Projet Majeure CLBD 4ETI

CPE Fighter : simulation, gestion et optimisation des interventions des services d'urgence

CPE Lyon

Cahier des Charges

Table des matières

1. Introduction

1.1. Contexte

Juin 2023. Lyon est la proie des incendies. Les services de secours font appel à vous et votre équipe pour les aider à combattre les flammes. Comment ? En concevant un nouveau système d'optimisation de gestion des équipes et matériels d'urgence ou *Emergency Manager*. Ce système, qui se présentera sous la forme d'une application web, équipera les différentes casernes, et permettra aux pompiers de déterminer qui (combien de pompiers) et quoi (quel type de matériel) déployer pour venir à l'assaut de tel ou tel incendie. Chaque feu éteint rapportera des points à votre équipe ; mais attention ! D'autres services d'urgence ont fait appel à des équipes concurrentes pour mettre au point leur propre *Emergency Manager* ! A vous donc d'imaginer les meilleurs algorithmes et les meilleures stratégies pour espérer être retenus par la Ville, vendre votre système et devenir les CPE Fighters 2023 !

Afin de reproduire le plus fidèlement possible les situations auxquelles sont confrontés les services d'urgence, vous devrez **gérer différents types de feux**, chacun d'entre eux pouvant être maîtrisé de manière plus ou moins efficace par un *agent extincteur*. Il vous faudra **également administrer des casernes et des véhicules** ayant des capacités et des effectifs qui ne sont pas infinis, soumis à des contraintes de ravitaillement, de fatigue, etc. (voir **Sections 2** et **3**).

L'objectif est d'aboutir à une solution web de supervision des incendies, sur laquelle on peut **visualiser les feux et les véhicules en déplacement ou en action, administrer les véhicules et les pompiers à envoyer en mission**, etc. (voir **Section 0**).

1.2. Organisation

Le projet est à réaliser par groupes de **4 étudiants**, durant les 28h prévues à cet effet, du **24 mai au 8 juin**.

1.3. Technologies à utiliser

- La partie *front-end* devra être réalisé à l'aide des outils et langages vus durant les cours de TLW et TIDAL (HTML, CSS, JavaScript ES6…), complétés par des bibliothèques et API spécialisées dans le traitement de données géographiques (Leaflet, Mapbox…) ;
- La partie *back-end*, quant à elle, pourra être développée en Java / Spring Boot.

1.4. Evaluation

L'évaluation se fera sous la forme d'une soutenance orale de **20 minutes**, le **9 juin (matin)**, durant laquelle tous les membres de chaque équipe devront participer à la présentation du travail effectué, des résultats obtenus et des difficultés rencontrées, et effectuer une démonstration.

2. Objets considérés / Modèle de données

2.1. Feux

Dans un simulateur de gestion d'incendies, les **feux** constituent naturellement l'un des éléments principaux du modèle de données. Comme nous le verrons plus loin, les feux sont générés aléatoirement par un *simulateur de feux (Fire Simulator)*.

A propos des simulateurs de feux

Deux simulateurs de feux peuvent être utilisés :

- un simulateur de test, fourni, que vous pourrez déployer en local sur vos machines pour mettre au point vos algorithmes ;
- le simulateur de jeu, commun à toutes les équipes, qui est celui avec lequel vous devrez interagir pour éteindre les incendies dans la ville, et qui vous attribuera des points (voir section 3).

Comme en situation réelle, vous devrez gérer plusieurs types de feux.

2.1.1. Classes de feux

On catégorise les feux en **classes de feux**, nommées A, B, C, D et E :

- **Classe A :** caractérise les feux « secs ». Il s'agit de matériaux solides dont la combustion forme des braises (bois, papier, tissu, etc.). Ce sont des matériaux particulièrement inflammables ;
- **Classe B :** caractérise les feux « gras ». Il s'agit des liquides et des solides liquéfiables (hydrocarbures, goudron, graisses, huiles, peintures, vernis, alcools, solvants…) ;
- **Classe C :** caractérise les feux de gaz (gaz naturels, gaz de pétrole liquéfiés comme le butane ou le propane, ou d'autres produits à l'état gazeux comme des produits chimiques, etc.) ;
- **Classe D :** caractérise les feux de métaux ;
- **Classe E :** caractérise les « feux d'origine électrique » provenant d'équipements électriques sous tension (cette classe n'existe plus en tant que telle dans la classification européenne, mais existe toujours dans le système américain).

Les différents types de feux avec lesquels on interagit dans le simulateur sont donnés par l'énumération `FireType` :

```
public enum FireType {
      A,
      B_Gasoline,
      B_Alcohol,
      B_Plastics,
      C_Flammable_Gases,
      D_Metals,
      E_Electric;
}
```

2.1.2. Attributs d'un feu

Un feu est caractérisé par :

- son identifiant,
- son type,
- les coordonnées de son foyer,
- son intensité,
- son étendue.

2.2. Agents extincteurs

Un agent extincteur est un produit agissant sur le feu en s'opposant à la réaction de combustion. Il existe différents agents extincteurs, chacun ayant une efficacité plus ou moins grande sur les différents types de feux :

- l'eau,
- l'eau avec additif ou mousse,
- dioxyde de carbone,
- poudre.

L'énumération `LiquidType` synthétise ces informations. Par exemple, l'élément

```
WATER(0.8f,0.8f,0.25f,0.25f,0.25f)
```

indique que l'eau est efficace à 80 % sur les feux de classe A et B, et à 25 % sur ceux des classes C, D et E.

💡 Vous pourrez, dans un premier temps, vous restreindre à l'agent extincteur spécial appelé `ALL`, efficace à 100 % sur tout type de feu.

2.3. Véhicules d'intervention

2.3.1. Types de véhicules

Pour lutter contre les feux, différents types de véhicules peuvent être déployés :

- des voitures (`CAR`),
- des véhicules de secours (`FIRE_ENGINE`),
- des fourgons-pompes (`PUMPER_TRUCK`),
- des camions citernes (`WATER_TENDER`),
- des camions-échelles (`TURNTABLE_LADDER_TRUCK`),
- des camions (`TRUCK`).

Chaque type de véhicule est caractérisé par :

- l'espace qu'il occupe au sein de la caserne,
- le nombre de pompiers qui peuvent prendre place à bord,
- son efficacité, entre 0 et 10 (pleine efficacité quand l'équipage est au complet),
- sa capacité (le nombre de litres de liquide qu'il peut transporter),
- la consommation de liquide (en L / s),

- la capacité du réservoir de carburant,
- la consommation de carburant (en L / km),
- la vitesse maximale (en km / h)

Ces données constituent l'énumération `VehicleType`.

2.3.2. Attributs d'un véhicule

La classe `VehicleDto` recense les différents attributs d'un véhicule d'intervention :

- son identifiant,
- son type,
- ses coordonnées,
- le type et la quantité de liquide qu'il transporte (en L),
- la quantité de carburant restante (en L),
- le nombre de pompiers à bord,
- une valeur d'*efficacité* comprise entre 0 et 10, calculée en fonction du nombre de pompiers à bord du véhicule par rapport à sa capacité maximale (autrement dit, une efficacité de 0 si le véhicule est vide, et 10 s'il est plein),
- l'identifiant de la caserne à laquelle il est rattaché ; il s'agit de la caserne de laquelle part le véhicule et dans laquelle il doit se rendre pour refaire le plein de carburant, de liquide extincteur, ou renouveler son équipage.

2.4. Casernes

Pour gérer les véhicules, vous devez administrer des casernes (*facilities*). La classe `FacilityDto` recense les différents attributs d'une caserne :

- son identifiant,
- son nom,
- ses coordonnées,
- l'espace maximal disponible pour les camions,
- la liste des camions gérés par la caserne,
- la capacité maximale pour les effectifs,
- les pompiers affectés à cette caserne.

3. FireSimulator

Afin de modéliser l'apparition d'incendies de diverses origines dans la ville, **un *simulateur de feux* (`FireSimulator`) est fourni.** Celui-ci a en charge d'une part de générer des feux (avec une fréquence, des coordonnées, une intensité, etc. aléatoires), et d'autre part de gérer leur évolution en fonction des véhicules d'intervention qui sont déployés.

3.1. Interface de supervision des feux

Pour interagir avec le FireSimulator, une API web est mise à votre disposition à cette adresse : http://vps.cpe-sn.fr:8081/.

Une documentation complète au format Swagger de cette API est disponible à l'adresse suivante :

http://vps.cpe-sn.fr:8081/swagger-ui/

L'API expose en premier lieu des *endpoints* permettant de gérer les incendies :

API de gestion des incendies	
`/fire`	Retourne la liste de tous les feux actifs
`/fire/reset`	Permet de réinitialiser la base de données des feux
`/config/creation`	(GET) : Retourne les paramètres régissant la création de feux (PUT) : Permet de modifier un de ces paramètres
`/config/behavior`	(GET) : Retourne les paramètres régissant le comportement des feux, comme la probabilité de réplication ou de propagation (PUT) : Permet de modifier un de ces paramètres

Remarque : toutes les méthodes GET renvoient le résultat au format JSON

L'API permet également de déployer des véhicules d'intervention. Comme indiqué ci-dessus, c'est le FireSimulator qui se charge alors de faire évoluer les feux en fonction des véhicules d'interventions présents (ou non) sur le lieu de l'incendie. Les *endpoints* à votre disposition sont :

API de gestion des véhicules	
`/vehicle (POST)`	Ajoute un nouveau véhicule dans le simulateur, en indiquant (dans un objet JSON) ses différents attributs. Le véhicule est créé dans la caserne spécifiée.
`/vehicle (GET)`	Retourne la liste de tous les véhicules
`/vehicle/{id} (GET)`	Retourne le véhicule d'identifiant *id*
`/vehicle/{id} (PUT)`	Permet de modifier / mettre à jour les attributs du véhicule *id* (notamment ses coordonnées)
`/vehicle/{id} (DELETE)`	Supprime le véhicule *id* du simulateur

/vehicle/move/{teamuuid}/{id} (PUT)	Déplace un véhicule aux coordonnées spécifiées

Pour administrer vos casernes, les *endpoints* suivants sont disponibles :

API de gestion des casernes	
/facility (POST)	Ajoute une nouvelle caserne dans le simulateur, en spécifiant (dans un objet JSON) ses différents attributs
/facility (GET)	Retourne la liste de toutes les casernes
/facility (DELETE)	Supprime toutes les casernes
/facility/{id} (GET)	Retourne la caserne d'identifiant *id*
/facility/{id} (DELETE)	Supprime la caserne d'identifiant *id*
/facility/object/{id} (GET)	Retourne la description *géométrique* de la caserne

3.2. Installation du simulateur de feu de test

Vous pouvez, si vous le souhaitez, utiliser un simulateur de test en local sur votre machine. Pour l'installer, il est nécessaire de disposer de Docker, docker-compose, un espace disque suffisant et une bonne connexion internet (pour la première fois).

1. Aller sur https://gitlab.com/js-project-gis-1/js-fire-simulator-starter
2. Cloner le dépôt sur votre poste :

```
git clone https://gitlab.com/js-project-gis-1/js-fire-simulator-starter.git
```

3. Démarrer le docker-compose

```
cd js-fire-simulator-starter

docker-compose up
```

3.3. Calcul de l'atténuation d'un feu

L'atténuation de chaque feu est calculée comme suit :

Find all vehicles in fire $\boldsymbol{f}$ range :

For all vehicles $\boldsymbol{v}$ found :

$$\begin{aligned} \boldsymbol{f}.intensity = \; & \boldsymbol{f}.intensity \\ & - \; \boldsymbol{v}.efficiency \times \boldsymbol{v}.liquidType.efficiency(\boldsymbol{f}.type) \\ & \times \; fireUpdateConfig.attenuationFactor \end{aligned}$$

4. Cahier des charges / fonctionnalités attendues

Les fonctionnalités à développer sont découpées selon les lots qui suivent.

4.1. Interface de supervision des feux

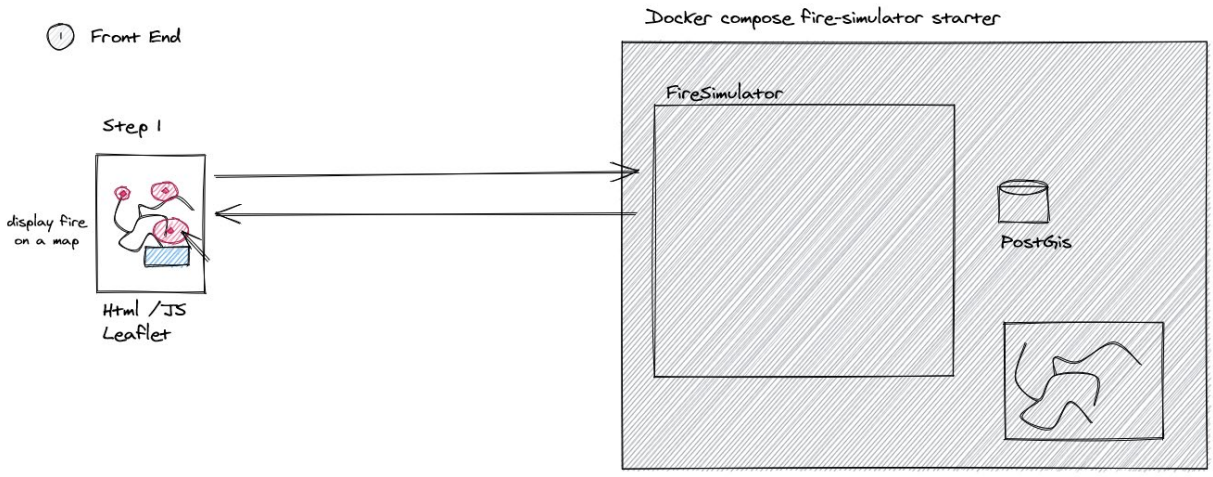
Le premier lot consiste à développer une interface web de supervision des feux. Autrement dit, il s'agit d'une page web exposant une carte de la zone géographique supervisée, sur laquelle les incendies sont affichés.

En plus du simple affichage des feux, on attend également les fonctionnalités suivantes :

- La visualisation des attributs d'un feu (intensité, étendue…) en cliquant dessus ;
- Une interface simple permettant d'afficher / masquer les feux selon leur type, leur intensité, leur étendue…

Pour développer cette interface, vous utiliserez la bibliothèque JavaScript Leaflet, spécialisée dans l'affichage de données géographiques. Ces données (le fonds de carte notamment) peuvent provenir de différents fournisseurs (OpenStreetMap, Google Maps…). Il est recommandé d'utiliser Mapbox[1], qui fournit gratuitement un grand nombre de services : tuiles cartographiques, calcul d'itinéraires, géocodage (c'est-à-dire le passage d'une adresse postale à des coordonnées) direct ou inverse…

.



4.2. Supervision des véhicules d'intervention

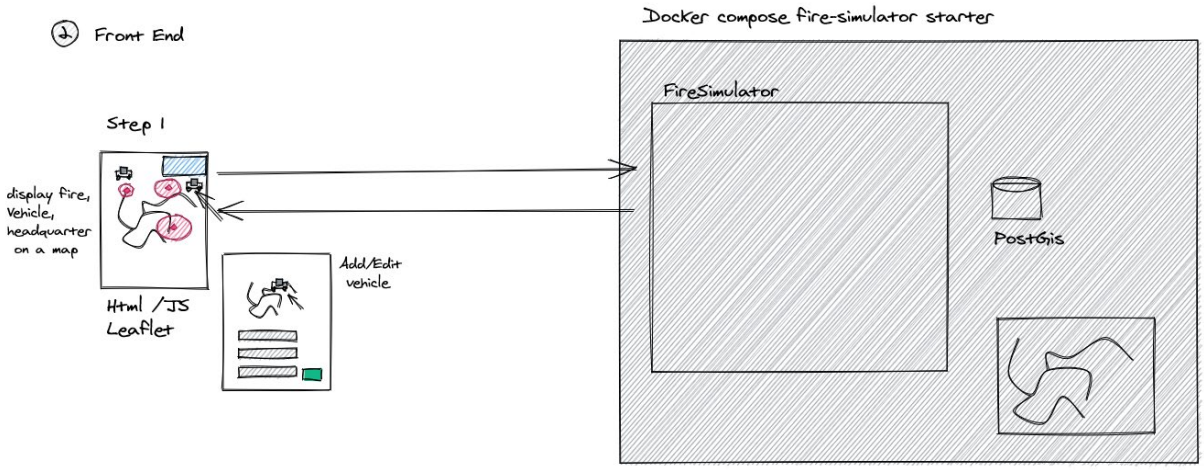
Le second lot consiste à étendre l'interface de supervision du Lot 1 afin de pouvoir visualiser les déplacements des véhicules d'intervention.

Comme pour le Lot 1, en plus du simple affichage des véhicules, quelques fonctionnalités supplémentaires sont attendues :

- Une interface simple permettant de créer / modifier / supprimer des véhicules d'intervention

[1] Depuis 2022, Mapbox demande un numéro de carte de crédit lors de l'inscription. Vous pouvez utiliser un générateur de CB en ligne pour pallier cet inconvénient. Il existe également des alternatives, mais souvent moins complètes.

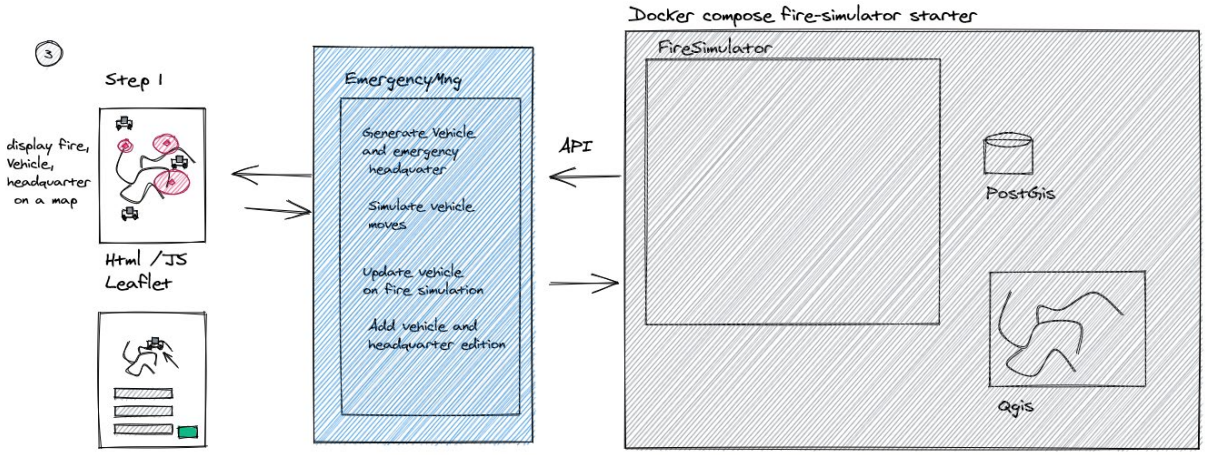
- La visualisation des attributs d'un véhicule (type, nombre d'équipiers, type de liquide, quantité de liquide restante…) en cliquant dessus ;
- Une interface simple permettant d'afficher / masquer les véhicules selon leur type, le type de liquide…



4.3. Gestionnaire d'urgences *(Emergency Manager)*

Le troisième lot est probablement le plus important puisqu'il consiste à développer le gestionnaire d'urgences ou *Emergency Manager*, qui contient toute la logique de déploiement des véhicules d'intervention vers les incendies. C'est donc ici que vous allez décrire quels véhicules sont affectés à quels incendies, et la manière dont ils se déplacent. Plusieurs niveaux de complexité peuvent être envisagés, et il est conseillé d'implémenter les fonctionnalités de manière incrémentale. Par exemple :

Fonctionnalité	Objectifs	Difficulté		
		1	2	3
Déplacement des véhicules				
	« Téléportation » sur le site de l'incendie	X		
	Déplacement en ligne droite		X	
	Déplacement en suivant le réseau routier			X
Affectation des véhicules	Affectation manuelle	X		
	Affectation gloutonne (premier véhicule disponible affecté)		X	
	Affectation optimisée (selon la distance, le type de feu…)			X



Afin de pouvoir communiquer avec le simulateur de feu fourni, il vous est fortement conseillé d'utiliser la librairie publique https://gitlab.com/js-project-gis-1/js-fire-simulator-public qui définit le format des objets à utiliser avec le simulateur, les différents enums et DTO, ainsi que des utilitaires permettant de transformer des coordonnées dans différentes projections et de calculer des distances entre deux points.

Vous pouvez intégrer cette librairie comme stipulé ici https://gitlab.com/js-project-gis-1/js-fire-simulator-public/-/packages/14132618 dans vos projets Maven.

4.4. Extensions de l'Emergency Manager

C'est le lot dans lequel vous pouvez laisser libre cours à votre imagination pour étendre au maximum les fonctionnalités de l'Emergency Manager. Parmi les extensions possibles / suggérées, on peut citer :

- La gestion des quantités et consommation d'agent extincteurs et de carburant :
  - un véhicule utilise une quantité de carburant proportionnelle à la distance qu'il parcourt ; s'il est à court de carburant, il doit retourner faire le plein dans sa caserne (attention à conserver suffisamment de carburant pour pouvoir revenir à la caserne !) ;
  - pour éteindre un incendie, un véhicule consomme une certaine quantité d'agent extincteur chaque seconde ; comme la capacité d'un véhicule en agent extincteur est limitée, il faut aussi prévoir de refaire le plein dans la caserne ;
- La gestion de la fatigue : on peut envisager qu'un pompier doive rester inactif pendant un certain temps avant de pouvoir repartir en mission ;
- La gestion de points de ravitaillement externes aux casernes (stations-essence, point d'approvisionnement en agents extincteurs…)
- Etc. !