RAPPORT DÉTAILLÉ DU PROJET WEBGIS

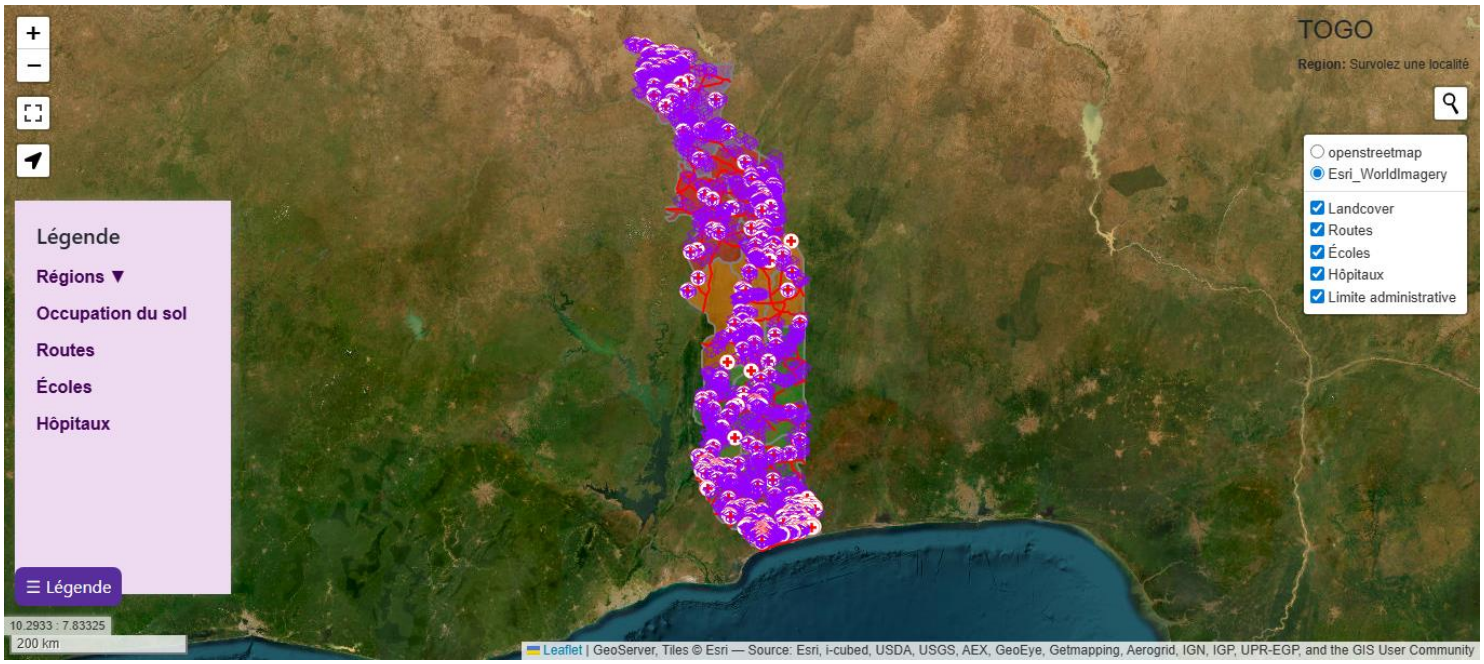
UE : Intégration et publication des données géospatiales

1. Introduction générale

Dans le cadre de l'unité d'enseignement Intégration et publication des données géospatiales, ce projet vise à concevoir une plateforme WebGIS interactive permettant la visualisation et l'interrogation de données géographiques à l'aide de la bibliothèque ***Leaflet*** *et du serveur cartographique* ***GeoServer****.*

L'objectif principal *est de démontrer la capacité à :*

- *Publier des données géospatiales via des services web (WMS/WFS),*
- *Intégrer ces services dans une application web,*
- *Combiner des données distantes (GeoServer) et locales (GeoJSON),*
- *Offrir une interface utilisateur interactive et ergonomique.*



Vue générale de la plateforme WebGIS après chargement complet

2. Architecture globale de la plateforme

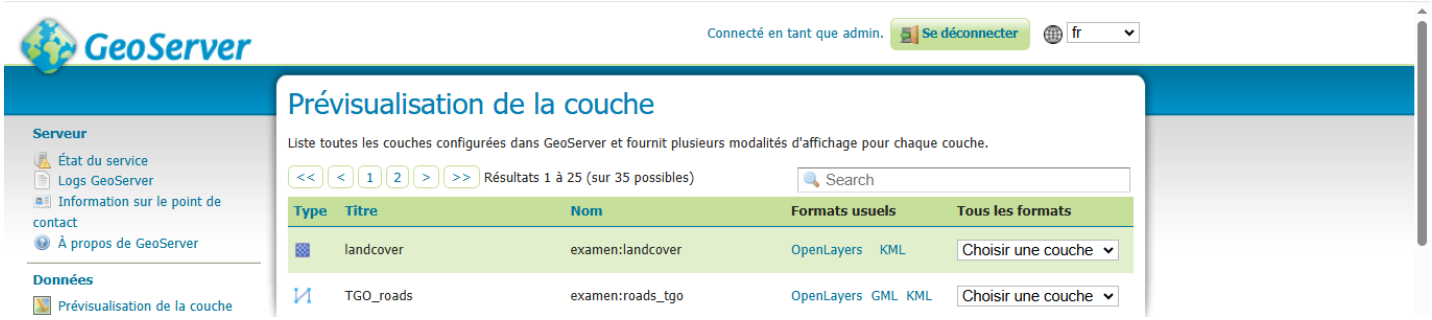
La plateforme repose sur une architecture ***client–serveur*** *simple :*

2.1. Côté client (Front-end)

- *HTML : structure de la page*
- *CSS : mise en forme et ergonomie*
- *JavaScript : logique applicative*
- *Leaflet : moteur cartographique*
- *Plugins Leaflet : fonctionnalités avancées*

2.2. Côté serveur

- *GeoServer :*
    - *Publication des couches géospatiales*
    - *Fourniture des services WMS et WFS*
- *Données stockées dans un Workspace nommé* ***examen***



Interface GeoServer montrant le Workspace examen et les couches publiées

3. Description détaillée des technologies utilisées

3.1. HTML (Structure de l'application)

Le fichier HTML constitue le squelette de l'application.
Il permet :

- *D'initialiser la carte via une balise* *** ,***
- *De charger les feuilles de style (CSS),*
- *D'importer les bibliothèques Leaflet et plugins,*
- *D'intégrer les scripts JavaScript.*

Extrait clé :

*** ***

Ce conteneur est indispensable car Leaflet y injecte dynamiquement la carte.

```html
<body>
    <div id="map"></div>
```

Code HTML montrant l'initialisation de la carte

3.2. CSS (Présentation et ergonomie)

Le CSS est utilisé pour :

- *Fixer la hauteur et la largeur de la carte,*
- *Styliser la légende (sidebar),*
- *Améliorer l'ergonomie (boutons, couleurs, icônes).*

Rôle du sidebar :

- *Afficher une légende dynamique,*
- *Permettre une lecture claire des couches affichées.*

```css
/* style.css > ... */
#map {
  height: 100vh;
  width: 100%;
}

/* ----- SIDEBAR ----- */
.sidebar {
  position: absolute;
  left: 10px;
  bottom: 10%;
  width: 200px;
  height: 60%;
  box-shadow: 2px 0 10px rgba(238, 233, 233, 0.2);
  padding: 20px;
  background-color: rgb(238, 218, 241);
  transform: translateX(-100%);
  transition: 0.35s ease;
  z-index: 9999;
}

.sidebar.open {
  transform: translateX(0);
}

.sidebar-content {
  overflow-y: auto;
  height: 80%;
  width: 107%;
}
```

```css
/* style.css > ... */
.sidebar-toggle {
  position: absolute;
  left: 10px;
  bottom: 50px;
  z-index: 10000;
  background: #582e9c;
  color: white;
  padding: 6px 10px;
  border: none;
  cursor: pointer;
  border-radius: 8px;
}

#legend {
  margin-top: 15px;
  font-family: Arial, sans-serif;
}

.legend-section-title {
  font-weight: bold;
  margin-top: 10px;
  cursor: pointer;
  user-select: none;
  color: #440358;
}

.legend-content {
  margin-left: 0;
  padding-left: 5px;
  display: block;
  color: #440358;
}
```

```css
/* style.css > ... */
.legend-content {
}

.legend-item {
  display: flex;
  align-items: center;
  margin-bottom: 6px;
}

.legend-color {
  width: 18px;
  height: 18px;
  border-radius: 4px;
  margin-right: 8px;
  border: 1px solid #1b068fe1;
}

.legend-label {
  font-size: 10px;
  color: #dffcf3;
}

.legend-icon {
  width: 18px;
  height: 18px;
  margin-right: 8px;
}
```

Légende personnalisée affichée dans le sidebar

4. Mise en place de la carte Leaflet

4.1. Initialisation de la carte

```javascript
// script.js > "Esri_WorldImagery"
let map = L.map('map', {
    zoomControl: true,
    fullscreenControl: true,
    fullscreenControlOptions: {
        position: 'topright'
    }
}).setView([8.32, 1], 7);
```

Explication :

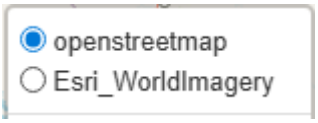
- *L.map() crée la carte Leaflet*
- *setView() centre la carte sur le Togo*
- *fullscreenControl permet l'affichage plein écran*

4.2. Fonds de carte (BaseMap)

Deux fonds de carte sont proposés :

- *OpenStreetMap*
- *Esri World Imagery*

```
baseLayers = {
    "OpenStreetMap": openstreetmap,
    "Esri Imagery": Esri_WorldImagery
};
```



Commutateur de fonds de carte

5. Intégration des couches géospatiales

5.1. Couche togo_admin2 (GeoJSON)

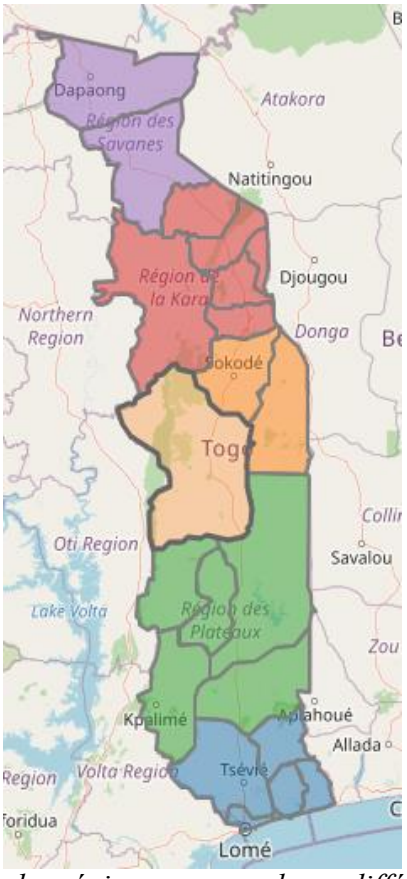
Cette couche est chargée localement via fetch().

fetch("togo_admin2.geojson")

Un style thématique est appliqué selon la région administrative.

Objectif :

- *Visualiser les limites administratives,*
- *Appliquer une sémiologie cartographique adaptée,*
- *Permettre l'interaction (survol, zoom).*



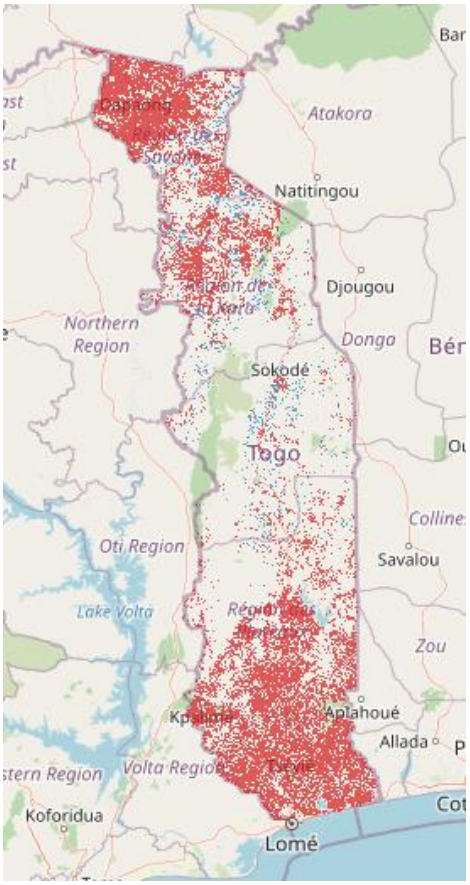
Carte des régions avec couleurs différenciées

5.2. Couche landcover (WMS – GeoServer)

```
const LandLayer = L.tileLayer.wms("http://localhost:8080/geoserver/wms", {
    layers: "examen:landcover",
    format: 'image/png',
    uppercase: true,
    transparent: true,
    continuousWorld: true,
    tiled: true,
    info_format: 'text/html',
    opacity: 0.7,
    identify: false,
    version: "1.1.0",
    attribution: "GeoServer"
});
```

Rôle du WMS :

- *Affichage raster*
- *Consultation visuelle*
- *Pas de modification côté client*



Couche d'occupation du sol affichée sur la carte

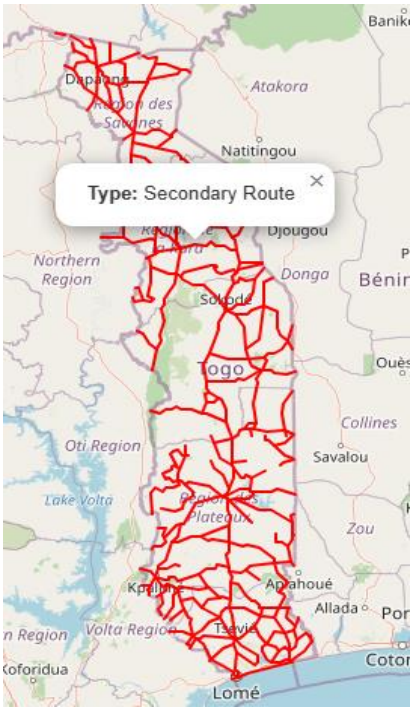
5.3. Couche roads_tgo (WFS – GeoServer)

La couche vecteur est chargée dynamiquement via le service WFS.

```
//Geoserver Web Feature Service
let url = "http://localhost:8080/geoserver/examen/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=examen:roads_tgo&outputFormat=application/json"
fetch(url)
    .then(res => res.json())
    .then(data => {
        road = L.geoJSON(data, {
            style: {
                color: "red",
                weight: 2
            },
            onEachFeature: (feature, layer) => {
                layer.bindPopup(`
                    <b>Type:</b> ${feature.properties.TYPE_ROUTE}
                `);
            }
        });
        checkLayersLoaded();
    });
```

Avantages du WFS :

- *Accès aux entités*
- *Affichage interactif*
- *Popups attributaires*



Routes affichées avec popup d'information

6. Couches « Écoles » et « Hôpitaux » – Données ponctuelles GeoJSON locales

6.1. Origine et nature des données

*Les couches **Écoles** et **Hôpitaux** représentent respectivement les infrastructures éducatives et sanitaires du TOGO.*
*Ces deux couches sont fournies sous forme de **fichiers GeoJSON locaux** (school.geojson et hospital.geojson) et contiennent des entités **ponctuelles**.*

Le choix du format GeoJSON permet :

- *Une intégration directe dans Leaflet,*
- *Une manipulation aisée côté client,*
- *Une indépendance vis-à-vis d'un serveur cartographique.*

6.2. Méthode d'intégration dans l'application WebGIS

*Les deux couches sont intégrées suivant la **même logique de chargement**, via l'API fetch() et la fonction L.geoJSON() de Leaflet.*

```
fetch("school.geojson")
    .then(response => response.json())
    .then(data => {
        school = L.geoJSON(data, {
            pointToLayer: function (feature, latlng) {
                return L.marker(latlng, { icon: ecoleIcon }).bindPopup(feature.properties.name);
            }
        });
        checkLayersLoaded();
    });
```

```
fetch("hospital.geojson")
    .then(response => response.json())
    .then(data => {
        hospital = L.geoJSON(data, {
            pointToLayer: function (feature, latlng) {
                return L.marker(latlng, { icon: hopitalIcon }).bindPopup(feature.properties.name);
            }
        });
        checkLayersLoaded();
    });
```

6.3. Rôle détaillé des composants utilisés

Élément	***Rôle***
fetch()	*Chargement asynchrone des fichiers GeoJSON*
response.json()	*Conversion des données en objets JavaScript*

L.geoJSON()	*Transformation en couches Leaflet*
pointToLayer()	*Conversion des points en marqueurs*
L.icon()	*Création d'icônes personnalisées*
bindPopup()	*Affichage des informations au clic*

6.4. Symbolisation cartographique

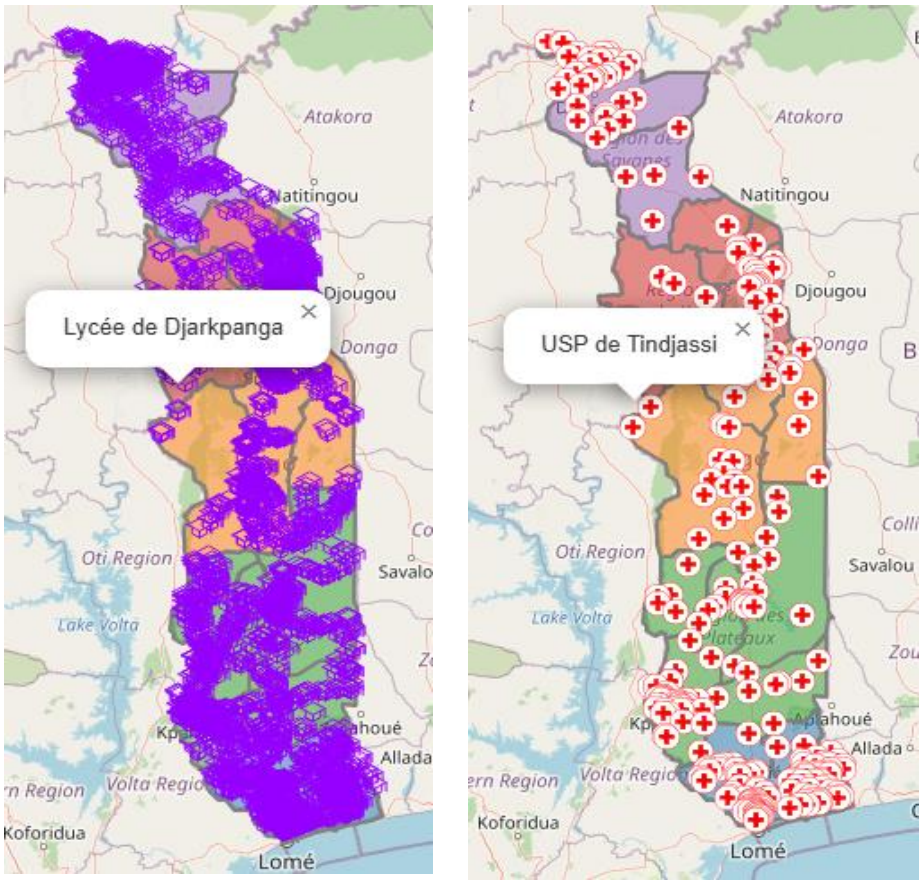
Afin de distinguer visuellement les deux types d'infrastructures, des ***icônes spécifiques*** *sont utilisées :*

```
var ecoleIcon = L.icon({
    iconUrl: 'school-outline-svgrepo-com.svg',
    iconSize: [15, 15],
    iconAnchor: [8, 15],
    popupAnchor: [0, -20]
});
```

```
var hopitalIcon = L.icon({
    iconUrl: 'hospital-first-aid-svgrepo-com.svg',
    iconSize: [15, 15],
    iconAnchor: [8, 15],
    popupAnchor: [0, -20]
});
```

Cette symbolisation :

- *Améliore la lisibilité de la carte,*
- *Respecte les principes de sémiologie graphique,*
- *Facilite l'interprétation par l'utilisateur.*



Affichage simultané des écoles et des hôpitaux avec leurs icônes respectives

7. Outils et contrôles Leaflet

Outil	*Rôle*
Géolocalisation	*Position de l'utilisateur*
Plein écran	*Meilleure visibilité*
Géocodage	*Recherche d'adresses*
Coordonnées souris	*Lecture spatiale*
Scale	*Référence métrique*
LayerSwitcher	*Gestion des couches*

8. Problème CORS : explication et solution

8.1. Origine du problème

Le navigateur bloque les requêtes WFS car :

- *La carte tourne sur http://127.0.0.1*
- *GeoServer tourne sur http://localhost:8080*

Ce sont deux ***origines différentes****.*

8.2. Solution recommandée

Activer CORS sur GeoServer

1. *Va dans le dossier WEB-INF de GeoServer (souvent geoserver/webapps/geoserver/WEB-INF/web.xml)*
2. *Ajoute ou décommente le filtre CORS :*

```xml
<filter>
    <filter-name>cross-origin</filter-name>
    <filter-class>org.eclipse.jetty.servlets.CrossOriginFilter</filter-class>
    <init-param>
        <param-name>allowedOrigins</param-name>
        <param-value>*</param-value>
    </init-param>
    <init-param>
        <param-name>allowedMethods</param-name>
        <param-value>GET,POST,OPTIONS,HEAD</param-value>
    </init-param>
</filter>

<filter-mapping>
    <filter-name>cross-origin</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

Fichier web.xml avec configuration CORS

- *Redémarre GeoServer après avoir modifié web.xml*

9. Conclusion générale

Ce projet WebGIS démontre une maîtrise avancée :

- *Des services web géospatiaux (WMS/WFS),*
- *De la bibliothèque Leaflet,*
- *De l'intégration serveur–client,*
- *De la conception d'interfaces cartographiques interactives.*

Malgré la problématique CORS, qui relève de l'administration serveur, la plateforme répond pleinement aux exigences pédagogiques et techniques de l'exercice.