MAPA INTERACTIVO PARA LA IDENTIFICACIÓN DE LAS PROBLEMÁTICAS AMBIENTALES EN EL MUNICIPIO DE MEDELLÍN

Camilo Ocampo
miloog@gmail.com

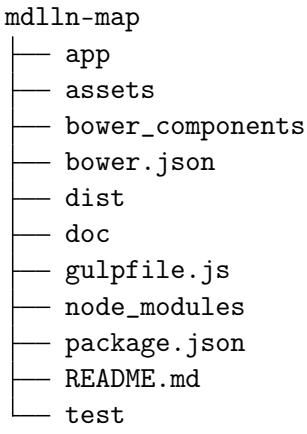
Diciembre 2016

1. Introducción

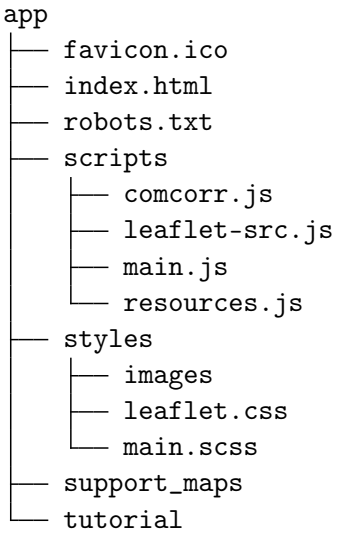
El mapa interactivo aquí documentado representa un producto enfocado y mínimo viable para mostrar información relevante de las problemáticas ambientales en el municipio de Medellín. Se seleccionó el desarrollo web como plataforma para el aplicativo dada su versatilidad a la hora de compartir la información tanto *online* como *offline*. Para su desarrollo se adoptó la filosofía *mobil first* que orienta el diseño hacia una plataforma responsive. Las herramientas fundamentales usadas para el desarrollo fueron HTML5, CSS3 y Javascript. Con el objetivo de mejorar la velocidad y eficiencia en el desarrollo se incluyeron dentro del entorno de desarrollo: El sistema de control de versiones Git [1], los administradores de paquetes npm [2] y Bower [3], y el ejecutador de tareas Gulp [4]. Finalmente se apoyó el diseño e interactividad de la plataforma haciendo uso de los *frameworks*: Bootstrap [5], jQuery y Leaflet [6].

Figura 1: Demostración de diseño responsive. Esta imagen puede encontrarse en: `assets/images/responsive.png`

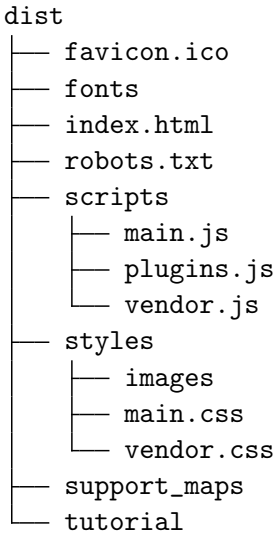
2. Distribución y estructura del proyecto



(a) Raíz del proyecto.



(b) Código fuente del aplicativo.



(c) Directorio de distribución.

Figura 2: Estrucura de directorios del proyecto.

2.1. Directorio raíz

El directorio raíz del proyecto se muestra en la figura 2a. Los archivos fuente de la aplicación se encuentran en el directorio `app`. La documentación, su fuente en LaTeX y este documento pueden encontrarse en el directorio `doc`. El directorio dist es generado automáticamente al compilar el proyecto y contiene los archivos necesarios para compartir la plataforma online u offline según la configuración.

Los archivos: `bower.json`, `gulpfile.js` y `package.json` corresponden a los archivos de configuración del entorno de desarrollo y los administradores de paquetes. Estos son los encargados de generar los directorios `bower_components`, `dist`, `node_modules` y `test`. Se recomienda prudencia al modificarlos, Los directorios `bower_components` y `node_modules` algunas dependencias fundamentales para la compilación del proyecto.

Finalmente el archivo `README.md`, contiene una guía para configurar un entorno de desarrollo en formato markdown.

2.2. Código fuente del aplicativo

En el directorio `app` se encuentran los archivos fuente del aplicativo. La estructura del directorio se puede apreciar en la figura 2b. Se destacan en este directorio los archivos `index.html`, `main.js` y `main.scss`. El aplicativo reside fundamentalmente en estos tres archivos, donde el primero contiene la estructura del contenido, el segundo las funciones y declaraciones de Javascript necesarias para su interactividad y el tercero la información de los estilos para el contenido. Este ultimo en formato SASS [7] que posteriormente será compilado en un archivo CSS.

Los archivos `leaflet-src.js` y `leaflet.css` corresponden al framework para mapas interactivos Leaflet.

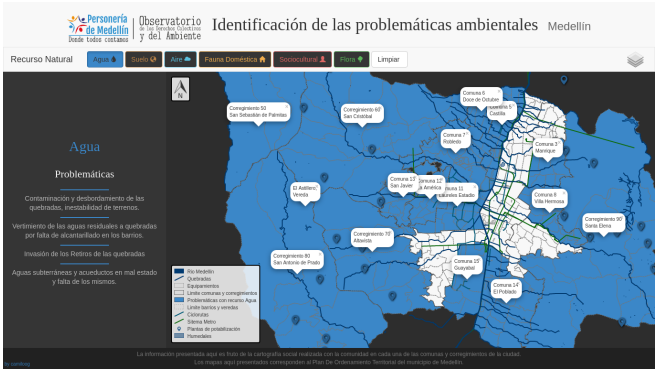
En el directorio `support_maps` se encuentra la información de los mapas en formato geoJson (Formato popular para contenido cartográfico en la web.).

En el directorio `tutorial` se encuentran las imágenes `.gif` mostradas en el manual de usuario del aplicativo.

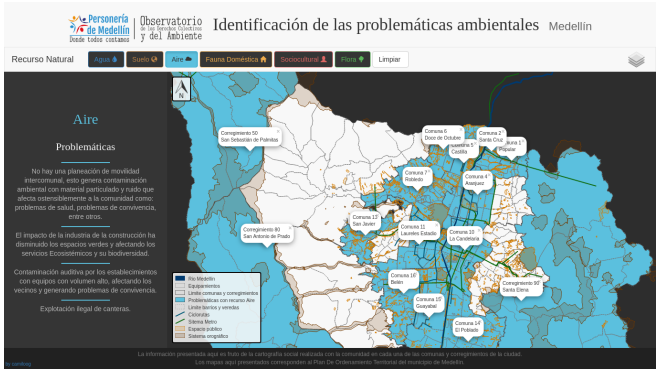
Los archivos dentro de la carpeta `app/styles/images/` son necesarios para la correcta presentación de la información. Entre estos se encuentran los logos institucionales, etc.

2.3. Activos del proyecto

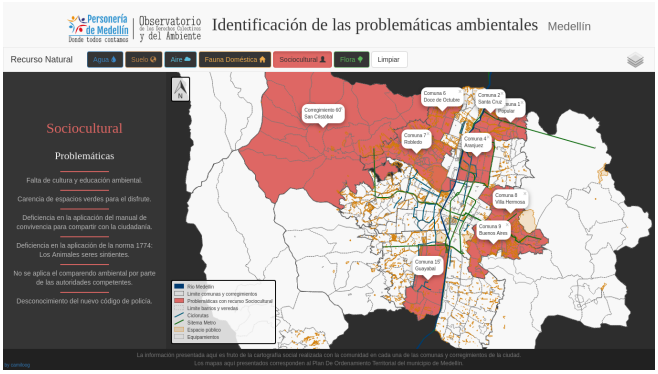
En la capeta `assets` del directorio raíz pueden encontrarse los mapas originales usados para el proyecto en diferentes formatos (`raw_maps.zip`), así como imágenes e impresiones de pantalla del aplicativo final (`assets/images/`). Cómo se mencionó antes, los mapas en formato geoJson pueden ser encontrados en `app/support_maps/` y las imágenes animadas mostradas en el manual de usuario en `app/tutorial/`.



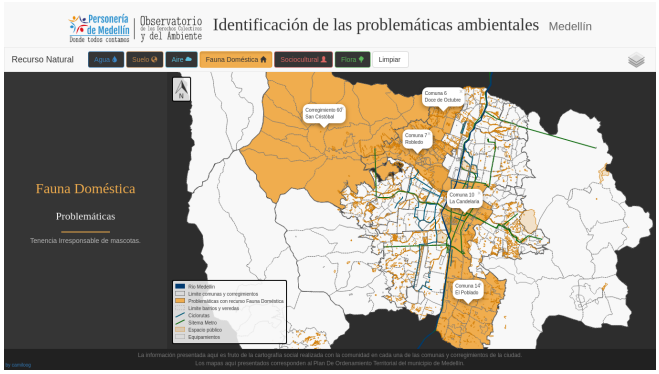
(a) Agua



(b) Aire



(c) Sociocultural



(d) Fauna doméstica

(e) Suelo

(f) Flora

Figura 3: Impresiónes de pantalla del aplicativo correspondentes a las problemáticas de cada recurso natural. Estas imágenes pueden ser encontradas en `assets/images/recurso_natural/`.

2.4. Estructura del control de versiones

El proyecto fue desarrollado haciendo uso del sistema de control de versiones Git. El proyecto cuenta con tres ramas: La rama `master` donde se encuentra todo el desarrollo actualizado, compilado y optimizado para la web; la rama `development` donde se realizan todos los desarrollos y pruebas antes de ponerlos en la rama `master` y finalmente la rama `offline-version` donde se optimiza la aplicación para su distribución *offline*.

2.5. Archivos para distribución

La carpeta `dist` contiene todos los archivos compilados y optimizados para su distribución, esta puede ser generada para publicar en la web desde las ramas `development` y `master` del control de versiones y optimizada para compartir *offline* desde la rama `offline-version`. Sin embargo su estructura es similar en ambas compilaciones dado que al compilar los archivos fuente estos se minimizan y optimizan en la misma estructura. la estructura del directorio de distribución se muestra en la imagen 2c. Los archivos `vendor.css`, `vendor.js` y `plugins.js` corresponden a los *frameworks* y demás dependencias usadas.

3. Configurar entorno de desarrollo para recompilar

3.1. Entorno de desarrollo

Para contribuir en el desarrollo del proyecto o recompilarlo se recomienda un entorno Unix (Linux, Mac OS), pues facilita la ejecución del entorno desde una linea de comandos.

Las siguientes dependencias deben ser instaladas en caso de no contar con ellas: `node.js`, su administrador de paquetes `npm` y `gulp`.

Se debe copiar el contenido del CD con el proyecto completo en un directorio o clonar el repositorio mediante git desde https://github.com/camiloog/mdlln-map. Luego de hacerlo ingrese en el directorio del proyecto:

```
git clone https://github.com/camiloog/mdlln-map.git
cd mdlln-map
```

Estando dentro del directorio del proyecto instalamos las dependencias necesarias:

```
npm install
bower install
```

Para montar y visualizar el proyecto en un servidor local (Configurado por las dependencias) simplemente ejecutamos la tarea `serve` de la configuración de `gulp`.

```
gulp serve
```

Para generar los archivos de distribución ejecutamos:

```
gulp serve:dist
```

Que generará la carpeta `dist` y montará el resultado en un servidor local.

3.2. Optimización web vs optimización offline

Como se mencionó antes, el sistema de control de versiones cuenta con una rama optimizada para la web (`development` o `master`) y una rama optimizada para su visualización offline (`offline-version`). En la versión web se cuenta con llamados al servidor a través de Ajax (Http request) para cargar los mapas requeridos de forma dinámica. Esto permite una mejor experiencia de usuario dado que los mapas no se descargan al tiempo, se descargan y grafícan según sean necesarios y el aplicativo se percibe más fluido. En la versión *offline* no se cuenta con un servidor que atienda a los los llamados de Ajax, por lo que

todos los mapas deben ser cargados al iniciar el aplicativo. A pesar de que no hay que descargar ningún archivo porque se cuenta con todos de forma local la aplicación offline tarda unos segundos en cargar todas las variables Javascript necesarias en el navegador. La versión *offline* puede ser compartida en un CD y visualizada en cualquier navegador.

Para compilar la versión optimizada para la web ingrese en la rama `development` dentro del directorio del proyecto y ejecute la tarea `serve:dist`.

```
cd mdlln-map
git checkout development
gulp serv:dist
```

Para compilar la versión *offline*:

```
cd mdlln-map
git checkout offline-version
gulp serv:dist
```

3.3. Consideraciones para modificar el código

El aplicativo reside principalmente en los archivos `app/scripts/main.js` y `app/styles/main.scss`. El código Javascript fue diseñado para estar encapsulado en funciones auto invocadas que permiten un diseño modular. Se recomienda continuar con este paradigma y comentar apropiadamente las funciones y variables para un código mantenible.

4. Manual de Usuario

A parte de este manual técnico se hace entrega de un manual de usuario con soportes gráficos que permitan al usuario entender el funcionamiento y herramientas del aplicativo. Este manual puede encontrarse en el mensaje de bienvenida de la aplicación. Los soportes gráficos pueden ser encontrados en `app/tutorial`.

5. Características responsive

La plataforma fue diseñada pensando en un aplicativo responsive bajo la filosofía *mobile first*. Esto implica que la distribución de los elementos en pantalla se auto ajustan y optimizan según el tamaño de la ventana o dispositivo tal como se muestra en la imagen 1.

Referencias

[1] Control de versiones git. https://git-scm.com/.

[2] Administrador de paquetes npm. https://www.npmjs.com/.

[3] Administrador de paquetes bower. https://bower.io/.

[4] Ejecutador de tareas gulp. http://gulpjs.com/.

[5] Bootstrap, responsive mobile first framework. http://getbootstrap.com/.

[6] Leaflet, mobile-friendly interactive maps. http://leafletjs.com/.

[7] Sass, preprocesador css. http://sass-lang.com/.