Documentación metodológica de código fuente 2025

Tabla de Contenido

Introducción **2**
Marco de Referencia **3**
Configuración de base de datos espacial PostgreSQL **3**
1. Alistamiento de capas 4
2. Creación y actualización de tablas en postgreSQL 5
Creación de capas en Geoserver **5**
1. Configuración de Espacios de trabajo, Almacenes de datos y Capas en Geoserver 5
2. Configuración de Estilos en Geoserver 5
Desarrollo y despliegue de sitios web estáticos para productos WebMap, StoryMaps y Experiencias de usuario específicas **6**
1. Desarrollo de sitios Web estáticos para WebMaps 6
2. Desarrollo de sitios Web estáticos para generación de interfaz independiente y creación de StoryMaps 7
Configuración Inicial 8
Configuración de GeoServer 8
Creación del Story Map 8
Sincronización del Mapa y la Presentación 9
Publicación y Verificación 9
Código de las plantillas de storymaps, centrado en el mapa y la principal de centrado en diapositivas **9**
Plantilla centrada en el mapa: 10
Plantilla principal o centrada en las diapositivas: 10
Conclusión 10
3. Desarrollo de sitios Web estáticos para creación de experiencias de usuario específicas como imágenes desplazándose en forma horizontal como insumo de narrativa 10
Configuración Inicial 10
Creación de la Página Web Estática 11
Publicación y Verificación 11

Conclusión 11
4. Desarrollo de sitios Web estáticos para creación de Storymaps 12
Generación de Story Maps mediante interfaz propia generada en sitio web estático 12
5. Desarrollo de sitios Web estáticos para creación de Webmaps 13
Desarrollo y despliegue de inventario general de Contenido de Arcgis Online mediante API de arcgis usando código python 14
Desarrollo de funcionalidad girar mapa 90 grados más brújula de indicador del Norte geográfico 14
Desarrollo de funcionalidad yuxtaponer capas (con slider izquierda a derecha apagar y prender capas) 17
1. Contexto Inicial 17
2. Análisis del Código Inicial 17
3. Solución Implementada 18
3.1. Modificación del HTML 18
3.2. Ajustes en el CSS 18
3.3. Incorporación del Control Deslizante al JavaScript 18
4. Resultado 19

Introducción

Para soportar el repositorio de información geográfica de la Secretaría de Cultura, Recreación y Deporte SCRD y centralizar tanto la información geográfica como los metadatos asociados a ella se utiliza GeoServer que es una poderosa herramienta de código abierto para compartir, analizar y editar datos geoespaciales. Cuando se combina con PostGIS, una extensión espacial para bases de datos PostgreSQL, se crea un sistema robusto y eficiente para el manejo de información geográfica. Este documento proporciona una guía paso a paso para cargar capas y aplicar estilos en GeoServer utilizando datos almacenados en una base de datos PostGIS.
El proceso de integración de PostGIS con GeoServer permite a los usuarios aprovechar las capacidades de almacenamiento y consulta de una base de datos espacial, mientras se benefician de las funciones de visualización y distribución de GeoServer. Esta combinación es particularmente útil para organizaciones que manejan grandes volúmenes de datos geoespaciales y requieren un acceso rápido y eficiente a través de servicios web estándar.

En el documento se explicará a detalle cada herramienta SIG libre con la que se planea gestionar y publicar la información geográfica y sus metadatos asociados para explotar el potencial de Geoserver y Postgres+PostGIS.

Marco de Referencia

Desde la Secretaría de Cultura Recreación y Deportes SCRD se tiene como objetivo unificar la información geográfica en un Geoportal o visor web que sea transversal a todas las entidades adscritas a la Secretaría, por lo tanto, se busca que esta nueva plataforma tenga una estructura que permita la convergencia de la información espacial que en general se maneja desde la Secretaría. Adicionalmente se revisarán herramientas SIG complementarias con las cuales se planea gestionar y publicar está información geográfica.

Esta plataforma o geoportal contiene la información debidamente estructurada, estandarizada, analizada, verificada y publicada para el público en general haciendo uso de desarrollos propios de la SCRD basados en Django y Angular además de la aplicación en donde se aloja la información geográfica del visor web o geoportal que es tecnología Geoserver más la extensión de PostgreSQL denominada POSTGIS.

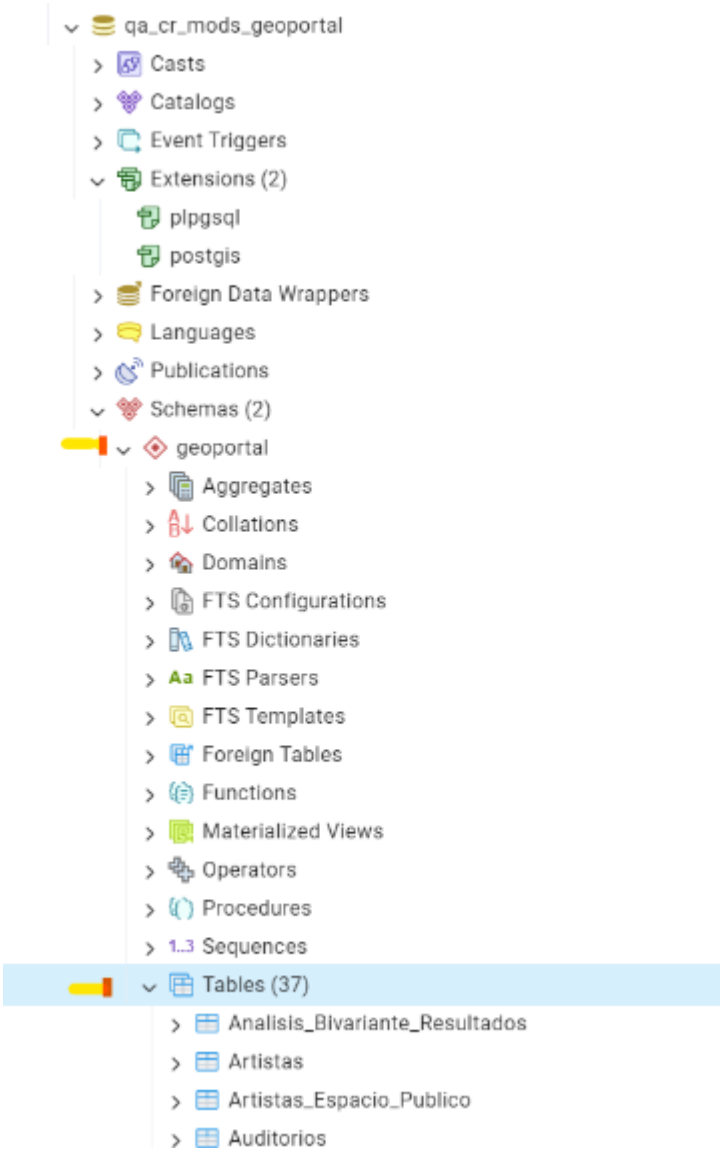
En estas herramientas GIS cabe resaltar, se gestionan las bases de datos espaciales y en cuanto al visor web Cultured este permitirá optimizar el acceso de los usuarios con el fin de presentar la información de una manera cada vez más organizada e intuitiva, estás optimizaciones se realizan estructurando algoritmos en código que implica principalmente el uso de Python en el backend, TypeScript/JavaScript en el frontend, HTML y CSS para la estructura y diseño de la interfaz, y SQL para la gestión de datos. La combinación de estos lenguajes permite crear una aplicación web geoespacial robusta y flexible además de las herramientas complementarias como (storymaps, dashboards, web maps y experiencias de usuario web) haciendo uso de herramientas de desarrollo web estático junto con plataformas de control de versiones GIT.

Configuración de base de datos espacial PostgreSQL

En el software Arcgis Experience Builder (de ahora en adelante Experience) se implementa la creación de widgets con código fuente en lenguaje type script, mediante este lenguaje de código y una determinada estructura de ficheros incluidos en una carpeta (que se encuentra en el

Explorador de archivos de Windows en la carpeta donde se instaló el software Experience, está ubicación se describirá más adelante) se puede incluir el widget dentro de los widgets personalizados del Experience, el proceso básico que permite crear y personalizar los widgets según las especificaciones requeridas se describe a continuación:

1. Alistamiento de capas



2. Creación y actualización de tablas en postgreSQL

Creación de capas en Geoserver

En el software Arcgis Experience Builder (de ahora en adelante Experience) se implementa la creación de widgets con código fuente en lenguaje typescript, mediante este lenguaje de código y una determinada estructura de ficheros incluidos en una carpeta (que se encuentra en el Explorador de archivos de Windows en la carpeta donde se instaló el software Experience, está ubicación se describirá más adelante) se puede incluir el widget dentro de los widgets personalizados del Experience, el proceso básico que permite crear y personalizar los widgets según las especificaciones requeridas se describe a continuación:

1. Configuración de Espacios de trabajo, Almacenes de datos y Capas en Geoserver

2. Configuración de Estilos en Geoserver

Los estilos en geoserver se crean a partir de códigos SLD creados mediante software Qgis o Udig. Para la generación del código SLD es importante tener en cuenta que la mayoría de estilos de Cultured y de las capas de cultura son capas de puntos que tienen como estilo una imagen formato PNG.

En el proceso es indispensable:

1. Guardar imagenes y complementos necesarios así como códigos SLD en Geoserver SCRD a través de software FileZilla con credenciales:

   Servidor: 172.16.10.253
   Nombre de usuario: steher
   Contraseña: Scrd-2023**

Puerto: 22

Ruta a la cual cargar imagenes
/usr/share/geoserver/data_dir/styles

2. Crear códigos en Udig apuntando a una imagen .png previamente guardada.

Desarrollo y despliegue de sitios web estáticos para productos WebMap, StoryMaps y Experiencias de usuario específicas

1. Desarrollo de sitios Web estáticos para WebMaps

Para crear un web map usando sitios estáticos y hospedado en GitHub Pages, se debe seguir un proceso sencillo. Primero, el usuario debe configurar un repositorio en GitHub. En este caso, se usará el repositorio ya existente de Edisson Mora (https://github.com/EdissonMora1234). El usuario debe asegurarse de tener los archivos necesarios como index.html, styles.css y main.js dentro del repositorio.

Una vez que los archivos están en el repositorio, se debe activar GitHub Pages. Esto se hace yendo a la configuración del repositorio, seleccionando la sección de Pages, eligiendo la rama principal y la carpeta raíz. GitHub generará una URL para el sitio web, permitiendo al usuario acceder a su web map.

Paralelamente, el usuario puede utilizar GitLab para el mismo propósito. Después de crear un proyecto en GitLab y subir los archivos necesarios, debe activar GitLab Pages en la configuración del proyecto. La URL para acceder al sitio en GitLab seguirá el formato https://webmaps1-storymap-basura.cultured.gov.co.

Para configurar las capas WMS, el usuario debe utilizar GeoServer, disponible en https://geoserver.scrd.gov.co/geoserver/web/?1. En GeoServer, se deben subir los datos espaciales y publicar las capas necesarias como servicios WMS. La URL del servicio WMS será utilizada para agregar capas al web map.

En el archivo index.html, el usuario debe incluir referencias a Leaflet, una biblioteca de código abierto para mapas interactivos. Además, los estilos básicos y el script para inicializar el mapa y agregar las capas WMS se deben definir en styles.css y main.js, respectivamente.

Al combinar estos elementos, el usuario puede crear un web map funcional. Leaflet permitirá agregar una capa base y superponer las capas WMS configuradas en GeoServer. Al finalizar, los cambios se deben subir al repositorio de GitHub o GitLab, y luego verificar que el web map se muestra correctamente en la URL generada por el servicio de Pages correspondiente.

Este método permite crear y publicar web maps de manera eficiente y económica, aprovechando servicios gratuitos y bibliotecas de código abierto. Con este manual, cualquier usuario puede configurar y publicar su propio web map utilizando GitHub Pages o GitLab Pages, y GeoServer para la gestión de capas WMS.

El código fuente para replicar un webmap según la plantilla catalogada como el Producto Mínimo Viable están alojados en:

https://github.com/EdissonMora1234/Plantilla_Webmap_PMV_LEYENDA_Y_ATRIBUTOS_VERSION6

El código contiene el archivo html y css básicos y las funciones de java script claves para las funcionalidades requeridas como tabla de atributos enlazada a control de capas, leyenda y todo con opción de minimización.

2. Desarrollo de sitios Web estáticos para generación de interfaz independiente y creación de StoryMaps

Para crear un Story Map con una interfaz de presentación que se sincronice con un mapa, utilizando la biblioteca Leaflet y capas de GeoServer, y guardarlo en un sitio estático tanto en GitHub Pages como en GitLab Pages, se debe seguir un proceso estructurado. A continuación se detallan los pasos para ambas plataformas.

Configuración Inicial

1. **GitHub Pages:**
   - Utilizar el repositorio existente en GitHub de Edisson Mora (https://github.com/EdissonMora1234).
   - Asegurarse de tener los archivos necesarios (`index.html`, `styles.css`, `main.js`) en el repositorio.
   - Activar GitHub Pages desde la configuración del repositorio, seleccionando la rama principal y la carpeta raíz. GitHub generará una URL para el sitio web.
2. **GitLab Pages:**
   - Crear un nuevo proyecto en GitLab y subir los archivos `index.html`, `styles.css`, y `main.js`.
   - Activar GitLab Pages en la configuración del proyecto, asegurándose de que la URL para el sitio siga el formato https://webmaps1-storymap-basura.cultured.gov.co.

Configuración de GeoServer

1. **Publicar Capas WMS:**
   - Utilizar GeoServer (https://geoserver.scrd.gov.co/geoserver/web/?1) para subir los datos espaciales y configurar las capas necesarias.
   - Publicar las capas como servicios WMS y anotar las URLs correspondientes.

Creación del Story Map

1. **Estructura del Proyecto:**
   - Los archivos `index.html`, `styles.css`, y `main.js` deben estar correctamente estructurados y configurados en el repositorio.
2. **Interfaz de Presentación:**
   - En el archivo `index.html`, se debe configurar la estructura básica del Story Map, incluyendo

referencias a Leaflet y una biblioteca para Story Maps (como Leaflet Story Maps o una solución similar).
    - En `styles.css`, se deben definir los estilos para la presentación y el mapa.
    - En `main.js`, se debe inicializar el mapa utilizando Leaflet, agregar las capas WMS desde GeoServer, y sincronizar las secciones de la presentación con la vista del mapa.

Sincronización del Mapa y la Presentación

1. **Inicializar el Mapa:**
    - Configurar Leaflet en `main.js` para inicializar el mapa y agregar una capa base.
2. **Agregar Capas WMS:**
    - Utilizar las URLs de los servicios WMS de GeoServer para agregar capas al mapa.
3. **Sincronizar Presentación y Mapa:**
    - Configurar eventos en `main.js` para que el desplazamiento a través de las secciones de la presentación actualice la vista del mapa y viceversa.

Publicación y Verificación

1. **Subir los Cambios:**
    - Para GitHub Pages, subir los cambios al repositorio de GitHub y verificar la funcionalidad en la URL generada.
    - Para GitLab Pages, subir los cambios al proyecto de GitLab y verificar la funcionalidad en la URL configurada.

Código de las plantillas de storymaps, centrado en el mapa y la principal de centrado en diapositivas

Se anexa códigos en formato (js/css/html) para realización de plantillas fundamentales para realizar y replicar storymaps a la fecha.

Plantilla centrada en el mapa:

https://github.com/EdissonMora1234/Plantilla_mapa_diapos_Storymap

Plantilla principal o centrada en las diapositivas:

https://github.com/EdissonMora1234/Plantilla_generalStorymaps

Conclusión

Este manual proporciona los pasos necesarios para crear y publicar un Story Map interactivo con una interfaz de presentación sincronizada con un mapa, utilizando GitHub Pages o GitLab Pages y GeoServer para la gestión de capas WMS. Utilizando la biblioteca Leaflet y configurando adecuadamente los archivos del proyecto mediante las plantillas creadas por la OTI-SCRD en los dos formatos de plantilla, es posible crear una experiencia interactiva y visualmente atractiva para los usuarios.

3. Desarrollo de sitios Web estáticos para creación de experiencias de usuario específicas como imágenes desplazándose en forma horizontal como insumo de narrativa

Para crear una página web estática que se pueda usar como un iframe y que permita desplazar imágenes horizontal o verticalmente con el scroll del mouse, se debe seguir un proceso claro y sencillo. A continuación, se detallan los pasos para implementar esta funcionalidad tanto en GitHub Pages como en GitLab Pages.

Configuración Inicial

En GitHub Pages, el usuario debe utilizar el repositorio existente en GitHub de Edisson Mora (https://github.com/EdissonMora1234). Se debe asegurar de tener

los archivos necesarios `index.html`, `styles.css`, `main.js`, entre otros en el repositorio. Luego, se debe activar GitHub Pages desde la configuración del repositorio, seleccionando la rama principal y la carpeta raíz. GitHub generará una URL para el sitio web.

En GitLab Pages, el usuario debe crear un nuevo proyecto en GitLab y subir los archivos `index.html`, `styles.css`, y `main.js`. Después, se debe activar GitLab Pages en la configuración del proyecto, asegurándose de que la URL para el sitio siga el formato https://webmaps1-storymap-basura.cultured.gov.co.

Creación de la Página Web Estática

Para la estructura del proyecto, los archivos `index.html`, `styles.css`, y `main.js` deben estar correctamente organizados y configurados en el repositorio. En el archivo `index.html`, se debe configurar la estructura básica de la página web, incluyendo un contenedor para las imágenes y las referencias a los archivos de estilo y script.

En el archivo `styles.css`, se deben definir los estilos necesarios para el contenedor y las imágenes, asegurándose de que el contenedor tenga un tamaño fijo y que las imágenes se dispongan en fila horizontalmente o en columna verticalmente, según se desee. Esto permitirá que las imágenes se desplacen de manera adecuada cuando se haga scroll con el mouse.

En el archivo `main.js`, se debe implementar la lógica para capturar el evento de scroll del mouse y desplazar las imágenes dentro del contenedor en la dirección deseada. Esto puede lograrse manipulando el desplazamiento del contenedor en respuesta al movimiento del scroll.

Publicación y Verificación

Una vez configurados los archivos, el usuario debe subir los cambios al repositorio de GitHub para GitHub Pages y verificar la funcionalidad en la URL generada por GitHub. De igual manera, para GitLab Pages, los cambios deben subirse al proyecto de GitLab y verificarse en la URL configurada.

Conclusión

Este manual proporciona los pasos necesarios para crear una experiencia de usuario con desplazamiento de imágenes utilizando sitios web estáticos. La implementación puede ser publicada en GitHub Pages o GitLab Pages, permitiendo que la página web sea utilizada como un iframe en otras aplicaciones. Usando HTML, CSS, y JavaScript, es posible crear una interfaz visualmente atractiva y funcional para mostrar imágenes que se desplazan con el scroll del mouse.

Un ejemplo es:
https://edissonmora1234.github.io/experience_imagenes/

4. Desarrollo de sitios Web estáticos para creación de Storymaps

Usando las plantillas generadas con los movimientos básicos se pueden personalizar cada vez más a tal punto de tener una replica muy parecida de los productos creados en la actualidad:

Generación de Story Maps mediante interfaz propia generada en sitio web estático

Los storymaps se realizaron según las plantillas:

1. Plantilla_Storymap_Centrado_Imagenes
2. Plantilla_MapaDiapos_Con_Leyenda_Storymap
3. Plantilla_mapa_diapos_Storymap
4. Plantilla_generalStorymaps

A partir de las plantillas se obtuvieron los productos storymap replica:

Versión Libre “Fortalecimiento del rol de la mujer en el sector cultural”:
https://edissonmora1234.github.io/StoryMap_RolDeLaMujer/

Versión propietaria de “Fortalecimiento del rol de la mujer en el sector cultural”:

https://storymaps.arcgis.com/stories/309203cd1fa14884bca2426f4252671b

Versión Libre “¡Que la basura no se vuelva paisaje!”: https://edissonmora1234.github.io/Storymap_BasuraNoPaisaje/

Versión propietaria de “Fortalecimiento del rol de la mujer en el sector cultural”: https://storymaps.arcgis.com/stories/b421b86f0efa4681bbdfa46cf49e1bf5

Versión Libre “Fichas locales”: https://storymaps.arcgis.com/stories/2845771d7c804377a36200544e731f6b

Versión propietaria de “Fichas locales”: https://edissonmora1234.github.io/Storymap_FichasLocales/

Se observa que los servicios OGC creados con geoserver y postgreSQL + POSTGIS complementados con interfaces en formato libre (css, html y js) son una herramienta robusta para explicar narrativas que contienen información geográfica.

Ajuste según observaciones de usuarios autores y despliegue oficial en Gitlab de la SCRD

Se han finalizado los ajustes y la migración de los productos StoryMap desarrollados por la Secretaría de Cultura, Recreación y Deporte (SCRD) al entorno institucional de GitLab, con la respectiva publicación en entornos accesibles mediante URLs específicas. Los productos actualmente disponibles son:

https://storymaps-roldelamujersectorcultura.cultured.gov.co/

https://storymaps-basuranopaisaje.cultured.gov.co/

https://storymaps-fichaslocales.cultured.gov.co/

Las mejoras implementadas en estos StoryMaps responden a las observaciones y recomendaciones contenidas en el siguiente documento:
🔗
https://drive.google.com/file/d/17AbrN55pfHQoQGyEf9Ii6alqSmEwRofV/view

Entre las principales actualizaciones realizadas se encuentran:

Implementación de desplazamiento fluido (scroll) para una navegación más intuitiva.

Adición de interactividad mediante enlaces internos y externos.

Mejoras visuales en la presentación de textos, disposición de contenidos, y despliegue de recursos informativos.

Estas modificaciones se llevaron a cabo directamente sobre los archivos fuente estáticos HTML, CSS y JavaScript, los cuales fueron editados y organizados dentro de la carpeta public/ de los respectivos proyectos en el repositorio institucional GitLab SCRD:

🔧
https://gitlabprd01.scrd.gov.co/OTI/cultured/territorios-creados-y-creativos/webmaps/storymaps

Finalmente, se solicita al equipo de soporte.sistemas la creación y/o actualización de los registros DNS correspondientes, con el fin de garantizar el correcto direccionamiento de las URLs públicas.

Con este proceso, se da por concluida la migración de los productos StoryMap al entorno de software libre de la SCRD, consolidando su mantenimiento y despliegue a través de GitLab como plataforma de control de versiones, automatización y publicación.

Recordatorio técnico sobre las mejoras implementadas

Durante la edición de los StoryMaps, se aplicaron mejoras clave que optimizan la experiencia de navegación y lectura en los sitios, entre ellas:

- En **JavaScript**, se integraron funciones específicas para:

    - Controlar el comportamiento del scroll entre secciones.
    - Activar o resaltar contenidos según la posición de la vista.
    - Agregar efectos de entrada o salida al visualizar componentes del StoryMap.
    - Gestionar eventos de navegación dinámica y botones de retorno al inicio.
- En **CSS**, se mejoraron:
    - Las clases para secciones con fondo personalizado.
    - El espaciado, márgenes y tipografía para mayor legibilidad.
    - El diseño responsivo, adaptando los contenidos a diferentes dispositivos.
    - Los efectos de transición y visibilidad progresiva al hacer scroll.

Estas implementaciones permitieron pasar de estructuras estáticas a una experiencia más rica, interactiva y moderna, manteniendo la simplicidad de uso y la legibilidad del contenido.

5. Desarrollo de sitios Web estáticos para creación de Webmaps

Se pueden crear webmaps que contengan las capas de geoserver a partir de la plantilla desarrollada con el código en formato (html, css y js) publicado desde el Gitlab de la SCRD, la plantilla es: Plantilla_Webmap_PMV_LEYENDA_Y_ATRIBUTOS_VERSION6

A partir de esta plantilla se pueden generar nuevos webmaps según prioridad Dirección Observatorio y Gestión del Conocimiento Cultural:

Los webmaps replicados en ambiente de pruebas son:

1. Indice_Equidad_Genero_Barrios_2021-2023
2. Indice_de_Cultura_Ciudadana_Barrios_2021_2023
3. Indice-Movilidad-2021-2023-localidades
4. Indice-Inclusion-2021-2023-localidades
5. Indice-Espacio-Publico-2021-2023-localidades
6. Indice-Equidad-de-genero-2021-2023-localidades
7. Indice-Cultura-Politica-y-Ciudadania-2021-2023-localidades
8. Indice-Convivencia-y-Cultura-de-Paz-2021-2023-localidades
9. Indice-Ambiente-2021-2023-localidades
10. Plantilla_Webmap_PMV_LEYENDA_Y_ATRIBUTOS_VERSION6

Desarrollo y despliegue de inventario general de Contenido de Arcgis Online mediante API de arcgis usando código python

El inventario general de Arcgis Online es extraído y compilado en un excel mediante un código en lenguaje python, este código es el siguiente: Inventario_Arcgis_Online.ipynb, el resultado de este código es un archivo en formato excel como el siguiente: ArcgisOnline_InventarioGeneral2024.

El código será ejecutado periódicamente con el fin de mantener actualizado el inventario de la licencia y el consumo de almacenamiento por tamaño de cada objeto que hay en Arcgis Online, lo anterior con el fin de gestionar de forma adecuada los créditos de la licencia de Arcgis Online, créditos que actualmente son limitados y deben ser utilizados de forma óptima.

Desarrollo de funcionalidad girar mapa 90 grados más brújula de indicador del Norte geográfico

Informe: Limitaciones de la Librería Leaflet para Girar el Mapa y Evaluación del Proyecto

En el desarrollo de un sistema web basado en la librería Leaflet, se planteó la necesidad de rotar el mapa 90 grados hacia la izquierda y mantener funcionalidades como la leyenda dinámica, el panel de atributos y un indicador estático del norte. Durante la implementación, se evidenciaron limitaciones técnicas inherentes a Leaflet y problemas específicos en la configuración, que ocasionaron la desconexión del mapa y dificultades de visualización. Este informe presenta las limitaciones observadas, el análisis de los problemas encontrados y un resumen de las soluciones implementadas.

Limitaciones de la librería Leaflet

Leaflet es una librería de mapeo ligera y ampliamente utilizada. Sin embargo, presenta ciertas limitaciones cuando se requiere manipular rotaciones complejas de mapas:

- **Transformaciones CSS no nativas**: Leaflet no tiene soporte nativo para rotar el mapa mediante su API. La rotación debe realizarse usando transformaciones CSS (`transform: rotate()`), lo cual afecta cómo se comportan las capas y los controles interactivos.
- **Problemas con el sistema de tiles**: La rotación puede desconectar o distorsionar las teselas del mapa base, ya que las coordenadas de los tiles no están diseñadas para alinearse correctamente después de una rotación.
- **Interacción con eventos del mouse**: El uso de transformaciones CSS desajusta las interacciones, como el zoom o el desplazamiento, lo que provoca comportamientos anómalos como saltos de vista o pérdida del contexto del mapa.
- **Sincronización del norte**: Aunque es posible rotar el mapa visualmente, Leaflet no reorienta automáticamente las capas vectoriales o los controles, lo que hace necesario realizar ajustes manuales.

Problemas observados durante el desarrollo

1. **Desconexión del mapa (pantallas grises):**
   - Cuando se aplicó `transform: rotate()` al contenedor del mapa, las teselas (tiles) dejaron de cargarse correctamente en ciertos momentos.
   - Esto ocurre porque las coordenadas de los tiles no coinciden con la nueva orientación visual del mapa.
2. **Zoom y desplazamiento erráticos:**

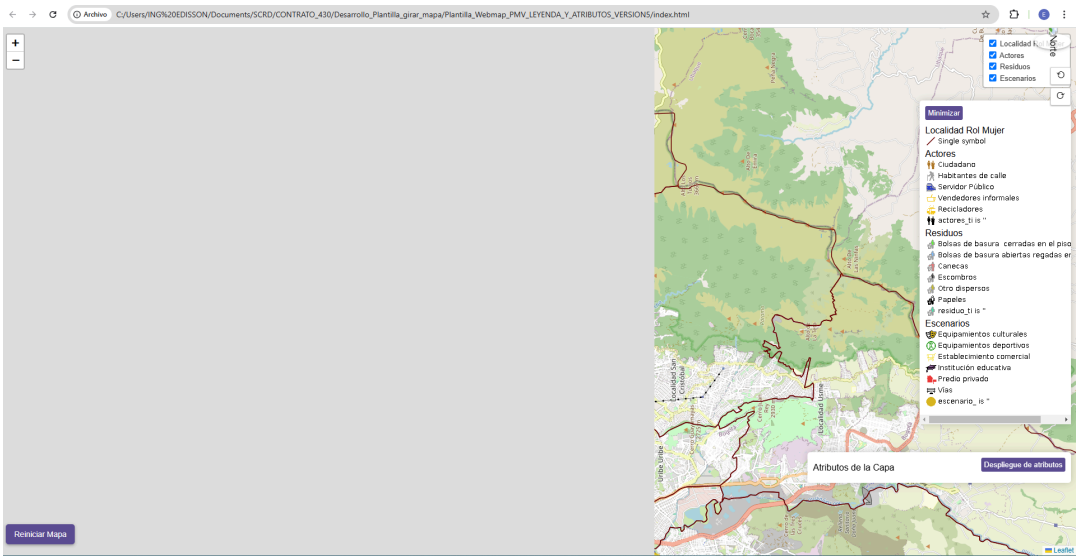
- El uso del scroll para hacer zoom se comportó de manera inusual, ya que los eventos del mouse no estaban correctamente sincronizados con la rotación aplicada.

3. **Problemas en la interpolación de capas:**
   - Las capas WMS y las teselas del mapa base presentaron inconsistencias visuales debido a la rotación, generando artefactos y desplazamientos inesperados.

Soluciones implementadas

1. **Rotación inicial del mapa:**
   - Se aplicó `transform: rotate(-90deg)` al contenedor del mapa (`#map`) para lograr la rotación deseada. Se ajustó `transform-origin: center` para mantener el punto de rotación.
2. **Indicador estático del norte:**
   - Se agregó un indicador del norte fijo en la esquina superior derecha del mapa para orientar al usuario.
3. **Interacción del usuario:**
   - Se mantuvieron todas las funcionalidades (leyenda dinámica, panel de atributos, botones interactivos), aunque se simplificó la rotación para evitar problemas de usabilidad en eventos interactivos.
4. **Depuración de eventos del mapa:**
   - Se probó la integración de las capas WMS y las interacciones con el panel de atributos para garantizar que los datos permanecieran accesibles.

Pantallazos generados durante el proceso



6. Recomendaciones finales

- **Alternativas a Leaflet:**
  - Si la rotación del mapa es un requisito crítico, se recomienda explorar otras librerías como Mapbox GL JS, que soporta rotaciones nativas y cambios de perspectiva.
- **Implementar plugins:**
  - Existe el plugin `Leaflet.RotatedMarker` para manejar rotaciones de marcadores. Para mapas completos, se podrían desarrollar extensiones personalizadas.
- **Evitar transformaciones complejas:**
  - Siempre que sea posible, usar la orientación predeterminada del mapa para garantizar un rendimiento óptimo y una experiencia de usuario fluida.

Desarrollo de funcionalidad yuxtaponer capas (con slider izquierda a derecha apagar y prender capas)

1. Contexto Inicial

Se solicitó implementar una funcionalidad en un mapa interactivo desarrollado con Leaflet, que permitiera superponer dos capas de un servidor GeoServer mediante un control deslizante (`slider`). La idea principal era que el movimiento del slider ajustara gradualmente la opacidad de las capas, mostrando una mientras la otra se ocultaba progresivamente. Este desarrollo debía respetar la estructura existente, que incluía un control de capas y una leyenda dinámica.

2. Análisis del Código Inicial

El código proporcionado ya contaba con:
- Un mapa inicializado con Leaflet.
- Una capa base de OpenStreetMap.
- Dos capas WMS cargadas desde un servidor GeoServer.
- Un control de capas que permitía alternar entre las capas.
- Funciones para actualizar una leyenda dinámica asociada a las capas activas.

Se identificó que el flujo del código era modular y estaba diseñado para

agregar funcionalidades de forma sencilla. Esto permitió integrar la nueva funcionalidad con mínimas modificaciones.

3. Solución Implementada

La solución se desarrolló en tres etapas:

3.1. Modificación del HTML

Se agregó un control deslizante dentro del contenedor del mapa para ajustar la opacidad de las capas. Este control se definió con un rango de valores entre `0` y `1`, representando el porcentaje de visibilidad de cada capa.

```
<div id="slider-container">
        <input type="range" id="layer-slider" min="0" max="1" step="0.01" value="0.5">
 </div>
```

3.2. Ajustes en el CSS

Esto posiciona y estiliza el control deslizante:

```
#slider-container {
        position: absolute;
        bottom: 20px;
        left: 50%;
        transform: translateX(-50%);
        z-index: 1000;
        background: rgba(255, 255, 255, 0.8);
        padding: 10px;
        border-radius: 5px;
        box-shadow: 0 2px 5px rgba(0, 0, 0, 0.3);
}

 #layer-slider {
        width: 200px;
}
```

3.3. Incorporación del Control Deslizante al JavaScript

Se añadieron dos pasos principales al código JavaScript:

1. Inicialización de las capas con opacidad ajustable:
 Cada capa WMS se cargó con una opacidad inicial específica:

```
var layer2021 = L.tileLayer.wms('URL_CAPA_2021', {
        layers: 'LAYER_NAME_2021',
        format: 'image/png',
        transparent: true,
        maxZoom: 21,
        opacity: 1
}).addTo(map);

 var layer2023 = L.tileLayer.wms('URL_CAPA_2023', {
        layers: 'LAYER_NAME_2023',
        format: 'image/png',
        transparent: true,
        maxZoom: 21,
        opacity: 0
}).addTo(map);
```

2. Gestión del evento del slider:
Se añadió un evento `input` al control deslizante para ajustar las opacidades de las capas en función de su valor:

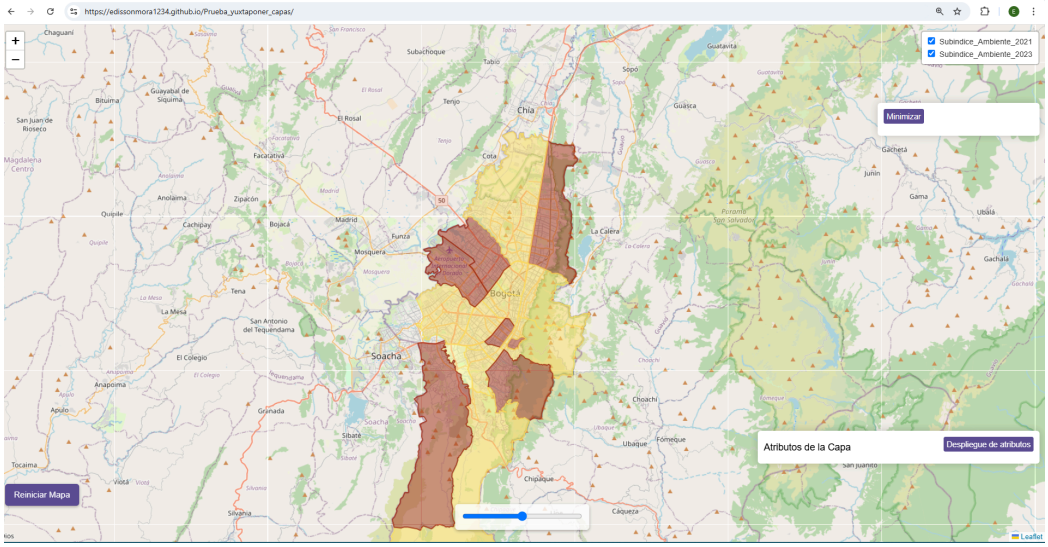
```
var slider = document.getElementById('layer-slider');
 slider.addEventListener('input', function () {
        var value = parseFloat(slider.value);
        layer2021.setOpacity(1 - value);
        layer2023.setOpacity(value);
});
```

4. Resultado

La funcionalidad de superposición gradual se integró con éxito. Ahora, el control deslizante permite explorar visualmente las diferencias entre las dos capas de GeoServer. Las capas mantienen su compatibilidad con el control de capas y la leyenda dinámica existentes.

Este desarrollo respeta la estructura original del código, garantizando su modularidad y extensibilidad para futuras mejoras.

El resultado del código se evidencia en el pantallazo:

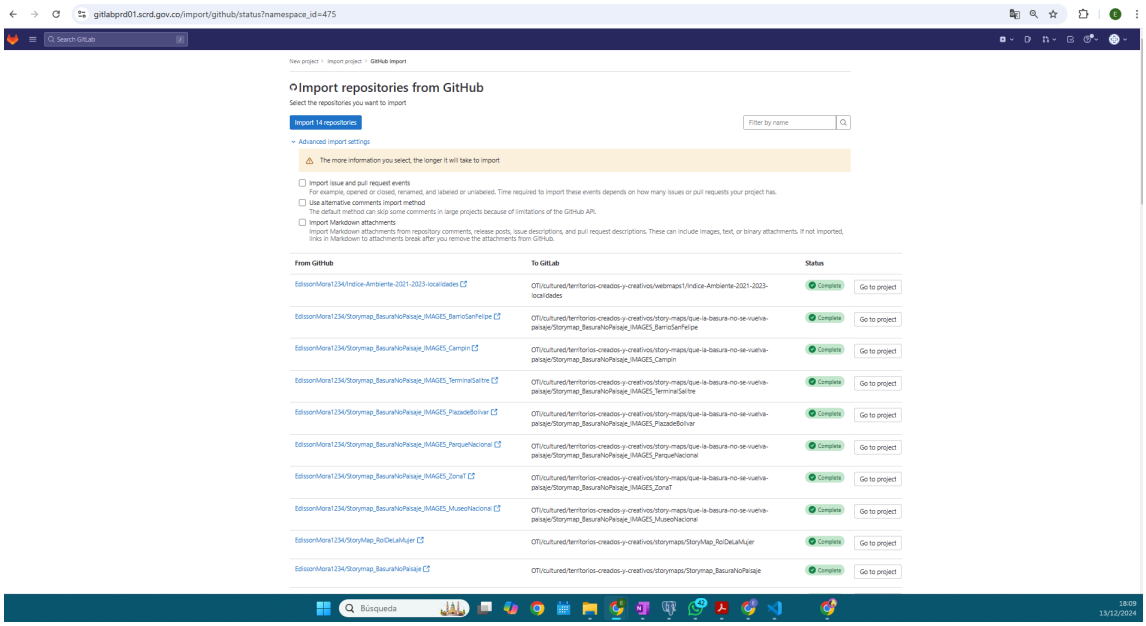


El producto se puede observar en:
https://edissonmora1234.github.io/Prueba_yuxtaponer_capas/

El código reposa en:
https://gitlabprd01.scrd.gov.co/OTI/cultured/territorios-creados-y-creativos/plantillas_pruebas_sig/Prueba_yuxtaponer_capas

Cabe resaltar que todos los desarrollos creados y mencionados en este documento y en otros del año 2024 se encuentran en el repositorio de la SCRD el cual es el GITLAB:
https://gitlabprd01.scrd.gov.co/OTI/cultured/territorios-creados-y-creativos

Se relaciona también pdf y pantallazo de 50 repositorios creados en el año:
https://drive.google.com/file/d/1bh-cqzHniDlaNWBTinRhGhDFdorUl4MY/view?usp=drive_link



Preparó	**Edisson Manuel Mora Madrigal** edisson.mora@scrd.gov.co	13/12/2024