UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA:

INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de:

Ingeniera e Ingeniero de Sistemas

TEMA:

REFACTORIZACIÓN DE LA APLICACIÓN WEB DE SEGUIMIENTO A DOCENTES CON MIRAS A LA CREACIÓN DE UNA SUITE DE HERRAMIENTAS INFORMÁTICAS DE APOYO A LA GESTIÓN DE LAS CARRERAS DE INGENIERÍA DE SISTEMAS Y CIENCIAS DE LA COMPUTACIÓN

AUTORA Y AUTOR:

ALEXANDRA CAROLINA CUTIUPALA NARVÁEZ

JOAO DANIEL TOINGA YARPAZ

TUTOR:

FRANKLIN EDMUNDO HURTADO LARREA

Quito, febrero del 2020

CESIÓN DE DERECHOS DE AUTOR

Nosotros, Alexandra Carolina Cutiupala Narváez con documento de identificación N°. 1724690571 y Joao Daniel Toinga Yarpaz, con documento de identificación N°. 2100216015, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación intitulado: "**REFACTORIZACIÓN DE LA APLICACIÓN WEB DE SEGUIMIENTO A DOCENTES CON MIRAS A LA CREACIÓN DE UNA SUITE DE HERRAMIENTAS INFORMÁTICAS DE APOYO A LA GESTIÓN DE LAS CARRERAS DE INGENIERÍA DE SISTEMAS Y CIENCIAS DE LA COMPUTACIÓN**", mismo que ha sido desarrollado para optar por el título de: INGENIERA E INGENIERO DE SISTEMAS, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en digital a la Biblioteca de la Universidad Politécnica Salesiana.

ALEXANDRA CAROLINA
CUTIUPALA NARVÁEZ
C.I.: 1724690571

JOAO DANIEL
TOINGA YARPAZ
C.I.: 2100216015

Quito, febrero del 2020

DECLARATORIA DE COAUTORÍA DEL TUTOR

Yo, declaro que bajo mi dirección y asesoría fue desarrollado el trabajo de titulación con el tema: REFACTORIZACIÓN DE LA APLICACIÓN WEB DE SEGUIMIENTO A DOCENTES CON MIRAS A LA CREACIÓN DE UNA SUITE DE HERRAMIENTAS INFORMÁTICAS DE APOYO A LA GESTIÓN DE LAS CARRERAS DE INGENIERÍA DE SISTEMAS Y CIENCIAS DE LA COMPUTACIÓN realizado por Alexandra Carolina Cutiupala Narváez y Joao Daniel Toinga Yarpaz, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerados como trabajo final de titulación.

Quito, febrero del 2020

FRANKLIN EDMUNDO HURTADO LARREA

CI: 1713382016

DEDICATORIA

Dedico el presente trabajo a mi madre Alexandra Narváez que con su paciencia y amor me enseñó a ser una mujer amable, responsable y generosa, a mi padre Ostacio Cutiupala es la persona que con su ejemplo de lucha y perseverancia me demostró que todo lo que uno se propone en la vida siempre se puede lograr.

A mis hermanos Christian, Diana, Fabricio por estar conmigo en cada momento especial celebrando cada logro que he cumplido en el transcurso de mi vida y también por su apoyo incondicional.

A mi novio Fernando por confiar en mí en todo momento y siempre estar a mi lado.

A toda mi familia porque son mi fortaleza para seguir adelante y cumplir todos mis sueños y metas.

Alexandra Carolina Cutiupala Narváez

AGRADECIMIENTOS

A nuestros padres por apoyarnos en todo este tiempo de nuestra carrera y brindarnos los consejos necesarios para alcanzar todas nuestras metas.

A la Universidad Politécnica Salesiana que con valores nos enseñaron a ser unas personas con calidad humana, también por la formación académica impartida por excelentes docentes que nos ayudaron a prepararnos todo este tiempo.

A nuestros amigos que siempre nos apoyaron y estuvieron con nosotros en los buenos y malos momentos.

A nuestro tutor el Ing. Franklin Hurtado que estuvo pendiente en desarrollo de este proyecto y por brindarnos su tiempo y guiarnos para poder cumplir con este reto.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En el presente documento se detalla el proceso de refactorización a la aplicación web de seguimiento a docentes, para adaptarlo a los nuevos requerimientos de las carreras de Ingeniería de Sistemas y Ciencias de la Computación, que mejora la mantenibilidad y facilita la integración de la aplicación. En el desarrollo de la refactorización se hace un análisis de las tareas de evaluación de los docentes para evidenciar las modificaciones y nuevas funcionalidades que se deben implementar en el software, detallando la metodología usada y cambios que se realizan basados en la aplicación de evaluación a docentes anterior, también se detallan las modificaciones en la arquitectura para poder conferir una mayor capacidad de integración mediante tecnologías modernas de desarrollo web a la fecha de presentación de este trabajo, y las pruebas que se ejecutaron al software.

ABSTRACT

This document details the process of web application refactoring for teacher monitoring, to adapt it to the new requirements of the Systems Engineering and Computer Science careers, which improves maintainability and facilitates the application integration. In the progress of the refactoring, an analysis of the evaluation tasks of the teachers is made to show the modifications and new functionalities that must be implemented in the software, detailing the methodology used and changes that are made based on the previous evaluation to teachers application, also modifications in the architecture are detailed to be able to confer a greater integration capacity by means of modern technologies of web development to the date of presentation of this work, and the tests that were executed to the software.

INTRODUCCIÓN

La naturaleza del software es cambiante “El software debe evolucionar para cubrir las necesidades cambiantes del cliente.” (Sommerville, 2011) y necesita de técnicas que se puedan utilizar para mejorar la experiencia de usuario y sobre todo modificar un software con una arquitectura orientada a servicios que facilite la integración total de la aplicación.

La técnica más adecuada para realizar cambios en una aplicación es la refactorización porque según Martin Fowler “La refactorización es una técnica controlada para mejorar el diseño de una base de código existente. Su esencia es aplicar una serie de pequeñas transformaciones para preservar el comportamiento” (Martin Fowler, 2018) , así permite mantener el código en “buena forma” y reestructurarlo sin alterar su estructura lógica, implementado nuevas funcionalidades en su estructura interna para obtener un software mantenible y adaptable.

Con el paso del tiempo la aplicación web de seguimiento a docentes ya no se ajusta a los nuevos requerimientos de integración y mantenibilidad “La evolución a arquitecturas de micro servicios hacen que la obsolescencia del software duela menos: los reemplazos son puntuales, indoloros para el resto del ecosistema y realizables en fases.” (Castro, 2017) por la falta de actualización de nuevas funcionalidades como: asignar docente de apoyo, evaluación basada en criterios y categorías, creación de grupos para asignar tareas por eso se necesita implementar algunas modificaciones tanto en backend y el frontend, todos estos cambios se generaron ya que hubo una variación en el proceso de evaluación a los docentes, por ejemplo ahora se realiza la evaluación en base a criterios y preguntas,

también se realizaron ajustes en la malla de la carrera de ingeniería de sistemas lo cual ocasionó un cambio de nombre en la carrera, ahora es conocida como Ciencias de la Computación lo que generó un nuevo distributivo de usuarios por materia.

También esta aplicación no fue diseñada para la integración con un criterio de suite porque según Yhorman Sierra "Una suite consiste en un conjunto de programas informáticos integrados en un paquete. Dicho paquete contiene aplicaciones que les permiten automatizar, optimizar y mejorar la operatividad de la organización" (Sierra, 2020) esta posee una estructura modular que dificulta el mantenimiento y adaptabilidad en el sistema informático de gestión de la Universidad Politécnica Salesiana, el cual genera datos en tiempo real para la toma de decisiones para los estudiantes y los docentes en beneficio de toda la organización.

Este documento se estructura de la siguiente manera:

Capitulo uno, se describe los principales conceptos en el marco teórico que sirven como una guía en el desarrollo del proyecto.

Capitulo dos, se realiza el análisis del proceso de seguimiento a docentes, sus roles y funciones, también se analiza la aplicación a refactorizar y se diseña una versión actualizada del sistema.

Capitulo tres, se refactoriza la aplicación implementado las nuevas funcionalidades y se ejecutan las pruebas necesarias para obtener una aplicación de calidad.

Justificación

Es por eso que la aplicación de seguimiento a docentes anterior debe ser modificada porque se encontró que su back-end y front-end estaba unidos, el código era redundante y su arquitectura no permitía realizar modificaciones, pero con la técnica de refactorización “Refactorización es realizar modificaciones en el código con el objetivo de mejorar su estructura interna, sin alterar su comportamiento externo” (Martin Fowler, 2018) se puede reorganizar el programa para restructurar el código y agregar nuevas funcionalidades basándonos en su lógica original, obteniendo así una versión mejorada con una arquitectura compatible que permita ser modificable para agregar nuevas funcionalidades y servicios.

Así la nueva versión lucirá con un distinto diseño de interfaces para facilitar la secuencia de pantallas en cada uno de los módulos, con la integración de las nuevas funcionalidades permitirá que el trabajo de la evaluación sea más eficiente, con una respuesta rápida para obtener un informe del cumplimento y desempeño de cada uno de los docentes.

Objetivos

Objetivo General

Refactorizar la aplicación web de seguimiento a docentes con miras a la creación de una suite de herramientas informáticas de apoyo a la gestión de las Carreras de Ingeniería de Sistemas y Ciencias de la Computación.

Objetivos Específicos

Analizar el código de la aplicación de seguimiento a docentes para incorporar nuevas capacidades enfocadas en la integración.

Reconstruir y probar adecuadamente la aplicación con enfoque en la arquitectura, de modo que se facilite la integración y la administración del sistema (suite).

Incorporar nuevas funcionalidades de acuerdo a los requerimientos actuales del proceso de seguimiento de cumplimiento de tareas de los docentes de las Carreras de Ingeniería de Sistemas y Ciencias de la Computación.

Marco Metodológico

Para la refactorización de la aplicación anterior se asiste a una serie de reuniones con el Director de Carrera y los jefes de áreas para determinar el proceso de evaluación de docentes y registrar todos los requerimientos.

El marco de trabajo Scrum se lleva a cabo mediante la asignación del equipo de trabajo con sus diferentes funciones como se muestra a continuación:

Scrum Master: Su función en el equipo de Scrum es revisar avances del proyecto y ser guía en el proceso

- Tutor del proyecto de titulación

Product Owner: Su función en el equipo de Scrum es asegurar que el software aporte valor al proceso de evaluación a docentes.

- Director de carrera, jefe de Área y coordinadores

Scrum Team: Su función en el equipo de Scrum es refactorizar la aplicación anterior a una versión mejorada

- Los autores de titulación

Para seguir con la estructura del marco de trabajo seleccionado se crea el Sprint Backlog (Anexo A), para hacer un seguimiento de todas estas tareas en el tiempo determinado se usa la herramienta JIRA, ya que permite listar todas las tareas que se realizan ubicadas en la parte del backlog como se muestra en la figura 1 la última parte del Sprint Backlog en

donde constan los últimos módulos del software que se construyeron y las pruebas que se ejecutaron.

Backlog de Seguimiento a Docentes

Figura 1 Tablero Scrum
Elaborado por: Carolina Cutiupala y Daniel Toinga

También se utiliza el tablero Kanban, en el cual se ve las tarjetas con todas las tareas asignadas de una forma visual como se muestra en la figura 2 el cual está dividido en tres fases que son: por hacer, en curso y listo en los que se pueden ubicar las diferentes tarjetas de tareas creadas anteriormente con su respectivo código para que todos los miembros del equipo tengan algo que hacer y puedan ver que tareas ya están finalizadas y seguir con las que encuentran por hacer.

Tarjetas de Tareas en el Tablero Kanban

Figura 2 Tablero Kanban
Elaborado por: Carolina Cutiupala y Daniel Toinga

Luego se realiza la proyección del tiempo asignado para cada tarea en el cual se muestra el seguimiento de la duración del proyecto, como se muestra en la figura 3 en el eje vertical se encuentran una lista de tareas y en el eje horizontal se encuentra una serie de tiempo (fechas) establecidas para la implementación para coordinar la duración de las actividades que se van a realizar en el proyecto.

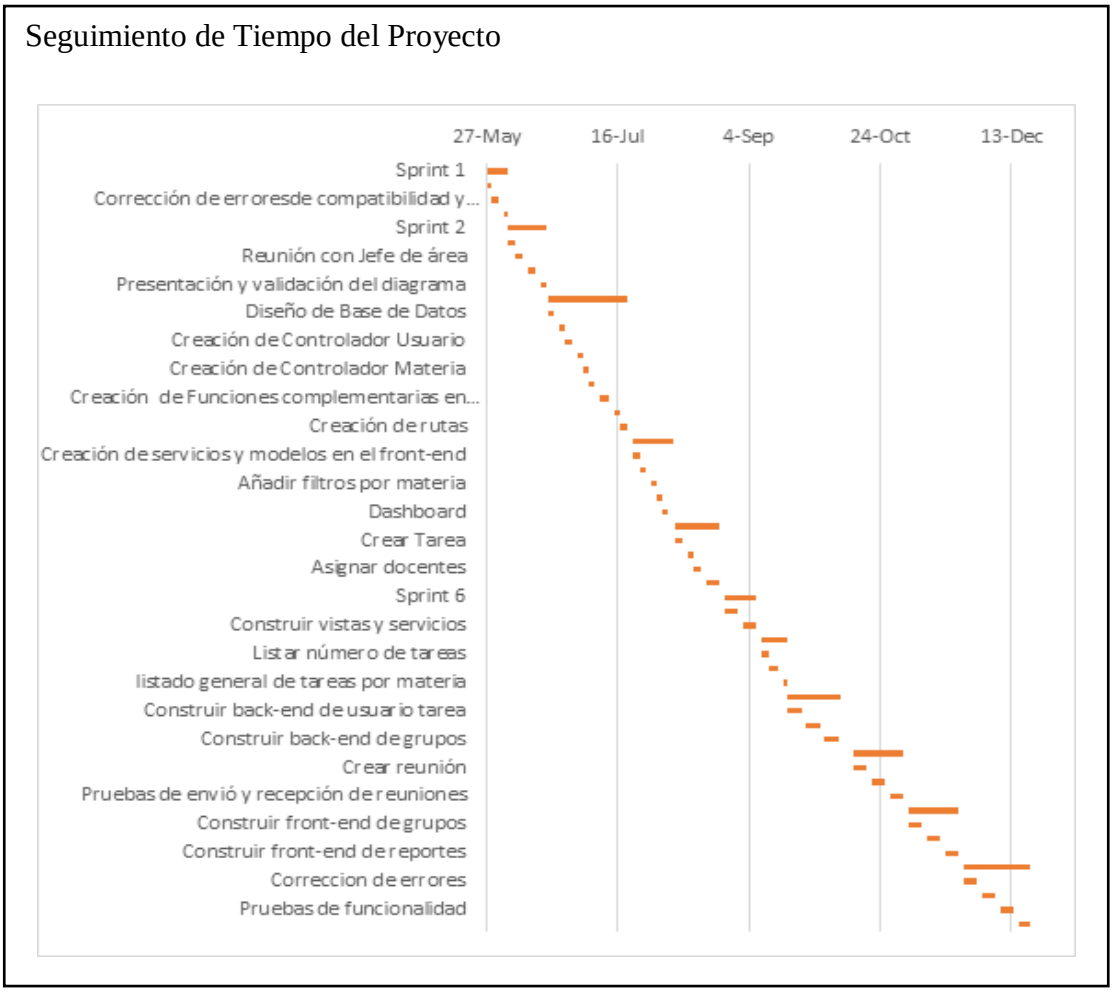


Figura 3 Diagrama gráfico de barras horizontales
Elaborado por: Carolina Cutiupala y Daniel Toinga

Al implementar el marco de trabajo se realizan ciertos cambios por ejemplo no se pude realizar reuniones diarias (Scrum diario) debido a que los docentes tenían su tiempo ya organizado y limitado.

Por ello se llevaron a cabo reuniones semanales donde se revisaba los avances, o se hacia la revisión del sprint.

Tanto la revisión del sprint como la retrospectiva del sprint, solo se realizaron entre el Scrum Master y el Scrum Team. Las revisiones con los Product Owner se realizaron por separado en distintos horarios.

Capítulo 1

1. Marco Teórico

1.1. Evaluación del desempeño docente

La evaluación permite tener una idea clara de cómo está trabajando el docente frente a sus alumnos y sus funciones asignadas por sus jefes superiores, mediante un proceso de calificación que puede ser tanto cualitativo como cuantitativo.

Existen dos objetivos principales para realizar la evaluación del desempeño entre ellos tenemos:

- Garantizar que los docentes se desempeñen de manera eficiente para beneficiar el aprendizaje de los estudiantes.
- El perfeccionamiento de la práctica docente por medio de la identificación de sus fortalezas y deficiencias, con el propósito de una formación constante. (Lejá, 2018)

Estos objetivos permiten tener una referencia de cómo trabaja el docente y cuáles serían sus fortalezas y debilidades ante los objetivos plateados por la institución.

1.2. Métodos de Evaluación del Desempeño

Los métodos de evaluación son muy importantes porque permite obtener resultados sobre el comportamiento y desempeño de las funciones designadas a las personas para ver cuál de ellas no está cumpliendo con sus funciones y ver la forma en la que se le puede orientar.

1.2.1. Método de evaluación por puntos

Este método trata de medir el rendimiento de los empleados mediante una escala numérica basada en diferentes factores determinados por los más altos niveles jerárquicos, es un

sistema el cual evalúa el puesto con mayor rapidez y puede ser menos manipulables que los otros métodos de evaluación.

Según el autor Valera “El valor de puntos que se le asigna a cada puesto representa a la suma de valores numéricos de grados para cada factor compensable que posee el puesto.” (Chamorro, 2011)

1.3. Método de evaluación de 90 grados

La evaluación de 90 grados es una herramienta que analiza a una persona a nivel laboral como muestra la figura 4 el nivel jerárquico superior hacia el nivel inferior, este método no es tan bueno para aplicarlo en las empresas ya que no genera un valor cuantitativo sino que se relaciona más factores subjetivos.

Esta evaluación en ocasiones provoca discordia porque si un trabajador no se lleva bien con algún miembro que esta evaluado puede llegar a obtener una puntuación muy baja a pesar de su buen desempeño en el trabajo ya que el resultado viene de la opinión de una persona.

Figura 4 Formato de evaluación de desempeño
Fuente: Marcela Hurtado González, Instructivo De Evaluación 90 Grados

1.4. Refactorización del Software

Para Martin Flower refactorizar "Es una técnica disciplinada para reestructurar un cuerpo de código existente, alterando su estructura interna sin cambiar su comportamiento externo" (Martin Fowler, 2018) y es así porque al aplicar una refactorización se puede obtener una mejora de mejorar su diseño después de que el código fue reorganizado esto no afecta directamente a su funcionalidad sino cambia su estructura para ser mantenible.

1.4.1. Refactorización con perspectiva al código

"La refactorización del código fuente es una de las técnicas utilizadas para intentar mejorar la mantenibilidad." (Emanuel Irrazábal ,Cristina Greiner,Gladys Daposo, 2015) El código es modificado para cumplir con los objetivos planteados agregándole un valor de mantenibilidad para posibles cambios de versión durante su funcionamiento.

Los principales objetivos para realizar refactorización en el código son:

- Limpiar del código y mejorarlo
- Corregir el código para añadir funcionalidades.
- Eliminar el código "muerto" y redundante para modularizar.
- Facilitar el futuro un mantenimiento y modificación sin problemas.

1.4.2. Refactorización con perspectiva en la arquitectura

El objetivo de la refactorización en la arquitectura del software es el apoyo que se le da ésta, porque una vez que se ha diseñado la mejor solución, se codifica y luego se refactoriza. Si no se tiene la idea de refactorización en el diseño de la arquitectura existe mucha presión para conseguir que el diseño inicial sea el correcto. (Alonso & Klaver, 2015)

Al refactorizar desde su arquitectura se crea un software compatible para agregar nuevos servicios a cada uno de los módulos existentes.

1.5. Scrum

Según Jeff Sutherland “Scrum incorpora los conceptos de mejora continua y productos mínimos viables para obtener realimentación inmediata de los consumidores, en lugar de esperar hasta que concluya el proyecto” (Sutherland, 2016) este enfoque muestra una idea clara de cómo las metodologías tradicionales pueden ocasionar problemas al final del proyecto ya que no existe que la posibilidad de interactuar con el usuario y recibir retroalimentación constante.

1.5.1. Evolución del proyecto

Scrum maneja una evolución incremental en el proyecto

- Revisión de las Iteraciones

Se realiza una revisión al finalizar cada sprint ya que esto genera un resultado aceptable o erróneo el cual se debe tratar con todo el equipo para que se mejore las funcionalidades del producto. (Palacio, 2015)

- Desarrollo incremental

Este es un desarrollo en el cual se generan varias interacciones con el usuario final durante la construcción y diseño del producto en forma continua y simultanea para ir verificando si el desarrollo se encuentra por buen camino.

- Auto-organización

"Son muchos los factores impredecibles en un proyecto. La gestión predictiva asigna al rol de gestor del proyecto la responsabilidad de su gestión y resolución." (Palacio, 2015) Scrum tiene la capacidad de auto-organizarse para tomar decisiones que se consideren oportunas.

- Colaboración

"Todos los miembros del equipo colaboran de forma abierta con los demás, según sus capacidades y no según su rol o su puesto." (Palacio, 2015)

Debido a su capacidad de auto-organización, el equipo de trabajo en Scrum conoce bien las capacidades de cada integrante, mejorando así, la colaboración entre sus integrantes, de manera que no hay una limitante en cuanto a roles, sino que todos aportan valor al producto según sus habilidades.

1.5.2. Eventos Scrum

Sprint: nombre que recibe cada iteración de desarrollo. Es el núcleo central que genera el pulso de avance por tiempos prefijados (time boxing).

Reunión de Planificación del sprint: reunión de trabajo previa al inicio de cada sprint en la que se determina cuál va a ser el objetivo del sprint y las tareas necesarias para conseguirlo.

Scrum diario: breve reunión diaria del equipo, en la que cada miembro responde a tres cuestiones:

- El trabajo realizado el día anterior.
- El que tiene previsto realizar.
- Cosas que puede necesitar o impedimentos que deben eliminarse para poder realizar el trabajo.

Cada uno de los integrantes actualiza la pila del sprint el tiempo o esfuerzo pendiente de sus tareas de esta manera se actualiza a su vez el gráfico con el que el equipo monitoriza el avance del sprint (burn-down) (Palacio, 2015)

Revisión del sprint: se realiza un análisis e inspección del incremento generado, y ajuste de la pila del producto si resulta necesario.

Retrospectiva del sprint: es una revisión de lo que sucedido durante el Sprint. La reunión entre el equipo analiza aspectos operativos de la forma de trabajo y crea un plan de mejoras para aplicar en el siguiente sprint. (Palacio, 2015)

1.6. Diagrama BPMN

"El Business Process Model and Notation (BPMN) es una notación gráfica estandarizada diseñada para representar la secuencia de actividades que conforman los procesos de negocio de una organización y los mensajes que fluyen entre los participantes y cada una de las actividades". (Chakray , 2017)

Los diagramas BPMN facilitan la interpretación de un proceso ya que lo representan gráficamente.

Los objetivos del BPM son:

- Lograr y mejorar la agilidad de negocio brindando una mayor capacidad a la organización para que pueda adaptarse a los diferentes cambios del entorno a través de los procesos integrados.
- Conseguir mayor eficacia mejorando la capacidad de una organización para lograr los objetivos estratégicos o de negocio planteados.
- Mejorar los niveles de eficiencia aumentando el grado de productividad tanto en calidad, costos y tiempos.

1.7. JIRA

"Jira Software está diseñado para que todos los miembros de tu equipo de software puedan planificar, supervisar y publicar un magnífico software." (Atlassian, 2020)

Esta herramienta de planificación para proyectos ágiles está compuesta de dos tableros: Scrum y Kanban ya que permite planificar de forma flexible también gestiona y brinda seguimiento de todas las tareas que se deben cumplir, genera estimaciones precisas para que el equipo trabaje con eficiencia y más presión.

1.8. Dia

La aplicación Dia forma parte de GNOME la que permite crear diagramas en forma modular con diferentes paquetes de acuerdo con la necesidad del usuario, "El formato que usa Dia para importar y exportar datos es XML, el cual se comprime en un archivo de formato gzip para generar un ahorro de espacio significativo en el caso de trabajar con muchos proyectos." (Noel, 2016), su interfaz y características son parecidas al programa

Visio de Windows lo que lo hace especial es que se lo actualiza periódicamente para incluir mejoras, su primera versión salió el año 2004.

1.9. AdobeXD

"Adobe XD: esta herramienta va a permitir diseñar las pantallas de lo que se pretende realizar como aplicación móvil." (Carlos Loor Loor,Javier Oyola Estrad,Nixon, Quezada Sanmartin, Geovanny Mocha Guacho, 2019)

"XD cuenta con una plataforma abierta, de modo que se integra perfectamente en las herramientas que más te gustan. También puedes acceder a cientos de complementos o crear el tuyo propio para ampliar el poder de la aplicación." (Adobe, 2020)

Esta herramienta permite crear maquetas funcionales de una aplicación, de esta manera se puede crear modelos más fieles al producto final y que el usuario pueda ver su funcionalidad, y todo esto sin escribir ni una línea de código.

1.10. Lucidchart

Es una herramienta que se utiliza para la creación de diagramas de flujo o de diseños de pantallas para aplicaciones, obteniendo una representación gráfica de todo el proceso que se debe seguir y permite trabajar de forma colaborativa con el equipo de trabajo.

"Lucidchart es un espacio de trabajo visual que combina diagramas, visualización de datos y colaboración para acelerar el entendimiento e impulsar la innovación." (Software, 2020)

1.11. Robo 3T

"También conocido como Robomongo, Robo 3T es un gestor de bases de datos para MongoDB que nació en 2013 y que el 14 de marzo de 2017 fue adquirido por la empresa 3T Software." (Iglesias, 2019)

Es un software que nos da una interfaz gráfica amigable para gestionar la base de datos de MongoDB, ya que de otro modo se tendría que ejecutar consultas manualmente por consola.

1.12. Postman

"Postman es una herramienta que se utiliza, sobre todo, para el testing de API REST, aunque también admite otras funcionalidades que se salen de lo que engloba el testing de este tipo de sistemas." (Lopez, 2019)

Es una herramienta que facilita la construcción de API REST ya que elimina la necesidad de crear un front-end para que sea el cliente, además permite guardar todas peticiones, compartirlas con otro usuario de Postman, y permite ver el código de la petición para usarlo en nuestro front-end.

1.13. Arquitectura de software

El objetivo de seleccionar un tipo de arquitectura en el software es descomponer un sistema en partes y mantener una relación entre estas partes. Dependiendo de los requerimientos del software como seguridad, mantenibilidad, rendimiento, reusabilidad, la elección de la arquitectura afectara el comportamiento del sistema y del desarrollo de este.

"La arquitectura es, más o menos, un particionado prudente del todo, en partes, con relaciones especificas entre las partes" (Clements, Garlan, Little, Nord, & Stafford, 2011)

1.13.1. Arquitectura cliente/servidor

Esta arquitectura separa al software en dos, el cliente, que es la que consume ciertos datos o funciones, y el servidor, que es quien provee esos datos o funciones. Esta arquitectura

es útil para separar la carga de procesos ya que, por ejemplo, el servidor se encarga de obtener los datos de la base de datos y enviarlos hacia el cliente, quien se encarga de procesarlos y presentarlos al usuario, lo cual mejora el rendimiento general de la aplicación ya que permite mantener varios clientes a la vez. (Stephens, 2015)

1.13.2. Arquitectura orientada a la web

La arquitectura orientada a servicios o SOA (Service Oriented Architecture) es un tipo de arquitectura que permite que los componentes del sistema trabajen de manera independiente y estén físicamente separados. La particularidad de los servicios es que presentan una manera estándar de acceder a diferentes recursos, ya sea de almacenamiento, de datos o de procesamiento, para que otros programas puedan consumir estos recursos independientemente del lenguaje de programación o la plataforma sobre la que se ejecuten. (Sommerville, 2011)

1.14. MongoDB

"MongoDB es una base de datos distribuida, basada en documentos y de uso general que ha sido diseñada para desarrolladores de aplicaciones modernas y para la era de la nube. Ninguna otra ofrece un nivel de productividad de uso tan alto" (MongoDB, 2020)

MongoDB es una base de datos NoSQL, esto quiere decir que no maneja el concepto de tablas y relaciones, en lugar de esto MongoDB almacena datos en una estructura de documentos. Estos documentos pueden tener varios pares de claves/valor como se manejaría en cualquier objeto creado en lenguaje de programación JavaScript. Los beneficios de utilizar una base de datos NoSQL son que el rendimiento es superior, es más escalable, los datos pueden ser de tipo estructurados, semi estructurados y no estructurados.

En una base de datos SQL se necesita saber de antemano el esquema de base de datos que se va a utilizar, ya que se construye en base a tablas, claves y relaciones. En este tipo de bases de datos cuando se necesite hacer cambios o agregar una nueva característica al software, generalmente se requiere hacer un cambio en el esquema de base de datos, y luego migrar todos los datos que se tenían almacenados, al nuevo esquema, en una base de datos grande esto implicaría un periodo de inactividad elevado.

Las bases de datos NoSQL están hechas para construirse sin un esquema predefinido, esto permite que se puedan hacer cambios fuertes en la aplicación sin interrumpir el servicio, facilitando el desarrollo y reduciendo el tiempo de administración de la base de datos (MongoDB, 2020)

1.15. NodeJs

"Concebido como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js está diseñado para construir aplicaciones en red escalables." (NodeJs, 2019)

NodeJs permite que ahora el código JavaScript se pueda correr en el servidor, esto lo hace usando el Motor V8 de Google, que es el mismo entorno de ejecución que utiliza el navegador Google Chrome (Kiessling, 2015)

Debido a que utiliza este entorno de ejecución de Chrome, NodeJs no maneja la concurrencia usando hilos del sistema operativo y es un entorno sin bloqueos, lo que hace que NodeJs sea ideal para construir sistemas escalables (NodeJs, 2019)

1.16. Express

"Express es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles" (Express, 2020)

Express es el framework para crear aplicaciones con NodeJs, de esta manera se puede dar más funcionalidad y facilitar el desarrollo de aplicaciones con NodeJs. Además, es de código abierto y dispone de una amplia gama de paquetes y librerías listas para usar (Brown, 2014)

1.17. Angular

Angular es un framework para JavaScript para desarrollar aplicaciones multiplataforma. No es lo mismo que AngularJs ya que este nombre se refiere a la primera versión de Angular 1.x la cual usaba JavaScript como lenguaje de programación de ahí el "js" al final, esta es una versión superior de angular la cual usa TypeScript como lenguaje de programación, por esto se quitó el "js" del nombre y optaron por un sistema de versionamiento numérico, la última versión a la fecha de redactar este documento es Angular 8. (Murray, Coury, Lerner, & Taborda, 2018)

Angular utiliza TypeScript como lenguaje de programación ya que este otorga más posibilidades a JavaScript permitiendo el uso de un tipado estático, además que permite las ayudas al momento de escribir el código. Aun así, el código JavaScript dentro de los archivos TypeScript es totalmente valido ya que, para llevar una aplicación Angular a producción, todo el código TypeScript es transpilado a JavaScript ya que es el lenguaje que interpreta el navegador (Alvarez & Basalo, Manual de Angular, 2018)

1.18. Patrones de diseño

1.18.1. Patrones de diseño creacionales

Se centran en manejar los métodos de creación de objetos de manera que se reduce la complejidad en el proceso de creación ya que los objetos se crean adaptándose a la situación en la que se trabaja (Alban Soliz, 2018)

Patrón Constructor

Permite la creación de objetos separándolos de su representación, este patrón fue creado con el objetivo de solucionar el problema de la formación de un constructor telescópico, que es lo que sucede cuando el constructor tiene demasiadas combinaciones de parámetros, que son como usar muchos constructores. Para esto el patrón constructor utiliza un objeto constructor, que es el que se encarga de recibir los parámetros de inicialización y luego devolver el objeto construido (Alban Soliz, 2018)

Patrón prototipo

Este patrón se basa en el hecho de que es más rápido clonar un objeto que crear uno desde cero, lo que hace es crear un objeto, u objetos, a partir de otros objetos que ya existen (Gamma, Helm, Johnson, & Vlissides, 1997)

Patrón Singleton

Crea una sola instancia de una clase, esto sirve para crear un acceso global a una clase. En este tipo de patrones la misma clase es la que se encarga de controlar que solo una instancia de ella sea creada (Gamma, Helm, Johnson, & Vlissides, 1997)

1.18.2. Patrones de diseño estructural

Este tipo de patrones se encargan de formar relaciones entre los objetos de tal forma que cuando se cambia una parte del sistema no sea necesario cambiar toda la estructura también (Osmani, 2017)

Patrón decorador

Este patrón tiene el objetivo de extender la funcionalidad de un objeto. Esto es útil cuando se tiene una gran cantidad de tipos de objetos, Decorador nos permite reutilizar el código para crear constructores de objetos diferentes sin necesidad de crear subclases para cada objeto (Osmani, 2017)

Patrón adaptador

Este tipo de patrón es útil sobre todo cuando tenemos que implementar software de terceros como puede ser una librería o un paquete, en nuestra aplicación. Este patrón funciona cambiando el nivel de abstracción del código (Timms, 2014)

Patrón puente

Se puede decir que el patrón puente lleva el patrón adaptador a otro nivel ya que, provee una interface en la que podemos implementar múltiples adaptadores (Timms, 2014).

1.18.3. Patrones de diseño de comportamiento

Se enfocan en cómo se comunican los objetos y en la manera de cómo estos objetos pueden trabajar juntos (Osmani, 2017)

Patrón iterador

El patrón iterador permite seleccionar secuencialmente cada ítem de un array o colección.

Es especialmente útil cuando se necesita procesar ítems en un array (Timms, 2014)

Patrón observador

Este patrón se compone de dos tipos de objetos, el sujeto y los observadores. Esto permite que distintos observadores se suscriban a las notificaciones de cambio del sujeto (Osmani, 2017)

Patrón publicación/suscripción

Es muy común confundir este patrón con el anteriormente mencionado patrón observador, ya que el comportamiento es similar, pero hay una diferencia que es necesario señalar.

El patrón observador requiere que un objeto observador se suscriba al sujeto del que desea recibir notificaciones, por otro lado, el patrón publicación/suscripción utiliza un canal de comunicación entre los suscriptores y el sujeto que dispara el evento. Esto evita que haya dependencia entre los observadores y el sujeto (Osmani, 2017).

1.19. MEAN Stack

Un stack es un conjunto de algo, por lo tanto el MEAN stack como es conocido comúnmente, es el conjunto de tecnologías para el desarrollo de aplicaciones web, su nombre es dado por las siglas de las tecnologías que usa, estas son, MongoDb, Express, Angular y NodeJs. Este estilo de programación es habitual y cada vez más reconocido debido a la facilidad que ofrece para el desarrollo, ya que, permite desarrollar aplicaciones complejas utilizando solamente un Lenguaje de programación, JavaScript.

Tanto en del lado del cliente, el servidor y la base de datos se usa la misma sintaxis, esto facilita el desarrollo y disminuye la curva de aprendizaje para desarrollar una aplicación,

ya que, no es necesario aprender un lenguaje de consultas SQL y un lenguaje adicional para back-end como PHP o Python.

Dicho esto, cabe destacar que la intención en ningún momento es restar utilidad a otros lenguajes de programación o consulta, el Stack MEAN lo que pretende es facilitar el desarrollo, al igual que lo hacen otros stacks parecidos como puede ser MERN (MongoDB, Express, React, NoeJs) o MEVN (MongoDB, Express, VueJs, NodeJs).

1.20. REST

"Representational State Transfer (REST) es una abstracción de los elementos de la arquitectura dentro de un sistema distribuido" (Fielding, 2000)

Es una manera de crear servicios web basándose principalmente en URLs que denotan un servicio o un recurso, al cual se puede acceder generalmente por medio del protocolo HTTP. REST maneja un conjunto de restricciones que no dictan la tecnología que se debe utilizar, más bien dicta las reglas que se debe seguir para transferir los datos entre los componentes.

Entre estas restricciones podemos listar:

- Sin estado: esto quiere decir que los servicios no deben guardar datos de los usuarios, cada servicio es independiente.
- Interfaz uniforme: cada recurso tiene una única URI, lo que se conoce como "endpoint" mediante la cual todos los usuarios pueden acceder.
- Sistema en capas: para mejorar la escalabilidad

- Usar hipermedios: separa los recursos de su representación, así se puede acceder por medio de diferentes formatos como HTML, XML, JSON, texto plano, etc. (Fielding, 2000)

1.20.1. Single page application(SPA)

"Una SPA es una aplicación entregada al navegador, que no necesita recargar la página durante el uso" (Mikowski & Powell, 2014)

Las SPA son entregadas desde el servidor al navegador para ser utilizadas por este, esto reduce el tiempo de espera que tomaba en recargar la siguiente página cuando se realiza una actividad en un software, ya que el servidor no necesita enviar cada página con datos al cliente, sino solo se enfoca en enviar los datos y el cliente (navegador) se encarga de presentarlos. Las SPAs se enfocan en mejorar la experiencia del usuario en el software reduciendo los tiempos de carga y aumentando la fluidez de la aplicación (Mikowski & Powell, 2014)

1.21. Git

Git es un sistema de control de versiones, este tipo de sistemas se encarga de controlar los cambios en el desarrollo de un software, permitiendo llevar un registro de los cambios realizados, quienes realizaron los cambios y cuando lo hicieron. Es útil cuando se trabaja en equipo ya que así se mantiene un registro del estado de la aplicación, permitiendo restaurar un estado anterior en el caso de que se genere un error (Alvarez, Alcazar, & León, Manual de Git, 2018)

1.22. Github

“Github es un servicio para alojamiento de repositorios de software gestionados por el sistema de control de versiones Git” (Alvarez, Alcazar, & León, Manual de Git, 2018)

Github proporciona una interfaz gráfica para poder gestionar nuestros repositorios, facilita la visualización de los cambios en el código, y facilita también el seguimiento de los desarrolladores que trabajan en el mismo repositorio.

1.23. Jmeter

Es una herramienta para realizar pruebas de carga de servidores en forma automática, “en Jmeter todo comienza por lo que se ha llamado Plan de pruebas” (Medina, 2014) también está conformado por un grupo de hitos el cual es un elemento obligatorio que se encarga de simular los usuarios recurrentes para efectuar las peticiones.

“Es un software de código abierto, una aplicación 100% Java diseñada para cargar pruebas del comportamiento funcional y medir el rendimiento” (The Apache Software Fundation, 1999-2019)

Al principio esta herramienta fue diseñada para pruebas de estrés en aplicaciones Web, pero hoy en día evoluciono en su arquitectura no sólo para llevar a cabo las pruebas en componentes habilitados en Internet (HTTP), sino además en Bases de Datos, programas en Perl, con solicitud FTP y prácticamente en cualquier otro medio.

1.24. Pruebas del sistema

Las pruebas de software ayudan a explorar, conocer y entender el producto que se está desarrollando, este proceso permite reducir la cantidad de errores al momento de presentar el software final a los usuarios.

1.24.1. Tipos de pruebas de software

Existen diferentes tipos de pruebas que se aplican para comprobar la calidad y la eficiencia del software entre ellas tenemos:

Pruebas funcionales

Este tipo de pruebas permite comprobar que todas las funcionales estén de acuerdo con los requerimientos del usuario y que se encuentren implementadas correctamente en el software.

Pruebas de aceptación

Su objetivo principal es enfocarse en la validación basada en el cumplimento de los requerimientos establecidos anteriormente por el cliente, el cual acepta y aprueba la funcionalidad del software.

Pruebas de rendimiento y carga

Este tipo de pruebas permiten validar la respuesta de una aplicación cuando es sometida a una carga de usuarios, transacciones o procesos simultáneamente, tratando de simular un ambiente de real de producción.

Capítulo 2

2. Análisis y diseño

2.1. Análisis de funciones y roles

El análisis comienza con la revisión del organigrama estructural de la Universidad Politécnica Salesiana, en este diagrama se identifica los actores, las áreas y la jerarquía de cargos que existen en esta organización.

Como se puede ver en la Figura 5 que se ubica al Vicerrector de sede como la máxima autoridad en el nivel superior, Consejo de carrera que se encarga de generar las propuestas y hacer cumplir para la aporbación universitaria , Director de Carrera crea reuniones, tareas y comunica las propuestas de nivel anterior, Jefes de Área oraganizan a su equipo y distribuir las tareas y al final los docentes los cuales se encargan de cumplir con las tareas asignadas.

Figura 5 Organigrama jerárquico de sede de UPS sur.
Elaborado por: Carolina Cutiupala y Daniel Toinga

Para definir bien las funciones y roles de las personas se identifica en los artículos del Reglamento Interno de la Universidad Politécnica Salesiana (Anexo B), en el que se seleccionó el artículo 18 y 21 ya que contienen los cargos y actividades que cumplen tanto el Director de Carrera y del Jefe de Área que se desglosan en la siguiente tabla:

Tabla 1 Roles y Funciones

ROL	FUNCIONES
Director de Carrera	• Evaluar a los docentes periódicamente • Planificar reuniones • Asignar tareas • Delegar responsabilidades
Jefe de Área	• Ayudar con el Director de Carrera en el sistema de evaluación • Planificar sus actividades y cumplir en el tiempo asignado • Dar seguimiento a los profesores en sus actividades designadas • Presentar reportes con las actividades realizadas por cada uno de sus integrantes.

Nota: Esta tabla contiene las funciones y roles

Con estas dos revisiones se pudo establecer el cargo y la función que debería cumplir cada una de las personas que van a participar en la refactorización de la aplicación anterior.

2.2. Análisis del proceso global de seguimiento a docentes

Mediante algunas reuniones con el Director de Carrera y los Jefes de Área se identifica el proceso de seguimiento a docentes, ya que no se contaba con un proceso documentado, pero se realiza una serie de entrevistas a las personas anteriormente nombradas las cuales resolvieron la dudas de cómo funciona el proceso actual porque lo tenían muy claro cómo funcionaba.

En base al análisis del proceso de evaluación del desempeño se concluye que se utiliza un método de evaluación de 90 grados en el cual los evaluadores utilizan criterios subjetivos para calificar el desempeño de los docentes.

El diagrama BPMN fue generado con la herramienta Bizagi que permita crear un flujo para entender mejor la lógica del negocio en el cual se estaba trabajando.

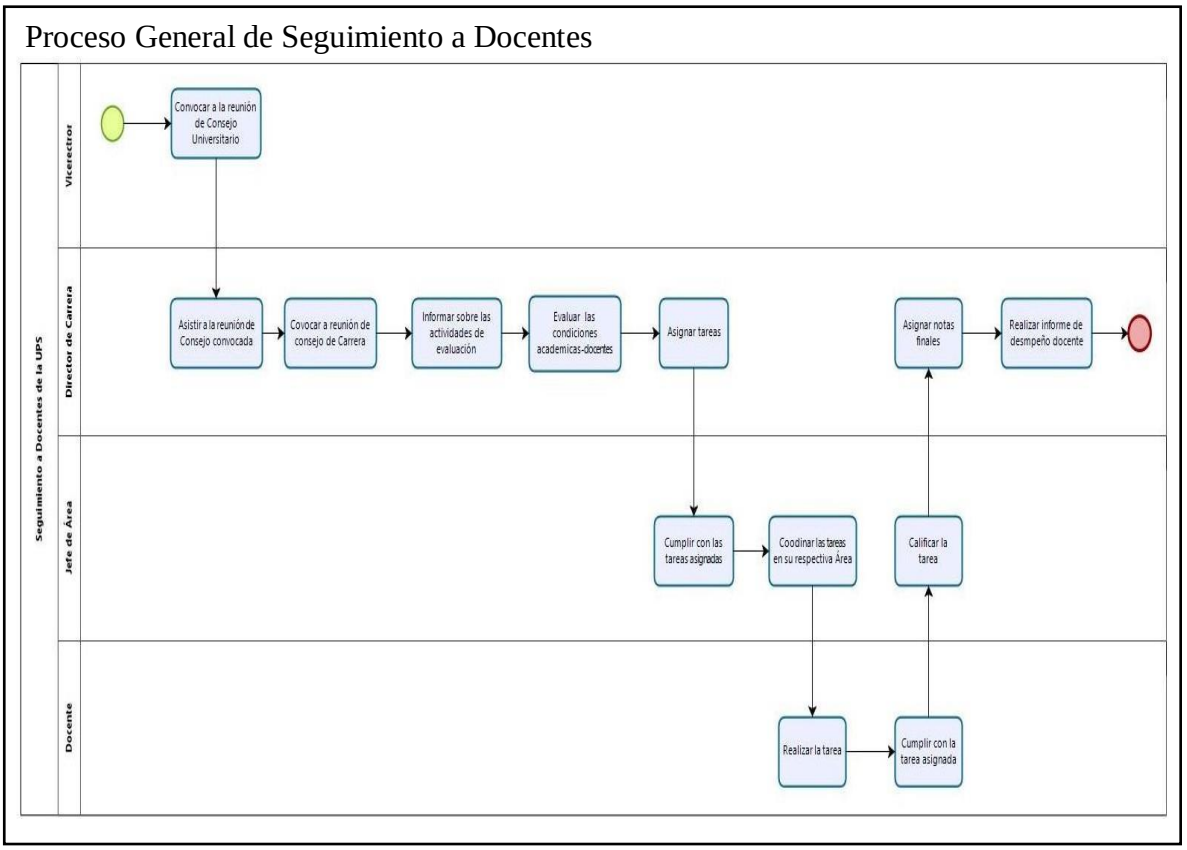


Figura 6 Diagrama BPMN del Proceso de Seguimiento a Docente
Elaborado por: Carolina Cutiupala y Daniel Toinga

En la figura 6 se puede evidenciar que tanto el Director de Carrera como los Jefes de Área asignan tareas a los docentes, pero se debe tener en cuenta que todos cumplen con el rol de Docente, por lo tanto Director de Carrera y Jefes de Área deberán también realizar las tareas. De estas tareas realizadas el Director de Carrera realizara el informe de desempeño docente con la calificación final de cada docente.

2.3. Análisis de la aplicación a refactorizar

En esta sección se realiza todo el proceso de análisis completo de la aplicación para entender todo lo posible sobre la aplicación anterior ya que es el primer paso para realizar la refactorización.

2.3.1. Análisis de la documentación de los Casos de Uso

Los siguientes casos de uso se analizaron de este modo:

- Se identifica a todos los actores que participaban en la aplicación a refactorizar
- Se ejecuta el programa para comprobar que los requisitos existan en el sistema.
- Se verifica el flujo básico de cada uno de los casos de usos.

Como se muestra en la figura 7 el diagrama de administrar tarea aquí el actor es el jefe de área ingresa al sistema, crea la tarea, revisa la tarea y al final califica la tarea asignada a los docentes.

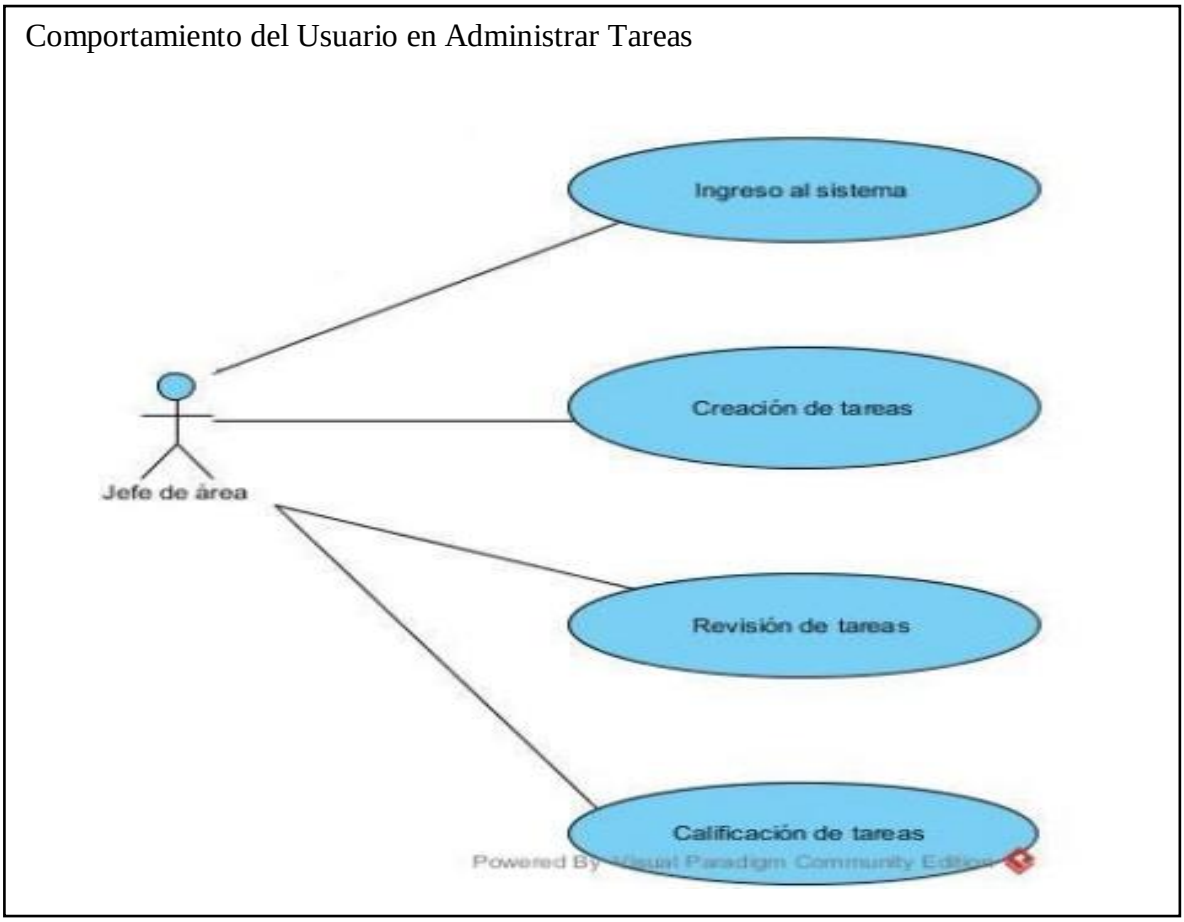


Figura 7 Caso de uso administrar tareas
Fuente: (Mora, 2017)

Como se muestra en la figura 8 el diagrama de registro de reuniones con su actor jefe de área ingresa al sistema luego se ubica en el módulo de tareas después selecciona el tipo de tareas, ingresa al detalle de reunión y registra la reunión planificada.

Interacción del Usuario en el Registro de Reuniones

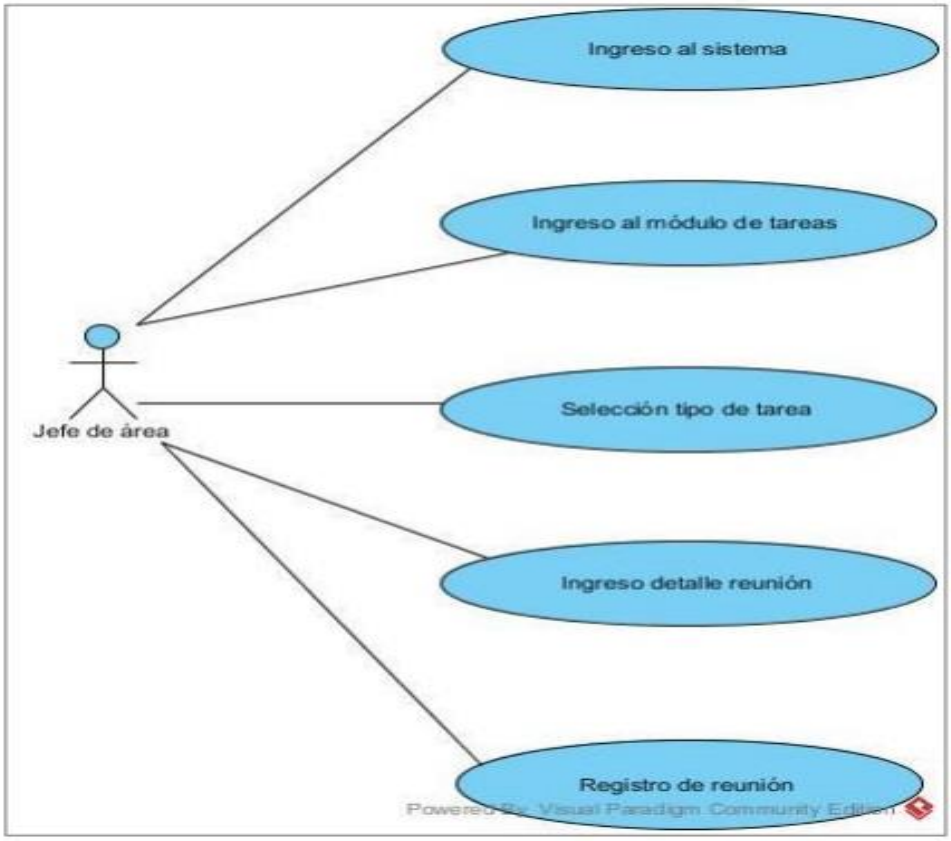


Figura 8 Caso de uso registrar reuniones
Fuente: (Mora, 2017)

Como se muestra en la figura 9 el diagrama global de envió de tareas con el actor general el cual ingresa al sistema, se dirige al módulo de tareas después selecciona una tarea y envía la tarea creada.

Interacción del Usuario Para el Envío de Tareas

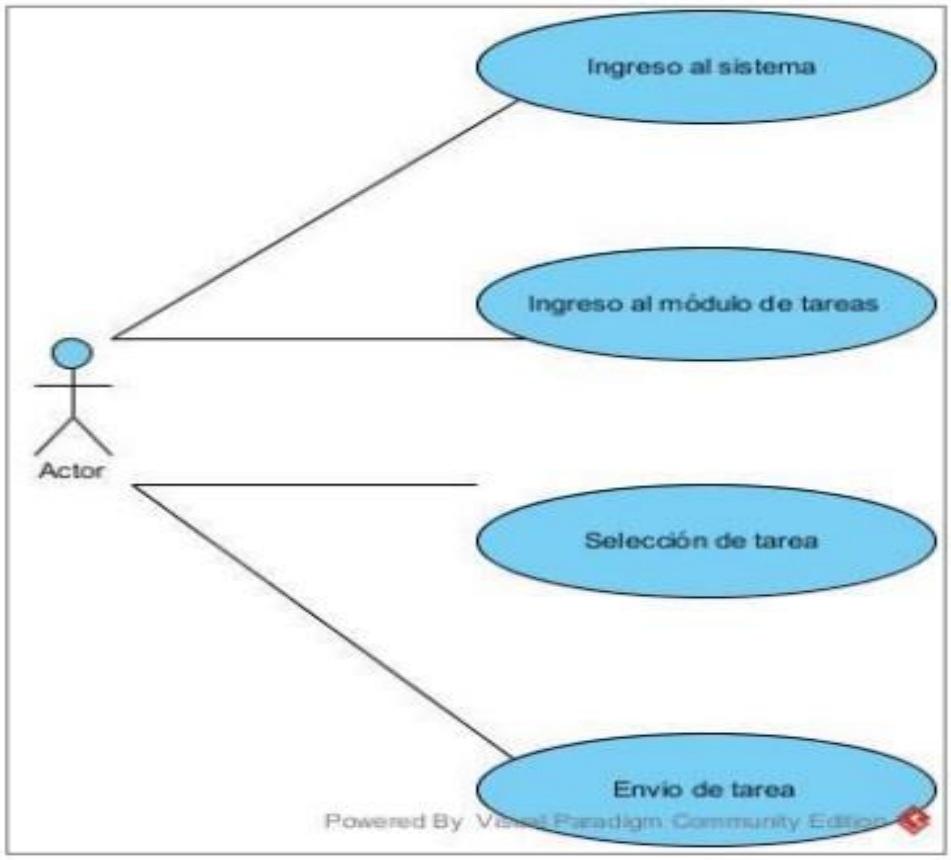


Figura 9 Caso de uso enviar tarea
Fuente: (Mora, 2017)

Se pudo notar que los casos de uso eran muy generales y su flujo tenía una secuencia muy básica solo estaban planteados con los jefes de área para arreglar ese problema definimos todos los actores que intervenían sus iteraciones para plantear ciertas modificaciones para mejorar el uso y el funcionamiento de la aplicación a refactorizar.

En la tabla 2 se definen los casos de uso que existían en la aplicación a refactorizar para tener una base y plantear las nuevas funcionalidades y donde se las debería agregar.

Tabla 2. Casos de Uso de la Aplicación Anterior

ID	CASO DE USO	DESCRIPCIÓN	ACTORES
001	Crear tareas	El jefe de área registra una tarea en el sistema	Jefe de área
002	Revisar tareas	El docente revisa se le ha asignado una tarea	Jefe de área
003	Calificar tareas	Se asigna una puntuación por las tareas realizadas	Jefe de área
004	Registrar las reuniones	Se establece una fecha y sus detalles	Jefe de área
005	Enviar tareas	El jefe de área escoge a los docentes que van a realizar las tareas asignadas	Jefe de área

Nota: Esta tabla contiene los casos de uso más utilizados en la aplicación anterior.

2.3.2. Análisis de código fuente

El análisis se realiza mediante la revisión del código fuente para identificar los diferentes paquetes, métodos y atributos implementados en el proyecto. Se comienza revisando el árbol de directorios del back-end como se muestra en la figura 10 en la cual se encuentra un directorio llamado “controllers” el cual contiene la mayoría de los controladores para las funcionalidades principales del programa, luego, se revisa el código de cada uno de los archivos contenidos en la carpeta “controllers” en busca de los métodos y funciones principales como: crear tarea, generar reportes y calificar tarea los cuales se mantendrán su lógica o se aplicara la refactorización para ser modificados.

Función Crea Tarea

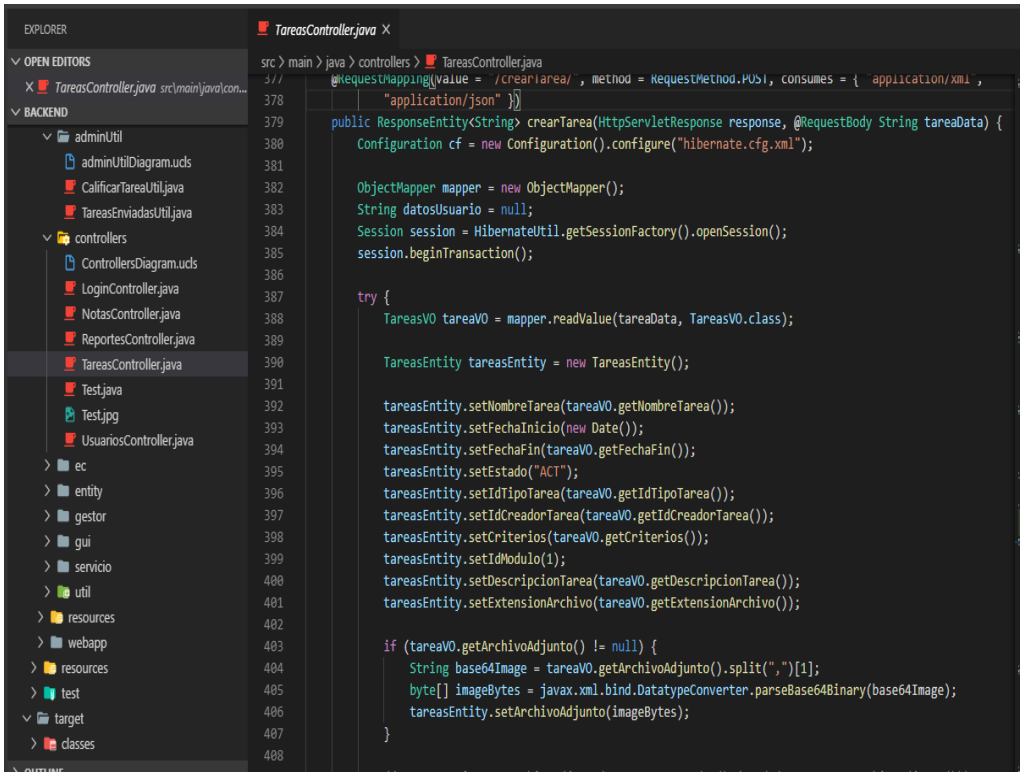


Figura 10 Código Fuente de la Función Crear Tareas
Fuente: (Mora, 2017)

2.3.3. Análisis de la base de datos

El análisis de la base de datos se encontró un modelo físico, como se muestra en la Figura 11 se describe una base de datos con un modelo relacional, donde cada una de las tablas posee una clave primaria que permite comunicarse con diferentes entidades, como se puede observar entidades como: usuarios, tareas, grupos de usuarios, módulos y perfiles pero también existen algunas tablas sueltas como las de criterios y periodos.

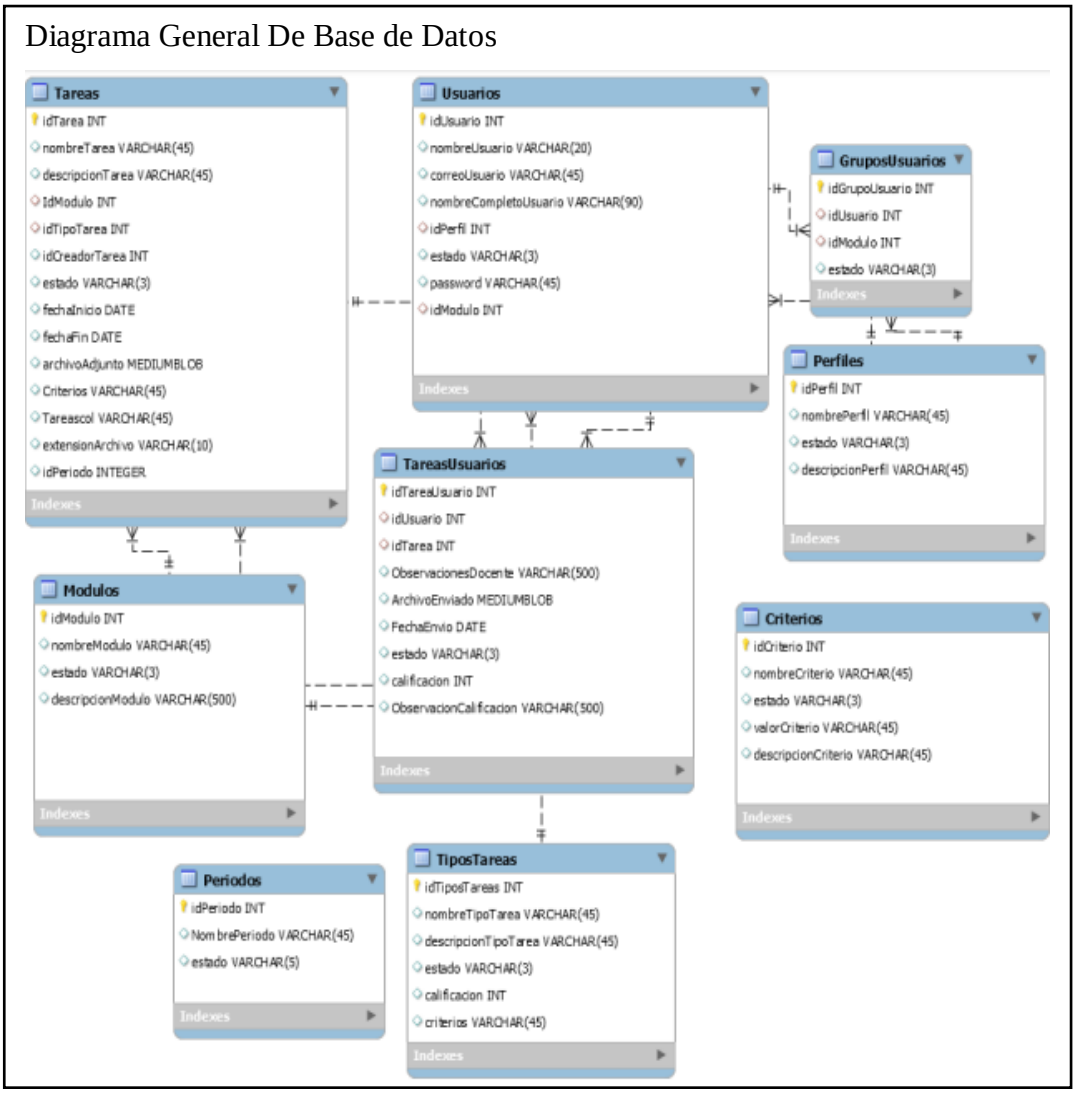


Figura 11 Base de Datos Relacional de la aplicación anterior
Fuente: (Mora, 2017)

Se realiza una comparación entre la base de datos MySQL y MongoDB con se muestra en la tabla 3 para elegir la mejor opción para cambiar o mantener de base datos de la aplicación a refactorizar.

Tabla 3 Comparación entre MySQL y MongoDB

MySQL	MongoDB
Base de datos relacional	Base de datos no relacional
Multiplataforma	
Esquema estático	Esquema dinámico
Escalabilidad vertical	Menor tiempo de recuperación ante fallas
Es una tecnología madura	Aún es una tecnología joven

Nota: Comparación de base relacional vs base no relacional

Después de analizar la tabla comparativa de las características que tienen cada una de las bases de datos, se propone un cambio de MySQL a MongoDB ya que como se pude observar es una base de datos no relacional el cual es un modelo más flexible a la hora de modificar la estructura y jerarquía de los datos, además es de gran ayuda cuando se requiere de cambios constantes en el software.

2.3.4. Revisión de requerimientos para la refactorización

Se realiza varias reuniones con las personas de rol de Director de Carrera y Jefes de Área Académico para recolectar información del proceso de evaluación y seguimiento a docentes, en este punto se hicieron ciertas validaciones de los requerimientos con lo cual se obtuvo como resultado los siguientes cambios:

- Evaluación basada en criterios.
- Categorización de tareas.
- Promedio general de docente por materia.
- Tareas de tipo con evidencia o sin evidencia.

Estos requerimientos se crearon a partir de una serie de preguntas formuladas. (Anexo C)

Se plantea una tabla categorizando todos los requerimientos que necesita la aplicación:

Tabla 4 Clasificación de Requerimientos

Requerimientos nuevos	**Requerimientos antiguos**
Evaluación por criterios y categorías configurables.	Crear tareas
Tareas con y sin evidencia	Crear reuniones
Asignar docente de apoyo	Asignar docente a tareas
Modificar tarea	Reportes por materia, docente
Crear grupos de usuarios para las tareas	Ingreso al sistema en base a credenciales creadas por el administrador
Incorporar nuevas capacidades de integración.	Criterios de evaluación fijos: cumplimiento, calidad y tiempo
Mejorar la escalabilidad y la mantenibilidad	Crear tarea con archivos adjuntos

Nota: Se lista todos los requerimientos requeridos

2.4. Diseño de la presentación de la aplicación refactorizado

2.4.1. Arquitectura

La arquitectura del sistema estará desarrollada como muestra la figura 12 con el stack MEAN (MongoDB, Express, Angular, NodeJS). La razón principal de seleccionar esta arquitectura es porque permite trabajar con un único lenguaje de programación que es JavaScript, además está basada en micro servicios, lo que facilita el desarrollo y aumenta la escalabilidad del software.

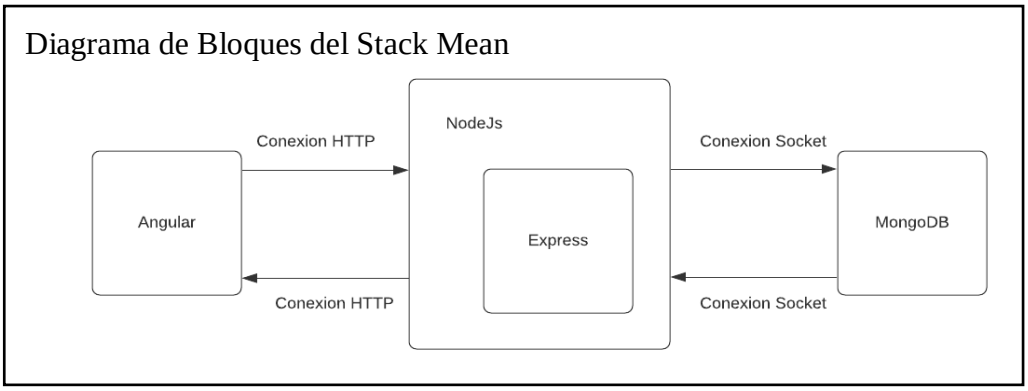


Figura 12 Arquitectura Stack MEAN
Elaborado por: Carolina Cutiupala y Daniel Toinga

2.4.2. Maquetación de Interfaces

Una vez que se entiende cómo funciona la aplicación a refactorizar se propuso una nueva interfaz que sea más amigable para los usuarios donde ellos puedan visualizar sus materias, y ver las tareas que deben cumplir para su evaluación. Para crear el diseño de las maquetas se utiliza la herramienta Adobe XD en su versión gratuita.

Se diseño 40 interfaces que permite proyectar una idea clara de la línea de diseño que debe tener la aplicación de las cuales se seleccionaron 5 maquetas que a continuación se presentan:

En la figura 13 se muestra la maqueta de la pantalla principal de un usuario con rol Director de Carrera o Jefe de Área. Esta vista está dividida en dos secciones, una para la parte administrativa y otra para la visualización de las materias asignadas al usuario.

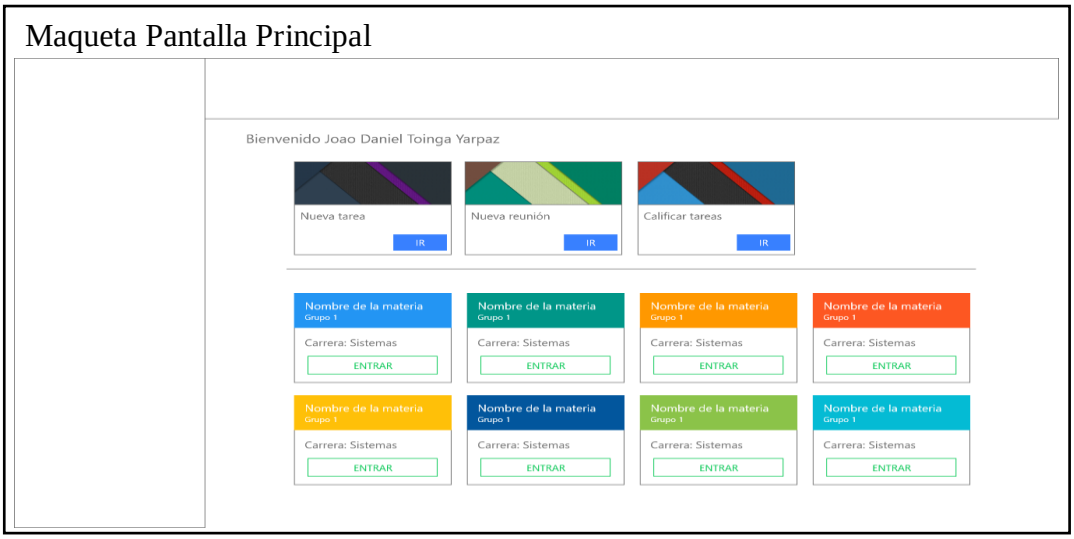


Figura 13 Nombres y etiquetas en el Dashboard
Elaborado por: Carolina Cutiupala y Daniel Toinga

En la figura 14 se muestra el dashboard de cada materia, aquí el usuario puede ver una lista de las tareas pendientes, así como un historial donde se muestra el estado de cada tarea.

El usuario también puede ver la nota global de las tareas calificadas y las reuniones pendientes.

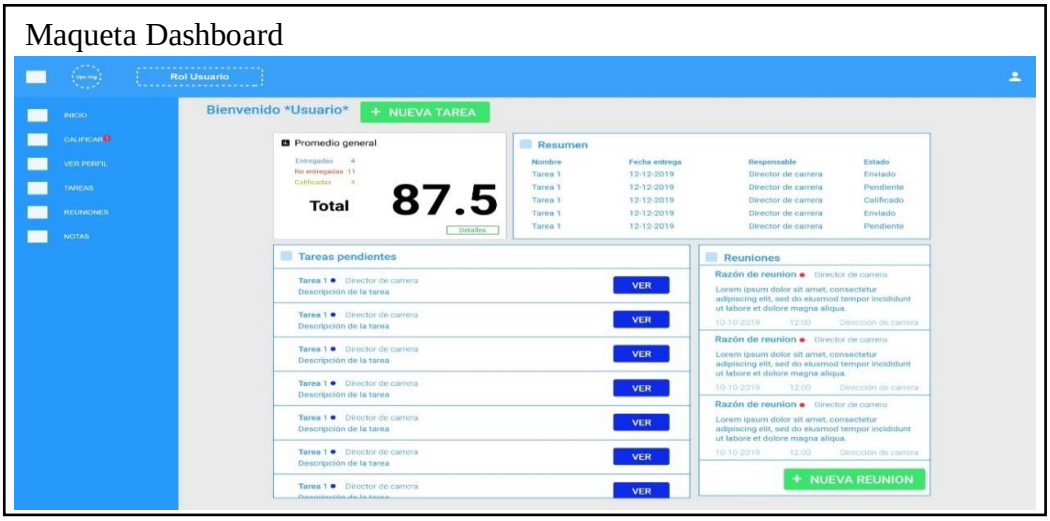


Figura 14 Pantalla Principal
Elaborado por: Carolina Cutiupala y Daniel Toinga

El proceso de crear una nueva tarea consta de tres pasos:

- Descripción de la tarea. – se ingresan los datos básicos de la tarea como, nombre, descripción, archivos adjuntos y fecha límite.
- Asignación de docentes. – La maqueta de la figura 15 muestra una lista de todos los usuarios en cada una de sus materias o gestión académica, aquí se puede seleccionar uno o varios usuarios a los cuales se les enviara la tarea creada en el punto anterior.
- Envió. – se muestra los datos de la tarea y una lista de los usuarios seleccionados, esto es para confirmar que todo es correcto y enviar la tarea a los usuarios.

Figura 15 Asignar docentes
Elaborado por: Carolina Cutiupala y Daniel Toinga

En la figura 16 se muestra la vista de calificación de una tarea se muestra, el nombre de la tarea en la parte superior, seguido por la descripción y la fecha en la que se entregó la tarea. La calificación de la tarea es basada en criterios, por lo que el usuario puede seleccionar uno o varios criterios los cuales darán la nota final de la tarea.

Figura 16 Calificar Tarea
Elaborado por: Carolina Cutiupala y Daniel Toinga

En el paso 1 del proceso de crear una nueva tarea, en la figura 17 se muestra que existe la opción de seleccionar una tarea de una lista de tareas creadas y guardadas anteriormente, esto con el propósito de utilizar tareas que asignan recurrentemente a los usuarios.

Figura 17 Crear Nueva Tarea
Elaborado por: Carolina Cutiupala y Daniel Toinga

2.4.3. Diagrama de la Base de Datos

Se utiliza MongoDB la base de datos fue creada utilizando el software Dia ya que permite hacer diagramas de bases de datos no relacionales.

Como se muestra en la figura 18 la nueva base de datos posee un esquema de colecciones organizadas con referencias en cada una de ellas para una mejor comprensión de la estructura de los datos. En este nuevo esquema se restructura dos tablas flotantes de la base original la cuales son: “Periodos” y “Criterios” ya que al ser tablas que se encuentran sueltas se dificulta la comprensión de su uso.

Además, se excluyó las tablas cuyos nombres son “Modulos”, “TiposTareas” y “Perfiles” ya que contienen datos fijos o que no se modifican recurrentemente, por lo cual se decide combinar algunos de sus atributos en las colecciones llamadas “Usuario” y “Tarea”.

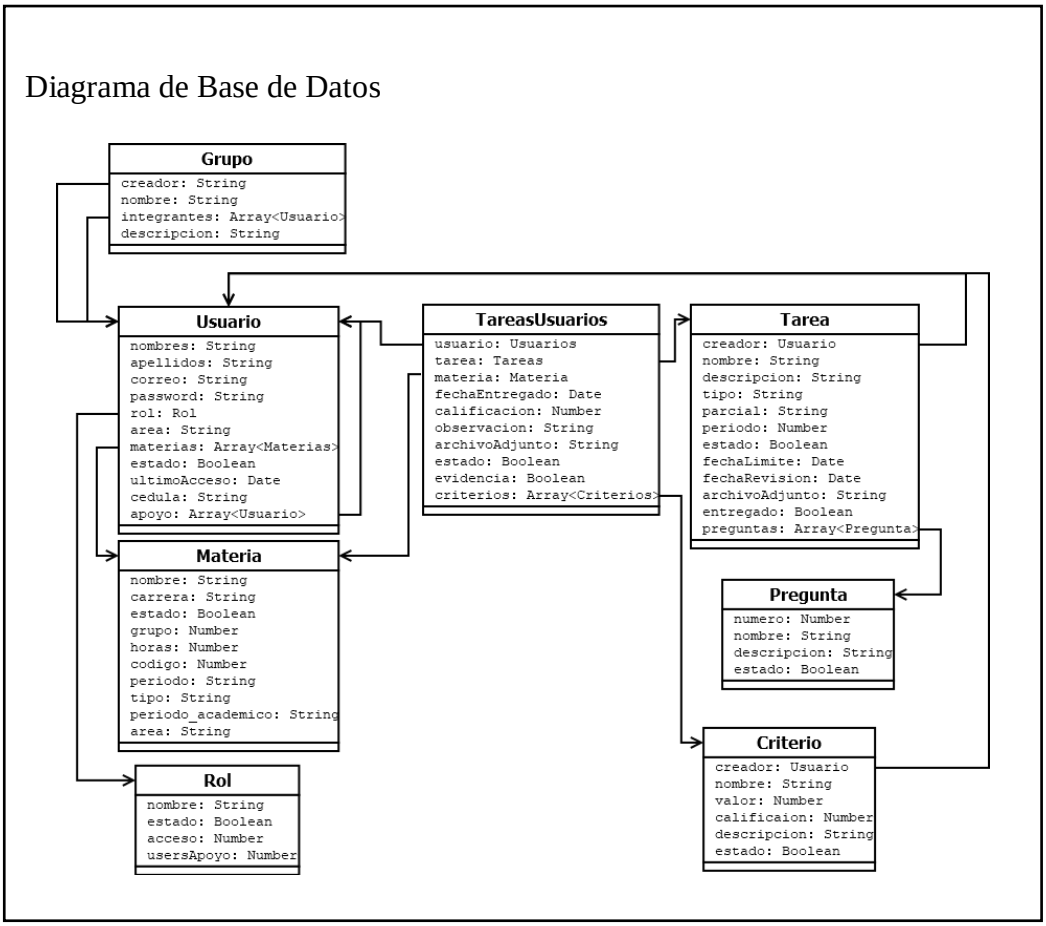


Figura 18 Diagrama de Base de Datos del Software refactorizado
Elaborado por: Carolina Cutiupala y Daniel Toinga

2.4.4. Diagrama de Componentes

Se describe el diagrama de componentes de la aplicación refactorizada como muestra la figura 19 la cual consta de seis componentes principales:

Front-end

- Módulos y componentes. - contienen toda la interfaz visual y sus controladores.
- Servicios. – se encargan de enviar y recibir los datos mediante el protocolo HTTP hacia el back-end.

Back-end:

- Router Module. - envía y recibe los datos del front-end y redirige las peticiones HTTP hacia sus correspondientes controladores.
- Controlador. – se encarga de procesar los datos enviados y recibidos de la base de datos.
- Mongoose Schemas. – es el modelo de los datos de la base de datos, se especifican atributos y tipos de datos.
- Data Base MongoDB. – base de datos no relacional.

Diagrama de componentes de la aplicación refactorizada

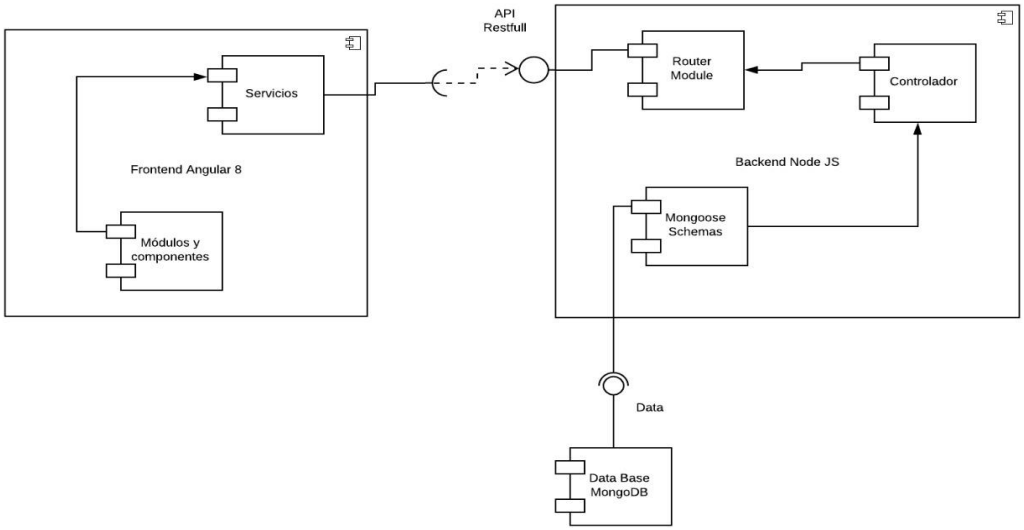


Figura 19 Diagrama de Componentes
Elaborado por: Carolina Cutiupala y Daniel Toinga

Capítulo 3

3. Refactorización y pruebas

3.1. Definición de la arquitectura

Este sistema se construye sobre la arquitectura cliente/servidor y utilizando el estilo arquitectónico REST. Una de las ventajas que ofrece este estilo de arquitectura es la escalabilidad ya que permite que cada recurso tenga un único punto de acceso. Esto conlleva la creación de una API REST en el lado del back-end la cual provee dichos puntos de acceso utilizando otro estilo de arquitectura que es de microservicios el cual mediante rutas URI hace que el sistema sea de fácil integración.

Además, se utiliza el patrón arquitectónico Modelo Vista Controlador (MVC) en cada componente de la arquitectura. En el back-end se utiliza este patrón utilizando NodeJs y Express, y en el front-end con el framework Angular 8.

3.2. Implementación de Patrones de Diseño de software

3.2.1. Implementación de patrones de diseño en el front-end

3.2.1.1. Singleton

El framework Angular 7, ya que utiliza el lenguaje de programación TypeScript, es más versátil que JavaScript para la creación de clases e interfaces, lo que permite utilizar el patrón Singleton como se muestran en las figuras 20 y 21 para mejorar el desempeño de la aplicación. Además, la estructura del framework permite crear aplicaciones SPA lo que implica que los componentes creados también pueden ser reutilizables. Se debe ser muy cuidadoso al implementar este patrón, ya que puede causar problemas en tiempo de ejecución.

Tarea de Usuario Código

evalDocentes > src > app > models > tareaUsuario.ts > ...

```typescript
import { Tarea } from './tarea';
import { User } from './user';
import { Materia } from './materia';
import { Criterio } from './criterio';
export class TareaUsuario {
    constructor(
        public usuario: User,
        public tarea: Tarea,
        public materia: Materia,
        public fechaEntregado: Date,
        public calificacion: Number,
        public observacionCalificacion: String,
        public comentario: String,
        public entregado: boolean,
        public calificado: boolean,
        public evidencia: boolean,
        public criterios: [Criterio],
        public _id?: string,
        public archivoAdjunto?: [string],
    ) { }
}
```

Figura 20 Clase Tarea Usuario
Elaborado por: Carolina Cutiupala y Daniel Toinga

Importación de la clase y componentes de Tareas Usuario

```typescript
import { Component, OnInit } from '@angular/core';
import { Location } from '@angular/common';
import { TareasUsuarioService } from '../../../services/tareas
import { ErrorDialogService } from '../../../services/error-di

import { TareaUsuario } from '../../../models/tareaUsuario';
import { CalificarTareaDialogComponent } from '../calificar-ta
import { MatDialog, MatTableDataSource } from '@angular/materi

@Component({
  selector: 'app-calificar-tarea',
  templateUrl: './calificar-tarea.component.html',
  styleUrls: ['./calificar-tarea.component.css'],
  providers: [
    TareasUsuarioService,
    ErrorDialogService
  ]
})
export class CalificarTareaComponent implements OnInit {
```

```typescript
import { saveAs } from "file-saver";
import { TareasUsuarioService } from '../../../services/tare
import { TareasService } from '../../../services/TareasServi
import { GLOBAL } from '../../../services/global';
import { ErrorDialogService } from '../../../services/error-
import { TareaUsuario } from '../../../models/tareaUsuario';

@Component({
  selector: 'app-hacer-tarea-dialog',
  templateUrl: './hacer-tarea-dialog.component.html',
  styleUrls: ['./hacer-tarea-dialog.component.css'],
  providers: [
    TareasUsuarioService,
    TareasService,
    ErrorDialogService
  ]
})
export class HacerTareaDialogComponent implements OnInit {
```

Figura 21 Importación de la clase Tarea Usuario en diferentes componentes
Elaborado por: Carolina Cutiupala y Daniel Toinga

3.2.1.2. Constructor

Cada componente de Angular tiene un constructor por defecto, esto permite inicializar los objetos o variables con datos provenientes del back-end, que se utilizaran para cargar las vistas como se muestra en la figura 22.

Patrón Constructor de Calificar Tarea

```
constructor(
  private tareasUsuarioService: TareasUsuarioService,
  public dialog: MatDialog,
  private _location: Location,
  private errorService: ErrorDialogService
) {
  this.tareasUsuarioService
  .getTareasporCalificar(localStorage.getItem('tokken')).subscribe(
    response => {
      this.tareaUser = response['tareas'];
      this.dataSource.data=response['tareas'];
    },
    error => {
      this.getSubscriptionError(error);
    }
  );
}
```

Figura 22 Patrón Constructor de la clase Calificar Tarea Component
Elaborado por: Carolina Cutiupala y Daniel Toinga

3.2.1.3. Observer

Angular utiliza este patrón en todos sus componentes principalmente para que los datos se puedan pasar a las vistas de manera sencilla como se muestran en las figuras 23 y 24 la cual su función principal es recibir los datos que envía el back-end de manera asíncrona, así se puede asegurar que, si se recibe datos o error, en ambos casos el flujo se completa y se cierra, sin bloquear la aplicación.

Implementación Patrón Observer

```
<div class="botones">
  <button mat-stroked-button (click)="onCancel()" class="cancelar" color="warn">Cancelar</button>
  <button mat-raised-button (click)="crearTarea(stepper,false)" [disabled]="!firstFormGroup.valid"
    class="siguiente" color="primary">Siguiente</button>
</div>
```

Figura 23 Patrón Observer en código HTML de la vista Crear Tarea
Elaborado por: Carolina Cutiupala y Daniel Toinga

Implementación del Patrón Observer en el Componente Crear Tarea

```
this.tareasService
.uploadFiles(
  localStorage.getItem('tokken'),
  response['tarea']._id,
  this.toUploadFiles
  )
.subscribe(
    result => {
      if (result['ok'] == true) {
        this.loading = false;
        stepper.next();
      } else {
        this.getSubscriptionError(result);
      }
    },
    error => {
      this.getSubscriptionError(error);
    }
  );
```

Figura 24 Patrón Observer en la clase Crear Tarea
Elaborado por: Carolina Cutiupala y Daniel Toinga

3.2.2. Patrones de diseño en el back-end

3.2.2.1. Reactor

Como se muestra en la figura 25 maneja las peticiones de manera asíncrona, NodeJs tiene tres maneras distintas de trabajar con este tipo de patrón, pero en el contexto de este proyecto como se muestra en la figura 26 solo se usan dos: Callbacks y Async/Await.

Patrón Reactor Implementado con Callbacks

```
tarea.save((err, tareaStored) => {
    if (err) {
        res.status(500).send({
            messaje: 'Error al guardar la tarea'
        });
    } else {
        if (!tareaStored) {
            res.status(404).send({
                messaje: 'No se ha registrado la tarea'
            });
        } else {
            res.status(200).send({
                tarea: tareaStored
            });
        }
    }
})
```

Figura 25 Patrón reactor implementado con Callbacks
Elaborado por: Carolina Cutiupala y Daniel Toinga

Patrón Reactor Implementado con Async/Await

```
async function getGradedTasks(req, res) {
    let userId = req.user.sub;
    try {
        let tareas = await Tarea.find({ creador: userId, fechaRevision: { $lte: new Date() } }, { _id: 1 });
        let obtained = await UserTask.find({ tarea: { $in: tareas }, calificado: false })
            .populate('usuario')
            .populate('tarea')
            .populate('materia')
            .populate('criterios');
        res.status(200).send({ tareas: obtained });
    } catch (error) {
        console.log(error);
        res.status(503).send({ message: 'Error en el servidor' });
    }
}
```

Figura 26 Patrón reactor implementado con async/await
Elaborado por: Carolina Cutiupala y Daniel Toinga

3.2.2.2. Factory

El patrón Factory permite tener una forma genérica de crear objetos encapsulando su implementación. Node utiliza una manera simple basada en módulos para implementar este tipo de patrón como se muestra en la figura 27, cabe destacar que en el contexto de

este proyecto este patrón se usa casi exclusivamente para la creación de los modelos para la base de datos. Por lo tanto, el ejemplo presentado a continuación refleja dos cosas: la implementación del patrón factory en el back-end, y el modelo de datos de la colección "Usuario" en la base de datos.

Implementación del Patrón Factory

```
var mongoose=require('mongoose');
var Schema=mongoose.Schema;

var UsuarioSchema=Schema({
    cedula:String,
    nombres:String,
    apellidos:String,
    correo:String,
    password:String,
    rol:{type:Schema.ObjectId, ref:'Rol'},
    materias:[{type:Schema.ObjectId, ref:'Materia'}],
    area: String,
    estado:Boolean,
    ultimoAcceso: Date,
    apoyo:[{type:Schema.ObjectId, ref:'Usuario'}],
});

module.exports=mongoose.model('Usuario',UsuarioSchema);
```

Figura 27 Implementación del patrón factory
Elaborado por: Carolina Cutiupala y Daniel Toinga

3.2.2.3. Middleware

Ya que las funciones middlewares tienen acceso a los objetos request y response, se pueden utilizar para que actúen como una barrera de acceso al sistema como se muestra en la figura 28. Los middlewares implementan la función next() para impedir o permitir el acceso a los recursos del back-end.

Implementación del Patrón Middleware

```
var jwt=require('jwt-simple');
var moment= require('moment');
var secret='clave_secreta_de_la_aplicacion_docentes';

exports.ensureAuth=function(req,res,next){
    if(!req.headers.authorization){
        return res.status(403).send({messaje:'La peticion no tiene la cabecera de autenticación'})
    }else{
        var token=req.headers.authorization.replace(/['"]+/g,'');

        try{
            var payload=jwt.decode(token,secret);
            if(payload.exp <= moment().unix()){
                return res.status(401).send({
                    messaje:'El token ha expirado'
                });
            }
        }catch(ex){
            return res.status(404).send({
                messaje:'El token no es valido'
            });
        }
        req.user=payload;
    }
    next();
}
```

```
api.put('/addCriteria/:id',[md_auth.ensureAuth],TareaController.addCriteria);
api.delete('/deleteCriteria/:id',[md_auth.ensureAuth],TareaController.deleteCriteria);
api.get('/getTask/:id',[md_auth.ensureAuth],TareaController.getTask);
api.get('/getAllTasks',[md_auth.ensureAuth],TareaController.getAllTasks);
api.get('/getAllListedTasks',[md_auth.ensureAuth],TareaController.getAllListedTasks);
```

Figura 28 Implementación del patrón middleware
Elaborado por: Carolina Cutiupala y Daniel Toinga

3.3. Código relevante

A continuación, se muestra el código más relevante de la aplicación ya sea porque representa una nueva funcionalidad en el software o una modificación de una funcionalidad anterior.

3.3.1. Services

Como se muestra en la figura 29 se crea una variable global llamada GLOBAL la cual contiene la URL principal de conexión con el servidor.

En el árbol de archivos del proyecto se crea una carpeta llamada Services como se muestra en la figura 30 la cual contiene todos los métodos de comunicación HTTP con la API

REST, esto es con el fin de mejorar la mantenibilidad del código ya que no se necesitará modificar cada función en cada clase que se implemente estos servicios. El reto aquí está en revisar cada URL usada en el software a refactorizar, revisar su funcionalidad y que no esté duplicada.

Variable Global para la Conexión

```
//Url para servicios
export var GLOBAL={
    url:'http://localhost:3789/api/'
}
```

Figura 29 Variable global para la conexión con el back-end
Elaborado por: Carolina Cutiupala y Daniel Toinga

Implementación de Servicios para la Conexión con el Back-end

```
import { Injectable } from '@angular/core';
import { map } from 'rxjs/operators';
import { GLOBAL } from './global';
import { HttpClient, HttpHeaders } from '@angular/common/http';

@Injectable()
export class UserService {
    public url: string;
    public identity;
    public token;

    constructor(private _http: HttpClient) {
        this.url = GLOBAL.url;
    }
```

```
getCarreras(token){
    const headers = new HttpHeaders({ 'Content-Type': 'application/json','Authorization':token });
    return this._http.get(this.url + 'getCarreras' , { headers })
        .pipe(map(res => res));
}
getPeriodos(token){
    const headers = new HttpHeaders({ 'Content-Type': 'application/json','Authorization':token });
    return this._http.get(this.url + 'getPeriodos' , { headers })
        .pipe(map(res => res));
}
updateUserData(token, id, data){
    const headers = new HttpHeaders({ 'Content-Type': 'application/json','Authorization':token });
    return this._http.put(this.url + 'updateUser/'+id,data, { headers })
        .pipe(map(res => res));
}
changePassword(token,data){
    const headers = new HttpHeaders({ 'Content-Type': 'application/json','Authorization':token });
    return this._http.put(this.url + 'changePassword',data, { headers })
        .pipe(map(res => res));
```

Figura 30 Implementación de servicios para la conexión con el back-end
Elaborado por: Carolina Cutiupala y Daniel Toinga

3.4.2. Refactorización de la función Crear tarea

Todas las funciones existentes en la aplicación anterior como: crear tarea, modificar tarea, crear reunión y calificar tarea se refactorizaron utilizando TypeScript en lugar de JavaScript en las funciones creadas con los servicios. Como se muestra en las figuras 31 y 32, la funcionalidad se mantiene respecto a la aplicación a refactorizar, pero la sintaxis cambia. Una de las tareas más difíciles de la refactorización de código es entender cómo se estructuran las clases en el árbol de directorios del proyecto, ya que, en una sola clase se encuentran demasiados métodos que se comunican con diferentes clases y con el backend

Extracto de la Función Crear Tarea Versión Anterior en JavaScript

```
app.$http.post('http://localhost:8080/sistEval/ws/crearTarea/', this.tareaData).then(functio
    loading_screen.finish();
    console.log("tarea");
    console.log(data);
    toastr.success('Tarea creada');
    console.log('UsuariosModulo >>>>>>>', app.tareasModel.usuariosModulo);
    resetFields();
    console.log('UsuariosModulo >>>>>>>', app.tareasModel.usuariosModulo);
}, function (data) {
    loading_screen.finish();
    console.log("ERROR");
    toastr.error('Error al crear tarea');
});
```

Figura 31 Extracto de la función crear tarea versión anterior en JavaScript
Elaborado por: Carolina Cutiupala y Daniel Toinga

Extracto de la Función Crear Tarea Refactorizada en TypeScript

```
this.tareasService.saveTarea(localStorage.getItem('tokken'), this.datoTarea).subscribe(
  response => {
    this.datoTarea._id = response['tarea']._id;
    if (!this.toUploadFiles || this.toUploadFiles.length == 0) {
      stepper.next();
    } else {
      this.tareasService.uploadFiles(
        localStorage.getItem('tokken'),
        response['tarea']._id,
        this.toUploadFiles
        ).subscribe(
          result => {
            if (result['ok'] == true) {
              this.loading = false;
              stepper.next();
            } else {
              this.getSubscriptionError(result);
            }
          },
          error => {
            this.getSubscriptionError(error);
          }
        );
    }
  },
```

Figura 32 Extracto de la función crear tarea refactorizada en TypeScript
Elaborado por: Carolina Cutiupala y Daniel Toinga

3.4.3. Historial de tareas

A continuación, se presenta la nueva implementación de la funcionalidad historial de tareas mediante la cual los usuarios con roles:Director de Carrera, Jefes de Área o Coordinadores, pueden acceder a todas las tareas que hayan creado anteriormente para modificar los datos de estas como se muestra en la figura 33, los datos modificados se verán inmediatamente reflejados en las pantallas principales de cada usuario al que se le haya asignado dicha tarea.

Clase Historial

```
export class HistorialComponent implements OnInit {

  public rol: number = 0;
  public token;
  public usuario;
  toUploadFiles: Array<File>;
  data: any[] = [];
  displayedColumns: string[] = ['tarea', 'periodo', 'limite', 'revision', 'creado', 'accion'];
  dataSource = new MatTableDataSource<any>(this.data);

  constructor(
    private tareasUsuarioService: TareasUsuarioService,
    private userService: UserService,
    private errorService: ErrorDialogService,
    public dialog: MatDialog,
    private _location: Location,
    private tareasService: TareasService,
  ) {
    this.token = localStorage.getItem('tokken');
    this.usuario = this.userService.getIdentity();
    this.refrescarTabla();
  }

  refrescarTabla() {
    this.tareasUsuarioService.getTareasCreadasUsuario(this.token, this.usuario._id).subscribe(
      response => {
        this.dataSource.data = response['tareas'];
      },
      error => {
        this.getSubscriptionError(error);
      }
    );
        this.tareasService.updateTarea(localStorage.getItem('tokken'), element._id, result.datos).subscribe(
          response => {
            if (result.archivos) {
              if (result.archivos.length == 0) {
                this.errorService.openDialog('Hecho', 'La tarea se ha modificado con exito');
                this.refrescarTabla();
              } else {

                this.tareasService.uploadFiles(localStorage.getItem('tokken'), element._id, result.archivos).
                  subscribe(
                    result => {
                      if (result['ok'] == true) {
                        this.errorService.openDialog('Hecho', 'La tarea se ha modificado con exito');
                        this.refrescarTabla();
                      } else {
                        this.getSubscriptionError(result);// Por probar respuesta de error
                      }
                    },
                    error => {
                      this.getSubscriptionError(error);
                    }
                  );
              }
            } else {
              this.errorService.openDialog('Hecho', 'La tarea se ha modificado con exito');
              this.refrescarTabla();
            }

          },
          error => {
            this.getSubscriptionError(error);
          }
```

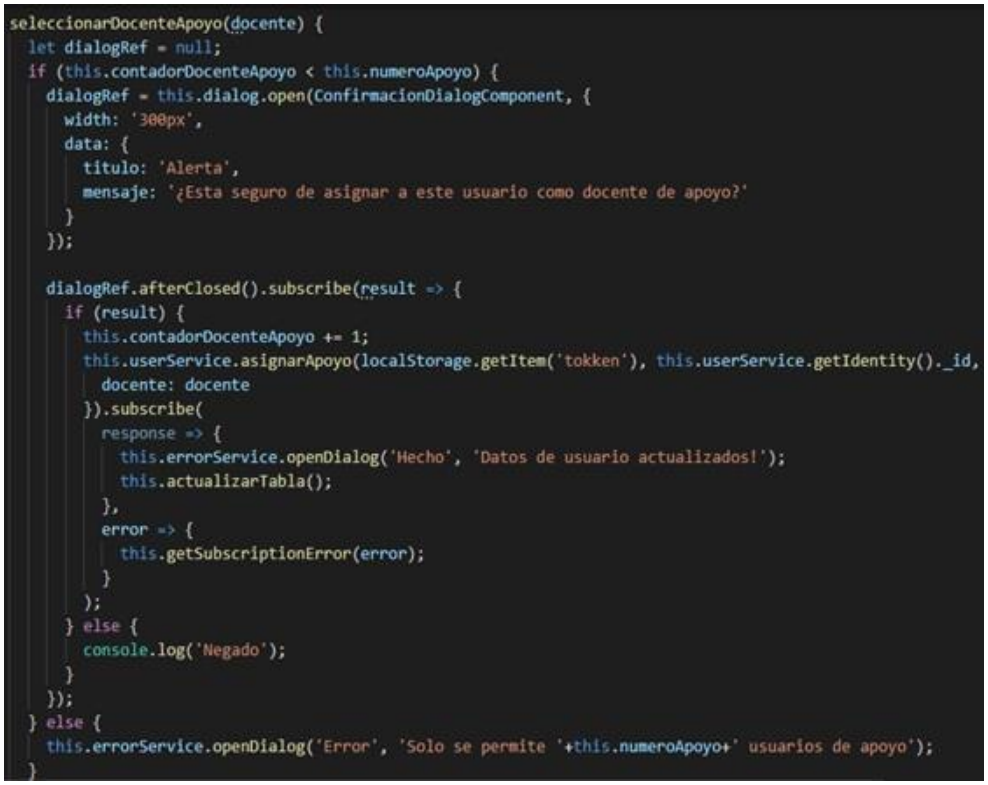
Figura 33 Extracto de la clase historial
Elaborado por: Carolina Cutiupala y Daniel Toinga

3.4.4. Asignación de docente de apoyo

Otra funcionalidad nueva es la de asignar docentes de apoyo como se muestra en la figura 34, esta funcionalidad está disponible para los roles: Director de Carrera, Jefes de Área y Coordinadores los cuales pueden elegir de la lista de docentes a sus respectivos docentes de apoyo para que cumplan con las tareas designadas, también el administrador que en

este caso es el Director de Carrera puede modificar el número máximo de docentes de apoyo que se pueden seleccionar.

Clase Asignación del Docente de Apoyo



```
seleccionarDocenteApoyo(docente) {
  let dialogRef = null;
  if (this.contadorDocenteApoyo < this.numeroApoyo) {
    dialogRef = this.dialog.open(ConfirmacionDialogComponent, {
      width: '300px',
      data: {
        titulo: 'Alerta',
        mensaje: '¿Esta seguro de asignar a este usuario como docente de apoyo?'
      }
    });

    dialogRef.afterClosed().subscribe(result => {
      if (result) {
        this.contadorDocenteApoyo += 1;
        this.userService.asignarApoyo(localStorage.getItem('tokken'), this.userService.getIdentity()._id,
          docente: docente
        }).subscribe(
          response => {
            this.errorService.openDialog('Hecho', 'Datos de usuario actualizados!');
            this.actualizarTabla();
          },
          error => {
            this.getSubscriptionError(error);
          }
        );
      } else {
        console.log('Negado');
      }
    });
  } else {
    this.errorService.openDialog('Error', 'Solo se permite '+this.numeroApoyo+' usuarios de apoyo');
  }
```

Figura 34 Asignación del docente de apoyo
Elaborado por: Carolina Cutiupala y Daniel Toinga

3.4. Catálogo de servicios

Todas las URI tendrán la cabecera común que es **http://localhost:3789/api**

Tabla 5 Catálogo de Servicios del Software de Seguimiento a Docentes Refactorizado

Método HTTP	URI	Función
Post	/login	Recibe el email y la contraseña del usuario y verifica si son correctos

Post	/saveUser	Recibe los datos de un nuevo usuario y lo guarda en la base de datos
Get	/getAllUsers	Devuelve los datos de todos los usuarios
Put	/updateUser	Actualiza los datos del usuario
Get	/getUserMat/:id	Recibe mediante la url el Id del usuario logueado y devuelve todas sus materias
Get	/getCarreras	Devuelve todas las carreras registradas
Put	/asignarApoyo/:id	Recibe mediante url el Id del usuario logueado y el id del usuario de apoyo mediante el body en la petición y modifica los datos del usuario logueado
Get	/getMisTareas	Devuelve todas las tareas asignadas al usuario logueado
Get	/getNumeroApoyo	Devuelve al número máximo de docentes de apoyo
Put	/updateNumeroApoyo	Actualiza el número máximo de docentes de apoyo
Get	/downloadFileTareaUsuario/:adjFile	Recibe como parámetro de url el nombre del archivo adjunto y permite la descarga de dicho archivo
Post	/saveUserTask	Guarda la tarea a cada usuario en la base de datos
Put	/updateUserTask/:id	Recibe el id del usuario mediante url y modifica la tarea asignada
Get	/getUserTask/:id	Recibe el id de la tarea del usuario y devuelve los datos de dicha tarea asignada

Get	/getAllUserTasks/:id	Recibe el id del usuario logueado y devuelve todas las tareas asignadas
Post	/uploadTareaUser/:id	Recibe el id del usuario logueado y archivos en el body de la petición, permite cargar archivos
Get	/getToGradeTasks	Recibe el id del usuario logueado y devuelve todas las tareas que tenga por calificar
Get	/getForReportsTasks	Recibe el id del usuario logueado y envía todos los datos de las tareas calificadas
Get	/getAllHistorialTasks	Recibe el id del usuario logueado y devuelve todas las tareas que ha creado
Put	/addCriteria/:id	Recibe el id del criterio mediante la url y añade el criterio enviado en el body de la petición
Delete	/deleteCriteria/:id	Recibe el id del criterio mediante la url y cambia el campo “estado” del criterio a “false” haciendo que el criterio quede inhabilitado
Get	/getAllTasks	Devuelve todas las tareas que se han creado
Get	/getAllListedTasks	Devuelve todas las tareas que se hayan guardado como plantillas
Get	/getAllMaterias	Devuelve todas las materias registradas
Get	/getMateria/:id	Recibe por url el id de la materia y devuelve todos sus datos
Post	/uploadEcxelFiles	Recibe el archivo Excel de distributivo docente por el body

		de la petición y guarda los datos en la base de datos
Get	/getMateriaXDocentes	Devuelve todos los datos del distributivo de docentes
Get	/getPeriodo	Devuelve el periodo actual de las materias
Post	/saveCriteria	Guarda los datos de un nuevo criterio enviado en el body de la petición
Put	/updateCriteria/:id	Recibe el id del criterio en la url y modifica los datos enviados en el body de la petición
Get	/getCriteria/:id	Devuelve los datos del criterio enviado en la url
Get	/getAllCriteria	Devuelve todos los criterios creados por el usuario logueado

Nota: Catálogo que contiene las URI de acceso al back-end para el software refactorizado

3.5. Plan de pruebas

Este plan fue creado para verificar que el software Refactorizado funcione correctamente de acuerdo con las especificaciones requeridas por los usuarios aplicando tres tipos de pruebas las cuales son: pruebas funcionales, rendimiento y aceptación que permitan detectar errores para poder corregirlos, el plan completo de pruebas se puede ver en el (Anexo D).

3.5.1. Objetivo del plan de pruebas

Comprobar la calidad del software Refactorizado, mediante la aplicación de varias pruebas que permitan identificar los fallos para corregirlos y cumplir con los requerimientos establecidos.

3.5.2. Alcance de las pruebas

Evaluar la calidad del software Refactorizado completamente integrado, para ello se probará cada uno de los módulos con sus diferentes roles de usuario entre ellos los siguientes:

- Módulo de administración
- Módulo de crear tarea
- Módulo de crear reunión
- Módulo de calificar tarea
- Módulo de asignar docente de apoyo

3.5.3. Enfoque de pruebas

Se tomó como referencia el diagrama de flujo del proceso, se seleccionó las pruebas funcionales, aceptación y de carga para probar que el software cumpla con la función designada a cada usuario, para obtener como resultado un software de calidad de evaluación de docentes exitoso basado en asignación de tareas y cumplimiento de estas**.**

3.5.3.1. Criterios de Entrada

Para comenzar la fase de pruebas el software debe cumplir con los siguientes criterios:

- El software de estar completamente integrado
- El administrador debe asignar un rol para el ingreso de cada usuario

3.5.3.2. Criterios de Salida

- Mensajes de error
- Botones bloqueados
- Mensajes de confirmación

- Bloqueo y redireccionamiento

3.5.4. Ejecución de pruebas

Las pruebas fueron ejecutadas en diferentes ambientes que permitieron cumplir con los objetivos planeados en el plan de pruebas, también se utiliza los casos de pruebas para cada una de las pruebas seleccionadas y poder registrar el resultado obtenido de cada una de ellas.

3.5.4.1. Pruebas funcionales

En las pruebas funcionales se evaluaron todos los módulos existentes anteriormente descritos, aplicando todos los casos pruebas creados que se encuentran en el Anexo E, pero se mostraran solo los casos pruebas más relevantes para comprobar que el software es funcional.

A continuación, se listarán en las tablas 6, 7, 8 y 9 los casos pruebas más relevantes utilizadas en el presente trabajo.

Tabla 6. Caso Prueba Probar Tarea

ID de caso de pruebas	**003-003**	
Título del caso de prueba	Probar el módulo de Crear Tarea	
Descripción	Se busca evaluar la tarea sea creada y asignada correctamente a los docentes seleccionados	
Precondiciones	• Que existan usuario registrados en la base de datos. • Que existan materias registradas.	
Pasos y condiciones de ejecución	1.- Que el rol del usuario no sea docente 2.- Clic en el botón Empezar de la tarjeta crear tarea 3.- Llenar el formulario de Descripción de la Tarea 4.- Seleccionar categorías y criterios 3.- Seleccionar a los docentes o grupos	
Variantes	**Resultado esperado**	**Resultado obtenido**

1. Si no se ha cargado el archivo de información académica	No permite crear la tarea.	Mensaje de error y no se puede continuar con el proceso de crear la tarea.
2. No se seleccionan docentes	Mensaje de error y no se puede continuar con el proceso	Si se muestra el mensaje de error
3. Se guarda la tarea como plantilla	Se muestra la tarea creada en la pestaña de plantillas de tareas.	No se visualiza ninguna tarea guardada.
4. Se carga archivo	Archivo asignado guardado en la base de datos	Que se visualice en la base
5. Campos vacíos en el formulario de descripción de la tarea	No se puede continuar con el proceso y el botón de siguiente desactivado	No se puede avanzar con el proceso el botón se encuentra inactivo
Responsable ejecución	Carolina Cutiupala, Daniel Toinga	
Comentarios		

Nota: Este es un caso prueba en el cual muestra los resultados al utilizar Crear Tarea

Tabla 7 Caso Prueba Probar el Funcionamiento de Calificar Tarea

ID de caso de pruebas	004-004	
Título del caso de prueba	Probar el módulo de Calificar Tarea	
Descripción	Se busca evaluar que el proceso de calificación sea correcto	
Precondiciones	• Que existan usuario registrados en la base de datos. • Que existan materias registradas.	
Pasos y condiciones de ejecución	1.- Clic en el botón Empezar de la tarjeta calificar tareas 2.- Seleccionar una tarea de la lista 3.- Seleccionar los criterios cumplido 4.- clic en el botón enviar	
Variantes	**Resultado esperado**	**Resultado obtenido**
1. No se selecciona ningún criterio	No se activa en botón enviar	No se activa en botón enviar
2. Si excede el porcentaje asignado por comentario	No se activa en botón enviar	No se activa en botón enviar
Responsable ejecución	Carolina Cutiupala, Daniel Toinga	
Comentarios		

Nota: Este es un caso prueba en el cual muestra los resultados al utilizar Calificar Tarea

Tabla 8 Caso Prueba Probar Subir Archivos

ID de caso de pruebas	005-005	
Título del caso de prueba	Probar el módulo de Subir archivos	
Descripción	Se busca evaluar que el archivo distributivo académico se cargue correctamente	
Precondiciones	Que haya un usuario con rol de administrador o Director de Carrera registrado en la base de datos.	
Pasos y condiciones de ejecución	1. Clic en el menú subir archivos 2. Clic en el botón seleccionar archivos 3. Seleccionar archivo 4. Clic en el botón subir archivo	
Variantes	**Resultado esperado**	**Resultado obtenido**
1. Se selecciona un archivo con extensión diferente de xlsx	Mensaje de error	No se carga el archivo
2. Seleccionar archivo incorrecto	Mensaje de error	Se cae el programa
Responsable ejecución	Carolina Cutiupala, Daniel Toinga	
Comentarios	Falta mensaje de archivo erróneo sin el formato establecido	

Nota: Este es un caso prueba en el cual muestra los resultados al utilizar subir archivos

Tabla 9 Caso Prueba Probar Asignar Docente de Apoyo

ID de caso de pruebas	008-008	
Título del caso de prueba	Probar el módulo de Seleccionar docentes de apoyo	
Descripción	Se busca evaluar que se asigne o se quite el permiso de docente de apoyo	
Precondiciones	Que existan usuarios registrados en la base de datos	
Pasos y condiciones de ejecución	1. Clic en el botón empezar de la tarjeta asignar docente de apoyo 2. Clic en el botón seleccionar de cualquier usuario 3. Clic en el botón si del cuadro de confirmación	
Variantes	**Resultado esperado**	**Resultado obtenido**

1. Selección de más de tres usuarios	Mensaje de error y usuario no seleccionado	Mensaje de error y usuario no seleccionado
2. Quitar permiso de usuario de apoyo	Mensaje de confirmación	Mensaje de confirmación
Responsable ejecución	Carolina Cutiupala, Daniel Toinga	
Comentarios		

Nota: Este es un caso prueba en el cual muestra los resultados al utilizar seleccionar docentes

3.5.4.2. Pruebas de aceptación

Las pruebas fueron realizadas por el personal docente con sus diferentes roles, así se observó que cada usuario ingresaba a su pantalla asignada para realizar sus actividades.

Los casos pruebas presentados a continuación en las tablas 10, 11 y 12 son los más relevantes en el proceso de pruebas con cada usuario:

Tabla 10 Prueba de Aceptación Director de Carrera

Rol: Director de Carrera	
Nombre	Patsy Prieto
Nombre caso prueba	Verificar generación reportes
Descripción	Se pretende probar que los reportes presenten la información adecuada y los cálculos de las notas sean correctos.
Resultado esperado	Visualización de reportes por tarea, materia, usuario y periodo entendible y bien calculado
Resultado obtenido	Visualizaciones y datos correctos
Comentarios	El usuario pide visualización de los reportes de manera gráfica y por pregunta de evaluación

Nota: Es un caso prueba con resultados utilizando el rol de Director de Carrera.

Tabla 11 Prueba de Aceptación Jefe de Área Formación Profesional

Rol: Jefe de Área	
Nombre	Ricardo Albarracín
Nombre caso prueba	Verificar modificación de tarea
Descripción	En el módulo historial de tareas se pretende probar que se puedan modificar los datos de una tarea enviada
Resultado esperado	Los datos de la tarea se actualicen con éxito
Resultado obtenido	Los datos de la tarea se actualizaron con éxito
Comentarios	Sin comentarios

Nota: Es un caso prueba con resultados utilizando el rol de Jefe de Área.

Tabla 12 Prueba de Aceptación jefe de Área Formación Básica

Rol: Jefe de Área	
Nombre	Daniela García
Nombre caso prueba	Verificar calificación de tareas
Descripción	Se pretende comprobar que la calificación de las tareas sea basada en criterios y que los resultados de los reportes sean correctos
Resultado esperado	Validación de valores máximos de los puntajes asignados a los criterios y su suma total correcta.
Resultado obtenido	Tareas calificadas con puntajes correctos
Comentarios	Los reportes deberían ser también por pregunta de evaluación

Nota: Es un caso prueba con resultados utilizando el rol de Jefe de Área.

Tabla 13 Prueba de Aceptación Coordinador

Rol: Coordinador de comisión	
Nombre	Daniel Díaz
Nombre caso prueba	Validación de campos
Descripción	Se pretende validar que los campos para ingreso de datos estén controlados según su especificación para: email, números, límite de caracteres.
Resultado esperado	Todos los campos validados correctamente
Resultado obtenido	Validación correcta
Comentarios	Aumentar el número máximo de caracteres para nombre y descripción de tarea

Nota: Es un caso prueba con resultados utilizando el rol de Coordinador.

Como resultado obtuvimos la completa aceptación del funcionamiento del software por parte de los docentes los cuales firmaron un acta de aceptación, que se encuentra en el Anexo F, asegurando que el software cumplía con los requerimientos establecidos.

Cabe destacar que los comentarios y sugerencias recibidas en estas pruebas fueron solventados en la versión final.

3.5.4.3. Pruebas de Rendimiento

Para las pruebas de rendimiento se utiliza la herramienta JMeter la cual facilita realizar pruebas sobre todos los endpoints que forman parte de las principales actividades del sistema que son: Crear tarea, Realizar tarea y Calificar tarea.

Para estas pruebas se utiliza una máquina virtual en el servidor de la UPS Campus Sur, el cual es el servidor de producción donde se cargó todo el sistema, incluida la base de datos, este servidor cuenta con las siguientes características:

Tabla 14 Especificaciones de Hardware del Servidor de Producción

Procesador:	4 núcleos
RAM:	8 GB
Almacenamiento:	500 GB
Sistema Operativo:	Ubuntu Server 18.04 LTS
IP:	172.17.42.236

Nota: Especificaciones para el servidor de producción.

En estas pruebas de rendimiento lo que se intento fue simular la carga de usuarios en el sistema dentro del flujo del software permitiendo probar secuencialmente las peticiones al servidor para esto se crea un proxy usando la herramienta JMeter y se configura el navegador Firefox para usarlo, de esta manera se pudo interceptar todas las peticiones HTTP que se ejecutan al realizar cualquier actividad en el software.

Para simular la carga de usuarios se tomó como base, los datos del documento distributivo de usuarios por docentes **(Anexo I)**, mediante el cual se generó la siguiente tabla.

Tabla 15 Número de Usuarios para Pruebas

Número de usuarios registrados	27
Número de usuarios con roles administrativos	6
Número de usuarios de apoyo	3

Nota: Números de usuarios obtenidos del distributivo de docentes.

Configuración de la herramienta de pruebas

La herramienta JMeter cuenta con varios parámetros configurables, para el objetivo de las pruebas de rendimiento principalmente se utiliza tres de ellos.

- Número de hilos: este parámetro simula el número de usuarios en el sistema

- Periodo de subida: es el tiempo en segundos durante el cual se irá activando cada hilo, es decir, simula la conexión secuencial de usuarios durante un periodo de tiempo.
- Contador de bucle: es el numero veces que se repite cada prueba.

Para determinar el periodo de subida se utilizará el siguiente formula:

Tiempo de subida = Numero de hilos * 3

Esto quiere decir que los usuarios ingresarán al sistema secuencialmente en un intervalo fijo de 3 segundos.

El contador de bucle se establece fijo en 5, ya que realizar cada prueba cinco veces provee resultados suficientes.

Criterios de aceptación

Para declarar cada prueba de rendimiento como exitosa se determina que la métrica principal es el tiempo, para representar mejor este criterio se crea la siguiente tabla:

Tabla 16 Criterios de Aceptación para el Rendimiento de la Aplicación

RESULTADO	TIEMPO
Correcto	menos 1 segundo
Aceptable	Entre 1 y 4 segundos
Erróneo	Mas de 4 segundos

Nota: Criterios basados en el tiempo de respuesta.

Caso 1

Prueba rendimiento con 6 usuarios en el escenario de crear tarea

En la primera prueba se configura la herramienta simulando la carga con 6 que es el número de los usuarios con roles administrativos.

Resultados Prueba de Rendimiento Crear Tarea para 6 Usuarios

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error
/api/getAllListedTasks-11	30	66	59	84	86	105	46	105	0,00%
/api/getTask/5df99a8c3825b036b041cd66-13	30	5	4	12	12	20	3	20	0,00%
/api/saveTarea-15	30	5	5	7	8	13	4	13	0,00%
/api/updateTarea/5e13bb98e9e0530c55da4945-17	30	4	3	4	4	16	3	16	0,00%
/api/getUserMat/5dec177fa38ba90a38faebdb-21	30	6	6	9	9	10	5	10	0,00%
/api/saveUserTask-19	30	135	131	158	169	218	114	218	0,00%
Total	180	37	6	126	134	169	3	218	0,00%

Figura 35 Resultados pruebas de rendimiento crear tarea para 6 usuarios
Elaborado por: Carolina Cutiupala y Daniel Toinga

En la figura 35 se muestra numéricamente los resultados de la prueba de rendimiento, siendo las comunas de interés las llamadas “Media”, “Max” y “%Error”.

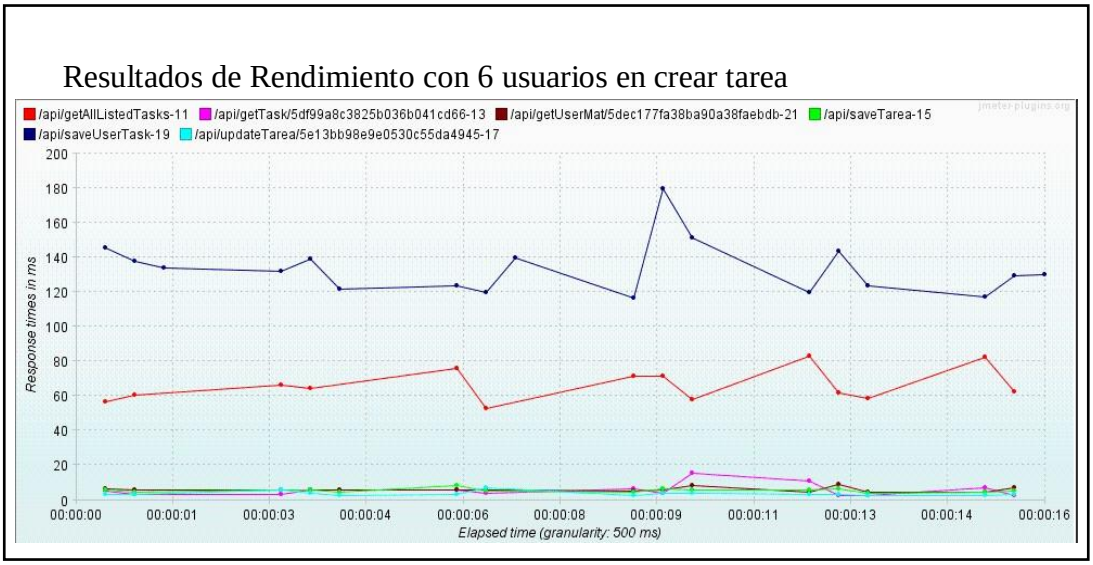


Figura 36 Gráfico resultados pruebas de rendimiento Crear tarea para 6 usuarios
Elaborado por: Carolina Cutiupala y Daniel Toinga

En la figura 36 se muestran gráficamente los resultados de la prueba de rendimiento, se puede evidenciar que la petición a la URI "/api/saveUserTask" es la que mayor tiempo de respuesta tiene, esto se debe a que el servidor debe hacer más escrituras a la base de datos.

Prueba rendimiento con 18 usuarios en el escenario de crear tarea

En la segunda prueba se configura la herramienta para simular la carga de 18 usuarios que es el número de usuarios con cargos administrativos, cada uno con sus 3 docentes de apoyo.

Pruebas con 18 Usuarios en Crear Tarea

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error
/api/getAllListedTasks-11	90	92	91	116	120	146	62	184	0,00%
/api/getTask/5df99a8c3825b036b041cd66-13	90	6	4	8	8	77	3	83	0,00%
/api/saveTarea-15	90	6	5	9	11	20	4	27	0,00%
/api/updateTarea/5e13bb98e9e0530c55da4945-17	90	4	4	5	6	16	2	22	0,00%
/api/getUserMat/5dec177fa38ba90a38faebdb-21	90	5	5	7	9	13	3	13	0,00%
/api/saveUserTask-19	90	124	120	144	153	179	97	203	0,00%
Total	540	39	6	120	132	157	2	203	0,00%

Figura 37 Resultados pruebas de rendimiento Crear tarea para 18 usuarios
Elaborado por: Carolina Cutiupala y Daniel Toinga

En la figura 37, se muestra numéricamente los resultados de la prueba de rendimiento para 18 usuarios, donde se puede evidenciar que el tiempo máximo de respuesta fue 203 milisegundos.

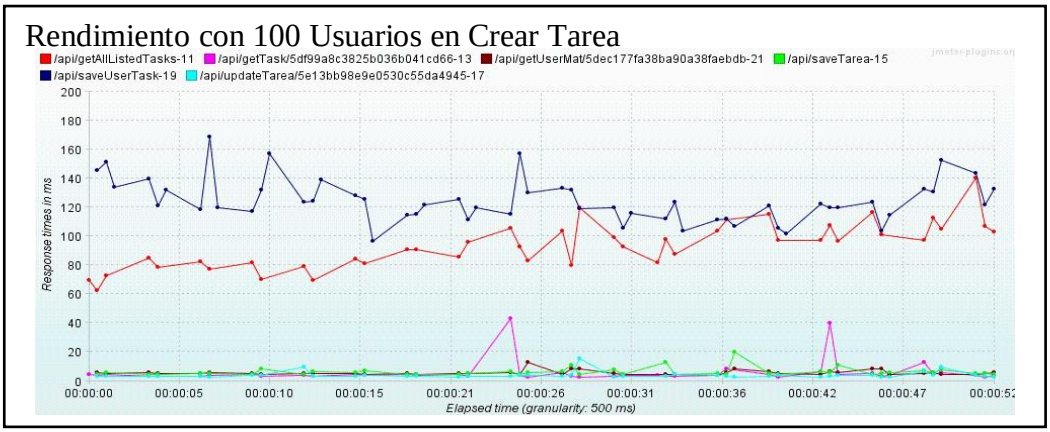


Figura 38 Gráfico resultados pruebas de rendimiento Crear tarea para 18 usuarios
Elaborado por: Carolina Cutiupala y Daniel Toinga

En la figura 38 se puede evidenciar gráficamente que la URI "/api/saveUserTask" sigue siendo la que tiene mayor tiempo de respuesta.

Resultado del rendimiento en el escenario de crear tarea entre la carga de 6 vs 18 usuarios

Nótese que el tiempo de respuesta en la prueba con 18 usuarios como se muestra en la figura 37 y 38 la última petición, es más elevado que con 6 usuarios como se muestra en las figuras 35 y 36, esto se debe a que se configura la prueba simulando enviar una tarea a todos los usuarios registrados, por lo cual el servidor debe realizar un mayor procesamiento para actualizar los datos en la base de datos.

El comportamiento de la aplicación es el esperado, ya que el motor de NodeJs no se bloquea, es decir, no deja de recibir peticiones y responderlas, pero si se existirá una respuesta más lenta es debido a que la carga excesiva en el servidor exige más al procesador.

En ambos casos el tiempo de respuesta no sobrepaso 1 segundo, por lo tanto, el rendimiento es correcto

Caso 2

Prueba de rendimiento con 27 usuarios en el escenario de realizar tarea

En esta prueba se configura la herramienta para simular la carga de 27 usuarios que es el número total de usuarios del sistema, esto es debido a que la actividad de realizar tarea la hacen todos los usuarios.

Rendimiento Realizar Tarea para 27 Usuarios

Etiqueta	# Muestras	Media	Mín	Máx	% Error	Rendimiento	Kb/sec	Sent K...
/api/getAllUserTasks/5defb59586bbb429d47d0a2f-2	135	21	11	97	0,00%	5,2/sec	18,21	8,42
/api/updateUserTask/5e06b28c7e1bfd2f799dee13-4	135	5	3	22	0,00%	5,2/sec	2,44	9,00
/api/getAllUserTasks/5defb59586bbb429d47d0a2f-6	135	19	11	112	0,00%	5,2/sec	18,24	8,43
Total	405	15	3	112	0,00%	15,5/sec	38,83	25,80

Figura 39 Resultados pruebas de rendimiento realizar tarea para 27 usuarios
Elaborado por: Carolina Cutiupala y Daniel Toinga

En la figura 39 se muestra numéricamente los resultados de la prueba de rendimiento para 27 usuarios, donde se puede evidenciar que el tiempo máximo de respuesta es 112 milisegundos y una media de 15 milisegundos.

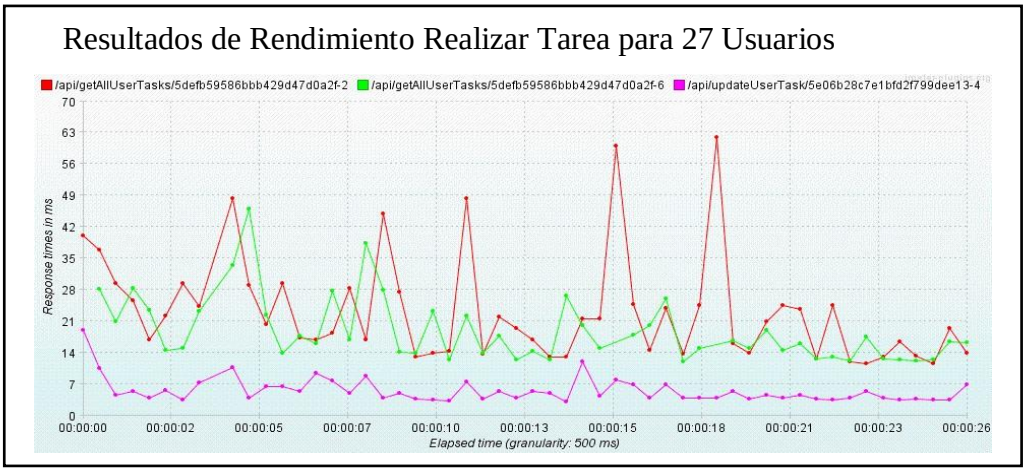


Figura 40 Gráfico resultados pruebas de rendimiento realizar tarea para 27 usuarios
Elaborado por: Carolina Cutiupala y Daniel Toinga

En la figura 40 se muestra gráficamente los resultados de la prueba de rendimiento con 27 usuarios, en la cual se puede evidenciar dos picos en el tiempo de respuesta que rondan los 63 milisegundos.

Prueba de sobrecarga con 100 usuarios en el escenario de realizar tarea

En la segunda prueba se configura la herramienta con una sobrecarga de usuarios para comprobar que la respuesta del servidor es correcta.

Se simula la sobrecarga de 100 usuarios los cuales ingresarán al sistema secuencialmente en un lapso de 27 segundos. Esta prueba se realiza 2 veces.

Rendimiento Realizar Tarea para 100 Usuarios

Etiqueta	# Muestras	Media	Mín	Máx	% Error	Rendimi...	Kb/sec	Sent KB/...
/api/getAllUserTasks/5defb59586bbb429d47d0a2f-2	200	17	10	52	0,00%	7,5/sec	26,32	12,17
/api/updateUserTask/5e06b28c7e1bfd2f799dee13-4	200	4	3	30	0,00%	7,5/sec	3,53	12,99
/api/getAllUserTasks/5defb59586bbb429d47d0a2f-6	200	15	10	66	0,00%	7,5/sec	26,33	12,17
Total	600	12	3	66	0,00%	22,3/sec	56,11	37,28

Figura 41 Resultados pruebas de rendimiento realizar tarea para 100 usuarios
Elaborado por: Carolina Cutiupala y Daniel Toinga

En la figura 41 se muestra el resultado de la prueba de sobrecarga con 100 usuarios donde se puede ver una disminución en el tiempo de respuesta, esto puede deberse a variaciones en la conexión al servidor o a la administración de recursos del servidor.

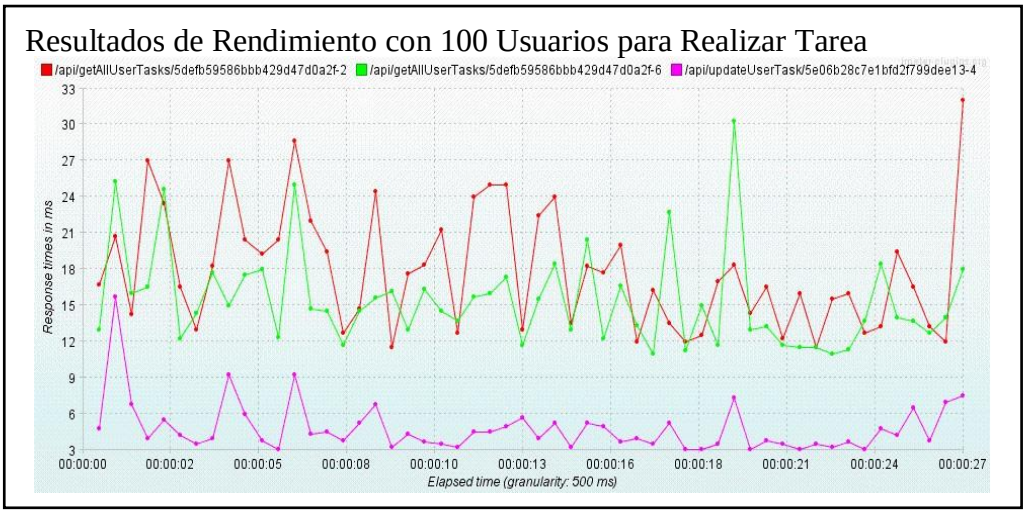


Figura 42 Gráfico resultados pruebas de rendimiento realizar tarea para 100 usuarios
Elaborado por: Carolina Cutiupala y Daniel Toinga

En la figura 42 se evidencia gráficamente que los tiempos de respuesta se mantienen constantes durante toda la prueba, lo que sugiere un mejor rendimiento de la aplicación.

Resultado del rendimiento en el escenario de calificar tarea entre la carga de 27 vs 100 usuarios

Como se puede apreciar en los gráficos el resultado no es el que se podría intuir ya que con una carga de 100 usuarios el servidor responde más rápido. Existen diversos factores que pudieron afectar estos resultados, como la velocidad de la red o procesos internos en el servidor, aun así, el comportamiento del servidor es aceptable ya que no existen bloqueos y los tiempos de respuesta se mantienen en el orden de los milisegundos.

En ambos casos como se muestra en las figuras 39 y 41 el tiempo de respuesta no sobrepaso 1 segundo, por lo tanto, el rendimiento es correcto.

Caso 3

Prueba de rendimiento con 6 usuarios en el escenario de calificar tarea

En la primera prueba se configura la herramienta para simular la carga de 6 usuarios los cuales son usuarios con cargos administrativos.

Rendimiento con 6 Usuarios para Calificar Tarea

Etiqueta	# Muestras	Media	Mediana	90% L...	95% ...	99% L...	Mín	Máx	% Error
/api/getToGradeTasks-154	30	310	295	345	373	448	266	448	0,00%
/api/updateUserTask/5dfa41fa8acbad0364...	30	11	5	10	26	90	3	90	0,00%
Total	60	160	90	331	345	379	3	448	0,00%

Figura 43 Resultados pruebas de rendimiento calificar tarea para 6 usuarios
Elaborado por: Carolina Cutiupala y Daniel Toinga

En la figura 43 se muestra numéricamente los resultados de la prueba de rendimiento para 6 usuarios, la cual presenta un tiempo de respuesta máximo de 448 milisegundos.

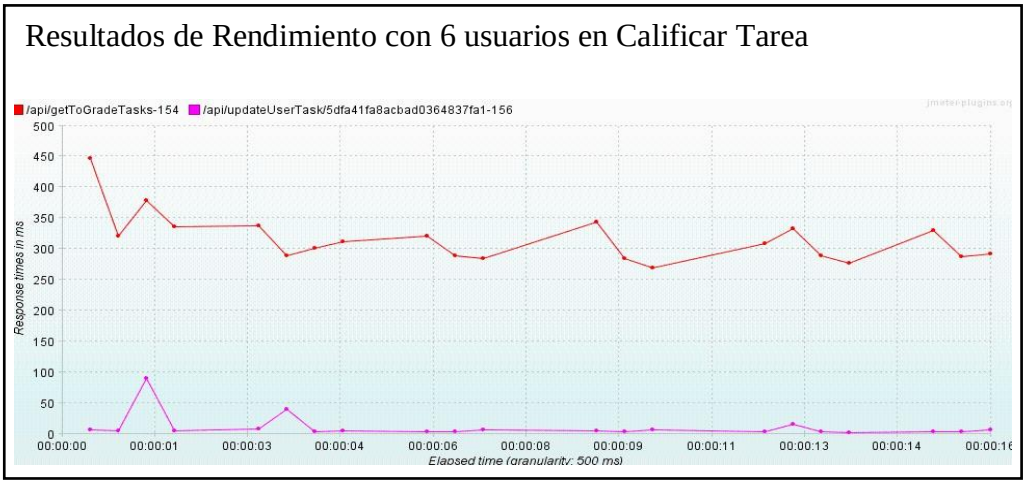


Figura 44 Gráfico resultados pruebas de rendimiento calificar tarea para 6 usuarios
Elaborado por: Carolina Cutiupala y Daniel Toinga

En la figura 44 se evidencia gráficamente que la petición a la URI "/api/getToGradeTasks" tiene un mayor tiempo de respuesta. Esto se debe a este endpoint envía muchos datos al cliente.

Prueba de rendimiento con 18 usuarios en el escenario de calificar tarea

En la segunda prueba se configura la herramienta para simular la carga de 18 que son los usuarios con roles administrativos cada uno son sus 3 docentes de apoyo.

Rendimiento con 18 Usuarios en Calificar Tarea

Etiqueta	# Muestras	Media	Mediana	90% L...	95% ...	99% L...	Mín	Máx	% Error
/api/getToGradeTasks-154	90	292	288	319	324	337	270	400	0,00%
/api/updateUserTask/5dfa41fa8acbad0364837f...	90	9	4	10	24	93	3	94	0,00%
Total	180	150	94	303	319	337	3	400	0,00%

Figura 45 Resultados pruebas de rendimiento Calificar tarea para 18 usuarios
Elaborado por: Carolina Cutiupala y Daniel Toinga

En la figura 45 se muestra numéricamente el resultado de la prueba de rendimiento para 18 usuarios, donde obtiene un tiempo de respuesta máximo de 400 milisegundos.

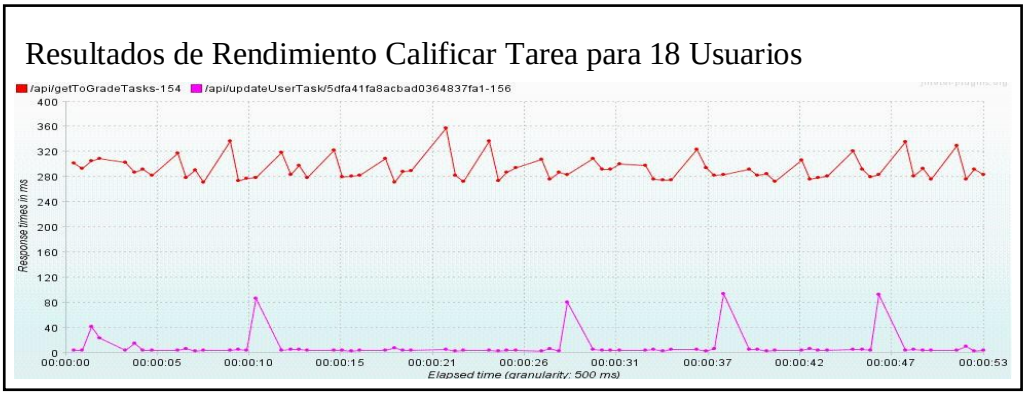


Figura 46 Gráfico resultados pruebas de rendimiento calificar tarea para 18 usuarios
Elaborado por: Carolina Cutiupala y Daniel Toinga

La figura 46 muestra gráficamente que la petición a la URI "/api/getToGradeTasks" sigue siendo la que tiene un tiempo de respuesta mayor.

Resultado del rendimiento en el escenario de calificar tarea entre la carga de 6 vs 18 usuarios

En estos dos escenarios, con 6 usuarios y con 18 usuarios se comprueba que el comportamiento del servidor con NodeJs es el esperado, los tiempos de respuesta no se ven demasiado afectados, el servidor con NodeJs no sufre ningún bloqueo. Cabe resaltar que mientras la carga de datos en la respuesta del servidor sea más elevada, la velocidad de respuesta disminuirá.

En ambos casos el tiempo de respuesta no sobrepaso 1 segundo, por lo tanto, el rendimiento es correcto.

Conclusiones

- Al aplicar la técnica de refactorización se obtuvo una aplicación con mejoras significativas tanto en su funcionalidad como en la reestructurar el código sin afectar su lógica original, en forma complementaria se hizo uso de los frameworks Angular y Express, lo que permitió mantener una mejor estructura en el código.
- El proceso de seguimiento a docentes a travez de la aplicación refactorizada permitirá a las carreras de Sistemas y Computación generar un reporte global del desempeño de los docentes mediante una evaluación basada en criterios y categorías las cuales son calificadas por medio de una ponderación por tarea y para la calificación final se realiza un promedio de todas las tareas asignadas.
- Como resultado de las pruebas del software se concluye que la creación de microservicios en el back-end permite al software mantener estables los tiempos de respuesta, además permite ser más escalable obteniendo así una nueva versión que facilita la integración de aplicación de seguimiento a docentes para la suite de gestión informática.

Lista de referencias

Adobe. (2 de Enero de 2020). *Adobe XD*. Obtenido de Funciones de XD: https://www.adobe.com/la/products/xd/details.html

Alban Soliz, R. (2018). *Utilización de patrones creacionales con TypeScript.* Cochabamba, Bolivia.

Alvarez, M., & Basalo, A. (2018). *Manual de Angular.* León, España: Desarrolloweb6.

Alvarez, M., Alcazar, I., & León, J. (2018). *Manual de Git.* León, España: Desarrolloweb6.

Atlassian. (2020). *Atlassian* . Obtenido de Jira Software: https://www.atlassian.com/es/software/jira

Brown, E. (2014). *Eb development with Node & Express.* Sebastopol, CA: O'REILLY.

Carlos Loor Loor,Javier Oyola Estrad,Nixon, Quezada Sanmartin, Geovanny Mocha Guacho. (2019). *Prototipo de una Aplicación móvil para el diseño de curva de carreteras.* Machala: Centro de convenciones UTMACH.

Castro, M. (28 de Marzo de 2017). *it_info_technology*. Obtenido de LA TECNOLOGÍA OBSOLETA SIGUE SIENDO UN PROBLEMA EN LAS EMPRESAS: https://www.infotechnology.com/negocios/La-tecnologia-obsoleta-sigue-siendo-un-problema-en-las-empresas-20170328-0003.html

Chakray . (20 de Septiembre de 2017). Obtenido de ¿QUÉ ES EL BPMN Y PARA QUÉ SIRVE?: https://www.chakray.com/es/que-es-el-bpmn-y-para-que-sirve/

Chamorro, J. P. (2011). *Diseño del subsistema de evaluación del desempeño por competencias para la institución Fondo Ecuatoriano Populorum Progressio (FEPP) oficina central.* Quito.

Clements, P., Garlan, D., Little, R., Nord, R., & Stafford, J. (2011). *Documenting software architectures: Views and beyond.* Westford, Massachusetts: Pearson Education.

Emanuel Irrazábal ,Cristina Greiner,Gladys Daposo. (2015). *La refactorización de software basada en valor.*

Eustaquio Martín Rodríguez , José Felix Ángulo Rasco, Juan Fernández Sierra,Susana Fernández Larragueta. (2015). *Evaluación de centros y profesores.* Madrid: Universidad Nacional de Educación a Distancia.

Express. (03 de Enero de 2020). *Aplicaciones web*. Obtenido de expressjs.com: https://expressjs.com/es/

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures.* Irvine, US: University of California.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1997). *Design Patterns: Elements of Reusable Object-Oriented Software.* New York, US: Pearson Education.

Iglesias, T. (23 de Mayo de 2019). *El Programador Documental*. Obtenido de Robo 3T – Gestor de bases de datos para MongoDB: http://programadordocumental.es/robo-3t-gestor-de-bases-de-datos-para-mongodb/

Instructivo De Evaluacion 90 Grados. (s.f.).

Kiessling, M. (2015). *The Node Beginner Book.* Victoria, British Columbia, Canada: Leanpub.

Lejá, V. E. (2018). *Evaluación del Desempeño Docente en el Sector Público.* QUETZALT.

Lopez, A. (3 de Junio de 2019). *OpenWebinars*. Obtenido de Qué es Postman y para qué sirve: https://openwebinars.net/blog/que-es-postman/

Martin Fowler, Kent Beck. (2018). *Refactoring.* Melrose, Massachusetts: Pearson Addison.

Medina, J. (2014). *Pruebas de rendimiento TIC.* Murcia: Ediciòn Kindle. Obtenido de Empezando con Apache JMeter: https://riptutorial.com/es/jmeter

Mikowski, M., & Powell, J. (2014). *Single Page Web Applications.* New York, US: Manning Publications Co.

MongoDB. (3 de Enero de 2020). *mongoDB*. Obtenido de mongoDB.es: https://www.mongodb.com/es

MongoDB. (3 de Enero de 2020). *NoSQL Databases Explained*. Obtenido de mongoDb.com: https://www.mongodb.com/nosql-explained

Mora, P. E. (2017). *Desarrollo de una aplicación web para la gestión del seguimiento del cumplimiento de tareas de los docentes de la carrera de ingeníeria en sistemasde la universidad politécnica salesiana.* Quito.

Murray, N., Coury, F., Lerner, A., & Taborda, C. (2018). *ng-Book The complete book on Angular 8.* San Francisco, US: Fullstack.io.

NodeJs. (03 de Enero de 2019). *Acerca*. Obtenido de nodejs.org: https://nodejs.org/es/about/

Noel. (25 de julio de 2016). *lignux*. Obtenido de Dia es una aplicación de diagramas desarrollada por Gnome: https://lignux.com/dia-es-una-aplicacion-de-diagramas-desarrollada-por-gnome/

Osmani, A. (2017). *Learning JavaScript design patterns.* Cambridge, UK: O'REILLY.

Palacio, J. (2015). *Scrum Manager I (las reglas de Scrum).* Scrum Manager.

Software, L. (1 de Enero de 2020). *Lucidchart*. Obtenido de Poder para ver, entender y hacer más.: https://www.lucidchart.com/pages/es

Sommerville, I. (2011). *Ingeniería de software.* Naucalpan de Juárez, México: Pearson Educación.

Sutherland, J. (2016). En *SCRUM: El arte de hacer el doble de trabajo en la mitad del tiempo* (pág. 220 páginas). Océano.

The Apache Software Fundation. (03 de Enero de 1999-2019). *Apache JMeter*. Obtenido de jmeter.apache.org: https://jmeter.apache.org/

Timms, S. (2014). *Mastering JavaScript.* Birmingham, UK: Packt Publishing.

Anexos

Para revisar los anexos del presente proyecto, por favor diríjase al CD.

Anexo A: Sprint y gráfica de seguimiento de tareas.

Anexo B: Reglamento interno de régimen de la Universidad Politécnica Salesiana

Anexo C: Formato de preguntas

Anexo D: Plan de Pruebas

Anexo E: Casos de Pruebas Funcionales

Anexo F: Actas de aceptación

Anexo G: Manual de Instalación

Anexo H: Manual de Usuario

Anexo I: Documento distributivo de los docentes