre base de données. Nous avons un document appelé « IoT Data ». Ce document contient une Collection appelé « Simple Data ». Dans cette Collection, nous avons trois champs de données qui indiquent des valeurs mises à jour en temps réel. Ces trois champs de données sont les suivants :

- *firefighters-inside* : le nombre de sapeurs-pompiers à l'intérieur du bâtiment en feu;
- *smoke-density* : le taux de CO_2 dans le bâtiment en feu;
- *temperature* : la température dans le bâtiment en feu.

```
{
  "type": "service_account",
  "project_id": "firestation-9d01e",
  "private_key_id": "d51c70de9eb17474afc0e68779a3936321ee42d8",
  "private_key": "-----BEGIN PRIVATE KEY -----END PRIVATE KEY-----\n",
  "client_email": "firebase-adminsdk-u2u21@firestation-9d01e.iam.gserviceaccount.com",
  "client_id": "114945489086992165715",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-u2u21%40firestation-9d01e.iam.gserviceaccount.com"
}
```


Figure 4.4 Données d'authentification de Google Firebase

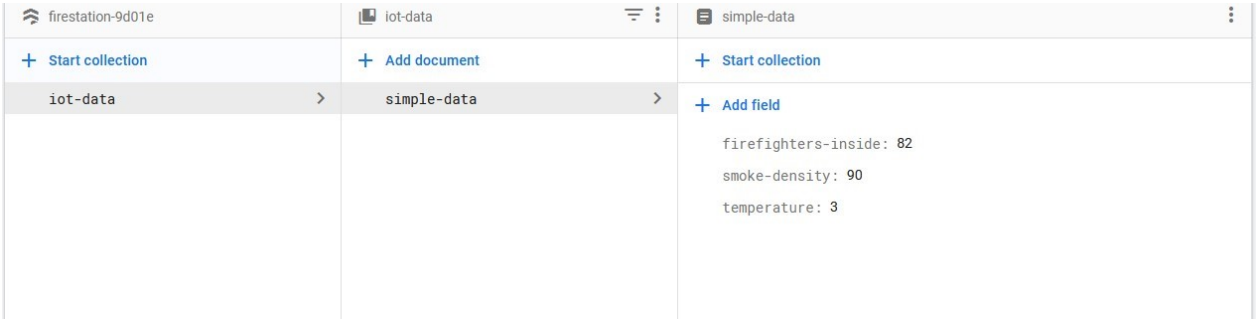

Figure 4.5 Base de données Firebase

4.1.1.3 **Implémentation et déploiement des microservices**

Dans cette section, nous allons implémenter le microservice de collecte des données IoT et le microservice de contrôle du bâtiment à distance.

a) Microservice de collecte des données IoT

Notre premier microservice est le service « IoTService ». Ce service est chargé de collecter les données IoT en temps réel et les ajouter à notre base de données FESNet. Il est mis en place grâce à l'API « Ray Remote » et à la classe « IoTService », comme le montre la Figure 4.6. Le corps du service se trouve à l'intérieur de la méthode « def run(self) ». Comme nous ne disposons pas des capteurs IoT, nous générons des données aléatoires grâce à la méthode « randrange (100) ». Les données générées sont ensuite stockées en temps réel dans la base de données FESNet en utilisant L'API Python de Google Firebase [97].

```
# Microservcie de collecte des données IoT
@ray.remote
class IoTService(object):
    def __init__(self):
        pass

    def run(self):
        # Connection à la base de données en temps réel Firestore
        # Récupération du Document 'Simple Data'
        # Pendant 5 minutes on génère des données aléatoires

        return 0

# Déploiement du microservcie
ray.init(address="auto")
iot_service = IoTService.remote()
ray.get(iot_service.run.remote())
```

Figure 4.6 Classe Python du service de collection des données IoT

Le code source du service est enregistré dans un fichier PPython appelé « iot_serice.py ». Pour déployer ce service dans le cluster, nous utilisons la commande ci-dessous. Le déploiement est exécuté grâce à la ligne de code « iot_service = IoTService.remote() » se trouvant à la fin du fichier.

```
Python iot_service.py
```

b) Microservice de contrôle du bâtiment à distance

Le deuxième microservice que nous mettons en place est un service qui permet le contrôle du bâtiment à distance via l'interface Web. Ce service est mis en place en utilisant les API *Ray Serve*, L'API FastAPI et la classe « RemoteControl », comme le montre la Figure 4.7.

Le décorateur « @serve.deployment(name="remote_control", route_prefix="/remote") » permet la communication avec le service en envoyant des requêtes « Hypertext Transfer Protocol » HTTP à l'adresse « 192.168.2.68:8000/remote/ ». L'adresse IP 192.168.2.68 est l'adresse IP du nœud principal du cluster « HeadNode ». Le port TCP 8000 et le port par défaut de l'API *Ray Serve*. Le chemin « /remote » est un chemin de notre choix spécifique à notre microservice de contrôle à distance. L'activation et la désactivation d'une valve à distance se fait avec la requête « 192.168.2.68:8000/remote/valve_id/valve_status ». Valve_Id est un chiffre indiquant le numéro de

la valve et valve_status indique le statut désiré. La valeur « true » indique que l'on souhaite activer la valve, la valeur « false » indique que l'on souhaite désactiver la valve. Lorsqu'une requête est émise pour contrôler à distance une valve, cette requête est traitée par la méthode « def controlValve(self, valve_id: int, valve_status: str) ».

```
# Configuration du microservice
ray.init(address="auto", namespace="serve")
app = FastAPI()

# Microservices de contrôle du bâtiment à distance
@serve.deployment(name="remote_control", route_prefix="/remote")
@serve.ingress(app)
class RemoteControl:


    @app.get("/valve/{valve_id}/{valve_status}")
    def controlValve(self, valve_id: int, valve_status: str):

# Déploiement du microservice
RemoteControl.deploy()
```

Figure 4.7 Classe PPython du service de contrôle à distance

Le code source du service est enregistré dans un fichier PPython « remote_control_service.py ». Pour déployer ce service dans le cluster, nous utilisons la commande suivante :

```
Python remote_control_service.py
```

Le déploiement est exécuté grâce à la ligne de code « RemoteControl.deploy() » se trouvant à la fin du fichier.

c) Gestion des ressources de microservices

L'objectif principal de notre modèle est de garantir le calcul en temps réel. Ceci est possible par une gestion fine des ressources allouées à chaque microservice. Pour chaque microservice, il est possible de définir la quantité de CPU, GPU et mémoire RAM à allouer.

Les lignes de code ci-dessous permettent d'allouer 2 CPU, 1 GPU et 2 Giga de mémoire RAM à notre mircroservice de collection des données.

```
@ray.remote (num_cpus=2, num_gpus=1, memory=2000 * 1024 * 1024)
class IoTService(object):
```

Ces ressources allouées peuvent être modifiées à tout moment. Cependant, il suffit de redéployer le microservice en spécifiant les nouvelles valeurs de ressources. La ligne de code ci-dessous redéploye notre service avec 3 CPU, 2 GPU et 4 Giga de mémoire RAM.

```
iot_service = IoTService.remote(cpus=3, num_gpus=2, memory=4000 * 1024 * 1024)
```

4.1.2 Implémentations du module 3 - serveur Web

Le système distribué fonctionne de manière invisible. Pour examiner la fonctionnalité de l'*application distribuée*, il faudrait se connecter avec un client (e.g., ordinateur, portable, montre, etc.). Comme présenté dans notre modèle, la connexion des clients au système distribué se fait par l'intermédiaire d'un *serveur Web*. Dans cette section, nous détaillerons l'implémentation et le fonctionnement du *serveur Web*. Le code source du *serveur Web* est présenté à l'Annexe D.

4.1.2.1 Création et déploiement du serveur Web

L'implémentation du *serveur Web* est faite en Python avec l'API Django [98]. Pour notre prototype, nous utilisons la version 3.2.4 de Django. L'installation de Django se fait avec la commande suivante :

```
Python -m pip install Django
```

Une fois Django installée, il est possible de vérifier sa version avec la commande suivante :

```
Python -m django –version
```

La création d'une application Web avec Django est très simple. Nous utilisons la commande ci-dessous pour créer un nouveau site appelé FireStation :

```
django-admin startproject firestation
```

Avec l'API Django, les sites Web sont divisés en plusieurs modules réutilisables appelés applications. Pour notre serveur, nous allons rajouter deux applications Django. L'application « Root » contiendra la page d'accueil de notre *serveur Web* et l'application « Fire Emergency »

contiendra le *Dashboard* dédié aux sapeurs-pompiers. La création de ces deux applications se fait avec les commandes suivantes :

```
Python manage.py startapp root

Python manage.py startapp fireemergency
```

Une fois le serveur créé, il est déployé avec la commande suivante, dont 192.168.2.68 est l'adresse IP utilisée et 9090 est le port TCP utilisé :

```
Python manage.py runserver 192.168.2.68:9090
```

La Figure 4.8 montre la page d'accueil de notre *serveur Web* et la Figure 4.9 montre la page dédiée à la gestion des incendies par les sapeurs-pompiers.

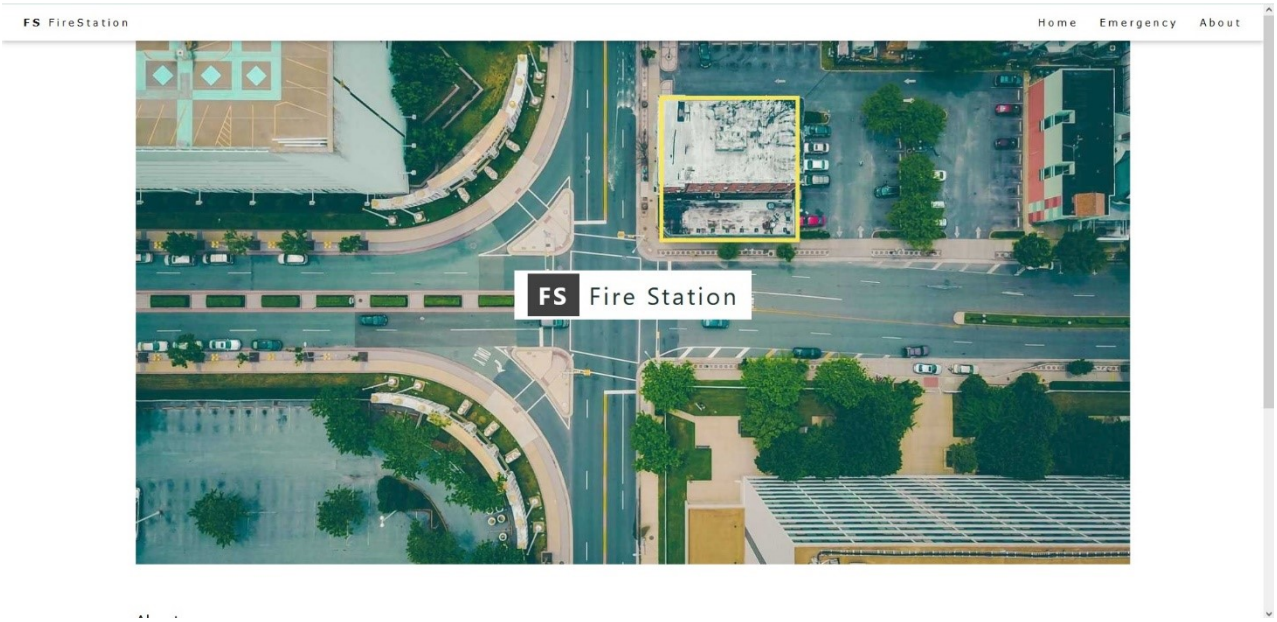

Figure 4.8 Classe Python du service de contrôle à distance

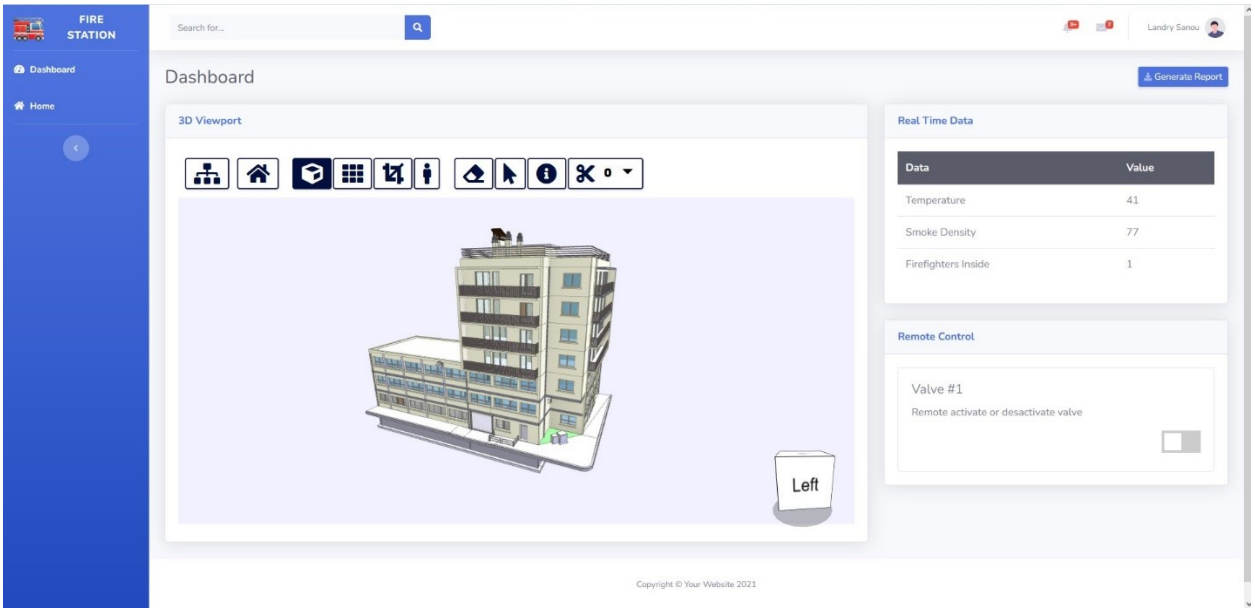

Figure 4.9 Page de gestion des incendies (Dashboard)

4.1.2.2 Implémentation de la vue 3D

L'affichage du modèle « *Building Information Modeling* » (BIM) à partir du *serveur Web* se fait avec l'API JavaScript Xeokit [99]. La Figure 4.10 nous montre un aperçu de la vue 3D « 3D viewport » situé dans la page du *Dashboard*. L'API Xeokit est incapable d'afficher le format « *Industry Foundation Classes* » (IFC) du BIM. Pour pallier ce problème, l'API Xeokit convertit les données IFC en son propre format de données appelé XKT (i.e., contraction de XeoKiT). La Figure 4.11 montre le processus de conversion utilisé par l'API Xeokit en trois étapes. La première étape consiste à convertir les données IFC au format « *JavaScript Object Notation* » (JSON) avec l'outil xeokit-metadata. Dans la deuxième étape, les données IFC sont converties au format « *Digital Asset Exchange* » (DAE) avec l'outil ifcConvert, puis les données DAE sont converties au format glTF avec l'outil COLLADA2GLTF. Dans la troisième étape, les données au format JSON et les données au format glTF sont combinées avec l'outil gltf2xkt pour créer les données au format XKT.

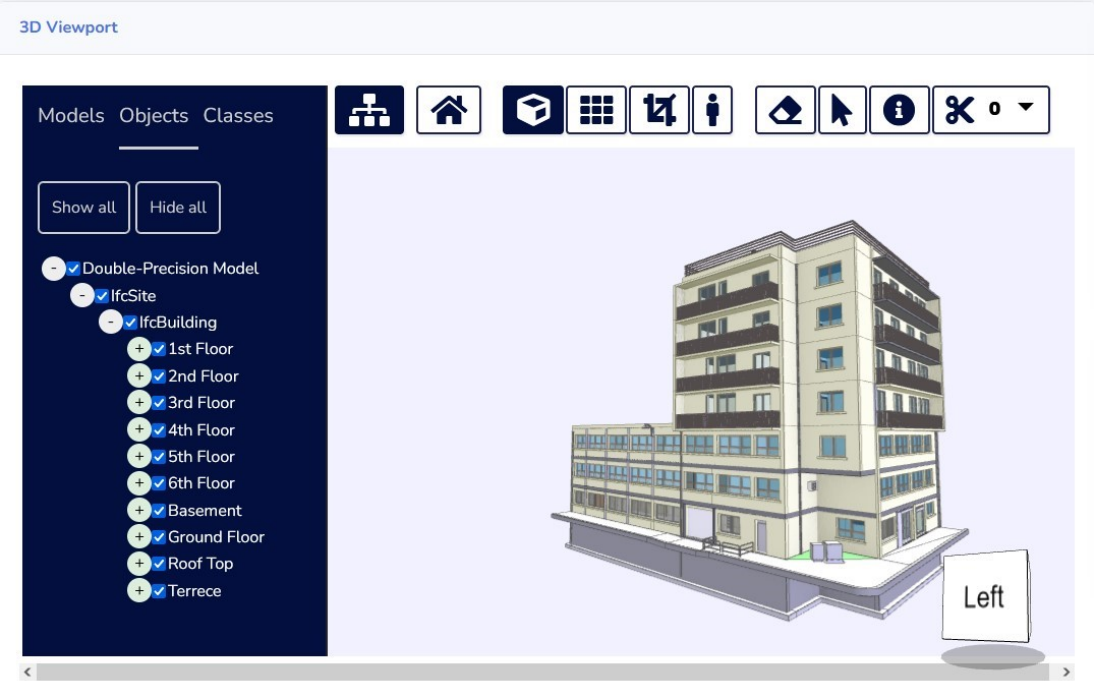

Figure 4.11 Vue 3D du bâtiment

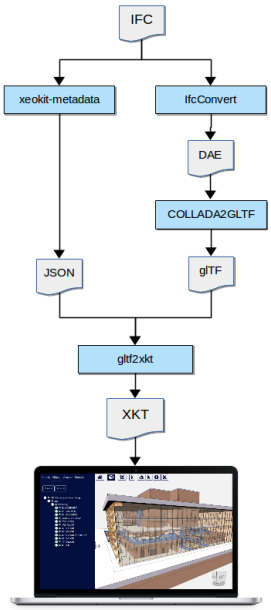

Figure 4.10 Conversion du format IFC au format XKT

4.1.2.3 Affichage des données en temps réel

Sur le *Dashboard*, la fenêtre des données en temps réel « *Real-Time Data* », affiche les données collectées en temps réel par le microservice IoTService. La Figure 4.12 montre l'affichage des données en temps réel.

Figure 4.12 Vue des données en temps réel

La vue Web est synchronisée avec la base de données en temps réel FESNet de sorte que tout changement dans la base de données entraîne une mise à jour automatique des données sur la vue Web. Cette synchronisation automatique est réalisée par la fonction JavaScript « onSnapshot » suivante de l'API Google Firebase :

```
const unsub = onSnapshot(doc(db, "iot-data", "simple-data"), (doc) => {
   console.log("Current data: ", doc.data());
   updateData(doc.data());
});
```

4.1.2.4 Contrôle à distance du bâtiment

Sur le *Dashboard*, la fenêtre de contrôle à distance « Remote Control » permet aux sapeurs-pompiers de contrôler à distance une valve. Cela se fait par l'envoi de requêtes « *Asynchronous JavaScript and XML* » (AJAX) au microservice de contrôle à distance « RemoteControl ». La

Figure 4.13 montre la vue « Remote Control » lorsque la valve est activée. La Figure 4.14 montre la vue « Remote Control » lorsque la valve est désactivée.

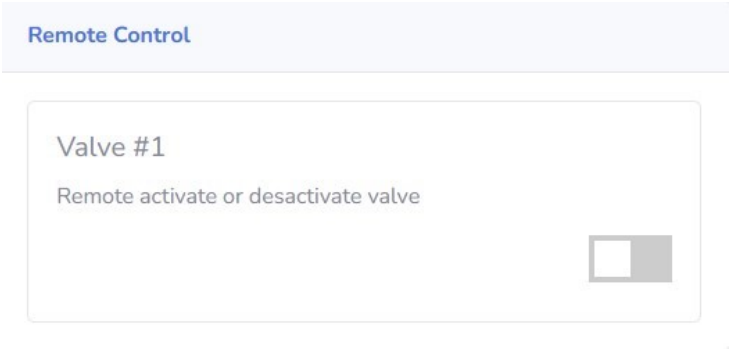

Figure 4.13 Valve désactivée

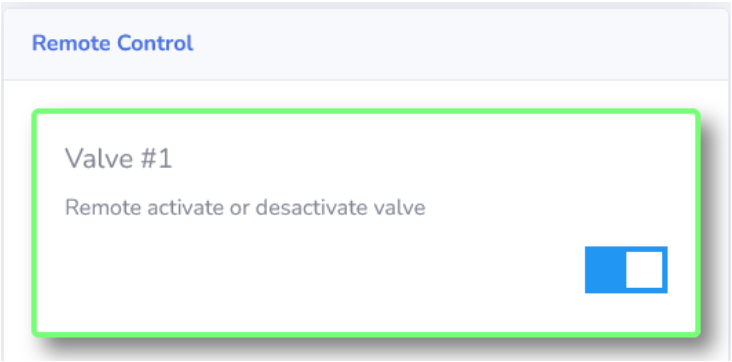

Figure 4.14 Valve activée

4.1.3 Implémentation du module 4 - application client mobile FireAlert

L'application client mobile FireAlert est une application Android dédiée aux occupants des bâtiments. En cas d'incendie, les personnes à l'intérieur du bâtiment pourront grâce à cette application recevoir des notifications et des informations utiles pour échapper à l'incendie. Pour implémenter cette application, nous utilisons Android Studio [100] et le langage de programmation Java [101]. L'API Android WebView [102] nous permet d'interconnecter l'application client mobile et le *serveur Web*. Le code source de l'application mobile est présenté à l'Annexe E.

Les Figures 4.15 à 4.19 présentent les quatre vues principales de l'application client mobile : la page d'accueil « Home », le menu de gauche, la vue des paramètres « Setting » et la vue incendie « Fire »

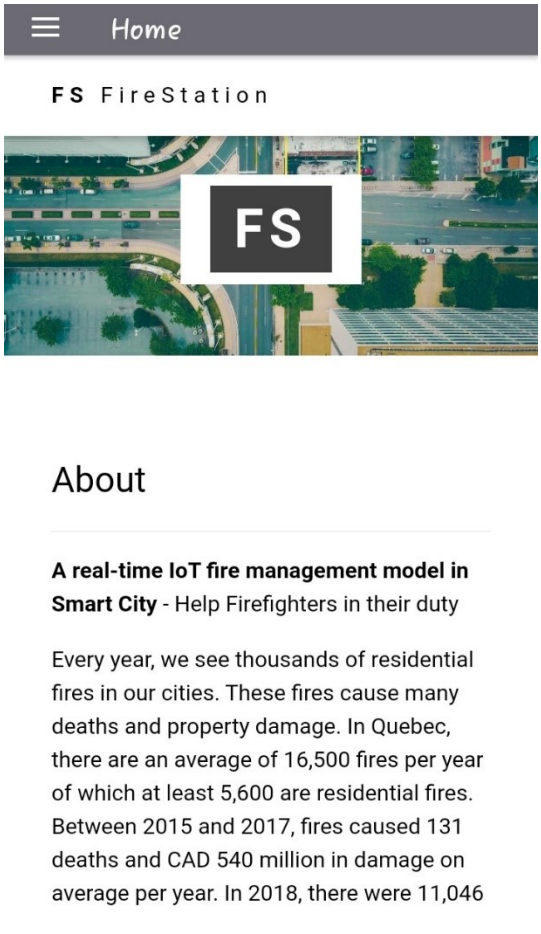

Figure 4.15 FireAlert - Page d'accueil

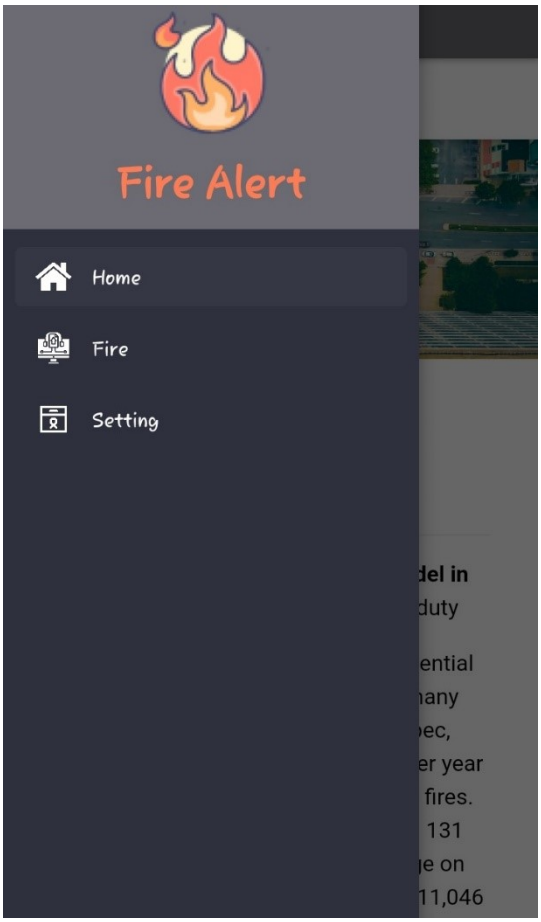

Figure 4.16 FireAlert - Menu de gauche

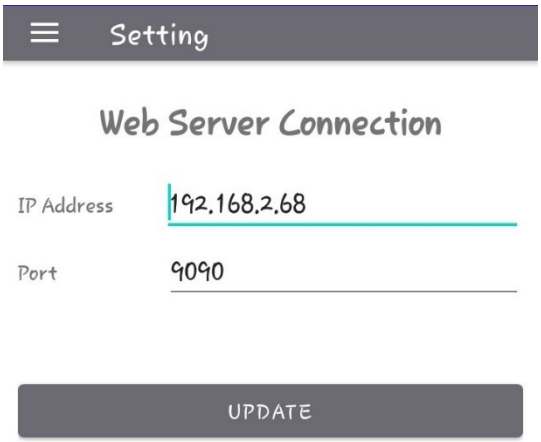

Figure 4.18 FireAlert - Vue des paramétrages

Figure 4.17 FireAlert - Vue incendie avec carte 3D du bâtiment

Figure 4.19 FireAlert - Vue incendie avec données en temps réel

4.1.3.1 Implémentation du menu de gauche

Le menu de gauche permet la navigation entre les vues Home, Fire et Setting. L'implémentation du menu de gauche se fait au niveau de la classe « MainActivity.java » avec les API Android « NavigationUI », « AppBarConfiguration » et « NavigationView », comme le montre la Figure 4.20.

4.1.3.2 Implémentation de vue des paramétrages

La page des paramétrages permet de configurer l'adresse IP et le port TCP du *serveur Web*. Une mauvaise configuration empêchera la connexion au *serveur Web* et l'accès aux données en temps réel. L'implémentation de la vue des paramétrages se fait dans la classe « SettingFragment.java »,

comme le montre la Figure 4.21. La vue des paramétrages utilise l'API Android « SharedPreferences » pour stocker les informations de configuration.

```
public class MainActivity extends AppCompatActivity {

   private AppBarConfiguration mAppBarConfiguration;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Toolbar toolbar = findViewById(R.id.toolbar);
      setSupportActionBar(toolbar);

      DrawerLayout drawer = findViewById(R.id.drawer_layout);
      NavigationView navigationView = findViewById(R.id.nav_view);
      // Passing each menu ID as a set of Ids because each
      // menu should be considered as top level destinations.
      mAppBarConfiguration = new AppBarConfiguration.Builder(
            R.id.nav_home, R.id.nav_fire, R.id.nav_setting)
            .setDrawerLayout(drawer)
            .build();
      NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment);
      NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
      NavigationUI.setupWithNavController(navigationView, navController);
   }
}
```

Figure 4.20 Classe MainActivity.java

4.1.3.3 Implémentation de vue des paramétrages

La page des paramétrages permet de configurer l'adresse IP et le port TCP du *serveur Web*. Une mauvaise configuration empêchera la connexion au *serveur Web* et l'accès aux données en temps réel. L'implémentation de la vue des paramétrages se fait dans la classe « SettingFragment.java », comme le montre la Figure 4.21. La vue des paramétrages utilise l'API Android « SharedPreferences » pour stocker les informations de configuration.

```
public class SettingFragment extends Fragment {

    private SettingViewModel settingViewModel;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        settingViewModel =
                new ViewModelProvider(this).get(SettingViewModel.class);
        View root = inflater.inflate(R.layout.fragment_setting, container, false);


        Button button = (Button) root.findViewById(R.id.update_setting);
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                String ipAdress = ((EditText)root.findViewById(R.id.ip_address)).getText().toString();
                String port     = ((EditText)root.findViewById(R.id.port)).getText().toString();

                SharedPreferences sharedPref =
getActivity().getPreferences(Context.MODE_PRIVATE);
                SharedPreferences.Editor editor = sharedPref.edit();
                editor.putString("Web_ip_address", ipAdress);
                editor.putString("Web_port", port);
                editor.apply();
            }
        });

        return root;
    }
}
```

Figure 4.21 Classe SettingFragment.java

4.1.3.4 Implémentation de la page d'accueil

La page d'accueil de l'application client mobile affiche exactement la même page d'accueil que celle du *serveur Web*. L'implémentation de la page d'accueil se fait dans la classe « HomeFragment.java », comme le montre la Figure 4.22.

```
public class HomeFragment extends Fragment {

    private HomeViewModel homeViewModel;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        homeViewModel =
                new ViewModelProvider(this).get(HomeViewModel.class);
        View root = inflater.inflate(R.layout.fragment_home, container, false);

        //get Web server data
        SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
        String ipAdress = sharedPref.getString("Web_ip_address", "192.168.2.68");
        String port     = sharedPref.getString("Web_port", "9090");
        String url = ipAdress + ":" + port;

        WebView WebView=root.findViewById(R.id.Web_view_home);
        WebView.loadUrl(url);
        WebView.setWebViewClient(new WebViewController());
        WebSettings WebSettings = WebView.getSettings();
        WebSettings.setJavaScriptEnabled(true);
        return root;
    }
}
```

Figure 4.22 Classe HomeJava

4.1.3.5 Implémentation de la vue incendie

La vue incendie sur l'application client mobile affiche une version réduite du *Dashboard* déduit aux sapeurs-pompiers. Dans cette vue dédiée aux personnes dans le bâtiment en feu est affichée une vue 3D du bâtiment, ainsi que des données en temps réel. L'implémentation de la vue incendie se fait dans la classe « FireFragment.java », comme le montre la Figure 4.23.

```
public class FireFragment extends Fragment {

    private FireViewModel galleryViewModel;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        galleryViewModel =
                new ViewModelProvider(this).get(FireViewModel.class);
        View root = inflater.inflate(R.layout.fragment_gallery, container, false);

        //get Web server data
        SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
        String ipAdress = sharedPref.getString("Web_ip_address", "192.168.2.68");
        String port     = sharedPref.getString("Web_port", "9090");
        String url = ipAdress + ":" + port + "/fire" ;

        WebView WebView=root.findViewById(R.id.Web_view_practice);
        WebView.loadUrl(url);
        WebView.setWebViewClient(new WebViewController());
        WebSettings WebSettings = WebView.getSettings();
        WebSettings.setJavaScriptEnabled(true);
        return root;
    }
}
```

Figure 4.23 Classe FireFragment.java

4.2 Évaluation du Modèle

Dans cette section, nous présentons l'évaluation du modèle proposé. Dans un premier temps, nous détaillerons les facteurs qui peuvent affecter le temps de calcul d'un algorithme pour un traitement donné dans le cas de gestion d'incendie qui exige un temps important. Ensuite, nous évaluerons en trois simulations la capacité du modèle qui consite à allouer de manière précise et dynamique les ressources (i.e., CPU, GPU et mémoire RAM) nécessaires au calcul en temps réel. Enfin, nous discuterons des résultats.

4.2.1 Variation du temps de calcul

La propagation du feu peut causer la mort des occupants du bâtiment en quelques minutes. Les sapeurs-pompiers disposent de très peu de temps pour intervenir. Les modèles existants faisant le

calcul du chemin d'évacuation [26], [60], [61] et les modèles réalisant la simulation de l'incendie utilisant le simulateur « Fire Dynamics Simulator » FDS [21], [22], [69]-[71] ne permettent pas de garantir que les calculs seront faits dans le temps imparti. De plus, la recherche du meilleur chemin d'évacuation a une importance négligeable si l'information n'est pas disponible au moment approprié. De ce fait, pour un système en temps réel, le temps de calcul des algorithmes est très important, car il doit respecter des délais très précis. Dans notre évaluation du modèle, nous nous concentrons sur les algorithmes qui demandent un temps important de calcul, comme par exemple la détermination du meilleur chemin d'évacuation, la simulation d'incendie, la détection de la position des gens dans le bâtiment, la détection et la classification des dangers, la collection et la distribution de l'information en temps réel, etc.Cependant, le temps de calcul de l'algorithme dépend de plusieurs facteurs telles que la quantité des données à traiter et les ressources disponibles (i.e., CPU, GPU et mémoire RAM). Nous utilisons l'algorithme Tri fusion en parallèle « *Parallel Merge Sort* » [103] pour mesurer la variation du temps de calcul en fonction de la quantité des données à trier et du nombre de processeurs (CPU). Cet algorithme est implémenté sous forme de microservice dans le fichier « sort_data_service.py ». Le code source complet est présenté à l'Annexe F. Le déploiement de ce microservice appelé « *SortDataService* » est effectué à l'aide de la commande suivante :

```
python sort_data_service.py
```

En utilisant cet algorithme, nous effectuons trois simulations. Dans chaque simulation, nous mesurons le temps de calcul du triage d'une quantité de données allant de 100 milles à 1 million, comme le montre la Figure 4.24.

```
@ray.remote (num_cpus=3)
class SortDataService(object):
   def __init__(self):
      pass

   ## quelques functions ##
   def run(self):
         print("sorting data")
         f = open("result.txt", "a")

         for i in range(1, 11):
            size = 100000*i
            data_unsorted = [random.randint(0, size) for _ in range(size)]

            for _ in range(3):
               start_time = time.time_ns()
               self.merge_sort_parallel(data_unsorted)
               end_time = time.time_ns()
               result = float((end_time - start_time))/1000000000
               f.write(f"{result}\n")

            f.write('--\n')

         f.close()
         return 0
```

Figure 4.24 Microservice SortDataService

Les caractéristiques de l'ordinateur utilisé pour ces trois simulations sont présentées au Tableau 4.1.

Tableau 4.1 Ordinateur utilisé

CPU	Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz 3.10 GHz
Nombre de Coeur physiques	2
Nombre de Coeur logiques	4
Mémoire RAM	16.0 GB
Carte graphique	NVIDIA GeForce 210

4.2.1.1 Simulation 1 – Mesure du temps de triage avec un seul processeur CPU

Dans la première simulation, nous mesurons le temps du triage des données en utilisant un seul processeur (CPU). Pour chaque quantité de données, nous effectuons la mesure trois fois, puis nous calculons la moyenne du temps de calcul. Le Tableau 4.2 illustre les valeurs du temps de calcul obtenues en fonction de la quantité des données à trier.

Tableau 4.2 Temps de calcul avec un seul processeur (CPU)

CPU = 1										
Données	100000	200000	300000	400000	500000	600000	700000	800000	900000	1000000
Temps 1 (s)	5.948975	6.920592	7.191194	7.152397	8.078424	10.51488	11.74482	11.81008	11.14614	12.16687
Temps 2 (s)	5.83259	7.39305	7.684009	7.201979	7.943916	9.482116	10.71805	11.13367	11.48663	13.16269
Temps 3 (s)	5.81696	6.67799	6.571157	7.765306	7.904918	9.301695	9.848574	11.45072	11.36447	13.87808
Temps moyen	**5.866175**	**6.99721**	**7.148787**	**7.373227**	**7.975752**	**9.766229**	**10.77048**	**11.46482**	**11.33241**	**13.06921**

4.2.1.2 Simulation 2 – Mesure du temps de triage avec deux processeurs (CPU)

Dans la deuxième expérience, nous mesurons le temps du triage des données en utilisant deux processeurs (CPU). Pour chaque quantité de données, nous effectuons la mesure trois fois, puis nous calculons la moyenne du temps de calcul. Le Tableau 4.3 présente les valeurs du temps de calcul obtenues en fonction de la quantité des données à trier.

Tableau 4.3 Temps de calcul avec deux processeurs (CPU)

CPU = 2										
Données	100000	200000	300000	400000	500000	600000	700000	800000	900000	1000000
Temps 1 (s)	5.799706	6.283519	6.180234	6.858992	7.022155	7.567823	7.914228	8.390892	8.635095	9.160838
Temps 2 (s)	5.770014	6.32597	7.086572	6.791169	7.553151	7.551332	8.220737	8.422693	8.672838	9.260737
Temps 3 (s)	6.136862	7.297879	7.269619	6.685614	7.158646	7.81994	8.048021	8.241481	8.822033	9.596808
Temps moyen	**5.902194**	**6.635789**	**6.845475**	**6.778592**	**7.244651**	**7.646365**	**8.060995**	**8.351689**	**8.709988**	**9.339461**

4.2.1.3 Simulation 3 – Mesure du temps de triage avec trois processeurs (CPU)

Dans la troisième expérience, nous mesurons le temps du triage des données en utilisant trois processeurs (CPU). Pour chaque quantité de données, nous effectuons la mesure trois fois, puis nous calculons la moyenne du temps de calcul. Le Tableau 4.4 résume les valeurs du temps de calcul obtenues en fonction de la quantité des données à trier.

Tableau 4.4 Temps de calcul avec trois processeurs (CPU)

CPU = 3										
Données	100000	200000	300000	400000	500000	600000	700000	800000	900000	1000000
Temps 1 (s)	5.763963	5.855894	6.227181	6.620689	6.812015	7.119442	7.513431	8.100507	8.040953	8.469315
Temps 2 (s)	5.632013	5.989843	6.241363	6.519311	6.74611	7.091187	7.606588	8.15995	8.118785	8.374354
Temps 3 (s)	5.578838	5.979696	6.400666	6.519056	6.842321	7.402779	7.746231	7.704354	8.204024	8.433953
Temps moyen	**5.658271**	**5.941811**	**6.289737**	**6.553018**	**6.800149**	**7.204469**	**7.622083**	**7.98827**	**8.121254**	**8.425874**

4.2.1.4 Résultats des simulations

Le Tableau 4.5 présente les valeurs moyennes des temps de calcul obtenues en fonction du nombre de processeurs (CPU) et la quantité des données à trier.

Tableau 4.5 Temps de calcul en fonction du nombre de processeurs (CPU) par rapport aux données à trier

Données	100000	200000	300000	400000	500000	600000	700000	800000	900000	1000000
CPU=1	5.866175	6.99721	7.148787	7.373227	7.975752	9.766229	10.77048	11.46482	11.33241	13.06921
CPU=2	5.902194	6.635789	6.845475	6.778592	7.244651	7.646365	8.060995	8.351689	8.709988	9.339461
CPU=3	5.658271	5.941811	6.289737	6.553018	6.800149	7.204469	7.622083	7.98827	8.121254	8.425874

La Figure 4.25 illustre ces représentations.

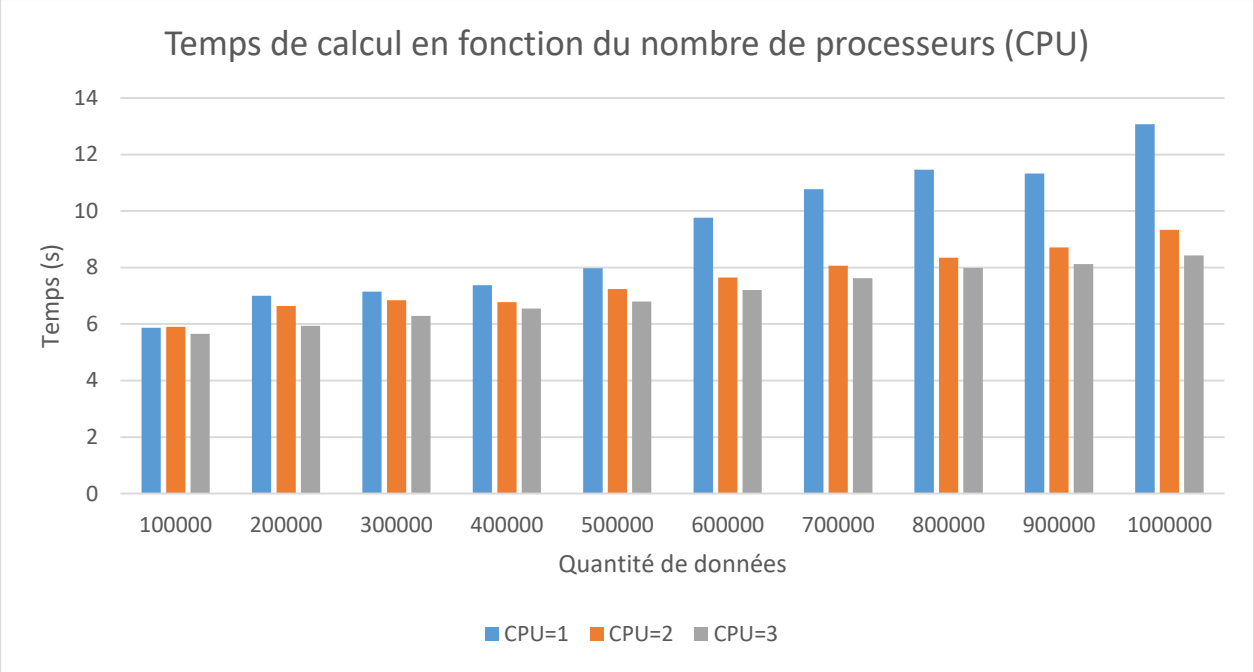

Figure 4.25 Temps de calcul en fonction du nombre de processeurs (CPU)

La Figure 4.25 montre clairement que le temps de calcul augmente en fonction de la quantité des données à trier, peu importe le nombre de processeurs (CPU) utilisé. Cependant, en augmentant le nombre de processeurs (CPU), il est possible de réduire le temps de calcul.

Pour une quantité de données 100 000, nous pouvons observer qu'avec l'utilisation de deux ou trois processeurs (CPU), le temps de calcul est plus long que celui dans le cas d'un seul processeur CPU. Ceci est dû au fait que l'initiation des processeurs (CPU) avant le début des calculs prend un certain temps. Pour une petite quantité de données, l'utilisation de plusieurs processeurs (CPU) est un désavantage. Cependant, plus la quantité de données augmente, plus la différence du temps de calcul augmente. Ainsi, l'utilisation de plusieurs processeurs (CPU) devient un avantage.

4.2.2 Allocation précise des ressources

Le temps de calcul varie en fonction des ressources (i.e., CPU, GPU et mémoire RAM) qui sont allouées à chaque microservice. Cependant, notre modèle proposé permet une allocation précise des ressources pour garantir que chaque microservice dispose des ressources nécessaires pour

traiter les données en temps réel. Nous faisons l'expérience suivante pour valider cette fonctionnalité d'allocation :

- Nous démarrons l'application distribuée avec un seul processeur (CPU). Le nombre de processeur (CPU) est indiqué avec l'option « –num-cpus=1 », comme désigné dans la commande suivante qui permet de créer le nœud « Head Node » :

```
ray --logging-level=debug start --head --port=6379 --dashboard-port=8265 -v --num-cpus=1
```

- Ensuite, nous déployons notre microservice « *SortDataService* » en indiquant l'utilisation de trois processeurs (CPU) pour garantir les temps de calculs requis. Le nombre de processeur (CPU) est indiqué à l'aide du décorateur « @ray.remote (num_cpus=3) », comme le montre la Figure 4.26.

```
@ray.remote (num_cpus=3)
class SortDataService(object):
   def __init__(self):
      pass
```

Figure 4.26 Classe du service « SortDataService »

En déployant notre microservice, nous obtenons une erreur indiquant que nous ne disposons pas assez de ressources (i.e., processeur (CPU)) pour le bon fonctionnement de notre microservice, comme le montre la Figure 4.27. En effet, nous avons déployé l'application distribuée avec un seul processeur (CPU). Cependant, nous avons besoin de trois processeurs (CPU) pour notre microservice.

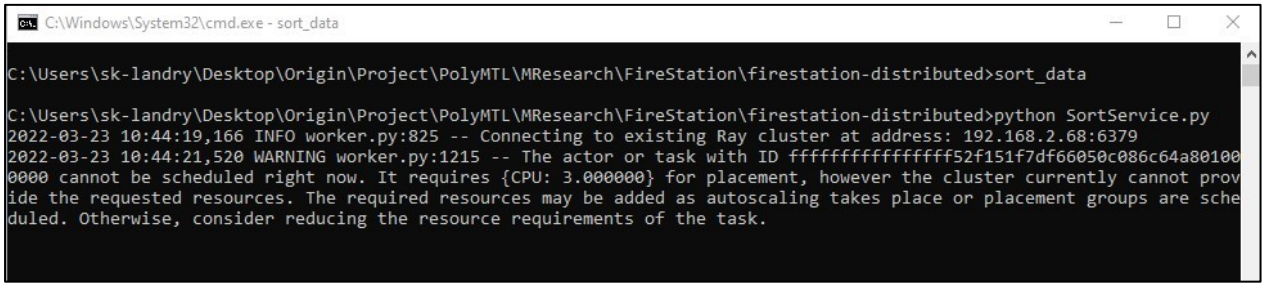

Figure 4.27 Message d'erreur de ressources insuffisantes

4.2.3 Allocation dynamique des ressources

Le modèle proposé utilise une grappe de serveurs qui est capable de s'adapter en temps réel pour garantir que les ressources nécessaires aux microservices soient disponibles. Dans la section 4.2.1.5, nous avons essayé de déployer un microservice nécessitant trois processeurs (CPU). Le cluster ne dispose pas de ressources nécessaire, il est possible de l'étendre en y rajoutant des nœuds de calcul.

La commande suivante permet de connecter à la grappe un nouveau nœud disposant de trois processeurs (CPU), comme désigné dans la commande suivante :

```
ray start --address=192.168.2.68:6379 --redis-password=5241590000000000 --num-cpus=3
```

Ce nouveau nœud est appelé un « Worker Node ». L'adresse IP 192.168.2.68:6379 est l'adresse du serveur principal « Head Node ». La Figure 4.28 illustre ces représentations.

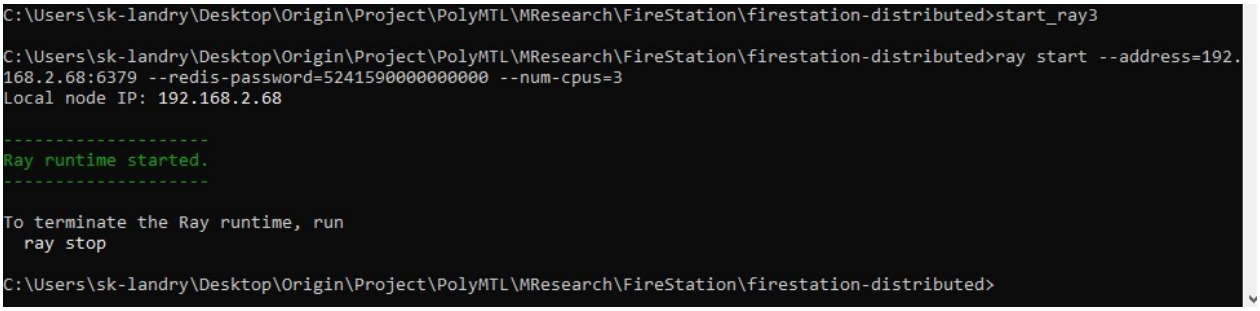

Figure 4.28 Ajout du nœud « Worker Node »

Une fois le nouveau nœud est ajouté, le cluster est maintenant capable de déployer et exécuter notre microservice, comme le montre la Figure 4.29.

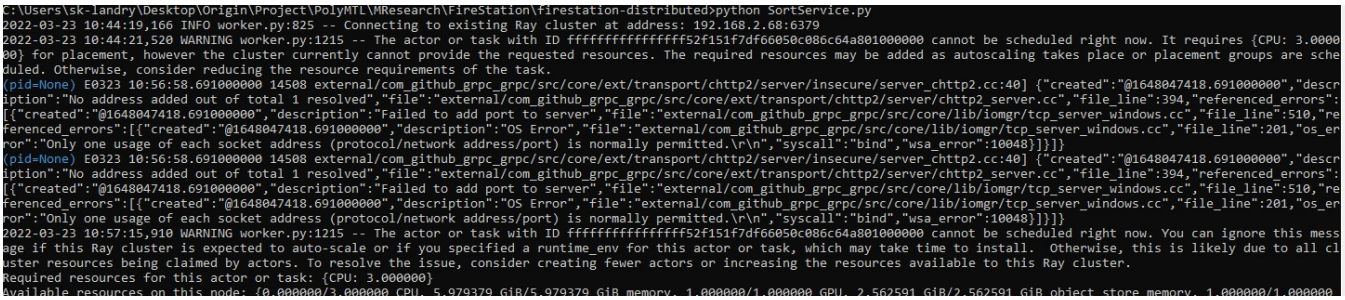

Figure 4.29 Exécution du microservice « SortDataService »

4.2.4 Discussion des résultats

Dans cette section, nous évaluons si notre modèle valide les requis annoncés au Chapitre 3 : le fonctionnent en temps réel, la diversité dans la collection de l'information, la diversité dans l'affichage de l'information, le contrôle à distance du bâtiment, l'adaptabilité et l'évolutivité du modèle.

- **Le fonctionnent en temps réel**

L'architecture distribuée et la pré-allocation de ressources (i.e., CPU, GPU et mémoire RAM) pour chaque service garantit que chaque service dispose des ressources nécessaires pour effectuer ses calculs dans les temps impartis. Pour ce qui est du traitement de l'information, le modèle garantit le calcul en temps réel. Cependant, pour ce qui est du processus total (i.e., collecte, traitement et affichage), cela dépendra des types de réseaux utilisés (e.g., LoraWan, 5G, etc.).

- **La diversité dans la collection de l'information**

Le modèle proposé permet de déployer un service pour chaque type de donnée pouvant être collecté. Cette flexibilité permet au modèle de n'avoir aucune limitation sur le type de données pouvant être collecté.

- **La diversité dans l'affichage de l'information**

Le modèle proposé utilise un *serveur Web* comme intermédiaire entre la plateforme distribuée et le client. Ceci garantit que tout appareil connecté (e.g., ordinateur, portable, montre, etc.) et capable d'utiliser le protocole http peut se connecter et afficher l'information requise.

- **Le contrôle à distance du bâtiment**

Le modèle dispose d'un service de « Remote Control » qui permet le contrôle à distance du bâtiment à partir du client.

- **L'adaptabilité et l'évolutivité du modèle**

L'architecture microservice du système distribué permet au modèle d'être adaptable et évolutif. Le modèle est adaptable, car il permet le déploiement des services aux besoins. En fonction du

bâtiment, de la quantité de personnes et des conditions de l'incendie, le modèle pourra allouer les ressources et services spécifiques à cette situation. Le modèle est évolutif où nous pouvons lui rajouter de nouvelles fonctionnalités (i.e., nouveaux services, nouveaux algorithmes, nouveaux affichages, nouveaux types de données) sans le briser.

Par rapport aux modèles existants, le nouveau modèle proposé garantit le calcul en temps réel. De plus, il est plus agile et n'impose pas un fonctionnement spécifique aux sapeurs-pompiers. Ce modèle permet donc d'exploiter l'expérience des sapeurs-pompiers au maximum en les laissant ajuster le modèle à leurs besoins.

CHAPITRE 5 CONCLUSION

Après avoir présenté l'ensemble des résultats du nouveau modèle proposé pour la gestion des incendies en temps réel dans les *villes intelligentes*, il nous revient maintenant de conclure le mémoire. La première section expose la synthèse des travaux. La deuxième section de cette conclusion présente les limitations de nos travaux. Enfin, la troisième section se concentre sur les travaux futurs.

5.1 Synthèse des travaux

Le travail présenté constitue les premières étapes vers un travail de recherche plus poussé. Nous avons identifié les défis rencontrés par les sapeurs-pompiers lors d'un incendie et analysé les solutions existantes, afin de proposer un modèle pertinent. Les incendies dans les bâtiments résidentiels sont des situations très stressantes où les pompiers disposent de très peu de temps (i.e., 5 à 10 minutes) pour sauver des vies et contrôler le feu. Pour aider les sapeurs-pompiers dans leur tâche, de nombreuses solutions utilisant les *« Technologies de l'Information et de la Communication » (TICs)* ont été mis en place. L'avènement des *villes intelligentes* et des nouvelles techniques technologies telles que la *5G*, les *objets connectés « Internet of Things » (IoT)*, le « *Building Information Model* » *(BIM)*, et l'*apprentissage machine* ont permis la création de solutions innovantes. Les solutions de gestion des incendies fonctionnent en trois étapes : *collecter l'information*, *traiter l'information* et *afficher l'information*. Les informations collectées consistent le plus souvent aux données sur la structure du bâtiment, aux données médicales des pompiers, aux données images et vidéos et aux données IoT (e.g., température, CO_2, etc.). Les traitements réalisés sur l'information sont le calcul de chemins d'évacuation, la surveillance de la santé des pompiers, la détection du point d'origine du feu, la détection et classification des dangers, la collection et distribution de l'information en temps réel et la simulation d'incendie. Les données utiles générées sont ensuite affichées avec plusieurs méthodes telles que la vue 3D, l'application mobile, l'application Web, la réalité augmentée (AR) et la réalité virtuelle (VR).

Les solutions existantes présentent quelques lacunes comme l'incapacité de fonctionner en temps réel et l'incapacité de s'adapter à la plupart des situations d'incendie. Pour être en temps réel, il faudrait respecter deux contraintes. La première est d'avoir un fonctionnement continu. La deuxième est de respecter une contrainte de temps bien définie, généralement de l'ordre de la

microseconde ou de la milliseconde. Les modèles en temps réel proposés, tels que ceux faisant du calcul du chemin en temps réel, respectent la première contrainte. De plus, ces modèles n'exploitent pas les données sonores. L'utilisation des données sonores peut se faire par le « *Sound Event Detection* » (SED) ou le « *Acoustic Scene Classification* » (ASC), qui permet de localiser les personnes dans le bâtiment. Enfin, les solutions existantes sont mal adaptées aux personnes handicapées (i.e., malvoyant, sourd, chaise roulante, personnes âgées).

Afin de pallier ces limitations, nous avons proposé un nouveau modèle de gestion des incendies garantissant le fonctionnement en temps réel. Ce modèle exploite les systèmes distribués et l'architecture microservice pour fournir un système robuste, adaptable et évolutif. Les systèmes distribués sont des systèmes où les calculs sont répartis sur plusieurs ordinateurs situés dans un réseau informatique. Ces ordinateurs qui peuvent être ajoutés et retirés du système en tout temps combinent leur ressource (i.e., CPU, GPU et mémoire RAM) pour former un système plus puissant. Les microservices sont une approche architecturale dans laquelle une application unique est composée de nombreux petits composants ou services faiblement couplés et pouvant être déployés indépendamment. La solution proposée est composée de quatre modules : le réseau de capteurs sans fil « *Fire Emergency Sensor Network* » *(FESNet)*, l'*application distribuée*, les *applications clients* et un *serveur Web* qui agit comme intermédiaire entre l'application distribuée et les applications clients. Afin de valider notre modèle, un prototype capable du traitement de l'information en temps réel et de l'affichage 3D des bâtiments a été implémenté. L'*application distribuée* et l'architecture microservice sont mises en place avec la librairie Python Ray. Quant au *serveur Web,* il a été développé avec la librairie Python Django. Par rapport aux modèles existants, le modèle proposé garantit le calcul en temps réel et il est capable de s'adapter à la diversité des scénarios de feu. Cependant, le temps d'exécution d'un algorithme dépend de la quantité des données à traiter, mais aussi des ressources (i.e., CPU, GPU et mémoire RAM) disponible. Afin de valider notre modèle, nous avons effectué trois simulations en utilisant l'algorithme « *Parallel Merge Sort* » [103]. Dans chaque simulation, nous avons utilisé un nombre différent de processeurs (i.e., un, deux et trois) pour trier une quantité de données variant de 100 milles à 1 million. Comme résultats de ces trois simulations, nous concluons que lorsque les ressources nécessaires pour le fonctionnement en temps réel d'un algorithme sont connues, le modèle proposé est capable d'allouer de manière précise les ressources demandées.

5.2 Limitations des travaux

Le modèle proposé est basé sur une architecture microservice. La principale différence avec une architecture traditionnelle (i.e., monolithique) est le découplement entre les services proposés et la plateforme de base où tournent ces services. Bien que les modèles existants viennent avec des services prédéterminés (e.g., calcul de chemin), le modèle proposé ne vient avec aucun service spécifique. Ceci n'est pas une limitation en soi, mais il s'agit d'une décision au niveau du design. En d'autres termes, notre modèle par lui-même est incomplet, il faudrait proposer des services qui puissent tourner sur cette nouvelle architecture.

Une des principales difficultés rencontrées dans cette recherche était l'absence d'environnement adéquat de test dû au manque de matériel et de données. Le réseau de capteurs sans fil *FESNet* n'a pas était mis en place, car nous ne disposons pas des capteurs et de la passerelle qui avait été prévue à cet effet. De plus, la valve d'eau pour simuler le contrôle à distance n'était pas disponible. Le prototype mis en place et les tests ont donc été menés au niveau logiciel avec les données IoT générées de manière aléatoire.

Une autre difficulté rencontrée est l'absence de « Dataset » publiques. Les données BIM et FDS sont difficiles à obtenir. Cependant, il n'existe pas de données BIM ou FDS, ni de scénarios de feu dédiés à la recherche sur la gestion des incendies. Dans les travaux existants, les auteurs ont utilisé soit des données 3D créées avec le logiciel Autodesk Revit ou des données 3D non publiques. Pour l'évaluation des modèles, les auteurs ont dû mettre en place leur propre scenario d'incendie. Cette hétérogénéité dans les données utilisées rend très difficile une comparaison précise des différents modèles.

5.3 Travaux futurs

La recherche sur la gestion des incendies avec les technologies informatiques est un domaine très intéressant. Afin de faire avancer ce domaine de recherche, nous pouvons citer quelques points à aborder comme suit.

La création de Datasets : Il faudrait développer de datasets ou scenarios de feu qui pourront être utilisés par tous les chercheurs du domaine pour évaluer leurs modèles. De tels datasets devront contenir trois types de données ou trois fichiers. Les données BIM (i.e., fichier .ifc), les données

FDS (i.e., fichier .fds) et un fichier décrivant le scenario de feu (i.e., cause du feu, point d'origine du feu, caractéristique des personnes à l'intérieur du bâtiment (e.g., nombre, âge, handicap, etc.)). Ces datasets peuvent être mis en place en se référant aux incendies réels qui arrivent dans nos villes.

La création d'environnements de test et simulation adéquats : Il faudrait ensuite mettre en œuvre une plateforme de simulation avec des critères de mesures bien définis. Par exemple, les modèles proposant le calcul de chemin d'évacuation pourront être comparés par : le temps de calcul ; le nombre de personnes évacuées, mortes ou blessées ; et le temps total de l'évacuation. En absence de telles plateformes et de critères bien définis, les comparaisons entre les modèles ne seront pas efficaces.

Le développement des sous-domaines de recherche existants : Dans la recherche sur la gestion des incendies dans les villes intelligentes, il existe deux grands sous-domaines, le premier concerne le secours des personnes dans le bâtiment (i.e., localisation et évacuation des occupants du bâtiment en feu), et le second concerne le contrôle et l'arrêt du feu (i.e., détection des causes du feu et de la simulation du feu). Dans nos travaux futurs, il serait intéressant de se focaliser dans un sous-domaine précis et améliorer les solutions déjà proposées. Comme travaux futurs, nous proposons de nous focaliser dans le premier sous-domaine qui conserve le secours des personnes dans le bâtiment en feu. De manière précise, nous proposons de créer un modèle intelligent qui utilise l'apprentissage machine ou les réseaux de neurones pour détecter l'état de santé des occupants du bâtiment en temps réel. Un tel modèle sera en mesure de détecter le niveau de brûlures ou le taux d'asphyxie d'une personne et en informer les sapeurs-pompiers. Ces derniers pourront alors préparer l'équipement de sauvetage adéquat à temps et mettre la priorité sur les personnes qui sont le plus en danger. Une fois ce modèle intelligent mis en place, il pourra être déployé en tant que microservice sur notre architecture distribuée. Les données sur l'état de santé des personnes dans le bâtiment pourront ensuite être affichées en temps réel sur la vue 3D.

RÉFÉRENCES

[1] Service de Sécurité Incendie de Montréal (SIM), "Firefighters", 2021. [En ligne]. Disponible : https://ville.montreal.qc.ca/sim/en/firefighter, [Dernier accès : 22-Dec-2021].

[2] Wang, J., Wei, G. & Dong, X. "A dynamic fire escape path planning method with BIM". J Ambient Intell Human Comput, 2021, vol. 12, pp. 10253–10265, doi: 10.1007/s12652-020-02794-2.

[3] J. Dugdale, M. T. Moghaddam and H. Muccini, "Agent-based Simulation for IoT Facilitated Building Evacuation," International Conference on Information and Communication Technologies for Disaster Management (ICT-DM), 2019, pp. 1-8, doi: 10.1109/ICT-DM47966.2019.9032909.

[4] Fire Safety Advice Centre, "Fire Emergency Evacuation Plan and the Fire Procedure", 2021. [En ligne]. Disponible : https://www.firesafe.org.uk/fire-emergency-evacuation-plan-or-fire-procedure, [Dernier accès : 22-Dec-2021].

[5] Walter W. Jones, William D. Davis, D D. Evans, David G. Holmberg, Steven T. Bushby, K A. Reed, "Workshop to Define Information Needed by Emergency Responders during Building Emergencies", NIST Interagency/Internal Report (NISTIR), 2005, pp. 7193, doi: 10.6028/NIST.IR.7193.

[6] IoT Agenda, "Smart City". [En ligne]. Disponible : https://internetofthingsagenda.techtarget.com/definition/smart-city, [Dernier accès : 19-Nov-2021].

[7] A. Latifah, S. H. Supangkat and A. Ramelan, "Smart Building: A Literature Review" International Conference on ICT for Smart Society (ICISS), 2020, pp. 1-6, doi: 10.1109/ICISS50791.2020.9307552.

[8] Shi, Jianyong & Jicao, Dao & Jiang, Liu & Pan, Zeyu. "Research on IFC- and FDS-Based Information Sharing for Building Fire Safety Analysis", Advances in Civil Engineering. 2019, pp. 1-18, doi: 10.1155/2019/3604369.

[9] Yuan Tian and Tonggui Jiang and Feng Wu, "Research on the Application of BIM Technology in Bridge Engineering", IOP Conf. Ser.: Earth Environ. Sci. 798 012014, 2018, doi: 10.1088/1755-1315/798/1/012014.

[10] Eftekharirad, Roshanak & Nik Bakht, Mazdak & Hammad, Amin, "Extending IFC for Fire Emergency Real-Time Management Using Sensors and Occupant Information", 2018, doi: 10.22260/ISARC2018/0137.

[11] S. Chaudhary, R. Johari, R. Bhatia, K. Gupta and A. Bhatnagar, "CRAIoT: Concept, Review and Application(s) of IoT," 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), 2019, pp. 1-4, doi: 10.1109/IoT-SIU.2019.8777467.

[12] R. Miller, "Response Time in Man-Computer Conversational Transactions," in Managing Requirements Knowledge, International Workshop on, SAN FRANSISCO, 1968 pp. 267. doi: 10.1109/AFIPS.1968.149 url: https://doi.ieeecomputersociety.org/10.1109/AFIPS.1968.149.

[13] Wenhong Tian, Yong Zhao, "Optimized Cloud Resource Management and Scheduling", Theories and Practices 1st Edition, 2014, ISBN: 9780128016459.

[14] Ministère de la Sécurité publique du Québec, "Statistiques concernant les incendies", 2019. [En ligne]. Disponible : https://www.securitepublique.gouv.qc.ca/securite-incendie/prevenir-incendie/semaine-prevention-incendies/statistiques-concernant-les-incendies.html, [Dernier accès : 06-Nov-2021]

[15] Ministry of the Solicitor General (Ontario), "Ontario Fire Incident Summary ", 2020. [En ligne]. Disponible : https://www.mcscs.jus.gov.on.ca/english/FireMarshal/MediaRelationsandResources/FireStatistics/OntarioFires/AllFireIncidents/stats_all_fires.html, [Dernier accès : 06-Nov-2021].

[16] Haosen Chen, Lei Hou, Guomin (Kevin) Zhang, Sungkon Moon, "Development of BIM, IoT and AR/VR technologies for fire safety and upskilling", 2021, Automation in Construction, vol. 125, pp. 103631, ISSN 0926-5805, doi: 10.1016/j.autcon.2021.103631.

[17] Wu, X., Park, Y., Li, A. et al. "Smart Detection of Fire Source in Tunnel Based on the Numerical Database and Artificial Intelligence", 2021. Fire Technology vol. 57, pp. 657–682, doi: 10.1007/s10694-020-00985-z.

[18] I. Sergi, A. Malagnino, R. C. Rosito, V. Lacasa, A. Corallo and L. Patrono, "Integrating BIM and IoT Technologies in Innovative Fire Management Systems," 2020 5th International Conference on Smart and Sustainable Technologies (SpliTech), 2020, pp. 1-5, doi: 10.23919/SpliTech49282.2020.9243838.

[19] Y. Gao, "IoT and Lightbend based Intelligent Platform for Fire Monitoring System," Second International Conference on Inventive Research in Computing Applications (ICIRCA), 2020, pp. 22-25, doi: 10.1109/ICIRCA48905.2020.9182960.

[20] Beata, Paul & Jeffers, Ann & Kamat, Vineet. "Real-Time Fire Monitoring and Visualization for the Post-Ignition Fire State in a Building". Fire Technology, 2018, vol. 54, doi: 10.1007/s10694-018-0723-1.

[21] Xu, Mingbiao & Peng, Dehong, "PyroSim-Based Numerical Simulation of Fire Safety and Evacuation Behaviour of College Buildings", International Journal of Safety and Security Engineering, 2020, vol. 10, pp. 293-299, doi: 10.18280/ijsse.100218.

[22] Lu, Xinzheng & Yang, Zhebiao & Xu, Zhen & Xiong, Chen, "Scenario Simulation of Indoor Post-earthquake Fire Rescue Based on Building Information Model and Virtual Reality", 2021, doi: 10.1007/978-3-030-51295-8_85.

[23] Xu, Z., Wei, W., Jin, W., & Xue, Q. "Virtual drill for indoor fire evacuations considering occupant physical collisions". Automation in Construction, 2020, vol. 109, pp. 102999.

[24] Mirahadi, Farid & Mccabe, Brenda & Shahi, Arash. "IFC-centric performance-based evaluation of building evacuations using fire dynamics simulation and agent-based modeling", Automation in Construction, 2019, vol. 101, pp. 1-16, doi: 10.1016/j.autcon.2019.01.007.

[25] Uwe Rüppel, Kristian Schatz, "Designing a BIM-based serious game for fire safety evacuation simulations", Advanced Engineering Informatics, vol. 25, Issue 4, 2011, pp. 600-611, ISSN 1474-0346, doi: 10.1016/j.aei.2011.08.001.

[26] Yan, Feng-ting & Hu, Yong-hao & Jia, Jin-yuan & Guo, Qinghua & Zhu, He-hua & Pan, Zhi-geng, "RFES: a real-time fire evacuation system for Mobile Web3D", Frontiers of Information Technology & Electronic Engineering, 2019, vol. 20, pp. 1061-1074, doi: 10.1631/FITEE.1700548.

[27] Ting-Kwei Wang, Chang Qin, "Integration of BIM, Bayesian Belief Network, and Ant Colony Algorithm for Assessing Fall Risk and Route Planning", Construction Research Congress, 2018, pp. 207-220, doi:10.1061/9780784481288.021.

[28] Xu, Zhen & Zhang, Zongcai & Lu, Xinzheng & Zeng, Xiang & Guan, Hong, Post-earthquake fire simulation considering overall seismic damage of sprinkler systems based on BIM and FEMA P-58. Automation in Construction, 2018, vol. 90, pp. 9-22, doi: 10.1016/j.autcon.2018.02.015.

[29] Dimyadi, Johannes & Spearpoint, Michael & Amor, Robert. "Sharing Building Information using the IFC Data Model for FDS Fire Simulation", Fire Safety Science, 2008, vol. 9, pp. 1329-1340, doi: 10.3801/IAFSS.FSS.9-1329.

[30] R. J. Scherer et al., "Towards a Multimodel Approach for Simulation of Crowd Behaviour Under fire and Toxic gas expansion in buildings," Winter Simulation Conference (WSC), 2018, pp. 3965-3976, doi: 10.1109/WSC.2018.8632341.

[31] Dimyadi, Johannes & Solihin, Wawan & Amor, Robert. "Using IFC to Support Enclosure Fire Dynamics Simulation", European Group for Intelligent Computing in Engineering (EG-ICE), 2018, doi: 10.1007/978-3-319-91638-5_19.

[32] Service de Sécurité Incendie de Montréal (SIM), "Statistique des incendies – Rapport 2020", 2021. [En ligne]. Disponible : https://ville.montreal.qc.ca/sim/en/activity-report, [Dernier accès : 06-Nov-2021].

[33] Kempna, Kamila & Hozjan, Tomaz & Smolka, Jan & Hošťálková, Markéta & Bergerová, Denisa & Kolaitis, Dionysios & Lukaszewska, Elzbieta & Steen-Hansen, Anne, "Fire-fighting and Bio-Based Materials", 2019, doi: 10.3929/ethz-b-000319565p.

[34] Service de Sécurité Incendie de Montréal (SIM), "Feux de cuisson". [En ligne]. Disponible : https://ville.montreal.qc.ca/sim/feux-de-cuisson, [Dernier accès : 19-Nov-2021].

[35] Radio Canada, "Un incendie majeur force l'évacuation de 149 logements à Dollard-des-Ormeaux", 2021. [En ligne]. Disponible : https://ici.radio-canada.ca/nouvelle/1799216/batiment-appartements-brasier-pompiers-brunswick-deacon, [Dernier accès : 22-Dec-2021].

[36] Cavlie, "Smart city and micro-mobility". [En ligne]. Disponible : https://www.cavliwireless.com/iot-applications-industries/smart-city.html, [Dernier accès : 22-Dec-2021].

[37] Mappedin, "Your Guide to Smart Buildings", 2020. [En ligne]. Disponible : https://resources.mappedin.com/blog/your-guide-to-smart-buildings, [Dernier accès : 22-Dec-2021].

[38] Fire Engineering, "Survivability Profiling: How Long Can Victims Survive in a Fire?", 2010. [En ligne]. Disponible : https://www.fireengineering.com/firefighting/survivability-profiling-how-long-can-victims-survive-in-a-fire-2/, [Dernier accès : 06-Nov-2021].

[39] William Grosshandler, Nelson Bryner, Daniel Madrzykowski, Kenneth Kuntz, "Report of the Technical Investigation of the Station Nightclub Fire", 2005, National Construction Safety Team Act Reports (NIST NCSTAR), url: https://www.nist.gov/publications/report-technical-investigation-station-nightclub-fire-nist-ncstar-2-volume-1

[40] Stanton, N. A.; Chambers, P. R. G. & Piggott, J., "Situational awareness and Safety", Situational awareness and safety. Safety Science, 200, 1vol. 39, pp. 189-204.

[41] Fatih Cavdur, Asli Sebatli-Saglam, Merve Kose-Kucuk, "A spreadsheet-based decision support tool for temporary-disaster-response facilities allocation", Safety Science, 2020, vol. 124, pp. 104581, ISSN 0925-7535, doi: 10.1016/j.ssci.2019.104581.

[42] S. -H. Lim, S. Gi Hong and K. Bok Lee, "Design and Implement of Firefighting-Active-Survey Terminal for Shortening Rescue Time", International Conference on Information and Communication Technology Convergence (ICTC), 2020, pp. 1577-1579, doi: 10.1109/ICTC49870.2020.9289498.

[43] Peeters, M. & Compernolle, T. & Passel, Steven, "Influence of information provided at the moment of a fire alarm on the choice of exit", Fire Safety Journal, 2020, vol. 117, pp. 103221, doi: 10.1016/j.firesaf.2020.103221.

[44] Wu, Baohu & Zhang, S, "Integration of GIS and BIM for indoor Geovisual Analytics". ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. XLI-B2, 2016, pp. 455-458, doi: 10.5194/isprs-archives-XLI-B2-455-2016.

[45] Meng, Qingfeng & Zhang, Yifan & Li, Zhen & Shi, Weixiang & Wang, Jun & Sun, Yanhui & Xu, Li & Wang, Xiangyu. "A review of integrated applications of BIM and related technologies in whole building life cycle", Engineering, Construction and Architectural Management. ahead-of-print, 2020, doi: 10.1108/ECAM-09-2019-0511.

[46] Perera, U., Kulatunga, U., Abdeen, F., Sepasgozar, S. and Tennakoon, M., "Application of building information modelling for fire hazard management in high-rise buildings: an investigation in Sri Lanka", Intelligent Buildings International, 2021, pp.1-15. doi: 10.1080/17508975.2021.1874858.

[47] Sun, Qi & Turkan, Yelda, "A BIM-based simulation framework for fire safety management and investigation of the critical factors affecting human evacuation performance", Advanced Engineering Informatics, 2020, vol. 44, pp. 101093, doi: 10.1016/j.aei.2020.101093.

[48] Liu, Zhansheng & Zhang, Anshan & Wang, Wensi, "A Framework for an Indoor Safety Management System Based on Digital Twin", Sensors, 2020, doi: 20. 5771. 10.3390/s20205771.

[49] A. Kanak, İ. Arif, O. Kumaş and S. Ergün, "Extending BIM to Urban Semantic Context for Data-driven Crisis Preparedness," 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2020, pp. 3813-3818, doi: 10.1109/SMC42975.2020.9283158.

[50] Q. Peijun, Y. Tao, Z. Xiaofei, J. Shaoen and C. Yanbo, "Emergency Evacuation Simulation Based on BIM Technology for Scientific Facility," 2020 International Signal Processing, Communications and Engineering Management Conference (ISPCEM), 2020, pp. 210-213, doi: 10.1109/ISPCEM52197.2020.00049.

[51] J. V. Raj and T. V. Sarath, "An IoT based Real-Time Stress Detection System for Fire-Fighters" 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 2019, pp. 354-360, doi: 10.1109/ICCS45141.2019.9065866.

[52] Cheng, Jack & Chen, Keyu & Wong, Peter & Chen, Weiwei & Li, Chun, "Graph-based network generation and CCTV processing techniques for fire evacuation", Building Research & Information, 2020, vol. 49, pp. 1-18, doi: 10.1080/09613218.2020.1759397.

[53] Fong, Simon & Bhatt, Chintan & Korzun, Dmitry & Yang, Shuang-Hua & Yang, Lili, "Internet of Breath (IoB): Integrative Indoor Gas Sensor Applications for Emergency Control and Occupancy Detection", 2019, doi: 10.1007/978-3-319-91337-7_32.

[54] R. K. Kodali and S. Yerroju, "IoT based smart emergency response system for fire hazards," 2017 3rd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2017, pp. 194-199, doi: 10.1109/ICATCCT.2017.8389132.

[55] G. Cavalera et al., "An Innovative Smart System based on IoT Technologies for Fire and Danger Situations," 2019 4th International Conference on Smart and Sustainable Technologies (SpliTech), 2019, pp. 1-6, doi: 10.23919/SpliTech.2019.8783059.

[56] REAX Engineering, “ASET/RSET Analysis”. [En ligne]. Disponible : https://reaxengineering.com/fire-protection-engineering/aset-rset-analysis, [Dernier accès : 7-Nov-2021].

[57] Zhang, Jun & Issa, Raja & Liu, Rui. "A Cyber-Physical System Approach for Intelligent Building Emergency Evacuation Signage Guidance", 2018, pp. 535-541, doi: 10.1061/9780784481288.052.

[58] A. Khan, A. A. Aesha, J. S. Aka, S. M. F. Rahman and M. J. Rahman, "An IoT Based Intelligent Fire Evacuation System," 2018 21st International Conference of Computer and Information Technology (ICCIT), 2018, pp. 1-6, doi: 10.1109/ICCITECHN.2018.8631945.

[59] Neto, Joaquim & Morais, A. Jorge & Gonçalves, Ramiro & Coelho, A., "A Multi-agent System for Recommending Fire Evacuation Routes in Buildings, Based on Context and IoT", 2019, doi: 10.1007/978-3-030-24299-2_34.

[60] F. Yan, Y. Hu, J. Jia, Q. Guo and H. Zhu, "Lightweight and Intelligent Real-Time Fire Evacuation on Mobile-WebVR Building," 2017 International Conference on Virtual Reality and Visualization (ICVRV), 2017, pp. 282-287, doi: 10.1109/ICVRV.2017.00065.

[61] Chou, Jui-Sheng & Cheng, Min-Yuan & Hsieh, Yo-Ming & Yang, I-Tung & Hsu, Hsin-Ting, "Optimal path planning in real time for dynamic building fire rescue operations using wireless sensors and visual guidance", Automation in Construction, 2019, vol. 99, pp. 1-17, doi: 10.1016/j.autcon.2018.11.020.

[62] Balboa, Adriana & González-Villa, Javier & Cuesta, Arturo & Abreu, Orlando & Alvear, Daniel. "Testing a real-time intelligent evacuation guiding system for complex buildings", Safety Science, 2020, vol. 132, pp. 104970, doi: 10.1016/j.ssci.2020.104970.

[63] Yen, Hong-Hsu & Lin, Cheng-Han & Tsao, Hung-Wei, "Time-Aware and Temperature-Aware Fire Evacuation Path Algorithm in IoT-Enabled Multi-Story Multi-Exit Buildings", Sensors, 2020, vol. 21, pp. 111, doi: 10.3390/s21010111.

[64] Li, Nan & Becerik-Gerber, Burcin & Krishnamachari, Bhaskar & Soibelman, Lucio, "4A BIM centered indoor localization algorithm to support building fire emergency response operations", Automation in Construction, 2014, vol. 42, pp. 78–89, doi: 10.1016/j.autcon.2014.02.019.

[65] A. Wang and J. Cheng, "Realization of Dynamic Information Data Transmission Function of Fire Scene Based on NB-iot," 2020 3rd International Conference on Electron Device and Mechanical Engineering (ICEDME), 2020, pp. 710-713, doi: 10.1109/ICEDME50972.2020.00167.

[66] GSMA, "Narrowband – Internet of Things (NB-IoT)". [En ligne]. Disponible : https://www.gsma.com/iot/narrow-band-internet-of-things-nb-iot/, [Dernier accès : 7-Nov-2021].

[67] L. Liu, P. Zhou, J. Wu, R. Wang, X. Fan and H. Zhu, "A Data Forwarding Approach for Opportunistic Mobile Sensor Networks in Fire-Rescue Scenario" 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD)), 2018, pp. 630-635, doi: 10.1109/CSCWD.2018.8465293.

[68] NIST, "Simulateur FDS". [En ligne]. Disponible : https://www.nist.gov/services-resources/software/fds-and-smokeview, [Dernier accès : 7-Nov-2021].

[69] M. Gamaleldin, Z. Liao, M. Asfour and L. Zhao, "Optimizing the Egress Route Using a New Smoke Emulator IoT System," in IEEE Internet of Things Journal, vol. 8, no. 11, pp. 9373-9382, 1 June1, 2021, doi: 10.1109/JIOT.2021.3056675.

[70] Y. Gao and H. Xue, "Design and simulation of building fire protection system based on BIM technology," 2017 29th Chinese Control and Decision Conference (CCDC), 2017, pp. 2729-2732, doi: 10.1109/CCDC.2017.7978976.

[71] Guofeng Ma, Zhijiang Wu, "BIM-based building fire emergency management: Combining building users' behavior decisions", Automation in Construction, 2020, vol. 109, pp. 102975, ISSN 0926-5805, doi: 10.1016/j.autcon.2019.102975.

[72] Dimyadi, Johannes & Spearpoint, Michael & Amor, Robert, "Using BIM to support simulation of compliant building evacuation", European Conference on Product and Process Modelling (ECPPM) 2016 At: Limassol, Cyprus, 2016.

[73] YAN, Fengting et al., "Smart Fire Evacuation Service Based on Internet of Things Computing for Web3D", Journal of Internet Technology, [S.l.], vol. 20, n. 2, pp. 521-532, mar. 2019, ISSN 2079-4029, url: https://jit.ndhu.edu.tw/article/view/2024.

[74] Chen, Xiu-Shan & Liu, Chi-Chang & Wu, I-Chen, "A BIM-based visualization and warning system for fire rescue", Advanced Engineering Informatics, 2018, vol. 37, doi: 10.1016/j.aei.2018.04.015.

[75] S. Lee, S. Park, S. Kim, S. H. Lee, S. Lee and S. Park, "Indoor Navigation System for Evacuation Route in case of Fire by using Environment and Location Data," 2020 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan), 2020, pp. 1-2, doi: 10.1109/ICCE-Taiwan49838.2020.9258143.

[76] H. Jiang, "Mobile Fire Evacuation System for Large Public Buildings Based on Artificial Intelligence and IoT", in IEEE Access, 2019, vol. 7, pp. 64101-64109, doi: 10.1109/ACCESS.2019.2915241.

[77] ThunderHead Engineering, "Estimating Simulation Requirements". [En ligne]. Disponible : https://www.thunderheadeng.com/pyrosim/benchmarks/, [Dernier accès : 06-Nov-2021]

[78] K. Imoto, N. Tonami, Y. Koizumi, M. Yasuda, R. Yamanishi and Y. Yamashita, "Sound Event Detection by Multitask Learning of Sound Events and Scenes with Soft Scene Labels," ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 621-625, doi: 10.1109/ICASSP40776.2020.9053912.

[79] U. S. Prakruthi, D. Kiran and H. Ramasangu, "High performance neural network based acoustic scene classification," 2018 2nd International Conference on Inventive Systems and Control (ICISC), 2018, pp. 781-784, doi: 10.1109/ICISC.2018.8398905.

[80] Statistics Canada, "Mobility issues", 2012. [En ligne]. Disponible : https://www150.statcan.gc.ca/n1/pub/89-654-x/89-654-x2016005-eng.htm, [Dernier accès : 06-Nov-2021].

[81] Kapalo, Peter & Spodyniuk, Nadiia, "Effect of the variable air volume on energy consumption - case study", IOP Conference Series: Materials Science and Engineering, 2018, vol. 415, pp. 012027, doi: 10.1088/1757-899X/415/1/012027.

[82] Felfernig, Alexander & Polat-Erdeniz, Seda & Uran, Christoph & Reiterer, Stefan & Atas, Muesluem & Tran, Thi Ngoc Trang & Azzoni, Paolo & Kiraly, Csaba & Dolui, Koustabh, "An overview of recommender systems in the internet of things", Journal of Intelligent Information Systems, 2019, doi: 52. 10.1007/s10844-018-0530-7.

[83] Kazmi, Aqeel & O'Hare, Gregory & O'Grady, Michael & Ruzzelli, Antonio & Delaney, Declan, "A Review of Wireless Sensor Network Enabled Building Energy Management Systems", ACM Transactions on Sensor Networks, 2014, doi: 10. 10.1145/2532644.

[84] Wikipedia, "Real Time Computing". [En ligne]. Disponible : https://en.wikipedia.org/wiki/Real-time_computing, [Dernier accès : 21-Nov-2021]

[85] PubNub, " How Fast is Real-time? Human Perception and Technology ", [En ligne]. Disponible : https://www.pubnub.com/blog/how-fast-is-realtime-human-perception-and-technology/, [Dernier accès : 21-Nov-2021]

[86] Wikipedia, "Distributed Computing". [En ligne]. Disponible : https://en.wikipedia.org/wiki/Distributed_computing, [Dernier accès : 21-Nov-2021]

[87] IBM, "Microservices". [En ligne]. Disponible : https://www.ibm.com/cloud/learn/microservices, [Dernier accès : 21-Nov-2021]

[88] A content Box, "Avantage et inconvénients des micro-services". [En ligne]. Disponible : https://www.acontentbox.org/what-are-microservices-the-pros-cons-and-how-they-work/, [Dernier accès : 21-Nov-2021]

[89] Raima, "9 Attributes of live real-time database", [En ligne]. Disponible : https://raima.com/live-real-time-databases/, [Dernier accès : 21-Nov-2021]

[90] Geethmi Nimantha Dissanayake, "A study on real-time database technology and its applications", 2020. [En ligne]. Disponible : https://thekeep.eiu.edu/theses/4822/, [Dernier accès : 21-Nov-2021]

[91] Wikipedia, "Time Complexity". [En ligne]. Disponible : https://en.wikipedia.org/wiki/Time_complexity, [Dernier accès : 13-Mar-2022]

[92] Ben-Gurion University of the Negev. "Web Server", [En ligne]. Disponible : http://132.72.155.230:3838/js/web-servers-1.html, [Dernier accès : 13-Mar-2022]

[93] Ray. [En ligne]. Disponible : https://www.ray.io/, [Dernier accès : 13-Mar-2022]

[94] Pip. [En ligne]. Disponible : https://pypi.org/project/pip/, [Dernier accès : 13-Mar-2022]

[95] Pyton. [En ligne]. Disponible : https://www.python.org/, [Dernier accès : 13-Mar-2022]

[96] Google Firebase. [En ligne]. Disponible : https://firebase.google.com/, [Dernier accès : 13-Mar-2022]

[97] Google Firebase, "Pyton API". [En ligne]. Disponible : https://firebase.google.com/docs/reference/admin/python, [Dernier accès : 13-Mar-2022]

[98] Django. [En ligne]. Disponible : https://www.djangoproject.com/, [Dernier accès : 13-Mar-2022]

[99] Xeokit. [En ligne]. Disponible : https://xeokit.io/, [Dernier accès : 13-Mar-2022]

[100] Android Studio. [En ligne]. Disponible : https://www.android.com/intl/en_ca/, [Dernier accès : 13-Mar-2022]

[101] Java, [En ligne]. Disponible : https://www.java.com/en/, [Dernier accès : 13-Mar-2022]

[102] Android Webview. [En ligne]. Disponible : https://developer.android.com/reference/android/webkit/WebView, [Dernier accès : 13-Mar-2022]

[103] Triage par merge en parallèle, [En ligne]. Disponible : https://gist.github.com/stephenmcd/39ded69946155930c347, [Dernier accès : 13-Mar-2022]

ANNEXE A ATELIER-NIST INFORMATION IMPORTANTE EN CAS D'URGENCE DANS UN BÂTIMENT

Le 3 mai 2004, le laboratoire *« National Institute of Standards and Technology » (NIST)* a organisé un atelier pour identifier les besoins en information des premiers intervenants (i.e., sapeurs-pompiers, police) en cas d'urgence dans un bâtiment (i.e., incendie, intervention policière, intervention médicale). Cet Annexe résume l'information importante dont ont besoin les sapeurs-pompiers tel que publié dans l'article « Workshop to Define Information Needed by Emergency Responders during Building Emergencies » [5].

a) Informations statiques

- Où se trouvent les entrées/sorties dans le bâtiment, emplacement des points d'accès tels que les portes, escaliers et ascenseurs?
- Quels sont les systèmes disponibles pour atténuer l'incident (par exemple, quelle est la meilleure colonne d'incendie)?
- Où se trouvent les dangers et les obstructions dans le bâtiment?
- Quels générateurs alimentent quels systèmes; qu'est-ce qui peut être éteint en toute sécurité?
- Où se trouvent les dangers à proximité, à l'extérieur du bâtiment, qui peuvent être affectés par l'incident, par exemple la propagation du feu d'un bâtiment à l'autre?
- Photos des côtés du bâtiment.
- Des dessins simplifiés doivent être utilisés au lieu des dessins complets du bâtiment.
- Ces dessins doivent inclure les services publics, les coffres-forts, les bornes fontaines, l'emplacement des portes, les limites des zones, les matières dangereuses, les murs coupe-feu, les accès au toit, les cages d'escalier. Les murs coupe-feu, l'accès au toit, les cages d'escalier, les ascenseurs, l'emplacement des caméras, les salles de contrôle de sécurité et d'incendie, et les sources d'eau.

b) Informations dynamiques

- Où se situe l'incident dans le bâtiment?
- Quelle est la taille des incendies et quelle est leur durée?
- Où se trouve la fumée dans le bâtiment?
- Des capteurs de fumée et de chaleur dans les cages d'escalier seraient utiles, en particulier pour l'évacuation.
- Les systèmes du bâtiment fonctionnent-ils pour atténuer l'incendie ou tout autre incident?

- Des personnes se trouvent-elles encore à l'intérieur et sont-elles en danger?
- Combien d'occupants sont handicapés ou à mobilité réduite et où sont-ils situés?
- Où se trouvent les produits chimiques toxiques qui pourraient être en cause?
- Quelle est la modification de l'intégrité structurelle du bâtiment pendant l'incident?
- Quelles sont les voies d'évacuation disponibles ou bloquées?
- Quelle est la meilleure cage d'escalier?
- Où se trouvent les lieux de triage?
- L'historique de la sécurité (contrôle d'accès) présente un intérêt particulier pour les forces de l'ordre.
- Quel est l'état des systèmes mécaniques (eau, électricité, ventilation)?
- Le système de gicleurs est-il activé?
- Les unités de traitement de l'air sont-elles arrêtées?
- L'alimentation électrique est-elle coupée?
- En ce qui concerne l'état des sous-systèmes du bâtiment, quels sont les systèmes qui fonctionnent encore (CVC, évacuation des fumées, ascenseurs, etc.)?
- Est-ce que tous les ascenseurs fonctionnent?
- Quels systèmes ont cessé d'émettre des rapports?
- Les relevés des capteurs sont-ils fiables ou l'état du système est-il incertain?
- Des diagnostics positifs et une détection des pannes sont nécessaires.
- Exposition des bâtiments voisins à toute menace tels que le feu, les matières dangereuses, le rayon d'explosion d'une bombe et profils chimiques/radiation des bâtiments.
- Conseils sur l'endroit où se placer (à l'extérieur pour les véhicules, à l'intérieur pour les civils).
- Prévision et état des conditions météorologiques sur le site à l'aide de capteurs placés à l'extérieur des bâtiments et/ou de stations météorologiques locales.
- Projection de l'endroit où l'eau ou la mousse s'écoulera.
- Type et durée de la formation requise pour utiliser les systèmes intelligents.

ANNEXE B RESUME DES MODELS EXISTANTS

Articles	Données utilisées	Analyses et services offerts	Présentation de l'information	Outils et techniques utilisés	Résultats et limitations
A BIM-based simulation framework for fire safety management and investigation of the critical factors affecting human evacuation performance [47]	Données BIM.	Système d'évacuation et de simulation d'incendie.	Aucune présentation. Les résultats consistent à la valeur «Required Safe Egress Time» RSET obtenue dans différentes situations d'évacuation.	Logiciel Pyrosim pour la simulation FDS, logiciel AnyLogic l'évacuation par «Agent-Based Modeling» (ABM), logiciel Autodesk Revit pour la création des données 3D.	La performance est mesurée en RSET. Modèle utilisé pour l'entraînement et la phase de construction de bâtiments et non pas en situation d'incendie.
A dynamic fire escape path planning method with BIM [2]	Données BIM, Données IoT (i.e., détecteur de fumée, capteur sans fil pour la position des occupants dans le bâtiment).	Chemin d'évacuation en temps réel (cas d'une personne et cas de plusieurs personnes), évitement des congestions dans le cas de plusieurs personnes et simulation d'incendie.	Aucune présentation. Le modèle fait ressortir le chemin meilleur d'évacuation.	Logiciel Pyrosim pour la simulation FDS, algorithme Dijkstra, logiciel Autodesk Revit pour la création des données 3D.	Meilleur temps d'évacuation que Chou et al [61]. La solution n'est pas vraiment en temps réel, car la simulation FDS ne marche pas en temps réel.

Articles	Données utilisées	Analyses et services offerts	Présentation de l'information	Outils et techniques utilisés	Résultats et limitations
A Framework for an Indoor Safety Management System Based on Digital Twin [48]	Données BIM, données IoT (i.e., température, humidité, fumée, oxygène, CO2, capteur magnétique de portail).	Surveillance de l'état de sécurité de la scène, classification du danger et évaluation du niveau, suggestions de traitement du danger.	Plateforme web, structure B/S (Brower/Server), 3D vue avec WebGL.	Machine learning SVM (Support Vector Machine) avec RBF(Radial Basis Function) et K-fold cross-validation.	Détection du feu (accuracy=100, precision=100, recall=100), détection surpopulation (accuracy=99.25, precision=97.07, recall=99.38), détection entrée illégale (accuracy=97.57, precision=98.15, recall=99.25).
Agent-based Simulation for IoT Facilitated Building Evacuation [3]	Données vidéos (i.e., CCTV camera), données IoT (i.e., détecteur de fumée, de température et de CO2).	Meilleur chemin d'évacuation en considérant l'aspect social des occupants.	Aucune présentation, la sortie consiste à la comparaison de performance entre le chemin le plus court et le chemin calculé par l'algorithme network flow.	Algorithme d'optimisation Network flow, «Agent Based Social Simulation» (ABSS), «Belief-Desire-Intention» (BDI), logiciel PedSim et Cplex solver, traitement d'image.	L'algorithme Network flow est plus performant que l'algorithme du plus court chemin. Il évite la congestion. L'attachement social ralentit l'évacuation. Les portes doubles internes adjacentes sont plus efficaces qu'une seule porte à double largeur. Test sur le bâtiment Alan turing en Italy.

Articles	Données utilisées	Analyses et services offerts	Présentation de l'information	Outils et techniques utilisés	Résultats et limitations
Graph-based network generation and CCTV processing techniques for fire evacuation [52]	Données BIM, données vidéos (i.e., CCTV caméra). Données IoT (température, CO2, CO).	Meilleur chemin d'évacuation.	Réalité augmentée sur téléphone intelligent.	Graph-Based Network combinant «Medial Axis Transform» (MAT) et «Visibility Graphs» (VG), Deep learning avec Faster R-CNN et ResNet-101.	MAT (nombre de nœuds=15, temps d'exécution=0.79ms, distance total=148), VG (nombre de nœuds=24, temps d'exécution=1.43ms, distance total=122), MAT+VG (nombre de nœuds=20, temps d'exécution=1.08ms, distance total=128).
An Innovative Smart System based on IoT Technologies for Fire and Danger Situations [55]	Données BIM, données IoT (i.e., capteur de température, humidité, fumée, CO et Capteur infrarouge passif) avec Bluetooth, Wi-Fi, Sigfox interface et module Z-Wave.	Envoi de notifications.	Application Web et application mobile.	Architecture client-serveur, Open Data interface, model view controller, security proxy, broker architecture, Firebase.	Le système proposé s'appelle SAFETY (Smart Aid System for Fire and dangEr situations).

Articles	Données utilisées	Analyses et services offerts	Présentation de l'information	Outils et techniques utilisés	Résultats et limitations
Integrating BIM and IoT Technologies in Innovative Fire Management Systems [18]	Données BIM, données IoT (i.e., capteur de température, humidité, fumée, CO, et Capteur infrarouge passif) avec Bluetooth, Wi-Fi, Sigfox interface et module Z-Wave.	Envoi de notifications.	Application Web et application mobile et vue 3D.	Architecture client-serveur, Open Data interface, model view controller, security proxy, broker architecture, Firebase.	Amélioration du modèle SAFETY avec une vue 3D.
A Multi-agent System for Recommending Fire Evacuation Routes in Buildings, Based on Context and IoT [59]	Données IoT (i.e., capteurs de feu).	Chemin d'évacuation en temps réel.	Téléphones intélligents des occupants du bâtiment.	Système multi-agent et système de recommandation.	Cet article est une proposition de recherche.
An IoT based Real-Time Stress Detection System for Fire-Fighters [51]	Capteurs sans fil dans les gants des sapeurs-pompiers (Galvanic Skin Response (GSR) sensor et heart rate sensor). Capteurs de température, humidité, CO et CO2.	Détection du stress chez les sapeurs-pompiers en temps réel.	Écran OLED.	ZigBee protocol, «Message Queuing Telemetry Transport» (MQTT) et Adafruit IO.	Travaux futurs : utilisation de l'apprentissage machine pour plus d'analyse des données.

Articles	Données utilisées	Analyses et services offerts	Présentation de l'information	Outils et techniques utilisés	Résultats et limitations
Development of BIM, IoT and AR/VR technologies for fire safety and upskilling[16]	Données BIM, données IoT (i.e., capteurs de chaleur, fumée, qualité de l'air.	Chemin d'évacuation, entraînement de sapeurs-pompiers.	Réalité augmentée (AR), Réalité virtuelle (VR).	Logiciel Unity Game Engine, logiciel Autodesk Revit pour créér le plan 3D, logiciel Windows Presentation Fondation et logiciel ThingSpeak.	La simulation immersive peut réduire la pression psychologique des sapeurs-pompiers et la distance de déplacement ainsi que d'améliorer l'efficacité de l'intervention. Les pompiers en formation peuvent également améliorer leur performance grâce à une formation répétée en RV.
Smart Detection of Fire Source in Tunnel Based on the Numerical Database and Artificial Intelligence [17]	Données IoT (i.e., capteur température).	Détection de la source du feu dans un tunnel (i.e., métro, voiture), détection de la taille du feu et des conditions de ventilation, simulation du feu.	Aucune présentation.	« Long-Short Term Memory Recurrent Neural Network » (LSTM-RNN) et simulateur FDS.	Une accuracy de 90%.

ANNEXE C CODE SOURCE DE L'APPLICATION DISTRIBUÉE

IoT_service.py

```
import ray
import time
import logging
import firebase_admin
from firebase_admin import credentials
from firebase_admin import firestore
import json
from random import randrange

# Start Ray.
raw_certificate = {
  "type": "service_account",
  "project_id": "firestation-9d01e",
  "private_key_id": "d51c70de9eb17474afc0e68779a3936321ee42d8",
  "private_key": "-----BEGIN PRIVATE KEY-----
\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQCrFnP0gJLpeHxV\n
yU07jyTCO9dVbofrd6BClo9XYSRObR2M89BBj/5WnbjQSamE68keSCC5XezBLUS1\n1e7Ra
c0muFTAFp91yFDzokYWuGZ2iRYXC/BULz09Zc1Rc1dMFggFqbkOZo3H8L9k\nd7UhLdqk
VX1cl9NfhNWt4A4Gfrg1FwtIb9KfuC/sAko1wuxKegysPfDKTA4g4/Gy\n2YA4OA6MBXVC
CFix/6OwOzMyRsKYLsuasP6LbgGDqIGHq0bOWRSkyeu71Jtr5yZ2\nrkkvz/odjpez5IpUEQO
KR04foMkcovLF1k6C/hzJi70mVpTTXHh6bHfSoGBXfecU\n4xpfKs7ZAgMBAAECggEAAv
C1JNl4bOKN56vr4JJ1tCoMg29z9SQbREZ+usA0vDpP\nPM0yrr9d3h9oysxkfIadPSCwaBWJD
GR0RVgik6vCg4aKDyEOALUYGw/Jx5MuUGqY\ngn3i7yqlEUlbs9yMSznBTb02E0OU7sygf
wZsCpPVdme+HbMoRhoSazuqBT30eXxF\nwtDU+KtJskwUz9N+CBH4V8TyDckldPNc3/1vC
MFfunp0z5jKGVETxbOAB9UqWn+V\no9alzGpmHNia4GI1ct3OtM6VXsCmh+zJU1rB7T6Sb
83q3DJkzv+OddS3InEVeV3p\ns409bii2P00w78DVIruCvSZp8OX4bOkLlCzGg/YOLwKBgQD
qEFPFGDK33IGxAOt2\nWO76wGXubePmx2QcqKB3ySg+qGdhU4oyruFOBTWvQ1xmtiY//
WHoMXYBWpFseCwa\n2KS/l9Raui7NjHNESbi2RHaRyFxZ2+ulejCWbW2q1o8Hg7GNFqUB
cjaUASEtkjz8\n7K1Wl8AnZ8jxNoNfAsqCdlvAVwKBgQC7HzMr5T5VicSrrwK5Y82SrHonm2
9lCZ/I\no/Y0DcdPLxrdNctiMqxwz83GouMLENjZwKfh2umhGKz0PfVyldedQiAZH7yPWaQx\
nyXoc1xkFDn7wMnfQJYFEonsuyOrc4j908Zkh7PLPN8fyZTdqrYJoWuv7HI8sGsHq\nUyhep
WCsTwKBgQDVOHxnPmztatVGHjVhQRCqolv4wGhs0dST+tT7aebcdWJ/ma96\nDi82eK538
yAYfPQJNNRcxbne79cr/akH57Tkx/XtlE+QddKHMbUvbNyqXuAqB5n5\ny365egP+Lahryv8S
LL+xuo6XcxP7RI2ERkJ9cI3eF8BIZEV2Zr996z4XzwKBgAVD\nHFpIqRcYt7ooQshZ4ScVyJ2
ihmBuA9tNmRffp1I3veaoo56QVD8b6OqEHCRshdyh\nyVbVpCpOmlaIA4YEZrraaDKsrM3P
NInd45w6BE8bGEDC+pGU7Gy0sQggyFIL3sL4\nJFlK440mQW40oHlUhc1PeUFnjTMtUXS4f
/MhbXWLAoGBAN7T1Es5a3SEadevlnNg\nryQE4l3N3AzmzvhjSEY57yXnGDsvaFJREi7b25
VdXCBSRUQpEEyYotRCxev6jCNb\n7gqc/pNBN1uAaAxmsmbUBNftLRJT4EgLQ2oXOjAa4
angw8/M2WjM5jccK6uXQ82r\nif8P/jpQj6YmjNwKil53oimk\n-----END PRIVATE KEY-----
\n",
```

```
  "client_email": "firebase-adminsdk-u2u21@firestation-9d01e.iam.gserviceaccount.com",
  "client_id": "114945489086992165715",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-
adminsdk-u2u21%40firestation-9d01e.iam.gserviceaccount.com"
}

@ray.remote #(num_cpus=2, num_gpus=1)
class IoTService(object):
    def __init__(self):
        pass

    def run(self):
        print("updating iot data")
        databaseURL     = 'https://firestation-9d01e-default-rtdb.firebaseio.com/'
        connection      = credentials.Certificate(raw_certificate)
        firebase_app    = firebase_admin.initialize_app(connection, {
            'databaseURL':databaseURL
        })

        db = firestore.client()
        doc_ref = db.collection(u'iot-data').document(u'simple-data')

        duration = 60 * 5
        for i in range(duration,0,-1):
            doc_ref.set({
            u'temperature': randrange(100),
            u'smoke-density': randrange(100),
            u'firefighters-inside': randrange(100)
            })

            time.sleep(1)




        return 0

# launch service
ray.init(address="auto")
iot_service = IoTService.remote()
ray.get(iot_service.run.remote())
```

Remote_control_service.py

```
import requests
import ray
from ray import serve
from fastapi import FastAPI

ray.init(address="auto", namespace="serve")
app = FastAPI()

@serve.deployment(name="remote_control", route_prefix="/remote")
@serve.ingress(app)
class RemoteControl:
    def __init__(self):
        pass

    @app.get("/")
    def get(self):
        return {"service":"Fire Station Remote Control"}

    @app.get("/valve/{valve_id}/{valve_status}")
    def controlValve(self, valve_id: int, valve_status: str):
        print ("valve_id : ", valve_id)
        print ("valve_status : ", valve_status)

        return {"valve_status" : valve_status}


# Deploy our class.
RemoteControl.deploy()
```

ANNEXE D CODE SOURCE DU SERVEUR WEB

firestation\fireemergency\templates\xeokit\index.html

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>Fire Station</title>

    <!-- Custom fonts for this template-->
    <link href="{% static 'fontawesome-free/css/all.min.css' %}" rel="stylesheet"
type="text/css"/>

    <link

href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,400i,600,600i,700,
700i,800,800i,900,900i"
        rel="stylesheet">

    <!-- Custom styles for this template-->
    <link href="{% static 'theme/css/sb-admin-2.min.css' %}" rel="stylesheet"/>

    <!-- Xeokit styles -->
    <link rel="stylesheet" href="{% static 'xeokit/lib/fontawesome-free-5.11.2-
web/css/all.min.css' %}" type="text/css"/>
    <link rel="stylesheet" href="{% static 'xeokit/dist/xeokit-bim-viewer.css' %}"
type="text/css"/>
    <link rel="stylesheet" href="{% static 'xeokit/css/style.css' %}"/>

</head>

<body id="page-top">

    <!-- Page Wrapper -->
    <div id="wrapper">
```

```
        <!-- Sidebar -->
        <ul class="navbar-nav bg-gradient-primary sidebar sidebar-dark accordion"
id="accordionSidebar">

            <!-- Sidebar - Brand -->
            <a class="sidebar-brand d-flex align-items-center justify-content-center"
href="index.html">
                <div class="sidebar-brand-icon" style="transform: scaleX(-1);">
                    <img src="{% static 'theme/img/icons8-fire-engine-48.png' %}" alt="...">
                </div>
                <div class="sidebar-brand-text mx-3">Fire Station</div>
            </a>

            <!-- Divider -->
            <hr class="sidebar-divider my-0">

            <!-- Nav Item - Dashboard -->
            <li class="nav-item active">
                <a class="nav-link" href="{% url 'fire:index' %}">
                    <i class="fas fa-fw fa-tachometer-alt"></i>
                    <span>Dashboard</span></a>
            </li>
            <li class="nav-item active">
                <a class="nav-link" href="{% url 'root:index' %}">
                    <i class="fas fa-fw fa-home"></i>
                    <span>Home</span></a>
            </li>

            <!-- Divider -->
            <hr class="sidebar-divider d-none d-md-block">

            <!-- Sidebar Toggler (Sidebar) -->
            <div class="text-center d-none d-md-inline">
                <button class="rounded-circle border-0" id="sidebarToggle"></button>
            </div>

        </ul>
        <!-- End of Sidebar -->

        <!-- Content Wrapper -->
        <div id="content-wrapper" class="d-flex flex-column">

            <!-- Main Content -->
            <div id="content">

                <!-- Topbar -->
```

```
            <nav class="navbar navbar-expand navbar-light bg-white topbar mb-4 static-top
shadow">

                <!-- Sidebar Toggle (Topbar) -->
                <button id="sidebarToggleTop" class="btn btn-link d-md-none rounded-circle mr-
3">
                    <i class="fa fa-bars"></i>
                </button>

                <!-- Topbar Search -->
                <form
                    class="d-none d-sm-inline-block form-inline mr-auto ml-md-3 my-2 my-md-0
mw-100 navbar-search">
                    <div class="input-group">
                        <input type="text" class="form-control bg-light border-0 small"
placeholder="Search for..."
                            aria-label="Search" aria-describedby="basic-addon2">
                        <div class="input-group-append">
                            <button class="btn btn-primary" type="button">
                                <i class="fas fa-search fa-sm"></i>
                            </button>
                        </div>
                    </div>
                </form>

                <!-- Topbar Navbar -->
                <ul class="navbar-nav ml-auto">

                    <!-- Nav Item - Search Dropdown (Visible Only XS) -->
                    <li id="search-button" class="nav-item dropdown no-arrow d-sm-none">
                        <a class="nav-link dropdown-toggle" href="#" id="searchDropdown"
role="button"
                            data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                            <i class="fas fa-search fa-fw"></i>
                        </a>
                        <!-- Dropdown - Messages -->
                        <div class="dropdown-menu dropdown-menu-right p-3 shadow animated--
grow-in"
                            aria-labelledby="searchDropdown">
                            <form class="form-inline mr-auto w-100 navbar-search">
                                <div class="input-group">
                                    <input type="text" class="form-control bg-light border-0 small"
                                        placeholder="Search for..." aria-label="Search"
                                        aria-describedby="basic-addon2">
                                    <div class="input-group-append">
                                        <button class="btn btn-primary" type="button">
```

```
                                <i class="fas fa-search fa-sm"></i>
                            </button>
                        </div>
                    </div>
                </form>
            </div>
        </li>

        <!-- Nav Item - Alerts -->
        <li class="nav-item dropdown no-arrow mx-1">
            <a class="nav-link dropdown-toggle" href="#" id="alertsDropdown"
role="button"
                data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                <i class="fas fa-bell fa-fw"></i>
                <!-- Counter - Alerts -->
                <span class="badge badge-danger badge-counter">3+</span>
            </a>
            <!-- Dropdown - Alerts -->
            <div class="dropdown-list dropdown-menu dropdown-menu-right shadow
animated--grow-in"
                aria-labelledby="alertsDropdown">
                <h6 class="dropdown-header">
                    Alerts Center
                </h6>
                <a class="dropdown-item d-flex align-items-center" href="#">
                    <div class="mr-3">
                        <div class="icon-circle bg-primary">
                            <i class="fas fa-file-alt text-white"></i>
                        </div>
                    </div>
                    <div>
                        <div class="small text-gray-500">March 12, 2022 - 10h30</div>
                        <span class="font-weight-bold">Fire on first floor !</span>
                    </div>
                </a>
                <a class="dropdown-item d-flex align-items-center" href="#">
                    <div class="mr-3">
                        <div class="icon-circle bg-success">
                            <i class="fas fa-donate text-white"></i>
                        </div>
                    </div>
                    <div>
                        <div class="small text-gray-500">March 12, 2022 - 10h35</div>
                        Fire reach second floor !!
                    </div>
                </a>
                <a class="dropdown-item d-flex align-items-center" href="#">
```

```
                        <div class="mr-3">
                            <div class="icon-circle bg-warning">
                                <i class="fas fa-exclamation-triangle text-white"></i>
                            </div>
                        </div>
                        <div>
                            <div class="small text-gray-500">March 12, 2022 - 10h40</div>
                            Fire reach third floor !!
                        </div>
                    </a>
                    <a class="dropdown-item text-center small text-gray-500" href="#">Show
All Alerts</a>
                </div>
            </li>

            <!-- Nav Item - Messages -->
            <li class="nav-item dropdown no-arrow mx-1">
                <a class="nav-link dropdown-toggle" href="#" id="messagesDropdown"
role="button"
                    data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                    <i class="fas fa-envelope fa-fw"></i>
                    <!-- Counter - Messages -->
                    <span class="badge badge-danger badge-counter">7</span>
                </a>
                <!-- Dropdown - Messages -->
                <div class="dropdown-list dropdown-menu dropdown-menu-right shadow
animated--grow-in"
                    aria-labelledby="messagesDropdown">
                    <h6 class="dropdown-header">
                        Message Center
                    </h6>
                    <a class="dropdown-item d-flex align-items-center" href="#">
                        <div class="dropdown-list-image mr-3">
                            <img class="rounded-circle" src="{% static
'theme/img/undraw_profile_1.svg' %}"
                                alt="...">
                            <div class="status-indicator bg-success"></div>
                        </div>
                        <div class="font-weight-bold">
                            <div class="text-truncate">Hi there! I am wondering if you can help me
with a
                                problem I've been having.</div>
                            <div class="small text-gray-500">Emily Fowler · 58m</div>
                        </div>
                    </a>
                    <a class="dropdown-item d-flex align-items-center" href="#">
```

```
                        <div class="dropdown-list-image mr-3">
                          <img class="rounded-circle" src="{% static
'theme/img/undraw_profile_2.svg' %}"
                            alt="...">
                          <div class="status-indicator"></div>
                        </div>
                        <div>
                          <div class="text-truncate">I have the photos that you ordered last
month, how
                            would you like them sent to you?</div>
                          <div class="small text-gray-500">Jae Chun · 1d</div>
                        </div>
                      </a>
                      <a class="dropdown-item d-flex align-items-center" href="#">
                        <div class="dropdown-list-image mr-3">
                          <img class="rounded-circle" src="{% static
'theme/img/undraw_profile_3.svg' %}"
                            alt="...">
                          <div class="status-indicator bg-warning"></div>
                        </div>
                        <div>
                          <div class="text-truncate">Last month's report looks great, I am very
happy with
                            the progress so far, keep up the good work!</div>
                          <div class="small text-gray-500">Morgan Alvarez · 2d</div>
                        </div>
                      </a>
                      <a class="dropdown-item d-flex align-items-center" href="#">
                        <div class="dropdown-list-image mr-3">
                          <img class="rounded-circle"
src="https://source.unsplash.com/Mv9hjnEUHR4/60x60"
                            alt="...">
                          <div class="status-indicator bg-success"></div>
                        </div>
                        <div>
                          <div class="text-truncate">Am I a good boy? The reason I ask is
because someone
                            told me that people say this to all dogs, even if they aren't
good...</div>
                          <div class="small text-gray-500">Chicken the Dog · 2w</div>
                        </div>
                      </a>
                      <a class="dropdown-item text-center small text-gray-500" href="#">Read
More Messages</a>
                    </div>
                  </li>
```

```
                    <div class="topbar-divider d-none d-sm-block"></div>

                    <!-- Nav Item - User Information -->
                    <li class="nav-item dropdown no-arrow">
                      <a class="nav-link dropdown-toggle" href="#" id="userDropdown"
role="button"
                        data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                        <span class="mr-2 d-none d-lg-inline text-gray-600 small">Landry
Sanou</span>
                        <img class="img-profile rounded-circle"
                          src="{% static 'theme/img/undraw_profile.svg' %}">
                      </a>
                      <!-- Dropdown - User Information -->
                      <div class="dropdown-menu dropdown-menu-right shadow animated--grow-
in"
                        aria-labelledby="userDropdown">
                        <a class="dropdown-item" href="#">
                          <i class="fas fa-user fa-sm fa-fw mr-2 text-gray-400"></i>
                          Profile
                        </a>
                        <a class="dropdown-item" href="#">
                          <i class="fas fa-cogs fa-sm fa-fw mr-2 text-gray-400"></i>
                          Settings
                        </a>
                        <a class="dropdown-item" href="#">
                          <i class="fas fa-list fa-sm fa-fw mr-2 text-gray-400"></i>
                          Activity Log
                        </a>
                        <div class="dropdown-divider"></div>
                        <a class="dropdown-item" href="#" data-toggle="modal" data-
target="#logoutModal">
                          <i class="fas fa-sign-out-alt fa-sm fa-fw mr-2 text-gray-400"></i>
                          Logout
                        </a>
                      </div>
                    </li>

                  </ul>

                </nav>
                <!-- End of Topbar -->

                <!-- Begin Page Content -->
                <div class="container-fluid">

                  <!-- Page Heading -->
```

```
            <div class="d-sm-flex align-items-center justify-content-between mb-4 dashboard-
header">
                <h1 class="h3 mb-0 text-gray-800">Dashboard</h1>
                <a href="#" class="d-none d-sm-inline-block btn btn-sm btn-primary shadow-
sm"><i
                        class="fas fa-download fa-sm text-white-50"></i> Generate Report</a>
            </div>

            <!-- Content Row -->

            <div class="row">

                <!-- Area Chart -->
                <div class="col-xl-8 col-lg-7">
                  <div class="card shadow mb-4">
                    <!-- Card Header - Dropdown -->
                    <div
                      class="card-header py-3 d-flex flex-row align-items-center justify-content-
between">
                      <h6 class="m-0 font-weight-bold text-primary">3D Viewport</h6>
                    </div>
                    <!-- Card Body -->
                    <div class="card-body bim-viewport">
                      <div class="bim-model">
                        <input type="checkbox" id="explorer_toggle"/>
                        <label id="explorer_toggle_label" for="explorer_toggle"
class="explorer_toggle_label xeokit-btn fas fa-2x fa-sitemap"></label>
                        <div class="row" id="myExplorer"></div>
                        <div class="row" id="myToolbar"></div>
                        <div class="" id="myViewer">
                          <canvas id="myCanvas"></canvas>
                          <canvas id="myNavCubeCanvas"></canvas>
                        </div>
                      </div>
                    </div>
                  </div>
                </div>

                <!-- Pie Chart -->
                <div class="col-xl-4 col-lg-5">
                  <div class="card shadow mb-4">
                    <!-- Card Header - Dropdown -->
                    <div
                      class="card-header py-3 d-flex flex-row align-items-center justify-content-
between">
                      <h6 class="m-0 font-weight-bold text-primary">Real Time Data</h6>
```

```
                    </div>
                    <!-- Card Body -->
                    <div class="card-body">
                      <div class="tab-pane fade show active" id="real-time-view"
role="tabpanel" aria-labelledby="real-time-view-tab">
                        <table class="table">
                          <thead class="thead-dark">
                            <tr>
                              <th scope="col">Data</th>
                              <th scope="col">Value</th>
                            </tr>
                          </thead>
                          <tbody>
                            <tr>
                              <td>Temperature</td>
                              <td id="temperature-data">0</td>
                            </tr>
                            <tr>
                              <td>Smoke Density</td>
                              <td id="smoke-density-data">0</td>
                            </tr>
                            <tr>
                              <td>Firefighters Inside</td>
                              <td id="firefighters-inside-data">0</td>
                            </tr>
                          </tbody>
                        </table>
                      </div>
                    </div>
                  </div>

                  <div id="remote-control" class="card shadow mb-4">
                    <!-- Card Header - Dropdown -->
                    <div
                      class="card-header py-3 d-flex flex-row align-items-center justify-content-
between">
                      <h6 class="m-0 font-weight-bold text-primary">Remote Control</h6>
                    </div>
                    <!-- Card Body -->
                    <div class="card-body">
                      <div class="card">
                       <div class="card-body">
                        <h5 class="card-title">Valve #1</h5>
                        <p class="card-text">Remote activate or desactivate valve</p>

                         <div class="text-right">
                          <label class="switch">
```

```
                                <input type="checkbox" id="remote-valve"> <span
class="slider"></span>
                                </label>
                            </div>

                        </div>
                    </div>
                        </div>
                    </div>
                </div>
            </div>
            <!-- /.container-fluid -->

        </div>
        <!-- End of Main Content -->

        <!-- Footer -->
        <footer class="sticky-footer bg-white">
            <div class="container my-auto">
                <div class="copyright text-center my-auto">
                    <span>Copyright &copy; Your Website 2021</span>
                </div>
            </div>
        </footer>
        <!-- End of Footer -->

    </div>
    <!-- End of Content Wrapper -->

</div>
<!-- End of Page Wrapper -->

<!-- Scroll to Top Button-->
<a class="scroll-to-top rounded" href="#page-top">
    <i class="fas fa-angle-up"></i>
</a>

<!-- Logout Modal-->
<div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabel"
    aria-hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabel">Ready to Leave?</h5>
                <button class="close" type="button" data-dismiss="modal" aria-label="Close">
```

```
                <span aria-hidden="true">×</span>
              </button>
            </div>
            <div class="modal-body">Select "Logout" below if you are ready to end your current session.</div>
            <div class="modal-footer">
              <button class="btn btn-secondary" type="button" data-dismiss="modal">Cancel</button>
              <a class="btn btn-primary" href="login.html">Logout</a>
            </div>
          </div>
        </div>
      </div>


    <!-- Modal -->
    <div class="modal fade" id="exampleModalCenter" tabindex="-1" role="dialog" aria-labelledby="exampleModalCenterTitle" aria-hidden="true">
      <div class="modal-dialog modal-dialog-centered" role="document">
        <div class="modal-content">
          <div class="modal-header">
            <h5 class="modal-title" id="exampleModalLongTitle">Remote Control Service</h5>
            <button type="button" class="close" data-dismiss="modal" aria-label="Close">
              <span aria-hidden="true">&amp;times;</span>
            </button>
          </div>
          <div class="modal-body">
            Firestation Remote Control Service is not available
          </div>
          <div class="modal-footer">
            <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
          </div>
        </div>
      </div>
    </div>

    <!-- Bootstrap core JavaScript-->
    <script src="{% static 'jquery/jquery.min.js' %}"></script>
    <script src="{% static 'bootstrap/js/bootstrap.bundle.min.js' %}"></script>

    <!-- Core plugin JavaScript-->
    <script src="{% static 'jquery-easing/jquery.easing.min.js' %}"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>

    <!-- Custom scripts for all pages-->
```

```
    <script src="{% static 'theme/js/sb-admin-2.min.js' %}"></script>

    <!-- Page level plugins -->
    <script src="{% static 'chart.js/Chart.min.js' %}"></script>

    <!-- Page level custom scripts -->
    <script src="{% static 'theme/js/demo/chart-area-demo.js' %}"></script>
    <script src="{% static 'theme/js/demo/chart-pie-demo.js' %}"></script>

    <!-- App tooltips libraries-->
<script src="{% static 'xeokit/lib/popper.js' %}"></script>
<script src="{% static 'xeokit/lib/tippy.js' %}"></script>

<script type="module">

    // Set up application

    import {Server, BIMViewer, LocaleService} from "{% static 'xeokit/dist/xeokit-bim-
viewer.es.js' %}";

    import {messages as localeMessages} from "{% static 'xeokit/locales/messages.js' %}";

    window.onload = function () {

        const requestParams = getRequestParams();

        // Localization
        const locale = requestParams.locale || "en";

        // Project to load into the viewer
        //const projectId = requestParams.projectId;
        const projectId = 'MAP';
        if (!projectId) {
            return;
        }

        // Open the explorer tab?
        const openExplorer = requestParams.openExplorer;
        setExplorerOpen(openExplorer === "true");

        const enableEditModels = (requestParams.enableEditModels === "true");

        // Server client will load data from the file systems
        const server = new Server({
            dataDir: "{% static 'data' %}"
        });
```

```
    // Create  BIMViewer that loads data via the Server
    const bimViewer = new BIMViewer(server, {
        localeService: new LocaleService({
            messages: localeMessages,
            locale: locale
        }),
        canvasElement: document.getElementById("myCanvas"), // WebGL canvas
        explorerElement: document.getElementById("myExplorer"), // Left panel
        toolbarElement: document.getElementById("myToolbar"), // Toolbar
        navCubeCanvasElement: document.getElementById("myNavCubeCanvas"),
        busyModelBackdropElement: document.getElementById("myViewer"),
        enableEditModels: enableEditModels
    });

    bimViewer.localeService.on("updated", () => {

        // Create tooltips on various HTML elements created by BIMViewer

        const changes = document.querySelectorAll('[data-xeokit-i18ntip]');

        changes.forEach((change) => {

            // Update text label
            if (change.dataset.xeokitI18n) {
                change.innerText = bimViewer.localeService.translate(change.dataset.xeokitI18n);
            }

            // Create text for tooltip
            if (change.dataset.xeokitI18ntip) {
                change.dataset.tippyContent =
bimViewer.localeService.translate(change.dataset.xeokitI18ntip);
            }

            // Create tooltip from text, or update existing tooltip
            if (change.dataset.tippyContent) {
                if (change._tippy) {
                    change._tippy.setContent(change.dataset.tippyContent);
                } else {
                    tippy(change, {
                        appendTo: "parent",
                        zIndex: 1000000,
                        allowHTML: true
                    });
                }
            }
        });
    });
```

```
    // Configure our viewer
    bimViewer.setConfigs({});

    // Log info on whatever objects we click with the BIMViewer's Query tool
    bimViewer.on("queryPicked", (event) => {
        bimViewer.getObjectInfo(event.projectId, event.modelId, event.objectId, (objectInfo) =>
{
            alert(JSON.stringify(objectInfo, null, "\t"));
        }, (errMsg) => {
            alert(errMsg);
        })
        console.log("queryPicked: " + JSON.stringify(event, null, "\t"));
    });

    bimViewer.on("addModel", (event) => { // "Add" selected in Models tab's context menu
        console.log("addModel: " + JSON.stringify(event, null, "\t"));
    });

    bimViewer.on("editModel", (event) => { // "Edit" selected in Models tab's context menu
        console.log("editModel: " + JSON.stringify(event, null, "\t"));
    });

    bimViewer.on("deleteModel", (event) => { // "Delete" selected in Models tab's context
menu
        console.log("deleteModel: " + JSON.stringify(event, null, "\t"));
    });

    //------------------------------------------------------------------------------------------------------
-
    // Process page request params, which set up initial viewer state
    //------------------------------------------------------------------------------------------------------
-

    // Viewer configurations
    const viewerConfigs = requestParams.configs;
    if (viewerConfigs) {
        const configNameVals = viewerConfigs.split(",");
        for (let i = 0, len = configNameVals.length; i < len; i++) {
            const configNameValStr = configNameVals[i];
            const configNameVal = configNameValStr.split(":");
            const configName = configNameVal[0];
            const configVal = configNameVal[1];
            bimViewer.setConfig(configName, configVal);
        }
    }
```

```
// Load a project
bimViewer.loadProject(projectId, () => {

    // The project may load one or models initially.

    // Withe request params, we can also specify:
    //  - models to load
    // - explorer tab to open


    const modelId = requestParams.modelId;
    if (modelId) {
        bimViewer.loadModel(modelId);
    }

    const tab = requestParams.tab;
    if (tab) {
        bimViewer.openTab(tab);
    }

    //
    window.setInterval((function () {
        var lastHash = "";
        return function () {
            const currentHash = window.location.hash;
            if (currentHash !== lastHash) {
                parseHashParams();
                lastHash = currentHash;
            }
        };
    })(), 200);
},
(errorMsg) => {
    console.error(errorMsg);
});

function getRequestParams() {
    var vars = {};
    window.location.href.replace(/[?&]+([^=&]+)=([^&]*)/gi, (m, key, value) => {
        vars[key] = value;
    });
    return vars;
}

function parseHashParams() {
    const params = getHashParams();
```

```
        const actionsStr = params.actions;
        if (!actionsStr) {
            return;
        }
        const actions = actionsStr.split(",");
        if (actions.length === 0) {
            return;
        }
        for (var i = 0, len = actions.length; i < len; i++) {
            const action = actions[i];
            switch (action) {
                case "focusObject":
                    const objectId = params.objectId;
                    if (!objectId) {
                        console.error("Param expected for `focusObject` action: 'objectId'");
                        break;
                    }
                    bimViewer.setAllObjectsSelected(false);
                    bimViewer.setObjectsSelected([objectId], true);
                    bimViewer.flyToObject(objectId, () => {
                        // FIXME: Showing objects in tabs involves scrolling the HTML within the tabs - disable until we know how to scroll the correct DOM element. Otherwise, that function works OK

                        // bimViewer.showObjectInObjectsTab(objectId);
                        // bimViewer.showObjectInClassesTab(objectId);
                        // bimViewer.showObjectInStoreysTab(objectId);
                    });
                    break;
                case "focusObjects":
                    const objectIds = params.objectIds;
                    if (!objectIds) {
                        console.error("Param expected for `focusObjects` action: 'objectIds'");
                        break;
                    }
                    const objectIdArray = objectIds.split(",");
                    bimViewer.setAllObjectsSelected(false);
                    bimViewer.setObjectsSelected(objectIdArray, true);
                    bimViewer.viewFitObjects(objectIdArray, () => {
                    });
                    break;
                case "clearFocusObjects":
                    bimViewer.setAllObjectsSelected(false);
                    bimViewer.viewFitAll();
                    // TODO: view fit nothing?
                    break;
                case "openTab":
```

```
                const tabId = params.tabId;
                if (!tabId) {
                    console.error("Param expected for `openTab` action: 'tabId'");
                    break;
                }
                bimViewer.openTab(tabId);
                break;
            default:
                console.error("Action not supported: '" + action + "'");
                break;
        }
    }
}

function getHashParams() {
    const hashParams = {};
    let e;
    const a = /\+/g;  // Regex for replacing addition symbol with a space
    const r = /([^&;=]+)=?([^&;]*)/g;
    const d = function (s) {
        return decodeURIComponent(s.replace(a, " "));
    };
    const q = window.location.hash.substring(1);
    while (e = r.exec(q)) {
        hashParams[d(e[1])] = d(e[2]);
    }
    return hashParams;
}

function setExplorerOpen(explorerOpen) {
    const toggle = document.getElementById("explorer_toggle");
    if (toggle) {
        toggle.checked = explorerOpen;
    }
}

window.bimViewer = bimViewer; // For debugging
};
</script>


<script type="module">
 import { initializeApp } from "https://www.gstatic.com/firebasejs/9.0.1/firebase-app.js";
 import { getAnalytics } from "https://www.gstatic.com/firebasejs/9.0.1/firebase-analytics.js";
 import { collection, getDocs, addDoc} from "https://www.gstatic.com/firebasejs/9.0.1/firebase-
firestore-lite.js";
```

```
 import { getFirestore, doc, onSnapshot} from
"https://www.gstatic.com/firebasejs/9.0.1/firebase-firestore.js";
 import { getDatabase, ref, set } from "https://www.gstatic.com/firebasejs/9.0.1/firebase-
database.js";

 const firebaseConfig = {
   apiKey: "AIzaSyD1u2pN7DpyyvkzGBYnzd6joJpE-R0v-rE",
   authDomain: "firestation-9d01e.firebaseapp.com",
   projectId: "firestation-9d01e",
   storageBucket: "firestation-9d01e.appspot.com",
   messagingSenderId: "381839794091",
   appId: "1:381839794091:web:7e9156c9c85c90c17b7cee",
   measurementId: "G-1PKVXBMNDQ"
 };
 // Initialize Firebase
 const app = initializeApp(firebaseConfig);
 const analytics = getAnalytics(app);
 const db = getFirestore(app);

/*try {
 const docRef = await addDoc(collection(db, "users"), {
  first: "Ada",
  last: "Lovelace",
  born: 1815
 });
 console.log("Document written with ID: ", docRef.id);
} catch (e) {
 console.error("Error adding document: ", e);
}

const querySnapshot = await getDocs(collection(db, "users"));
querySnapshot.forEach((doc) => {
 console.log(`${doc.id} => ${doc.data()}`);
});*/

function updateData(iotData)
{
   $('#temperature-data').html(iotData['temperature']);
   $('#smoke-density-data').html(iotData['smoke-density']);
   $('#firefighters-inside-data').html(iotData['firefighters-inside']);
}

const unsub = onSnapshot(doc(db, "iot-data", "simple-data"), (doc) => {
   console.log("Current data: ", doc.data());
   updateData(doc.data());
});
```

```
$("#remote-valve").change(function () {
   var value = $(this).prop("checked") == true;

   $.ajax({
      url: "{% url 'fire:remote-valve' %}",
      data: {
        'valve_status': value,
        'valve_id' : 1
      },
      dataType: 'json',
      success: function (data) {

         console.log(data);

         $("#remote-valve").prop("checked", data.valve_status == "true");

         //console.log($("#remote-control").find(".card" ));

         if(data.valve_status == "true")
         {

           $("#remote-control").find( ".card" ).addClass("remote-active");
         }
         else
         {
            $("#remote-control").find(".card" ).removeClass("remote-active");
         }

         if (data.service == "down")
         {
            $('#exampleModalCenter').modal('show');
         }
         else if(data.service == "failed")
         {
            $('#exampleModalCenter').modal('show');
         }
         else
         {
           //
         }
      }
    });
});

</script>

</body>
```

```
</html>
```

ANNEXE VE CODE SOURCE DE L'APPLICATION MOBILE FIREALERT

MainActivity.java

```
package com.example.firestation;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.Menu;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.navigation.NavigationView;

import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.nav_view);
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
```

```
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_fire, R.id.nav_setting)
                .setDrawerLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
```

WebViewController.java

```
package com.example.firestation;

import android.webkit.WebResourceRequest;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class WebViewController extends WebViewClient {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        view.loadUrl(url);
        return true;
    }
}
```

FireFragment.java

```
package com.example.firestation.ui.fire;

import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.webkit.WebSettings;
import android.webkit.WebView;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;

import com.example.firestation.R;
import com.example.firestation.WebViewController;

public class FireFragment extends Fragment {

    private FireViewModel galleryViewModel;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        galleryViewModel =
                new ViewModelProvider(this).get(FireViewModel.class);
        View root = inflater.inflate(R.layout.fragment_gallery, container, false);


        //get web server data
        SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
        String ipAdress = sharedPref.getString("web_ip_address", "192.168.2.68");
        String port     = sharedPref.getString("web_port", "9090");
        String url = ipAdress + ":" + port + "/fire" ;

        WebView webView=root.findViewById(R.id.web_view_practice);
        webView.loadUrl(url);
        webView.setWebViewClient(new WebViewController());
        WebSettings webSettings = webView.getSettings();
        webSettings.setJavaScriptEnabled(true);
        return root;
    }
}
```

FireViewModel.java

```
package com.example.firestation.ui.fire;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

public class FireViewModel extends ViewModel {

    private MutableLiveData<String> mText;

    public FireViewModel() {
        mText = new MutableLiveData<>();
        mText.setValue("This is gallery fragment");
    }

    public LiveData<String> getText() {
        return mText;
    }
}
```

HomeFragment.java

```
package com.example.firestation.ui.home;

import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
```

```
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProvider;

import com.example.firestation.R;
import com.example.firestation.WebViewController;

public class HomeFragment extends Fragment {

    private HomeViewModel homeViewModel;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        homeViewModel =
                new ViewModelProvider(this).get(HomeViewModel.class);
        View root = inflater.inflate(R.layout.fragment_home, container, false);

        //get web server data
        SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
        String ipAdress = sharedPref.getString("web_ip_address", "192.168.2.68");
        String port     = sharedPref.getString("web_port", "9090");
        String url = ipAdress + ":" + port;

        WebView webView=root.findViewById(R.id.web_view_home);
        webView.loadUrl(url);
        webView.setWebViewClient(new WebViewController());
        WebSettings webSettings = webView.getSettings();
        webSettings.setJavaScriptEnabled(true);
        return root;
    }
}
```

HomeViewModel.java

```
package com.example.firestation.ui.home;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

public class HomeViewModel extends ViewModel {
    private MutableLiveData<String> mText;

    public HomeViewModel() {
        mText = new MutableLiveData<>();
        mText.setValue("This is home fragment");
```

```
    }

    public LiveData<String> getText() {
        return mText;
    }
}
```

SettingFragment.java

```
package com.example.firestation.ui.setting;

import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.widget.Button;
import android.widget.EditText;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;

import com.example.firestation.R;
import com.example.firestation.WebViewController;

public class SettingFragment extends Fragment {

    private SettingViewModel settingViewModel;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        settingViewModel =
                new ViewModelProvider(this).get(SettingViewModel.class);
        View root = inflater.inflate(R.layout.fragment_setting, container, false);


        Button button = (Button) root.findViewById(R.id.update_setting);
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                String ipAdress = ((EditText)root.findViewById(R.id.ip_address)).getText().toString();
                String port     = ((EditText)root.findViewById(R.id.port)).getText().toString();
```

```
            SharedPreferences sharedPref =
getActivity().getPreferences(Context.MODE_PRIVATE);
            SharedPreferences.Editor editor = sharedPref.edit();
            editor.putString("web_ip_address", ipAdress);
            editor.putString("web_port", port);
            editor.apply();
        }
    });

    return root;
  }
}
```

SettingViewModel.java

```
package com.example.firestation.ui.setting;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

public class SettingViewModel extends ViewModel {

  private MutableLiveData<String> mText;

  public SettingViewModel() {
    mText = new MutableLiveData<>();
    mText.setValue("This is slideshow fragment");
  }

  public LiveData<String> getText() {
    return mText;
  }
}
```

Mobile_Navigation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/mobile_navigation"
    app:startDestination="@+id/nav_home">
```

```
    <fragment
        android:id="@+id/nav_home"
        android:name="com.example.firestation.ui.home.HomeFragment"
        android:label="@string/menu_home"
        tools:layout="@layout/fragment_home" />

    <fragment
        android:id="@+id/nav_fire"
        android:name="com.example.firestation.ui.fire.FireFragment"
        android:label="Fire"
        tools:layout="@layout/fragment_gallery" />

    <fragment
        android:id="@+id/nav_setting"
        android:name="com.example.firestation.ui.setting.SettingFragment"
        android:label="Setting"
        tools:layout="@layout/fragment_setting" />
</navigation>
```

ANNEXE F ALGORITHME DE TRIAGE PAR MERGE

```
import ray
import sys
import time
import ray.util.multiprocessing as mp
import math
import random

@ray.remote (num_cpus=3)
class SortDataService(object):
   def __init__(self):
      pass

   def merge(self, *args):
      # Support explicit left/right args, as well as a two-item
      # tuple which works more cleanly with multiprocessing.
      left, right = args[0] if len(args) == 1 else args
      left_length, right_length = len(left), len(right)
      left_index, right_index = 0, 0
      merged = []
      while left_index < left_length and right_index < right_length:
         if left[left_index] <= right[right_index]:
            merged.append(left[left_index])
            left_index += 1
         else:
            merged.append(right[right_index])
            right_index += 1
```

```
    if left_index == left_length:
        merged.extend(right[right_index:])
    else:
        merged.extend(left[left_index:])
    return merged


def merge_sort(self, data):
    length = len(data)
    if length <= 1:
        return data
    middle = int(length / 2)
    left = self.merge_sort(data[:middle])
    right = self.merge_sort(data[middle:])
    return self.merge(left, right)


def merge_sort_parallel(self, data):
    # Creates a pool of worker processes, one per CPU core.
    # We then split the initial data into partitions, sized
    # equally per worker, and perform a regular merge sort
    # across each partition.
    processes = 2 #multiprocessing.cpu_count()
    pool = mp.Pool(processes=processes)
    size = int(math.ceil(float(len(data)) / processes))
    data = [data[i * size:(i + 1) * size] for i in range(processes)]
    data = pool.map(self.merge_sort, data)
    # Each partition is now sorted - we now just merge pairs of these
    # together using the worker pool, until the partitions are reduced
    # down to a single sorted result.
    while len(data) > 1:
        # If the number of partitions remaining is odd, we pop off the
        # last one and append it back after one iteration of this loop,
        # since we're only interested in pairs of partitions to merge.
        extra = data.pop() if len(data) % 2 == 1 else None
        data = [(data[i], data[i + 1]) for i in range(0, len(data), 2)]
        data = pool.map(self.merge, data) + ([extra] if extra else [])
    return data[0]

def run(self):
    print("sorting data")

    f = open("result.txt", "a")

    for i in range(1, 11):

        size = 100000*i
```

```
        data_unsorted = [random.randint(0, size) for _ in range(size)]

        for _ in range(3):
            start_time = time.time_ns()
            self.merge_sort_parallel(data_unsorted)
            end_time = time.time_ns()
            result = float((end_time - start_time))/1000000000
            f.write(f"{result}\n")

        f.write('--\n')

    f.close()
    return 0

# launch service
ray.init(address="auto")
service = SortDataService.remote()
result = service.run.remote();

time.sleep(6)
ray.get(result)
```