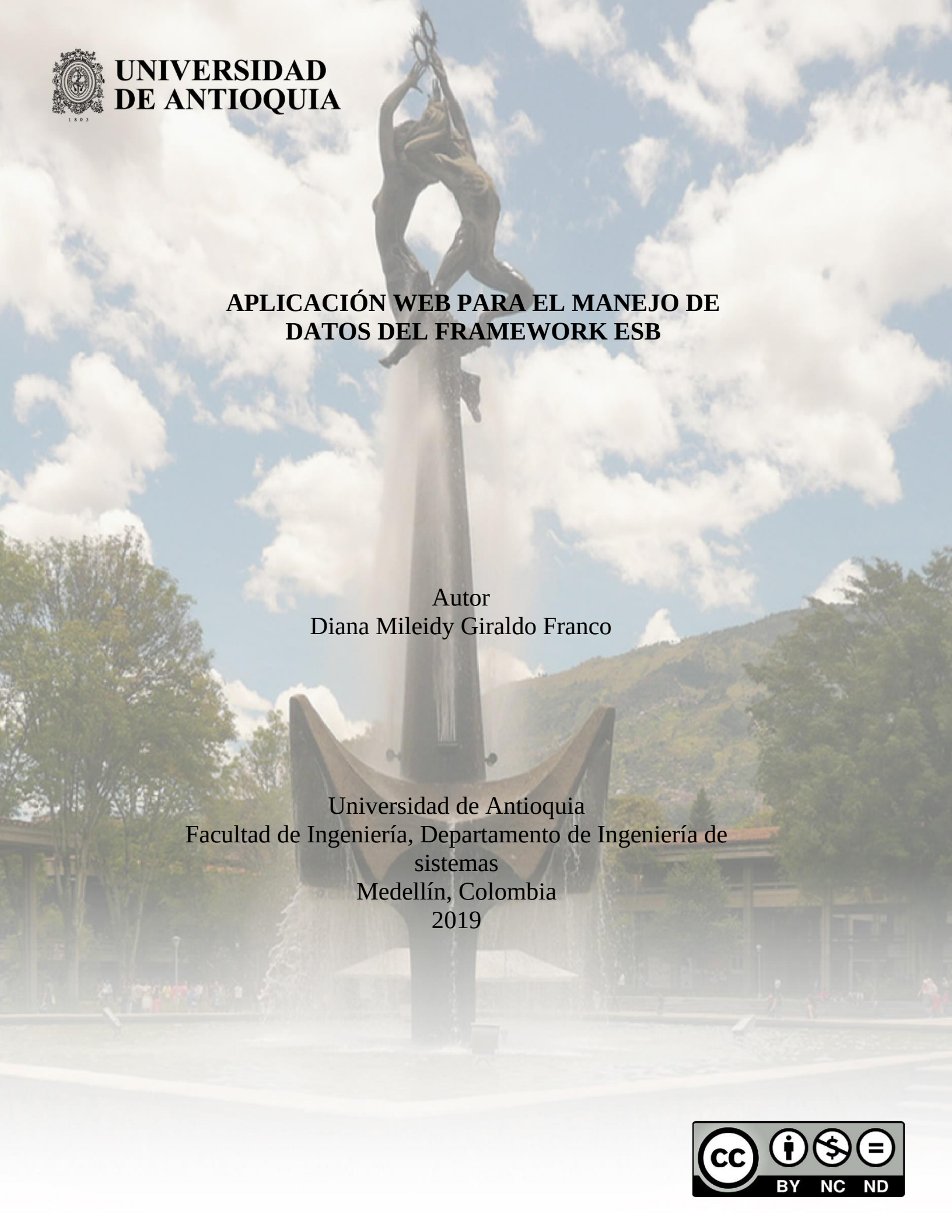

UNIVERSIDAD DE ANTIOQUIA

APLICACIÓN WEB PARA EL MANEJO DE DATOS DEL FRAMEWORK ESB

Autor
Diana Mileidy Giraldo Franco

Universidad de Antioquia
Facultad de Ingeniería, Departamento de Ingeniería de sistemas
Medellín, Colombia
2019

Aplicación web para el manejo de datos del framework ESB

Diana Mileidy Giraldo Franco

Informe de práctica
como requisito para optar al título de:
Ingeniería de Sistemas.

Asesores:

Fredy Alexander Rivera Vélez, PhD
Departamento de Ingeniería de Sistemas, UDEA

Miguel Ríos
Ingeniero de Sistemas

Universidad de Antioquia
Facultad de Ingeniería, Departamento de Ingeniería de Sistemas
Medellín, Colombia
2019.

Contenido

Aplicación web para el manejo de datos del framework ESB

Resumen

El personal encargado del área de integración de servicios de la empresa Sofka Tecnologies hallaba inconvenientes cuando manipulaba la información alojada en la base de datos de cada cliente, tales como la inserción de forma manual, debido a sus dependencias y claves foráneas, que varían en diferentes tipos de desarrollos, al igual que la generación de *scripts* dependiendo cada ambiente (Desarrollo, Prueba, Preproducción, Producción).

Debido a estos problemas se identificó la necesidad de que los desarrolladores centralizaran todas sus funciones en una misma herramienta, para que no hubiera una manipulación directa de la base de datos, Por esta razón, mediante esta práctica académica se creó una aplicación web para el manejo de base de datos del *framework* ESB con el fin de realizar el manejo adecuado de la información de los servicios que contiene cada empresa cliente, mejorar tiempos de respuesta y minimizar los errores al momento de realizar esta manipulación.

La aplicación se desarrolló bajo una metodología ágil, y está compuerta por una capa de lógica llamada back-end, y otra capa que interactúa con los usuarios del área de integración llamada front-end. Además, se desarrollaron varios componentes más como un generador de código que permite a los desarrolladores generar el script SQL para los diferentes ambientes. Otro componente importante fue el de inteligencia operacional para realizar análisis de los datos de cada servicio en tiempo real, optimizar las tomas de decisiones, detectar amenazas, realizar monitoreo de las transacciones exitosas y fallidas por cada servicio, y determinar cuántos servicios se encuentran desarrollados y desplegados en cada ambiente.

En cuanto a la gestión de tiempo, la aplicación se realizó con el propósito de que fuera más ágil para ingresar o consultar un servicio a la base de datos, para responder rápidamente a las inquietudes de los analistas de integración de las empresas clientes.

Introducción

La compañía tiene un equipo humano especializado en el área de bus de integración de servicio empresarial (Enterprise Service Bus, ESB), basado en la arquitectura SOA (Service-Oriented Architecture). Para facilitar el desarrollo en los clientes (empresas), se cuenta con un *framework* propio, compuesto de varias capas, pero este proyecto se enfocó en las capas: receiver, controller, aif y adapter. Todas estas capas tienen en común la base de datos de Oracle, la cual contiene toda la información necesaria para el funcionamiento de los servicios expuestos al consumidor final.

Para el manejo de información del *framework* ESB con la base de datos, los desarrolladores antes de este proyecto utilizaban algún software de preferencia, para realizar cualquier tipo de operación, siendo las más comunes, inserciones y consultas. Sin embargo, como todo se realizaba de forma manual, se presentaban errores a la hora de insertar o manipular la información. Esto se debía a que los datos debían ser insertados en un orden especifico, debido a las dependencias o claves foráneas que se tienen con respecto a las otras tablas. Adicionalmente, la información puede variar dependiendo del desarrollo que se ejecute.

Generalmente, los clientes pueden llegar a manejar varios ambientes de desarrollo, pruebas, preproducción y producción. En cada ambiente se maneja una base de datos única, por lo cual los desarrolladores debían generar *scripts* de un ambiente a otro.

Ante esta situación, se desarrolló una aplicación Web bajo la metodología Kanban, la cual nos permite tener una mayor visualización de las etapas del desarrollo y el flujo de trabajo, detectando rápidamente retrasos y así lograr estimaciones de tiempos adecuados por cada tarea. La aplicación cuenta con una conexión a la base de datos de *framework* ESB de la compañía, por un medio seguro para el manejo de información de los servicios, mejorando la administración de los datos buscando que se almacenen con mayor precisión. Además, agiliza el tiempo de respuesta a la hora de llevar estos registros a diferentes ambientes de desarrollo, y dar soluciones pertinentes a los analistas de integración de las empresas clientes.

Con la integración de servicios se permite que los sistemas que los contienen se comuniquen entre sí, y así lograr dar soluciones unificadas. Gracias a esta aplicación no solo es posible que las empresas obtengan comunicación entre sus servicios, sino que también los desarrolladores realicen un manejo adecuado de la información y, como valor agregado, los tiempos de respuesta de los desarrolladores ante alguna problemática sean más rápidos debido a la capa de monitoreo, que tiene un análisis dinámico del negocio en tiempo real, para detectar situaciones de ineficiencias, oportunidades y amenazas.

Objetivo general:

Crear una aplicación web, en la cual los desarrolladores puedan realizar funciones básicas de manera óptima en cada una de las capas que componen el *framework* ESB, con el fin de aislar el contacto directo a las bases de datos del mismo, proveyendo una interfaz amigable e intuitiva.

Objetivos específicos

- Analizar el funcionamiento de cada capa del *framework* dentro de la base de datos.
- Realizar el análisis de requerimientos funcionales y no funcionales de la aplicación.
- Diseñar una interfaz en la cual los desarrolladores puedan ingresar, consultar, actualizar y/o eliminar registros de un servicio.
- Emplear la base de datos del *framework* ESB de la compañía para que desde la aplicación web se administren los datos.
- Implementar un generador de código SQL para exportar los *scripts* de la base de datos, con el objetivo de que puedan ser insertados en los diferentes ambientes existentes.

Marco Teórico

A continuación, se abordan diferentes términos empleados en el presente informe que deben ser entendidos en el contexto del proyecto, como arquitectura SOA, *framework*, bus de integración de servicios, capas del *framework*, colas MQ, grupos de ejecución e inteligencia operacional, tomando como base lo planteado por distintos autores y entidades oficiales.

El conocimiento de una arquitectura SOA fue fundamental en el desarrollo del proyecto debido a que el *framework* ESB de la compañía es un componente de una arquitectura de este tipo. Por lo tanto, para efectos prácticos del proyecto, se define la arquitectura SOA como un estilo de Arquitectura de Software basado en la definición de servicios reutilizables, con interfaces públicas bien definidas, donde los proveedores y consumidores de servicios interactúan en forma desacoplada para realizar los procesos de negocio [1].

Por su parte, el bus de servicio empresarial (ESB) es una colección de patrones arquitectónicos basados en la integración de aplicaciones empresariales [2]. Por tal motivo, la función principal del *framework* ESB es integrar diferentes servicios de acuerdo con las necesidades de los clientes. Según Martínez, un *framework* agrega funcionalidad extendida a un lenguaje de programación, automatiza muchos de los patrones de programación para orientarlos a un determinado propósito, proporcionando una estructura al código, mejorándolo y haciéndolo más entendible y sostenible, y permite separar en capas la aplicación [3].

Siguiendo lo expuesto en el párrafo anterior, el *framework* ESB de la compañía proporciona diversas funciones en el código y se compone de varias capas que permiten tener alta reutilización de componentes [4], como se puede ver en la Figura 1.

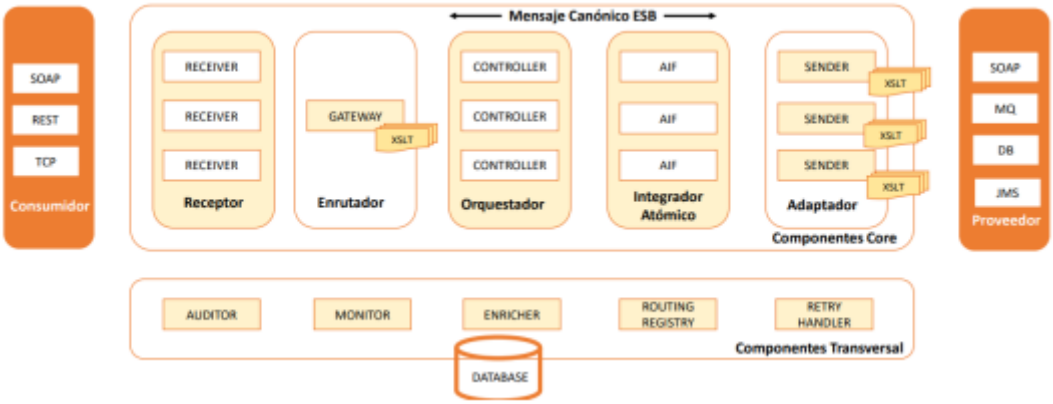

Figura 1. Framework ESB (Fuente: SOFKA)

Las capas del *framework* que son relevantes en el presente proyecto son las siguientes:

RECEIVER (Receptores de interfaz): esta capa es la entrada principal de todas las interfaces expuestas desde el ESB para consumidores, encargada de recibir el mensaje que se desea enviar al emisor.

CONTROLLER (Controlador / Orquestador): el propósito de esta capa se cumple cuando una interfaz requiere un complejo de lógica de orquestación de servicios o enrutamiento que requiere integrar con más de un sistema de emisor.

AIF (Integrador atómico): es el componente lógico de mensajería canónico por cada interfaz de destino con alta reutilización. El propósito de esta capa es realizar algunos tipos de transformaciones o traducciones de mensajería.

SENDER (Adaptadores): el objetivo de esta capa es tener componentes reutilizables para integrar con varios sistemas de destino, basado en diferentes protocolos de comunicación.

Cada capa se encuentra asociada a una tabla de la base de datos del *framework* ESB, cuyas configuraciones se pueden consultar en la Tabla A1 de los anexos.

En la base de datos hay dos términos que contiene cada capa mencionada anteriormente y que se deben tener claros para la administración de los datos, a saber: grupos de ejecución y cola.

Los grupos de ejecuciones son una agrupación de flujos de mensajes que se han asignado a un intermediario. El intermediario impone un grado de aislamiento entre los flujos de mensajes de los distintos grupos de ejecución, asegurándose de que se ejecuten en espacios de direcciones distintos o como procesos únicos [5]. El *framework* los utiliza como pequeños servidores donde se encuentran desplegados los flujos de cada servicio. Por otra parte, una cola es un contenedor de mensajes, donde se pueden recuperar y transferir los mensajes [6]. Por medio del *framework*, los servicios utilizan las colas para dejar los mensajes tanto de entrada como de salidas, buscando que se puedan recuperar en otra capa del *framework*.

Los desarrolladores deben conocer el modelo de negocio de las empresas clientes, por tal motivo se considera como base principal en la aplicación, el componente de inteligencia operacional [7], ya que les permite un análisis dinámico de negocios, en tiempo real el cual nos trae una mayor visibilidad y comprensión en el sentido de las operaciones de negocios.

Metodología

El desarrollo se realizó bajo la metodología Kanban [8], con el objetivo de gestionar de manera general cómo se iban completando las tareas, estableciendo metas asequibles. Se realizó una reunión con los desarrolladores del área de integración y con el arquitecto, para hacer la especificación de requerimientos, basados en historias de usuarios con criterios de aceptación y casos de usos de cada funcionalidad, los cuales pueden ser consultados en las Figuras A1 hasta A6 de los anexos.

Se desarrolló la estimación a cada tarea basada en los objetivos y el alcance de la aplicación para mejores resultados, utilizando la herramienta Trello [9], para definir el flujo del trabajo donde se tuvo una visión general del proyecto.

Se inició por las capacitaciones necesarias para desarrollar el proyecto. La primera capacitación fue sobre el *framework* ESB de la empresa, la cual duró un mes, en donde fueron explicados sus componentes y funciones. Luego la capacitación continuó con los *frameworks* Angular y Spring Boot, la cual duró otro mes.

El desarrollo empezó con la capa de datos, en donde se duplicó la base de datos del *framework*, para utilizarla como base de datos para pruebas. El desarrollo continuó con la capa del back-end en donde está contenida la lógica de la aplicación. Dos semanas después se empezó el desarrollo de la capa del front-end encargada de la interactividad con el usuario. El desarrollo se continuó de forma paralela con ambas capas.

Todos los viernes se realizaba una reunión con el arquitecto y con el líder del área de Integración, en la cual se mostraba los avances del proyecto, obteniendo retroalimentaciones, y de ser necesario capacitaciones sobre algunos temas de programación como Bootstrap, Angular, Java entre otros.

En la etapa final del desarrollo se necesitaron tres semanas para realizar las pruebas y verificar su buen funcionamiento. Durante el transcurso de las mismas se dedicaron 1,5 semanas para una capacitación sobre pruebas de software, que abarcaba pruebas unitarias y de integración utilizando librerías como Junit, Mockito, Qunit y mocha.js. Durante las últimas 1.5 semanas, la empresa decidió pasar el proyecto al área de QA para que realizara las pruebas finales.

Semanalmente, se hizo la debida documentación del proceso y se realizaron entregas cada 15 días de un informe al área de integración, donde se detallaba el avance, con los objetivos cumplidos y objetivos pendientes para la próxima entrega.

Resultados y análisis

Con las capacitaciones brindadas por el arquitecto antes de inicializar el desarrollo fue posible abarcar a nivel conceptual el funcionamiento de todas las capas que contiene el *framework* ESB de Sofka.

La especificación de requerimientos se obtuvo a través de una reunión con el arquitecto y los desarrolladores del área de integración, basándonos en historias de usuarios con criterios de aceptación y casos de usos de cada funcionalidad.

La aplicación brinda a los desarrolladores del área de integración una herramienta para mejorar la administración de los datos, permitiéndoles agregar, consultar y actualizar adecuadamente la información de los servicios, sus colas y grupos de ejecución, con mayor agilidad y facilidad.

Los desarrolladores lograron obtener un mejor rendimiento en tiempo, ahora pueden monitorear los servicios, y entender el modelo de negocio de los clientes con mayor facilidad dando respuestas rápidas a los analistas en cuanto a fallas, dudas y opiniones. También pueden generar y descargar con más rapidez el *script* de un registro específico, con el fin de aislar el contacto directo a las bases de datos del *framework* ESB.

Diagrama de Contexto

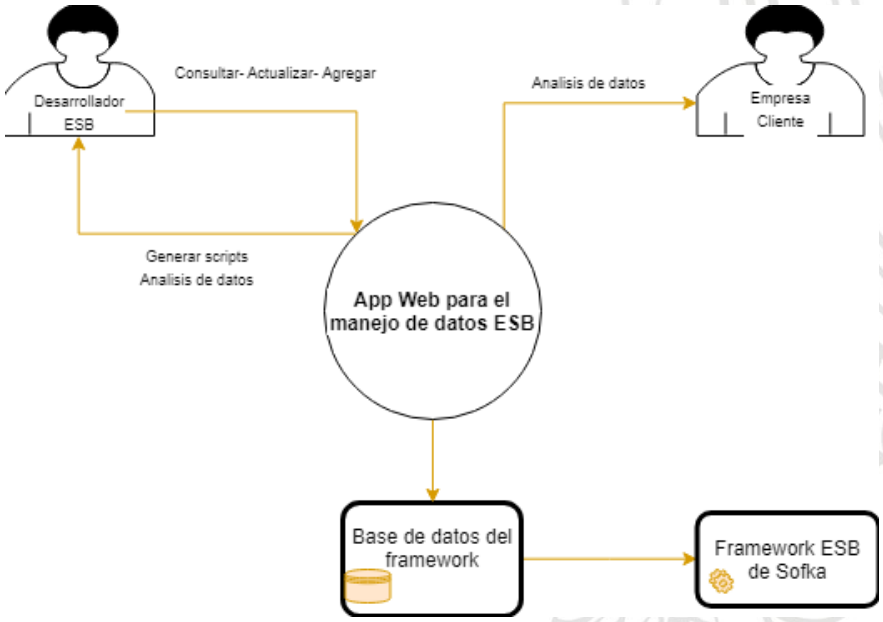

Figura 2. Contexto de la aplicación (Fuente: SOFKA)

La aplicación permite que un desarrollador pueda consultar, actualizar y agregar un nuevo servicio a la base de datos del *framework* que conecta con el *framework* ESB de Sofka. Además, le da acceso para generar *scripts* o realizar análisis de los datos, mientras que la empresa cliente solo tiene permiso para realizar análisis de los datos utilizando el componente de inteligencia operacional (Figura 2).

Prueba realizada a un servicio sobre su consumo en los últimos meses

Se realizó la prueba a un servicio ya configurado para poder realizar una consulta por medio de la aplicación. En la Figura 3 se observa una gráfica con el consumo del servicio para hacer una comparación, con los status 200 que son los casos exitosos, y los status 500 que son los casos fallidos. Como resultado se obtiene la Tabla 1, en donde se detallan los criterios de aceptación y el tipo de prioridad.

Figura 3. Consumo de un servicio (Fuente: SOFKA)

Tabla 1. Detalle de prueba de la app framework ESB (Fuente: SOFKA)

Detalle de prueba	
Categoría	Funcional
Prioridad	Media
Resumen	Consumo de un servicio en los últimos meses
Descripción	El *framework*, en su tiempo de ejecución, genera asíncronamente logs de eventos, que contienen información de status y fechas
Evaluación de impacto	Impacto medio
Criterio de aceptación	En caso de consultar el consumo de un servicio se debe verificar que el servicio exista. Si muestra grafica sin datos, despliega un mensaje a los desarrolladores informando que el servicio no se ha consumido.
Elementos afectados	Componentes de inteligencia operacional
Confidencialidad	Sí

Prueba de los *scripts* generados de un servicio

Se realizó la prueba a un servicio ya configurado para poder realizar una consulta de los *scripts* por medio de la aplicación. En la Figura 4 se observan los *scripts* generados que describen el servicio, para su respectiva manipulación con otras bases de datos. Como resultado se obtiene la Tabla 2, en donde se detallan los criterios de aceptación, el tipo de prioridad y el nivel de impacto producido por la prueba.

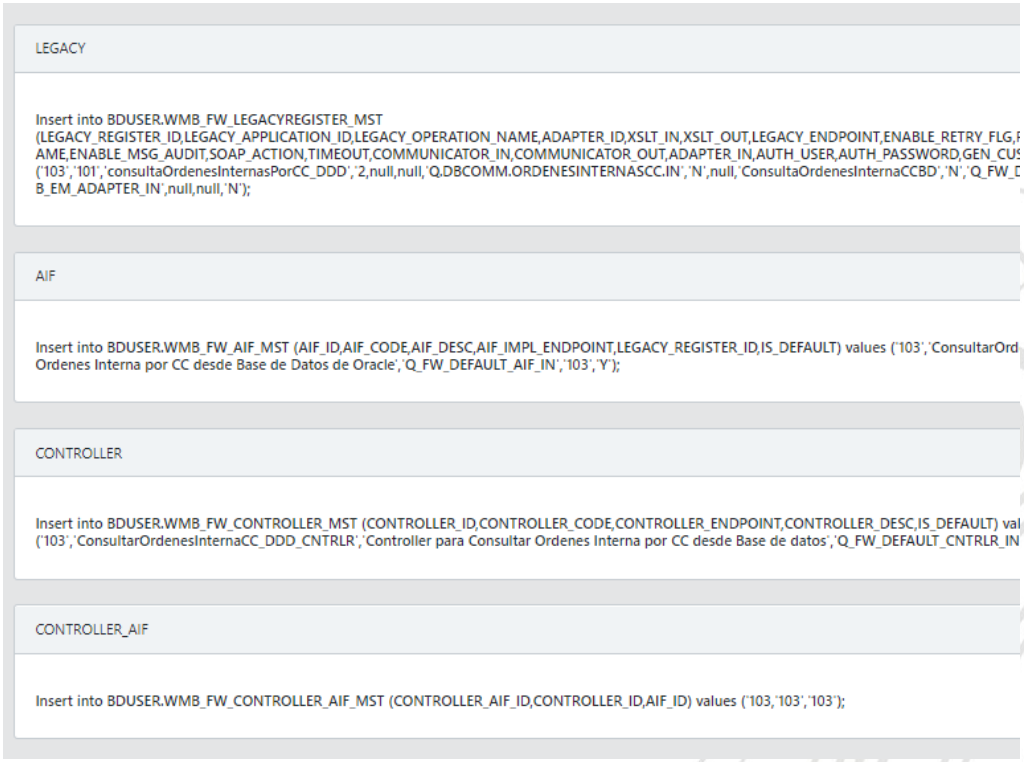

Figura 4. Scripts de un servicio (Fuente: SOFKA)

Tabla 2. Detalle de prueba de la app framework ESB (Fuente: SOFKA)

Detalle de prueba	
Categoría	Funcional
Prioridad	Alta
Resumen	Generación de *script* dependiendo el entorno
Descripción	El *framework* genera los *scripts* de un servicio específico dependiendo del entorno que desee el desarrollador
Evaluación de impacto	Impacto crítico
Criterio de aceptación	-En caso que el filtro aplicado a la consulta falle, el sistema debe permitir realizar otra vez la consulta. -En caso de que el usuario no haya seleccionado ningún registro para generar el *script*, el sistema debe permitir volver a generarlo.
Elementos afectados	Componente de consulta
Confidencialidad	Sí

Conclusiones

La compañía Sofka es experta en desarrollo de software, ofrece servicios y soluciones a clientes en cuanto a tecnología, y está comprometida con garantizar el éxito de sus proyectos.

Para el desarrollo de la aplicación se realizó un análisis completo, en compañía del arquitecto, de todas las capas del *framework* dentro de la base de datos, y así realizar una interfaz en la cual los desarrolladores, lograran manejar la información de los servicios de cada cliente, teniendo como fundamento los requerimientos funcionales y no funcionales tomados al inicio del proyecto.

Ahora realizan consultas, actualizan y agregan servicios de las empresas clientes con mayor agilidad, tiempo que pueden invertir en los desarrollos, además en momentos de instalación a diferentes entornos, los desarrolladores hacen la entrega de script solicitados sin demora, para sus respectivas ejecuciones.

Haber realizado mis prácticas empresariales en la compañía Sofka fue una de mis mejores experiencias, ya que era un aprendizaje continuo al lado de personas talentosas, llenas del conocimiento y la disposición necesaria para enseñar, donde mejoré mis habilidades de análisis y desarrollo de software. Además, la realización de esta aplicación web me motivó entre otras cosas a experimentar nuevas tecnologías, a investigar y estudiar por cuenta propia los temas que estuvieran en mis manos y a aprender nuevos lenguajes de programación, cosas que en definitiva me nutren en demasía y me dejan aún más preparada para mi desarrollo profesional.

Referencias Bibliográficas

[1] DELGADO, Andrea, et al. Desarrollo de aplicaciones con enfoque SOA. Montevideo. Instituto de Computación. 2006.4p

[2]Anónimo. Introduction to the Microsoft ESB Guidance [en línea]. Microsoft, MSDN. (27 de julio de 2010). [Consultado: 15 de febrero de 2019]. Disponible en internet: https://msdn.microsoft.com/en-us/library/ff648282.aspx

[3]MARTINEZ, Gustavo. DISEÑO DE FRAMEWORK WEB PARA EL DESARROLLO DINÁMICO DE APLICACIONES. Abril de 2010. Universidad Tecnológica de Pereira. ISSN 0122-1701

[4] DUTTA, Goutam. FWK SOFKA. Medellin. Sofka technologies. 28 de octubre de 2016

[5] Anónimo. Grupos de ejecución [en línea]. IBM. (28 de febrero de 2015). [Consultado: 12 de febrero de 2019]. Disponible en internet: https://www.ibm.com/support/knowledgecenter/es/SSKM8N_8.0.0/com.ibm.etools.mft.doc/ae00270_.htm

[6] Anónimo. Colas de IBM MQ [en línea]. IBM. (s.f). [Consultado: 12 de febrero de 2019]. Disponible en internet: https://www.ibm.com/support/knowledgecenter/es/SSFKSJ_8.0.0/com.ibm.mq.explorer.doc/e_queues.htm

[7] Anónimo. Big data e inteligencia operacional: una conexión para la vida. [en línea]. IDbox. (23 de Noviembre de 2018) [Consultado: 06 de Junio de 2019]. Disponible en internet: https://idboxrt.com/blog/big-data-e-inteligencia-operacional/

[8] Anónimo. ¿Por qué utilizar la metodología Kanban? [en línea]. Kanbantool. (s.f). [Consultado: 20 de febrero de 2019]. Disponible en internet: https://kanbantool.com/es/metodologia-kanban

[9] Anónimo. Trello. [en línea]. Atlassian (13 de septiembre de 2011) [Consultado: 20 de febrero de 2019]. Disponible en internet: https://trello.com/

[10] Anónimo. Introducción a MQ Explore. [en línea]. IBM. (17 de mayo de 2017). [Consultado: 20 de febrero de 2019]. Disponible en internet: https://www.ibm.com/support/knowledgecenter/es/SSFKSJ_8.0.0/com.ibm.mq.explorer.doc/help_home_wmq.htm?pos=2

Anexos

Las tablas de configuración del *framework* se pueden observar en la Tabla A1, las cuales fueron conectadas y configuradas dentro de la aplicación web.

Tabla A1. Base de datos del framework ESB (Fuente: SOFKA)

FW_APPLICATION_MST	Tabla para listar todos los sistemas destinarios que son para integrar
FW_ADAPTER_MST	Tabla para listar los adaptadores configurado para el *framework*
FW_LEGACYREGISTER_MST	Tabla para listar cada interfaz (servicio/operación) expuesto por sistema de destinarios. Se referencian los adaptadores y aplicaciones
FW_AIF_MST	Tabla para listar los flujos atómicos
FW_CONTROLLER_MST	Tabla para listar los controladores
FW_CONTROLLER_AIF_MST	Tabla para listar la relación entre controlador y flujos atómicos
FW_RECEIVER_MST	Tabla para listar los receptores expuesto por ESB

Casos de uso

- Consultar operación de un servicio

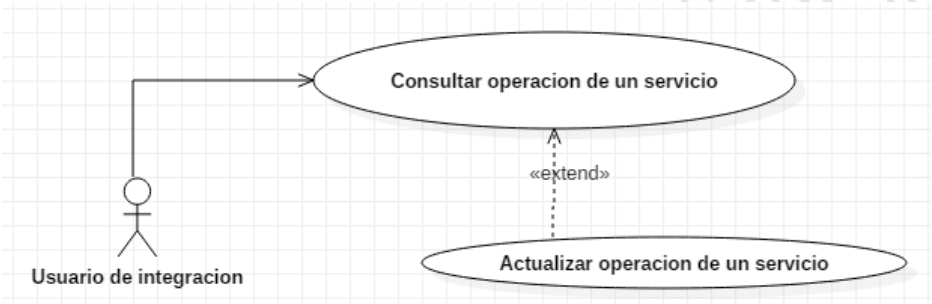

Figura A1. Caso de uso: Consultar operación de un servicio (Fuente: SOFKA)

- Registrar operación de un servicio

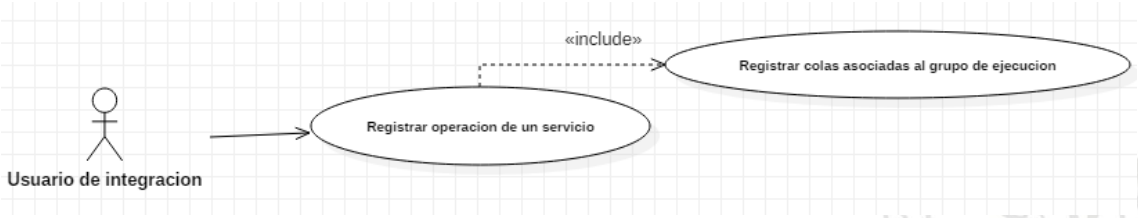

Figura A2. Caso de uso: Registrar operación de un servicio (Fuente: SOFKA)

- Actualizar operación de un servicio

Figura A3. Caso de uso: Actualizar operación de un servicio (Fuente: SOFKA)

- Agregar un legado a un receiver existente

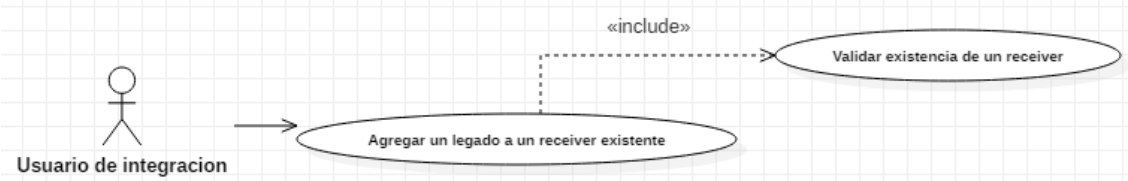

Figura A4. Caso de uso: Agregar un legado a un receiver existente (Fuente: SOFKA)

- Reutilizar un legado para un nuevo receiver

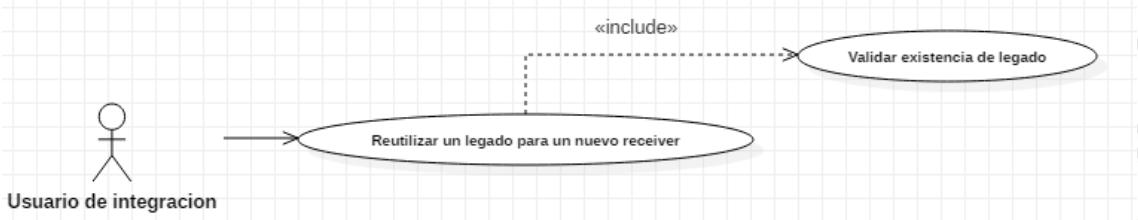

Figura A5. Caso de uso: Reutilizar un legado para un nuevo receiver (Fuente: SOFKA)

- Generar script de una operación de un servicio

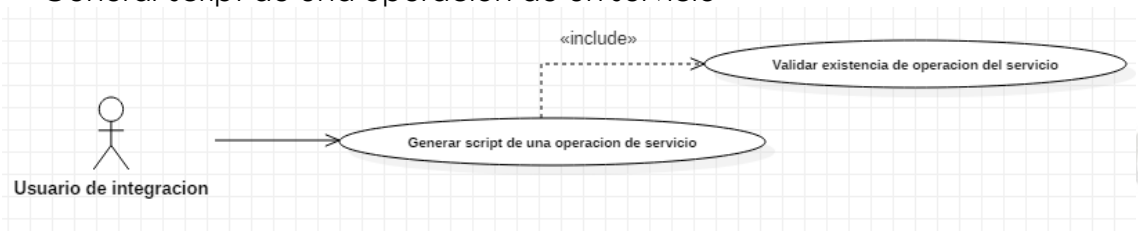

Figura A6. Caso de uso: Generar script de una operación de un servicio (Fuente: SOFKA)