Année universitaire

2019 - 2020

Module 4-7-PRJ-SC

Projet transversal

Sujet

Oscar CARRILLO

Grégory MOREL

Françoise PERRIN

Préambule

Compétences visées

À l'issue de ce module vous serez capable de :

- Analyser et synthétiser des spécifications / un cahier des charges.
- Développer des solutions logicielles/matérielles en prenant en compte des contraintes de déploiement en réseaux et en respectant les bonnes pratiques de conception et de programmation orientée objet.
- Concevoir un protocole de transmission des données issues des objets communicants.

Moyens

Il s'agit de mettre en œuvre et d'intégrer de multiples connaissances/compétences à travers un projet réalisé en équipe de 4 élèves, munis de compétences en développement logiciel et en réseau :

- Analyse d'un cahier des charges, des interactions entre les différents éléments et découpage du projet.
- Conception et réalisation d'une architecture réseau.
- Conception d'une chaîne IoT et définition du protocole d'échange des informations.
- Conception de l'architecture logicielle (Design Pattern utilisés (si utilisés…), Diagrammes de classes, Schéma de la BD) et développement de la partie applicative en Java.
- Réalisation d'une infrastructure web qui permettra de récupérer, transférer et présenter de l'information.
- Analyse, traitement et visualisation de grands volumes de données (Big Data).

Projet

Objectif

Ce projet consiste à simuler des incendies sur une ville, et leur prise en charge par les flottes d'urgence qui vont intervenir.

La figure 1 ci-dessous représente le fonctionnement global de l'application.

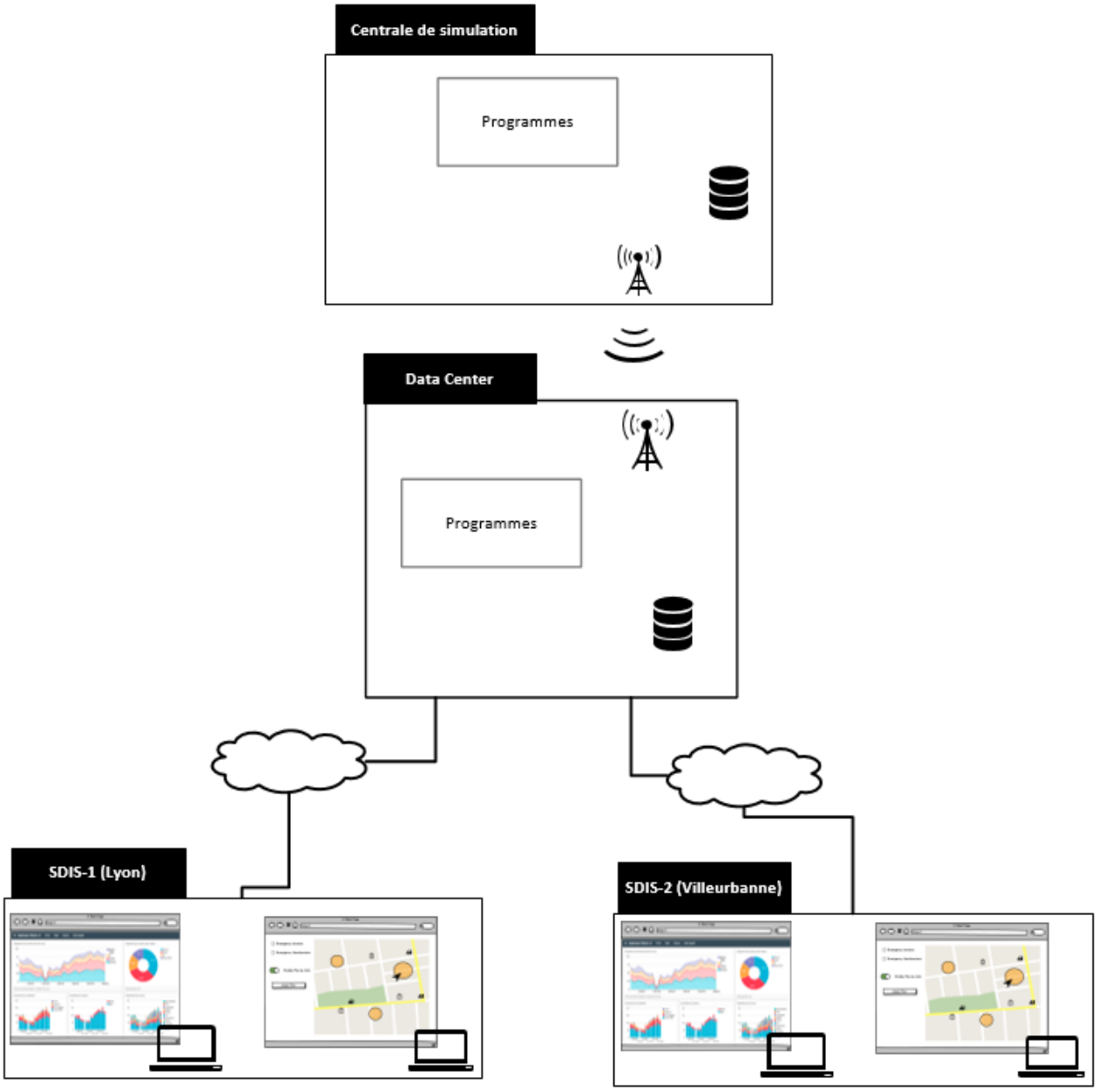

Figure 1 – Fonctionnement global de l'application

La centrale de simulation – début action

Son rôle est de générer des feux dont les coordonnées, l'intensité et la fréquence sont à définir dans le programme. Ces feux sont stockés dans une BD de simulation.

Les coordonnées et intensité de ces feux sont envoyées par ondes radio au Data Center, à l'image de détecteurs de feux qui transmettent une alerte à une centrale de supervision.

Le Data Center – milieu action

Le Data Center est la partie « métier » des services de gestion d'incendie. Il est constitué d'un ensemble de programmes allant de la réception des alertes incendies à leur prise en charge. La première action de ce groupe de programmes est donc l'acquisition des données des feux simulés et envoyées par radio (étape précédente), pour alimenter et mettre à jour une BD de feux « réels ». Disposer des deux BD « feux simulés » et « feux réels » permet de réaliser un couplage faible entre la centrale de simulation et le data center.

Ensuite, la logique métier : un programme appelé *Emergency Manager* gère une flotte de services d'urgence (casernes, camions, pompiers), détecte les feux (dans la BD) et déploie les ressources nécessaires pour les éteindre (quel(s) camion(s) de quelle(s) caserne(s) est (sont) affectés à quel(s) feu(x)). Il pourra libérer les ressources affectées à un feu lorsqu'il est éteint.

Enfin, un certain nombre de données doivent être envoyées dans un serveur de stockage sur le Cloud à des fins de statistiques et visualisations.

Les écrans de supervision des casernes

Grâce à des écrans de supervision, les casernes peuvent visualiser dans un navigateur web les feux en cours dans la ville (ou toute autre zone géographique choisie), ainsi que la position des différents véhicules de secours. Différents contrôles permettent d'afficher / masquer les feux, les véhicules de secours, etc.

Un Dashboard affiche des statistiques en fonction de l'historique des données collectées.

La centrale de simulation – fin action

Le simulateur détecte la présence ou non des services d'urgence près des feux (dans BD de feux réels) et réduit ou augmente leur intensité (dans la BD de feux simulés).

Architecture fonctionnelle de la solution

L'architecture fonctionnelle globale ainsi que les technologies à utiliser pour mettre en œuvre ce projet sont imposées comme le montre la figure 2.

Vous êtes en revanche très libre de concevoir chaque partie comme vous l'entendez. Il existe particulièrement plusieurs manières de gérer les interactions entre le Serveur Web (TLI), la DataBase, le Simulator et l'EmergencyManager.

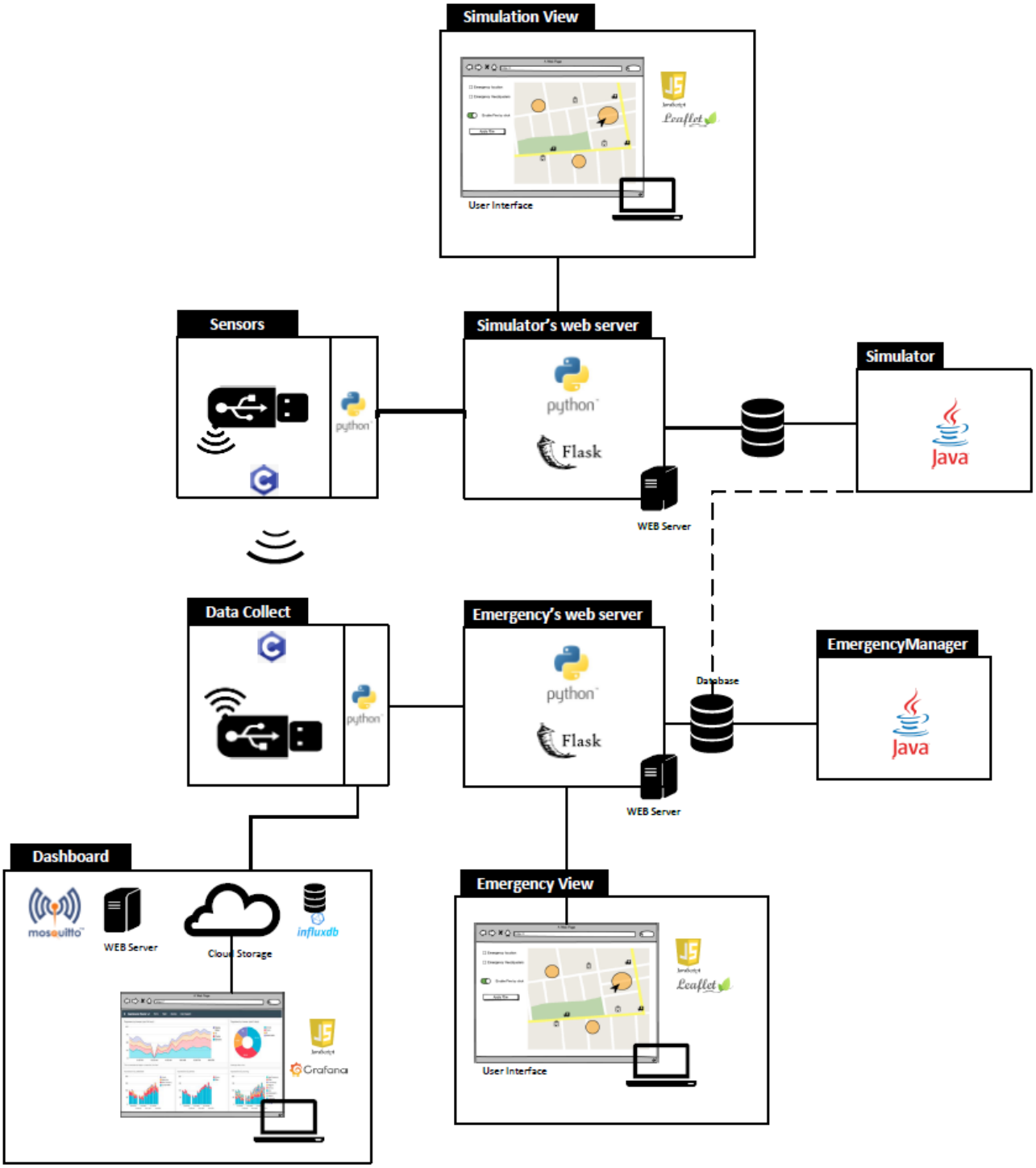

Figure 2 – Architecture fonctionnelle de l'application

Quelques bonnes pratiques de développement à mettre en œuvre :

- Chaque fonctionnalité doit pouvoir être **testée séparément**. Prévoyez des jeux de tests en conséquence.
- Les responsabilités doivent être bien séparées : accès BD, objets métier, rendu visuel, etc.
- Concevez et développez de manière **incrémentale** :
    - Gérez dans un premier temps un seul feu, une seule caserne dotée d'un seul camion, camion qui se « téléporte » instantanément vers le feu en 1 tour de jeu.
    - Ensuite, ajoutez des feux, des camions, des casernes avec affectation des ressources au feux dans l'ordre d'arrivée.
    - Prenez ensuite en compte le déplacement des camions ; pour simplifier, considérez dans un premier temps que les camions se déplacent en ligne droite entre leur position initiale et le feu.
    - En fonction du temps disponible, ajoutez d'autres règles « métier » à la phase d'affectation des ressources : différents types de camions, choix du ou des camion(s) à envoyer sur un sinistre (le plus près, le premier disponible, le mieux adapté, etc.). Vous pourrez également utiliser des API de calcul d'itinéraires disponibles sur le web afin de déplacer vos camions le long des routes réelles.

  Pour cela, respectez le principe « Ouvert aux extensions - Fermé aux modifications » en pensant abstraction, encapsulation, polymorphisme et délégation.

Architecture réseau de la solution

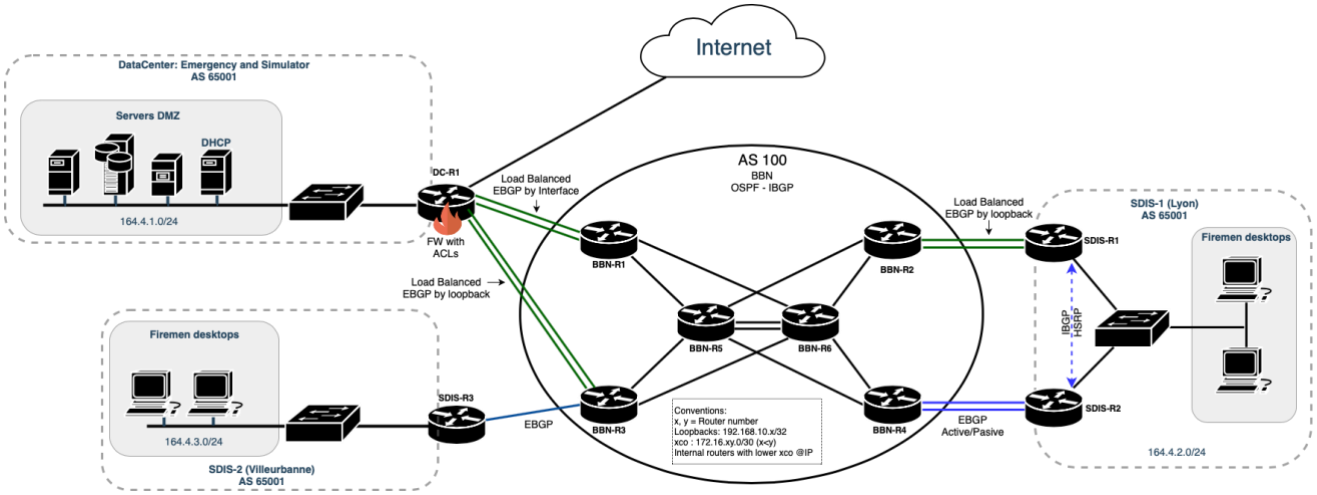

Figure 3. Architecture Réseau

L'architecture réseau de la solution est présentée dans la Figure 3 ; dans cette architecture il y aura 3 sites principaux :

- Un Data Center (AS 65001)
- SDIS de Lyon qui compte un central de supervision (AS 65001)
- SDIS de Villeurbanne (AS 65001)

Tous les sites seront interconnectés grâce à un BackBone BGP (AS 100)

Cette configuration sera la base pour pouvoir généraliser le déploiement à d'autres casernes, n'hésitez pas à fournir des scripts de génération des configurations pour les routeurs des nouvelles casernes en utilisant des moteurs de templates Jinja.

Pour la modélisation de votre architecture, vous allez utiliser le logiciel GNS 3.0 et les ordinateurs pour les validations d'interconnections devront être virtualisés par des machines virtuelles créées par vous sous VirtualBox. A minima, il faut créer les VMs suivantes :

- Serveur simulation
- Serveur Bases de Données PostgreSQL
- Serveur Emergency Management
- Serveurs Web
- Client SDIS Lyon
- Client SDIS Villeurbanne

Rem : les différents serveurs du DataCenter peuvent être dans la même VM.

Architecture IOT de la solution

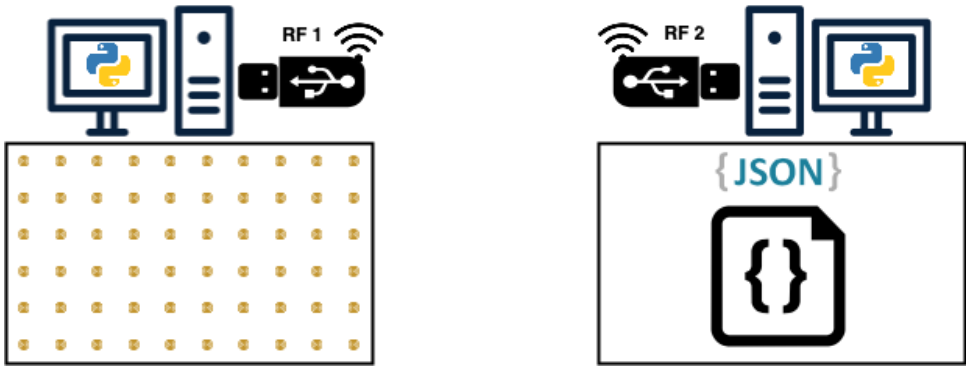

Figure 4. Schéma IoT

L'architecture à implémenter pour votre solution IoT sera à faire à l'aide des puces RF Sub 1GHz de Techno Innov comme dans la figure 4 et en respectant les chemins de génération et acquisition des données suivantes :

- les données à traiter seront issues d'une matrice de 10 points horizontaux par 6 points verticaux représentant le déploiement des capteurs d'intensité de feu à différents endroits dans la ville. L'intensité des feux sera une valeur entière entre [0,9].
- des données de tests de simulation pourront être fabriquées à partir de l'outil de simulation graphique disponible dans le git du projet.

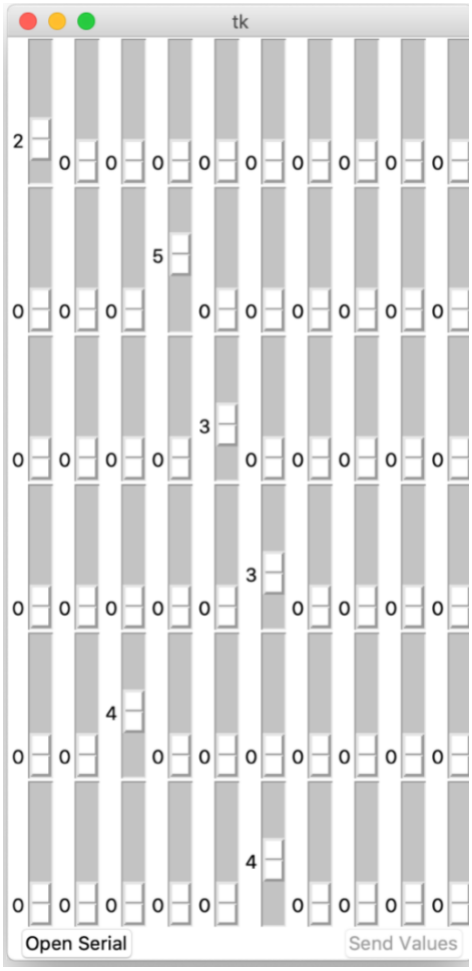

Figure 5. Application génération données capteurs

- L'outil de "génération de données capteur" va écrire dans le port USB les valeurs des différents points en suivant le format (`colonne, ligne, intensité`) pour chaque point de la grille. Par exemple, la sortie USB série des données générées dans la Figure 5 sera la suivante :

(0,0,2)	(5,1,0)	(0,3,0)	(5,4,0)
(1,0,0)	(6,1,0)	(1,3,0)	(6,4,0)
(2,0,0)	(7,1,0)	(2,3,0)	(7,4,0)
(3,0,0)	(8,1,0)	(3,3,0)	(8,4,0)
(4,0,0)	(9,1,0)	(4,3,0)	(9,4,0)
(5,0,0)	(0,2,0)	(5,3,3)	(0,5,0)
(6,0,0)	(1,2,0)	(6,3,0)	(1,5,0)
(7,0,0)	(2,2,0)	(7,3,0)	(2,5,0)
(8,0,0)	(3,2,0)	(8,3,0)	(3,5,0)
(9,0,0)	(4,2,3)	(9,3,0)	(4,5,0)
(0,1,0)	(5,2,0)	(0,4,0)	(5,5,4)
(1,1,0)	(6,2,0)	(1,4,0)	(6,5,0)
(2,1,0)	(7,2,0)	(2,4,4)	(7,5,0)
(3,1,5)	(8,2,0)	(3,4,0)	(8,5,0)
(4,1,0)	(9,2,0)	(4,4,0)	(9,5,0)

- Un microcontrôleur RF Sub 1Ghz (RF1) sera connecté au port USB du simulateur pour lire les sorties dans le format spécifié au point précédent, un protocole sécurisé devra être créé pour transmettre ces valeurs à un autre microcontrôleur par voie sans fils.
- Un microcontrôleur RF Sub 1GHz (RF2) sera connecté à un autre port USB et devra collecter par un protocole sans fils sécurisé les informations des intensités des feux. Ces valeurs devront être écrites par la sortie UART dans un format de votre choix.
- Les valeurs écrites par RF2 dans UART, seront collectées par une application qui va faire une double fonction :
    - Faire appel à un API REST DAO (format JSON) pour enregistrer dans la base de données Emergency Management les valeurs des intensités des feux reçus.
    - Envoyer par messages MQTT les valeurs des intensités des feux à un dashboard dans le cloud.

Nous mettons donc à votre disposition un petit programme de "génération de données capteur" (sorties potentielles de la carte) pour tester votre chaine IoT indépendamment du programme réel de simulation. Il est disponible sur le git :

- https://github.com/CPELyon/4irc-19-20-proj-transversal

Pour intégrer l'ensemble des couches, il suffira alors de modifier ce programme pour qu'il interroge la BD et écrive les données sur le RF1.

Annexes

Déroulement du projet

1	29/11 AM	Définition de l'architecture globale et découpage du projet en tâches et affectation aux membres de l'équipe (Trello ou autre). Prise en main des différentes technos à utiliser. Conception et mise en œuvre de l'architecture réseau.
2	02/12 PM	
3	05/12 AM	
4	05/12 PM	
5	6/12 AM	Présentation/Démo architecture réseau. Présentation architecture globale et gestion du projet
6	16/12 PM	Conception logicielle de l'application. Mise en œuvre de la chaîne IoT. Début développement.
7	18/12 AM	
8	18/12 PM	
9	19/12 PM	
10	20/12 AM	Présentation/Démo Chaîne IOT. Présentation Conception logicielle : Diag. de classes/séquences et Schéma BD.
11	06/01 PM	Développement logiciel (métier, rendu Web, interactions avec BD).
12	07/01 AM	
13	07/01 PM	
14	08/01 AM	
15	08/01 PM	
16	09/01 PM	
17	10/01 AM	
18	10/01 PM	Démo applicatif métier.
19	13/01 AM	Intégration de l'ensemble des composants. Développement partie analyse/visualisation Big data.
20	13/01 PM	
21	14/01 AM	Soutenance + Démo finales.
22	14/01 PM	

Évaluation

- ✓ Infrastructure Réseau : Démo maquette GNS 3 connectivité sites de casernes et data center en respectant des contraints d'accès depuis des machines virtuelles.
- ✓ Infrastructure IoT : Démo collecte, envoi et réception des données des feux dans la ville ainsi que génération d'appel REST.
- ✓ Analyse fonctionnelle de l'application, découpage en tâches et affectations.
- ✓ Diagrammes de classes et schéma de la BD.
- ✓ Soutenance et démonstration finale.

Documentation à consulter

Java	Javadoc • https://docs.oracle.com/javase/8/docs/api/
	Lire/écrire dans BD en Java (Java DataBase Connectivity) • https://openclassrooms.com/fr/courses/26832-apprenez-a-programmer-en-java/26258-jdbc-decouvrez-la-porte-dacces-aux-bases-de-donnees • https://www.tutorialspoint.com/postgresql/postgresql_java.htm
	Invoquer une requête http en Java • https://www.baeldung.com/java-http-request
	Librairie et tuto pour convertir objet <-> JSON et réciproquement • https://github.com/FasterXML/jackson-databind/
	Planification de tâches • https://docs.oracle.com/javase/7/docs/api/java/util/Timer.html • https://www.jmdoudoux.fr/java/dej/chap-planification_taches.htm
Partie web	Python Flask • https://openclassrooms.com/fr/courses/1654786-creez-vos-applications-web-avec-flask • https://www.tutorialspoint.com/flask/index.htm • https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world
	Système d'information géographique • https://leafletjs.com/ • https://leafletjs.com/plugins.html • https://docs.mapbox.com/api/
	Test d'API • Postman : https://www.getpostman.com/
Partie réseau	Configuration routeurs • https://cisco.netacad.com • https://blogs.cisco.com/developer/network-configuration-template
Partie IoT	Micro-contrôleur https://github.com/CPELyon/modules-techno-innov
	Accès REST API https://www.geeksforgeeks.org/get-post-requests-using-python/
Partie Cloud	MQTT • https://thingsmatic.com/2016/06/24/a-self-hosted-mqtt-environment-for-internet-of-things-part-2/ • https://hub.docker.com/_/eclipse-mosquitto
	Grafana • https://thingsmatic.com/2017/03/02/influxdb-and-grafana-for-sensor-time-series/ • https://hub.docker.com/r/grafana/grafana/
	InfluxDB https://hub.docker.com/_/influxdb