**TROISIEME**

**PARTIE**

**:**

**ETUDE**

**TECHNIQUE**

**ET**

**IMPLEMENTATION**

**DE**

**LA**

**SOLUTION**

## CHAPITRE 6 : ETUDE TECHNIQUE

L’étude technique va s’attarder sur la conception et l’analyse de faisabilité du projet. Elle visera à répondre à des questions techniques spécifiques. Dans cette partie, nous allons donc aborder la conception globale et détaillée de notre projet.

La conception globale de notre projet implique une vue d’ensemble qui permet de rassembler et de coordonner tous les éléments nécessaires pour sa réalisation, en identifiant les grandes composantes, les interactions entre celles-ci et les flux de données. Elle permet de définir l’architecture générale et les principaux modules du système.

La conception détaillée, quant à elle, se penche sur chaque composant individuellement. Elle spécifie les détails techniques, les algorithmes, les interfaces, les bases de données, etc. Cette étape est cruciale pour garantir la faisabilité technique du projet.

### 6.1. ANALYSE

**SmartHelp** est une solution destinée à détecter et signaler des catastrophes[[1]](#footnote-1) (inondations, incendies, tremblements de terre, crises sanitaires) dans les zones enclavées[[2]](#footnote-2). Elle s’appuie sur une plateforme de suivi en temps réel connectée à un réseau de dispositifs émetteurs/récepteurs, utilisant les infrastructures MPLS/VPN existantes. Une Intelligence Artificielle intervient pour filtrer et structurer les données scrappées, facilitant ainsi l’identification des risques et l’aide à la prise de décision dans les contextes d’urgence.

**6.1.1. Exigences fonctionnelles**

Plusieurs fonctionnalités clés peuvent être identifiées, parmi lesquelles :

* Alertes en temps réel ;
* Imagerie en temps réel ;
* Surveillance à distance des zones enclavées ;  Système d’authentification sécurisé .

**6.1.2. Exigences non fonctionnelles**

Entre autres, nous avons :

* La performance
* La précision
* La sécurité
* La robustesse
* La disponibilité
* La convivialité
* Les standardisations des interfaces

### 6.2. CONCEPTION GENERALE

Dans la phase de développement de notre solution, nous avons choisi d’adopter la méthode **DSDM (Dynamic System Development Method)**. Cette approche agile présente plusieurs avantages, notamment la capacité à fournir rapidement des informations pertinentes en cas de catastrophe, l'encouragement au travail collaboratif en équipe et une forte interaction avec le client tout au long du processus.

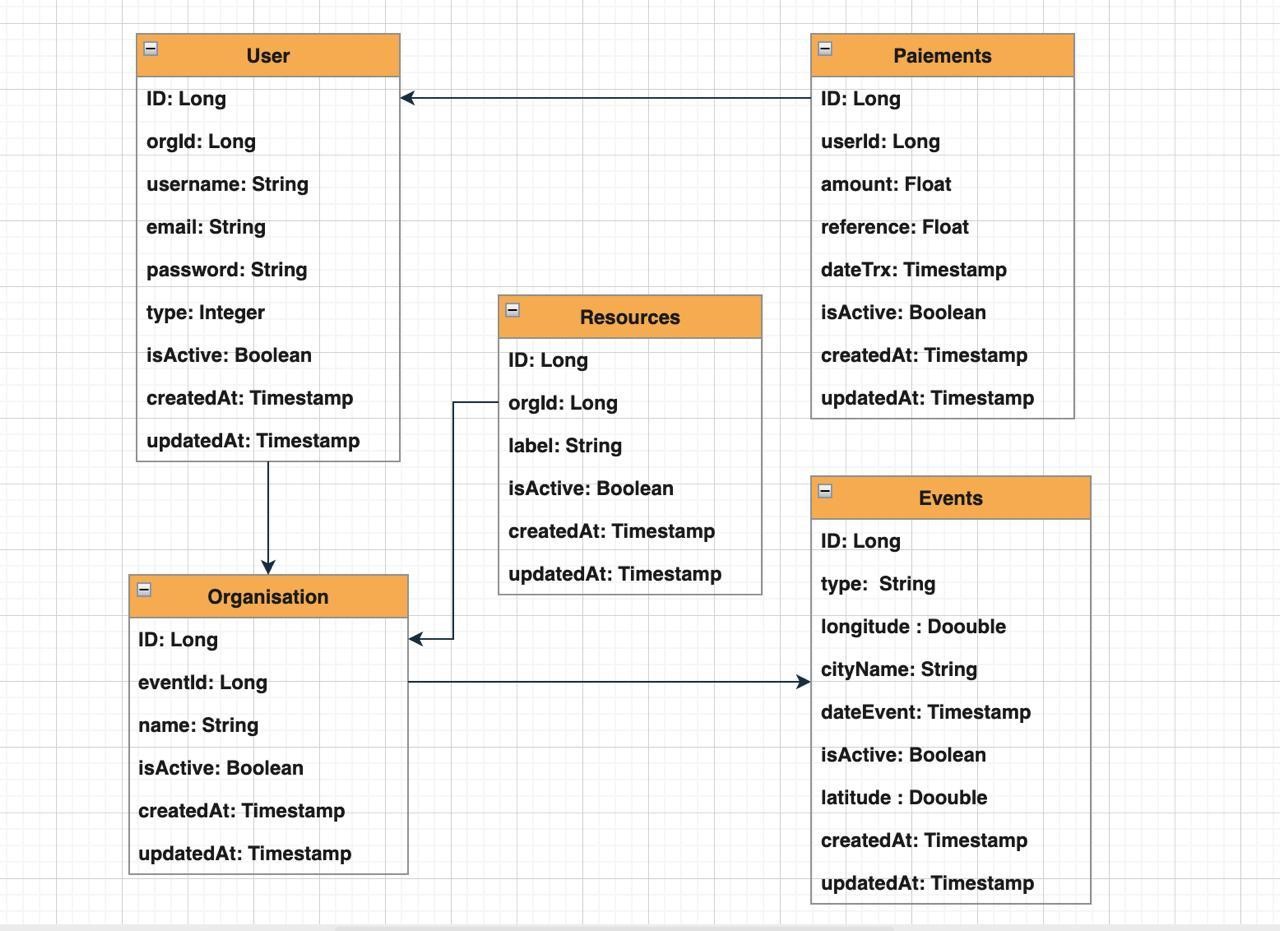
Deux approches de conception logicielle ont été mobilisées : **l’approche fonctionnelle** et **l’approche exploratoire**.

* L’**approche fonctionnelle** nous a permis de structurer les codes de contrôle du dispositif en modules bien définis, facilitant ainsi leur interaction et leur intégration.
* L’**approche exploratoire**, quant à elle, a joué un rôle essentiel dans la phase initiale. Elle nous a permis de mieux cerner le problème de recherche, d’analyser le contexte et de comprendre les besoins réels des utilisateurs.

Par la suite, nous avons étudié les architectures matérielles et logicielles les plus adaptées à la mise en œuvre de notre solution, dans le but de définir une **architecture système globale** cohérente. Sur le plan logiciel, nous avons opté dans un premier temps pour une plateforme répondant aux caractéristiques suivantes :

**6.2.1. Diagramme de classe**

Le **diagramme de classes** représente la structure statique du système en modélisant les classes, leurs attributs, leurs méthodes et leurs relations. Il sert de support à la conception, facilite la compréhension du système et constitue une base pour le développement et la maintenance du logiciel.



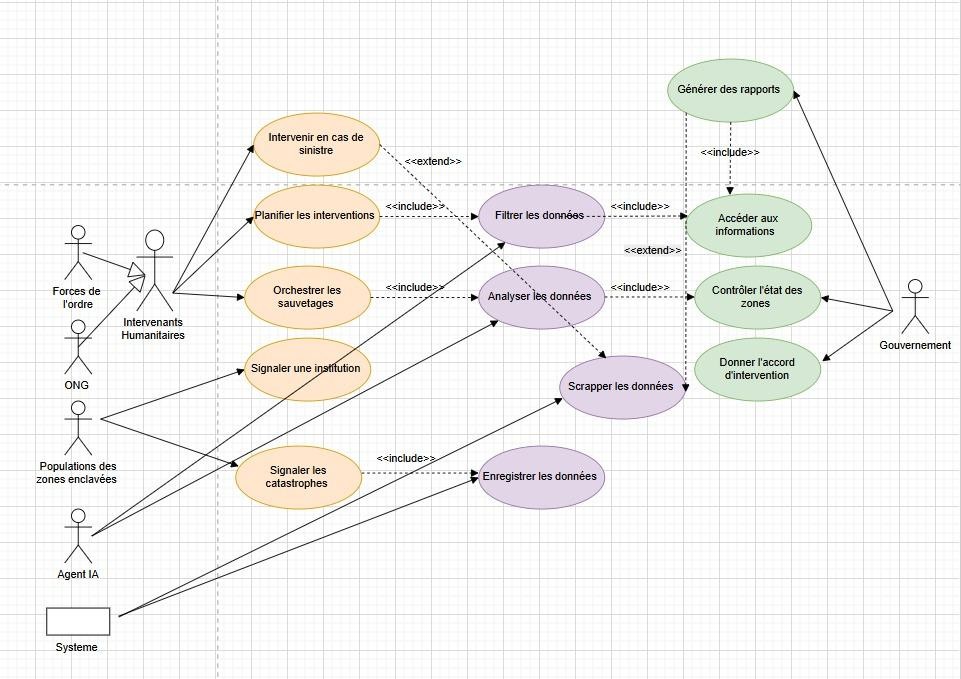
#### Figure 23: Diagramme de classe (Source : les promoteurs)

**6.2.2. Diagramme de cas d’utilisation**

Le diagramme de cas d’utilisation montre les différentes interactions possibles entre les utilisateurs (appelés "acteurs") et le système. Il représente ce que le système est censé faire du point de vue de l’utilisateur, sans détailler le fonctionnement interne.

Il sert à identifier les fonctionnalités principales du système (les "cas d’utilisation") et qui les utilise. Il répond à la question : *Qui fait quoi avec le système ?*

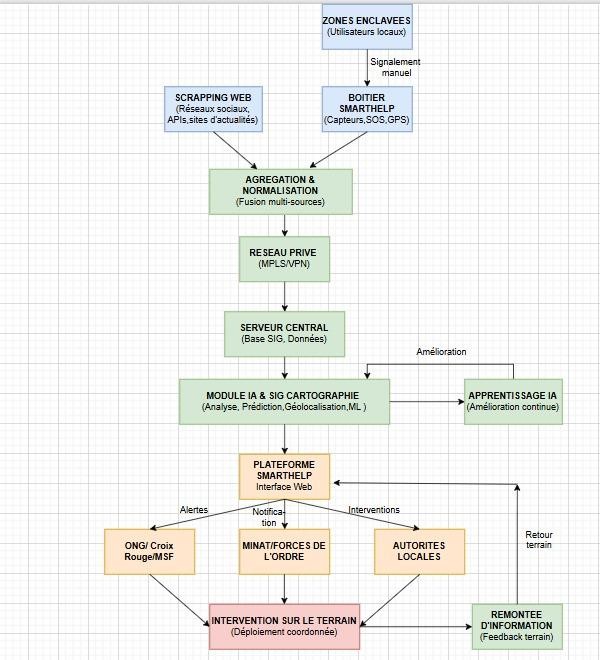
C’est un outil précieux pour définir les exigences fonctionnelles du système et pour communiquer efficacement entre les développeurs, les utilisateurs et autres parties prenantes sur le comportement attendu du logiciel.



#### Figure 24: Diagramme de cas d’utilisation (Source : les promoteurs)

**6.2.3. Diagramme de flux**

Dans un projet technique, le diagramme de flux sert à représenter de manière simple et visuelle comment les informations ou les données circulent à travers le système. Il indique uniquement où les données entrent, par où elles passent, et où elles sortent sans expliquer en détail comment elles sont traitées à chaque étape. L’objectif est de comprendre le cheminement des flux d’information dans le dispositif ou le système technique.



***Figure 25 : Diagramme de flux (Source : les promoteurs)***

### 6.3. ATELIER DE PRODUCTION

Dans le contexte de notre projet, nous avons organisé notre processus de production en trois grands ateliers spécialisés :

* Atelier électronique ;  Atelier informatique ;
* Atelier de modélisation.

Chaque atelier joue un rôle essentiel dans la réalisation de notre solution, et leur collaboration garantit un produit final fonctionnel et performant.

**6.3.1. Atelier électronique**

Cet espace est dédié au montage des composants électroniques nécessaires à notre solution. Il englobe la conception de circuits, le soudage, les tests et l’assemblage des cartes électroniques et des dispositifs ainsi que de leurs mises en réseau.

**6.3.2. Atelier informatique**

Ici, nous nous concentrons sur le développement logiciel et la programmation. Cet atelier couvre la création d’application, la configuration des systèmes, la gestion des bases de données, des serveurs et la mise en place des interfaces utilisateurs.

**6.3.3. Atelier de modélisation et de simulation**

Dans l’atelier, nous travaillons sur l’amélioration de la Map, l’IA et l’extraction et le filtrage des données afin de recueillir le maximum d’information pour optimiser les interventions humanitaires.

### 6.4. PROCESSUS DE REALISATION

Le processus de réalisation décrit les différentes étapes qui jalonnent la réalisation d’un projet. Il permet de planifier les opérations à mettre en place pour concevoir un prototype. Pour réaliser notre solution, le travail va s’effectuer dans 3 ateliers qui sont :

Atelier

é

lectronique

Atelier

informatique

Atelier

de

mod

é

lisation

et

de

simulation



#### Figure 26 : Processus de réalisation (Source : les promoteurs)

Le montage électronique de nos dispositifs émetteur et récepteur se fera en atelier. C’est également dans cet atelier que nous vérifions également que les montages pour le dispositif fonctionnent correctement. Pour cela nous aurons besoins des équipements suivants :

* **Modem Wifi** : Pour nos travaux de recherche et de développement de la solution
* **Poste de travail** : Laptop ou Desktop de minimum 16 Go de RAM, au moins 50 Go de mémoire interne vide et d’un processeur qui avoisine les 3Ghz pour un entrainement rapide.

Nous allons vous présenter les étapes nécessaires pour la conception de nos dispositifs ainsi, les canaux de communications réseaux et d’interconnexions de nos différents dispositifs :

Dans cet atelier, nous développons la plateforme de contrôle de notre prototype. Le choix du langage de programmation pour cette plateforme est le Python et nous déployons la plateforme sur Desktop.

### 6.5. LES ENVIRONNEMENTS DE DEPLOIEMENT

Afin de mettre sur pieds notre solution basée sur le l’imagerie 3D et l’IA, nous aurons besoin du matériel suivant :

**6.5.1. Environnement matériel**

Celui-ci est constitué des équipements constituant principalement le matériel de conception et montage du dispositif, et ils sont les suivants :

1. **Outils d’assemblage électronique / prototypage**
   * Fer à souder (avec support et éponge)
   * Fil à souder (étain + flux)
   * Pince coupante
   * Pince à dénuder
   * Tournevis de précision (kit complet)
   * Pinces brucelles (antistatiques, pour composants SMD)
   * Planche de montage (breadboard)
   * Câbles Dupont (mâle-mâle, mâle-femelle, etc.)
   * Multimètre numérique (pour tester les tensions, continuité, etc.)
   * Boîte à outils complète (clés, pinces, etc.)
2. **Équipements de test, mesure et sécurité**
   * Alimentation de laboratoire (régulée, variable en tension et courant)
   * Générateur de signaux (si besoin de test RF)
   * Boîte de rangement pour composants électroniques
3. **Matériel informatique**
   * Ordinateurs portables ou fixes (au moins 2, dont 1 pour développement, 1 pour supervision)
   * Disque dur externe ou SSD portable (sauvegardes)
   * Clés USB de transfert
   * Logiciels utiles installés (IDE Arduino, Visual Studio Code, QGIS, etc.)
4. **Matériaux et composants électroniques**
   * Microcontrôleurs (Arduino, ESP32, etc.)
   * Modules de communication (GSM, etc.)
   * Modules GPS / IMU
   * Capteurs divers
   * LED, résistances, transistors, relais, etc.
   * Connecteurs, borniers, jumpers
5. **Équipements mécaniques et de terrain**
   * Boîtiers de protection (étanches si possible)
   * Vis, écrous, boulons (jeu complet)
   * Supports, rails, châssis (pour dispositifs fixes ou portables)
   * Ruban adhésif isolant, velcro, colliers de serrage
6. **Systèmes d’alimentation**
   * Chargeurs compatibles
   * Panneaux solaires + régulateur de charge

**6.5.2. Environnement logiciel**

L’environnement logiciel pour la mise en place du projet est présenté tel que suit

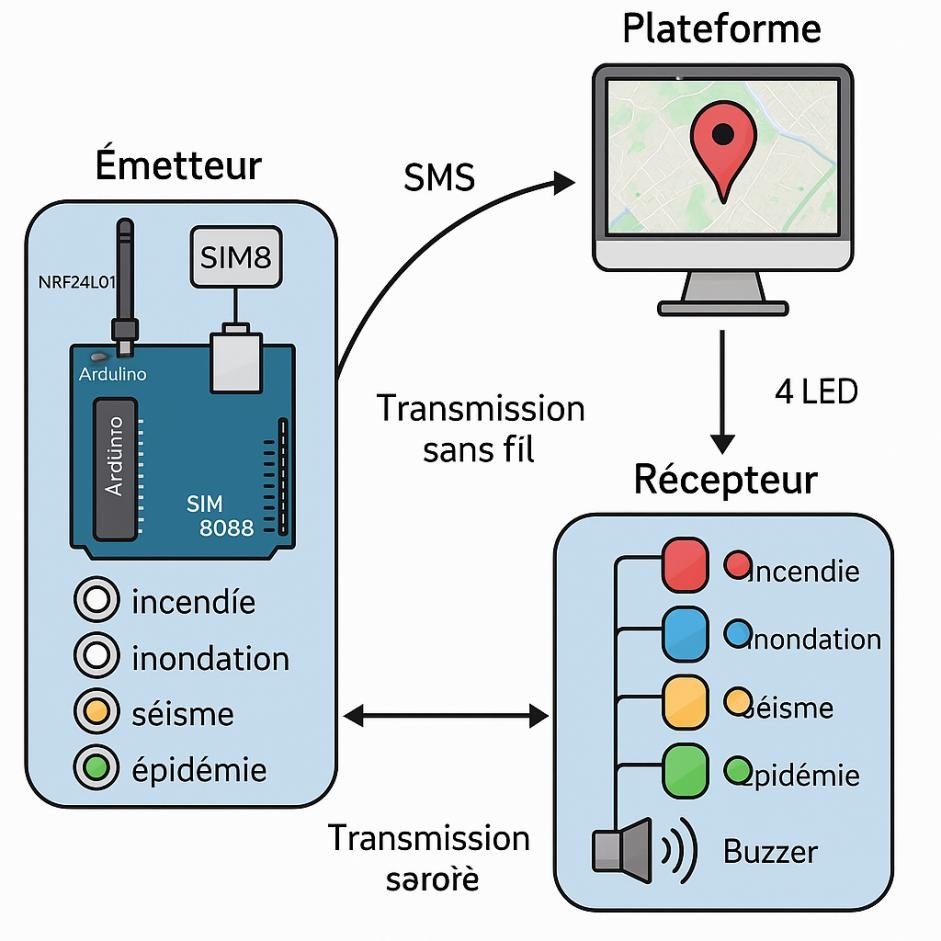
|  |  |  |  |
| --- | --- | --- | --- |
| **Windows**  **Professional** | **10** |  | C ’ est le système d ’ exploitation utilisé pour le développement, la documentation et le déploiement de notre solution. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Microsoft**  **2016** | **Word** |  | Exploiter pour la rédaction du mémoire. |
| **Microsoft**  **2016** | **Excel** |  | Exploiter pour le plan financier de notre projet. |
| **Star UML** |  |  | Aider à concevoir, documenter et visualiser l ’ architecture et le comportement d ’ un système logiciel à l ’ aide de diagrammes UML standardisés. |
| **PostMan** |  |  | Permet de construire, tester et documenter des requêtes HTTP vers des API REST ou SOAP. Utilisé pour simuler l ’ envoi de données et analyser les réponses des serveurs web et également  texter Les API |

|  |  |  |
| --- | --- | --- |
| **Visual Studio Code** |  | Sert à écrire, modifier, exécuter et déboguer du code dans plusieurs langages de  programmation (Python, JavaScript,  HTML, etc.). Il dispose d ’ une large gamme d ’ extensions pour adapter l ’ environnement à différents projets. |
| **Mistral IA** |  | Utiliser dans le cadre du filtrage des  données recueillies |
| **GNS 3** |  | Permet de créer des laboratoires virtuels pour simuler des topologies réseau a l ’ instar du réseau MPLS/VPN dans le cadre de notre projet. Très utilisé pour les formations  CCNA/CCNP ou les tests de configurations |

|  |  |  |
| --- | --- | --- |
|  |  | réseau sans matériel physique. |
| **Arduino** |  | Utilisée pour créer des projets électroniques  interactifs (automatisation, robotique, capteurs, etc.). Permet de lire des données de capteurs et de contrôler des actionneurs (LED, moteurs, etc.) via un code C++ simple. |
| **Django** |  | C ’ est une Framework Web Python opensource qui facilite le développement d ’ applications robustes et évolutives. Nous l ’ utilisons pour concevoir la  plateforme |
| **HTML** |  | C ’ est un langage utilisé pour la conception de notre plateforme web |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **CSS** |  | Langage utilisé pour decrire la presentation des documents HTML et XML |
| **JavaScript** |  | Langage de programmation utilisé pour rendre les pages web interactives et dynamiques |



***Figure 27 : Architecture générale fonctionnelle***

### 6.6. CONCEPTION LOGICIEL POUR L’EXTRACTION DE DONNEES ET TRAITEMENT POUR ENVOI SUR LA PLATEFORME

**1. Extraction quotidienne des événements (RSS GDACS)**

Chaque jour, un script Python génère dynamiquement l’URL du flux RSS GDACS en utilisant la date du jour et la veille.

Ce script est exécuté quotidiennement (via un cron ou une tâche planifiée). Il utilise requests pour télécharger le contenu RSS et le paquet feedparser (ou, en l’absence de feedparser, xml.etree.ElementTree ) pour parser le XML et extraire les champs souhaités (titre, description, type d’événement, date, coordonnées géographiques, etc.).

Par exemple, **feedparser** lit facilement les flux RSS en retournant une liste d’objets « entry » contenant les titres, descriptions, dates, etc. de chaque item . On prend soin de n’inclure que les événements récents et pertinents (par exemple en filtrant sur la date du jour et de la veille) et de gérer les erreurs HTTP (code de réponse) avant d’analyser le XML.

**Extrait de code du scraping**

standard\_url = ["http://www.gdacs.org/xml/rss.xml](http://www.gdacs.org/xml/rss.xml)" response = requests.get(standard\_url) root = ET.fromstring(response.content) for item in root.findall('.//item'):

try:

event\_id = item.find('.//gdacs:eventid', namespaces).text if item.find('.//gdacs:eventid', namespaces) is not None else ""

title = item.find('title').text if item.find('title') is not None else "" description = item.find('description').text if item.find('description') is not None else "" geo\_point = item.find('.//geo:Point', namespaces) latitude = "" longitude = "" lat\_elem = geo\_point.find('./geo:lat', namespaces) long\_elem = geo\_point.find('./geo:long', namespaces) latitude = lat\_elem.text longitude = long\_elem.text

**2. Injection dans l’API Django REST**

Les données extraites doivent être converties en objets Django et stockées en base via l’API REST.

Pour cela :

* Modèle Django ( models.py ) : on crée un modèle Event (ou Evenement ) avec des champs correspondant aux données du RSS, par exemple : title (CharField), description (TextField), event\_type (CharField), date (DateTimeField), latitude (FloatField), longitude (FloatField), alert\_level (CharField), etc. Un champ id ou event\_id unique peut être utilisé pour éviter les doublons.
* Serializer DRF ( serializers.py ) : on définit un EventSerializer , de préférence en héritant de serializers.ModelSerializer , en indiquant simplement le modèle et les champs à exposer . Par exemple :



Cette approche ModelSerializer génère automatiquement les champs du modèle sans avoir à les redéclarer manuellement.

* Vue ou ViewSet DRF ( views.py ) : on peut créer un ViewSet ou une APIView pour gérer les opérations CRUD sur les événements. Par exemple, un EventViewSet permettrait de lister ( GET /api/events/ ), créer ( POST ), mettre à jour, etc. Les données scrappées peuvent être injectées en utilisant un serializer. DRF utilise donc serializer.save() pour créer l’instance grâce à la méthode create() du serializer, comme illustré dans la documentation officielle.
* Base de données : Django crée la table correspondante (avec SQLite par défaut en local) via une migration. Les événements sont ainsi stockés et liés à l’API. En production ou en locale, on peut utiliser manage.py makemigrations puis migrate pour initialiser notre table Event.

**3. Intégration front-end (dashboard HTML/JS)**

Le dashboard.html appelle l’API Django REST pour récupérer les événements en JSON et les afficher. Typiquement, on utilise JavaScript (vanilla ou framework léger) pour effectuer une requête asynchrone, par exemple :



Carte Leaflet

Après avoir obtenu la liste des événements (objet JSON), le script JavaScript crée des marqueurs pour chaque événement. Par exemple, on peut utiliser la syntaxe suivante : L.marker([evt.latitude, evt.longitude]).addTo(map);

On peut personnaliser l’icône du marqueur selon le type ou le niveau d’alerte (par exemple, une icône rouge pour les événements de type « critique »).

Chaque marqueur peut afficher un popup contenant les détails de l’événement (titre, date, description). Il est également possible de :

* Regrouper les marqueurs par couches,
* Utiliser L.markerClusterGroup pour le clustering dynamique.

Pour charger les données sur la carte, une autre méthode consiste à transformer les événements en objet GeoJSON, et à utiliser : L.geoJSON(geojsonData).addTo(map);

Cela permet à Leaflet d’afficher automatiquement les points avec un style commun et des popups configurables.

Graphiques Chart.js

On peut également exploiter les mêmes données JSON pour alimenter des graphiques via Chart.js.

Par exemple, pour créer un diagramme en anneau affichant le nombre d’événements par niveau d’alerte :

1. On compte les occurrences de chaque niveau ('critical', 'high', etc.).
2. On construit un objet au format suivant :

{

labels: [...], data: [...]

}

1. On passe cet objet au constructeur Chart new Chart(ctx, { type: 'doughnut', data: dataSet, options: { ... }

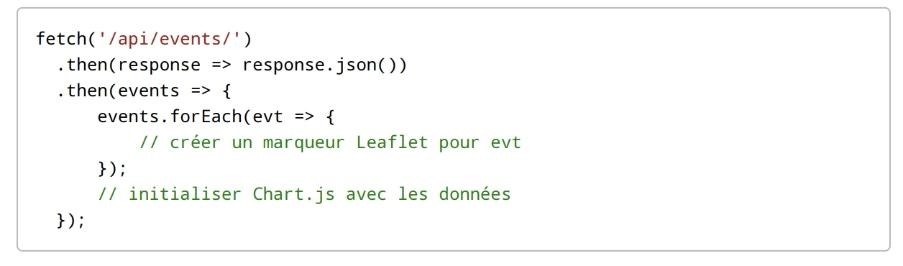
});

Chart.js fournit une API simple. Il suffit de :

* Sélectionner le <canvas> correspondant,
* Instancier un objet Chart,
* Spécifier le type de graphique ('bar', 'line', 'pie', etc.) et les données.

Cela permet de créer différents types de visualisations sur la carte:

* Histogrammes (barres),
* Courbes (évolution temporelle),
* Secteurs (répartition par type ou niveau),





**4. Filtrage et enrichissement avec l’API Mistral AI**

Pour adapter les données au contexte camerounais, on peut utiliser l’API Mistral AI via les endpoints de complétion, de classification ou d’embeddings.

1. **Classification de la Pertinence Locale**

Chaque événement collecté (scrappé) peut être transmis à un modèle de classification de Mistral via une requête :POST /v1/classifications

Un prompt approprié pourrait être :

« Cet événement est-il situé au Cameroun ou a-t-il un impact significatif au Cameroun ? »

Le modèle retourne alors :0

* + Une note de pertinence ou
  + Une catégorie de classification

Ce qui permet de filtrer les événements et de ne conserver que ceux jugés pertinents pour le Cameroun.

1. **Génération de Tags et Évaluation d’Urgence**

On peut également demander au modèle Mistral de :

* + Générer des tags automatiquement,
  + Évaluer le niveau d’urgence d’un événement sur la base de sa description, via un prompt du type : « Classify urgency level as high / medium / low »

En pratique, la requête envoyée est au format JSON :

{

"model": "nom\_du\_modèle",

"input": "texte de l’événement"

}

1. **Utilisation des Embeddings pour la Similarité Sémantique**

L’API mistral-embed permet de comparer sémantiquement la description d’un événement avec des expressions-clés liées au Cameroun.

* + On génère un vecteur d’embedding pour le mot *"Cameroon"*,
  + Un autre vecteur pour la description de l’événement,
  + Puis on calcule la similarité cosinus entre les deux.

Un score élevé indique que l’événement est localement pertinent.

Ce procédé s’inspire des systèmes RAG (Retrieval-Augmented Generation), où les embeddings permettent un filtrage sémantique efficace.

1. **Enrichissement des Données par l’IA**

Enfin, Mistral AI peut être utilisé pour enrichir les données, par exemple :

* + Traduction ou résumé de la description dans une langue locale,
  + Inférence d’informations manquantes, comme la prédiction du type de catastrophe si ce champ est vide.

1. **Cartographie avec OpenStreetMap/Leaflet**

La carte interactive utilise Leaflet.js avec les tuiles OpenStreetMap. Leaflet est agnostique quant aux fournisseurs de tuiles – on peut donc charger celles d’OSM ainsi :



Cette ligne (exemple tiré de la documentation Leaflet) permet d’ajouter un fond de carte OpenStreetMap (OSM) au conteneur map. Leaflet offre ensuite la possibilité d’ajouter divers éléments visuels tels que des marqueurs (L.marker), des cercles, des polygones, et même des couches de chaleur (via le plugin Leaflet.heat) pour représenter visuellement les zones à risque. Chaque zone ciblée peut être matérialisée par un point ou un polygone représentant une concentration d’événements.

Les marqueurs peuvent être personnalisés à l’aide d’icônes spécifiques (par exemple avec FontAwesome) afin de distinguer les différents types de catastrophes : incendie, inondation, séisme, etc. Lorsque le nombre d’événements devient important, l’utilisation de clusters de marqueurs (grâce à leaflet.markercluster) permet d’améliorer considérablement la lisibilité de la carte.

Enfin, Leaflet prend en charge les événements de souris (comme click ou hover) pour afficher des popups contenant des informations détaillées, avec une mise en forme soignée comme illustré dans le code du tableau de bord fourni.

**7. Schéma d’architecture du système**

Le système se décompose en plusieurs composants locaux : un scraper (script Python), le serveur API Django REST, la base de données, et le frontend (HTML/JS). Par exemple :

Le système repose sur plusieurs composants interconnectés pour collecter, stocker, analyser et visualiser les événements liés aux catastrophes.

Un scraper Python, exécuté quotidiennement via une tâche cron, récupère le flux RSS de GDACS (Global Disaster Alert and Coordination System). Les événements extraits sont ensuite envoyés à une API Django REST via des requêtes POST pour être enregistrés.

L’API Django REST expose les endpoints nécessaires (notamment /api/events/) pour permettre les opérations de création, lecture, mise à jour et suppression (CRUD). Les données reçues sont validées à l’aide de serializers, puis stockées en base via le Django ORM, avec une base de données SQLite utilisée dans ce cas.

Le dashboard web, développé en HTML et JavaScript, interagit avec l’API pour afficher dynamiquement les événements. Il se compose principalement de deux modules visuels :

* Leaflet (avec OpenStreetMap) : pour l’affichage cartographique interactif des événements, avec des marqueurs personnalisés, des couches de risques et des fonctionnalités de clustering.
* Chart.js : pour la visualisation graphique des données sous forme de diagrammes (répartition par niveau d’alerte, évolution temporelle, etc.).

Enfin, une intégration avec l’API Mistral AI (cloud) permet une analyse sémantique avancée des événements, notamment pour :

* Filtrer les événements pertinents pour le Cameroun,
* Effectuer une classification par niveau d’urgence,
* Générer des tags ou enrichir les données,
* Utiliser des embeddings pour mesurer la pertinence contextuelle des événements avant ou pendant l’injection dans le système.

Scrapper les données → API Django REST → Base de données

Scrapper les données → Mistral AI (filtrage) → Frontend (HTML/JS)

Frontend (HTML/JS) → Leaflet + Chart.js (visualisation) → API Django REST

(requêtes GET vers l'API et réception JSON) → Frontend (HTML/JS)

**8. Technologies utilisées**

Le projet repose sur un ensemble cohérent de technologies couvrant le backend, le frontend, la visualisation cartographique, l’intelligence artificielle et la persistance des données.

**Langages**

Le développement s’appuie principalement sur Python pour le scraper et le backend (avec Django), ainsi que sur JavaScript, HTML et CSS pour le frontend.

**Bibliothèques Python**

Les bibliothèques utilisées incluent notamment :

* requests pour les appels HTTP,
* feedparser pour lire les flux RSS,
* xml.etree.ElementTree pour l’analyse XML,
* datetime, csv (ou éventuellement pandas) pour la gestion des données,
* Le framework Django avec Django REST Framework pour la création d’API REST,
* Et, de manière optionnelle : python-dotenv pour la configuration, schedule ou cron pour l’automatisation.

**Frameworks et outils Django**

Le backend repose sur Django, avec l’utilisation de Django REST Framework pour la sérialisation des données (serializer), les vues (viewset) et le routage (router).

**Frontend**

Le frontend utilise :

* Leaflet.js pour la cartographie interactive, avec les plugins MarkerCluster et Heatmap pour le regroupement et la visualisation thermique,
* Chart.js pour la génération de graphiques dynamiques (diagrammes circulaires, barres, courbes, etc.),
* Tailwind CSS pour le stylage rapide et responsive,
* FontAwesome pour l’affichage d’icônes personnalisées.

**Cartographie**

La carte interactive est basée sur Leaflet.js, avec des tuiles fournies par OpenStreetMap, une solution libre et gratuite pour l’affichage cartographique.

**API d’intelligence artificielle**

Le projet intègre l’API Mistral AI, utilisée pour la classification, le filtrage sémantique, la génération de tags et les embeddings.

**Base de données**

Les données sont stockées localement dans une base SQLite, avec possibilité d’extension vers d’autres SGBD supportés par Django, tels que PostgreSQL avec PostGIS pour les besoins en données spatiales.

**Autres outils**

Le projet utilise également cron (sous Linux) pour la planification des tâches, ainsi que Git pour le contrôle de version.

## CHAPITRE 7 : DEPLOIEMENT ET SIMULATION

### 7.1. PROCEDURE DE DEPLOIEMENT D’UNE ARCHITECTURE MICROSERVICE

Dans le cadre du développement de notre projet, nous mettons en place une API RESTful en utilisant **Django** et **Django REST Framework (DRF)** sous Windows. Ce rapport décrit les étapes suivies pour configurer l'environnement, créer le projet, et mettre en place une API basique.

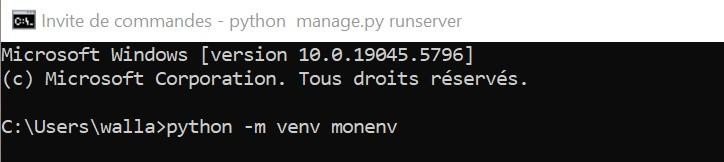
**Configuration de l’environnement Pré-requis**

* **Système d'exploitation** : Windows 10
* **Python** : Version 3.9+ (vérifiée via la commande python --version)
* **Gestionnaire de paquets** : Pip (inclus avec Python)

**Création d'un Environnement Virtuel**

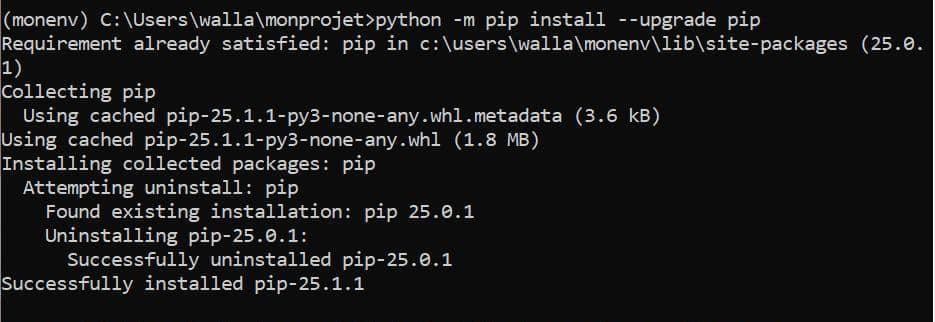
|  |
| --- |
| python -m venv monenv |

Pour isoler les dépendances du projet avec la commande :



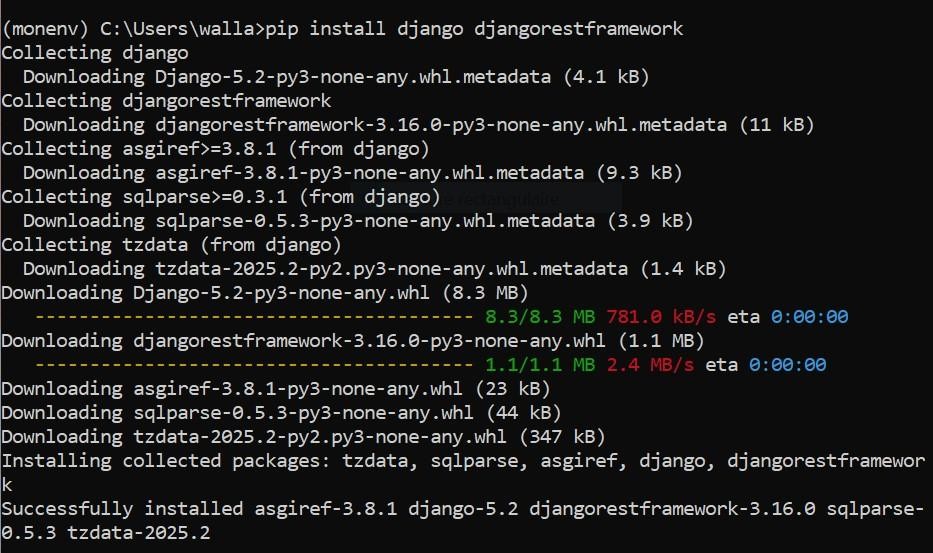
**Mise à jour du gestionnaire de paquets et installation des dépendances**

Le gestionnaire de paquets Pip a été mis à jour (de la version 25.0.1 à la version 25.1.1) avant l’installation des dépendances :



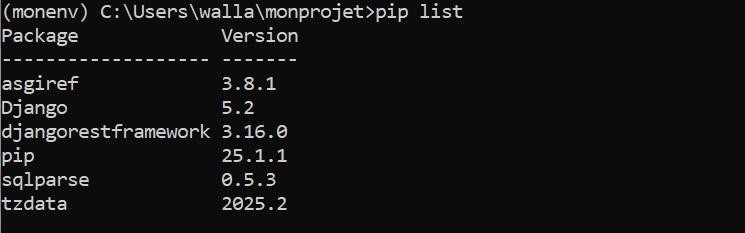
|  |
| --- |
| pip install django djangorestframework |

Les paquets suivants ont été installés : Django et DjangoRestFramework avec la commande



|  |
| --- |
| pip list |

La vérification de l’installation des paquets s’est fait avec la commande



**Cr**

**é**

**ation**

**du**

**Projet**

**Django**

U

n

nouveau

pr

o

jet

appel

é

mon

p

rojet

a

é

t

é

cr

éé

avec

la

commande

django

-

admin

s

tartproje

c

t

monproje

t

et

on

s

e

d

é

place

dans

le

dossier

du

projet

(

/

monprojet

)

pour

ex

é

cuter

les

commandes

q

ui

vont

suivre

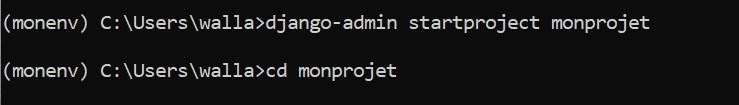
avec

la

commande

cd

monprojet



**Applications des migrations manquantes pour lancer le projet**

La commande migrate a **initialisé votre base de données** avec les tables nécessaires pour les fonctionnalités de base de Django. C’est une étape obligatoire pour tout nouveau projet .

Le super utilisateur est le **compte administrateur** de notre projet Django. Il est indispensable pour :

* **Gérer le backoffice** (ajout/suppression de données).
* **Configurer les droits d'accès**.
* **Debugger** en phase de développement.

6.

Lancement

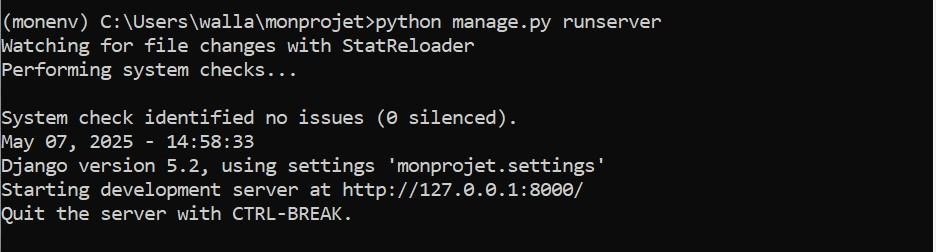
du

serveur



|  |
| --- |
| python manage.py runserver |

Le serveur est lancé avec la commande



Pour accéder à l’interface d’administration il faut entrer sur le navigateur

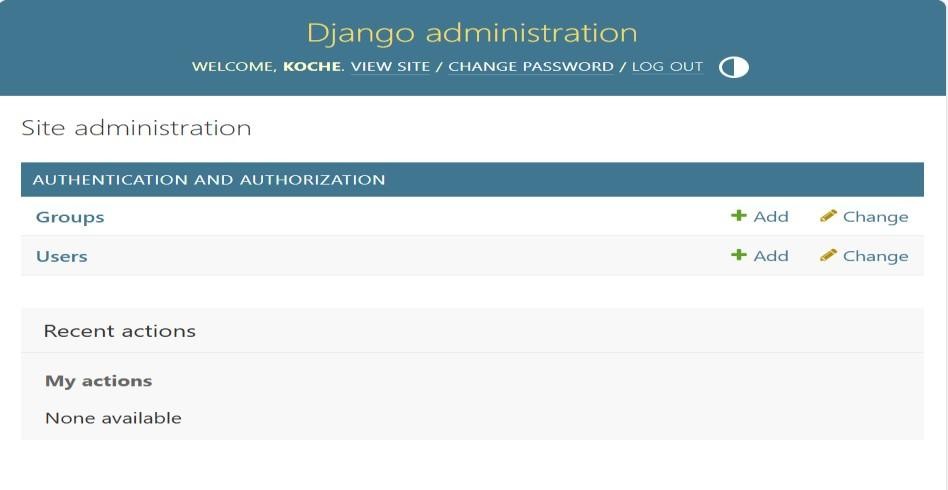
[http://127.0.0.1:8000/admi](http://127.0.0.1:8000/admin/)

[n](http://127.0.0.1:8000/admin/)

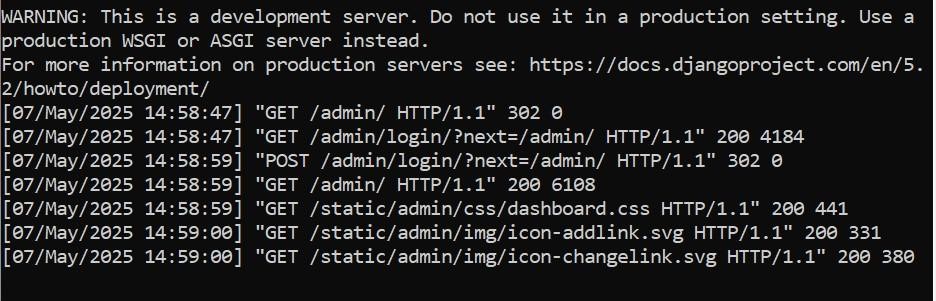
[/](http://127.0.0.1:8000/admin/)



Apres saisie des informations de connexion du super utilisateur créé plus haut nous sommes redirigés vers l’interface suivantes :



Les informations de navigation s’affichent en temps réel sur le CMD



**Création d'une API RESTful avec Djang**

Pour créer une API RESTful avec le framework Django. L'objectif est de mettre en place une architecture de base comprenant un projet Django nommé gdacs\_api et une application interne nommée events.

**Étapes de mise en place Étapes préliminaires :**

* **Se positionner dans le répertoire du projet avec la commande cd monprojet ;**
* **Activer l’environnement virtuel** afin d’isoler les dépendances d’un environnement préalablement créé avec la commande

o monenv\Scripts\activate ;

* **Créer le projet gdacs\_api** avec la commande django-admin startproject gdacs\_api ;

|  |
| --- |
| cd gdacs\_api |

* **Se déplacer dans le dossier du projet afin** de poursuivre les configurations avec la commande;

|  |  |
| --- | --- |
| python | manage.py |

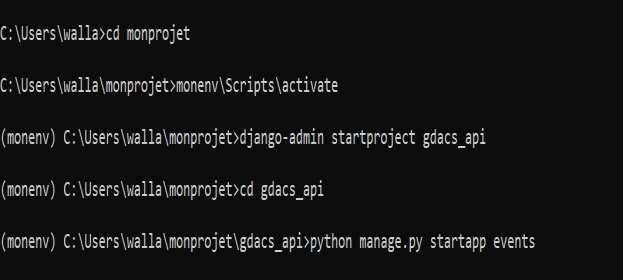
* **Création de l’application** events avec la commande

startapp

event

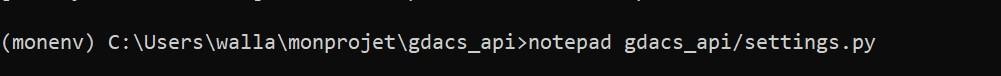
s

.



**Étape 1 : Configuration du fichier settings.py**

Dans le fichier gdacs\_api/settings.py ouvert avec la commande ci-dessous, nous avons effectué deux types de modifications :



1. **Ajout des applications installées**

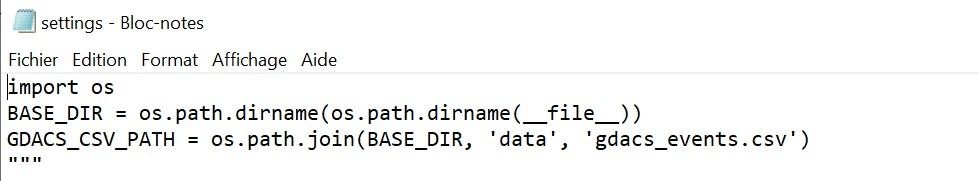
Nous avons ajouté les applications rest\_framework (de Django REST Framework) et events (l'application que nous avons créée) à la liste INSTALLED\_APPS :

Cela permet à Django de reconnaître et d’intégrer ces modules dans le projet.

1. **Définition du chemin d’accès vers le fichier CSV**

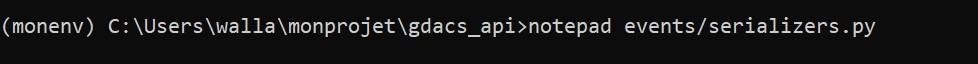
Nous avons ensuite défini une variable de configuration contenant le chemin vers le fichier CSV qui sera utilisé dans l’application events. Cette variable est définie en bas du fichier settings.py :

Cela permet de centraliser et de sécuriser l’accès au fichier de données, tout en facilitant sa réutilisation dans le code de l’application.



**Étape 2 : Création du fichier serializers.py**

Nous avons créé le fichier events/serializers.py car il n’existait pas puis nous avons défini un **serializer** pour transformer les objets de données en format JSON (et vice-versa).



Voici le contenu ajouté :



**Étape 3 : Création du fichier views.py**

Nous avons également créé le fichier le fichier events/views.py pour définir les **vues de l’API**.



Ces vues lisent les données depuis le fichier CSV, les convertissent en objets Python, puis les retournent sous forme de réponse JSON grâce au serializer.



Cette vue permet de **lister les événements** contenus dans un **fichier CSV**, via une **requête HTTP GET** sur l’URL /api/events/.

**Étape 4 : Configuration des routes URL**

1. **Configuration dans events/urls.py**

Nous avons d’abord ou modifié le fichier events/urls.py pour y enregistrer la vue EventListAPIView :



Cette configuration permet d’associer l’URL /events/ à la vue qui retourne la liste des événements au format JSON.

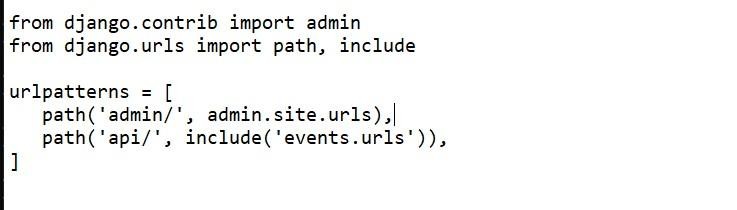


1. **Inclusion dans gdacs\_api/urls.py**

Ensuite, dans le fichier principal gdacs\_api/urls.py, nous avons inclus les URLs de l’application events.



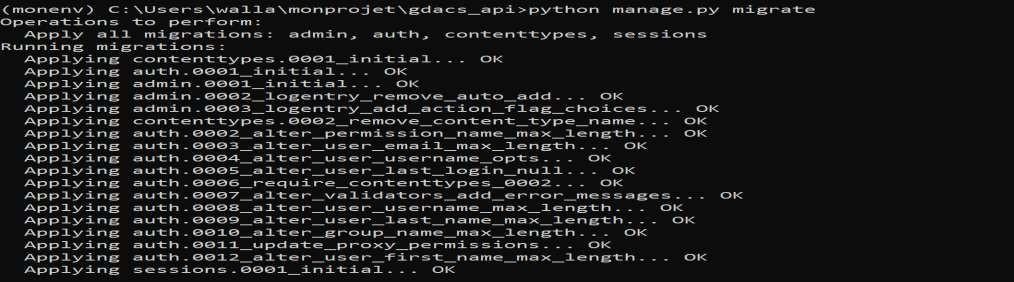
Ainsi, la vue sera accessible à l’URL **/api/events/**, ce qui respecte les bonnes pratiques REST pour nommer les points d'accès API.



**Étape 5 : Exécution du projet et test de l’API**

1. **Application des migrations**

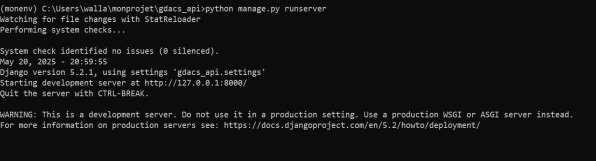
Même si notre API lit les données depuis un fichier CSV (et donc n’a pas forcément besoin d’un modèle en base de données), nous avons tout de même effectué les migrations nécessaires pour initialiser les composants par défaut de Django :



Cela permet notamment de préparer l’interface d’administration et d’assurer que la base de données est fonctionnelle.

1. **Démarrage du serveur de développement**

Nous avons ensuite lancé le serveur Django avec la commande suivante :



Le serveur a démarré avec succès et était accessible à l'adresse :

http://127.0.0.1:8000/

1. **Test de l’API dans le navigateur**

Nous avons saisi l’URL suivante dans le navigateur pour tester notre API : <http://127.0.0.1:8000/api/events/>

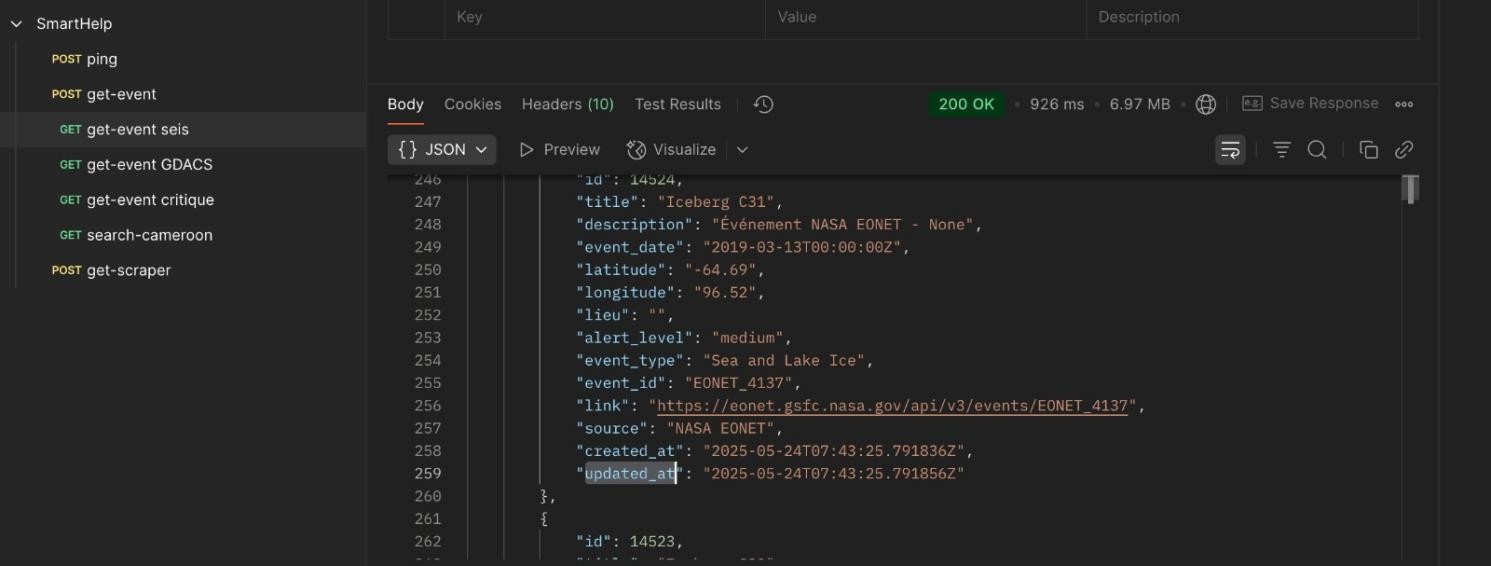


Cela a retourné une **réponse JSON contenant la liste des événements** lus à partir du fichier CSV, confirmant que :

* Le fichier CSV a bien été trouvé et lu,
* Les données ont été correctement sérialisées,
* Le point d’accès /api/events/ fonctionne parfaitement.



**Test de l’API sur Postman**

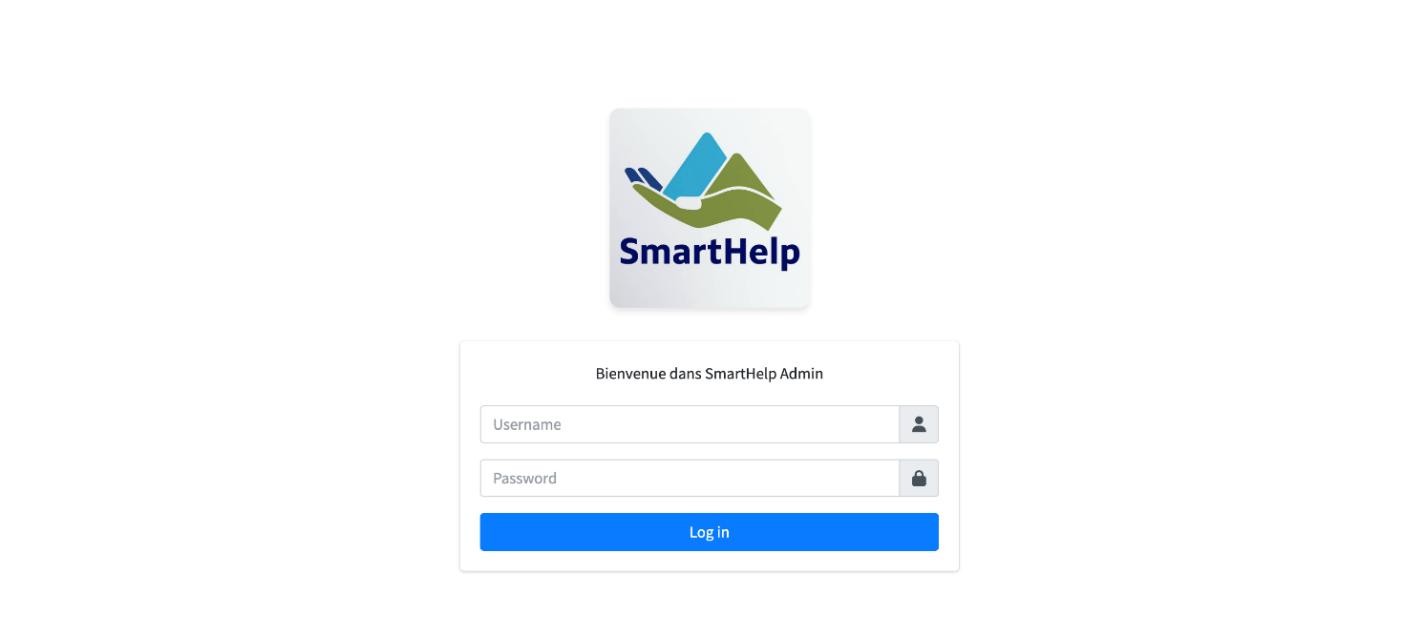


***Figure 28 : Exemple de requête pour tester l’API de récupération des données scrappées***

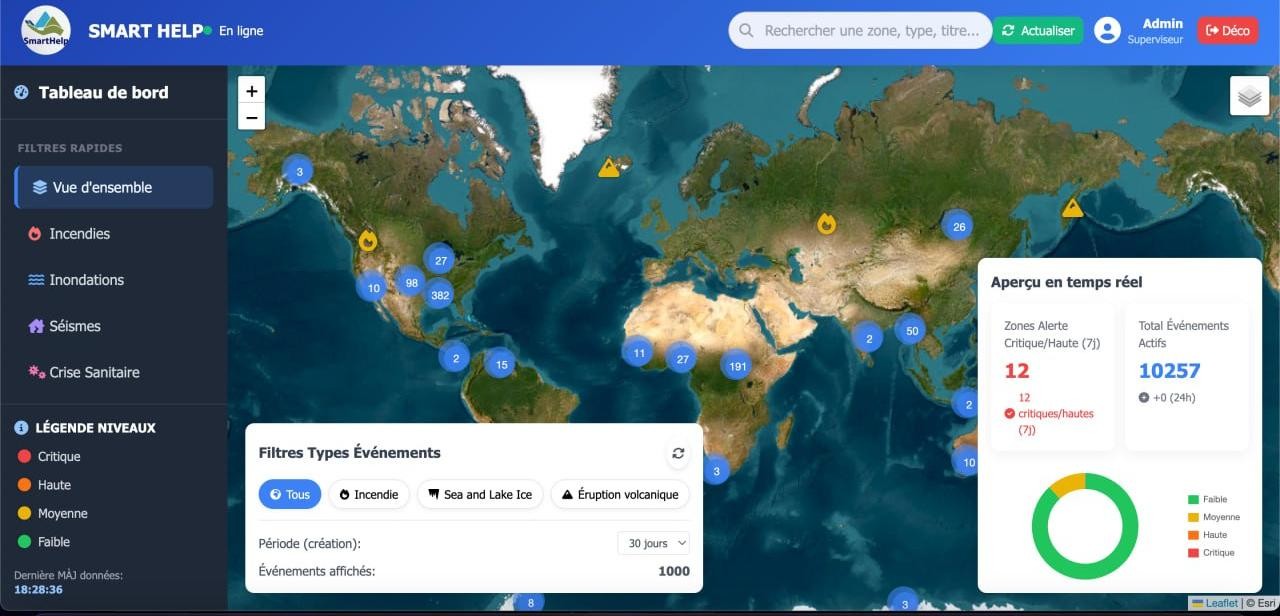
### 7.2 CREATION DE LA PLATEFORME

Dans le cadre de notre projet, cette plateforme vise plus à surveillance des zones enclavées en temps réel, afin de s’assurer de la véracité des faits en cas de catastrophes et même de continuer la surveillance des zones même en situation des défaillances des dispositifs sur sites. Et grâce à l’IA utilisée nous avons un rendu de l’évolution de l’occurrence des catastrophes sur l’étendue du territoire national. C’est ainsi les images des interfaces de la plateforme sont les suivantes :

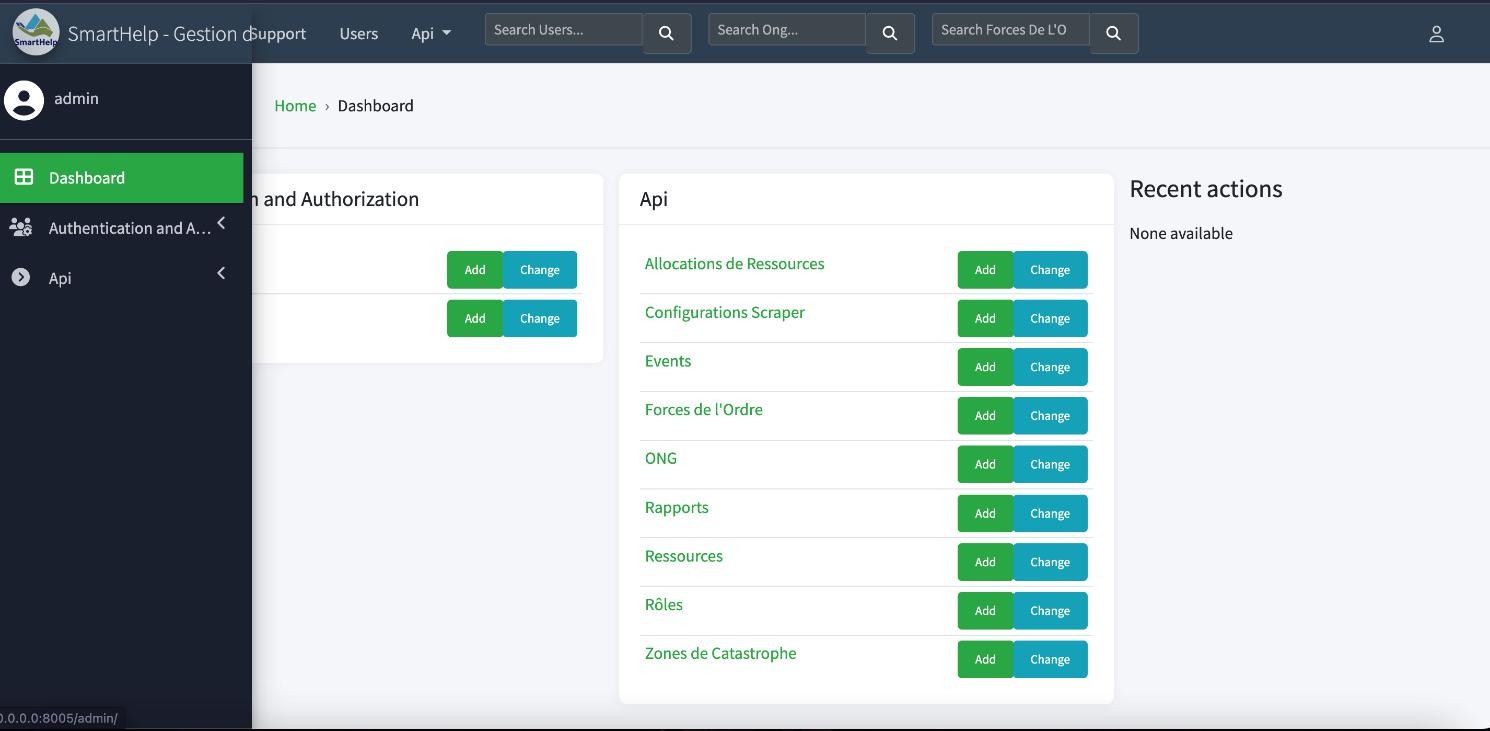
**7.2.1 Interfaces côté Administrateur**



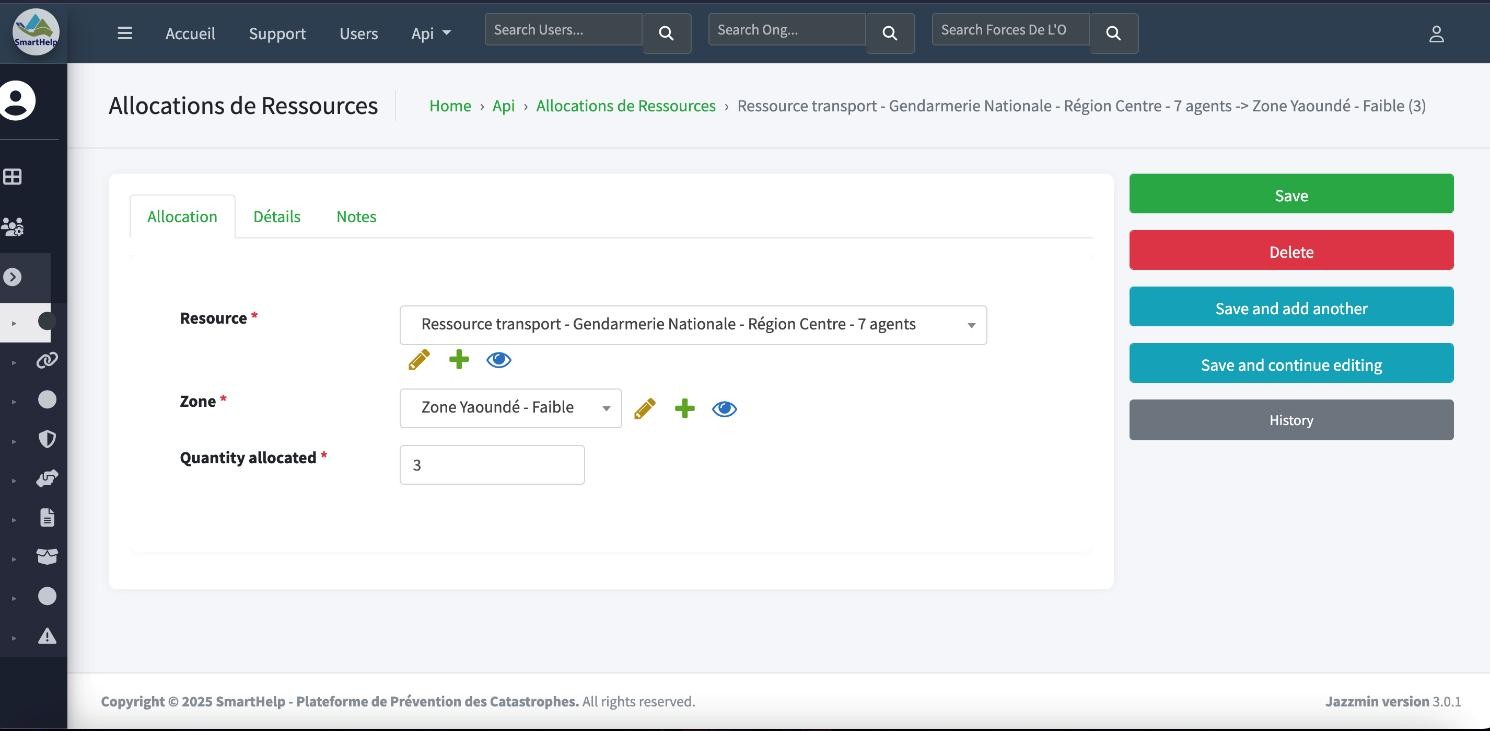
#### Figure 29 : Login de l’interface Administrateur de SmartHelp (Source : SmartHelp)



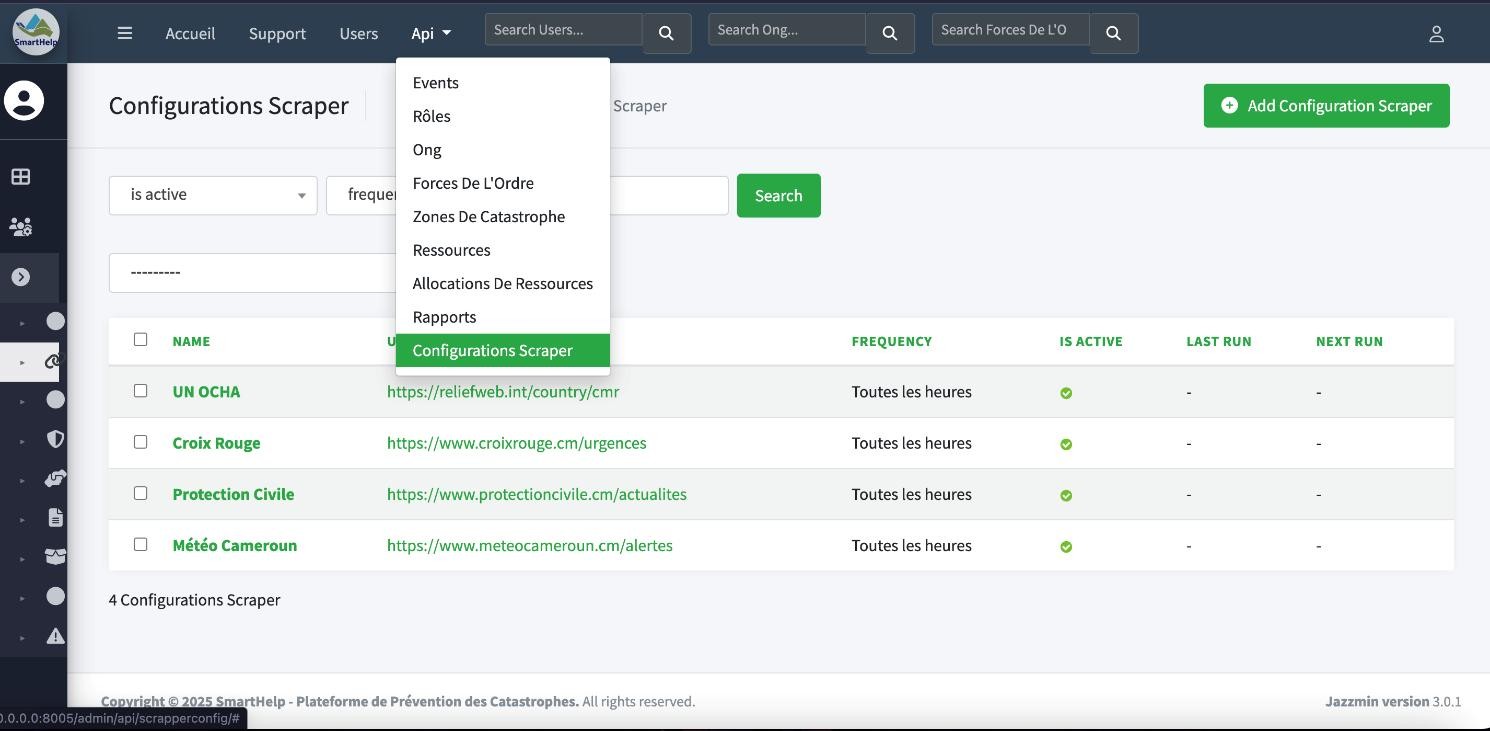
#### Figure 30 : Interface de visualisation des zones aynat eu une catastrophe



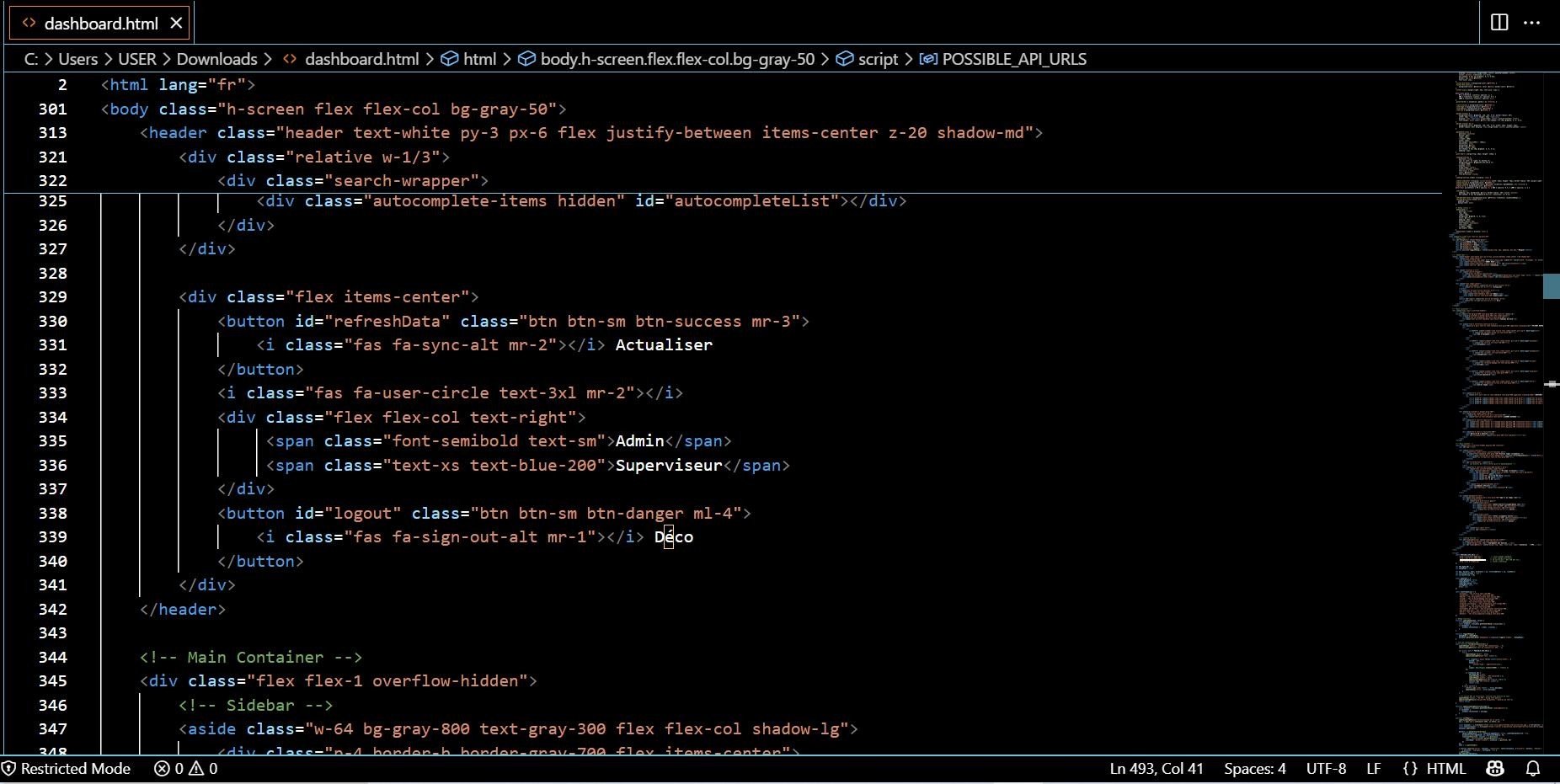
#### Figure 31 : Interface du Dashboard de configuration des interfaces utilisateurs



#### Figure 32 : Interface d’enregistrement des ressources disponibles

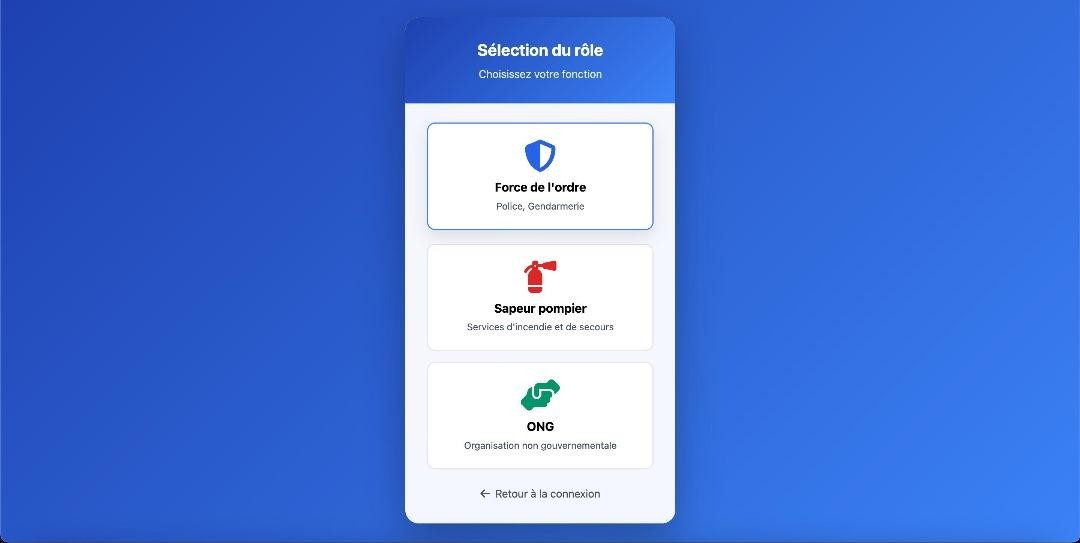


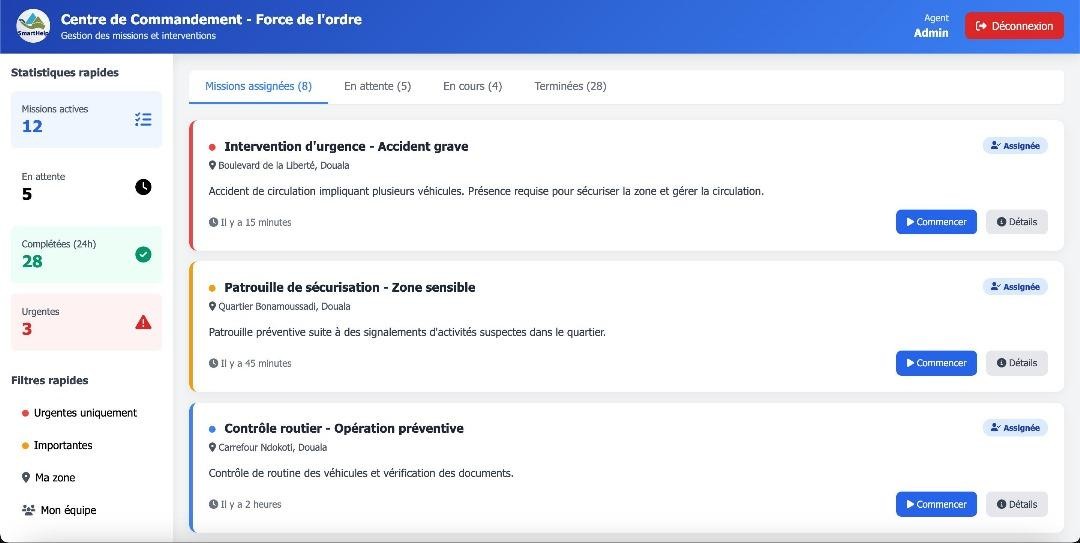
#### Figure 33 : Interface de visualisation des données Scrappées



#### Figure 34 : aperçu du code de la plateforme sur Visual Studio Code

**7.2.2 Interfaces côté Utilisateur**





***Figure 35 : Dashboard de suivie des operations***

### 7.3 MONTAGE DES CIRCUITS ET DES DISPOSITIFS

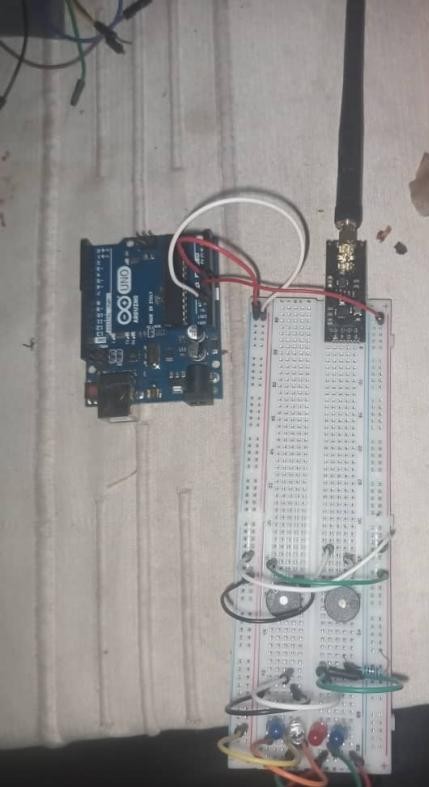
Concernant la conception du dispositif, on a les images suivantes des circuits et des dispositifs :

**7.3.1 Dispositif à installer dans les zones enclavées**



#### Figure 36 : aperçu du câblage du dispositif émetteur

**7.3.2 Dispositif émetteur de signal en cas de catastrophe**



#### Figure 37 : aperçu du câblage du dispositif recepteur

**7.3.3 Communication et interconnexion des dispositifs**

La mise en place d’un système d’alerte rapide pour signaler des sinistres dans des zones enclavées impose des exigences strictes en matière de connectivité : fiabilité, faible latence, sécurité et couverture géographique étendue. Le **réseau MPLS (Multiprotocol Label Switching)** l’un d’élément constitutif de l’épine dorsale de ce projet. Il assure l’acheminement sécurisé et priorisé des signaux d’alerte depuis les dispositifs sur le terrain jusqu’au centre de traitement et de contrôle que nous sommes, pour une intervention très rapide.

**Le rôle central du MPLS dans le système d’alerte**

Le MPLS n’est pas un simple moyen de transport de données : c’est un **réseau intelligent**, capable de **diriger les paquets en fonction d’étiquettes (labels)** plutôt qu’en s’appuyant uniquement sur les adresses IP. Cette approche permet un **routage plus rapide, plus sécurisé, et mieux contrôlé**.

Dans notre système :

* Lorsqu’un utilisateur appuie sur le bouton d’alerte, un **signal encapsulé** est immédiatement généré par le dispositif.
* Ce signal est envoyé vers une **passerelle réseau** équipée pour l’interconnexion MPLS.
* Dès lors, le paquet circule **au sein du réseau MPLS/VPN de l’opérateur**, selon un chemin prédéfini (Label-Switched Path – LSP) avec une **classe de service prioritaire**.

Ce mécanisme garantit que **l’information critique ne subit aucun ralentissement**, même en cas de congestion réseau.

**Avantages stratégiques du MPLS pour notre projet**

**Qualité de service (QoS) garantie :**Le MPLS permet de marquer les paquets d’alerte comme étant de haute priorité, assurant ainsi une latence minimale.Les alertes ne sont pas noyées dans le trafic internet classique, elles sont traitées à part, avec des garanties de délai.

**Sécurité par isolation (VPN MPLS) :**Le trafic d’alerte circule dans un réseau privé virtuel (VPN MPLS), totalement séparé d’Internet.Aucun accès public : seuls les équipements autorisés du réseau peuvent émettre ou recevoir les données.

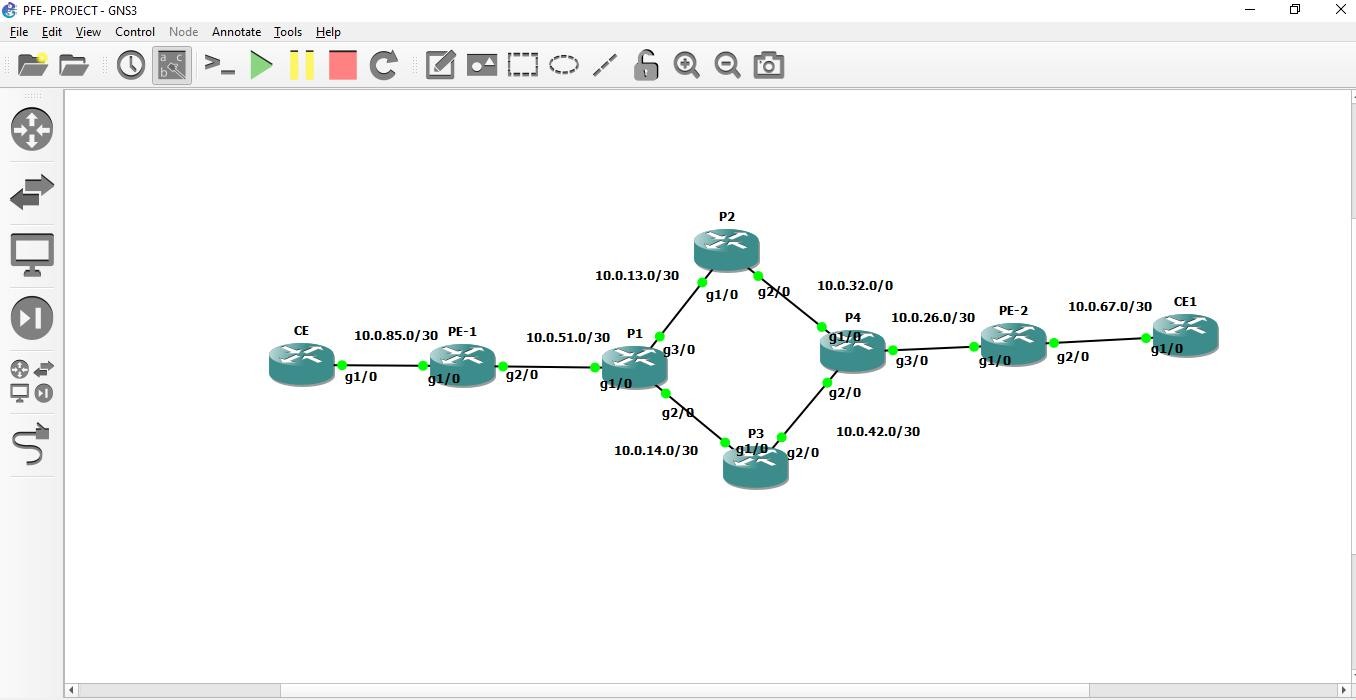
**Fiabilité et résilience :**En cas de défaillance d’un nœud, le MPLS redirige automatiquement les paquets via une route alternative sans intervention humaine.Cela assure une disponibilité élevée, indispensable dans des contextes d’urgence.

**Facilité d’interconnexion multisites :**Le MPLS permet de connecter les zones enclavées, le centre de supervision, via un réseau maillé cohérent, peu importe leur éloignement géographique.

**Architecture technique avec MPLS**

**Étapes de transmission du signal d’alerte via MPLS :**

* **Événement déclencheur** : Appui sur le bouton du dispositif local.
* **Encapsulation** : Le signal est encapsulé avec des métadonnées ( localisation, type de sinistreà travers les LEDs au niveau du recepteur ).
* **Entrée MPLS** : La passerelle SMS/IP injecte le paquet dans le VPN MPLS via le routeur opérateur.
* **Transit MPLS** : Le signal circule de manière prioritaire dans le backbone MPLS selon un chemin optimisé.
* **Réception locale** : Arrivée dans le centre de traitement où les données sont vérifiées et analysées.
* **Diffusion sécurisée** : Transmission vers le centre de contrôle puis retransmis les différents intervenants en fonction de la catastrophe via API ou interfaces utilisateurs sécurisées, toujours via la plateforme de contrôle et de supervision.



#### Figure 38: exemple de présentation de l’architecture de la configuration du reseaux mpls/vpn sur le simulateur GNS3

Une étude comparative des offres MPLS/VPN de MTN Cameroun et Camtel, basée sur les informations disponibles et les données récentes du marché camerounais.

**Étude comparative des offres MPLS/VPN de MTN Cameroun et Camtel**

**1. Présentation des opérateurs**

* **MTN Cameroun** : Deuxième acteur télécommunication du Cameroun en nombre d’abonnés , MTN dispose d’une infrastructure étendue couvrant 97 % de la population en 2G et 90 % en 3G, avec des projets 5G en préparation. MTN est reconnu pour ses investissements importants (plus de 62 milliards FCFA prévus en 2025) visant à renforcer la qualité et la capacité de son réseau ;
* **Camtel** : Opérateur historique, Camtel propose des solutions MPLS/VPN sur une infrastructure fibre optique nationale. Camtel bénéficie d’un partenariat avec MTN pour l’accès à l’infrastructure réseau via un accord de roaming national, ce qui améliore sa couverture et sa qualité de service.

**2.Offres MPLS/VPN : caractéristiques principales**

#### Tableau 38 : Caractéristiques principales

|  |  |  |  |
| --- | --- | --- | --- |
| Critère | MTN Cameroun |  | Camtel |
| **Technologie** | MPLS VPN Layer 2 et Layer  optique, micro-ondes | 3, fibre | MPLS VPN Layer 2, fibre  optique |
| **bande passante**  **pour la liason VPN**  **MPLS** | De 1 Mbps à 11 Mbps et plus |  | De 2 Mbps à 155 Mbps |
| **Tarification indicative**  **pour la co-**  **location** | De 68 000 fcfa 75 000 fcfa  100000 fcfa | jusqu ’ a | À partir de 175 000 fcfa |
| **SLA et**  **garanties** | SLA avec disponibilité >98 %, technique dédié | support | SLA variable, support technique selon contrat |
| **Sécurité** | Connexions cryptées, gestion centralisée, outils de monitoring | | Connexions sécurisées MPLS,  gestion centralisée |
| **Couverture réseau** | Très étendue grâce à infrastructure propre et roaming avec Camtel | | Couverture améliorée via partenariat avec MTN |

#### (Source : Les Promoteurs)

1. **Performances réseau et qualité de service**
   * **MTN** affiche un débit moyen 4G descendant de 16,7 Mb/s en 2025, avec une qualité variable selon les zones, notamment des saturations en zones urbaines.
   * **Camtel** a un débit moyen 4G plus faible (6,3 Mb/s), mais bénéficie d’une fibre optique nationale et d’une amélioration de la couverture grâce à l’accord avec MTN.
2. **Avantages tarifaires et économiques**
   * **MTN** propose une tarification plus élevée, justifiée par une infrastructure plus robuste, des SLA stricts et un support technique avancé. Les coûts unitaires par Mbps diminuent avec l’augmentation du débit, intéressant les grandes entreprises et multisites.
   * **Camtel** offre des tarifs plus compétitifs pour les débits faibles à moyens, adaptés aux PME et aux sites avec des besoins limités. Le partenariat avec MTN permet d’améliorer la qualité tout en conservant des prix attractifs .
3. **Points forts et limites**

#### Tableau 39 : Points forts et limites

|  |  |  |
| --- | --- | --- |
| Opérateur | Points forts | Limites |
| **MTN** | * Large couverture et infrastructure moderne * SLA et support technique avancés - Offre évolutive jusqu’à très haut débit * Projets 5G en cours | * Coût plus élevé * Saturation réseau en zones urbaines * Débit moyen 4G en baisse   récente |
| **Camtel** | * Tarifs compétitifs pour PME * Fibre optique nationale * Partenariat avec MTN pour   meilleure couverture   * Solutions adaptées aux besoins locaux | * Débit moyen inférieur * SLA et supports variables - Moins d ’ investissements   récents visibles |

L’intégration du **réseau MPLS/VPN** dans notre dispositif d’alerte constitue un choix stratégique fondamental. Il garantit une **transmission rapide, sécurisée et fiable** des alertes, dans un contexte où chaque seconde compte. Grâce à cette infrastructure, les intervenants en cas de sinistre et partenaires sont assurés de recevoir les informations en temps réel, avec une précision et une fiabilité qui peuvent faire la différence entre la vie et la mort sur le terrain. **Pour** besoins importants en bande passante ,une qualité de service élévée et un rapport technique dedié dans le contexte de notre solution, MTN cameroun est la solution la plus adaptée, une qualité de service élevée et un support technique dédié, **MTN Cameroun** est la solution la plus adaptée

1. **Catastrophe:** Pertubation grave affectant le fonctionnement d’une communauté qui depasse sa capacité à y faire face utilisqnt ses propres ressources. [↑](#footnote-ref-1)
2. **Zone enclavée :** territoire geographiquement isolé; sans accés et/ou à la communication difficile et voie de transport degradée [↑](#footnote-ref-2)