260302 - Integration des outils Leaflet et appel des couches SIG extérieures (raster notamment)

C'est une excellente nouvelle ! La boucle est bouclée : ton serveur lit la base de données, l'API transmet les coordonnées, et ton navigateur dessine le résultat.

Pour passer à la vitesse supérieure, nous allons ajouter la capacité de **créer de nouvelles zones directement depuis la carte**. L'idée est d'utiliser un petit outil de dessin : tu dessines un polygone, tu lui donnes un nom, et "paf", il s'enregistre dans PostGIS.

Voici le plan :

1. **Côté Serveur** : Créer une route `POST /api/save-zone` pour recevoir le nouveau polygone.
2. **Côté Frontend** : Ajouter la barre d'outils **Leaflet.draw** pour dessiner sur la carte.

Étape 1 : La route d'enregistrement (`server.js`)

Ouvre ton fichier `server.js` et ajoute cette route juste avant le `app.listen` :

JavaScript

```
// --- SAUVEGARDER UNE NOUVELLE ZONE ---
app.post('/api/save-zone', async (req, res) => {
  try {
    const { nom, geojson } = req.body;
```

```
    // Requête SQL pour insérer le polygone
    // ST_GeomFromGeoJSON convertit le JSON du navigateur en géométrie PostGIS
    const query = `
      INSERT INTO zones_interet (nom, geom)
      VALUES ($1, ST_SetSRID(ST_GeomFromGeoJSON($2), 4326))
      RETURNING id;
    `;

    const result = await pool.query(query, [nom, JSON.stringify(geojson.geometry)]);

    res.json({ status: "success", message: "Zone enregistrée !", id: result.rows[0].id });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: "Erreur lors de la sauvegarde" });
  }
});
```

N'oublie pas de redémarrer avec `pm2 restart alteris-api` après avoir sauvegardé.

Étape 2 : L'interface de dessin (`index.html`)

Nous allons ajouter la bibliothèque **Leaflet.draw** dans ton fichier HTML.

1. Ouvre `public/index.html`.
2. Ajoute les liens CSS/JS de Leaflet.draw dans le `<head>` et avant la fin du `<body>`.
3. Ajoute la logique de sauvegarde.

Voici la partie du code à modifier/ajouter dans ton script :

HTML

```
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/leaflet.draw/1.0.4/leaflet.draw.css"/>

<script
```

```
src="https://cdnjs.cloudflare.com/ajax/libs/leaflet.draw/1.0.4/leaflet.draw.js"></script>

<script>
    // ... (ton code existant) ...

    // 5. Configuration de l'outil de dessin
    const drawnItems = new L.FeatureGroup();
    map.addLayer(drawnItems);

    const drawControl = new L.Control.Draw({
        edit: { featureGroup: drawnItems },
        draw: {
            polygon: true,   // On autorise les polygones
            polyline: false,
            rectangle: true,
            circle: false,
            marker: false
        }
    });
    map.addControl(drawControl);

    // 6. Action quand le dessin est terminé
    map.on(L.Draw.Event.CREATED, async function (e) {
        const layer = e.layer;
        const geojson = layer.toGeoJSON();

        const nomZone = prompt("Nom de cette nouvelle zone ?");
        if (!nomZone) return;

        // Envoi au serveur
        try {
            const reponse = await fetch('/api/save-zone', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify({ nom: nomZone, geojson: geojson })
            });
            const res = await reponse.json();

            if(res.status === "success") {
                alert("Zone '" + nomZone + "' enregistrée avec succès !");
```

```
                location.reload(); // On rafraîchit pour voir la nouvelle zone orange
            }
        } catch (err) {
            alert("Erreur lors de la sauvegarde.");
        }
    });
</script>
```

Comment tester ?

1. Rafraîchis ta page.
2. Tu verras une nouvelle **barre d'outils à gauche**.
3. Clique sur l'icône **Polygone**, dessine une forme sur la carte et ferme-la en cliquant sur le premier point.
4. Entre un nom (ex: "Zone Test 2").
5. La page va se recharger et ta nouvelle zone apparaîtra en orange, prête à être interrogée par un clic !

Final du server-.js

```
const express = require('express');
const bodyParser = require('body-parser');
const pool = require('./db');

const app = express();
const PORT = 3001;

app.use(bodyParser.json());
app.use(express.static('public'));

// 1. RÉCUPÉRER TOUTES LES ZONES
app.get('/api/zones', async (req, res) => {
  try {
    const query = `SELECT id, nom, ST_AsGeoJSON(geom) as geometrie FROM zones_interet;`;
```

```
    const result = await pool.query(query);
    const features = result.rows.map(row => ({
      type: "Feature",
      properties: { nom: row.nom },
      geometry: JSON.parse(row.geometrie)
    }));
    res.json({ type: "FeatureCollection", features: features });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: "Erreur chargement zones" });
  }
});

// 2. VÉRIFIER SI UN CLIC EST DANS UNE ZONE
app.post('/api/check-zone', async (req, res) => {
  try {
    const geoJsonInput = req.body;
    const query = `
      SELECT nom FROM zones_interet
      WHERE ST_Contains(geom, ST_SetSRID(ST_GeomFromGeoJSON($1), 4326));
    `;
    const result = await pool.query(query, [JSON.stringify(geoJsonInput)]);
    if (result.rows.length > 0) {
      res.json({ status: "success", message: `Le point est DANS : ${result.rows[0].nom}` });
    } else {
      res.json({ status: "success", message: "Le point est HORS zone." });
    }
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: "Erreur calcul spatial" });
  }
});

// 3. SAUVEGARDER UNE NOUVELLE ZONE
app.post('/api/save-zone', async (req, res) => {
  try {
    const { nom, geojson } = req.body;
    const query = `
      INSERT INTO zones_interet (nom, geom)
      VALUES ($1, ST_SetSRID(ST_GeomFromGeoJSON($2), 4326))
      RETURNING id;
    `;
    const result = await pool.query(query, [nom, JSON.stringify(geojson.geometry)]);
    res.json({ status: "success", id: result.rows[0].id });
  } catch (err) {
    console.error(err);
```

```
    res.status(500).json({ error: "Erreur sauvegarde" });
  }
});

app.listen(PORT, () => {
  console.log(`□□ Serveur Alteris prêt sur le port ${PORT}`);
});
```

Final du frontend - index.html

avec appel des images aériennes de Google (IGN par la suite)

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Alteris - SIG Expert</title>

    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" />
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/leaflet.draw/1.0.4/leaflet.draw.css"/>

    <style>
        body { margin: 0; padding: 0; font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; }
        #maCarte { width: 100vw; height: 100vh; background: #222; }

        .info-panel {
            position: absolute; top: 10px; right: 10px; z-index: 1000;
            background: rgba(255, 255, 255, 0.9); padding: 15px; border-radius: 8px;
            box-shadow: 0 0 15px rgba(0,0,0,0.3); max-width: 250px;
        }
        h3 { margin: 0 0 5px 0; color: #e67e22; }
        p { margin: 0; font-size: 0.9em; color: #333; }
    </style>
</head>
<body>

    <div class="info-panel">
        <h3>Alteris Geomatic</h3>
```

```
    <p><b>Dessin :</b> Utilisez les outils à gauche.</p>
    <p><b>Test :</b> Cliquez sur la carte pour vérifier si vous êtes dans une zone.</p>
</div>

<div id="maCarte"></div>

<script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet.draw/1.0.4/leaflet.draw.js"></script>

<script>
    // 1. DÉFINITION DES FONDS DE CARTE
    const osm = L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        attribution: '© OpenStreetMap'
    });

    const googleSat = L.tileLayer('http://{s}.google.com/vt/lyrs=s&x={x}&y={y}&z={z}', {
        maxZoom: 20,
        subdomains:['mt0','mt1','mt2','mt3'],
        attribution: '© Google Satellite'
    });

    // 2. INITIALISATION DE LA CARTE (Satellite par défaut)
    const map = L.map('maCarte', {
        center: [48.85, 2.35],
        zoom: 12,
        layers: [googleSat]
    });

    // 3. SÉLECTEUR DE COUCHES
    const baseMaps = {
        "Satellite (Google)": googleSat,
        "Plan (OSM)": osm
    };
    L.control.layers(baseMaps).addTo(map);

    // 4. GESTION DES DESSINS (Stockage temporaire à l'écran)
    const drawnItems = new L.FeatureGroup();
    map.addLayer(drawnItems);

    const drawControl = new L.Control.Draw({
        edit: { featureGroup: drawnItems },
        draw: {
            polygon: { allowIntersection: false, showArea: true },
            rectangle: true,
            polyline: false, circle: false, marker: false, circlemarker: false
        }
```

```
});
map.addControl(drawControl);

// 5. CHARGEMENT DES ZONES DEPUIS POSTGIS
async function chargerZones() {
    try {
        const reponse = await fetch('/api/zones');
        const data = await reponse.json();

        L.geoJSON(data, {
            style: {
                color: '#e67e22',
                weight: 3,
                fillOpacity: 0.3
            },
            onEachFeature: function (feature, layer) {
                layer.bindPopup("<b>Zone :</b> " + feature.properties.nom);
            }
        }).addTo(map);
    } catch (err) {
        console.error("Erreur chargement zones:", err);
    }
}

// 6. ENREGISTREMENT D'UNE NOUVELLE ZONE
map.on(L.Draw.Event.CREATED, async function (e) {
    const layer = e.layer;
    const geojson = layer.toGeoJSON();

    const nomZone = prompt("Nom de la nouvelle zone ?");
    if (!nomZone) return;

    try {
        const reponse = await fetch('/api/save-zone', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ nom: nomZone, geojson: geojson })
        });

        if (reponse.ok) {
            alert("Zone '" + nomZone + "' enregistrée avec succès !");
            location.reload(); // On rafraîchit pour voir la zone orange
        }
    } catch (err) {
        alert("Erreur lors de la sauvegarde sur le serveur.");
    }
```

```
    });

    // 7. VÉRIFICATION SPATIALE AU CLIC
    map.on('click', async function(e) {
        // On ne déclenche pas le test si l'utilisateur est en train de dessiner
        if (drawControl._toolbars.draw._activeMode) return;

        const point = {
            type: "Point",
            coordinates: [e.latlng.lng, e.latlng.lat]
        };

        try {
            const reponse = await fetch('/api/check-zone', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify(point)
            });
            const res = await reponse.json();

            L.popup()
                .setLatLng(e.latlng)
                .setContent(res.message)
                .openOn(map);
        } catch (err) {
            console.error("Erreur lors du clic spatial:", err);
        }
    });

    // Lancement au démarrage
    chargerZones();

  </script>
</body>
</html>
```

Revision #1
Created 2026-03-02 10:12:35 UTC by Julien

Updated 2026-03-02 10:23:22 UTC by Julien