# Rapport de Stage

Migration du moteur cartographique de django-mapentity vers MapLibre GL JS dans une démarche de modernisation de Geotrek-admin

Étudiant : KAYA RHABBY Pascia Hershe

Tuteur professionnel : M. JEAN ETIENNE CASTAGNEDE

Tuteur académique : Pierre LUDMANN

07/06/2025

# Sommaire

# Liste des Figures



# Liste des Tableaux

# Glossaire

| Termes | Définitions |
|---|---|
| SIG | Systèmes d'Information Géographique. Outils informatiques permettant la capture, le stockage, l'analyse et la visualisation de données géolocalisées. |
| PME | Une PME (Petite et Moyenne Entreprise) est une entreprise dont la taille, en termes d'effectifs et de chiffre d'affaires, se situe entre celle des microentreprises (très petites structures) et des grandes entreprises. |
| Django | Framework web Python basé sur le modèle MVT (Modèle, Vue, Template), utilisé pour développer des applications web robustes et structurées. |
| GeoDjango | Extension de Django (le framework web Python) conçue pour faciliter le développement d'applications web géospatiales. |
| PostgreSQL | Système de gestion de base de données relationnelle libre et open source, réputé pour sa robustesse, sa conformité aux standards SQL, et ses fonctionnalités avancées. |
| PostGIS | Extension spatiale pour le système de gestion de base de données relationnelle PostgreSQL. |
| Leaflet JS | Bibliothèque JavaScript open source légère et très populaire pour créer des cartes interactives sur le web. |
| MAPBOX | Plateforme de cartographie et de localisation qui fournit des outils et des services pour créer des cartes personnalisées et intégrer des fonctionnalités géospatiales dans des applications web et mobiles. |
| Maplibre GL JS | Bibliothèque JavaScript open source permettant d'afficher des cartes interactives vectorielles directement dans un navigateur web. |
| GitLab | Plateforme web de gestion du code source et de collaboration DevOps, construite autour de l'outil de gestion de versions Git. |
| API REST | API REST signifie Application Programming Interface utilisant le style architectural REST (Representational State Transfer). |
| GeoJSON | Format de fichier basé sur JSON, utilisé pour représenter des données géographiques (points, lignes, polygones) de manière structurée et lisible. |

| | |
|---|---|
| CSV | Format de fichier texte simple, utilisé pour représenter des données tabulaires (comme un tableau ou une feuille Excel). |
| GPX | Format de fichier XML standardisé pour stocker et échanger des données géographiques. |
| Shapefile | Format pour stocker et échanger des données géographiques vectorielles (points, lignes, polygones). |
| KML | Format basé sur XML utilisé pour représenter et échanger des données géographiques (points, lignes, polygones, images superposées, etc.). |
| OpenStreetMap | Base de données cartographique libre et collaborative, alimentée par des contributeurs du monde entier. |
| DOM | Le DOM (Document Object Model) est une interface de programmation qui représente la structure d'un document HTML ou XML sous forme d'un arbre d'objets. Chaque élément (balise, texte, attribut, etc.) du document devient un nœud dans cet arbre, que l'on peut manipuler via des langages comme JavaScript. |
| SVG | SVG (Scalable Vector Graphics ) est un format de fichier et une technologie basée sur le XML pour décrire des images vectorielles. |
| GPU | Graphics Processing Unit (processeur graphique) |
| HTML | HyperText Markup Language, c'est le langage de balisage standard utilisé pour créer et structurer les pages web et leur contenu. |
| PNG | Portable Network Graphic, c'est un format de fichier d'image numérique conçu pour stocker des images bitmap avec ou sans transparence, tout en assurant une compression sans perte |
| bbox | Bounding Box (boîte englobante). Rectangle défini par deux points opposés, utilisé pour restreindre une zone d'analyse géographique |

# PRÉSENTATION DE L'ENTREPRISE : MAKINA CORPUS

## I - Présentation générale

Makina Corpus est une société de services en ingénierie logicielle (SSIL) spécialisée dans le développement d'applications web et mobiles innovantes, exclusivement basées sur des logiciels libres. Elle conçoit, développe et intègre des applications métiers, des portails complexes, ainsi que des solutions SIG avec traitement d'informations spatiales et cartographie interactive.

Elle propose une offre complète : conception, architecture, développement, gestion de projets, maintenance, hébergement, audits techniques, formation, ergonomie et design graphique. L'accent est mis sur la facilité d'usage, l'ergonomie et l'intuitivité des interfaces, afin de fournir des solutions à la fois puissantes et accessibles.

Implantée en France, essentiellement à Toulouse et Nantes, l'entreprise emploie entre 40 et 50 salariés selon les années. Elle travaille pour des grands comptes (Orange, Airbus, La Poste, Météo France, SNCF, Ministère de l'Éducation Nationale), mais aussi pour de nombreuses collectivités territoriales, PME et startups.

Makina Corpus est fortement positionnée auprès des acteurs du développement durable, des projets territoriaux et de la gestion environnementale, avec l'ambition de devenir prestataire de référence en éco-informatique.

L'entreprise est un promoteur engagé du logiciel libre, tant sur le plan technique que philosophique. Elle contribue activement à de nombreux projets open source par exemple :

- la librairie django-leaflet[1], téléchargée plus de 12 000 fois,
- Les projets Geotrek[2]
- Et d'autres projets disponibles sur le GitHub de l'entreprise[3]

Elle encadre également des étudiants dans le cadre du Google Summer of Code (GSoC) et partage son expertise à travers de nombreux articles de blog techniques sur makina-corpus.com/blog/metier[4].

[1] https://github.com/makinacorpus/django-leaflet/
[2] https://github.com/GeotrekCE
[3] https://github.com/makinacorpus/
[4] https://makina-corpus.com/blog/

## I.a. - Méthodes de travail

Makina Corpus met en œuvre des méthodes agiles concrètes et efficaces comme Scrum et Kanban, basées sur la collaboration, la transparence et l'adaptabilité. Les échanges réguliers entre les clients et les équipes (daily meetings, revues hebdomadaires) permettent une correction rapide des problèmes et une réponse adaptée aux besoins réels.

Le travail collaboratif est un pilier fort de l'entreprise :

- Les projets sont toujours portés par des équipes et non des individus.
- Le pair programming et les revues de code sont systématiques.
- En cas de blocage, les développeurs peuvent s'appuyer sur leur équipe ou consulter les experts référents de l'entreprise.

## I.b. - Recherche et innovation

Makina Corpus investit plus de 15 % de son chiffre d'affaires en recherche et développement, avec deux docteurs et plusieurs ingénieurs impliqués dans des programmes R&D. L'entreprise collabore avec des laboratoires publics, dont :

- Laboratoire IHM Elipse de l'IRIT (Institut de Recherche en Informatique de Toulouse)
- Co-Design Lab de Telecom ParisTech
- Unité Mixte de Recherche Environnement-Ville-Société du CNRS
- Unité Mixte de Recherche EcoLab – Laboratoire d'écologie fonctionnelle du CNRS, de l'Université Paul Sabatier et de l'Institut National Polytechnique de Toulouse, etc.

Projets de recherche notables :

- Projet Accessimap, co-financé par l'Agence Nationale de Recherche (ANR), visant à concevoir une plate-forme numérique permettant aux déficients visuels de se représenter et de s'approprier des concepts spatiaux (espaces, itinéraires…). Le consortium réunit une équipe de recherche en informatique spécialisée en interactions et technologies d'assistance pour Déficients Visuels (IRITELIPSE), une équipe de recherche en sciences humaines et sociales spécialiste des pratiques de design en relation avec les nouveaux objets numériques (Telecom ParisTech, Co-Design Lab), un centre d'éducation spécialisée (CESDV-IJA) et Makina Corpus.
- Le projet de recherche cartographique GéoTopia avec l'Unité Mixte de Recherche Environnement-Ville-Société du CNRS, labellisé au sein du TGE Adonis (très grande infrastructure de recherche).

Makina Corpus réalise un chiffre d'affaire d'environ 3M€ comme le témoigne le tableau ci-après :

| Année | Chiffre d'affaires | Effectifs |
|---|---|---|
| 2022 | 3 763 k€ | 48 |
| 2021 | 4 342 k€ | 53 |
| 2020 | 3 940 k€ | 49 |

*Tableau 1 : Synthèse des éléments financiers relatifs à Makina Corpus*

# II - Présentation de L'équipe Geotrek

L'équipe en charge du développement de Geotrek-admin se compose actuellement de dix membres. Elle comprend six développeurs, un chef de projet, un chargé de support ainsi que deux stagiaires, dont moi-même.

L'équipe applique une méthodologie agile fondée sur le cadre Scrum, structurée autour de rituels réguliers qui assurent un suivi fluide et efficace du développement.

Chaque lundi matin, une réunion hebdomadaire (souvent appelée hebdo) est organisée afin de passer en revue l'ensemble des tickets de développement figurant sur le tableau GitLab. Cette réunion permet de faire le point sur les tâches terminées (done), de planifier les tâches à venir (next, todo), d'évaluer les priorités et d'identifier d'éventuels blocages. Ce suivi visuel via GitLab facilite la gestion collaborative du projet et améliore la coordination entre les différents membres de l'équipe.

En complément, une réunion courte, appelée « daily », se tient chaque après-midi. Lors de ce point rapide, chacun partage l'état d'avancement de ses tâches et signale tout problème technique rencontré. En cas de blocage persistant, une réunion technique spécifique ou un échange en présentiel est immédiatement organisé afin de débloquer la situation au plus vite, sans freiner le reste de l'équipe.

Un suivi spécifique est également prévu pour les stagiaires, chacun ayant un créneau adapté à son organisation. En ce qui me concerne, une réunion quotidienne est organisée chaque matin, du mardi au vendredi à 10h45 et lundi 10h00. Ces moments permettent de faire le point sur les fonctionnalités développées, d'échanger sur celles à venir, ainsi que de remonter les éventuelles difficultés rencontrées.

Ces pratiques agiles favorisent une communication constante, renforcent l'entraide entre les membres de l'équipe et garantissent une visibilité claire sur l'avancement du projet.

# INTRODUCTION

Les Systèmes d'Information Géographique (SIG) sont des logiciels permettant aux institutions publiques, telles que les parcs naturels nationaux ou régionaux, les conseils départementaux, les communautés de communes ou encore les régions, de mieux organiser la gestion et la maintenance de leur territoire. Ces outils offrent une vision globale et précise de l'état des infrastructures (passerelles, bancs, signalétiques, etc.) en les localisant et en renseignant leur état. Cette capacité de suivi facilite la planification des interventions et contribue à répondre à des enjeux variés, allant de la sécurité à la valorisation touristique, en passant par l'accessibilité des sites. Cependant, les SIG tels que QGIS ou ArcGIS restent complexes à prendre en main. La richesse fonctionnelle de ces outils, bien que précieuse, exige une certaine expertise technique. Leur utilisation nécessite souvent une formation approfondie, voire des compétences spécifiques, notamment en SQL. Ce niveau d'exigence constitue un frein pour les agents non spécialistes qui ont pourtant besoin de manipuler des données géographiques dans leur travail quotidien.

Dans cette optique, en 2012, le Parc national des Écrins, le Parc national du Mercantour et le Parco delle Alpi Marittime ont lancé un appel d'offres pour la création d'un outil plus accessible. L'objectif étant de concevoir une solution à la fois intuitive et adaptée aux besoins concrets de gestion et de valorisation du territoire. C'est ainsi qu'est née Geotrek, une suite applicative open source composée de quatre modules. Geotrek-admin étant Le cœur du dispositif destiné aux gestionnaires de territoire. Développée par Makina Corpus, Geotrek-admin s'appuie sur un socle technologique robuste (Django, GeoDjango, PostgreSQL/PostGIS) et propose une interface cartographique interactive à travers la librairie django-mapentity, pensée pour faciliter la manipulation des objets géographiques.

Cependant, cette interface repose encore sur Leaflet 0.7, une version désormais obsolète qui montre ses limites en termes de performances, notamment lorsqu'il s'agit d'afficher ou d'interagir avec un grand volume de données géospatiales. Cette contrainte impacte directement l'expérience utilisateur, en particulier dans les contextes à forte densité d'information géographique. Dans l'objectif de moderniser cette brique essentielle de l'application, une migration vers MapLibre GL JS a été engagée. Cette technologie moderne, basée sur WebGL, permet un rendu cartographique accéléré par le GPU, compatible avec l'affichage de tuiles vectorielles, et plus adapté aux exigences actuelles en matière de fluidité et de réactivité. Le stage s'inscrit dans cette démarche de transition technologique et a pour objectifs de :

1. Remplacer Leaflet par MapLibre GL JS dans la librairie django-mapentity ;
2. Refondre progressivement le code JavaScript pour l'aligner avec des pratiques modernes, en s'éloignant notamment de jQuery.
3. Adapter l'intégration cartographique au sein de Geotrek-admin en tirant parti de cette nouvelle base ;

Ce travail de migration et de modernisation pose les bases d'une nouvelle génération de l'interface cartographique de Geotrek-admin, plus performante, évolutive et adaptée aux besoins actuels des utilisateurs terrain.

# III - ÉTAT DE L'ART ET ANALYSE TECHNIQUE INITIALE

## III.a. - Les Systèmes d'informations géographiques (SIG )

Un système d'information géographique est un système informatique permettant de stocker, gérer, analyser et visualiser des données géospatiales. L'une de ses principales caractéristiques est la superposition de couches d'information géographique : chaque couche représente un type de donnée spécifique (routes, bâtiments, zones naturelles, réseaux, etc.) et peut être affichée, combinée ou interrogée pour produire une lecture enrichie du territoire. Cette approche multicouche permet de croiser plusieurs sources de données et d'obtenir une vision plus complète et précise d'un espace donné.

L'illustration suivante montre un exemple de superposition de couches dans un SIG, où différentes entités géographiques sont combinées pour produire une carte analytique.

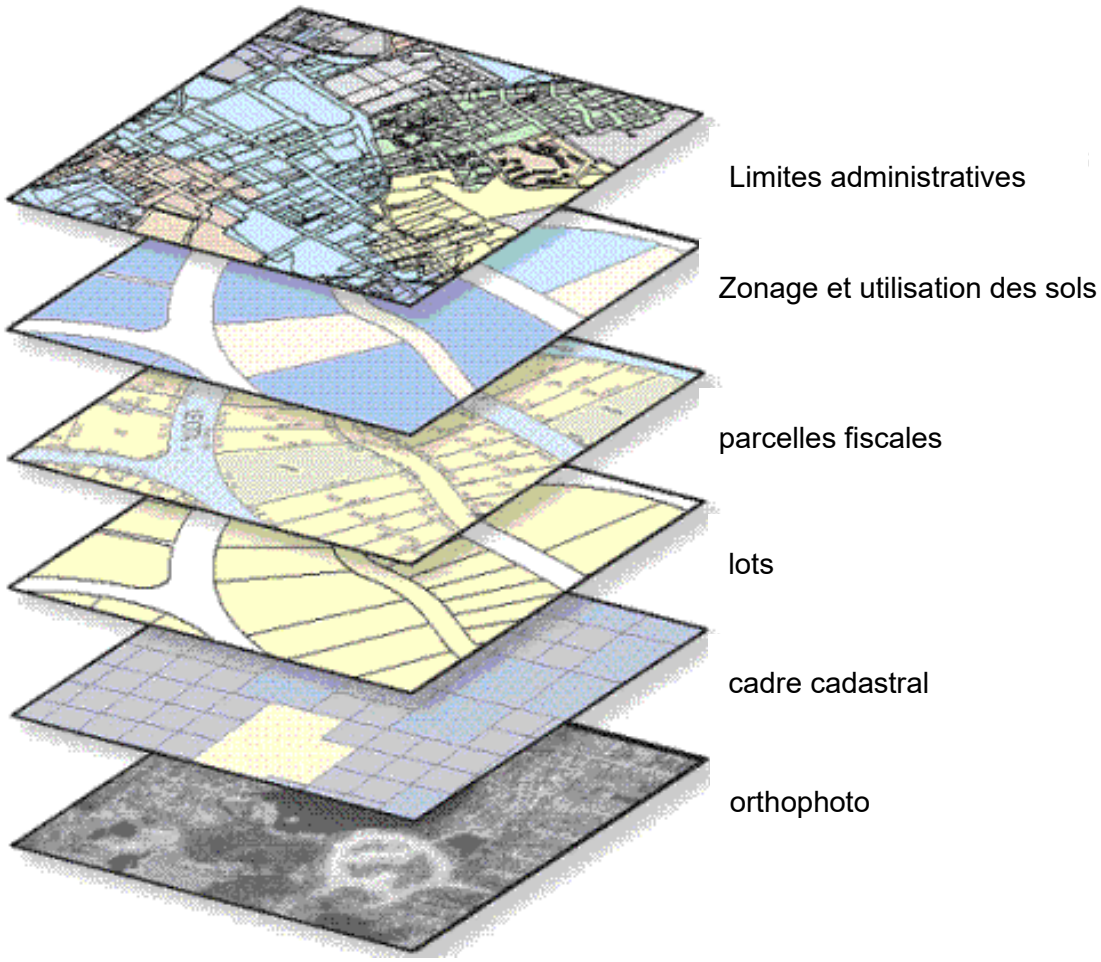


*Figure 1 : Principe de superposition de couche dans un SIG*

## III.b. - Rôles d'un Système d'Information Géographique (SIG)

Les systèmes d'information géographique vont bien au-delà de la simple représentation de données spatiales sur une carte. Ils constituent de véritables outils d'analyse, d'aide à la décision et de communication, capables de transformer des données géographiques brutes en informations pertinentes et exploitables.

Un SIG remplit ainsi plusieurs fonctions essentielles, parmi lesquelles :

- **Gestion des données :** Un SIG est un système fondamental d'enregistrement. On peut stocker et intégrer des informations à partir des systèmes professionnels et de sources faisant autorité, amplifiant ainsi l'utilité des données.

- **Cartographie et visualisation :** De cartes et tableaux de bord numériques à l'imagerie satellite, à la technologie 3D et au SIG en temps réel, un SIG donne vie aux données, en nous aidant à mieux saisir les problématiques et à les résoudre.

- **Analyse spatiale :** La plupart des données ont une composante de localisation. Avec les outils d'analyse spatiale, nous trouvons des relations cachées et générons de nouvelles informations des données.

- **Communication :** Les cartes et les tableaux de bord communiquent rapidement des idées complexes. La science et les données élaborent une conception commune, en privilégiant la collaboration et la résolution des problèmes.

## III.c. - Domaine d'application d'un Système d'Information Géographique (SIG)

Les applications pratiques des systèmes d'information géographique sont vastes et variées, touchant presque tous les domaines où la localisation est une composante clé. Voici quelques exemples d'applications concrètes des SIG :

- **Gestion Urbaine et Planification Territoriale** : Les SIG sont utilisés pour la planification de l'utilisation des terres, la visualisation de l'impact des projets de développement, l'aménagement urbain et la gestion des infrastructures publiques.

- **Environnement et Conservation** : Les SIG aident dans la gestion des ressources naturelles, l'analyse des changements environnementaux, la planification de la conservation, et le suivi de la biodiversité.

- **Gestion des Catastrophes et Réponses d'Urgence** : Les SIG permettent de modéliser les risques naturels, de planifier les interventions d'urgence, et de coordonner les opérations de secours en cas de catastrophes.

## III.d. - Présentation de Geotrek-Admin

Geotrek-admin est une application web open source développée par Makina Corpus, destinée à la gestion des équipements touristiques et patrimoniaux, à la planification des interventions terrain et à la valorisation des itinéraires de randonnée. Utilisée par de nombreuses collectivités, parcs naturels et gestionnaires d'espaces, elle permet de centraliser l'ensemble des données territoriales : sentiers, signalisation, patrimoine, interventions.

Développée avec des technologies libres telles que PostGIS, GeoDjango et des bibliothèques cartographiques web open source telle que Leaflet, Geotrek-admin constitue un exemple concret de SIG web opérationnel au service des collectivités. Elle intègre toutes les fonctions fondamentales d'un Système d'Information Géographique moderne :

- stockage et gestion de données géospatiales,
- visualisation cartographique interactive,
- recherche et filtres thématiques,
- exports cartographiques et interconnexion avec d'autres systèmes.

Au-delà de son rôle en tant qu'outil de gestion cartographique, Geotrek-admin est également la pierre centrale de l'écosystème Geotrek. Elle alimente plusieurs autres modules qui prolongent ses fonctionnalités sur d'autres supports :

- Geotrek-rando **:** portail web public de valorisation des randonnées,
- Geotrek-widget : composants cartographiques embarquables sur d'autres sites,
- Geotrek-mobile : application mobile de consultation et de saisie d'informations terrain.

Grâce à cette architecture modulaire, Geotrek-admin joue un rôle structurant dans la gestion, l'exploitation et la diffusion de données géographiques à destination des acteurs publics et de leurs usagers.

L'illustration suivante montre la vue liste de l'application.

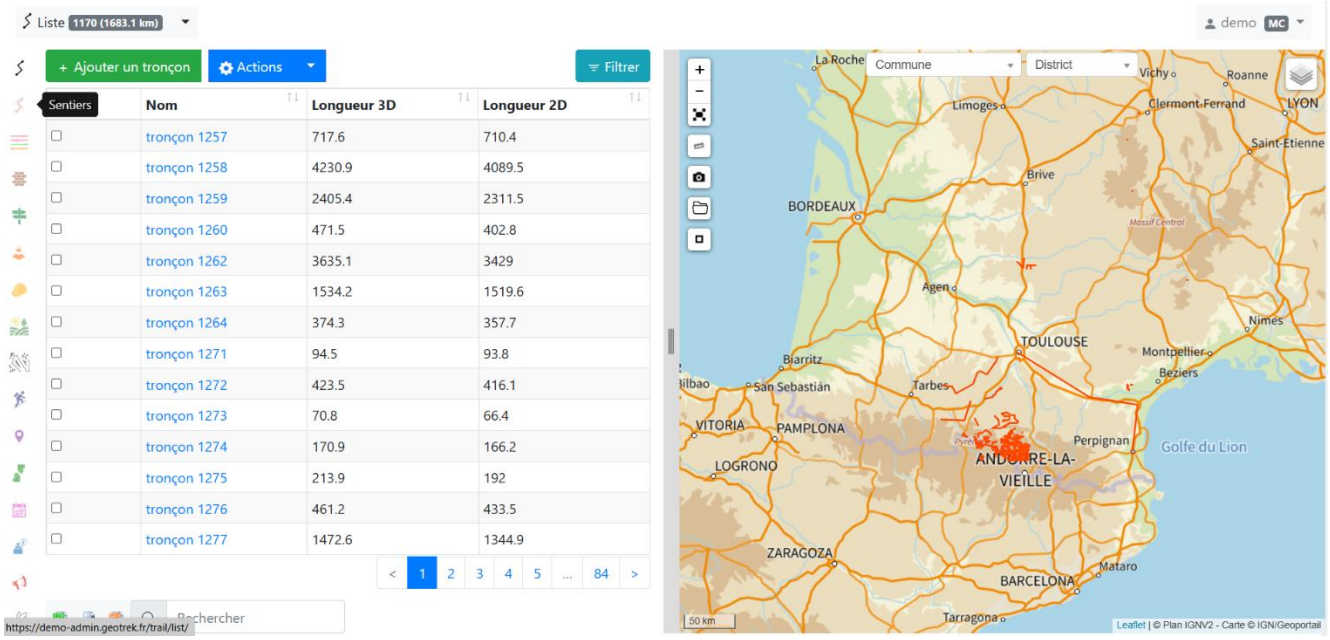


*Figure 2 : Interface graphique de la vue liste de Geotrek-admin*

## III.e. - Présentation de django-mapentity

Django-mapentity est une librairie Django open-source développée par Makina Corpus, qui a pour objectif de faciliter la création rapide d'interfaces web cartographiques pour des applications métiers basées sur des données géographiques.

À l'origine, django-mapentity faisait partie intégrante de l'application Geotrek-admin. Elle y jouait un rôle central dans la gestion de l'affichage cartographique et la génération automatique d'interfaces liées aux entités géographiques. Face à son potentiel de réutilisation, elle a ensuite été extraite du projet pour devenir une librairie indépendante, maintenue séparément, afin de pouvoir être utilisée dans d'autres applications Django orientées SIG.

### III.e.i. - Fonctionnement général

Django-mapentity repose sur une architecture générique qui permet, avec très peu de configuration, de générer automatiquement les vues essentielles pour les modèles géométriques : liste, détail, ajout et modification. Chaque page de visualisation intègre également une carte interactive basée sur Leaflet, facilitant la consultation des objets géographiques.

En complément, le framework intègre une API REST entièrement dynamique, capable d'exposer automatiquement les modèles géographiques sous forme de données GeoJSON. Cette API s'accompagne de nombreuses fonctionnalités standards telles que la recherche, le filtrage et la pagination. Le framework permet également l'export des données au format CSV, GPX ou Shapefile, offrant une grande souplesse pour leur réutilisation, leur analyse ou leur partage.

L'illustration suivante montre la vue liste de l'application.

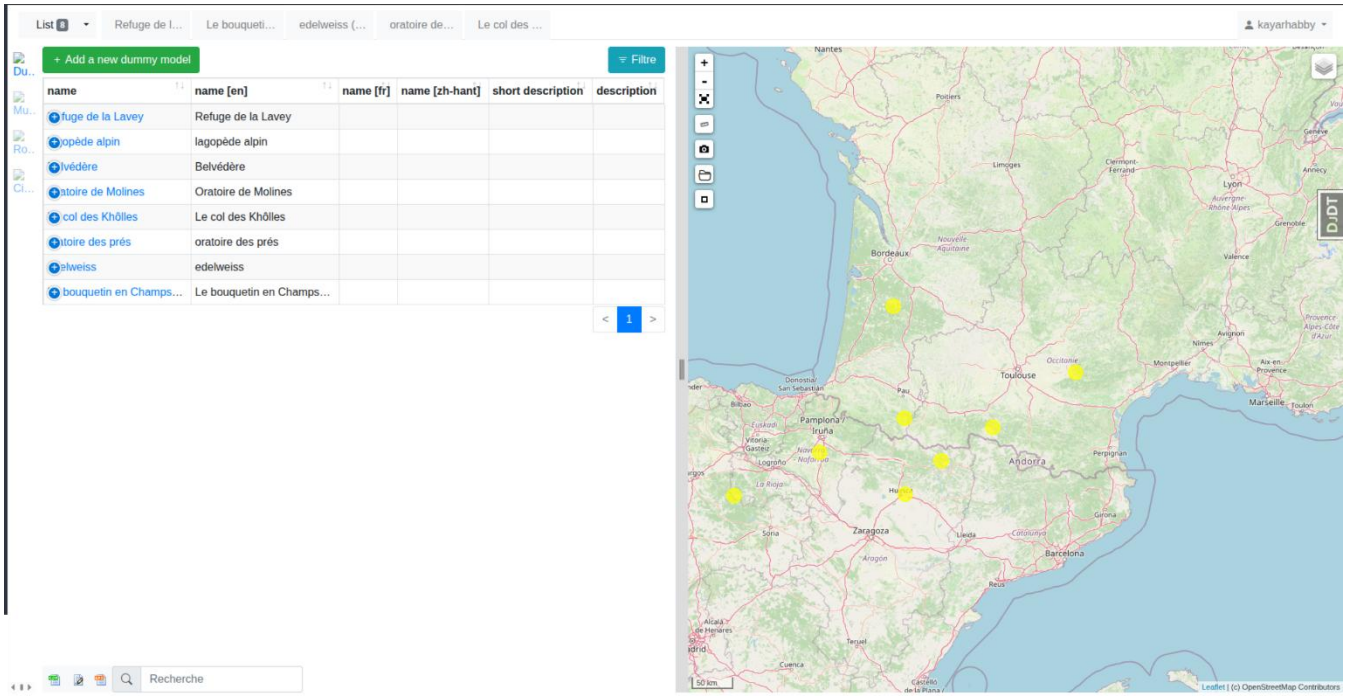


*Figure 3 : Interface graphique de la vue liste de django-mapentity (version leaflet)*

## III.f. - Présentation de django-leaflet

Django-leaflet est une librairie Django open source développée par Makina Corpus. Elle a pour but de faciliter l'intégration du moteur cartographique Leaflet dans les applications Django. Elle repose sur le moteur JavaScript Leaflet, très populaire pour sa facilité d'utilisation.

Les principaux objectifs de django-leaflet sont :

- Afficher rapidement une carte interactive dans un Template Django à l'aide d'une balise **{% leaflet_map %},**
- Ajouter des fonds de carte (ex. OpenStreetMap),
- Proposer des outils de navigation (zoom, centrage, affichage de couches, réinitialisation de la carte),
- Fournir un widget de formulaire (LeafletWidget) pour l'édition de géométries (point, ligne, polygone) dans l'interface admin ou les vues métiers.

Elle est donc particulièrement utile pour les projets Django manipulant des objets géographiques, notamment en association avec GeoDjango et une base de données PostGIS.

### III.f.i. - Limitations techniques

Malgré sa popularité, django-leaflet présente aujourd'hui de sérieuses limitations techniques. La version utilisée dans Geotrek-admin repose encore sur Leaflet 0.7, sortie en 2013, aujourd'hui obsolète et non maintenue.
Les principales limitations associées à cette version sont les suivantes :

- **Pas de support WebGL** : Leaflet 0.7 repose uniquement sur le DOM et SVG, ce qui empêche l'exploitation du GPU du navigateur. Le rendu cartographique devient lourd et peu fluide dès qu'un certain nombre de géométries est atteint.

- **Injection directe dans le DOM** : Chaque entité géographique est transformée en élément SVG, directement inséré dans le DOM. Cela entraîne une saturation rapide, provoquant des ralentissements, des blocages, voire des plantages du navigateur.

- **Aucune prise en charge native des tuiles vectorielles** : Impossible d'utiliser les tuiles modernes (vectortiles) telles que celles proposées par Mapbox, MapLibre ou d'autres fournisseurs. Cela freine la mise à l'échelle et rend la carte inutilisable sur des projets à forte volumétrie.

- **Code JavaScript figé et couplé à jQuery** : L'implémentation repose sur une architecture JavaScript ancienne, fortement dépendante de jQuery, avec peu de modularité. Toute évolution, migration ou refactorisation s'en trouve complexe et coûteuse.

L'illustration suivante montre concrètement comment Leaflet injecte chaque géométrie comme élément SVG dans le DOM, soulignant ainsi les limites structurelles de cette approche en matière de performance et de scalabilité.

```
<!-- Map Panel -->
<div class="col-12 col-md-6" id="panelright">
  <div class="map-panel">
    <div id="maphead">…</div>
    <div id="mainmap" class="leaflet-container leaflet-touch leaflet-retina leaflet-fade-anim"
     tabindex="0">
      <div class="leaflet-map-pane" style="transform: translate3d(-23px, 0px, 0px);">
        <div class="leaflet-tile-pane">…</div>
        <div class="leaflet-objects-pane">
          <div class="leaflet-shadow-pane"></div>
          <div class="leaflet-overlay-pane"> == $0
            <svg class="leaflet-zoom-animated" width="543" height="1184" viewBox="-65 -191 543 1184"
             style="transform: translate3d(-65px, -191px, 0px);">…</svg>
          </div>
          <div class="leaflet-marker-pane"></div>
          <div class="leaflet-popup-pane"></div>
        </div>
      </div>
      <div class="leaflet-control-container">…</div>
    </div>
  </div>
</div>
</div>
</div>
```

*Figure 4 : Insertion des géométries spatiales par Leaflet*

# IV - MISE EN ŒUVRE DE LA MIGRATION

La migration du moteur cartographique Leaflet (via django-leaflet) vers MapLibre GL JS s'est inscrite dans une démarche progressive et structurée. L'ensemble du travail a été amorcé au sein de la librairie django-mapentity, socle partagé entre plusieurs projets.

## IV.a. - Analyse du code existant de django-mapentity

La première phase de la migration a consisté à mener une analyse technique détaillée du code de la librairie django-mapentity, afin d'identifier les parties critiques du système cartographique basé sur django-leaflet. Cette étape était essentielle pour comprendre les mécanismes internes de l'affichage, de l'interaction et du chargement des données géographiques, avant toute substitution par MapLibre. Cette analyse a permis de dégager une structure événementielle cohérente, mais fortement couplée à django-leaflet, à jQuery et à des composants personnalisés qui interagissent étroitement entre eux. Le code repose sur plusieurs fichiers JavaScript clés, notamment « mapentity.map.js », « mapentity.views.js », « mapentity.js » et sur une logique de données injectées dans le DOM.

## IV.b. - Initialisation de la carte et des vues

L'initialisation de la carte débute dans le fichier mapentity.js, où un objet context est extrait depuis le DOM. Ce contexte contient des métadonnées comme le nom de la vue (viewname), le modèle concerné (modelname), ou encore une éventuelle clé primaire (pk). Ces informations permettent de déclencher dynamiquement des événements personnalisés selon la page affichée (ex. : liste ou détail d'un objet).

Une fois le contexte identifié, un premier événement entity:view:{viewname} est émis, suivi d'un map:init capté par django-leaflet qui se charge d'initialiser la carte. Deux autres événements fondamentaux sont ensuite déclenchés : entity:map (événement global) et entity:map:{viewname} (spécifique à la vue en cours). C'est cette architecture événementielle qui permet à l'application de configurer dynamiquement les couches, les contrôles et le comportement de la carte selon le contexte utilisateur.

```
// Views
var context = $('body').data();
console.debug('View ', context.modelname, context.viewname);
$(window).trigger('entity:view:' + context.viewname, [context]);

// Maps
$(window).on('map:init', function (e) {
    var data = e.originalEvent ?
               e.originalEvent.detail : e.detail;
    $.extend(data, context);
    $(window).trigger('entity:map', [data]);
    $(window).trigger('entity:map:' + context.viewname, [data]);
```

*Figure 5 : initialisation de la carte dans mapentity.js.*

## IV.c. - Logique événementielle par type de vue

### IV.c.i. - Vue liste

Dans la vue liste, la carte affiche automatiquement l'ensemble des objets géographiques visibles, récupérés depuis une API REST qui renvoie des données au format GeoJSON.

Une synchronisation dynamique relie la carte et la table des objets : lorsqu'un utilisateur applique un filtre ou déplace la carte, les données affichées s'actualisent automatiquement pour refléter ces changements, aussi bien sur la carte que dans la liste.

La carte propose également des outils permettant d'importer des données supplémentaires aux formats GeoJSON, GPX ou KML, ainsi que des fonctionnalités comme le recentrage, la capture d'écran, l'affichage en plein écran et la mesure de distance. Ces outils rendent l'exploration et la manipulation des données géographiques plus simples et plus interactives directement depuis l'interface.

### IV.c.ii. - Vue détail

Dans la vue détail, la carte affiche un unique objet géographique correspondant à l'élément sélectionné. Les données de cet objet sont récupérées via un appel AJAX vers l'API REST, qui renvoie une réponse au format GeoJSON. Un contrôle de capture d'écran est également disponible, et il tient compte du « contexte courant » (notamment le zoom, le centrage ou les couches visibles), afin de permettre une exportation fidèle à ce que l'utilisateur voit.

Dans django-mapentity, la vue détail est conçue pour offrir une consultation claire et ciblée d'un objet, avec une carte interactive centrée sur cet élément, accompagnée de ses informations attributaires (nom, type, description, etc.). La carte ne propose ici ni synchronisation avec une liste ni filtrage, contrairement à la vue liste. L'accent est mis sur l'édition ou la consultation d'un seul objet, ce qui simplifie l'interface.


*Figure 6 : Interface graphique de la vue détail de django-mapentity (version leaflet)*

### IV.c.iii. - Vue formulaire

Lorsqu'on accède à un formulaire pour créer ou modifier un objet géographique, la carte inclut un outil de dessin interactif qui permet de tracer, déplacer ou modifier des formes (points, lignes, polygones) directement sur la carte, sans avoir besoin de saisir manuellement des coordonnées.

Les géométries dessinées sur la carte sont automatiquement reliées aux champs du formulaire : toute modification effectuée d'un côté est immédiatement reflétée de l'autre. Cela offre à l'utilisateur une interface visuelle claire et intuitive pour définir ou ajuster la localisation et la forme des objets, tout en maintenant la cohérence des données enregistrées.

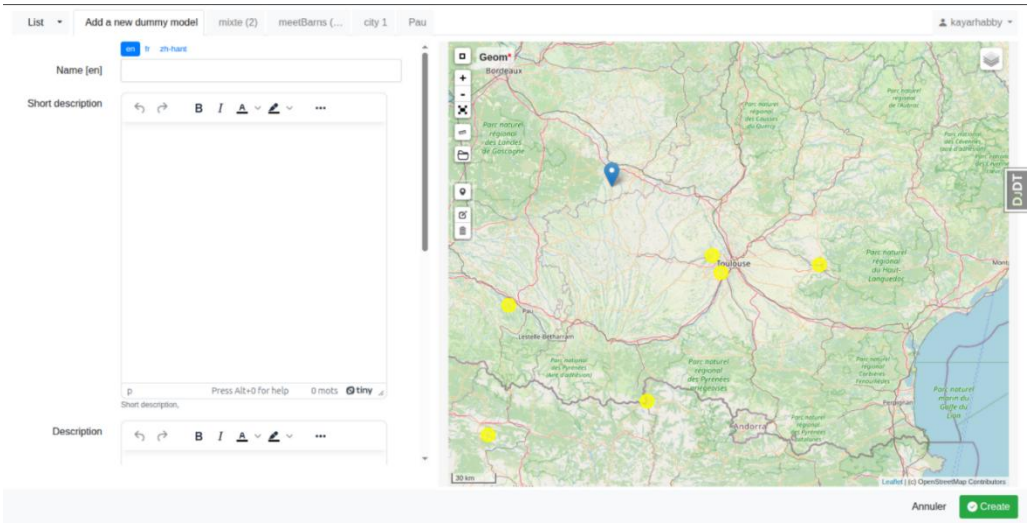


*Figure 7 : Interface graphique de la vue formulaire de django-mapentity (version leaflet)*

L'illustration ci-après illustre la séquence d'initialisation de la carte et des vues de django-mapentity dans le code de base, depuis la récupération du contexte jusqu'au déclenchement des événements responsables du rendu cartographique et de l'affichage dynamique des couches selon la vue active (liste ou détail).

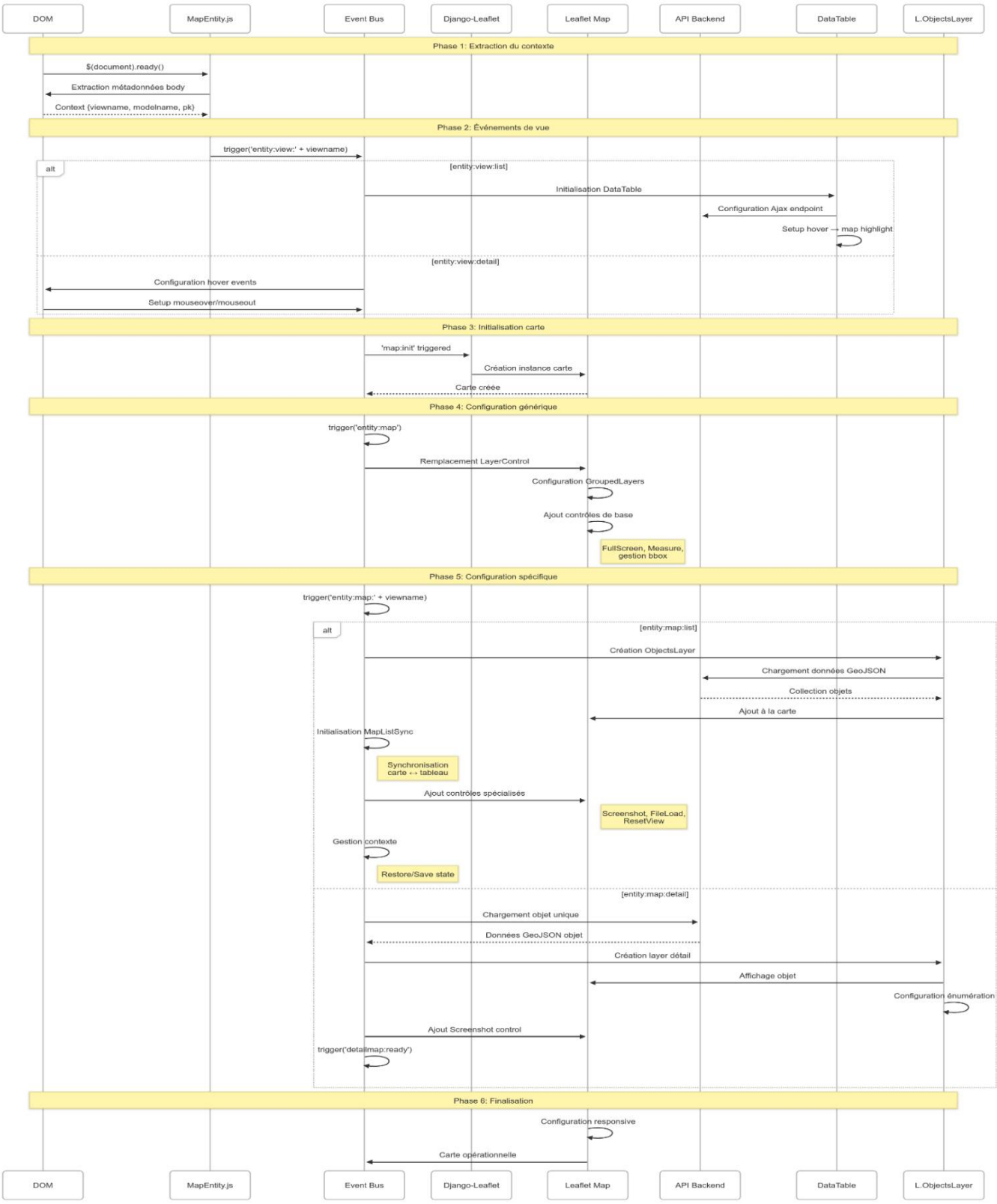


*Figure 8 : Enchaînement des événements JS de la carte : logique et effets fonctionnels*

## IV.d. - Détails des classes cartographiques principales

La carte interactive de l'application django-mapentity repose sur plusieurs modules JavaScript personnalisés, développés autour de la bibliothèque django-leaflet. Ces composants permettent d'afficher des objets géographiques, de gérer les filtres dynamiques, et d'assurer une synchronisation fluide entre la carte, les listes et les formulaires.

- **Affichage des objets géographiques (ObjectsLayer)** : Ce module interroge une API REST pour récupérer les entités à afficher sur la carte. Les réponses sont fournies au format GeoJSON, ce qui permet leur affichage direct via Leaflet. ObjectsLayer permet également de mettre à jour dynamiquement les objets en fonction de filtres, de surligner certains éléments, ou d'ajouter de nouvelles couches à la carte.

- **Synchronisation carte/liste (MapListSync) :** Ce composant maintient la cohérence entre la carte, la liste d'objets et les filtres. Si un filtre est appliqué via un formulaire, la carte se met à jour. Inversement, un déplacement ou un zoom sur la carte entraîne une mise à jour automatique de la liste.

- **Filtres dynamiques (TogglableFilter) :** Ce module gère l'affichage des filtres, souvent générés à partir des modèles Django. Il agit directement sur les éléments affichés sur la carte et dans la liste, en interaction avec MapListSync.

- **Addition et Modification des géométries (GeometryField) :** Utilisé dans les formulaires d'addition et de modification , ce composant permet de dessiner, modifier ou supprimer des géométries directement sur la carte, en s'appuyant sur leaflet-draw.

- **Sauvegarde de l'état de la carte (Mapentity.Context) :** Ce module enregistre l'état courant de la carte (zoom, centre, couches visibles…) afin de pouvoir le restaurer plus tard. Cela permet à l'utilisateur de retrouver facilement une vue précédente même après un rafraichissement de la page.

- **Historique (Mapentity.History) :** Il enregistre les filtres récemment appliqués, offrant la possibilité de revenir rapidement à un état antérieur de la liste ou de la carte.

## IV.e. - Dépendances JavaScript externes

L'ensemble du front cartographique repose également sur des librairies JavaScript, qui assurent des fonctionnalités avancées :

- **Leaflet.fullscreen :** permet de passer la carte en mode plein écran grâce à un contrôle ajouté dynamiquement.

- **GeometryUtil + Leaflet.Snap :** ensemble utilisé pour aligner automatiquement une géométrie dessinée sur un objet existant. Essentiel pour la précision dans les tracés.

- **Leaflet.Spin :** permet l'ajout de spinners (animations de chargement) sur les composants de la carte (formulaires, couches, filtres).

- **Leaflet.LayerIndex :** calcule les emprises géographiques des objets (via getBounds()) et les indexe spatialement pour des traitements rapides (zoom, fitBounds...).

- **Leaflet.GroupedLayerControl :** propose une interface de gestion des couches plus lisible, en regroupant les couches de base et les overlays dans des catégories distinctes.

- **Leaflet.Label :** gère l'affichage des popups et infobulles associées à des objets ou à des couches vectorielles.

- **Leaflet.MeasureControl + RTree :** permet de mesurer distances et surfaces, avec un calcul spatial optimisé grâce à un arbre géométrique (RTree).

- **Leaflet.FileLayer + toGeojson :** permet d'importer des fichiers KML, GPX, GeoJSON. Les fichiers KML et GPX sont automatiquement convertis en GeoJSON via togeojson avant d'être ajoutés à la carte comme une nouvelle couche interactive.

La quasi-totalité de ces librairies ont été développé et maintenu par l'entreprise à l'exception de RTree et toGeojson.

## IV.f. - Migration vers MapLibre GL JS et refonte JavaScript

### IV.f.i. - Phase 1 : Vue liste

La première étape de la migration a consisté à reconstruire le fonctionnement complet de la vue liste, qui est la plus riche en interactions cartographiques. Cette phase a permis de valider l'intégration de MapLibre GL JS en remplacement de Leaflet, tout en conservant les principales fonctionnalités existantes :

- Les contrôles cartographiques ont été réintégrés dans la nouvelle interface : plein écran, navigation, mesure, import de fichiers (GeoJSON, GPX, KML), sélection des couches et affichage de l'échelle.

- Une nouvelle classe « MaplibreObjectsLayer » a été introduite, en remplacement de « LeafletObjectsLayer », afin d'assurer une compatibilité complète avec MapLibre. Elle conserve le principe initial : charger les objets géographiques via l'API REST en format GeoJSON.

- Le système de journalisation des actions cartographiques, tel que les déplacements ou zooms, a été reconstruit avec « MaplibreMapentityHistory ».

- La synchronisation entre la carte, la liste et les filtres a été conservée et adaptée via « MaplibreMaplistSync » et « MaplibreTogglableFilter », garantissant que seuls les objets visibles ou filtrés soient affichés sur la carte.

Plusieurs classes ont été entièrement repensées ou recréées, en s'appuyant sur les logiques héritées de Leaflet, mais en adoptant une approche plus moderne, basée sur du JavaScript natif et sans jQuery.

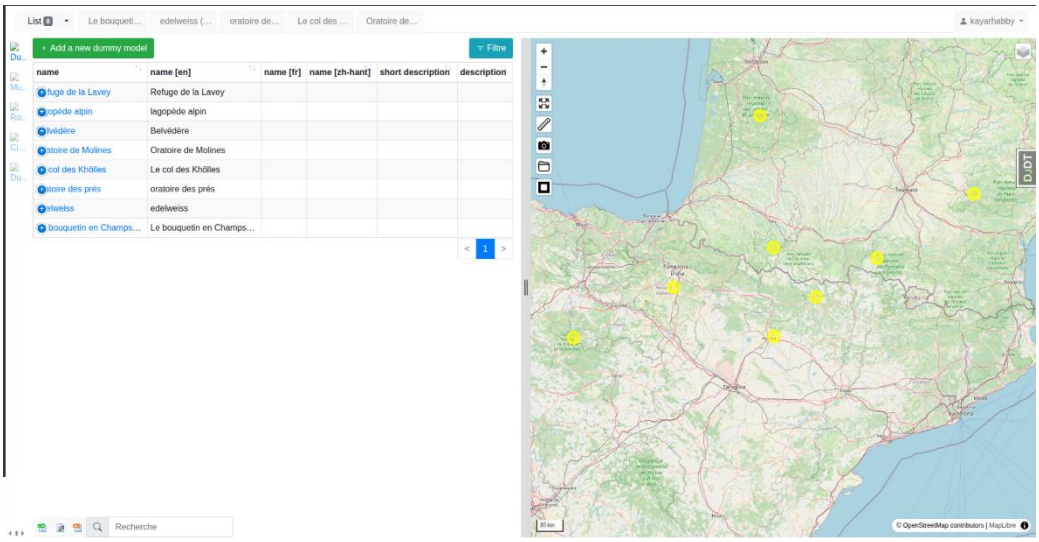


*Figure 9 : Interface graphique de la vue liste de django-mapentity (version Maplibre)*

L'initialisation de la carte est prise en charge par la classe « MaplibreMap », qui configure et charge MapLibre en se basant sur les paramètres définis dans le fichier de configuration (settings.py). Pour la gestion des fichiers cartographiques (import/export des formats GeoJSON, KML, GPX), les classes « MaplibreFileLayerControl » et « MaplibreFileLoader » interviennent, s'appuyant sur la librairie « togeojson » pour assurer le parsing et la conversion des données.

Par ailleurs, les classes « MaplibreRectangle » et « MaplibreSerializers » contribuent au filtrage spatial en définissant une boîte englobante graphique (bbox) qui est transmise au backend pour limiter les données retournées.

Deux contrôles ont notamment nécessité une attention particulière :

- **MaplibreMeasureControl :** bien que son objectif soit identique à celui de la version Leaflet, son comportement visuel diffère. Par exemple, l'affichage de la mesure n'apparaît pas au même endroit. Il s'appuie sur une classe annexe, « MaplibreMeasureDistanceDisplay », dédiée à l'affichage dynamique des valeurs mesurées.

- **MaplibreLayerControl :** bien que globalement équivalent, quelques différences notables subsistent : dans Leaflet, le menu s'ouvre au survol ; ici, un clic est requis. De plus, l'ajout des couleurs devant les noms des couches dans le menu n'a pas été réimplémenté.

La classe « MaplibreObjectsLayer » est une réécriture complète de LeafletObjectsLayer, reprenant les mêmes responsabilités :

- Chargement et affichage des entités via GeoJSON.
- Calcul des emprises géographiques pour permettre un focus sur les objets (notamment lors d'un double-clic ou d'une redirection depuis une autre vue).
- Ajout dynamique de couches d'objets et de fonds de plan.

Les classes suivantes ont également été recréées dans un esprit de continuité fonctionnelle, avec une logique modernisée : MaplibreMaplistSync, MaplibreTogglableFilter, MaplibreMapentityContext, MaplibreMapentityHistory , MaplibreResetViewControl. Les noms des méthodes ont été conservés pour faciliter la maintenance, bien que leur implémentation soit différente pour s'adapter aux API MapLibre et JavaScript vanilla. Seules quelques portions, à l'image du tooltip dans le filtre, ont temporairement conservé du code jQuery afin de préserver l'interface visuelle d'origine. Les fichiers javascript n'ayant aucun lien direct avec la carte leaflet ont également été laissé tel quel, c'est notamment le cas pour le fichier « RelatedObjectLookups.js ».

| Classe | Rôle principal |
|---|---|
| **MaplibreMap** | Initialise la carte MapLibre avec les paramètres globaux |
| **MaplibreObjectsLayer** | Charge et affiche dynamiquement les objets sur la carte |
| **MaplibreLayerControl** | Gère le menu des couches cartographiques |
| **MaplibreMeasureControl** | Permet la mesure interactive sur la carte |
| **MaplibreMeasureDistanceDisplay** | Affiche dynamiquement les distances mesurées |
| **MaplibreFileLayerControl** | Contrôle d'import/export des fichiers cartographiques |
| **MaplibreFileLoader** | Charge les fichiers GPX/KML/GeoJSON et les transforme si besoin |
| **MaplibreRectangle** | Dessine un rectangle de sélection pour filtrage spatial |
| **MaplibreSerializers** | Sérialise les données à envoyer vers le backend |
| **MaplibreMapentityMap** | Logique cartographique selon la vue (liste, détail) |
| **MaplibreMapentityView** | Gère les initialisations liées aux vues |
| **MaplibreMapentity** | Point d'entrée principal de l'initialisation cartographique |
| **MaplibreMaplistSync** | Synchronise la liste et la carte |
| **MaplibreMapentityHistory** | Gère le journal des filtres appliqués |
| **MaplibreMapentityContext** | Sauvegarde et restauration de l'état de la carte |
| **MaplibreTogglableFilter** | Gère l'UI de filtrage et la communication avec la carte |
| **MaplibreResetViewControl** | Réinitialise la vue de la carte à son état initial |

*Tableau 2 : Liste non exhaustive des classes développées compatibles avec MapLibre*

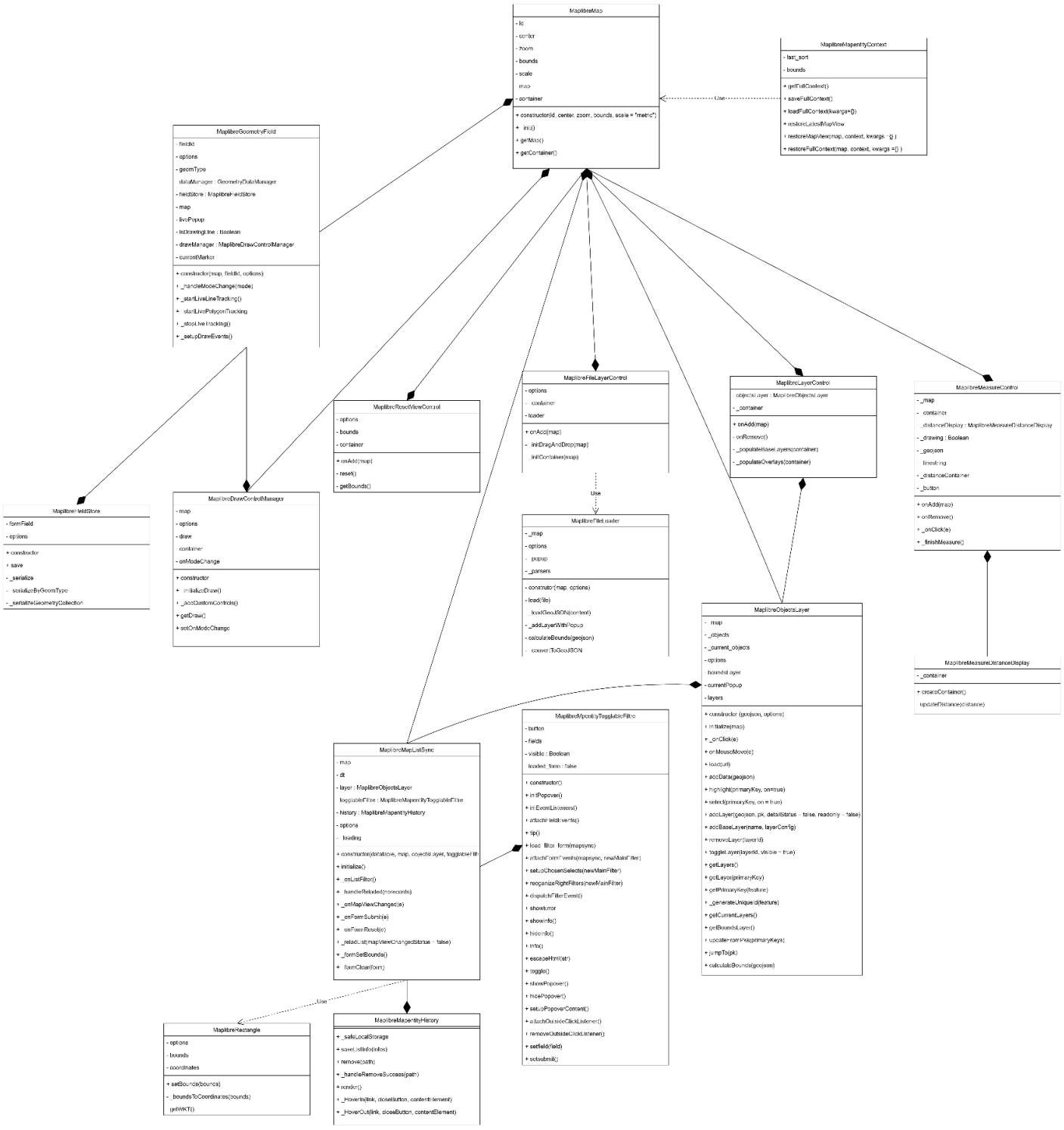


*Figure 10 : Diagramme de classe - Mapentity avec Maplibre GL JS*

### IV.f.ii. - Phase 2 : Vue détail

La migration de la vue détail a été amorcée une fois le système de gestion contextuelle et l'initialisation cartographique stabilisés. Toutefois, certaines actions comme le clic sur l'objet, les surbrillances ou l'affichage de fenêtres d'information ont été volontairement désactivées. Cela permet d'assurer que l'utilisateur observe uniquement les données sans interaction, conformément au comportement souhaité dans cette vue.

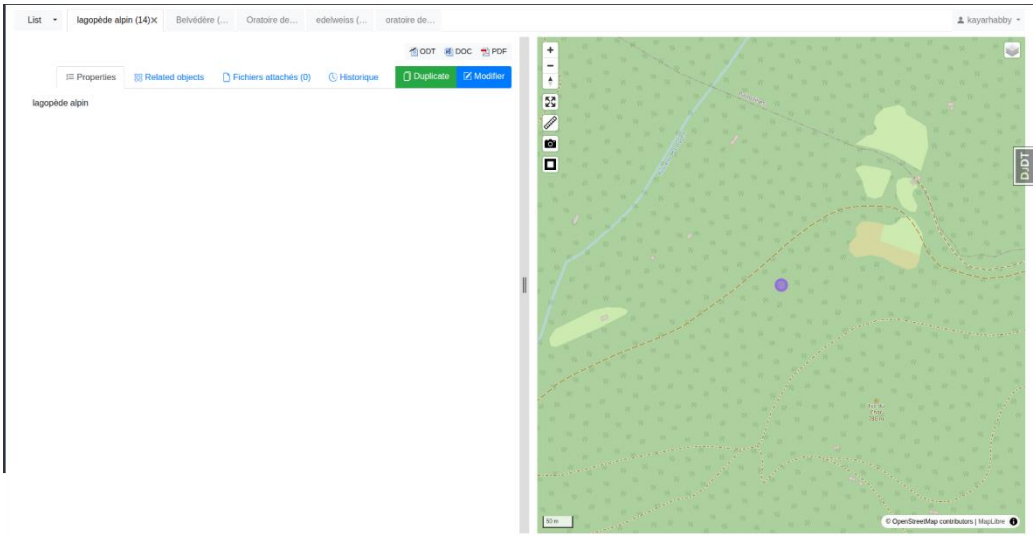


*Figure 11 : Interface graphique de la vue détail de django-mapentity (version Maplibre)*

### IV.f.iii. - Phase 3 : Vue formulaire

La troisième phase de la migration porte sur la vue formulaire, utilisée pour créer ou modifier un objet géographique. Elle se divise en deux volets techniques principaux : l'adaptation du widget Django à MapLibre, puis la mise en place des outils de dessin compatibles.

Le premier volet consiste à remplacer le LeafletWidget de django-mapentity, historiquement basé sur django-leaflet, par une version compatible avec MapLibre. Cela a nécessité une refonte du widget côté Python, ainsi qu'une adaptation du template HTML de rendu, notamment pour afficher la carte miniature

associée au champ de délimitation (bbox). Ces modifications sont intégrées directement dans le code source de django-mapentity afin d'assurer une compatibilité native.

Parallèlement, le second volet vise à reconstruire les outils de dessin, auparavant fournis par Leaflet-Draw. Pour cela, la librairie « maplibre-gl-draw » a été utilisé. Plusieurs classes JavaScript ont été développées pour gérer cette interaction :

- **MaplibreFieldStore :** responsable de la gestion des données géométriques côté frontend, notamment la sérialisation et la désérialisation ;

- **MaplibreGeometryField :** gère l'interface visuelle de la carte dans le formulaire, en synchronisation avec les champs HTML Django ;

- **MaplibreDrawControlManager :** configure et contrôle les outils de dessin (polygone, ligne, point), et transmet les données au widget.

L'ensemble de ces composants est conçu pour reproduire un comportement fluide et intégré dans le formulaire, tout en assurant une compatibilité stricte avec la logique Django côté serveur.

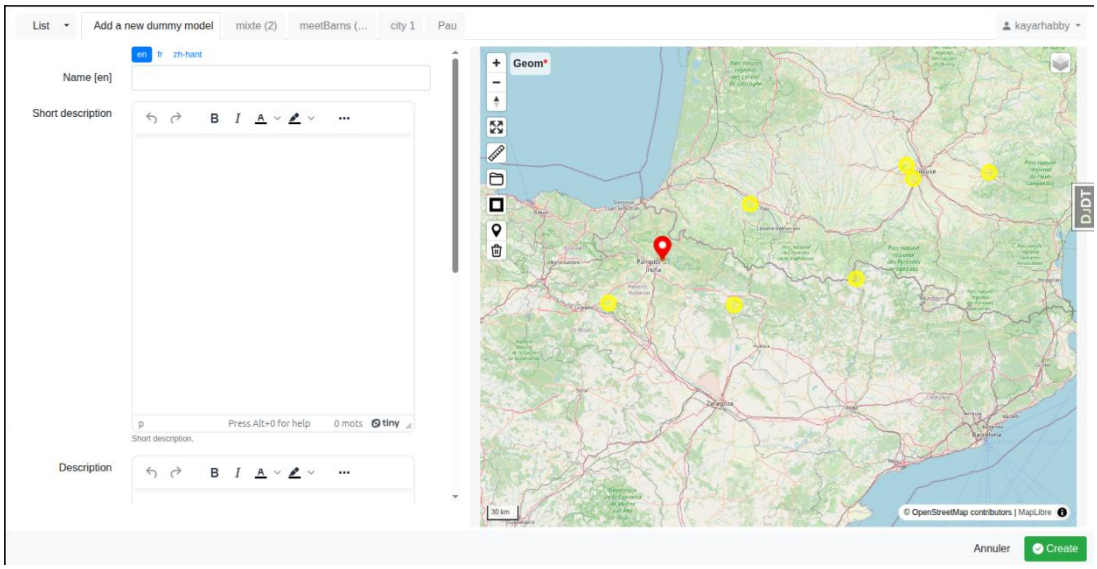


*Figure 12 : Interface graphique de la vue formulaire de django-mapentity (version Maplibre)*

### IV.f.iv. - Phase 4 : Contrôle de capture d'écran

La dernière phase prévue est la reconstruction du contrôleur de capture de carte (Screenshot), fonction très spécifique et complexe à reproduire avec MapLibre. Si son interface est relativement simple dans la version actuelle (un bouton permettant de capturer l'état visuel de la carte), sa mise en œuvre réelle repose sur plusieurs éléments sensibles.

La principale difficulté réside dans la coordination entre le rendu de la carte et le déclenchement de la capture. En effet, dans un environnement WebGL comme celui de MapLibre, les éléments cartographiques ne sont pas stockés directement dans le DOM mais rendus via un Canvas WebGL, ce qui rend leur manipulation plus délicate. Il faut donc s'assurer que la carte ait fini de se rendre avant de lancer la capture.

Screamshotter pour fonctionner efficacement, doit être capable de :

- intercepter la sortie du Canvas avec un rendu fidèle à ce que voit l'utilisateur,
- masquer ou afficher temporairement certains éléments (légendes, contrôles),
- exporter une image propre (PNG) et la proposer à l'utilisateur.

Enfin, cette fonctionnalité doit s'articuler avec le système de restauration du contexte cartographique « MaplibreMapentityContext », en validant que des méthodes comme restoreFullContext() ou restoreMapView() permettent de restituer exactement l'état de la carte au moment de la capture.

### IV.f.v. - Intégration finale dans Geotrek-admin

Une fois la migration de django-mapentity finalisée, une phase d'intégration dans Geotrek-admin est prévue. L'objectif sera d'adapter progressivement les vues internes de l'application pour qu'elles reposent sur la nouvelle implémentation de la librairie django-mapentity avec le nouveau moteur MapLibre.

# V - RÉSULTATS OBTENUS ET CONTRIBUTIONS

## V.a. - État d'avancement de la migration cartographique

À ce stade du stage, la migration de Leaflet vers MapLibre GL JS dans la librairie django-mapentity est bien avancée. La vue liste a été entièrement migrée, avec un remplacement progressif de tous les composants dépendants de Leaflet. Les contrôleurs, la gestion des couches, l'import de fichiers cartographiques, la synchronisation entre carte et liste, les filtres et les outils de dessin ont été recréés pour MapLibre, tout en respectant la logique fonctionnelle initiale.

La vue détail a également été migrée. Elle permet l'affichage d'un objet géographique unique, dans une carte rendue non interactive, comme dans la version d'origine : clics, affichage de fenêtres d'information, surbrillances et interactions sont désactivés.

La vue formulaire a également été migrée. La carte MapLibre a été intégrée au widget Django, et le contrôleur de dessin (maplibre-gl-draw) est pleinement opérationnel. La sérialisation de la géométrie dessinée ainsi que son envoi au backend ont été mises en place, tout comme l'enregistrement correct de cette géométrie dans la base de données. Ces fonctionnalités sont désormais fonctionnelles. La migration de cette vue est donc en grande partie finalisée.

Par ailleurs, plusieurs fonctionnalités jusque-là présentes dans le code mais non opérationnelles ont été réadaptées et remises en état de marche. C'est notamment le cas de la surbrillance des champs dans la liste lors d'un simple clic, ainsi que du focus automatique sur un objet de la carte lors d'un double-clic sur une entrée de la liste.

Enfin, la migration du module de capture d'écran est planifiée et devrait être achevée d'ici la fin du stage. En revanche, l'intégration de la nouvelle version de django-mapentity dans Geotrek-admin est encore incertaine : elle pourra éventuellement être amorcée, mais ne sera probablement pas finalisée avant la fin du stage, et pourrait faire l'objet d'un nouveau ticket ou chantier par la suite.

### V.b. - Valeur ajoutée pour le projet

Le travail accompli représente une refonte complète du moteur cartographique de django-mapentity. Cette migration permet de sortir d'un socle technique obsolète et contraignant, et d'installer une base moderne, plus performante, modulable et maintenable.

La migration s'est accompagnée d'une réécriture du code, dans chaque fichier JavaScript où Leaflet intervenait. Tous les composants (liste, détail, formulaire) ont été reconstruits autour de MapLibre, dans une logique de substitution progressive. Le projet bénéficie ainsi d'une base saine pour accueillir de futures évolutions cartographiques sans dépendre d'une technologie vieillissante.

Ce travail aura également un impact direct sur Geotrek-admin, dont le frontend cartographique pourra progressivement adopter la nouvelle version, dans le respect des comportements existants.

## VI - BILAN PERSONNEL ET COMPÉTENCES ACQUISES

### VI.a. - Compétences techniques développées

Lorsque j'ai commencé ce stage, je ne connaissais ni Django, ni la programmation orientée objet en Python. J'ai donc consacré trois (3) semaines à apprendre le Framework, comprendre la structure des templates, les vues génériques, les formulaires et les interactions entre le backend et le frontend en développant plusieurs projet Django basique afin de mieux assimiler l'environnement et les concepts des systèmes d'informations géographiques qui allait par la suite faire l'objet de mon stage.

En parallèle, j'ai approfondi mes connaissances en JavaScript moderne (ES6), et appris à organiser le code en modules réutilisables, tout en respectant les principes d'interaction événementielle propres à MapLibre.

Enfin, j'ai acquis une première expérience concrète en cartographie web, en manipulant des objets géospatiaux, des formats de fichiers spécifiques (GeoJSON, KML, GPX), en utilisant MapLibre GL JS et ses extensions.

## VI.b. - Compétences transversales

Sur le plan méthodologique, j'ai appris à travailler de manière structurée, en découpant mon travail en phases progressives (analyse, refactorisation, migration, test). L'approche était très concrète : les tests ont été réalisés manuellement, en validant à chaque étape la fidélité du rendu et le bon fonctionnement des composants. Ce cadre m'a également permis de renforcer mon autonomie, en prenant les initiatives nécessaires pour progresser de manière indépendante tout en restant en contact régulier avec mes tuteurs (Jean Etienne et Joaquim).

Par ailleurs, j'ai dû rapidement m'adapter à de nouveaux outils et concepts, notamment Django et les notions géographiques liées au projet, ce qui a renforcé ma capacité à assimiler rapidement un environnement technique et métier complexe.

En cas de blocage, j'ai pu m'appuyer sur l'expertise des membres de l'équipe, notamment des développeurs frontends expérimentés dans les bibliothèques JavaScript modernes et les outils SIG (Maxime, Florian et Benjamin). L'expertise de Marine, chargée support au sein de Makina, a également été précieuse pour mieux comprendre les aspects métiers des différents composants présents dans l'ancienne version.

## VI.a. - Lien avec la formation

Ce stage s'inscrit pleinement dans ma formation en Sciences et Technologies de l'Information, option Systèmes d'Information et Réseaux. Il m'a permis d'appliquer des notions clés comme l'architecture des systèmes d'information et la structuration des données spatiales, en lien direct avec les problématiques de gestion et de visualisation de données géographiques.

En participant à la migration du moteur cartographique et à la réécriture complète des composants JavaScript, j'ai pu approfondir mes compétences en développement orienté objets et mieux comprendre le dialogue entre frontend et backend. Cette approche a renforcé ma capacité à concevoir des applications maintenables et adaptées à l'évolution des technologies.

Enfin, cette expérience m'a sensibilisé à l'enjeu de la dette technique et à l'importance de moderniser des briques logicielles tout en conservant la logique fonctionnelle historique. Elle m'a également appris à travailler dans un contexte open source, avec une attention particulière portée à la qualité et aux performances des applications.

## VI.b. - Apport du stage dans mon projet professionnel

Ce stage m'a permis de me plonger dans un projet réel, structuré, avec un impact concret sur des utilisateurs, et une exigence forte en termes de rigueur technique. J'ai pris conscience de l'importance de comprendre le fonctionnement global d'une application du backend au frontend et d'intégrer les contraintes fonctionnelles dans la mise en œuvre technique.

Ce stage alimente naturellement mon ambition de devenir architecte logiciel, un rôle qui exige une vision d'ensemble, une capacité à structurer et faire évoluer une base de code, et une compréhension approfondie des différents niveaux d'un système.

# CONCLUSION

Ce stage a représenté une opportunité stimulante et formatrice, à la croisée du développement web, des systèmes d'information géographiques, et des problématiques d'architecture logicielle. Travailler sur la migration de Leaflet vers MapLibre dans un projet open source comme django-mapentity m'a permis de m'immerger dans un code existant complexe, d'en comprendre la logique, et d'y apporter une refonte moderne, en cohérence avec les besoins de performance, de maintenabilité recherché par l'entreprise.

L'expérience acquise au cours de ce stage m'a permis d'évoluer à la fois sur le plan technique et sur ma posture professionnelle. J'ai appris à analyser, découper et conduire un chantier technique de fond, en me formant sur des outils que je ne maîtrisais pas à l'origine, et en collaborant efficacement avec des encadrants expérimentés.

Ce stage s'inscrit pleinement dans mon projet de devenir architecte logiciel, en m'amenant à prendre de la hauteur sur les interactions entre couches applicatives, les choix techniques durables, et les besoins métiers sous-jacents. Il constitue une étape clé dans mon parcours, et une base solide pour la suite de ma spécialisation systèmes d'information et réseaux.

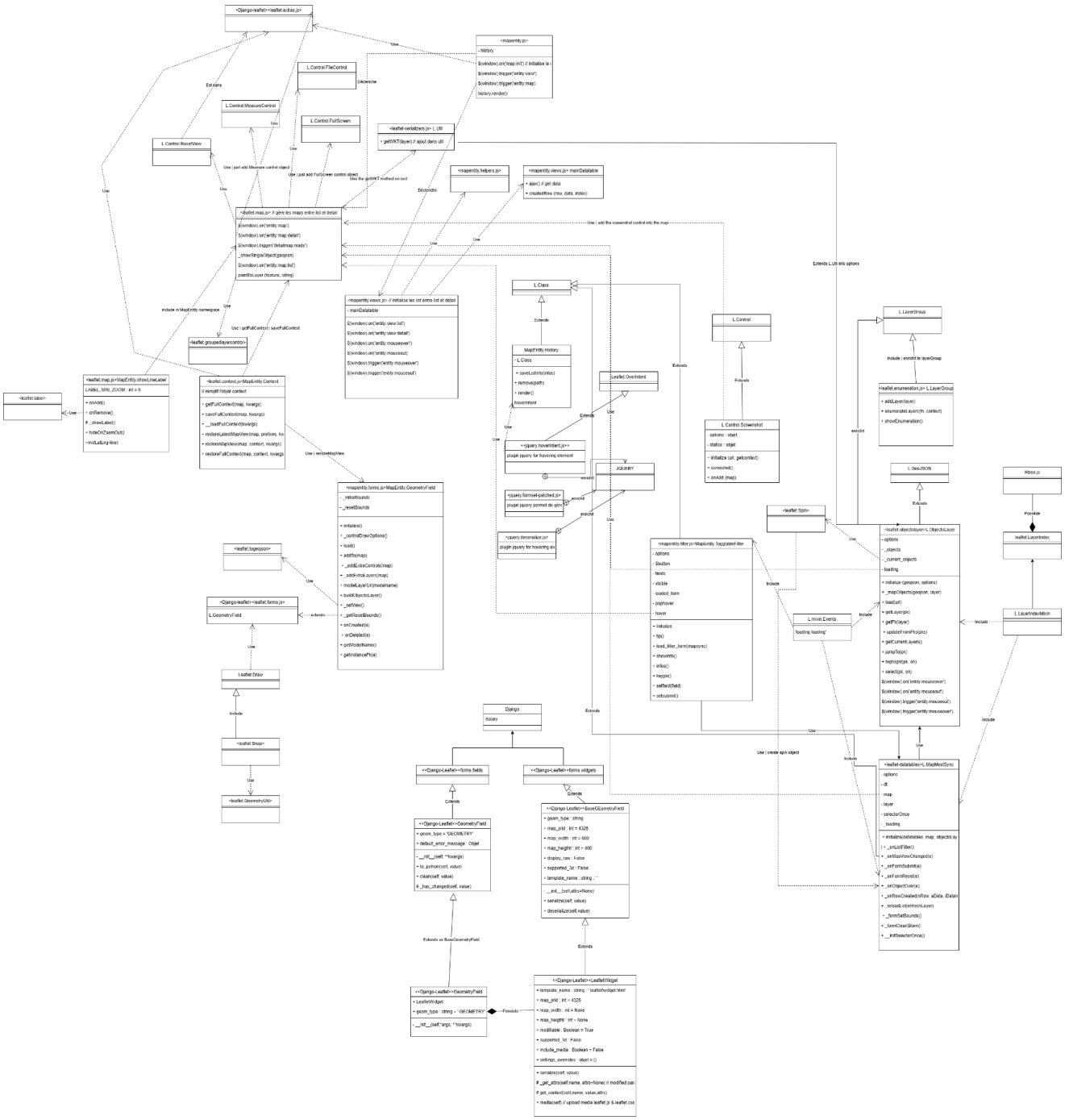

*Figure 13 : Schématisation de l'utilisation et inclusion des classes dans les différentes fichiers*

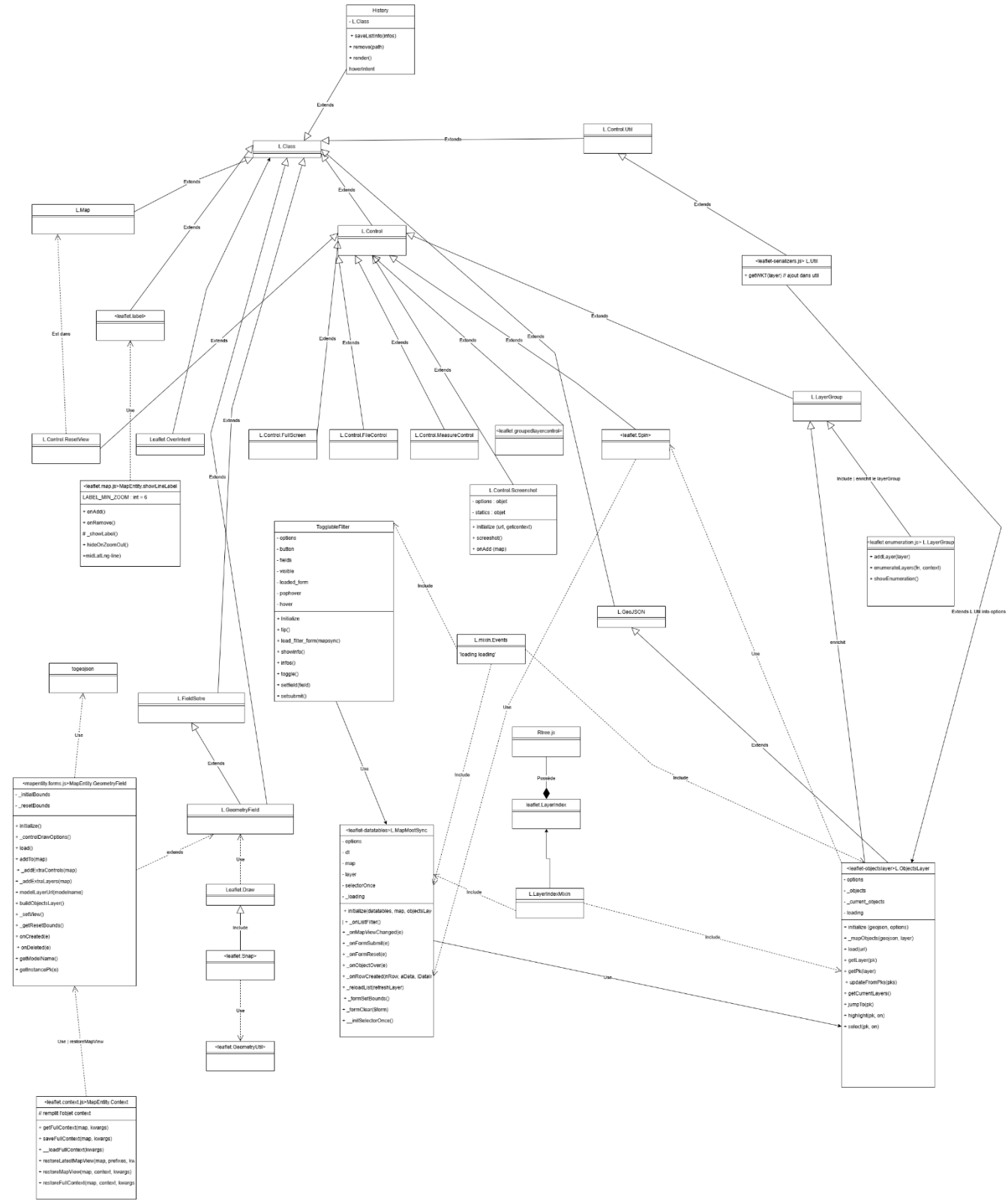


*Figure 14 : Diagramme de classe - Mapentity avec django-leaflet*

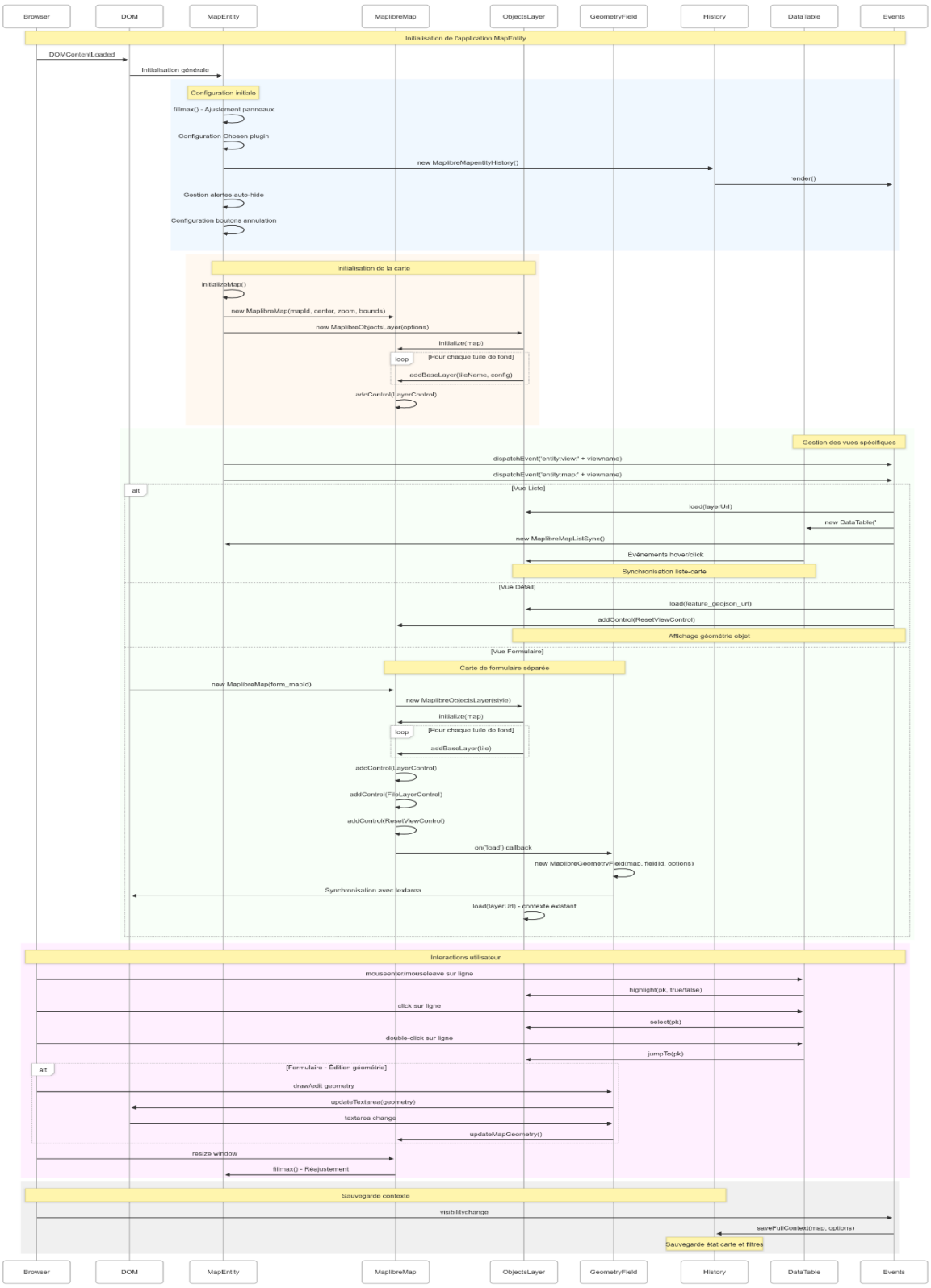


*Figure 15 : Diagramme de séquence - Mapentity avec Maplibre*

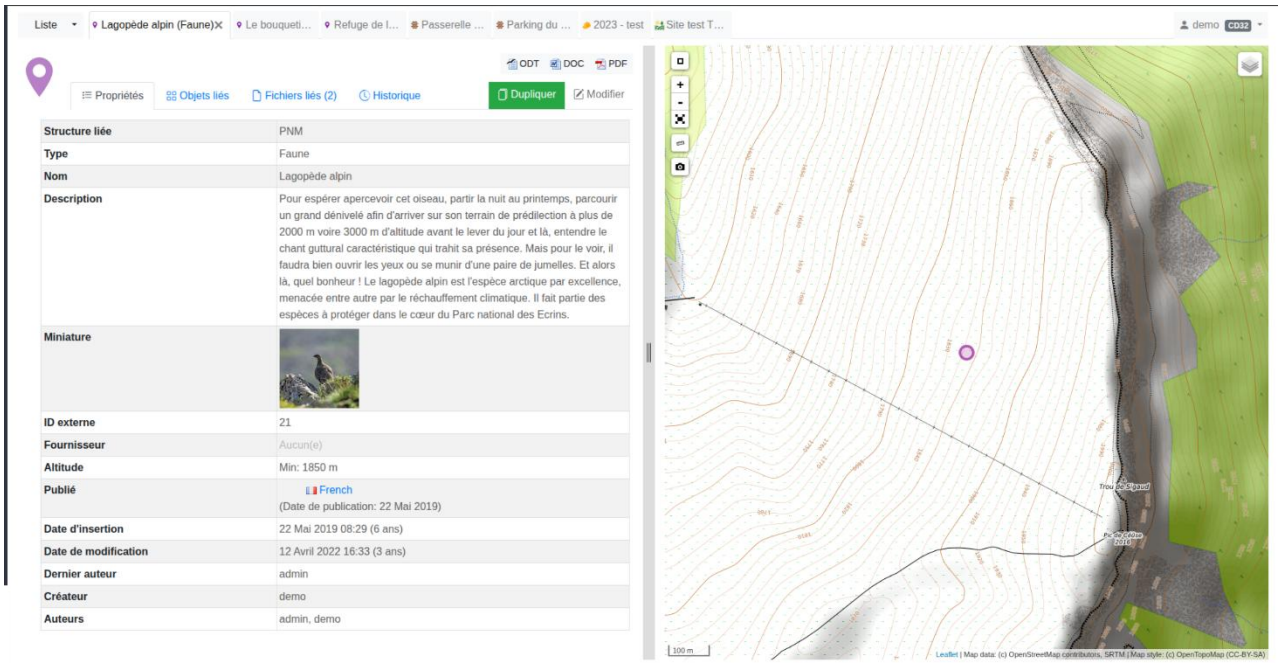


*Figure 16 : Interface graphique de la vue détail (Geotrek)*

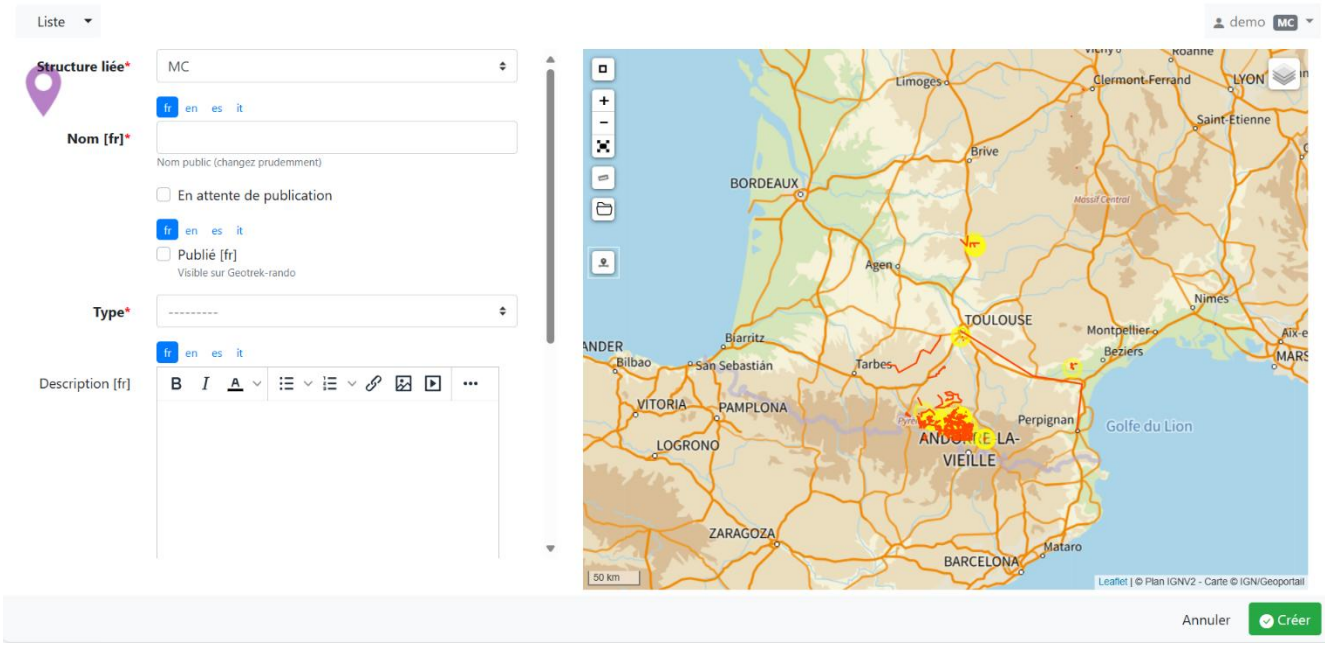


*Figure 17 : Interface graphique de la vue formulaire (Geotrek)*

# BIBLIOGRAPHIE


1. Corpus, M. (s. d.). *MapEntity Documentation*.

2. *Documentation—Leaflet—A JavaScript library for interactive maps*. (s. d.). Consulté 18 juin 2025, à l'adresse https://leafletjs.com/reference.html

3. *GeoDjango | Django documentation*. (s. d.). Django Project. Consulté 19 juin 2025, à l'adresse https://docs.djangoproject.com/en/5.2/ref/contrib/gis/

4. *GeoDjango Tutorial | Django documentation*. (s. d.). Django Project. Consulté 16 juin 2025, à l'adresse https://docs.djangoproject.com/en/5.2/ref/contrib/gis/tutorial/

5. Leplatre, M. (s. d.). *Des cartes avec GeoDjango et Leaflet | Makina Corpus*. Consulté 16 juin 2025, à l'adresse https://makina-corpus.com/logiciel-libre/des-cartes-avec-geodjango-et-leaflet

6. *Makina corpus/django-mapentity*. (2025). [JavaScript]. Makina Corpus. https://github.com/makinacorpus/django-mapentity (Édition originale 2013)

7. *MapLibre GL JS*. (s. d.). Consulté 18 juin 2025, à l'adresse https://www.maplibre.org/maplibre-gl-js/docs/

8. *Qu'est-ce qu'un SIG ? | Technologie de cartographie de système d'information géographique*. (s. d.). Consulté 7 juin 2025, à l'adresse https://www.esri.com/fr-fr/what-is-gis/overview

9. *Représentation et modélisation des informations géographiques dans un SIG | ArcGIS Resource Center*. (s. d.). Consulté 7 juin 2025, à l'adresse https://resources.arcgis.com/fr/help/getting-started/articles/026n0000000r000000.htm

10. SIG [Définition & Application] Système d'Information Géographique – smappen. (s. d.). *smappen*. Consulté 7 juin 2025, à l'adresse https://www.smappen.fr/sig/

11. *Welcome to Django Leaflet's documentation ! —Django Leaflet 0.20 documentation*. (s. d.). Consulté 16 juin 2025, à l'adresse https://django-leaflet.readthedocs.io/en/latest/

12. Bastion Potiron, (s. d.), Avec Geotrek, gérez, administrez et valorisez vos sentiers et activités touristiques | Makina Corpus. Consulté16 juin 2025.

13. Makina Corpus, Présentation Makina Corpus, Consulté 16 Juin 2025.

# RESUME

Dans le cadre de mon stage de 4e année chez Makina Corpus, je participe actuellement à la migration du moteur cartographique de django-mapentity, utilisé notamment par la plateforme Geotrek-admin. L'objectif principal est de moderniser l'affichage des cartes en remplaçant Leaflet par MapLibre GL JS, afin de réduire la dette technique liée à l'utilisation de bibliothèques obsolètes et de préparer l'application à une meilleure montée en charge.

Après une étude approfondie de l'architecture existante, j'ai entrepris la réécriture complète des composants JavaScript liés à l'affichage des listes, des détails et des formulaires cartographiques. Le projet est structuré selon une architecture orientée objets, ce qui facilite la maintenabilité et la clarté du code.

Ce stage, qui se poursuit encore pour un mois, m'offre une expérience précieuse pour mieux comprendre les enjeux du développement logiciel dans un contexte professionnel, en alliant modernisation technique et respect des fonctionnalités existantes. Il me permet également de découvrir le fonctionnement d'un projet open source et de consolider mes compétences en développement frontend.

### Mots-clés

SIG web, MapLibre, Leaflet, Django, cartographie interactive, open source, Geotrek-admin, frontend, Python, développement logiciel, Django-mapentity, Django-leaflet

# ABSTRACT

As part of my fourth-year internship at Makina Corpus, I am currently working on migrating the mapping engine of django-mapentity, which is notably used by the Geotrek-admin platform. The main goal is to modernize map rendering by replacing Leaflet 0.7 with MapLibre GL JS, in order to reduce technical debt from outdated libraries and to better prepare the application for scalability.

After an in-depth analysis of the existing architecture, I have undertaken a complete rewrite of the JavaScript components responsible for displaying lists, details, and map forms. The project is structured following an object-oriented architecture, which improves code clarity and maintainability.

This internship, which will continue for another month, is giving me valuable experience in understanding the challenges of software development in a professional setting, balancing technical modernization with the need to preserve existing features. It is also allowing me to discover how an open source project works and to further strengthen my frontend development skills.

### Keywords

Web GIS, MapLibre, Leaflet, Django, interactive mapping, open source, Geotrek-admin, front-end, Python, software development, Django-mapentity, Django-leaflet