Modelo de Pruebas

Plataforma de software: Mapa RNI

Proyecto: Visualización de los niveles de radiación no ionizante en un mapa digital de la ciudad de San José de Cúcuta

V. 1.0.0

1. Introducción

El presente documento expone y define el modelo de mantenimiento y pruebas propuesto en el desarrollo de la aplicación Mapa RNI, la cual hace parte del proyecto de grado "Visualización de los niveles de radiación no ionizante en un mapa digital de la ciudad de San José de Cúcuta".

1.1 Alcance

El presente documento cubre específicamente los aspectos relacionados con el proceso de mantenimiento y pruebas de la aplicación. Como tal, este documento es parte de una documentación más completa que cubre todas las fases del desarrollo (desde el análisis hasta las pruebas), y se distribuyen de manera digital como un anexo al informe final del proyecto.

1.2 Definiciones, acrónimos y abreviaturas

- **ANE:** Agencia nacional del espectro.
- **ICNIRP:** Comisión internacional de protección contra la radiación no ionizante.
- **Backend:** Código ejecutado en el servidor en la nube.
- **Frontend:** Código ejecutado en el cliente (navegador) web.
- **OMS:** Organización mundial de la salud.
- **RNI:** Radiación no ionizante.
- **UIT:** Unión internacional de telecomunicaciones.

2. Detección y corrección de errores

Como se menciona en el documento de análisis, la aplicación ha sido desarrollada bajo una metodología de software iterativa e incremental, bajo la cual se realizan pequeñas iteraciones que añaden nuevas funcionalidades al software. Esto permite tener versiones intermedias funcionales antes de finalizar el producto, y sobre estas versiones intermedias puede realizarse todo tipo de verificaciones por parte de usuarios reales como si fuera un producto final. De esta forma pueden encontrarse errores y corregirse previamente antes de publicar la versión final.

Durante este proceso de iteraciones se descubrieron los siguientes errores. Para cada uno de ellos se describe la solución planteada, y se especifica si esta solución ha sido o no adoptada. A diferencia del resto de la documentación del software, la sección de mantenimiento no es una sección finalizada: Aunque el software ha sido ampliamente analizado y revisado, siempre existe la posibilidad de detectar un nuevo error, bien sea por cambios en las tecnologías que se utilizan, por incompatibilidades con determinadas tecnologías o simplemente por detalles que no fueron detectados durante la revisión. En caso de encontrar estos detalles o errores, inmediatamente deben expresarse en este documento y proceder a su solución.

Esto aplica también para nuevas funcionalidades: Cualquier nueva utilidad que pueda añadirse al software una vez este sea finalizado, se considera una acción de mantenimiento y debo ser informada en este documento.

Problema	Descripción	Estado
Error en la visualización de puntos elegidos por el usuario para realizar interpolación.	La interpolación depende de la selección de un punto específico, y para ello, es necesario utilizar el API de geolocalización del usuario. Por seguridad, Google ha decidido que estas funciones solo estén disponibles para aplicaciones que se ejecuten sobre protocolo seguro HTTPS y no el protocolo HTTP por defecto. La solución ha sido llevar la plataforma a un protocolo seguro HTTPS utilizando Let's encrypt.	Solucionado
El mapa de calor dibujado sobre el mapa se distorsiona al hacer zoom.	El mapa de color de Google Maps se dibuja de forma estándar a una distancia determinada. Sin embargo, al hacer zoom, este mapa se redimensiona, estableciéndose alrededor de cada punto, perdiendo así su visión de conjunto. Más que un error, es el comportamiento por defecto de los mapas de Google. La librería de mapas permite cambiar el tamaño en píxeles del mapa de calor. Entonces la solución ha sido multiplicar el tamaño en píxeles por un factor de crecimiento tomando como base la proyección mercator del mapa, haciendo crecer y disminuir los colores al mismo tiempo que se redimensiona el mapa.	Solucionado

La paleta de colores sobre el mapa oculta el contenido en ciertas pantallas pequeñas.	Inicialmente, la paleta de colores estaba ubicada en el extremo derecho del mapa, y ocupaba un espacio bastante amplio. Se ha solucionado mediante dos medidas: La primera, mover la paleta al sector izquierdo, donde hay menos botones, y por tanto, más espacio disponible sin inferir con el contenido. La segunda, redimensionar la imagen de la paleta de colores de tal forma que muestre la información utilizando el menor espacio posible.	Solucionado
Algunos enlaces en la plataforma presentan fallas.	La plataforma se ha desarrollado utilizando Windows para el desarrollo y pruebas. Sin embargo, el servidor en el cual se ha alojado la aplicación utiliza Linux. En Windows, los nombres de carpetas y archivos no tienen en cuenta mayúsculas/minúsculas, pero Linux sí. Por tanto, algunos enlaces no coinciden exactamente en su escritura por alguna letra mayúscula, pero como durante el desarrollo se utilizaba Windows, no fueron notorios estos problemas sino hasta ser verificados en el servidor. La solución a este error ha sido la más sencilla: corregir todos los enlaces hasta hacer coincidir las mayúsculas y minúsculas con los nombres de los archivos y carpetas.	Solucionado
Fallos en la inserción de archivos vía archivos CSV	Al subir un archivo csv para cargarlo en la base de datos, los datos del archivo no aparecían. Luego de analizar el error, se encontró que la carpeta de subida de archivos no tenía permisos de edición, por tanto, no se podían añadir nuevos archivos allí. La solución fue dar a la carpeta los respectivos permisos.	Solucionado
Fallo en la descarga de datos de la base de datos como Excel.	Como se ha explicado en la arquitectura, los datos se pueden exportar en formato Excel a través de la librería PHPExcel. Sin embargo, en un primer momento esta funcionalidad retornaba un Excel vacío. El problema surgió porque la librería PHPExcel utiliza en sus procesos la extensión PHPZip, la cual no estaba instalada en el servidor. La solución fue instalar la extensión PHPZip en el servidor y reiniciarlo, tras lo cual los archivos Excel fueron generados correctamente.	solucionado

3. Pruebas de software

El software ha sido probado de diferentes formas. La primera de ella, más básica pero no por ello menos importante, es la revisión directa: El enlace de la aplicación ha sido compartido con diferentes personas para que ellas interactuaran y revisaran su funcionamiento. Esto resulta importante pues se revisa la visualización desde diferentes dispositivos y navegadores según las preferencias de cada usuario, y se pueden encontrar nuevos errores u oportunidades de mejora que no hubiesen sido previamente descubiertos.

Sin embargo, no es suficiente con realizar pruebas a simple vista. Para constatar el correcto funcionamiento de la plataforma se realizan diferentes tipos de pruebas predefinidas, las cuales son descritas a continuación.

3.1 Pruebas de funcionamiento y rendimiento

Las pruebas de funcionamiento se realizan una vez finalizado el software. Con ellas se puede analizar cada uno de los aspectos de la carga de la aplicación dentro del navegador en diferentes condiciones de red.

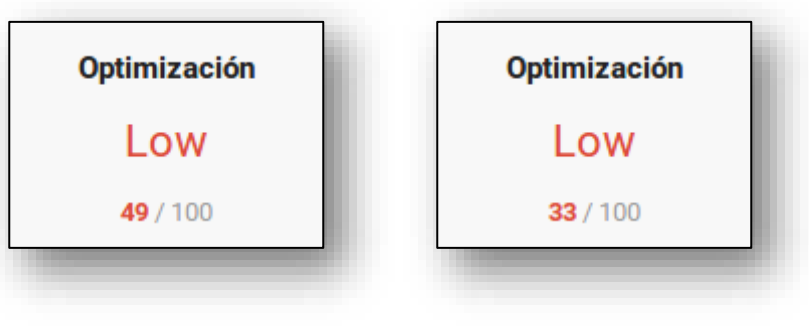
Para ello se ha utilizado Google Page Speed Insights, un completo analizador de aplicaciones web que presenta diferentes revisiones y alternativas de mejora. La metodología al realizar las pruebas es la siguiente: En primer lugar, se realiza una revisión con la herramienta, la cual otorga una calificación y unos consejos de mejora. Todos estos consejos son aplicados dentro del código fuente en la última iteración, luego de lo cual se actualiza la versión de la aplicación en el servidor y se ejecutan de nuevo las pruebas para analizar si los cambios han surgido efecto.

Es importante remarcar que esta herramienta NO revisa el código fuente de la aplicación (lo cual se hará en la sección 3.2 con otra herramienta) y por tanto no analiza la calidad de la aplicación. En su lugar, analiza la forma en la que el navegador carga la página, y las acciones que

podrían realizarse para acelerar esta carga. Estas optimizaciones muchas veces no dependen de la aplicación en sí, sino de configuraciones en el servidor o compresión de ciertos recursos.

La herramienta Google Page Speed Insights otorga una calificación para la carga de un sitio web, entre 0 y 100 puntos (siendo 0 sin optimización, y 100 completamente optimizado) en dispositivos de escritorio, y otra calificación entre 0 y 100 para dispositivos móviles. Si bien alcanzar una puntuación de 100 puntos es prácticamente imposible, pues requeriría la utilización de herramientas adicionales, cambio en los estándares y soluciones fuera del presupuesto, en general una puntuación por encima de 50 se considera positiva, siempre y cuando se lleven a cabo las optimizaciones propuestas.

En una primera revisión con la herramienta, los resultados fueron los siguientes:



Primera evaluación con Google Page Speed Insights. A la izquierda, la calificación en dispositivos móviles. A la derecha, la calificación en dispositivos de escritorio.

Entre las optimizaciones planteadas se encuentran:

- Optimizar el peso de las imágenes, el cual puede reducirse hasta un 79%.
- Habilitar en el servidor la compresión de recursos CSS/JavaScript.
- Optimizar el uso de la caché en el servidor.
- Optimizar el peso de los recursos CSS/JS.

La gran ventaja de esta herramienta es que no hace falta implementar estas soluciones a mano, sino que permite descargar las soluciones directamente y aplicarlas. Una vez aplicadas, la nueva calificación es la siguiente:

Primera evaluación con Google Page Speed Insights. A la izquierda, la calificación en dispositivos móviles. A la derecha, la calificación en dispositivos de escritorio.

Una vez aplicados los cambios la optimización ha mejorado notablemente. Si bien podrían conseguirse mayores optimizaciones, sería necesario recurrir a herramientas más técnicas fuera del alcance de este proyecto, y además la calificación obtenida asegura un buen tiempo de carga en los navegadores.

3.2 Pruebas de código fuente

Las pruebas de código fuente proporcionan la contraparte a las pruebas de rendimiento. En esta sección no se analizará la página, ni su carga en el navegador, sino su código a bajo perfil, en busca de errores de seguridad, duplicaciones de código que puedan evitarse, errores en programación susceptibles de ser corregidos, entre otros.

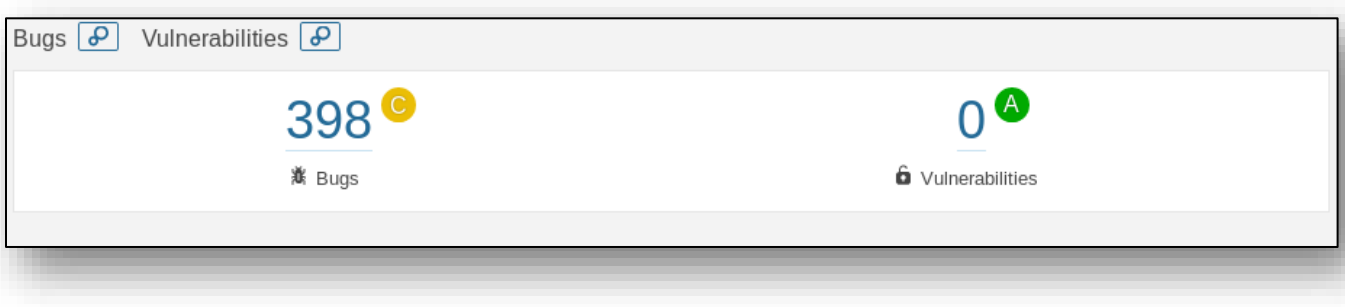
Para ellos se utiliza la herramienta SonarQube, la cual aporta un completo set de herramientas con este fin. A continuación, se presentan los resultados obtenidos por SonarQube, teniendo presente que en sucesivas ejecuciones de la herramienta se corrigieron algunos bugs importantes.

SonarQube presenta los resultados en dos formas: En primer lugar, de forma numérica cualitativa (cuantos errores de determinado tipo se encontraron), pero, en segundo lugar, más importante aún, se encuentra el resultado de forma descriptiva a través de una letra (A, B, C, D o

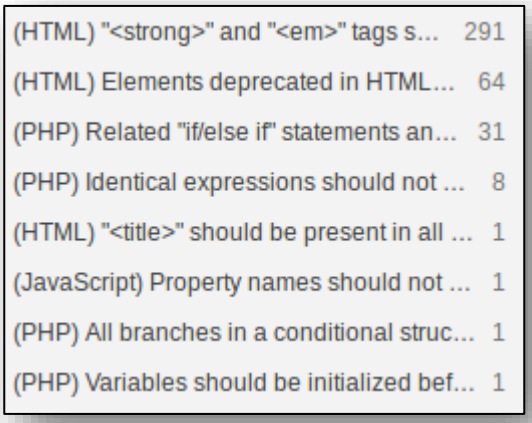
E). Los resultados marcados con A implican que no existen fallas en ese item, mientras los resultados marcados con E implican fallas catastróficas.

3.2.1 Análisis de Bugs

Los bugs y las vulnerabilidades son fallas en la programación que pueden ser corregidas. Un bug puede ser simplemente un punto y coma faltante, o una instrucción que ha quedado obsoleta en versiones recientes del lenguaje de programación, en cuyos casos no representa problemas de gravedad. Pero también pueden ser errores críticos en el almacenamiento de datos que puedan alterar el funcionamiento del sistema. Por eso en esta sección es fundamental analizar la letra asignada al item, para analizar el verdadero impacto sobre el software.



La anterior es la revisión inicial de la sección bugs y vulnerabilidades. Cabe destacar que no existe ninguna vulnerabilidad grave que afecte el funcionamiento del sistema, motivo por el cual se encuentra una calificación A en esta sección. No ocurre lo mismo con la sección de bugs, por lo cual se analiza su contenido para encontrar la causa de las 398 fallas. Cabe destacar que a pesar del alto número de bugs, la calificación sigue siendo C, un punto intermedio. Esto se debe a que son en su gran mayoría bugs menores. Aun así se realiza un análisis para mejorar esta calificación.



En primer lugar, se detecta que la mayoría de los bugs encontrados (291) son similares: "(HTML) "" and "" tags should be used". Al analizar este error, encontramos que se refiere al uso de una etiqueta HTML muy utilizada en el código: la etiqueta <i>. Según la documentación de la herramienta, esta etiqueta se utiliza para generar textos en cursiva, pero se encuentra obsoleta. Sin embargo, aclara la misma documentación que existe una excepción: La etiqueta <i> puede usarse siempre y cuando sea para insertar iconos (es justamente el uso que se la ha dado en la aplicación) pero para ello debo indicarse que es un icono, añadiendo el parámetro aria-hidden="true". Se realiza tal cual lo indicado en las 291 apariciones de la etiqueta <i>, y con ello la nueva medición es la siguiente:

Si bien es mucho mejor, aún puede ser mejorable. Analizando los bugs resultantes, hay de tres tipos principales:

- Errores de HTML de utilización de etiquetas obsoletas.
- Errores de JavaScript que están relacionados al funcionamiento del mapa.
- Errores en la librería de PHP para generar el archivo Excel.

Los primeros son perfectamente corregibles. Sin embargo, los otros dos dependen de servicios externos y por tanto están fuera del alcance. Aun así se corrigen todos los errores corregibles, logrando la siguiente calificación:

Finalmente, solo 42 errores se encuentran en el código, y estos hacen parte de las librerías externas utilizadas. Aun así, estos errores no son vulnerabilidades, y por tanto no permitirán fallas de seguridad en la aplicación

3.2.2 Mantenibilidad del código

La mantenibilidad del código es la facilidad para realizar cambios sobre el en futuras versiones. Un código difícil de leer no es por sí mismo un error, ni causará errores en su ejecución. sin embargo, si hará mucho más complicado realizar correcciones o añadir nuevas ediciones. Para esto la herramienta incluye un nuevo concepto, el “mal olor del código”, un concepto abstracto que indica lo difícil que resultará hacer mantenimiento al código.

Esto lo realiza asignando un puntaje en una escala arbitraria al orden del código y calculando la cantidad de tiempo que sería necesario para entender el código en su totalidad sin ningún tipo de documentación. Con base en esto, añade de nuevo una calificación descriptiva (A,

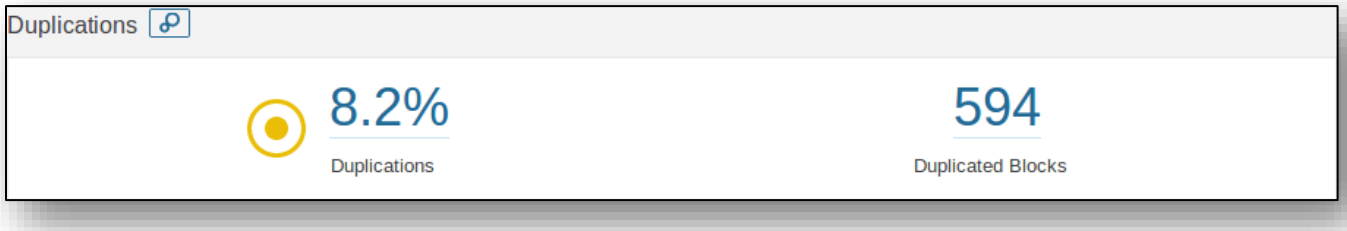
B, C, D, o E), siendo "A" una fácil mantenibilidad (buen olor), y siendo "E" una mantenibilidad casi imposible sin reestructurar totalmente el código (mal olor).

La medición sobre el código es la siguiente:

Se ha asignado la calificación "A" al código, por lo cual no ha sido necesario tomar medidas con respecto a esta calificación.

3.2.3 Duplicación del código

En teoría, el código no debería duplicarse en ningún sitio. Si una misma acción se va a realizar varias veces, en lugar de reescribir el código, debería convertirse en un método que es llamado en múltiples ocasiones. La medición de código duplicado arroja lo siguiente:



Si bien es un porcentaje relativamente bajo, analizamos a profundidad en donde se han encontrado las duplicaciones para tomar medidas si es necesario:

excel/Classes/PHPExcel/Reader/Excel5/Color/BIFF5.php	96.1%	74
excel/Classes/PHPExcel/Reader/Excel5/Color/BIFF8.php	96.1%	74
navbar.html	84.7%	149
excel/Classes/PHPExcel/Shared/CodePage.php	63.7%	100
excel/Classes/PHPExcel/Writer/OpenDocument/Styles.php	63.4%	59
excel/Classes/PHPExcel/Writer/Excel2007/Theme.php	62.8%	546
footer.html	56.5%	35
excel/Classes/PHPExcel/Writer/OpenDocument/MetaInf.php	51.1%	45
excel/Classes/PHPExcel/Chart/Properties.php	49.2%	179
excel/Classes/PHPExcel/CachedObjectStorage/PHPTemp.php	46.3%	93
excel/Classes/PHPExcel/CachedObjectStorage/DiscISAM.php	43.1%	90
excel/Classes/PHPExcel/Writer/OpenDocument/Settings.php	42.9%	33
index.html	41.2%	187
excel/Classes/PHPExcel/Writer/OpenDocument/Meta.php	37.5%	36
excel/Classes/PHPExcel/CachedObjectStorage/MemoryGZip.php	37.3%	50
excel/Classes/PHPExcel/CachedObjectStorage/MemorySerialized.php	36.9%	48
excel/Classes/PHPExcel/Chart/GridLines.php	34.0%	161
excel/Classes/PHPExcel/CachedObjectStorage/Igbinary.php	33.3%	50
excel/Classes/PHPExcel/Writer/PDF/DomPDF.php	33.0%	36

La anterior es la lista de archivos con mayor duplicación de código, su porcentaje de duplicación y su cantidad de líneas duplicadas, respectivamente. Como podemos analizar, la gran mayoría de archivos con duplicaciones de código se encuentran en la carpeta Excel, justamente la librería para la generación de archivos Excel. Esta es una librería externa y fuera del alcance, por tanto, son duplicaciones que no hacen parte de la aplicación Mapa RNI.

3.2.4 Estadísticas finales

Según las mediciones de SonarQube, el proyecto cuenta con aproximadamente 60 mil líneas de código. En su gran mayoría (58 mil líneas) son código PHP, lo cual tiene sentido teniendo en cuenta que es el único lenguaje usado en el servidor. Aun así, debe tenerse en cuenta que algunas de estos miles de líneas hacen parte de la librería PHPExcel que es externa y no ha sido desarrollada dentro de este proyecto, sino que se encuentra disponible con licencia libre.

A continuación, se encuentran 682 líneas de JavaScript, el lenguaje utilizado para las visualizaciones en el mapa. En la realidad esta cifra es algo mayor, sin embargo, siguiendo los consejos de las pruebas de rendimiento mencionados en la sección 3.1, se ha optimizado la cantidad de JavaScript utilizado para una carga más rápida.

Se encuentran también exactamente 666 líneas de código HTML para la estructura de la página, 331 líneas de CSS para el diseño (al igual que el JavaScript, el número sería algo más alto, pero ha sido comprimido para una carga más rápida), y 6 líneas del formato de intercambio XML.