GENERACIÓN DE REPORTES Y DASHBOARD PARA SISTEMA DE MONITOREO ESTRUCTURAL

Diego Alfonso Varas Moya
Departamento de Ingeniería Informática
Universidad de Concepción

Nombre Profesor Guía: Gonzalo Rojas Durán
Comisión: Guillermo Cabrera Vives, Julio Godoy del Campo

3 de abril de 2020

Abstract

En este trabajo se aborda el desafío de la visualización de datos y generación de reportes para el monitoreo estructural de puentes; la captura de estos datos se realiza a través de sensores ubicados en puentes que transmiten datos a una plataforma software en una alta frecuencia. En esta memoria, se especificó, diseñó e implementó un sistema de visualización y reportes de datos recientes, utilizando distintas alternativas gráficas y textuales, con capacidad de interacción. Las pruebas de rendimiento realizadas permitieron tomar decisiones de implementación que mejoran la escalabilidad de las visualizaciones.

Índice

1. Introducción

El monitoreo de salud estructural (o SHM, por sus siglas en inglés) se refiere al proceso de implementación de estrategias para la detección y caracterización de daños en infraestructuras como lo son puentes, edificios, entre otros. En particular, este es un proceso que en los puentes de Chile ha dependido únicamente de la inspección visual en terreno, que, aunque puede detectar los problemas evidentes que ya posee un puente, no puede hacer mucho a la hora de detectar los defectos invisibles al ojo humano.

Es en este contexto donde surge el proyecto FONDEF titulado “Plataforma de Monitoreo Estructural de Puentes” (código IT18I0112), adjudicado por la Facultad de Ingeniería, cuyo objetivo principal es el desarrollo de una plataforma de monitoreo de salud estructural para puentes. Se requiere que el sistema transmita y capture datos provenientes de diferentes tipos de sensores, donde se debe poder consultar acerca del estado funcional del sensor, identificarlo y obtener sus metadatos, además de las mediciones hechas por el sensor con su respectivo timestamp (marca de tiempo).

Asimismo, surgen requerimientos de visualización e interpretación de datos, que requieren elegir las mejores formas de interacción, otorgando flexibilidad al usuario para realizar diversas consultas. Para ello, se debe proveer alternativas de visualización que faciliten el análisis de los datos capturados a través de los sensores.

El solo hecho de ver los datos sin procesar, de una red de sensores almacenados en una base de datos, no satisface las necesidades de los usuarios. Los datos deben analizarse y mostrarse al usuario de una manera que permita obtener fácilmente información de ellos, especialmente los datos recientes, ya que en ellos se encuentra la información más actual de la estructura, la que al ser visualizada correctamente puede proveer al usuario información crucial del estado de la estructura sensorizada.

Obtener datos de mediciones de estructuras sensorizadas reales representa un desafío esencial para el desarrollo de este trabajo, ya que en la mayoría de los casos estos datos no son de uso público y además son muy escasos. Tener a disposición datos reales para el desarrollo de este subsistema es clave para tener una visión inicial de lo que se espera de él y probar su rendimiento. Por otra parte, las alternativas de visualización que se provean deberán ser escalables, permitiendo soportar un número variable de sensores, tipo de estructura (tipos de puentes), frecuencias de medición y cantidad de datos almacenados.

1.1. Objetivo General

Implementar alternativas de visualización de datos almacenados, los cuales previamente debieron ser obtenidos a partir de una infraestructura de monitoreo estructural de puentes, otorgando la capacidad de interacción al usuario en sus consultas, de acuerdo a requerimientos definidos.

1.2. Objetivos Específicos

- Proveer alternativas de visualización de datos recientes según los requerimientos de la plataforma de monitoreo estructural de puentes.

- Dotar al experto de monitoreo de capacidades interactivas de exploración de los datos a través de múltiples alternativas de visualización.

- Permitir la generación de reportes imprimibles a partir de una visualización de datos reciente, para su análisis por parte de autoridades y agentes relevantes.

1.3. Alcances y limitaciones

- Este trabajo consiste en la creación de un módulo de visualizaciones históricas recientes, el cual va a formar parte de un sistema completo de SHM, que está actualmente en desarrollo.

- Para validar las visualizaciones y la generación de reportes, se implementó un dashboard mediante un framework web en una máquina portátil de bajos recursos.

- Como fuentes de datos, se utilizaron conjuntos de datos representativos obtenidos de un caso de estudio realizado en China [4], los cuales son almacenados en una base de datos.

- Los datos deberán ser obtenidos a través de sensores y almacenados en una base de datos montada en un servidor, esto último escapa del alcance de este trabajo.

- Las visualizaciones de tipo 3D se escapan del alcance de este trabajo.

- No se utilizarán datos en tiempo real para las visualizaciones de este trabajo, sólo datos de tipo históricos, ya almacenados previamente en una base de datos.

1.4. Descripción del Informe

El presente informe consta de cuatro secciones principales. Primero se revisaron los sistemas actuales de SHM y las posibles herramientas para la construcción del subsistema de visualización de datos recientes, en la siguiente sección se detalla los requerimientos para las visualizaciones y los reportes, luego se detalla el desarrollo de la solución, que involucra el diseño e implementación del subsistema, para finalmente realizar una evaluación del rendimiento para generar las visualizaciones del subsistema.

2. Discusión Bibliográfica

La implementación de capacidades de visualización de datos recientes requiere de un repositorio desde donde se extraigan los datos a desplegar, alternativas de visualización que provean información complementaria de diferentes sensores, y de estrategias de escalabilidad que permitan ofrecer visualizaciones en un tiempo razonable, atendiendo la frecuencia de muestreo de los datos recibidos. En este capítulo se presenta una revisión de los principales conceptos asociados al Monitoreo de Salud Estructural, con referencias bibliográficas y revisión de herramientas disponibles que abordan los mencionados requerimientos (que son detallados en Capítulo 3).

Las pruebas e inspecciones de estructuras como puentes, edificios, presas, etc., se han llevado a cabo desde que la humanidad comenzó a realizar construcciones complejas. El monitoreo de salud estructural, conocido como SHM (Structural Health Monitoring según sus siglas en inglés), se origina en la industria mecánica, aeronáutica y aeroespacial como se indica en el artículo “A review of structural health” [7]. El objetivo de esta disciplina, es monitorear continuamente el comportamiento de partes críticas de una estructura y obtener advertencias tempranas que puedan ser corregidas de inmediato o en una instancia de mantenimiento normal, dependiendo de la gravedad del problema detectado.

Para la Ingeniería Civil, SHM es un área relativamente nueva, encontrándose artículos relacionados aproximadamente desde el año 1996 [7]. Sin embargo, los últimos desarrollos, la disminución del costo de los sensores y los avances en tecnologías de la información (TI), han hecho que los sistemas SHM ofrezcan mayores prestaciones, propiciando su uso en la Ingeniería Civil. El presente trabajo se centra en el “Monitoreo de la Salud Estructural de Puentes”.

Un caso de estudio realizado en el año 2008, de un puente colgante ubicado en la ciudad de Harbin, China [4], presenta un problema de benchmark (prueba de rendimiento) con datos obtenidos a través de su propio sistema de SHM, mediante sensores posicionados sobre el mismo puente. Los datos y el estudio se centran sobre la detección de daños en algunas vigas y la evaluación del estado de los cables de sujeción del puente, durante un periodo de tiempo que comprendió desde que el puente se encontraba en un estado “sano”, hasta que comenzó a presentar daños estructurales visibles. Estos datos sirven como guía para comparar las condiciones de este puente con uno similar y así poder detectar posibles daños.

En otro caso de estudio, [21] se desarrolló una plataforma de procesamiento y análisis de datos en tiempo real para el monitoreo estructural de puentes. Esta plataforma consta de módulos de procesamiento, análisis y visualización de datos, todos integrados a través de interfaces gráficas de usuario (GUI). Las aplicaciones están diseñadas y adaptadas para ejecutarse en tiempo real al ordenar automáticamente los datos entrantes y redirigirlos a los módulos de procesamiento para generar visualizaciones de los desplazamientos y los movimientos del puente. Con esta capacidad, después de la ocurrencia de algún evento extremo como tormentas de viento, terremotos o impactos de barcos, es posible evaluar la condición del puente de manera oportuna para así poder prevenir mayores daños en este.

Para los sistemas de SHM existen diversos tipos de sensores lo cuales dependen de la estructura que se desee monitorear. En el caso del monitoreo de la salud estructural de puentes, los sensores más utilizados son los medidores de tensión Carlson (mide las deformaciones internas de una estructura), strain gauge (mide la deformación, presión, par y carga de estructuras, existen de tipo alambre vibrante o de lámina), inclinómetros (miden la inclinación del plano con respecto de la horizontal en una estructura), sensores de grietas y/o juntas (sirve para medir el desplazamiento de grietas y/o juntas), sensores de temperatura (sirve para medir la temperatura ambiente) y acelerómetros (sirve para medir la vibración o aceleración del movimiento de una estructura, existen de tipo piezoresistivos, piezoeléctricos, capacitivos o de pozo), siendo este último tipo de sensor el más utilizado. Para los datos que proporcionan estos tipos de sensores, las visualizaciones más utilizadas [22] son las de tipo Lineplot (gráfico de líneas), junto con las de tipo Barcharts (gráfico de barras) o Histogramas. Estas visualizaciones se suelen usar para representar el cambio de los datos medidos por los sensores en el tiempo, junto con ver la tendencia y periodicidad de valores en los conjuntos de datos visualizados.

Por otro lado, en nuestro país, según el informe de la Dirección de Vialidad del año 2018 [8], existen 13 puentes que presentan una estructura dañada, lo que implica que deben realizarse acciones en el corto plazo (menos de 1 año) para repararlos. Estos 13 puentes representan un 1,35% de la muestra que fue revisada (1.034 puentes), y corresponden a aquellos puentes de largo mayor a 30 metros.

Actualmente en Chile no existe un sistema de monitoreo de salud estructural de puentes que maneje el Ministerio de Obras Públicas. Este, solo cuenta con monitoreos visuales en terreno, para definir el estado de salud de un puente.

2.1. Sistemas actuales de SHM

Alrededor del mundo se utilizan diferentes sistemas de SHM, en donde se procesan datos y metadatos obtenidos a través de sensores para su posterior visualización. Para los usuarios de SHM (propietarios, consultores, proveedores de tecnología, contratistas, investigadores, etc.) la visualización es comúnmente un "cuello de botella" [23] que puede conducir a una utilización ineficiente del software de SHM o incluso abandonar el uso del sistema. Para que esto no ocurra, resulta indispensable conocer herramientas actuales utilizadas para este propósito. De acuerdo a esto, para el desarrollo del presente trabajo, se revisaron las herramientas utilizadas para el Monitoreo de la Salud Estructural de Puentes, las que ofrecen alternativas de visualización de datos obtenidos de sensores.

Uno de los softwares más utilizados en SHM es ARTeMIS Modal, desarrollado por la empresa Structural Vibration Solutions A/S [3]. Un ejemplo de ello es la monitorización a través de sensores (acelerómetros, tensiómetros y estación meteorológica) del campanario de la Iglesia de San Vittore, Arcisate, Italia [9]. Este software, dentro de su módulo base de administración de datos (Data Manager Base Module), posee 4 módulos internos destinados a SHM (Detección de daños, Historial de parámetros modales, Análisis de deriva y Adquisición de datos). La visualización de datos obtenidos de sensores se concentra en estos 4 módulos, en donde, al ingresar a cada uno de ellos, se muestran gráficos acordes a los datos seleccionados, dentro de un rango de medición. La mayoría de las visualizaciones son materializadas por gráficos de barra (barchart) y de línea (lineplot) personalizados por el usuario (Figura 2.1), que pueden incluir fórmulas matemáticas, que por ejemplo permitan el suavizado de los datos. También es posible calcular métricas que puedan ser útiles para el usuario. El resto de las visualizaciones son de tipo 3D, lo cual se escapa al ámbito de este trabajo.

Figura 2.1: Visualización de datos obtenidos de sensores, mediante el software ARTeMIS Modal [Fuente: http://www.svibs.com/SHM]

El software ARTeMIS no tiene la capacidad de recibir y mostrar datos en tiempo real, por lo que, para que haya una actualización de estos, deben ser cargados con anterioridad a una base de datos.

Algunos módulos poseen alertas, posiblemente para indicar daños, accionadas al superar umbrales establecidos por el usuario o determinados por el mismo software, además en las últimas versiones se ha añadido un apartado aún en fase experimental de análisis de datos mediante machine-learning.

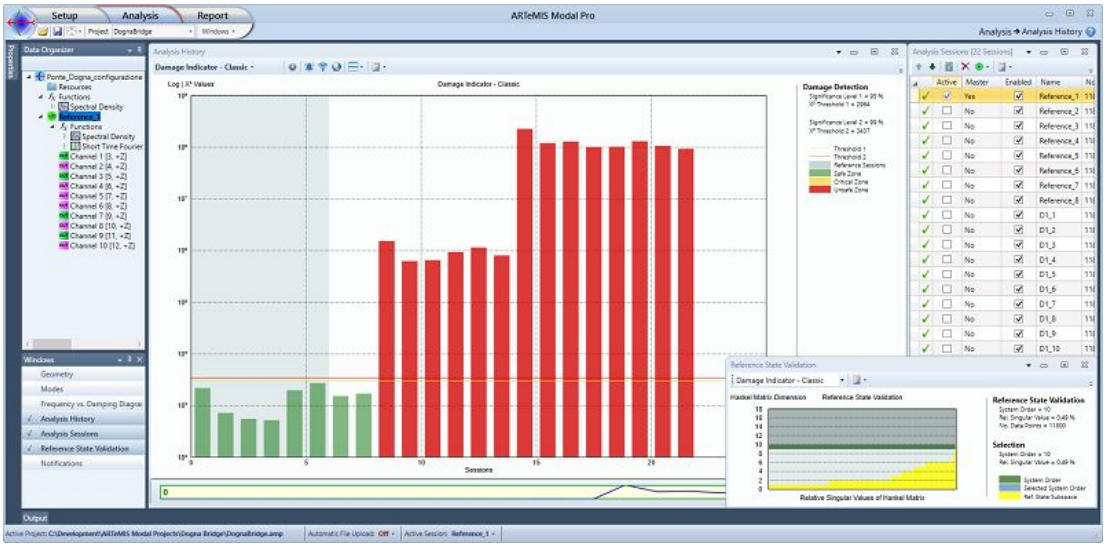

Figura 2.2: Visualización de detección de daños de un puente italiano, mediante el software ARTeMIS Modal [Fuente: http://www.svibs.com/SHM]

En el gráfico de la Figura 2.2, se pueden apreciar 8 mediciones de referencia (sin daños) indicadas con las barras verticales verdes que permanecen por debajo de los umbrales, junto a ellas, es posible visualizar 14 mediciones del puente dañado, indicadas por las barras verticales de color rojo, en donde todas superan los umbrales significativamente. La Figura 2.2, también muestra el diagrama de validación de estado de referencia utilizado para validar el modelo de referencia (Costado inferior derecho).

Por otro lado, en cuanto a los archivos requeridos para cargar los datos obtenidos de sensores, ARTeMIS Modal admite varios formatos de entrada, en particular se centra en utilizar archivos en UFF (Universal File Format), en donde los datos deben tener como formato de codificación de caracteres a UTF-8 y además estar organizados en columnas separadas por un espacio, en donde la primera fila es el nombre de la columna.

Otra alternativa es BRIMOS (Bridge Monitoring System), creado por la empresa VCE Vienna Consulting Engineers ZT GmbH. Este Software ofrece un método para la identificación de sistemas, detección de daño en puentes [1] y otras estructuras civiles. En cuanto a las visualizaciones, BRIMOS funciona de la misma manera que ARTeMIS, al igual que en los requerimientos de datos.

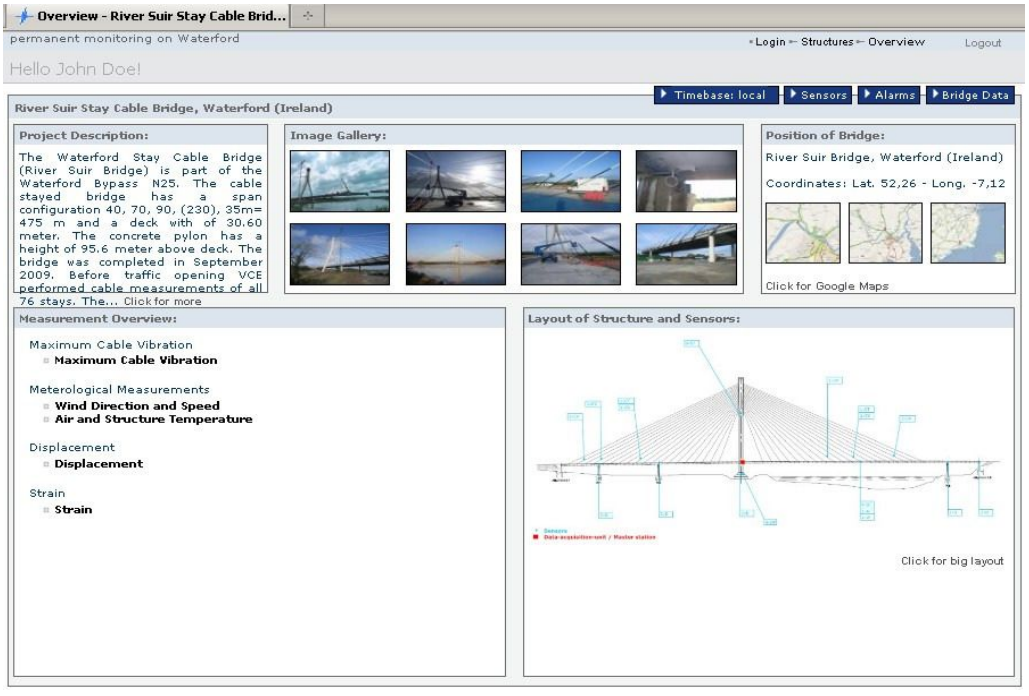

Figura 2.3: Dashboard del software BRIMOS, del puente sobre el río Suir (Waterford, Irlanda) [Fuente: http://www.brimos.com/BRIMOS/HTML/en/press/publications.html]

En la Figura 2.3, se muestra la pantalla de inicio del software BRIMOS, que proporciona detalles del puente. En la sección ubicada en parte inferior derecha, es posible visualizar los sensores instalados en el puente, en donde se puede cliquear para ingresar a la visualización de los datos medidos en diferentes formas de presentación. Los datos se pueden mostrar en diferentes períodos de tiempo, lo que permite evaluar el cambio en el comportamiento de una estructura de una temporada a la siguiente, o de un día a otro.

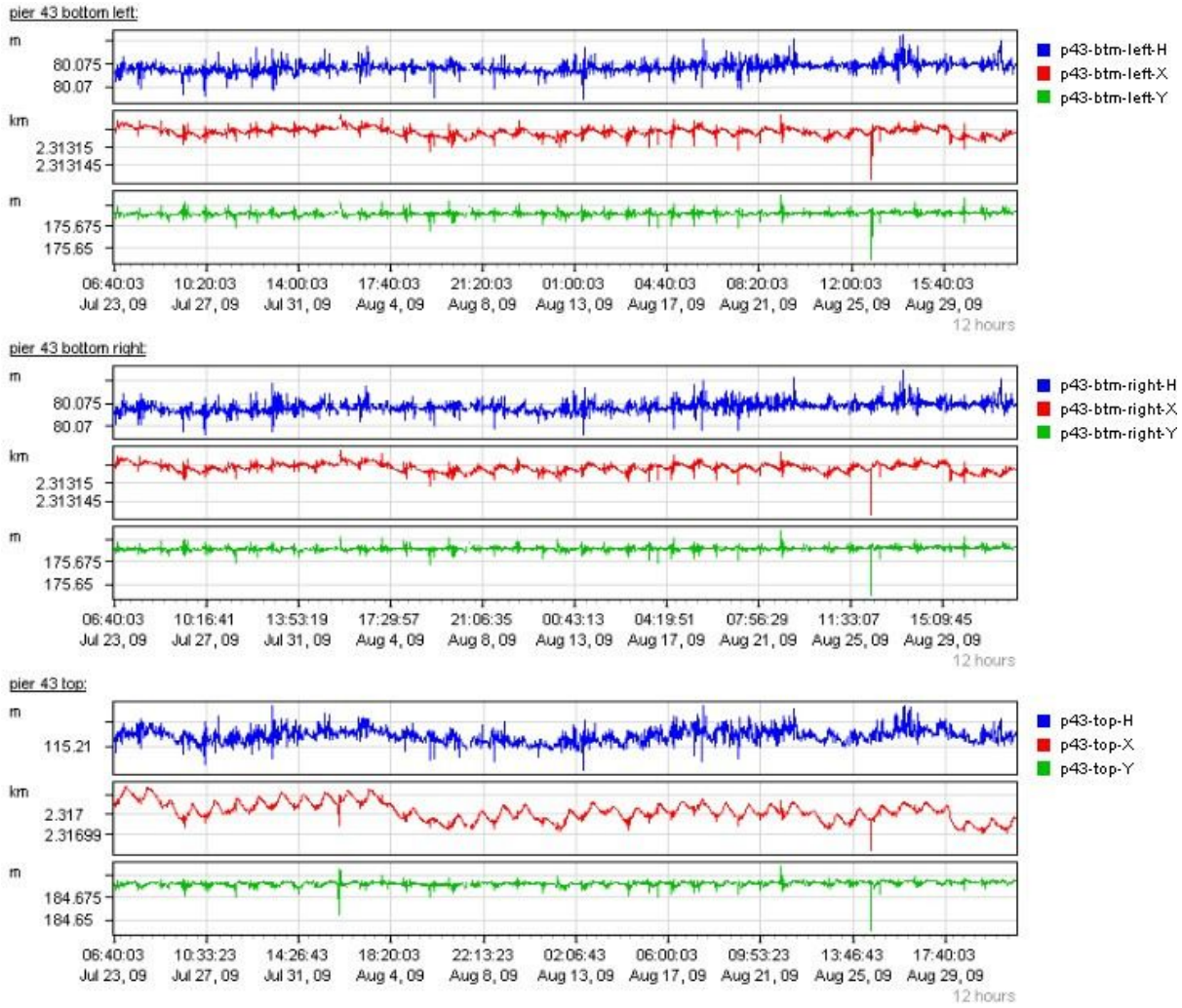

Figura 2.4: Ejemplo de Visualización de datos del Puente Beska Danubio (SERBIA) [Fuente: http://www.brimos.com/BRIMOS/HTML/en/press/publications.html]

Los datos presentados en la Figura 2.4 de forma gráfica, también pueden ser presentados de forma tabular, se pueden exportar los datos en formato de hoja de cálculo para su uso posterior, o descargar información adicional del puente, como informes generados en línea o resultados de evaluación de riesgos.

Por último, el sistema SHM Live [2] es un sitio web acoplado con una base de datos que administra y despliega datos monitoreados en tiempo real en cualquier lugar del

mundo. La base de datos puede recibir datos de muchos sistemas y sensores. La información se incluye en la base central desde una unidad que se encarga, a su vez, de recolectar los datos directamente de todos los sistemas de adquisición de datos instalados. Dada su funcionalidad basada en la visualización de datos en tiempo real, este sistema de monitoreo escapa del ámbito en el cual se desarrolla este trabajo.

Los sistemas ARTeMIS Modal, BRIMOS y SHM Live están entre los sistemas más completos e integrales para el monitoreo de puentes a nivel mundial, de entre muchos otros. Cabe mencionar que estos son proyectos comerciales y no son sistemas de código abierto (open source). Estos sistemas son además instalados a la medida para cada estructura a la cual fueron adquiridos, incluyendo la sensorización de la estructura en algunos casos. Las empresas detrás de estos, son las encargadas del mantenimiento del sistema y de los equipos según corresponda, además de la capacitación del personal para su utilización.

2.2. Alternativas para la creación de un Dashboard de SHM

Debido a los continuos avances en las tecnologías de la información y la comunicación a un ritmo acelerado cada vez se manejan más y más datos. A menudo quienes trabajan con datos están abrumados con la información producida en cada segundo por una multitud de sistemas. Este fenómeno es generalmente conocido como sobrecarga de información. El problema se agrava aún más cuando la manera de presentar la información es deficiente. Los dashboard pueden ofrecer una solución al problema de sobrecarga de información, al proporcionar una herramienta de gestión de la información que monitoriza, analiza y muestra de manera visual los indicadores clave de desempeño o KPI (Key Performance Indicator), métricas y datos fundamentales para hacer un seguimiento del estado o cambios de estos en el tiempo, incluyendo además la gestión del rendimiento de datos e incorporando varios conceptos y aplicaciones como mapas de estrategia y cuadros de mando.

Para las plataformas de SHM es crucial tener un dashboard que permita visualizar de la mejor manera la información extraída de los sensores, en donde dada la cantidad de sensores que pueden estar en una estructura, es posible que se produzca una sobrecarga de información, la cual debe ser canalizada correctamente para ser llevada a los usuarios.

Actualmente existen variadas plataformas y librerías de lenguajes de programación que forman frameworks, los cuales nos permiten utilizar y diseñar un dashboard para tener un sistema propio de SHM.

2.2.1. Suit ELK

La suite ELK que significa Elasticsearch, Logstash y Kibana [10], es un conjunto de herramientas de código abierto (open source) que se combinan para crear una herramienta de administración de registros que permite la monitorización, consolidación y análisis a partir de datos sin procesar.

- **Elasticsearch** es una base de datos distribuida en donde toda la información se organiza en nodos. Al igual que distribuye la información, distribuye el procesamiento. Cuando se realiza una consulta o búsqueda y esa información se encuentra distribuida, será cada nodo el que procese dicha información y devuelva los resultados. Por tanto, podemos obtener mejores rendimientos. A su vez, es un motor de búsqueda avanzado, en el cual se puede filtrar todo tipo de datos a través de una API simple. La API es RESTful (es una arquitectura que se ejecuta sobre HTTP y hace referencia a un servicio web que la implementa), por lo que no solo puede usarse para el análisis de datos, sino también para producción de aplicaciones basadas en la web.

- **Logstash** es la parte de preprocesamiento antes de guardar la información en Elasticsearch. Es una herramienta destinada a organizar y buscar archivos de registro donde se trabajarán los eventos, antes de ser almacenados en la base de datos. Pero también se puede usar para limpiar y transmitir grandes volúmenes de datos desde todo tipo de fuentes a una base de datos.

- **Kibana** es la interfaz visual para Elasticsearch que funciona en el navegador web. Sirve para visualizar los datos almacenados en Elasticsearch y generar los dashboard personalizados.

Lo que se busca al utilizar ELK es tomar toda la información, procesarla y almacenarla de forma distribuida, así es posible escalar a Big Data y obtener buenos rendimientos con grandes volúmenes de datos, transformándolos en visualizaciones, en las que se pueden identificar anomalías, comportamientos, eventos, peaks, etc., tanto de forma gráfica y visual, como en reportes.

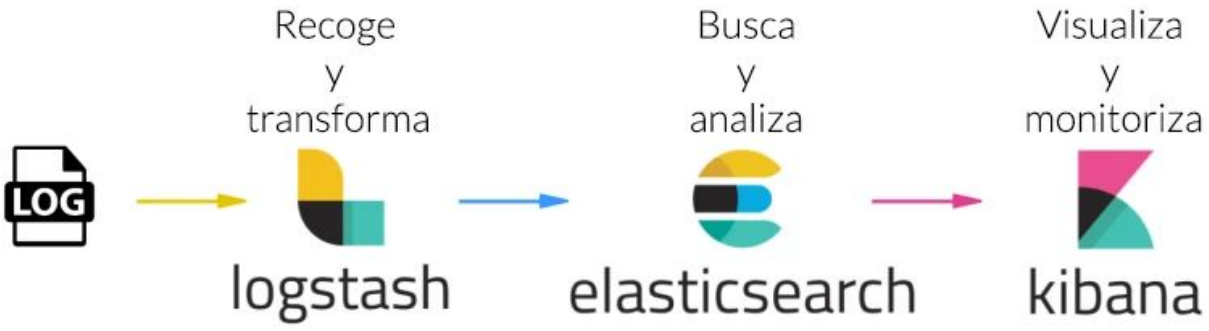

Figura 2.5: Esquema del funcionamiento de la Suite ELK
[Fuente: https://www.elastic.co/es/what-is/elk-stack]

En resumen, lo que la Suite ELK hace es:

- Recolectar logs de eventos y de aplicaciones.
- Procesar dicha información y la pone a disposición de quienes lo necesiten.
- Tiene módulos de seguridad para implementar que cada usuario tenga niveles de acceso, evitando que se acceda a información que no le corresponda.
- Da formato a los campos y los convierte en opciones de búsqueda y filtrado, ya que hay logs que tienen formatos irregulares, ya sean de aplicaciones propias o de servicios que no siguen ningún formato. Dichos logs son prácticamente cadenas de texto muy complicadas de entender.
- Presentar toda la información en visualizaciones, donde se pueden realizar búsquedas, filtrados, agregaciones y ver la información mucho mejor que módulos de texto.

La Suite ELK fuerza la utilización de su dashboard propio (Kibana) para la visualización de los datos, sin libertad de acción para utilizar una visualización creada en Kibana en una plataforma propia, y solo dando como opción exportar la visualización como una imagen o un pdf. Además, la Suite ELK agrega su propio motor de búsqueda (Elasticsearch) con una base de datos integrada, el cual también tiene limitantes, como solo soportar respuestas de tipo json, no permitiéndole otros lenguajes como xml o csv y sufrir en algunas ocasiones pérdida de integridad de datos (o split-brain).

2.2.2. Lenguaje de Programación Python

Python [15] es un lenguaje de programación ampliamente utilizado, incluso para modelos de datos y archivos de configuración. Las 3 herramientas de diseño web o frameworks web que se mencionan a continuación se basan en él y nos proporcionan diferentes alternativas para el diseño y creación de un dashboard.

Un framework web es un conjunto de componentes que ayudan a desarrollar sitios web de manera fácil y rápida. Permite manejar la creación de páginas web, autenticación de usuarios (registrarse, iniciar sesión, cerrar sesión), administración del sitio web, como subir archivos, etc.

2.2.2.1. Django

Django [11] es un framework de aplicaciones web gratuito y de código abierto (open source) escrito en el lenguaje de programación Python.

Django permite simplificar el proceso de creación de sitios web o dashboard, que a menudo es muy complejo. Enfatiza la reutilización de componentes, utilizando menos código y privilegia el desarrollo rápido. También proporciona una interfaz creativa, actualizada, de lectura y eliminación generada mediante introspección y construida a través de modelos de administrador. El framework Django utiliza su propio modelo de usuario incorporado, el cual facilita la autorización y autenticación de API.

Desde su sitio web [11] podemos ver algunos sitios destacados creados en Django como National Geographic, Disqus, Instagram, Mozilla Foundation y Pinterest, que son webs con un altísimo tráfico de datos y usuarios.

Si bien Django es un potente framework para la creación de sitios web, no está orientada a la creación de sitios analíticos, teniendo que utilizar distintas librerías como complementos para intentar lograr esta tarea, no teniendo en cuenta si estas tienen una total integración con este framework.

2.2.2.2. Flask

Flask [12] es un framework web mucho más genérico que Django, que se puede usar para construir casi cualquier aplicación, incluyendo sitios web o dashboard. Está diseñado para que al comenzar su aprendizaje sea rápido y fácil, con la capacidad de ir escalando a la creación de aplicaciones cada vez más complejas. Comenzó como una simple envoltura o capa por sobre Werkzeug [13] y Jinja [14] y se ha convertido en uno de los framework de aplicaciones web de Python más populares. Flask ofrece sugerencias, pero no impone ninguna dependencia o diseño del proyecto. Depende del desarrollador elegir las herramientas y bibliotecas que desea utilizar. La comunidad ofrece muchas extensiones que facilitan la adición de nuevas funciones.

Sin embargo, su API (Application Programming Interface) es bastante limitada en el ámbito analítico, por lo que se tendría que codificar mucho o usar variadas extensiones que funcionan como si estuvieran integradas en el propio Flask. La necesidad de extensiones surge para suplir módulos que el framework no posee, como, por ejemplo, manejo de cargas, validaciones de formularios, tecnologías de autenticación abiertas y muchas herramientas más, que en cambio Django si posee.

2.2.2.3. Plotly Dash

Plotly Dash [16] como se menciona en su sitio web es un framework de Python para crear aplicaciones web analíticas y dashboard. Está construido sobre Plotly.js [17], React [18] y Flask [12]. Vincula elementos modernos de la interfaz de usuario como menús desplegables, controles deslizantes y gráficos a su código analítico de Python. Es ideal para crear aplicaciones de visualización de datos con interfaces de usuario altamente personalizadas en Python puro. Es particularmente adecuado para cualquier persona que trabaje con datos en Python.

A través de un par de patrones simples, Plotly Dash extrae todas las tecnologías y protocolos necesarios para crear una aplicación interactiva basada en la web. Es lo suficientemente simple como para que pueda vincular una interfaz de usuario alrededor de su código Python en poco tiempo.

Si bien las aplicaciones Dash se ven en el navegador web, no se tiene que escribir ningún Javascript o HTML. Al utilizar el navegador lo convierte en multiplataforma y por ende está listo para dispositivos móviles si es requerido.

Es complementario a herramientas de BI como la Suite ELK. Estas herramientas funcionan muy bien para datos estructurados. Pero cuando se trata de análisis y transformación de datos, es difícil superar la amplitud y flexibilidad de lenguajes de programación y comunidades como la de Python. Plotly Dash elimina muchas de las complejidades en la creación de interfaces de usuario, lo que le permite crear un hermoso front-end (Capa de presentación, que el usuario final ve) para su back-end (Capa de acceso a datos) de análisis de datos personalizado.

Plotly Dash es una biblioteca de código abierto (open source), lanzada bajo la licencia permisiva del MIT (Instituto Tecnológico de Massachusetts). Plotly desarrolla Dash y ofrece además una plataforma para implementar fácilmente aplicaciones en un entorno empresarial.

2.2.3. Almacenamiento de datos

Si bien antes se mencionó que la Suite ELK, a través de su módulo Elasticsearch, provee una base de datos, su utilización se limita a los propios componentes de la Suite ELK, dejando de lado las demás opciones propuestas, como Django, Flask y Plotly Dash. Es por esto que se ve como alternativa la utilización como motor de base de datos a PostgreSQL junto con su extensión TimescaleDB.

PostgreSQL [19] es un potente sistema de base de datos relacional de objetos de código abierto (open source) con más de 30 años de desarrollo activo que le ha valido una sólida reputación de fiabilidad, solidez de características y rendimiento.

TimescaleDB [20] mejora el uso de PostgreSQL, cuando se trabaja con datos de series temporales. Creado en PostgreSQL y probado para cargas de trabajo en los entornos más exigentes. Debido a estas características y a las funciones analíticas de series de tiempo incorporadas, TimescaleDB aumenta la productividad del desarrollador y optimiza la eficiencia de los recursos.

Dado que este trabajo se centra en la visualización de los datos, no es de relevancia indagar en más alternativas para el almacenamiento de los datos.

2.3. Discusión

A nivel nacional es de gran importancia implementar un sistema para el Monitoreo de la Salud Estructural de puentes. Alrededor del mundo se hace monitoreo constante sobre estas estructuras, utilizando herramientas como ARTeMIS, BRIMOS o SHM Live de manera tal de evitar la ocurrencia de accidentes que pongan el riego la vida. Las herramientas ya creadas para este propósito, son de un alto costo y a su vez dejan amplias áreas del manejo de los sistemas, mantenimiento y sensorización a las empresas que brindan estos sistemas.

Por esto es necesario construir una plataforma propia y a la medida, evitando vincularse con alguna empresa externa y teniendo el control total de todo el sistema creado, para eso es necesario elegir una herramienta adecuada a las necesidades del sistema a crear, que sea escalable y que permita una integración con los sistemas que actualmente existen.

Para esto, según la información expuesta durante esta sección, se pueden tener las siguientes reflexiones para la elección de esta herramienta:

- La Suite ELK nos brinda una completa solución para crear un sistema de SHM, que comprende desde la base de datos, hasta la visualización de estos, sin embargo, su poca personalización y nula integración con otras plataformas, para utilizarla como complemento de un sistema complejo, hace que no sea una buena opción.
- Es posible usar Flask o Django para crear todo tipo de aplicaciones web, si bien son herramientas poderosas, carecen de módulos específicos para crear webs analíticas. Estos módulos, si bien pueden ser creados o añadidos a través de extensiones, en algunos casos puede que su integración no sea completa llevando a fallos o tener que realizar un trabajo adicional para su correcto funcionamiento.
- Dash en cambio es un framework que se utiliza específicamente cuando necesitamos construir una aplicación analítica, el cual, es el fin de este trabajo, por ende, es una muy buena alternativa, ya que integra los módulos analíticos que no integra Django ni Flask.
- A diferencia de la Suite ELK, las demás opciones para construir un dashboard analítico requieren de un motor de base de datos, para poder extraer lo necesario para mostrar en las visualizaciones, es por eso que se propone el uso de PostgreSQL con su extensión TimescaleDB.

3. Requerimientos

En esta sección se presentan los requerimientos para la creación del subsistema de visualización de datos recientes de la plataforma de monitoreo estructural, cuyo principal objetivo es poder brindar al usuario un módulo que le permita visualizar datos de tipo históricos dentro de un rango de tiempo máximo de 14 días. Los requerimientos se dividen en dos tipos, que son: de visualización y de generación de reportes.

3.1. Requerimientos de visualización

Para la creación del subsistema es necesario definir visualizaciones para cada tipo de sensor, las cuales pueden ser comunes entre sensores.

A los sensores que se le deben definir visualizaciones son:

- Acelerómetro, es un sensor que sirve para medir la fuerza de aceleración, ya sea estática o dinámica, estos sensores son útiles para medir vibraciones y movimientos en un sistema.

- Strain Gauge, es un sensor utilizado para medir la deformación de un objeto.

- Inclinómetro, es un sensor que permite medir con exactitud inclinaciones horizontales y ángulos.

- LVDT, es un sensor utilizado para medir desplazamientos lineales.

- Weather Station, es un conjunto empaquetado de sensores que miden diversas variables meteorológicas como temperatura, humedad, velocidad del viento, presión, etc.

Todas las visualizaciones que se implementen deben ser capaces de comunicar la información o ideas complejas de forma clara, precisa y eficiente, de modo que ayude a los usuarios a analizar y razonar sobre datos expuestos. La creación de estas visualizaciones está sujeta a la disponibilidad de datos de los diferentes tipos de sensores antes mencionados.

Para lograr lo anterior, se tienen cuatro puntos que son claves para generar una visualización, estos son:

- Tener un Dataset limpio, esto permite disponer de un conjunto de datos limpio en el formato adecuado que requiere la visualización.
- Transmitir un mensaje único, no mezclar información que no tiene relación y según el tipo de gráfico elegir el nivel de precisión necesario a mostrar.
- Elegir el gráfico adecuado, la visualización elegida va en función de lo que queremos transmitir.
- Diseño y color, permite destacar lo que interesa de forma clara y fácil de entender, mediante el uso de colores.

Se espera tener diferentes tipos de visualizaciones disponibles, las que pueden ser:

- Lineplot
- Boxplot
- Histograma
- Barcharts

Además, es necesario añadir interacción a las visualizaciones, ya que este hecho facilita y amplía la comprensión de la información que queremos transmitir.

Las interacciones necesarias son:

- Seleccionar 1 o más sensores a visualizar
- Seleccionar la ventana de tiempo a visualizar (1 hora, 1 día, 7 días y 14 días)
- Seleccionar la fecha y hora según corresponda de los datos a visualizar
- Agregar propiedades a las visualizaciones (Líneas de control y detección de peaks)
- Navegación dentro de las visualizaciones (Mover el gráfico, seleccionar partes de este, zoom y ver detalles al seleccionar puntos específicos)

3.1.1. Lista de Requerimientos

Los requerimientos de visualización descritos se resumen en la siguiente lista:

1. El sistema debe procesar los datos provenientes de los sensores y previamente almacenados en una base datos, para generar datasets limpios que faciliten su visualización.
2. El sistema debe proveer visualizaciones para los diferentes tipos de sensores a partir de estos datasets.

3. El sistema debe proveer visualizaciones comunes para todo tipo de sensor considerado y específicas para facilitar la visualización de tipos de sensores que lo requieran.
4. El sistema debe proveer al usuario la capacidad de interactuar con las visualizaciones provistas para facilitar el análisis de acuerdo a sus propios requerimientos.

3.2. Requerimientos de la generación de reportes

Los reportes a generar en el subsistema, deben ser informes que organicen y exhiban información contenida en las visualizaciones creadas. Su función principal debe ser aplicar un formato determinado a los datos para mostrarlos por medio de un archivo que sea fácil de interpretar por los usuarios.

El reporte, de esta forma, debe conferir una mayor utilidad a los datos, realizando un resumen de estos en conjunto con las visualizaciones correspondientes.

En resumen, las características que deben tener los reportes son:

- Claridad
- Concisión
- Debe tener un orden lógico
- Debe ser breve

Por todo lo anterior, se entiende que estos documentos son una herramienta importante en cualquier empresa, ya que gracias a ellos se plasma información que puede ser relevante y analizada posteriormente por los usuarios.

3.2.1. Lista de Requerimientos

Los requerimientos de reportes descritos se resumen en la siguiente lista:

1. El reporte debe generarse a partir de la información que se está visualizando.
2. El reporte debe contener información relevante y resumida de los datos visualizados.
3. El reporte debe contener las visualizaciones creadas.
4. El reporte debe ser imprimible.

4. Desarrollo de la solución

En esta sección, se aborda el diseño e implementación del subsistema de visualización de datos recientes.

4.1. Diseño

Para el diseño del subsistema primero se determinó su arquitectura, para luego definir diferentes alternativas de visualización, composición del dashboard y composición de reportes.

4.1.1. Arquitectura completa del sistema SHM

Como el subsistema de visualización de datos recientes a implementar formará parte de un sistema completo de SHM, este sistema tiene ya definida su arquitectura de hardware y software. En el equipo de desarrollo del proyecto FONDEF en el que se enmarca este trabajo, se definió la siguiente arquitectura global (Figura 4.1).

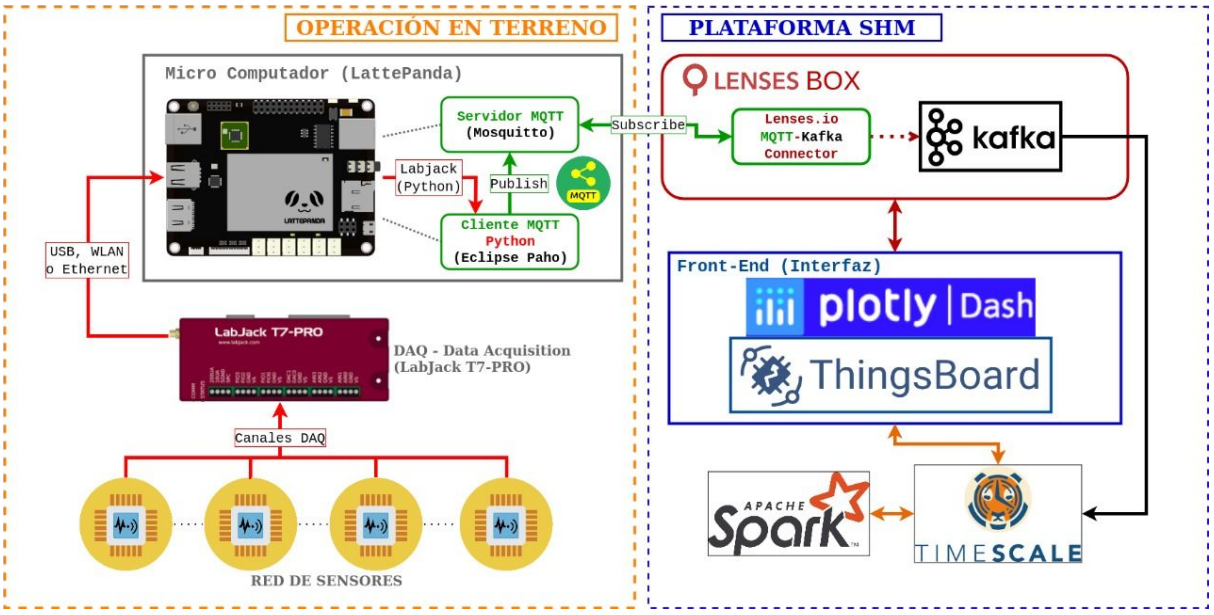

Figura 4.1: Arquitectura de la plataforma

La arquitectura de la plataforma de SHM se divide en dos partes: primero la infraestructura de sensores que se encuentra en terreno (Figura 4.1, izquierda), que comprende una red de sensores, que son manejados a un “DAQ” (Data Acquisition System), el cual por medio de una conexión por USB, WLAN o Ethernet transmite a

un microcomputador, que preprocesará los datos para ser enviados por medio de un servidor con protocolo “MQTT” (MQ Telemetry Transport) a la plataforma de SHM.

Como segunda parte de esta arquitectura, se encuentra la plataforma software de SHM (Figura 4.1, derecha). En esta parte, se encuentra el almacenamiento y visualización de los datos. Primeramente, los datos son recibidos por “Apache Kafka”, que mantiene una cola de mensajes, bajo el patrón publicación-suscripción, para que luego estos sean almacenados en la base de datos construida en “PostgreSQL” con su complemento “TimescaleDB”. Así, al momento de requerir estos datos, se garantice la eficiencia y consistencia de la información entregada. Por sobre esto se encuentra la herramienta “Things Board”, que permite crear visualizaciones en tiempo real y por otro lado aparece lo relacionado con este trabajo que consiste en la implementación de un subsistema de visualización de datos recientes. Para su creación, se optó por utilizar el framework “Plotly Dash” (como se muestra en la Figura 4.1), el cual permite diseñar aplicaciones web analíticas como se mencionó anteriormente.

4.1.2. Arquitectura del subsistema de visualización de datos recientes

El uso del framework Plotly Dash fuerza la utilización de la arquitectura Modelo - Vista - Presentador (MVP), la cual es un patrón de arquitectura para implementar interfaces de usuario derivado de la arquitectura Modelo - Vista - Controlador (MVC). El MVP divide una aplicación en tres partes: el Modelo, la Vista y el Presentador, estas partes están conectadas entre sí a través del presentador, esto permite aumentar la modularidad en el desarrollo simultáneo y la reutilización del código.

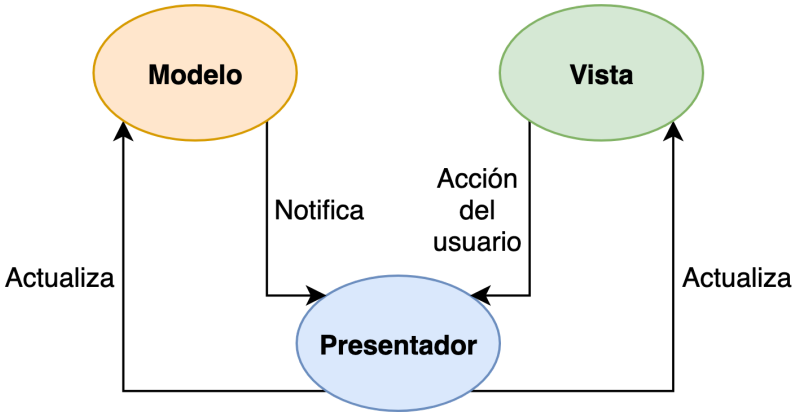

Figura 4.2: Interacción entre componentes

Dada la Figura 4.2, la interacción de componentes se describe de la siguiente manera:

- El modelo almacena datos que se recuperan según instrucciones del presentador.
- La vista genera resultados para el usuario en función de los cambios provistos por el presentador, la vista no tiene conocimiento del modelo.
- El presentador actúa tanto en el modelo como en la vista; envía instrucciones al modelo para actualizar su estado y a la vista para cambiar la información que es presentada a los usuarios.

La arquitectura MVP recién mencionada, es posible llevarla a la implementación del subsistema de la siguiente manera:

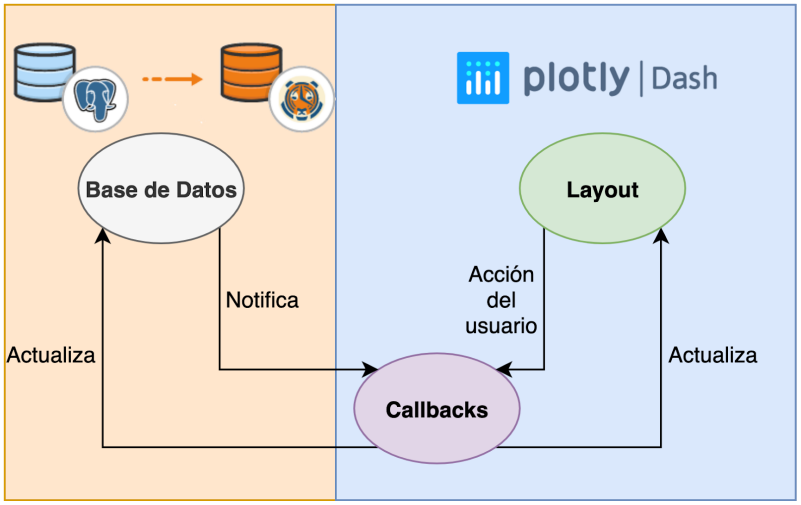

Figura 4.3: Interacción entre componentes de la implementación del subsistema

Como se muestra en la Figura 4.3:

- El modelo es equivalente a la “base de datos”, que se implementó en “PostgreSQL” con su complemento “TimescaleDB”.
- La vista corresponde al “layout” que se genera mediante “Plotly Dash”
- El presentador es equivalente a los “callbacks” que posee el framework “Plotly Dash”. Estos son funciones que permiten crear y modificar las vistas del dashboard, consultando información a la base de datos.

4.1.3. Alternativas de Visualización

Saber qué tipo de visualización usar depende fuertemente de los datos que vayamos a mostrar, ya que cada visualización representará la información de distinta manera. Lo más importante es que la visualización ayude a interpretar correctamente los datos, ya que así aportamos la información de manera clara al usuario. Es por esto, que se presenta un resumen de la utilización de cada tipo de visualización a utilizar.

- Lineplot (Figura 4.4), se usan para mostrar un mismo tipo de dato y su evolución, por lo que son muy útiles para identificar las tendencias.

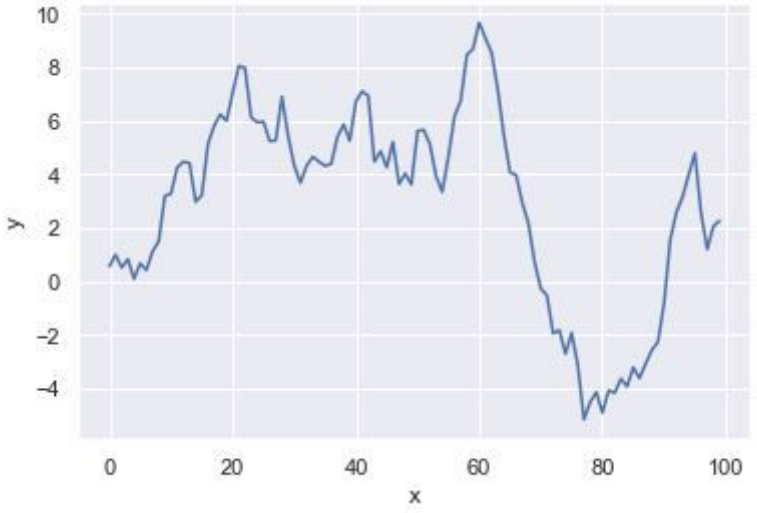

Figura 4.4: Ejemplo de Lineplot

- BoxPlot (Figura 4.5), representa lo que es conocido como las cinco figuras claves del resumen de un grupo de datos numéricos. Las cinco figuras en cuestión son: el mínimo, cuartil inferior o Q1 (25% de los datos), mediana o Q2 (el 50% de los datos), cuartil superior o Q3 (75% de los datos), valor máximo y datos fuera del máximo y mínimo llamados outliers.

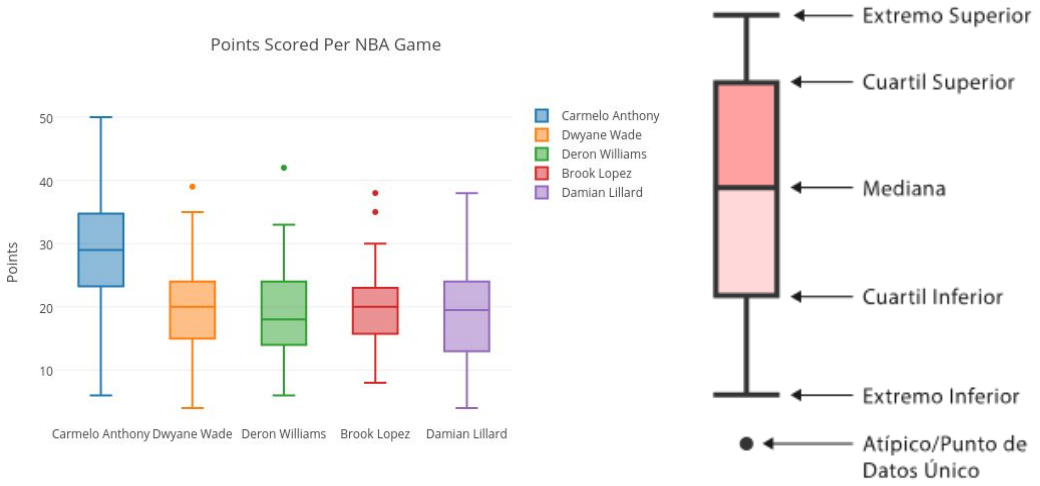

Figura 4.5: Ejemplo de gráfica BoxPlot
(Lado izquierdo, gráfica de ejemplo y lado derecho, descripción de partes de la caja)

- Histograma lineal o circular (Figura 4.6), es una representación gráfica de una variable en forma de barras, donde la superficie de cada barra es proporcional a la frecuencia de los valores representados.

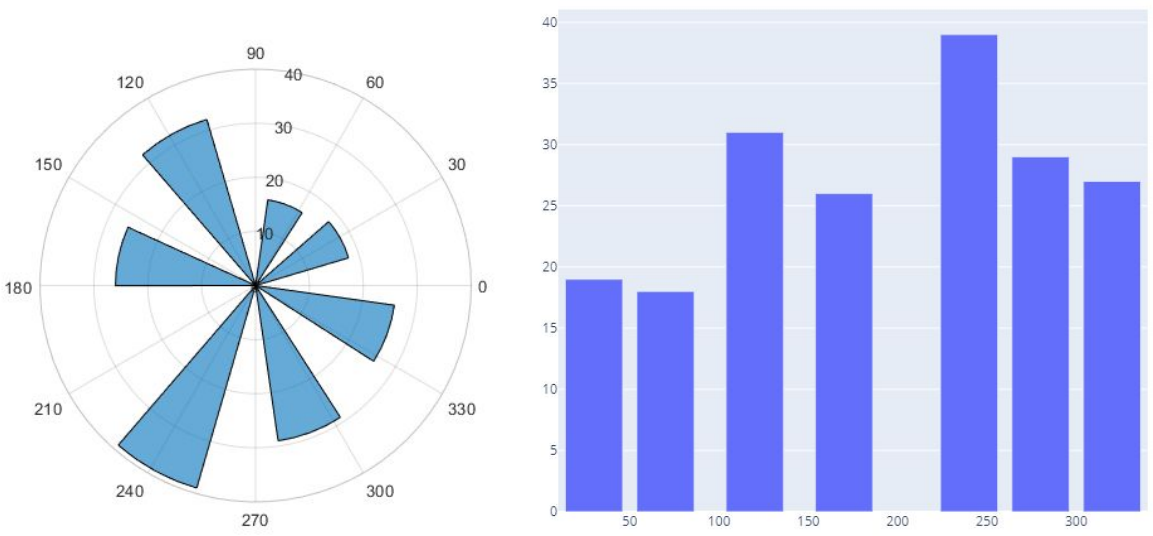

Figura 4.6: Ejemplo de Histograma Circular y Lineal
(Izquierda histograma circular y derecha histograma lineal)

- Gráfico OHLC (Open-high-low-close) (Figura 4.7), se usan para mostrar un mismo tipo de dato y su evolución, pero contiene más información que un gráfico Lineplot, ya que por cada periodo se visualizan datos de apertura (valor de entrada o primer dato medido), máximo, mínimo y cierre (valor de salida o último dato medido) (Figura 4.8).

Figura 4.7: Ejemplo de gráfico OHLC

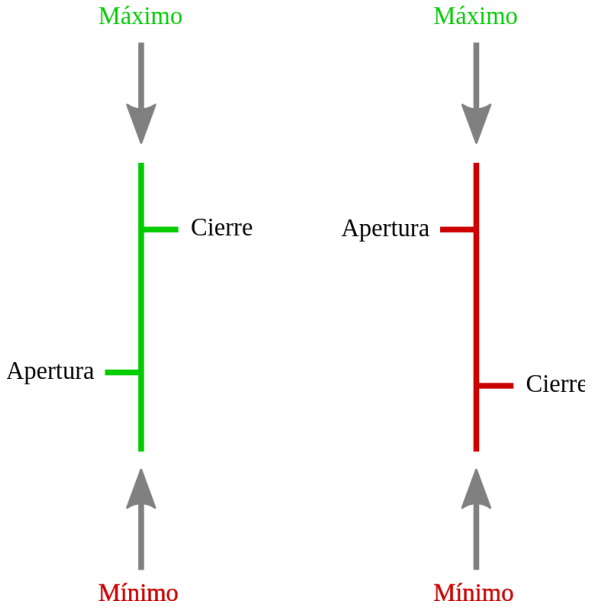

Figura 4.8: Variables en cada periodo en el gráfico OHLC con colores más utilizados (Color verde cuando el valor de apertura es menor que valor de cierre y color rojo cuando el valor de apertura es mayor al valor de cierre)

4.1.4. Composición del dashboard

El dashboard consta de 3 partes principales (Figura 4.9), las cuales son panel de datos de sensores y propiedades de gráficos, panel de indicadores resumen y paneles de gráficas (una gráfica principal y 2 secundarias). Además de tener un botón dedicado a los reportes, ubicado en la esquina superior derecha.

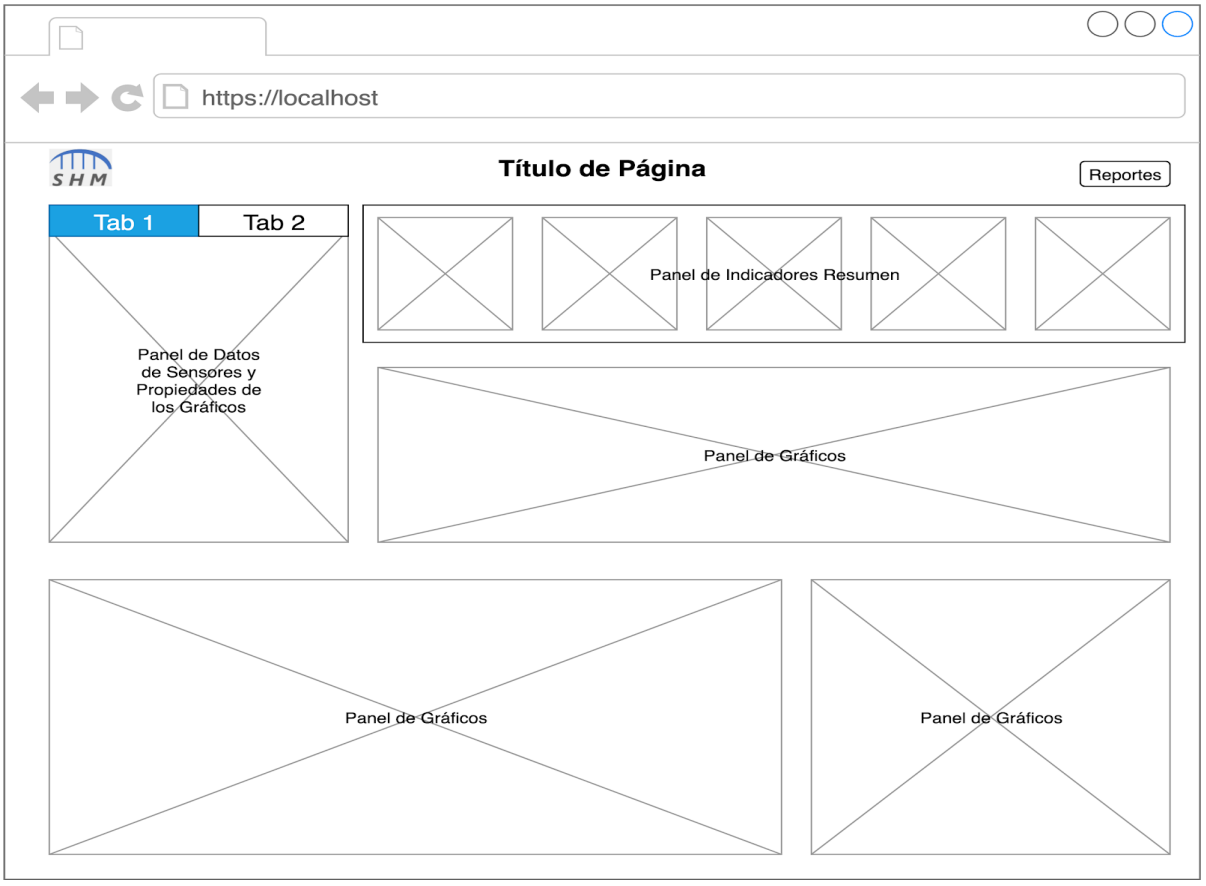

Figura 4.9: Estructura básica del dashboard

4.1.5. Composición del Reporte

El reporte consta de 3 partes principales (Figura 4.10), las cuales son: la sección de información general, la sección de datos resumen de los indicadores y la sección de gráficas (se divide en las 3 gráficas que se visualizan en el dashboard).

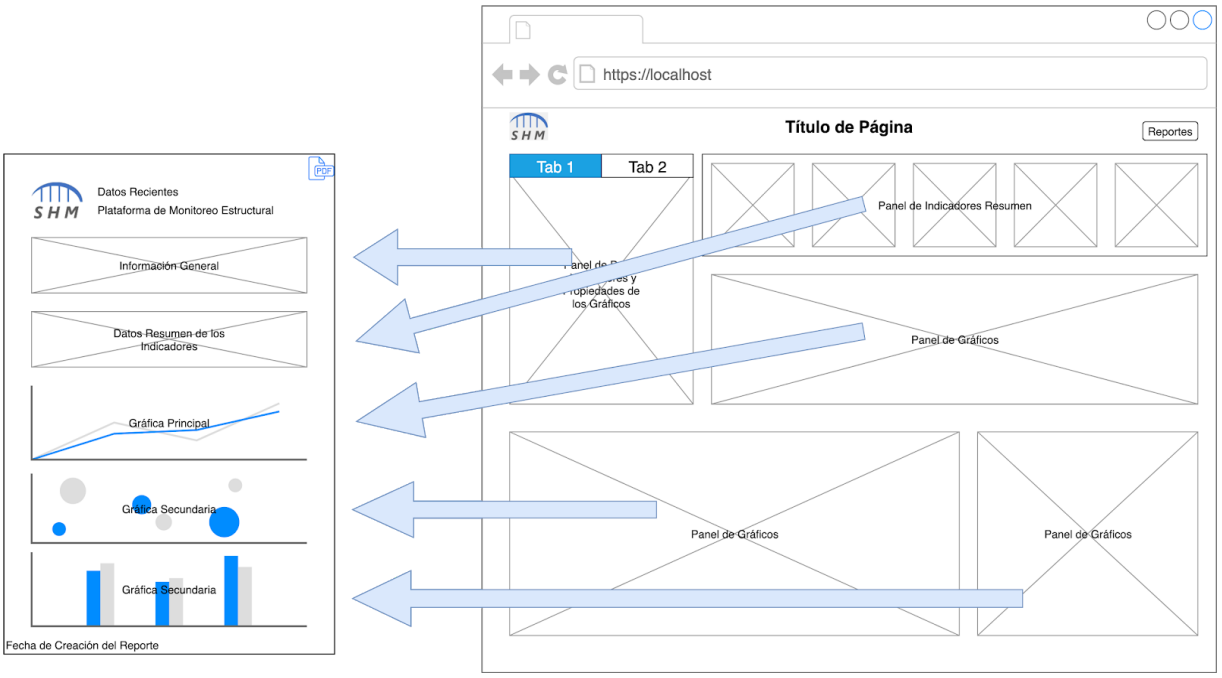

Figura 4.10: Correlación de partes del dashboard con el contenido del reporte

Como se muestra en la Figura 4.10, cada sección del dashboard está contenida en el reporte, por ende, contiene en su totalidad toda la información de los sensores que el usuario está visualizando. Esta modelo fue definido en conversaciones reiteradas con los expertos en el Monitoreo de Salud Estructural, en el marco del proyecto del cual es parte esta memoria.

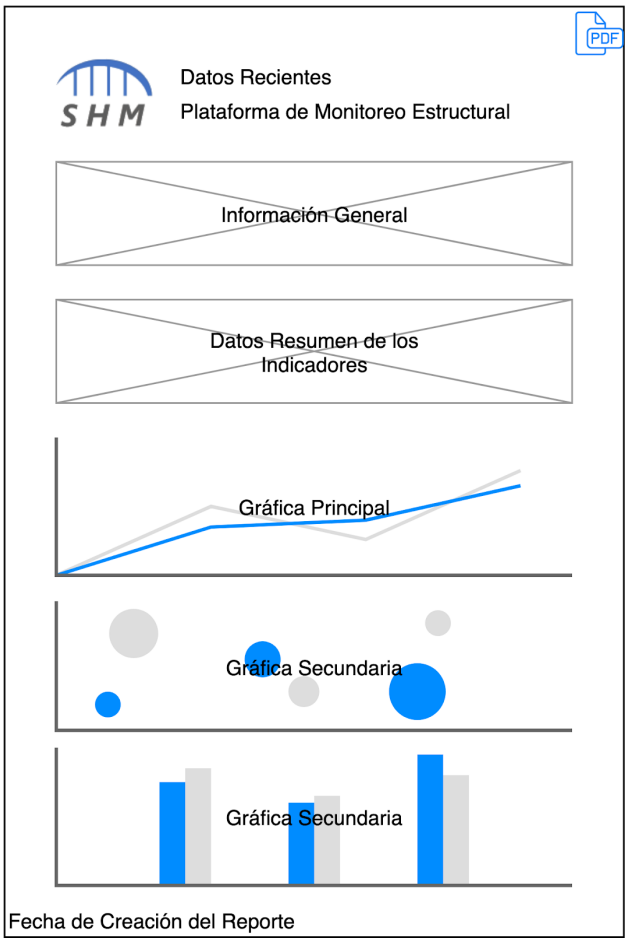

Figura 4.11: Estructura del Reporte

Entrando en más detalle en la estructura del reporte (Figura 4.11), en la sección de "Información general", se describe la información seleccionada por el usuario referente a los datos, ya sea el sensor a visualizar, la fecha y ventana de visualización. En la sección "Resumen de los indicadores", se muestra la información que está contenida en los indicadores, la cual está relacionada con la gráfica principal, además dependiendo del caso, también se mostrará la información referente a las líneas de control del gráfico principal, sólo si el usuario agregó esta propiedad a la visualización. En la sección de "Gráficas", se mostrará primero la gráfica principal y luego las dos gráficas secundarias que incluyen la interacción del usuario con estas, por ejemplo, si se le realiza zoom a cualquiera de las visualizaciones y se genera el reporte, esta interacción se verá reflejada en el reporte. Por último, se termina con la fecha y hora en la que se generó el reporte.

4.2. Implementación

El subsistema de visualización de datos recientes se implementó en el lenguaje de programación Python versión 3.6.9 y el framework Plotly Dash versión 1.9.1, que se basa en este lenguaje de programación, permitiendo una total integración entre ambas herramientas.

4.2.1. Archivos necesarios para el funcionamiento del subsistema

Para el funcionamiento del subsistema mediante el framework Plotly Dash, es necesario contar con la siguiente estructura de archivos (Figura 4.12), los cuales son necesarios para su correcta ejecución.

Figura 4.12: Estructura de archivos

Según la estructura que se muestra en la Figura 4.12, se tiene:

- En el archivo “app.py”, se encuentra todo el diseño del dashboard del subsistema, junto con llamadas de sistema “callbacks”, propias del framework, que permiten modificar partes del diseño del dashboard y generar las visualizaciones y reportes.
- En el archivo “dataframe.py”, se encuentran todas las funciones que trabajan con los datos extraídos de la base de datos, permitiendo el preprocesamiento de la información para la creación de los dataframe.
- En la carpeta “assets” (la cual puede ser opcional, ya que varios de sus parámetros se pueden manejar directamente en el archivo “app.py”, pero propicia un mejor ordenamiento), propia también del framework Plotly Dash.

- Contiene el archivo “styles.css” y “s1.css”, para personalizar ciertos parámetros que permiten tener control en la visualización, en cuanto a tipografía, tamaños de cuadros, tamaño de página, colores, disposición, etc.
- Contiene además todos los complementos de imágenes, en este caso el icono “favicon.ico”, que es la miniatura que aparece al abrir el subsistema y el logo “SHM-logo.bmp”, que es el que aparece en la esquina superior izquierda del dashboard.

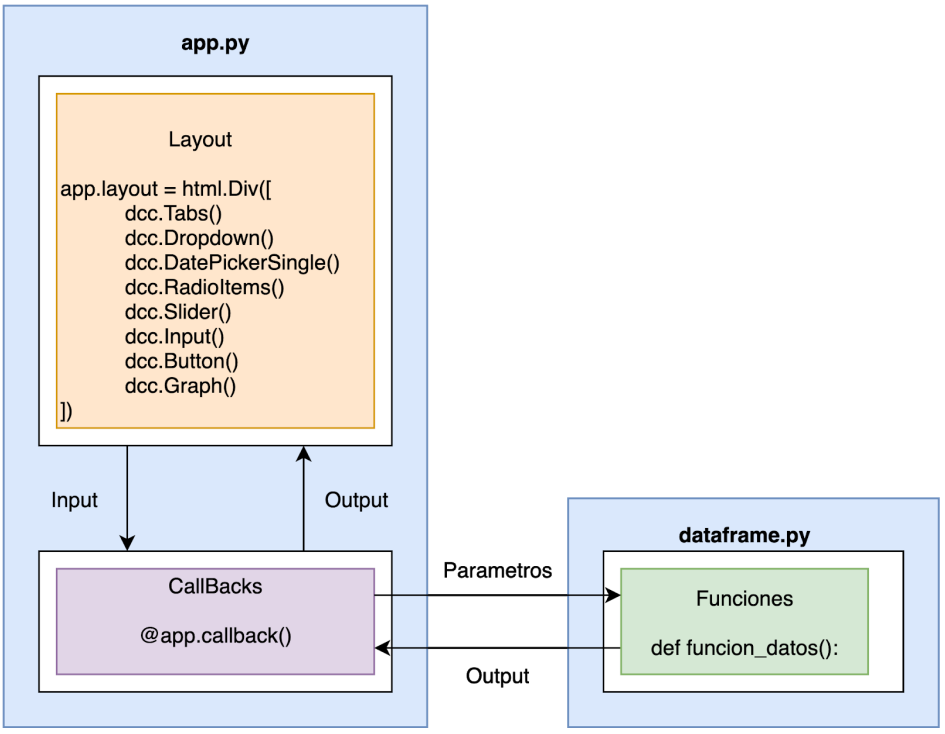

Figura 4.13: Esquema de interacción de archivos.
Cada archivo es representado por el cuadro de color celeste.

Más en profundidad, en la Figura 4.13, se muestra la interacción entre archivos, que permite el funcionamiento del framework, aquí se puede notar como en el archivo “app.py”, en donde el “layout”, que es la estructura que contiene todo el diseño del framework, interactúa con los “callbacks”, entregando los parámetros necesarios para que estos se los envíen a las funciones que están contenidas en el archivo “dataframe.py”. Aquí, estas funciones trabajan los datos obtenidos de la base de datos, para así retornar a los “callbacks”, la información solicitada y que estos puedan retornar al “layout”, sus nuevas propiedades o información según corresponda para que este se actualice. Este proceso “Input- Output” es un ciclo, que se ejecuta en todo momento, mientras el subsistema está en ejecución. Esto permite que cuando se interactúa con el dashboard, este vaya cambiando según lo que el usuario seleccione.

4.2.2. Desarrollo a través del framework

Para implementar el dashboard parte del subsistema mediante el framework Plotly Dash, se utilizó la librería “dash_html_components”, que permite utilizar una abstracción pura de Python de elementos de HTML, CSS y JavaScript. Así, en lugar de escribir sentencias HTML o usar plantillas HTML, el diseño de este se va componiendo usando estructuras de Python, lo que lo hace más sencillo.

```
import dash_html_components as html

html.Div([
    html.H1('Código en Plotly Dash'),
    html.Div([
        html.P('Dash convierte las clases de Python en HTML'),
        html.P('Esta conversión ocurre por detrás gracias al
                front-end JavaScript de Dash')
    ])
])
```

Código html escrito con sentencias Python

```
<div>
    <h1>Código equivalente en Html</h1>
    <div>
        <p>Dash convierte las clases de Python en HTML</p>
        <p>Esta conversión ocurre por detrás gracias al
           front-end JavaScript de Dash</p>
    </div>
</div>
```

Código escrito con sentencias Html

Figura 4.14: Ejemplo de un trozo de código escrito con sentencias Python y su equivalente escrito en HTML

También se utilizó la librería “dash_core_components” y “dash_bootstrap_ components” que contienen un conjunto de componentes de “alto nivel” como controles deslizantes, gráficos, menús desplegables, tablas, botones y más, permitiendo su inmediata utilización en conjunto con los componentes HTML antes mencionados.

El conjunto de estas librerías permite implementar y diseñar de forma “fácil” la estructura y componentes del dashboard, a su vez, las propiedades de estos elementos se definen en el archivo “style.css” y “s1.css”, que se encuentra en la carpeta “assets”, como se mostró en la Figura 4.12.

4.2.3. Datos utilizados

Para este trabajo se utilizó un dataset representativo perteneciente a un puente real (el puente “Tianjin Yonghe”), que se encuentra ubicado en Harbin, provincia de Heilongjiang, China [4]. Esto con el fin de poder crear visualizaciones adecuadas para la plataforma en cuestión.

Este dataset proporcionó datos para 2 tipos de sensores (acelerómetro y weather station), los cuales fueron recopilados por el sistema de SHM del puente “Tianjin Yonghe” durante enero del 2008, periodo en el que es posible identificar el paso de un estado saludable del puente hasta uno dañado del mismo.

En el set de datos correspondiente a los acelerómetros (Figura 4.15), hay un total de 15 columnas por archivo, en donde la primera columna presenta el tiempo de medición, mientras que las siguientes 14 columnas denotan el historial del tiempo de aceleración, recopilada utilizando 14 acelerómetros uniaxiales que se habían instalado en la cubierta del puente. La frecuencia de muestreo de la aceleración corresponde a 100 Hz (0,01 seg).

	1	2	3	4	5	6	7	8	9	10
1	2.008e+13	-0.019	0.097	-0.036	-0.038	-0.027	0.13	0.132	0.086	0.007
2	2.008e+13	-0.023	0.097	-0.034	-0.037	-0.017	0.145	0.131	0.094	0.023
3	2.008e+13	0.026	0.106	-0.044	-0.042	0.018	0.173	0.127	0.141	0.027
4	2.008e+13	0.079	0.115	-0.046	-0.079	0.052	0.16	0.099	0.16	0.033
5	2.008e+13	0.1	0.135	-0.059	-0.104	0.068	0.129	0.072	0.169	0.039

Figura 4.15: Extracto del set de datos de sensores de tipo acelerómetro del 1 al 9 (10 columnas, donde la columna 1 es el tiempo de medición y desde la columna 2 hasta la columna 10 corresponden a las mediciones de los acelerómetros 1 al 9)

En el set de datos de la weather station (Figura 4.16), hay un total de 5 columnas por archivo, en donde la primera columna presenta el tiempo de medición, mientras que las siguientes 4 columnas presentan la velocidad del viento (m/s), el ángulo del viento (grados), temperatura ambiente (°C) y humedad, respectivamente. La frecuencia de muestreo de las condiciones ambientales es de 20 Hz (0,05 seg).

```
Nombre del software de adquisición:     tider-entironment
Tiempo:         2008-4-1 0:38:3  a  2008-4-1 1:38:3
Frecuencia de muestreo: 10
Numero de canales:      4
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Número de canal:        0
Medida:Velocidad del viento
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Número de canal:        1
Medida: Dirección del viento
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Número de canal:        2
Medida: Temperatura
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Número de canal:        3
Medida:Humedad
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Tiempo          |0            |1            |2            ||3
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
20080401003803  3.311          41.775         7.123          -3.241
20080401003803  3.296          42.468         7.520          -3.210
20080401003803  3.296          42.468         7.520          -3.210
20080401003803  3.179          42.728         7.123          -3.210
20080401003803  3.179          42.728         7.123          -3.210
20080401003803  3.237          42.902         7.184          -3.241
20080401003803  3.237          42.902         7.184          -3.241
20080401003803  3.208          42.988         7.184          -3.149
```

Figura 4.16: Extracto del set de datos del conjunto de sensores weather station (5 columnas, donde la columna "tiempo" corresponde al timestamp (tiempo de medición) y desde la columna 0 hasta la columna 3 corresponden a las mediciones de velocidad del viento, dirección del viento, temperatura y humedad respectivamente)

Todos estos datos fueron almacenados en una base de datos (el esquema de la base de datos se encuentra en el Anexo 8.1) creada en PostgreSQL con su complemento TimescaleDB, de ahí son consultados por el presentador del dashboard para generar las visualizaciones correspondientes.

4.2.4. Dashboard

A continuación, se describe la implementación de los diferentes paneles (Figura 4.9) que componen el dashboard implementado (Figura 4.17).

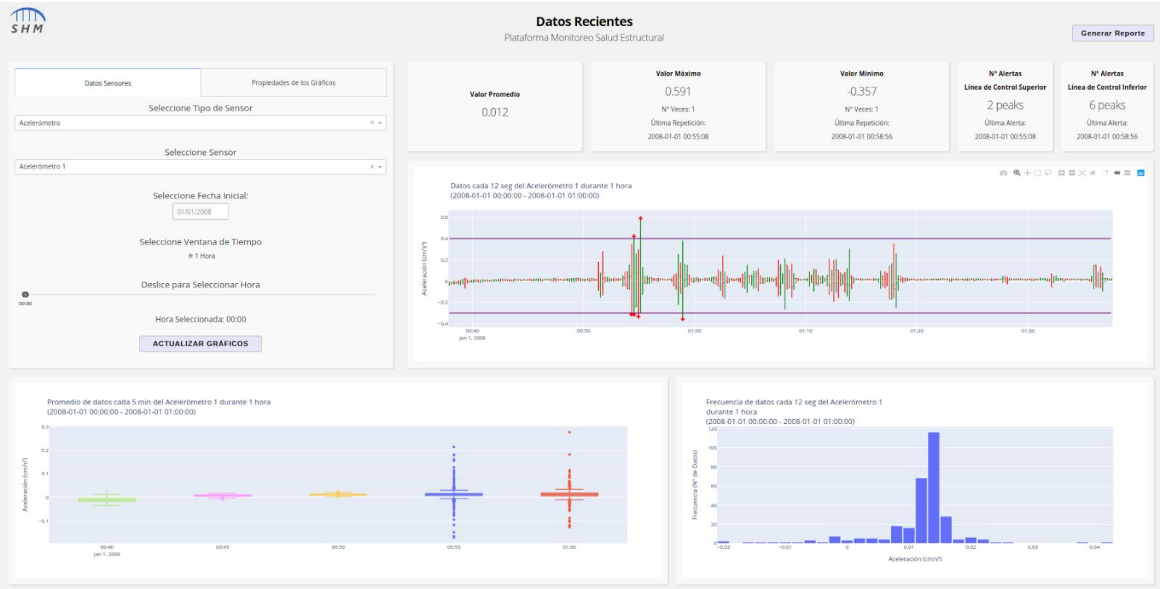

Figura 4.17: Vista completa del dashboard implementado

4.2.4.1. Panel de datos de sensores y propiedades de los gráficos

En este panel (Izquierda, Figura 4.18) se provee al usuario el control de los datos que se muestran en las visualizaciones, permitiéndole elegir el tipo de sensor y el sensor de ese tipo a visualizar (por defecto, se puede seleccionar solo 1 sensor a la vez, esto puede ser cambiado en la pestaña “Propiedades de los Gráficos”), además de elegir el día de la visualización, del cual, a partir de él se puede seleccionar la ventana de tiempo (1 hora, 1 día, 7 días y 14 días) que se desee visualizar. En el caso especial de elegir visualizar una ventana de tiempo de 1 hora, se desplegará una función extra, la cual, mediante un deslizador, se podrá seleccionar la hora a visualizar. Todas estas funciones están contenidas en la pestaña “Datos Sensores” de este panel.

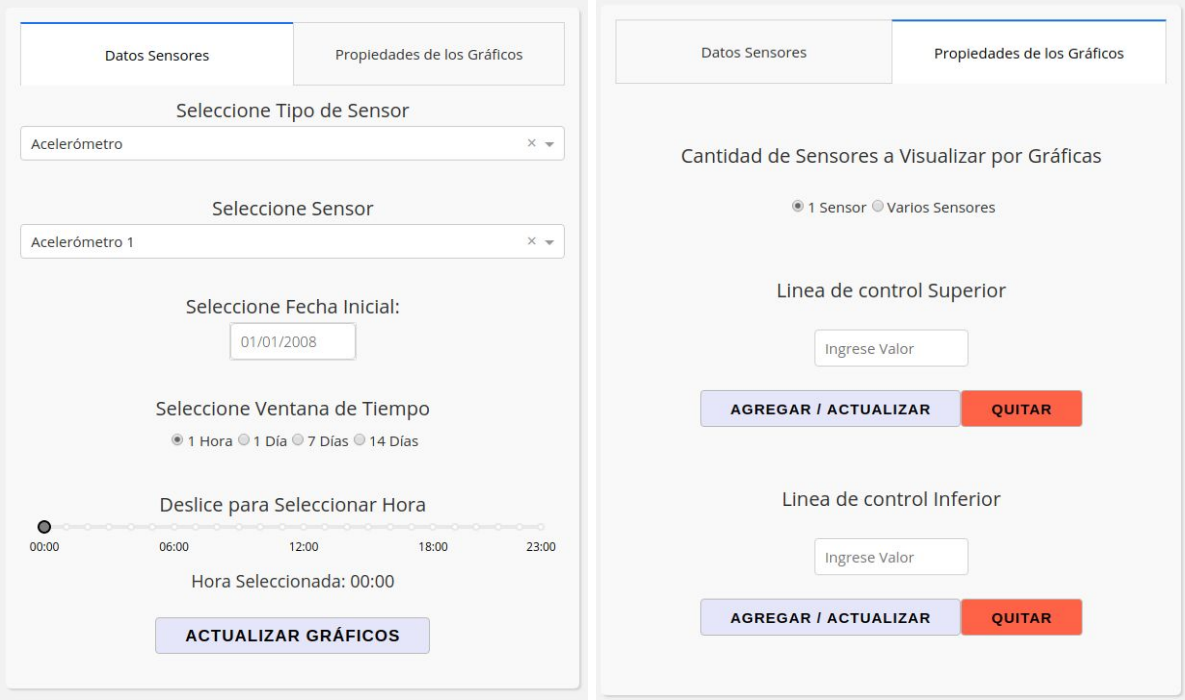

Figura 4.18: Panel de sensores, datos y propiedades de los gráficos. Lado izquierdo pestaña "Datos Sensores" y lado derecho pestaña "Propiedades de los Gráficos"

La segunda pestaña "Propiedades de los Gráficos" (Derecha, Figura 4.18), provee al usuario la posibilidad de mostrar en las visualizaciones más de un sensor a la vez y además la posibilidad de agregarle líneas de control a la visualización principal.

Al seleccionar la opción de visualizar varios sensores a la vez (siempre que el sensor lo tenga disponible), lo primero que cambia es la pestaña de "Datos Sensores" (Figura 4.19), en la cual, en el apartado de seleccionar sensor, se habilita el poder seleccionar más de 1 sensor a la vez, mientras que el resto de las opciones no cambian, los cambios surtirán efecto cuando se presione el botón "Actualizar Gráficos".

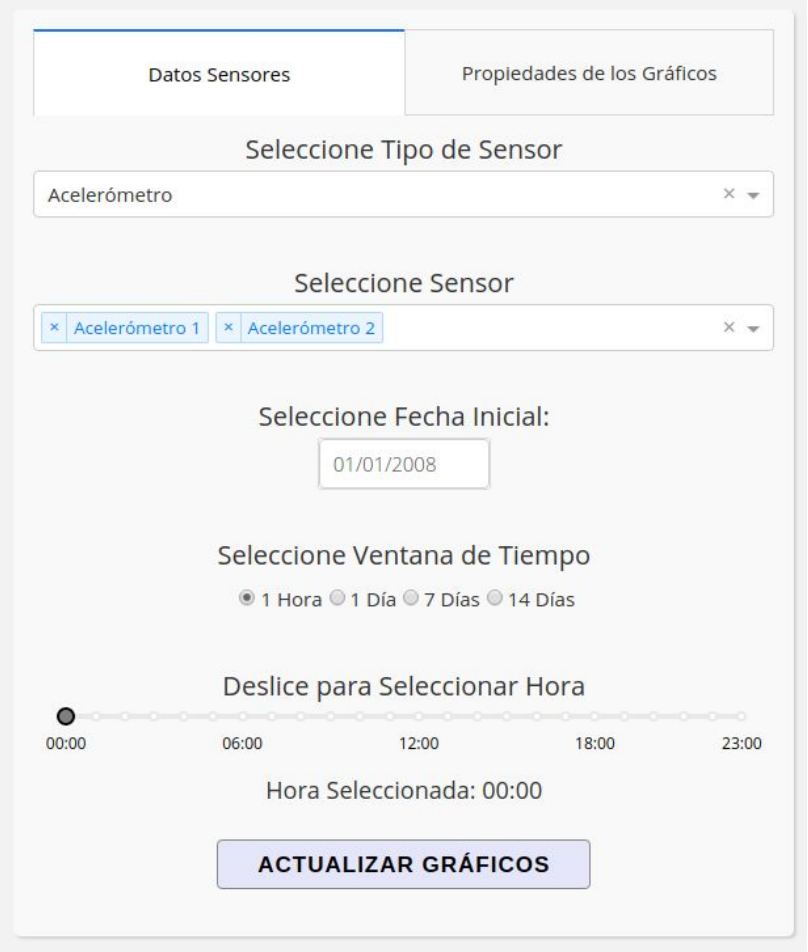

Figura 4.19: Nueva pestaña "Datos Sensores"

Cuando los cambios surtan efecto, tanto la visualización principal (Figura 4.20), como las secundarias (Figura 4.21), tendrán la capacidad de mostrar varios sensores a la vez.

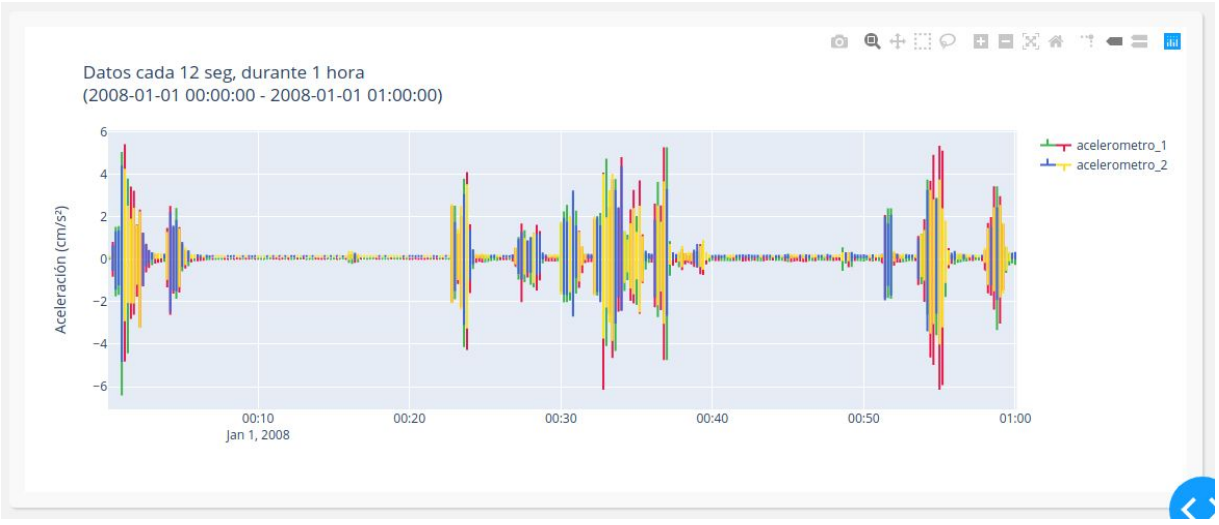

Figura 4.20: Ejemplo de gráfica principal con 2 sensores
(En el costado derecho se encuentra las combinaciones de colores por sensor)

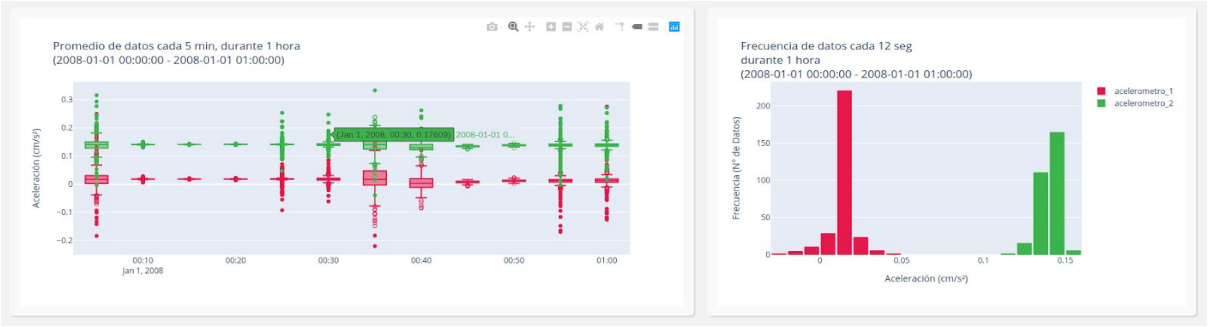

Figura 4.21: Ejemplo de gráficas secundarias con 2 sensores
(Los colores correspondientes a cada sensor se especifican en el costado derecho)

Por otro lado, las líneas de control que pueden ser tanto superior como inferior, además como propiedad extra se marcan en la visualización todos los puntos que superen dichas líneas denominados peaks, que le permiten al usuario tener mayor claridad de las variables que superan las líneas de control indicadas (Figura 4.22).

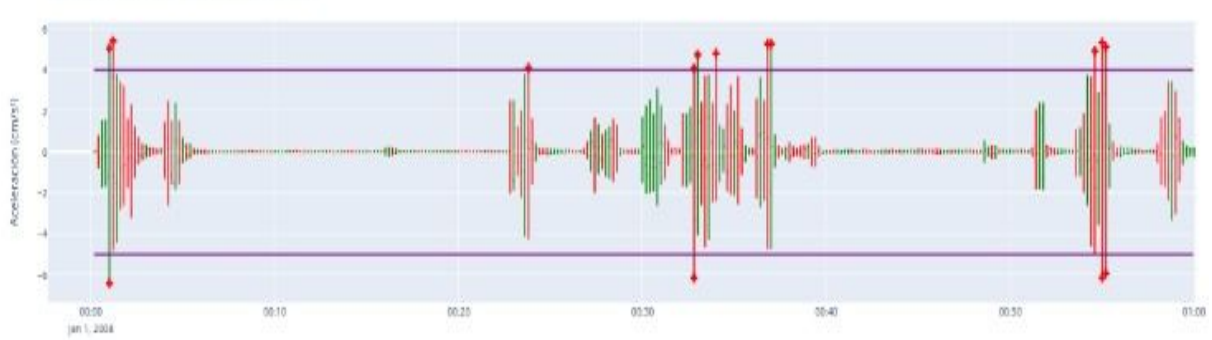

Figura 4.22: Ejemplo de visualización con líneas de control.
Se muestran líneas de control tanto superior como inferior (Color morado), además de los puntos marcados con cruces que superan estas líneas denominados peaks (Color Rojo).

4.2.4.2. Panel de indicadores resumen

En este panel se incluyen 3 indicadores fijos (solo se actualizan cuando el usuario cambia los datos a visualizar) y 2 que dependen de si se agrega una línea de control, ya sea inferior o superior, completando con datos estos paneles en esos casos.

Los paneles fijos (Figura 4.23) muestran al usuario información relevante calculada a partir de los 300 datos utilizados para generar la visualización principal. El primero de estos indicadores, ordenados de izquierda a derecha, indica el valor promedio de los datos, luego le sigue el indicador de valor máximo, el que muestra el máximo

valor dentro de los 300 datos analizados, además si este valor se repite, indica cuantas veces lo hace y por último indica la fecha y hora de la última repetición de este valor, la misma información se muestra para el siguiente indicador con datos de valor mínimo.

Figura 4.23: Panel Resumen sin datos en indicadores de Líneas de control (Primeros 3 paneles de izquierda a derecha son los paneles fijos)

Los indicadores que dependen de las líneas de control agregadas (Figura 4.24), se ubican a la derecha de los indicadores fijos, estos paneles se completan cuando el usuario agrega estas líneas de control a la visualización principal. Primero se ubica el indicador de alertas de la línea de control superior y a su derecha, el indicador de alertas de la línea de control inferior, ambos paneles indican primero el número de puntos que superan el valor de su correspondiente línea de control (peaks) y luego se indica la fecha y hora del último peak que superó esta línea.

Figura 4.24: Panel Resumen con datos en indicadores de Líneas de control (Últimos 2 paneles de derecha a izquierda)

4.2.4.3. Panel de Gráficos

En esta sección del dashboard se encuentran actualmente 3 paneles dedicados a las visualizaciones, estos se dividen en el panel de visualización principal y paneles secundarios.

El diseño del dashboard y su implementación, permiten que se puedan agregar paneles o cambiar las visualizaciones mediante código fácilmente (comentado en el código fuente del dashboard).

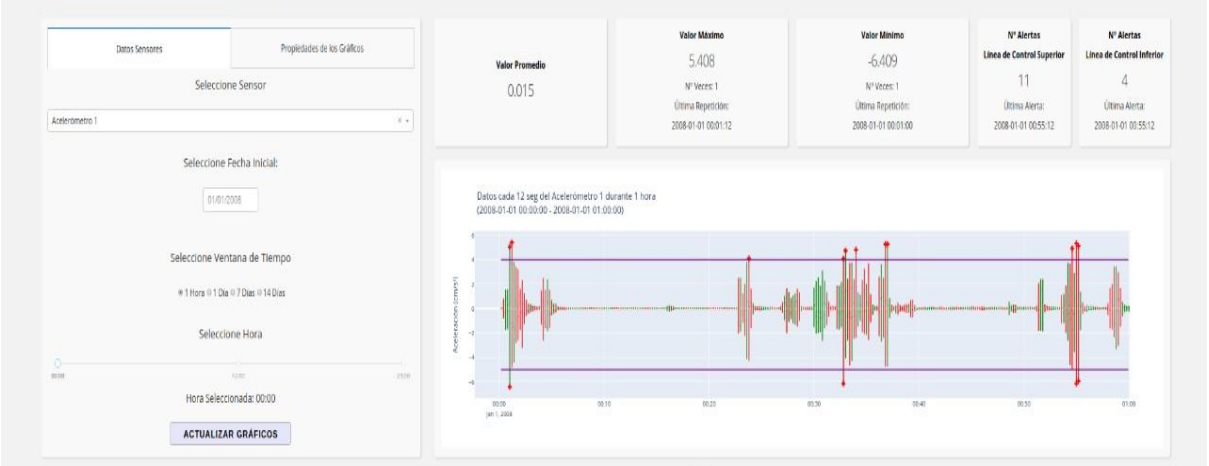

Figura 4.25: Panel de visualización principal
Ubicado al costado derecho del "panel de datos de sensores y propiedades de los gráficos" y debajo del "panel de indicadores resumen"

El panel de visualización principal (Figura 4.25), provee al usuario una visualización que le muestra una mayor cantidad de información relevante proveniente de las mediciones del sensor, para así poder, con la ayuda de las líneas de control que se le pueden agregar, llegar a detectar alguna anomalía, con el fin de inferir el estado actual de la estructura en el rango de tiempo que se está visualizando. En la Figura 4.25, se muestra de ejemplo el gráfico OHLC (Open-high-low-close) para los sensores de tipo acelerómetro.

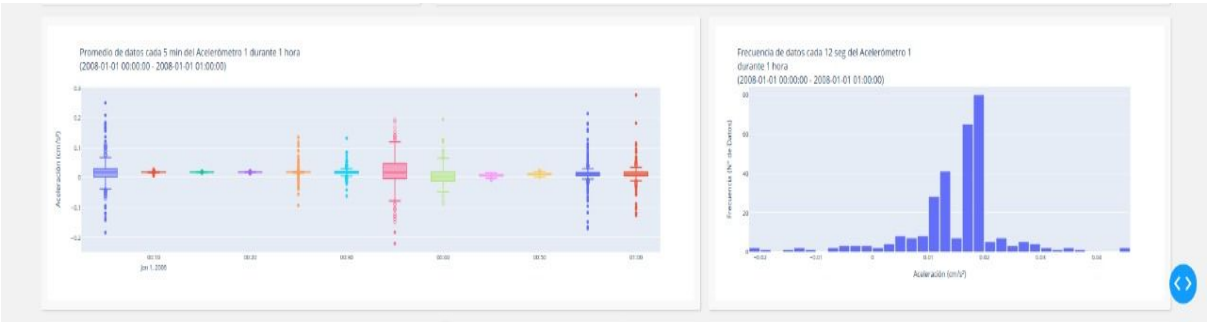

Figura 4.26: Paneles secundarios
Visualizaciones secundarias correspondientes a los sensores de tipo acelerómetro

Los paneles secundarios (Figura 4.26), muestran visualizaciones que le proveen al usuario información extra, como por ejemplo la frecuencia de los datos dentro del rango de tiempo seleccionado (Visualización del lado derecho de la Figura 4.26). Estas visualizaciones suelen ser de un tamaño un poco menor a la visualización principal en la distribución de paneles del dashboard, para generar una diferencia entre ellas.

4.2.5. Reportes

Esta sección del dashboard, sirve para generar reportes imprimibles (formato PDF) de los datos que se visualizan en la plataforma. Para acceder a este apartado solo basta presionar el botón llamado “Generar Reporte”, que se encuentra en la esquina superior derecha de la plataforma (Figura 4.17), el cual está en todo momento disponible para su uso. Al presionar este botón, se genera de inmediato un reporte en formato pdf, que contiene la información que en ese momento el usuario está visualizando.

A continuación, se presentan ejemplos de reportes generados por la plataforma en diferentes escenarios.

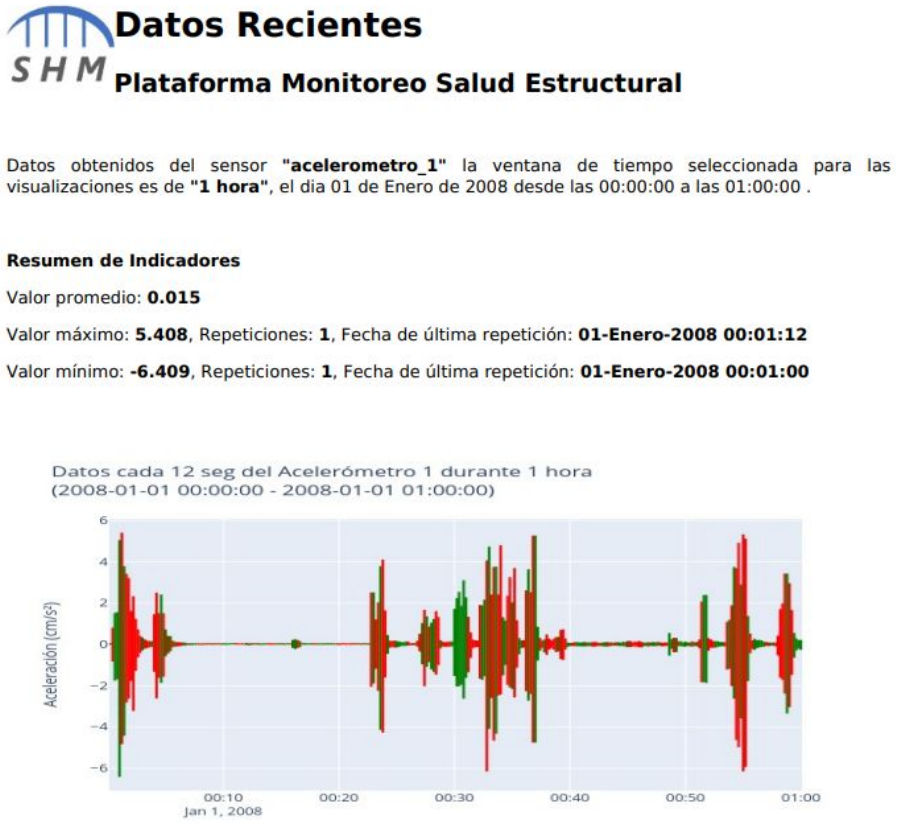

Figura 4.27: Ejemplo de Reporte generado sin líneas de control y un sólo sensor

Datos Recientes

Plataforma Monitoreo Salud Estructural

Datos obtenidos del sensor **"acelerometro_1"** la ventana de tiempo seleccionada para las visualizaciones es de **"1 hora"**, el dia 01 de Enero de 2008 desde las 00:00:00 a las 01:00:00 .

Resumen de Indicadores

Valor promedio: **0.015**

Valor máximo: **5.408**, Repeticiones: **1**, Fecha de última repetición: **01-Enero-2008 00:01:12**

Valor mínimo: **-6.409**, Repeticiones: **1**, Fecha de última repetición: **01-Enero-2008 00:01:00**

Líneas de control

Valor de línea de control superior: **2**

Peaks superiores: **49 peaks**

Fecha último peak superior detectado: **01-Enero-2008 00:59:00**

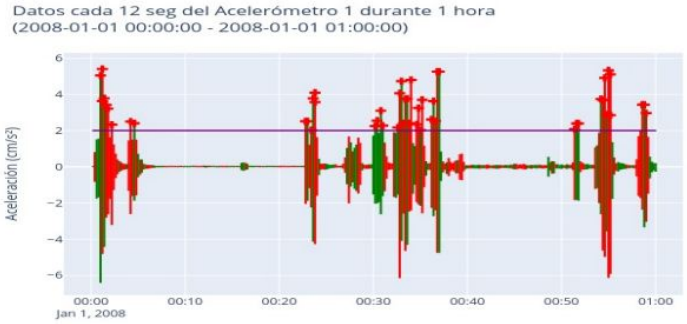

Figura 4.28: Ejemplo de Reporte generado con una línea de control y un sólo sensor

Datos Recientes

Plataforma Monitoreo Salud Estructural

Datos obtenidos de los sensores **"acelerometro_1,acelerometro_2,acelerometro_3,"**, la ventana de tiempo seleccionada para las visualizaciones es de **"1 hora"**, el dia 01 de Enero de 2008 desde las 00:00:00 a las 01:00:00 .

Resumen de Indicadores

(Datos del último sensor seleccionado)

Valor promedio: **0.027**

Valor máximo: **3.336**, Repeticiones: **1**, Fecha de última repetición: **01-Enero-2008 00:01:00**

Valor mínimo: **-3.317**, Repeticiones: **1**, Fecha de última repetición: **01-Enero-2008 00:01:00**

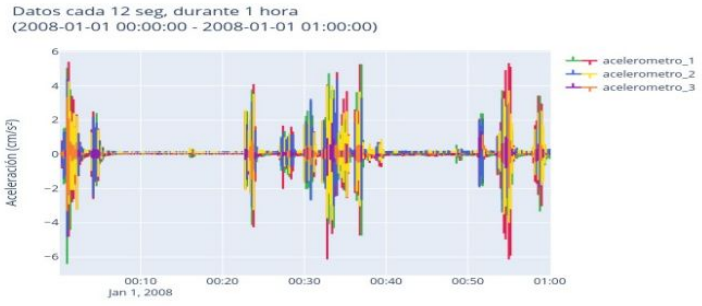

Figura 4.29: Ejemplo de Reporte generado sin líneas de control y con 3 sensores seleccionados

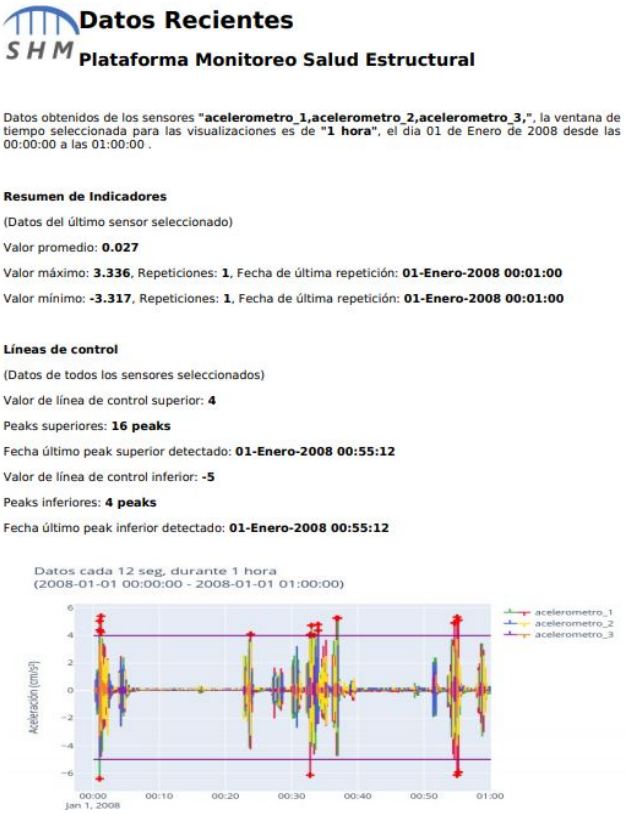

Figura 4.30: Ejemplo de Reporte generado con ambas líneas de control y con 3 sensores seleccionados

4.2.6. Consideraciones de Big-Data

Todos los dataframe generados a partir de los datos almacenados que se utilizan para las visualizaciones contienen 300 datos. Esta cantidad fue definida en conversaciones reiteradas con los expertos en el Monitoreo de Salud Estructural, en el marco del proyecto del cual es parte esta memoria. Los dataframe se generan dividiendo en 300 intervalos los tiempos de medición a visualizar (1 hora, 1 día, 7 días y 14 días), a estos intervalos se le calcula el promedio y este dato es el que se guarda y por ende se generan 300 datos de esta manera.

Esta reducción de datos sirve para sintetizar el gran volumen de datos que se tiene almacenados, mejorar su procesamiento y así poder usar correctamente el subsistema. Este se vuelve poco usable (sufre constantes interrupciones de ejecución) cuando se superan los 2 millones de datos en la visualización, lo que también implica que se vuelva difícil de entender, dada la cantidad de información desplegada. Esto, ya que al comienzo se pensaba usar directamente la totalidad de los puntos disponibles, por ende, se crearon visualizaciones, en donde al superar los 2 millones de datos (por ejemplo, una visualización de 1 hora (360.000 datos por sensor), para 6 sensores a la vez), estas sufrían constantes interrupciones de

ejecución, no permitiendo su uso. Por ejemplo, para un día común de visualización, medido por un sensor acelerómetro, con una frecuencia de 100Hz (100 datos por segundo), se tienen más de 8 millones de datos en un solo día, los cuales no podrían ser usados directamente sin procesar, por lo antes mencionado. Además, el subsistema debe poder mostrar visualizaciones en un rango máximo de 14 días, lo que implicaría, si se usan todos los datos obtenidos por día de medición a 100 Hz, se generará una visualización de más de 120 millones de datos.

Las visualizaciones creadas, son de tipo WebGL (Web Graphics Library), la cual es una especificación estándar para definir una API (interfaz de programación de aplicaciones), para la renderización de gráficos 3D dentro de cualquier navegador web, permitiendo la aceleración por hardware de la GPU (unidad de procesamiento gráfico). Esto podría dar pie, para aumentar la cantidad de datos por visualización, teniendo en cuenta realizar una mejora en la GPU del equipo en donde se realicen las pruebas.

4.2.7. Comentarios de la implementación

Las gráficas expuestas en la sección 4.1.3 “Alternativas de Visualización”, se probaron con los datos que actualmente se tienen disponible de los sensores de tipo acelerómetro y weather station (sección 4.2.3 “Datos utilizados”).

Lo usual para visualizar los datos de los sensores de tipo acelerómetro, es la utilización de la gráfica Lineplot, la que permite mostrar la evolución de una variable, es por esto, que este tipo de visualización es la más utilizada para mostrar datos obtenidos de sensores. Si bien Lineplot es la más utilizada, la cantidad de datos a visualizar es muy alta (mediciones de 100 Hz) y por ende, es necesario hallar la manera de mostrar la mayor cantidad de información en cada visualización.

La utilización de 300 datos por visualización (sección 4.2.6 “Consideraciones de Big-Data”), conlleva una pérdida de información, la cual debe minimizarse lo más posible, es por esto que la utilización de Lineplot (Figura 4.4) deja de ser la más adecuada, ya que por ejemplo, tomando 1 hora de medición de un acelerómetro se tienen 360.000 datos, los que son reducidos mediante promedio de intervalos a 300, entonces al generar la visualización por cada dato ploteado en el eje x hay un solo dato al igual que en el eje y (generando 300 puntos a visualizar), lo que genera una gran pérdida de información para el usuario. Para resolver esto, entra en juego otro tipo de visualización, en este caso el gráfico OHLC (Figura 4.7), el cual al igual que el Lineplot, permite mostrar la evolución de un tipo de dato en el tiempo, pero por cada dato muestra 4 variables (Figura 4.8) y no una como el Lineplot, es por esto que tomando el mismo ejemplo anterior, igual se generan 300 puntos a visualizar,

pero por cada uno de ellos se muestra más información (4 variables), la cual, corresponde a el valor inicial, valor final, valor máximo y valor mínimo del dato en ese punto específico, así con la misma cantidad de puntos es posible entregarle al usuario mayor información, lo que para el propósito de la plataforma es crucial y aumenta las posibilidades de análisis a partir de las visualizaciones.

Por otro lado, la fuerte necesidad de entregarle al usuario la mayor información posible a partir de los datos obtenidos de sensores, crea la necesidad de tener más visualizaciones que permitan reforzar esta idea. Es por esto que como alternativas se proponen incluir junto con la gráfica OHLC, las gráficas Boxplot e histograma.

La gráfica Boxplot, se eligió porque proporciona una visión general de la distribución de los datos (por ejemplo, si la mediana no está en el centro del box, la distribución no es simétrica), también son útiles para ver la presencia de valores atípicos (outliers) y además pertenece a las herramientas de la estadística descriptiva, que permiten ver cómo es la dispersión de los puntos con la mediana, los percentiles 25, 75, y los valores máximos y mínimos. Esto, podría ayudar al usuario a tener claridad de datos atípicos en pequeños rangos de tiempos. Para la confección de este tipo de gráfica, se subdividieron los rangos de tiempo a visualizar (1 hora, 1 día , 7 días y 14 días) en 12 rangos, para los de 1 hora, 1 día y 7 días, mientras que para el de 14 días se dividió en 14 rangos, esto con el fin de tener más información a mostrar en la gráfica, ya que cada uno de estos nuevos rangos tiene su propio dataframe y por ende en cada uno de ellos hay 300 datos, pudiendo así calcular de mejor manera los datos estadísticos que proporciona esta visualización.

La gráfica histograma, se eligió también porque es útil cuando existe una gran cantidad de datos, ya que indica la frecuencia de un hecho, permitiendo interpretar de mejor forma las variaciones de los datos, complementando aún más la información expuesta por las gráficas OHLC y Boxplot. Para crear este gráfico, se utiliza el mismo dataset de promedios de datos que la gráfica OHLC.

A pesar de solo tener disponibles datos para 2 tipos de sensores y según lo expuesto anteriormente, podemos decir que la gráfica OHLC, Boxplot e histograma sirven como visualizaciones comunes para todos los tipos de sensores (sección 3.1 “Requerimiento de las visualizaciones”), (En el conjunto de sensores “weather station”, puede servir para los datos de temperatura y humedad) ya que en todos ellos se deben reflejar la evolución de sus mediciones. A su vez se descarta el uso de la visualización Lineplot, por los argumentos antes mencionados.

En el caso de los datos de dirección del viento (medido en grados) y velocidad del viento (medido en m/s), que se obtienen del conjunto de sensores weather station, se utilizó la gráfica de tipo histograma circular, la cual permite, dados este tipo de

dato representar de mejor manera la frecuencia de estos datos, ya que permite modelar de forma bidimensional el sistema de coordenadas polares, representando gráficamente la dirección del viento según su velocidad, la que se representa en el eje radial, facilitando su comprensión (parte izquierda de la figura 4.6). Siendo en este caso una gráfica específica para un tipo de sensor.

Por último, se intentó implementar un gráfico de tipo Lineplot para mostrar la transformada de Fourier de los datos de sensores de tipo acelerómetros. En estos datos es posible distinguir dos frecuencias: una es la frecuencia de muestreo, que es la que se utiliza para la captura de los datos (100Hz en los datos disponibles) y la otra es una frecuencia de análisis o recorrido de la señal, esta frecuencia es la que se extrae para generar la FFT (Transformada Rápida de Fourier), permitiendo detectar ciertas anormalidades que no se detectan fácilmente en los datos al solo mostrarlos. Así, aplicar la FFT permitiría detectar y aislar una frecuencia de los datos que está provocando alguna anomalía. El problema de esta visualización radica en la gran cantidad de datos que necesita, para crearla es necesario tener completamente, para cada ventana de tiempo, todos los datos que se tengan disponibles, volviendo al problema planteado al inicio, que no es posible visualizar en este caso los 100 Hz o más si es que estuvieran disponibles, es por esto que esta visualización se descartó y no se incluyó en el dashboard como alternativa de visualización.

5. Experimentos y Resultados

En esta sección, se detallan las pruebas de rendimiento realizadas, que sirven para minimizar los tiempos de carga de las visualizaciones.

Todas las pruebas fueron realizadas en un computador portátil con un procesador Intel© Core™ i3-8130 U que funciona a 2.20GHz, 12 gigas de memoria ram DDR4 a 2600Mhz, una GPU Nvidia GeForce MX 130 de 2 gigas de Vram y el sistema operativo utilizado fue Linux Mint 19.3 con el Kernel 5.3.0-40-generic.

5.1. Pruebas de Rendimiento

La implementación del dashboard debe considerar el tiempo de carga de las visualizaciones que se muestran. Estos tiempos están fuertemente relacionados con la cantidad de datos que se deben obtener de la base de datos, para así crear los dataframe a utilizar, pero no así, del tipo de visualización que se crea.

La obtención y procesamiento de estos datos lleva tiempo, el cual debe ser siempre el menor posible, ya que para el usuario es tedioso esperar para poder ver las visualizaciones del dashboard. Es por esto que se probaron 2 métodos para generar los dataframe tratando de minimizar los tiempos para generar las visualizaciones.

Los datos utilizados para esta prueba son los pertenecientes a los sensores de tipo acelerómetro, cuyos datos tienen una frecuencia de 100Hz y fueron obtenidos de mediciones reales en el puente “Tianjin Yonghe”, como se mencionó con más detalle en el “Capítulo 4”, sección 4.2.3 “Datos utilizados”.

5.1.1. Interacción directa con la base de datos

En primera instancia, para generar los dataframe, a la base de datos no se le consultó directamente por el dato almacenado, sino que se le pide que retorne datos promedio dados los rangos de tiempo elegidos (1 hora, 1 día, 7 días y 14 días), esto quiere decir, por ejemplo, para el “Acelerómetro 1”, se seleccionó como rango de tiempo “1 hora”, este rango se divide en 300 partes, por lo que se calculan promedios de datos cada 12 segundos, para así poder obtener los 300 datos pertenecientes al rango de “1 hora”, este proceso realiza 300 consultas a la base de datos, en donde cada consulta obtiene en este caso 1200 datos, los cuales se promedian y se obtiene 1 solo dato, así estas 300 consultas en total tienen un tiempo promedio de 0,7 segundos.

El detalle de los tiempos por consulta de todos los rangos se puede ver en la siguiente tabla (Tabla 5.1).

Rango de tiempo para generar promedio	Datos obtenidos por consulta a la base de datos	Datos totales consultados (300 consultas)	Cantidad de datos finales	Tiempo total de consultas, por rangos a la base de datos
12 segundos	1.200	360.000	300	0,7 segundos
4 minutos y 48 segundos	28.800	8.640.000	300	7,5 segundos
33 minutos y 36 segundos	201.600	60.480.000	300	73,2 segundos
1 hora, 7 minutos y 12 segundos	403.200	120.960.000	300	158,4 segundos

Tabla 5.1: Cantidad de datos obtenidos por rangos de tiempo y su tiempo de consulta
(Tiempos promediados obtenidos luego de realizar 10 mediciones por cada rango)

Luego, se realizó el mismo procedimiento para calcular los tiempos, en donde se cambió la forma de obtener el promedio de los datos, pasando de obtener directamente el promedio al realizar la consulta, a retornar todos los datos almacenados, para luego mediante un procesamiento realizado mediante el lenguaje Python calcular el promedio de esos datos, para así poder generar los 300 datos de los dataframe.

Los nuevos tiempos son presentados en la Tabla 5.2, que muestra el tiempo total para generar un dataframe de 300 datos, este tiempo total se calcula de la suma del tiempo de la consulta a la base de datos, más el tiempo que se demora mediante un programa escrito en lenguaje Python en calcular el promedio de estos datos.

Rango de tiempo para generar promedio	Datos obtenidos por consulta a la base de datos	Datos totales consultados (300 consultas)	Tiempo de consulta la base de datos	Tiempo de cálculo del promedio	Tiempo total
12 segundos	1.200	360.000	0,5 segundos	0,1 segundos	0,6 segundos
4 minutos y 48 segundos	28.800	8.640.000	12,6 segundos	0,5 segundos	13,1 segundos
33 minutos y 36 segundos	201.600	60.480.000	77,6 segundos	3,1 segundos	80,8 segundos
1 hora, 7 minutos y 12 segundos	403.200	120.960.000	152,1 segundos	6,1 segundos	158,2 segundos

Tabla 5.2: Cantidad de datos obtenidos por rangos de tiempo, tiempo de consulta a base de datos y tiempo de cálculo del promedio
(Tiempos promediados obtenidos luego de realizar 10 mediciones por cada rango)

En el Gráfico 5.1, que se presenta a continuación, se compara ambos tiempos de generación de dataframe.

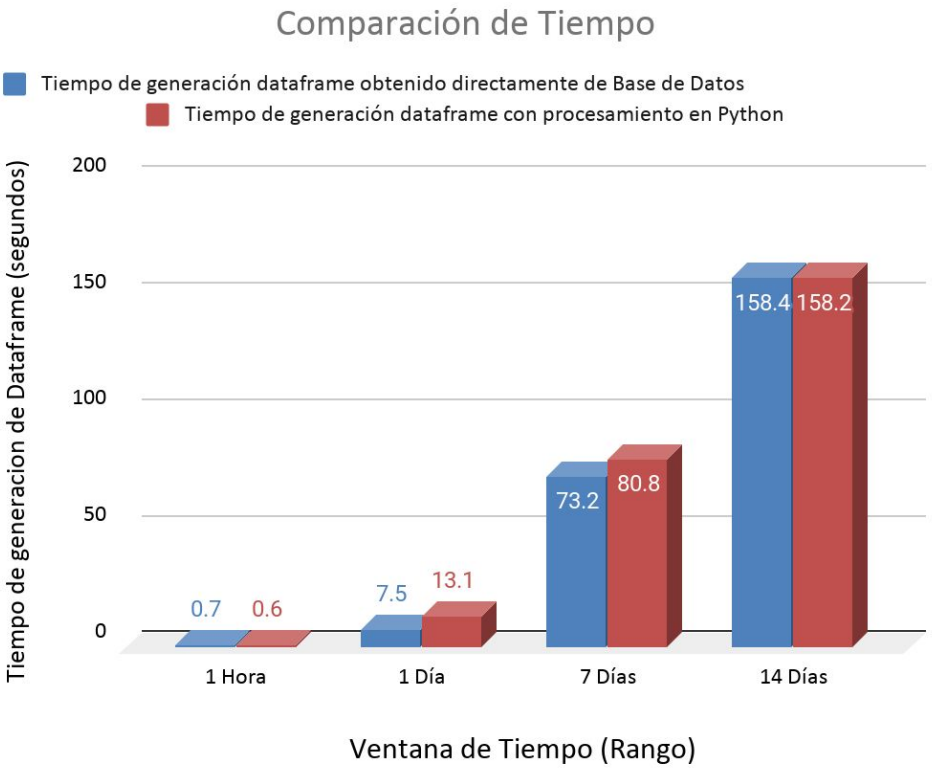

Gráfico 5.1: Comparación de tiempo de generación de dataframe

En el Gráfico 5.1, se puede apreciar que la variación de tiempo entre ambas alternativas es pequeña, solo notoriamente en los rangos de 1 día y 7 días, con una diferencia de 5,6 segundos y 7,6 segundos respectivamente, pero como se mencionó al comienzo de esta sección, se pretende generar la visualización en el menor tiempo posible, por lo que la mejor opción entre ambas es generar el dataframe a partir del dato calculado directamente desde la base de datos (línea de color azul).

5.1.2. Pre cálculo de dataframe

Otra alternativa es calcular todos los dataframe previamente, para luego cuando el usuario lo requiera solo utilizarlos evitando el tiempo que le toma al sistema generar el dataframe.

Los dataframe se generan por cada rango de tiempo y para cada sensor, además que en algunos casos especiales se necesiten más de un dataframe por visualización o un dataframe especial con más datos.

Tomando en consideración solo los sensores de tipo Acelerómetro (14 sensores), sólo 14 días (con las 24 horas con datos a 100Hz), disponibles de datos para estos sensores en la base de datos y generar los dataframe con los promedios calculados desde la base de datos (menor tiempo de generación de dataframe), se obtuvieron los resultados de la Tabla 5.3.

Rango de Tiempo	Cantidad de dataframes	Tiempo estimado de generación del dataframe
1 hora	336	3 minutos y 54 segundos
1 día	196	24 minutos y 30 segundos
7 días	98	1 hora, 59 minutos y 33 segundos
14 días	14	36 minutos y 57 segundos

Tabla 5.3: Cantidad de dataframe por rango de tiempo y tiempo estimado para 14 sensores

Como se muestra en la tabla 5.3, para 14 sensores en total se generan 644 dataframe que sirven para poder abordar todos los requerimientos del usuario en cuanto a estos sensores, por otro lado, en cuanto al tiempo de generación, existen 2 formas de verlo, la primera es secuencialmente, en donde se suman todos los tiempos, dando un total estimado de 3 horas, 4 minutos y 54 segundos para generar

estos dataframes, sin embargo existe la posibilidad de que se puedan generar en simultáneo, proponiendo tener 4 hilos de ejecución (uno por cada rango de tiempo), en donde tomando el tiempo más largo de generación de dataframe, estos tomarían aproximadamente 2 horas en generarse, reduciendo significativamente el tiempo de generación, esto se podría lograr si la máquina en la cual se ejecuta esta acción, es considerablemente más potente que en la que se han realizado las pruebas (las especificaciones técnicas se mencionaron al comienzo de esta sección).

Tomando el otro escenario en el que se generan los dataframe secuencialmente (el escenario más común), se deben tener más de 3 horas en las que solo el sistema se dedique a crear los dataframes necesarios para su correcto uso y funcionamiento, siendo un tiempo muy alto como para considerarlo una opción para reducir los tiempos en la generación de las visualizaciones.

6. Conclusiones

Los objetivos propuestos al inicio de este trabajo, tanto el general como los específicos, se cumplieron a cabalidad. De esta forma se logró proveer alternativas de visualización de datos recientes según los requerimientos propuestos, luego se dotó a las visualizaciones con capacidades interactivas de exploración de los datos brindadas a través de múltiples alternativas de visualización y por último se permitió la generación de reportes imprimibles a partir de una visualización de datos reciente, para su posterior análisis y discusión. Destacando la creación de un dashboard analítico para contener las visualizaciones diseñadas y poder generar los reportes.

En un comienzo, solo se pensó en crear visualizaciones dentro de una plataforma ya desarrollada, como era el caso de la utilización de Kibana (perteneciente a la Suite ELK), la cual se descartó por la nula integración con las partes del sistema de monitoreo SHM que está en desarrollo como proyecto FONDEF. Es por lo anterior, que no se siguieron indagando en alternativas ya implementadas y se centró el análisis en buscar herramientas que permitieran la creación de un dashboard personalizado. Ahí es donde surge el framework Plotly Dash como mejor alternativa, en donde se tuvo que aprender a utilizar y junto con ello, aprender técnicas para el manejo de grandes volúmenes de información en el lenguaje Python. Todo esto, permitió consolidar las visualizaciones creadas en un solo lugar, permitiendo una total integración con los demás módulos de la plataforma.

En cuanto a la creación de las visualizaciones, el mayor problema que se tuvo durante la realización de este trabajo, surgió en la obtención de datos reales de diferentes tipos de sensores, los cuales en su gran mayoría no son de uso público y por ende no están disponible en plataformas para su descarga. Es por esto que los datos utilizados en este trabajo no englobaron todos los tipos de sensores que se esperaban, solo limitándose a los de tipo acelerómetro y weather station que se obtuvieron del puente “Tianjin Yonghe”, ubicado en Harbin, provincia de Heilongjiang, China, usados para un trabajo previo relacionado con el monitoreo de la salud estructural de puentes [4]. Otro problema relacionado con las visualizaciones, surgió en el manejo de una gran cantidad de datos, los que no podían ser directamente visualizados, ya que producían un bajo rendimiento en el subsistema y por ende se le debía hacer un procesamiento previo, el cual según las pruebas realizadas arrojó que la mejor manera de hacer este procesamiento en las condiciones actuales es que en cada consulta a la base de datos, se generan los dataframes en la medida que estos sean requeridos.

Como trabajo futuro, se plantea la integración de este subsistema con el resto de la plataforma desarrollada en el marco del proyecto FONDEF al cual pertenece este

trabajo, brindándoles un módulo completo de visualizaciones de datos recientes a la plataforma, que permite la personalización e interacción de las visualizaciones, junto con generar reportes de ellas. Además se espera poder continuar estudiando formas de mejorar los tiempos de respuesta de las visualizaciones y evaluar la alternativa de generar los dataframes previamente con un equipo de prestaciones superiores a las que se utilizaron en este trabajo. Este subsistema creado, también provee la facilidad de realizar un análisis visual preliminar para detectar los aspectos clave presentes en los datos, como valores extremos, relaciones entre variables, tendencias, patrones y peaks. Para ello dispone de un entorno gráfico que permite visualizar datos usando diferentes tipos de visualizaciones y de manera sencilla modificar los datos a visualizar e interactuar con las visualizaciones propuestas para cada tipo de sensor.

7. Referencias

[1] Wenzel, H. (2009a). From Structural Health Monitoring to Risk based Infrastructure Management. Paper presented at the 4th International Conference on Structural Health Monitoring on Intelligent Infrastructure (SHMII-4) 2009, Zurich, Switzerland.

[2] Inaudi, D. (2012). SHM Live Web-based Data Management Software.

[3] ARTeMIS Modal Pro User Manual 2019, Structural Vibration Solutions A/S.

[4] Li, S., Li, H., Liu, Y., Lan, C., Zhou, W., Ou, J. (2014), SMC structural health monitoring benchmark problem using monitored data from an actual cable-stayed bridge. Struct. Control Health Monit., 21: 156-172. doi:10.1002/stc.1559

[5] Martin Stöger, Martin Fritz, R. Berger. (2010). Web based Monitoring and Assessment of Bridges and Structures. IABMAS 2010 - The Fifth International Conference on Bridge Maintenance, Safety and Management July 11-15, 2010, Philadelphia, USA

[6] Shah, N., Willick, D. & Mago, V. (2018). A framework for social media data analytics using Elasticsearch and Kibana. Wireless Network.

[7] Sohn, H., Farrar, C. R., Hemez, F. M., Czarnecki, J. J. (2002) A Review of Structural Health Monitoring Literature 1996-2001., article, January 1, 2002; United States. University of North Texas Libraries, UNT Digital Library.

[8] Ministerio de Obras Públicas. (2018). Informe de Puentes Dirección de Vialidad. Retrieved September 11, 2019, from https://www.mop.cl/Prensa/Paginas/InformePuentes.aspx

[9] Sazonov, E., Krishnamurthy, V., & Schilling, R. (2010). Wireless Intelligent Sensor and Actuator Network - A Scalable Platform for Time-synchronous Applications of Structural Health Monitoring. Structural Health Monitoring, 9(5), 465–476.

[10] Elasticsearch B.V. (2020) THE ELASTIC STACK - Conoce los productos principales Retrieved enero, 2020, from https://www.elastic.co/es/elastic-stack

[11] Django Software Foundation. (2020). Why Django? Retrieved enero, 2020, from https://www.djangoproject.com/start/overview/

[12] Armin Ronacher and contributors. (2015). Flask. Retrieved enero, 2020, from https://www.palletsprojects.com/p/flask/

[13] Armin Ronacher and contributors. (2015). Werkzeug. Retrieved enero, 2020, from https://www.palletsprojects.com/p/werkzeug/

[14] Armin Ronacher and contributors. (2015). Jinja. Retrieved enero, 2020, from https://www.palletsprojects.com/p/jinja/

[15] Python Software Foundation. (2020). About. Retrieved enero, 2020, from https://www.python.org/about/

[16] Plotly. (2020). Dash operationalizes Python & R models at scale. Retrieved enero, 2020, from https://plot.ly/dash/

[17] Plotly. (2020). Plotly Open Source Graphing Libraries. Retrieved enero, 2020, from https://plot.ly/graphing-libraries/

[18] Facebook Inc. (2020). React Una biblioteca de JavaScript para construir interfaces de usuario. Retrieved enero, 2020, from https://es.reactjs.org/

[19] The PostgreSQL Global Development Group. (2020). New to PostgreSQL? Retrieved enero, 2020, from https://www.postgresql.org/

[20] Timescale, Inc. (2020). How TimescaleDB works. Retrieved enero, 2020, from https://www.timescale.com/products

[21] Desjardins, S. L., Londoño, N. A., Lau, D. T., & Khoo, H. (2006). Real-Time Data Processing, Analysis and Visualization for Structural Monitoring of the Confederation Bridge. Advances in Structural Engineering, 9(1), 141-157.

[22] Zankl, S. (2009). Visualizing Sensor Data.

[23] Branko Glisic, Daniele Inaudi, and Nicoletta Casanova "SHM process as perceived through 350 projects", Proc. SPIE 7648, Smart Sensor Phenomena, Technology, Networks, and Systems 2010, 76480P (8 April 2010); https://doi.org/10.1117/12.852340

8. Anexo

8.1. Esquema de la base de datos