🗺️ CLIOPE - Atlas de Energías Renovables

CLIOPE es un atlas web interactivo de energías renovables para la región de Mendoza, Argentina. Proporciona visualización geoespacial de datos sobre biomasa y energía solar.

📋 Índice

🎯 Descripción General

CLIOPE es una aplicación web estática que permite:

- **Visualización interactiva** de mapas con múltiples capas de datos
- **Análisis geoespacial** de recursos energéticos renovables
- **Navegación intuitiva** entre diferentes tipos de información
- **Interfaz responsive** para diferentes dispositivos

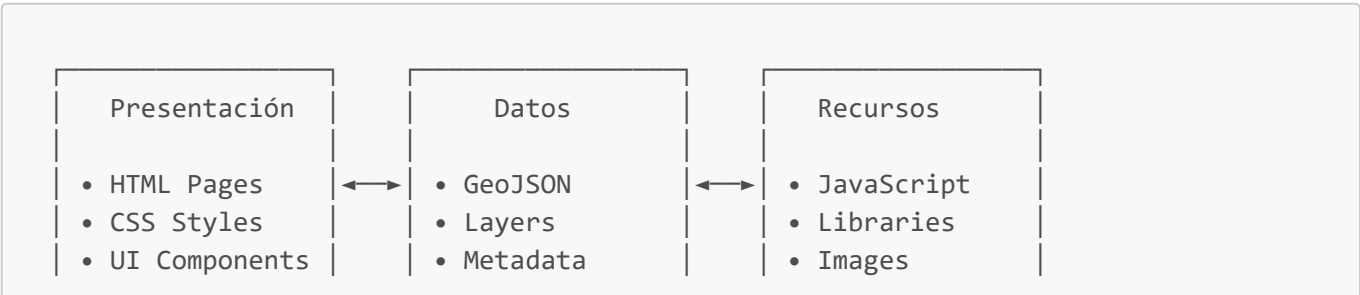
Alcance Geográfico

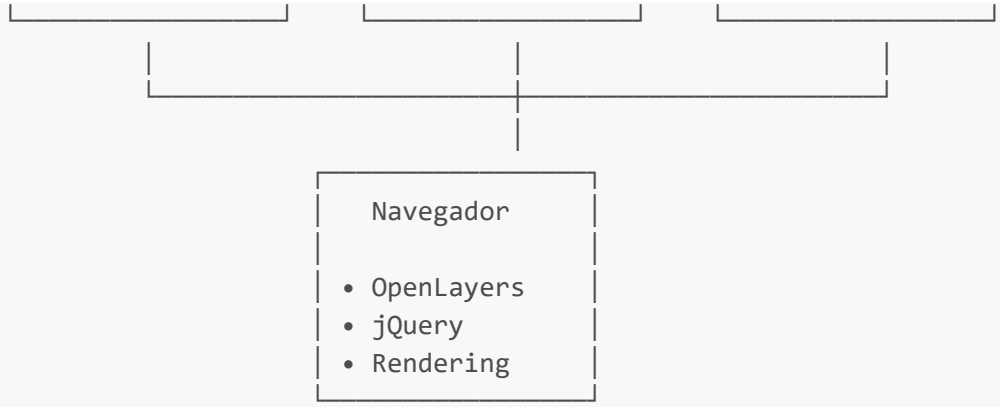
- **Región**: Mendoza, Argentina
- **Departamentos**: Todos los departamentos de la provincia
- **Coordenadas**: Sistema WGS84 (EPSG:4326)

Tipos de Energía

- **Biomasa**: Olivo, peral, plantaciones forestales, residuos agrícolas
- **Solar**: Radiación solar, potencial fotovoltaico
- **Infraestructura**: Plantas de procesamiento, bodegas, conserveras

🏗️ Arquitectura del Proyecto





Patrones de Diseño

- **MVC Implícito**: Separación entre vista (HTML/CSS), modelo (GeoJSON) y controlador (JS)
- **Modular**: Cada funcionalidad en archivos separados
- **Progresivo**: Carga de recursos bajo demanda

🗁 Estructura de Archivos

```
CLIOPE-DEFINITIVO/
│
├── 🗎 Páginas Principales
│   ├── index.html              # Página de inicio
│   ├── mapa.html               # Visualizador principal de mapas
│   ├── mapa2.html              # Vista alternativa de mapas
│   ├── biomasa.html            # Sección específica de biomasa
│   ├── solar.html              # Sección específica de energía solar
│   ├── acerca_de.html          # Información del proyecto
│   ├── manual.html             # Manual de usuario
│   ├── tecnico.html            # Documentación técnica
│   └── referencias.html        # Referencias y fuentes
│
├── 🎨 Estilos y Diseño
│   ├── css/
│   │   ├── main.css            # Estilos principales
│   │   ├── cliope.css          # Estilos específicos del proyecto
│   │   ├── normalize.css       # Reset CSS
│   │   └── own_style.css       # Estilos personalizados
│   │
│   ├── fonts/                  # Fuentes de iconos
│   │   ├── icomoon.eot
│   │   ├── icomoon.svg
│   │   ├── icomoon.ttf
│   │   └── icomoon.woff
│   │
│   └── images/                 # Recursos gráficos
│       ├── logo.png
│       ├── layers-2x.png
│       └── marker-*.png
│
```

```
├── 💾 Datos Geoespaciales
│   ├── data/                  # Datos específicos exportados
│   │   ├── exp_BiomasaOlivo.js
│   │   └── exp_BiomasaPeral.js
│   │
│   └── layers/                # Capas GeoJSON principales
│       ├── DepartamentosdeMendoza_0.js
│       ├── BiomasaOlivo_47.js
│       ├── BiomasaPeral_46.js
│       ├── EnergiaAlmendro_28.js
│       ├── Centralesdegeneracin_94.js
│       └── [130+ archivos de capas]
│
├── ⚙️ JavaScript y Lógica
│   ├── js/
│   │   ├── main.js            # Lógica principal de la aplicación
│   │   ├── leaflet-src.js     # Librería de mapas (fuente)
│   │   ├── leaflet.js         # Librería de mapas (minificada)
│   │   ├── leaflet-hash.js    # Plugin para URLs con hash
│   │   ├── jquerytoggle.js    # Controles de alternancia
│   │   └── Autolinker.min.js  # Conversión automática de enlaces
│   │
│   ├── jQuery/                # Librería jQuery
│   │   └── jquery.js
│   │
│   └── lib/                   # Librerías adicionales
│       └── [librerías varias]
│
├── 📚 Recursos y Documentación
│   ├── pdf/                   # Documentos en PDF
│   ├── pictures/              # Imágenes de referencia
│   ├── resources/             # Recursos adicionales
│   └── misc/                  # Archivos varios
│
└── 🛠️ Configuración
    ├── debug.log              # Logs de depuración
    └── EspecificacionDirectorios # Especificaciones del proyecto
```

🔧 Tecnologías Utilizadas

Frontend

Tecnología	Versión	Propósito
HTML5	-	Estructura de páginas
CSS3	-	Estilos y diseño responsive
JavaScript	ES5/ES6	Lógica de aplicación
jQuery	3.x	Manipulación DOM y eventos
OpenLayers	6.x	Mapas interactivos y GIS

Datos

Formato	Uso
GeoJSON	Datos geoespaciales principales
JavaScript Objects	Configuración de capas
JSON	Metadatos y configuraciones

Herramientas de Desarrollo

- **Ninguna** (Desarrollo vanilla)
- Editores recomendados: VS Code, Sublime Text
- Navegadores soportados: Chrome, Firefox, Safari, Edge

🗺️ Capas de Datos

Categorías Principales

1. División Administrativa

- `DepartamentosdeMendoza_0.js` - Límites departamentales
- Geometría: Polígonos
- Atributos: Nombre, código, área

2. Biomasa Agrícola

```
// Estructura típica de capa de biomasa
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [-68.8458, -32.8895]
      },
      "properties": {
        "nombre": "Zona Productiva",
        "tipo_biomasa": "Olivo",
        "potencial_mwh": 150.5,
        "superficie_ha": 1200,
        "departamento": "Maipú"
      }
    }
  ]
}
```

Tipos de Biomasa:

- **Frutales**: Olivo, Peral, Manzano, Duraznero, Cerezo
- **Frutos Secos**: Almendro, Nogal
- **Forestales**: Plantaciones comerciales
- **Residuos**: Agrícolas, ganaderos, industriales

3. Energía Solar

- Radiación solar acumulada (mensual)
- Potencial fotovoltaico
- Coordenadas de instalaciones

4. Infraestructura Energética

- Centrales de generación
- Plantas de procesamiento
- Red de distribución

Nomenclatura de Archivos

```
[TipoRecurso][Subtipo]_[ID].js

Ejemplos:
- BiomasaOlivo_47.js
- EnergiaAlmendro_28.js
- Centralesdegeneracin_94.js
```

⚡ Funcionalidades

Visualización de Mapas

- **Zoom y paneo** interactivo
- **Capas superpuestas** con transparencia
- **Controles de capas** para mostrar/ocultar
- **Popup informativos** al hacer clic
- **Leyendas** dinámicas por capa

Navegación

- **Menú principal** entre secciones
- **Breadcrumbs** para orientación
- **URLs con hash** para bookmarking
- **Navegación responsive** en móviles

Análisis de Datos

- **Filtrado** por departamento
- **Búsqueda** por texto
- **Estadísticas** agregadas

- **Comparación** entre regiones

Exportación

- **Capturas** de pantalla del mapa
- **Datos** en formato estándar
- **Reportes** personalizados

🚀 Instalación y Ejecución

Requisitos

- Navegador web moderno
- Servidor web local (opcional pero recomendado)

Método 1: Apertura Directa

```
# Simplemente abrir en navegador
start index.html  # Windows
open index.html   # macOS
xdg-open index.html # Linux
```

Método 2: Servidor Local

```
# Con Python
python -m http.server 8000
# Luego ir a http://localhost:8000

# Con Node.js
npx http-server
# Luego ir a http://localhost:8080

# Con VS Code Live Server
# Instalar extensión Live Server
# Clic derecho en index.html → "Open with Live Server"
```

Método 3: Hosting Web

```
# Subir todos los archivos a servidor web
# Configurar servidor para servir archivos estáticos
# Asegurar que extensión .js sea servida correctamente
```

🔧 Gestión de Capas

Agregar Nueva Capa

1. Preparar Datos GeoJSON

```
// nuevo_recurso.geojson
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [-68.123, -32.456]
      },
      "properties": {
        "nombre": "Sitio de Interés",
        "valor": 123,
        "categoria": "nueva"
      }
    }
  ]
}
```

2. Convertir a Formato CLIOPE

```
// layers/NuevoRecurso_XX.js
var json_NuevoRecurso_XX = {
  "type": "FeatureCollection",
  "name": "NuevoRecurso",
  "features": [
    // ... datos GeoJSON
  ]
}
```

3. Integrar en Mapa

```
// En mapa.html o mapa2.html
var nuevaCapaLayer = new ol.layer.Vector({
  source: new ol.source.Vector({
    features: (new ol.format.GeoJSON()).readFeatures(json_NuevoRecurso_XX)
  }),
  style: estiloPersonalizado
});

map.addLayer(nuevaCapaLayer);
```

4. Agregar Controles

```
<!-- En el panel de capas -->
<label>
  <input type="checkbox" onchange="toggleLayer('nuevo-recurso')">
  Nuevo Recurso
</label>
```

Modificar Capa Existente

1. Localizar Archivo

```
# Buscar en layers/
grep -l "nombre_recurso" layers/*.js
```

2. Editar Datos

```
// Modificar propiedades en layers/RecursoExistente_XX.js
"properties": {
  "nombre": "Nuevo Nombre",
  "valor_actualizado": 456,
  "fecha_actualizacion": "2025-09-03"
}
```

3. Actualizar Referencias

```
// Verificar que el nombre de variable coincida
var json_RecursoExistente_XX = {
  // datos actualizados
}
```

🔧 API de Componentes

Gestión de Mapas

```
// Inicializar mapa
function initMap(targetId, centerCoords, zoomLevel) {
  const map = new ol.Map({
    target: targetId,
    layers: [
      new ol.layer.Tile({
        source: new ol.source.OSM()
      })
    ],
```

```
    view: new ol.View({
      center: ol.proj.fromLonLat(centerCoords),
      zoom: zoomLevel
    })
  });
  return map;
}

// Agregar capa
function addGeoJSONLayer(map, geoJsonData, style) {
  const layer = new ol.layer.Vector({
    source: new ol.source.Vector({
      features: (new ol.format.GeoJSON()).readFeatures(geoJsonData)
    }),
    style: style
  });
  map.addLayer(layer);
  return layer;
}

// Controlar visibilidad
function toggleLayerVisibility(layer, visible) {
  layer.setVisible(visible);
}
```

Gestión de Estilos

```
// Estilo por defecto
function getDefaultStyle() {
  return new ol.style.Style({
    fill: new ol.style.Fill({
      color: 'rgba(255, 255, 255, 0.6)'
    }),
    stroke: new ol.style.Stroke({
      color: '#319FD3',
      width: 1
    }),
    image: new ol.style.Circle({
      radius: 5,
      fill: new ol.style.Fill({
        color: '#319FD3'
      })
    })
  });
}

// Estilo basado en propiedades
function getPropertyBasedStyle(feature) {
  const value = feature.get('potencial_mwh');
  const color = value > 100 ? '#ff0000' : '#00ff00';
```

```
  return new ol.style.Style({
    fill: new ol.style.Fill({ color: color }),
    stroke: new ol.style.Stroke({ color: '#000', width: 1 })
  });
}
```

Eventos y Interacciones

```
// Click en feature
map.on('click', function(evt) {
  map.forEachFeatureAtPixel(evt.pixel, function(feature) {
    const properties = feature.getProperties();
    showPopup(properties, evt.coordinate);
  });
});

// Hover effects
map.on('pointermove', function(evt) {
  const feature = map.forEachFeatureAtPixel(evt.pixel, function(feature) {
    return feature;
  });

  if (feature) {
    map.getTargetElement().style.cursor = 'pointer';
  } else {
    map.getTargetElement().style.cursor = '';
  }
});
```

🛠 Mantenimiento

Respaldos Regulares

```
# Crear respaldo con fecha
backup_name="cliope_backup_$(date +%Y%m%d)"
tar -czf "${backup_name}.tar.gz" CLIOPE-DEFINITIVO/

# Respaldar solo datos críticos
zip -r layers_backup.zip layers/ data/
```

Validación de Datos

```
// Verificar integridad de GeoJSON
function validateGeoJSON(data) {
  try {
    const parsed = JSON.parse(data);
```

```
    return parsed.type === 'FeatureCollection' &&
           Array.isArray(parsed.features);
  } catch (e) {
    return false;
  }
}

// Verificar coordenadas
function validateCoordinates(coordinates) {
  const [lng, lat] = coordinates;
  return lng >= -180 && lng <= 180 && lat >= -90 && lat <= 90;
}
```

Optimización de Performance

```
// Clustering para muchos puntos
function addClusteredLayer(source) {
  const clusterSource = new ol.source.Cluster({
    distance: 40,
    source: source
  });

  return new ol.layer.Vector({
    source: clusterSource,
    style: getClusterStyle
  });
}

// Lazy loading de capas
function loadLayerOnDemand(layerId) {
  if (!loadedLayers.has(layerId)) {
    // Cargar solo cuando sea necesario
    loadGeoJSONLayer(layerId).then(layer => {
      map.addLayer(layer);
      loadedLayers.set(layerId, layer);
    });
  }
}
```

Monitoreo y Logs

```
// Sistema de logging simple
const Logger = {
  info: (message) => console.log(`[INFO] ${new Date().toISOString()}:
${message}`),
  warn: (message) => console.warn(`[WARN] ${new Date().toISOString()}:
${message}`),
  error: (message, error) => {
    console.error(`[ERROR] ${new Date().toISOString()}: ${message}`, error);
```

```
    // Opcional: enviar a servicio de monitoreo
  }
};

// Uso
Logger.info('Mapa inicializado correctamente');
Logger.warn('Capa no encontrada, usando respaldo');
Logger.error('Error al cargar datos', error);
```

🤝 Contribución

Estructura para Nuevos Desarrolladores

1. **Clonar** repositorio y familiarizarse con estructura
2. **Probar** funcionalidad existente en navegador
3. **Identificar** área de mejora o nueva funcionalidad
4. **Desarrollar** siguiendo patrones existentes
5. **Documentar** cambios y nuevas funcionalidades

Estándares de Código

- **Nomenclatura**: camelCase para variables, PascalCase para constructores
- **Comentarios**: JSDoc para funciones principales
- **Indentación**: 2 espacios
- **Compatibilidad**: ES5 para máxima compatibilidad

CLIOPE - Atlas de Energías Renovables para Mendoza
Versión: 1.0 | Última actualización: Septiembre 2025