

|  |
| --- |
| **Universidad Internacional de La Rioja (UNIR)**  **ESIT**    **Máster en Análisis y Visualización de Datos Masivos** |
|  |
| RailML Dashboard |

**Trabajo Fin de Máster**

**presentado por:** Sánchez Moreno, Adrián

**Director/a:** Marcelo Martí, Julio

Ciudad: Madrid

Fecha: 13 Junio

# Resumen

**Nota:** En este apartado se introducirá un breve resumen en español del trabajo realizado (extensión máxima: 150 palabras). Este resumen debe incluir el objetivo o propósito de la investigación, la metodología, los resultados y las conclusiones.

**Palabras Clave:** Se deben incluir de 3 a 5 palabras claves en español

Índice de Contenido

[Resumen 2](#_Toc11944333)

[Introducción 5](#_Toc11944334)

[Contexto y estado del arte 7](#_Toc11944335)

[Objetivos 8](#_Toc11944336)

[Metodología de trabajo 9](#_Toc11944337)

[RailML 10](#_Toc11944338)

[Requisitos 17](#_Toc11944339)

[Requisitos no funcionales 18](#_Toc11944340)

[Requisitos funcionales 18](#_Toc11944341)

[Descripción de la herramienta 19](#_Toc11944342)

[Control de configuración 20](#_Toc11944343)

[Evaluación 22](#_Toc11944344)

[Conclusiones 23](#_Toc11944345)

[Líneas futuras 24](#_Toc11944346)

[Referencias y enlaces 25](#_Toc11944347)

[Anexos 26](#_Toc11944348)

Índice de Ilustraciones

[Ilustración 1. Teoría de las cinco estrellas 10](file:///C:\Users\adri_\Desktop\ADRIAN\UNIR\TFM\railMLDashboard\TFM_ASM.docx#_Toc11429884)

# Introducción

En el mundo de la seguridad y la señalización ferroviaria se generan infinidad de datos en relación a los sistemas que entran en juego. Datos de señales, agujas, vías, conexiones, plataformas, balizas, redes, restricciones de velocidad, elementos de campo, paquetes de datos, aspectos de señales, etcétera.

Cada empresa tiene un formato de datos diferente, lo que dificulta el intercambio de datos entre empresas, se pierde mucho tiempo y genera un gran coste el crear interfaces para que este intercambio sea posible.

Además estos datos son prácticamente ilegibles. Miles de líneas con información, siendo muy difícil su interpretación. Es complicado tener una idea general del proyecto simplemente echando un vistazo al fichero de datos, o imposible realizar tareas de validación.

En 2002 se funda railM, cuyo objetivo es satisfacer las necesidades de los grupos de usuarios interesados en mantener un estándar internacional de la industria para el intercambio de datos ferroviarios. Es un formato abierto de intercambio de datos basado en XML. Gracias a este formato se soluciona el problema que se tenía en cuanto a la comunicación entre diferentes sistemas de distintas empresas.

Este trabajo tiene como principal objetivo solucionar el problema de interpretación de los datos. Proponiendo una página web en la que cualquier usuario de este formato de datos pueda hacer uso de ella. Alimentando a la web con un fichero railML se mostrará un resumen de los datos, un dashboard en el que el usuario pueda hacerse una idea de cómo es el proyecto al que se enfrenta. Señales y sus aspectos, número de vías, agujas…información global del proyecto, que pueda ser útil para los usuarios de este formato de datos para que puedan tomar decisiones con echar un vistazo al dashboard generado.

Esta memoria constará de los siguientes capítulos, en los que se abordarán los siguientes temas:

* El primer capítulo [**“Contexto y estado del arte”**](#_Contexto_y_estado)se estudiará el campo de los dashboards en el ámbito ferroviario, se hará un repaso a este tema durante toda su historia, se incluirán referencias, estudios, etcétera.
* El segundo capítulo [**“Objetivos”**](#_Objetivos) se centrará en los puntos a conseguir.
* El tercer capítulo [**“Metodología de trabajo”**](#_Metodología_de_trabajo) contemplará los pasos a seguir para conseguir los objetivos mencionados anteriormente.
* El cuarto capítulo [**“RailML”**](#_RailML) hará un repaso por este formato de datos, su historia, objetivos, licencias, etcétera.
* El quinto capítulo [**“Requisitos”**](#_Requisitos) fijará los requisitos software que cumplirá la herramienta a desarrollar.
* El sexto capítulo [**“Descripción de la herramienta”**](#_Descripción_de_la) se detallará el desarrollo de la herramienta así como su funcionalidad e interfaz de usuario.
* El séptimo capítulo [**“Control de configuración”**](#_Control_de_configuración) se centrará en cómo se ha gestionado el desarrollo de la herramienta.
* El octavo capítulo [**“Evaluación”**](#_Evaluación) validará la usabilidad de la herramienta y su aplicabilidad.
* El noveno capítulo [**“Conclusiones”**](#_Conclusiones) englobará el alcance y la relevancia de la aportación.
* El décimo capítulo [**“Líneas futuras”**](#_Líneas_futuras) aportará ideas para continuar con este trabajo.
* El undécimo capítulo [**“Referencias y enlaces”**](#_Referencias_y_enlaces) incluirá las pertinentes referencias y enlaces.
* El duodécimo capítulo [**“Anexos”**](#_Anexos) incluirá el código o fragmentos del mismo.

# Contexto y estado del arte

# Objetivos

El objetivo general de este trabajo es la generación de un dashboard en el que se recoja un resumen de la información guardada en un fichero de datos con formato railML para que los usuarios de este tipo de ficheros puedan de un vistazo saber qué es lo que esconden los datos.

Este trabajo intenta ayudar al problema de enfrentarse a un fichero de datos de gran volumen difícil de analizar sin herramientas o visualizadores. Miles de líneas tediosas de leer, en las que no se sacan conclusiones sólidas.

Este dashboard se desplegara al alimentar a una aplicación web con un fichero de datos de este tipo. Una vez que el usuario proporcione el fichero, se “parsearan” los datos, se analizarán, se hará una limpieza de los mismos, se harán cálculos, se determinará la información a mostrar, evaluando e identificando la información relevante.

Una vez que hechas todas estas tareas previas, se desplegará el dashboard en la misma página web.

El dashboard recogerá la información mínima para poder tener una visión global del proyecto al que se enfrenta. Visualizando gráficas, tablas, nombre del proyecto, estaciones, conexiones, elementos del proyecto, etcétera. Con toda esta información sintetizada el usuario podrá tomar decisiones y conocer de manera sencilla cómo es el proyecto o a qué tipo de proyecto se está enfrentando.

Además una vez que se muestra el dashboard, el usuario podrá volver a cargar otro fichero sí lo que desea es analizar otro fichero distinto.

# Metodología de trabajo

A continuación se detallarán una serie de pasos a seguir para conseguir los objetivos descritos anteriormente.

Una de las primeras cuestiones a analizar es los datos que se quieren visualizar. El tipo de fichero, su procedencia, extensión, cómo se van a leer esos datos, etcétera.

Después de elegir el fichero con el que trabajar, se hará un estudio profundo de lo que abarca dicho fichero, eligiendo la información más importante del mismo para su posterior visualización.

Una vez elegidos los datos a analizar y con los que se va a centrar la aplicación, se decidirán las herramientas más convenientes para llevarlo a cabo, lenguajes de programación, librerías, bases de datos, tecnologías…

Cuando ya se hayan tomado estas decisiones, se empezará a pensar en la codificación y en cómo se va a estructurar el código. Para ello se trabajará en los [requisitos](#_Requisitos), en los que se detallará todas las funcionalidades de la aplicación, así como especificaciones técnicas necesarias.

Para la codificación se diferenciarán los siguientes módulos a llevar a cabo:

* Lectura de los datos.
* Comprobación del esquema del fichero de datos.
* División de los datos en pequeños data frames para analizarlos por separados.
* Limpieza de los data frames.
* Cálculos necesarios.
* Visualización.

Una vez terminada la codificación se pasará a la fase de verificación, en la que se determinará si se cumplen los objetivos y los requisitos definidos. La verificación se analizará con unos test funcionales, en los que se detallará cada prueba que se haga. Toda lo pertinente a la fase de verificación se detallará en el capítulo de [“Evaluación”](#_Evaluación).

# RailML

El objetivo de [railML.org](https://www.railml.org/index.php/en/) satisfacer las necesidades de los grupos interesados en un estándar industrial europeo para el intercambio de datos ferroviarios.

Esta iniciativa fue fundada en 2002 debido a la dificultad que se presentada a la hora de conectar diferentes aplicaciones informáticas ferroviarias. El primer esquema de datos ralML fue desarrollado en 2005. Después de las primeras experiencias con el desarrollo y uso de railML 1.x, railML2.0 fue publicado en 2009.En 2014 comenzó una cooperación con [UIC](https://uic.org/) (International unión of railways), siguiendo a muchas empresas ferroviarias estatales (ÖBB, DB, SNFC, ...) para convertirse en socios de railML.

El desarrollo de railML tiene lugar en grupos de trabajos de expertos en el área del software ferroviario, modelización de datos e informáticos. Los grupos de trabajo están definidos por los cuatro diferentes esquemas de railML.

Cada grupo de trabajo se reúne por separado una vez cada dos o tres meses para discutir sobre diferentes problemas y futuros pasos a seguir en el desarrollo. La comunidad de railML se reúne dos veces al año en conferencias oficiales de railML.org.

La teoría de las cinco estrellas para datos abiertos, definida por

Tim Berners-Lee, inventor de la Web e iniciador de los Datos Enlazados (Linked Data), define cinco niveles para datos abiertos en su teoría de las cinco estrellas (<https://5stardata.info/es/>), así como el coste y beneficio para consumidores y editores.



Ilustración 1. Teoría de las cinco estrellas

La primera estrella define que como consumidor se pueden ver, imprimir, guardar localmente. Ingresar los datos en cualquier otro sistema, cambiar los datos como se quiera y compartir los datos con quien se quiera. Como editor es simple publicarlos, y no se tiene que explicar repetidamente a otros que pueden usar tus datos.

La segunda estrella, permite para el consumidor favorecerse de los beneficios de la primera estrella, además de poder procesarlos directamente con software propietario para agregarlos, hacer cálculos, visualizaciones, etcétera. Como editor sigue siendo simple publicarlos.

Como consumidor en la tercera estrella, se sigue englobando todo lo explicado en la segunda estrella, además de poder manipular los datos de cualquier forma que se quiera, sin limitación de características o de uso de algún tipo de software en particular. Como editor se podría necesitar convertidores o plug-ins para exportar los datos desde el formato propietario. Sigue siendo relativamente simple publicarlos.

La cuarta estrella sigue manteniendo las ventajas del nivel tres y además, se pueden enlazar desde cualquier otro sitio (Web o local), se pueden marcar como favoritos, se pueden reutilizar partes de los datos, se pueden reutilizar herramientas y librerías disponibles, se pueden combinar sin problemas los datos con otros. Las URI (Identificador de recursos uniforme) son un esquema global por lo que si dos cosas tienen la misma URI entontes así fue intencionado, y de ser así se estará en camino de alcanzar el nivel de cinco estrellas de datos. Como editor se tendrá el control granular sobre estos datos y puedes optimizar su acceso, otros editores de datos pueden ahora enlazarse a tus datos, se tendrá que asignar URIs a los datos y pensar en cómo representarlos.

El nivel cinco, como los demás niveles, gozará de las ventajas del nivel cuatro y además, se podrá descubrir más datos (relacionados) mientras se consumen los datos, se podrá aprender directamente acerca del esquema de datos, ahora se tiene que manejar enlaces rotos de datos, como los errores 404 en las páginas web. Como editor se los datos se pueden hacer descubiertos, se podrá incrementar el valor de los datos y se podrá ganar los mismos beneficios de los enlaces como los consumidores.

De acuerdo con esta teoría los esquemas de railML están disponibles como datos abiertos y no propietarios (“tres estrellas”) en la página web de railML.org. Lo que significa que los usuarios pueden manipular los datos libremente, sin la necesidad de poseer un paquete software.

RailML es un formato de intercambio de datos ferroviarios de código abierto. Esto significa que está disponible para cualquier persona de manera gratuita. La organización coordina el trabajo de railML y no es una empresa comercial con la intención de obtener beneficio.

Aunque varias empresas relacionadas con el negocio ferroviario, desarrolladores de software y otras organizaciones apoyan la idea de un modelo de intercambio de datos común, único y gratuito para el sector ferroviario, algunos servicios no se pueden proporcionar de manera gratuita o no pueden ser proporcionados a la comunidad.

A continuación se detalla una lista con los servicios gratuitos, los servicios de pago y los servicios que no ofrece railML.org.

Los servicios gratuitos son:

* Acceso y descarga de los esquemas railML.
* Uso de los esquemas railML como parte de la interfaz del software o servicio.
* Acceso a la información de la comunidad y datos, con una cuenta de usuario en [www.railML.org](http://www.railML.org).
* Miembro de railML.org.
* Acceso y uso para propósitos de pruebas a ficheros de datos de ejemplo, los cuales se utilizan en el desarrollo de este trabajo.
* Acceso al fórum de railML.
* Participación en los grupos de desarrollo.
* Acceso a la wiki de railML.
* Presentación de propuestas de ampliación de esquemas railML.
* Participación en las conferencias y eventos de railML.
* Descarga y uso de railVIVID: The railML viewer and validator.
* Información esencial sobre railML, railML.org y el uso de railML.

Los servicios de pago son:

* Reuniones de presentación de productos, así como talleres generales y específicos sobre railML.
* Consultoría por expertos de railML.org sobre las mejores prácticas de gestión de datos / estrategia de intercambio.
* Adquisición de la certificación obligatoria (tarifa única) para el software o la interfaz railML antes de su uso.
* Costos por cesar y desistir de advertencias y honorarios legales causados ​​por el uso indebido de la propiedad intelectual de railML.org y la violación de los términos de la licencia.

Los servicios que no son ofrecidos son:

* Programación para la implementación de interfaces de exportación o importación de railML en un software.
* Acceso a bases de datos de cronograma, infraestructura, enclavamiento o material rodante en formato de archivo railML.

railML.org es una asociación de desarrollo de empresas e instituciones independientes. Sus sugerencias, comentarios y críticas son las bases del desarrollo y consolidación de los esquemas de railML. Los socios se dividen en tres grupos: usuarios, desarrolladores y colaboradores. Algunos de los socios más importantes y conocidos son Adif, CAF, Siemens, Thales, Alstom o Bombardier.

La especificación de railML contiene esquemas para cuatro grandes áreas: infraestructura, cronograma, material rodante y enclavamiento. Estos cuatro subesquemas están divididos a su vez en más subesquemas que abordan áreas más específicas. Se están planificando y discutiendo subschemas de railML adicionales como la lista de tripulantes, la gestión de activos o los datos en tiempo real.

Un fichero railML puede contener otros subesquemas. La división en subesquemas permite que cada aplicación individual compatible con railML aplique las partes del esquema completo, por ejemplo, Los subschemas - en varias combinaciones.

El subesquema de cronograma está centrado en la descripción del calendario ferroviario, incluidas todas las diversas facetas que necesitan las aplicaciones de intercambio de datos. En concreto, el esquema de cronograma contiene la siguiente información:

* Períodos de operación: Los días de operación para servicios de trenes o de clasificación.
* Partes del tren: las partes básicas de un tren como una secuencia de operación o puntos de control con las mismas características, como la formación y el período de operación. La parte del tren incluye la información real con respecto a la trayectoria del tren, así como la información del horario correspondiente.
* Trenes: Una o más partes del tren conforman un tren y representan la vista operativa o comercial de la carrera del tren.
* Lista: Las partes del tren se pueden vincular para formar las circulaciones necesarias para la clasificación (horarios de material rodante).

El subesquema de material rodante railML se centra en la descripción del material rodante de los ferrocarriles, incluidas todas sus diversas facetas que las aplicaciones de intercambio de datos consideran necesarias. En particular, el esquema de material rodante railML contiene la siguiente información:

* Vehículos: las características de los vehículos ferroviarios individuales o las familias de vehículos se describen en esta parte del esquema. La descripción de los vehículos considera algunos datos generales utilizados para organizar los activos, como la denominación, la clasificación o los números de los vehículos, tal como los da su operador. La mayoría del esquema se centra en la estructura para almacenar los diversos aspectos técnicos de los vehículos ferroviarios con respecto a su sistema de propulsión, características de la carrocería, frenos o servicios instalados dentro del vehículo.
* Formaciones: las características de los juegos de trenes o partes de los mismos formados por varios vehículos diferentes o similares se describen en esta parte. Esta combinación de vehículos se utiliza para describir las características del tren según sea necesario. Sin embargo, la consistencia lógica entre la formación y los vehículos de los que está hecha no se aplica por el esquema. Debe ser asegurado por la aplicación que produce los datos.

El subesquema de infraestructura railML se centra en la descripción de la infraestructura de la red ferroviaria que incluye todas sus diversas facetas que son necesarias para las aplicaciones de intercambio de datos. Concretamente este esquema contiene la siguiente información:

* Topología: La red de seguimiento se describe como un modelo de borde de nodo topológico.
* Coordenadas: todos los elementos de la infraestructura ferroviaria pueden ubicarse en un sistema de coordenadas de dos o tres dimensiones arbitrario.
* Geometría: La geometría de la vía puede ser descrita en términos de radio y gradiente.
* Los elementos de infraestructura ferroviaria: incluyen una variedad de activos relevantes para ferrocarriles que se pueden encontrar debajo, sobre o cerca de la vía férrea.
* Otros elementos ubicados: engloban elementos que están estrechamente relacionados con la infraestructura ferroviaria.

El esquema de enclavamiento railML se centra en la información que los administradores de infraestructura suelen mantener en los planes de señal y las tablas de bloqueo de ruta. Los principales usuarios de este esquema son proveedores interconectados y simuladores.

En este subesquema se recoge la siguiente información:

* Preparación de datos: es el proceso de adaptar un sistema de enclavamiento y señalización ferroviaria a un proyecto específico. Los errores en los datos afectan la seguridad; es demasiado obvio que un aspecto incorrecto de la señal puede causar accidentes terribles. Por eso se invierten mucho tiempo y esfuerzo en las pruebas. Un formato de intercambio de datos estándar permitirá la automatización de la transferencia de datos y reducirá el número de errores al eliminar el factor humano. Esto creará mayores niveles de seguridad a un costo sustancialmente menor.
* Los programas de simulación: calculan el impacto del enclavamiento y la configuración de la seña. Cosas como cambiar una señal, usar un impulso de puntos más rápido o acortar bloques pueden tener un impacto significativo. Los algoritmos de simulación son cada vez más potentes y alcanzan un nivel de precisión donde los segundos son importantes. En la actualidad, el comportamiento en tiempo real del enclavamiento y la señalización a menudo es desconocido. El esquema railML IL permite a los modelistas absorber rápidamente información sobre los sistemas de enclavamiento, como el comportamiento de la sincronización y las rutas, y analizar el impacto en la capacidad del ferrocarril.

Este último subesquema es en el que se centrará este trabajo, al ser de gran importancia los datos de configuración del enclavamiento.

# Requisitos

<https://www.ecured.cu/Requisitos_de_Software>

https://medium.com/grupo-carricay/c%C3%B3mo-escribir-un-buen-documento-de-especificaci%C3%B3n-de-requisitos-de-software-fd8bb3b5a39a

Los requisitos en el desarrollo de aplicaciones informáticas son un punto clave en el proyecto. Muchos de estos proyectos fracasan debido a una mala definición, especificación o administración de requisitos. Requisitos incompletos o un mal manejo de los cambios de los mismos son motivos más que suficientes para fracasar.

El [Glosario](https://www.ecured.cu/index.php?title=Glosario&action=edit&redlink=1) de Terminología Estándar de Ingeniería de [Software](https://www.ecured.cu/Software) (IEEE: Standard Glossary of Software Engineering Terminology) define al requisito como:

1. Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
2. Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
3. Una representación en forma de documento de una condición o capacidad como las expresadas en 1 o en 2.

Debido a que los requisitos son las necesidades del producto que se debe desarrollar en cualquier proyecto de software, es importante no perder de vista que un requisito debe ser especificado por escrito como todo contrato o acuerdo entre dos partes; posible de probar o verificar para poder comprobar si se cumplió con él o no; consistente que no entre en contradicción con otros requisitos y conciso, o sea, fácil de leer y entender. Además, un requisito deber estar completo, es decir, que proporcione la información suficiente para su comprensión. Y por último no debe ser ambiguo para no causarle confusiones al lector.

En el documento ERS (Especificación de requisitos software) es donde se registran las necesidades del negocio (requisitos del cliente y necesidades del usuario) y se definen los requerimientos que debe cumplir el producto, sistema o software a desarrollar. Por tanto, es el medio de comunicación entre los clientes, usuarios y desarrolladores.

Los **requisitos no funcionales** o **atributos de calidad** son propiedades o cualidades que el producto debe tener, es decir, restricciones o condiciones que el producto que está siendo desarrollado debe cumplir. Por ejemplo, restricciones de sistema operativo, de ambiente, rapidez, seguridad, usabilidad, etc.

Los **requisitos funcionales** son las necesidades que el software debe cumplir de manera satisfactoria, es decir, las funciones que el software será capaz de realizar. Estos requerimientos son cálculos, detalles técnicos, manipulaciones de datos (entrada y salida) y otras funciones específicas que se supone, que el software debe cumplir.

A continuación se detallan los requisitos funcionales y no funcionales.

## Requisitos no funcionales

* La aplicación se desarrollará en Windows 10.
* La aplicación será una aplicación web.
* La aplicación será compatible con la mayoría de navegadores (mínimo Google Chrome y Mozilla Firefox).
* La aplicación se desarrollará en Python 3.
* Se utilizará la librería *dash*, para la creación de dashboards.
* Se utilizará la librería *pandas*, para limpieza de datos y data frames.
* Se utilizará la librería *element tree xml* para la lectura de datos del fichero XML (Lenguaje de Marcado Extensible).

## Requisitos funcionales

* El usuario será capaz de proporcionar el fichero de datos que desee, siempre y cuando el fichero cumple el esquema railML, y contenga el subesquema de enclavamiento.
* El usuario será capaz de volver a proporcionar un nuevo fichero.
* El dashboard se generará automáticamente, una vez que el usuario proporcione el archivo, sin necesidad de más interacción por parte del usuario.
* El dashboard tendrá una cabecera con el nombre del archivo, del proyecto y datos generales.
* El dashboard contendrá información del subesquema de enclavamiento.
* El dashboard contendrá tablas y gráficos de los elementos del enclavamiento tales como agujas, señales, vías, etcétera.

# Descripción de la herramienta

# Control de configuración

https://w3.ual.es/~rguirado/posi/Tema5-Apartado5.pdf

La gestión de configuraciones del software (GCS) es un conjunto de actividades desarrolladas para gestionar los cambios a lo largo del ciclo de vida. La GCS es una actividad de garantía de calidad del software que se aplica en todas las fases del proceso de ingeniería software.

http://mmorejon.github.io/blog/que-es-la-gestion-de-configuracion/

Son muchas las definiciones existentes sobre esta disciplina. Todo especialista que la ha definido ha aportado nuevos puntos de vista, así como tareas a tener en cuenta. En ocasiones, pudiera existir diferencias dentro de estos conceptos en cuanto a nombres o prioridades entre tareas a realizar, sin embargo, todos señalan la importancia de esta disciplina. Roger S. Pressman la definió de la siguiente manera:

“El arte de coordinar el desarrollo de software para minimizar la confusión se denomina gestión de configuración. La gestión de configuración es el arte de identificar, organizar y controlar las modificaciones que sufre el software que construye un equipo de programación. La meta es maximizar la productividad minimizando los errores”.

Otros autores la han definido como:

“La GCS es una disciplina de la ingeniería de software que comprende herramientas y técnicas (procesos o metodología) que una compañía utiliza para manejar los cambios en sus software activos“.

En 1972 apareció SCCS (Source Code Control System), fue uno de los primeros programas de control de versiones. Fue desarrollado por IBM y después fue adaptado por UNIX.

A SCCS le suceden programas como RCS (Revision Control System, 1982), PVCS (Polytron Version Control System, 1958), CVS (Concurrent Version System, 1990), Clercase (1992), VSS (Visual Source Safe, 1995), Perforce (1995), Accurev (1999), Subversion (2000), Bitkeeper (2000) y finalmente Git (2005).

<https://git-scm.com/>

Git es un sistema de control de versiones libre y “open source” diseñado para manejar todos los proyectos, desde pequeños a grandes proyectos con rapidez y eficiencia.

Para desarrollar este trabajo se ha utilizado Git como sistema de control de versiones.

https://www.deustoformacion.com/blog/programacion-diseno-web/que-es-para-que-sirve-github

Además se utilizará GitHub para alojar el proyecto. GitHub es un sitio web y un servicio en la nube que ayuda a los desarrolladores a almacenar y administrar su código, al igual que llevar un registro y control de cualquier cambio sobre este código.

Hay algunos conceptos de Git que son necesarios para entender los siguientes párrafos. A continuación se defines algunos de los conceptos:

http://www.ictea.com/cs/index.php?rp=/knowledgebase/3481/iQue-es-un-repositorio.html

* **Repositorio**: Un repositorio, depósito o archivo es un sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos.
* **Commit**: Un "commit" es la acción de guardar o subir tus archivos a tu repositorio remoto (una actualización de tus cambios) también puede hacerse al local (depende donde hayas creado tu repositorio). https://codigofacilito.com/articulos/commits-administrar-tu-repositorio
* **Rama**: Las ramas son caminos que puede tomar el desarrollo de un software, algo que ocurre naturalmente para resolver problemas o crear nuevas funcionalidades. En la práctica permiten que nuestro proyecto pueda tener diversos estados y que los desarrolladores sean capaces de pasar de uno a otro de una manera ágil. https://desarrolloweb.com/articulos/trabajar-ramas-git.html

El link del repositorio GitHub para acceder a su contenido es el siguiente:

<https://github.com/adriSM07/railMLdashboard>

En el repositorio se encontraran tres ramas. Una rama “master” en la que se encontrarán versiones finales etiquetadas de la aplicación. Una rama “memory-branch” en la que se han ido haciendo “commits” del documento de la memoria. Y una última rama “development” en la que se ha ido desarrollando la aplicación.

# Evaluación

# Conclusiones

# Líneas futuras

# Referencias y enlaces

# Anexos