UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Ingeniería en Sistemas Computacionales

ANÁLISIS DE INTEGRACIÓN ENTRE EL FRAMEWORK FRONTEND ANGULAR Y LA PLATAFORMA JAVA ENTERPRISE COMO BACKEND PARA OPTIMIZAR EL PROCESO DEL TIEMPO DE RESPUESTA AL USUARIO, UTILIZANDO LA CARACTERÍSTICA DE EFICIENCIA DE DESEMPEÑO DE LAS NORMAS ISO/ICE 25010.

Trabajo de Grado previo a la obtención del título de Ingeniera en Sistemas Computacionales

Autor:

Maldonado Arias Tamia Johanna

Director:

Msc. Rea Peñafiel Xavier Mauricio

Ibarra – Ecuador

2023

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DEL AUTOR	
CÉDULA DE IDENTIDAD	1004671010
APELLIDOS Y NOMBRES	MALDONADO ARIAS TAMIA JOHANNA
DIRECCIÓN	OTAVALO – NUEVO SANTIAGUILLO
E-MAIL	tjmaldonado@utn.edu.ec
TELÉFONO MÓVIL	0989742264
DATOS DE LA OBRA	
TÍTULO	ANÁLISIS DE INTEGRACIÓN ENTRE EL FRAMEWORK FRONTEND ANGULAR Y LA PLATAFORMA JAVA ENTERPRISE COMO BACKEND PARA OPTIMIZAR EL PROCESO DE TIEMPO DE RESPUESTA AL USUARIO, UTILIZANDO LA CARACTERÍSTICA DE EFICIENCIA DE DESEMPEÑO DE LAS NORMAS ISO/IEC 25010.
AUTOR	MALDONADO ARIAS TAMIA JOHANNA
FECHA	19/05/2023
PROGRAMA	PREGRADO
TÍTULO POR EL QUE OPTA	INGENIERA EN SISTEMAS COMPUTACIONALES
ASESOR	MSC. XAVIER MAURICIO REA PEÑAFIEL

2. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se lo desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en casos de reclamación por parte de terceros.

Ibarra, al día 19 del mes de mayo de 2023

Firma: Tamia Maldonado A

Nombre: Tamia Johanna Maldonado Arias

Cédula: 1004671010

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN DEL ASESOR

Por medio del presente, yo Msc. Xavier Mauricio Rea Peñafiel, certifico que la Srta. Tamia Johanna Maldonado Arias, portadora de la cédula de identidad Nro. 1004671010, ha trabajado en el desarrollo del proyecto de tesis **"Análisis de integración entre el framework frontend Angular y la plataforma Java Enterprise como backend para optimizar el proceso del tiempo de respuesta al usuario, utilizando la característica de eficiencia de desempeño de la normas ISO/IEC 25010"**, previo a la obtención del título de Ingeniera en Sistemas Computacionales, realizado en su totalidad con responsabilidad.

Es todo cuanto puedo certificar en honor a la verdad.

Firmado electrónicamente por:
XAVIER MAURICIO REA PEÑAPIEL

Msc. Xavier Mauricio Rea Peñafiel

DIRECTOR DE TESIS

DEDICATORIA

A mis padres por su apoyo incondicional a lo largo de mi vida personal y de formación académica.

A mi familia por contribuir con sus experiencias y motivación a la consecución de mis objetivos.

Tamia Maldonado Arias

AGRADECIMIENTO

A mis padres por su incansable confianza, cariño y preocupación por mi bienestar.

Mis sinceros agradecimientos a Msc. Mauricio Rea Peñafiel por su guía como docente y tutor, por su tiempo, esfuerzos y consejos para la culminación de este proyecto. A Msc. Antonio Quiña por sus recomendaciones y observaciones en su calidad de asesor.

A la Universidad Técnica del Norte por brindarme la oportunidad de adquirir conocimientos y valores importantes para la vida.

Un especial agradecimiento a los docentes y personal que forma parte de la Facultad de Ingeniería en Ciencias Aplicadas por compartir sus experiencias con todos los estudiantes que se han crecido profesionalmente dentro de sus instalaciones.

Tamia Maldonado Arias

RESUMEN

La presente investigación tiene como objetivo analizar la integración entre el framework frontend Angular y la plataforma JAVA Enterprise como backend para optimizar el proceso de tiempo de respuesta al usuario, utilizando la característica de eficiencia de desempeño de las normas ISO/IEC 25010.

Estructuralmente el documento contiene una parte introductoria donde se detalla el enfoque del problema con los antecedentes, la situación actual y la prospectiva, también se establecieron objetivos, alcance y justificación para el proyecto propuesto.

En el capítulo uno, se definió la parte teórica del proyecto que consta de conceptos fundamentales de las herramientas tecnológicas y el marco de trabajo utilizado para el desarrollo del sistema.

En el capítulo dos, se desarrolló un servicio web para producir la información de los proyectos del sistema SIAD el cual es consumido por la aplicación web Angular, reemplazando a las pantallas de JSF de JavaEE, se utilizó como marco de trabajo la metodología SCRUM.

En el capítulo tres, muestra la validación de los resultados obtenidos con la ejecución de los dos tipos de pantalla, se utilizó el estándar ISO / IEC 25010 como guía de evaluación.

Finalmente, se detallaron las conclusiones y recomendaciones generadas en la elaboración de proyecto.

ABSTRACT

The present research aims to analyze the integration between Angular as a frontend framework and JAVA Enterprise Edition Platform as a backend to optimize the user response time process, using the performance efficiency characteristic of the ISO / IEC 25010 standards.

Structurally, the document contains an introductory part where the approach to the problem is detailed with background, current situation; prospective, objectives, scope and justification for the proposed project were also established.

On chapter one, the theoretical part of the project was defined, which consists of fundamental concepts of the technological tools and the process framework used for the development of the system.

Chapter two, a web service was developed to produce the information from SIAD System which is consumed by an Angular web application, replacing the JavaEE JSF screens, also SCRUM methodology was used as a process framework.

Chapter three shows the validation of the results obtained with the use of the two kinds of screens, the ISO / IEC 25010 standard was used as an evaluation guide.

Finally, conclusions and recommendations generated in the preparation of the project were detailed.

CONTENIDO

INTRODUCCIÓN

Antecedentes

Las aplicaciones empresariales almacenan y manejan una cantidad considerable de datos por lo cual deben ser diseñadas, construidas y producidas mediante la integración de sistemas heterogéneos y distribuidos, previendo el escalamiento y la reutilización de sus componentes para reducir el esfuerzo considerado en su codificación y el uso de recursos (Oracle, 2014).

Los usuarios necesitan acceder a la información constantemente ya sea para visualizarla o efectuar modificaciones por lo tanto las aplicaciones deben reaccionar de manera inmediata a cualquier petición realizada, proporcionando datos actualizados e íntegros.

Actualmente los sistemas y aplicaciones desarrolladas en la plataforma Java Enterprise Edition no muestran que los métodos de sus componentes se activen de manera automática ante las actualizaciones producidas en la información por parte de los usuarios (Nieto Lemus, 2015).

Situación Actual

Los sistemas creados en la plataforma JEE no cuentan con pantallas que brinden las características que ofrece la programación reactiva principalmente la actualización automática y el soporte de fallos, por su parte Angular ofrece estas funcionalidades a través de la librería RxJS, permitiendo que los sistemas sean responsivos, resilientes, elásticos y orientados a mensajes (Thompson, Bóner, Farley, & Kuhn, 2014).

La programación reactiva es un conjunto de directrices de cómo se debe programar un sistema para trabajar con datos que se desconocen cuando se generan, pero que se espera reaccionar y actuar en consecuencia (Ollivier, 2016). En la actualidad existen varios lenguajes en los que se utiliza el paradigma de la programación reactiva como una extensión, entre los cuales se destaca la implementación RxJS, para JavaScript.

Prospectiva

El presente análisis ofrece documentación relativa a la integración entre el framework frontend Angular y la plataforma JavaEE como backend, logrando optimizar el despliegue de información al usuario, cumpliendo con la característica de eficiencia

de desempeño del estándar ISO/IEC 25010, que representa el desempeño relativo a la cantidad de recurso utilizados bajo determinadas condiciones (ISO/IEC 25000, 2019).

Planteamiento del problema

Los métodos de los componentes de la plataforma JavaEE muestran un bajo índice de reacción ante la modificación de la información generada por el usuario, debido a que en una aplicación clásica de JavaEE existe un claro vínculo entre una solicitud y un hilo. El hilo de la solicitud espera y bloquea más hilos hasta que la llamada vuelva al sistema mediante la respuesta esperada, ocasionando una demora significativa entre solicitudes. JavaEE puede proporcionar mecanismos para procesar secuencias, eventos o mensajes, pero para lograrlo necesita de una serie de API adicionales que el sistema actual no utiliza, es por ello que la programación reactiva ofrece ventajas sobre este tipo de prácticas (Limburg, 2019).

Debido al alto número de usuarios concurrentes y el alto nivel de actualización de datos se ocasiona un mayor consumo de recursos y no brinda un constante flujo de datos manteniendo en ciertos momentos la vista o el modelo de datos desactualizados. Además, al presentarse situaciones de error, los fallos no son manejados de manera adecuada y el usuario podría no visualizar la información que busca.

En la Fig. 1 se representa el árbol de problemas que permite identificar el problema.

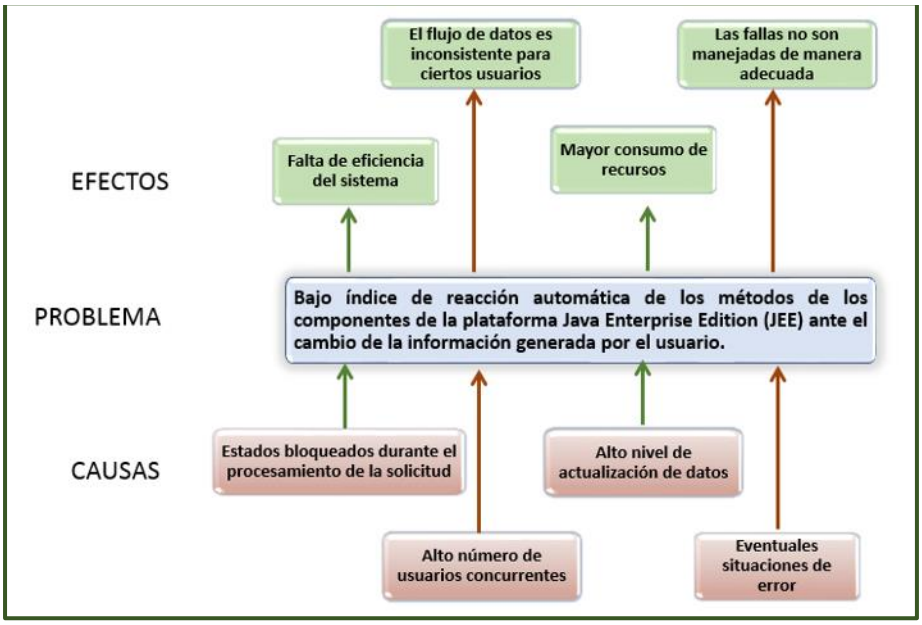


Fig. 1: Árbol de problemas
Fuente: Propia

Objetivo General

Analizar la integración entre el framework frontend Angular y la plataforma Java Enterprise como backend para optimizar el proceso del tiempo de respuesta al usuario utilizando la característica de eficiencia de desempeño de las normas ISO/IEC 25010.

Objetivos Específicos

1. Elaborar un marco teórico del tema de investigación para exponer sus características y cualidades.
2. Implementar en el módulo de Proyectos del Sistema SIAD (Sistema Integrado de Actividades Docentes) varias pantallas utilizando el framework Angular e integrándolas con los componentes EJB de la aplicación JavaEE, con base en SCRUM como marco de trabajo de desarrollo de software.
3. Verificar los resultados mediante la característica de eficiencia de desempeño de las normas ISO/IEC 25010.

Alcance

La implementación de las pantallas creadas con el framework frontend Angular en el Módulo de Proyectos, optimiza el proceso de tiempo de respuesta al usuario y se generó documentación que respalda el análisis entre Angular y la plataforma JavaEE como backend.

Las pantallas del Módulo de Proyectos están desarrolladas con las siguientes tecnologías:

- Arquitectura Web Service Rest
- Framework Frontend Angular
- Lenguaje de programación Java Enterprise
- IDE Eclipse 2019-06
- Servidor de aplicaciones Wildfly v14.0
- GitHub.

Se utilizó para verificar los resultados, la característica de eficiencia de desempeño de las normas ISO/ICE 25010.

El marco de trabajo SCRUM permitió obtener resultados de manera rápida gracias a la definición de iteraciones que se orientan principalmente al flujo de comunicación entre los diferentes roles (Blokehead, 2016).

Justificación

El presente proyecto se enfoca en los siguientes Objetivos de Desarrollo Sostenible planteados por la ONU y UNESCO:

9b. Apoyar el desarrollo de tecnologías, la investigación y la innovación nacionales en los países en desarrollo, incluso garantizando un entorno normativo propicio a la diversificación industrial y la adicción de valor a los productos básicos, entre otras cosas (Naciones Unidas, 2019).

9c. Aumentar significativamente el acceso a la tecnología de la información y las comunicaciones y esforzarse por proporcionar acceso universal y asequible a Internet en los países menos adelantados de aquí a 2020 (Naciones Unidas, 2019).

Social

El desarrollo del presente análisis permite que se amplíe la documentación en temas referentes a la integración entre las plataformas JEE y Angular, proporcionado guías para el desarrollo futuro.

Tecnológico

Las herramientas tecnológicas están en constante evolución por lo que es imperativo actualizar los conocimientos y optimizar el desarrollo de aplicaciones y sistemas web (Garzás & Piattini, 2015).

CAPÍTULO 1

1. MARCO TEÓRICO

1.1. Integración de sistemas

La evolución de la tecnología ha permitido que existan diversas herramientas para el desarrollo de sistemas y aplicaciones web, obligando a quienes se involucran en estas actividades a mantener una constante actualización de conocimientos, ya que hoy en día es común encontrar sistemas que además de utilizar sus recursos, utilizan o comparten dichos recursos con programas externos.

Este tipo de situación se da también al momento de empezar con el diseño de un nuevo aplicativo, ya que la combinación de varias herramientas permite obtener mejores resultados de los que se conseguirían al regirse a un solo tipo de herramienta. Uno de los principales diseños que ayudan a lograr la colaboración entre sistemas son las arquitecturas Web Service.

1.1.1. Arquitectura Web Service REST

Actualmente se construyen soluciones basadas en arquitecturas REST. Estas arquitecturas soportan varios niveles de madurez (Álvarez Caules, 2018):

Nivel 0: Swamp of POX (Plain Old XML)

Se caracteriza por la utilización del método POST de HTTP para el intercambio de mensajes en formato XML entre el cliente y el servidor a través de ULRs de acceso, genera mensajes complejos por lo que es necesario poseer los ficheros WSDL para construir el cliente.

Nivel 1: Recursos

Un recurso se refiere a un elemento que forma parte de los objetos de negocio, que al ser publicado permite su inserción, eliminación, actualización y búsqueda. No establece un estándar para la asignación de nombres de URL, por lo que cada recurso puede presentar distintas denominaciones para cada operación.

Nivel 2: Verbos Http

Su principal aporte fue la convención concreta a nivel de nombre de URLs y los verbos HTTP que se usan para cada operación:

- **GET:** Obtiene los datos:
  /proyectos obtendrá la lista completa de Proyectos (Recursos),
  /proyectos/32 obtendrá la lista de proyectos que corresponda al usuario que corresponda el identificador señalado.
- **POST:** Inserta una nueva entrada en un listado.
- **PUT:** Actualiza los datos indicados.
- **DELETE:** Borra la entrada correspondiente al identificador especificado.

Nivel 3: **Hypermedia as the Engine of Application State (HATEOAS)**

Se caracteriza porque los Recursos pueden estar relacionados entre ellos a través del uso de links, lo que permite que, al solicitar una lista, esta pueda incluir links a los recursos relacionados a cada uno de sus datos.

1.1.2. Aplicaciones Frontend

Es la parte de del software que interactúa con los usuarios (Echazú & Rodríguez, 2018), permitiendo que naveguen dentro de la página web. Es conocida también como el lado del cliente, incluye todo lo que se ve en la pantalla cuando se accede a una aplicación web, desde el tipo de letras, colores, adaptación para distintas pantallas, efectos de ratón, teclado, movimientos, desplazamiento y efectos visuales (Chapaval, 2018).

Las características principales que debe cumplir para crear una agradable experiencia del usuario son: sencilla de usar, atractiva y funcional.

1.1.3. Aplicaciones Backend

Es la capa de acceso a datos, se encarga de la seguridad y rendimiento interno de las aplicaciones que se hallan en el servidor. Consiste en un servidor, una aplicación y una base de datos, se toman los datos, se procesa la información y se envía la respuesta al usuario (Chapaval, 2018).

1.1.4. Ventajas

Al utilizar una arquitectura de desarrollo basada en API REST, se obtienen una clara separación entre el cliente y el servidor, facilitando la realización de cambios en cada uno de ellos con la condición de que aun proporcionen las utilidades para la que se crearon, además las aplicaciones de software frontend no solo se limitarían a sitios web, ya que pueden expandirse a aplicaciones móviles en diferentes tecnologías y lenguajes (Alvarez, Ruso, Ruiz, & Segura, 2014).

API REST brinda también libertad a la hora de crear nuevas utilidades tanto en frontend como en el backend sin afectar la estructura de cada uno, en cuanto a la experiencia del cliente, se muestra una mejoría pues únicamente recibe información plana, lo que brinda un tiempo de transmisión menor, en la mayoría de casos se obtiene un menor consumo de recursos físicos (Alvarez, Ruso, Ruiz, & Segura, 2014).

1.2. Plataforma de Programación Java Enterprise

Java Platform, Enterprise Edition (Java EE) es el estándar en software empresarial impulsado por la comunidad, se desarrolla utilizando el Proceso de la Comunidad Java, con contribuciones de expertos de la industria, organizaciones comerciales y de código abierto, grupos de usuarios de Java e innumerables personas. Cada versión ensambla nuevas características que se alinean con las necesidades de la industria, mejora la portabilidad de las aplicaciones y aumenta la productividad del desarrollador (Oracle and/or its affiliates, 2017).

JSF es un framework de desarrollo web Java orientado a componentes del lado del servidor, que permite reducir el tiempo en el proceso de desarrollo y de mantenimiento de aplicaciones web empresariales, también se destaca por ofrecer un enfoque personalizable y estandarizado para crear interfaces de aplicaciones de usuario. Proporciona una forma estándar para la resolución de problemas que aparecen con frecuencia durante el desarrollo de aplicaciones web, tales como validación, navegación, creación de plantillas y flujos de páginas. (Saleh, Christensen, & Wadia, 2013)

Arquitectura de JSF

La arquitectura de JSF se basa en el popular patrón MVC (Model View Controller).

- **Modelo,** está representado por los beans y el código de fondo. El bean es el que contiene la lógica del negocio y en el que se realizan las operaciones necesarias.
- **Vista,** es la tecnología de renderización de JSF que define el diseño de la página y el contenido.
- **Controlador,** está representado por el Servlet Faces, el cual es responsable de manejar el envío de la solicitud y la navegación de las páginas.

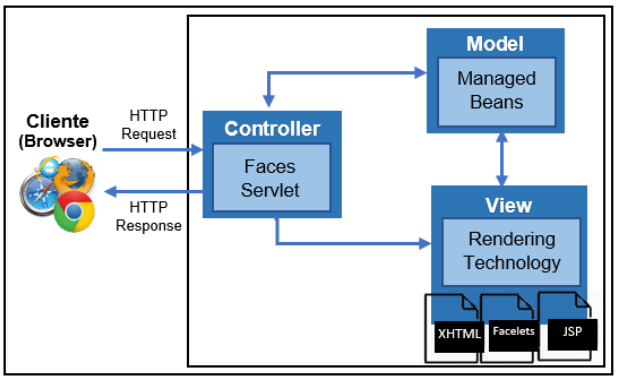


Fig. 2: Arquitectura de JSF - MVC
Fuente: (Saleh et al., 2013)

Ciclo de vida de JSF

El ciclo de vida es una secuencia de fases por las cuales pasa una petición desde que se recibe en el servidor hasta que se genera el resultado que es una página HTML. El ciclo de vida de procesamiento de solicitudes de JSF tiene seis fases, en donde cada una de estas fases tiene un objetivo que cumplir (Amstrong, y otros, 2005) (Leiva, Geancarlo, 2014):

- **Restore View** (Vista de restauración): aquí se obtiene el árbol de componentes que corresponde a la vista JSF de la petición.
- **Apply Request Values** (Aplicar valores de solicitud): cuando se obtiene el árbol de componentes se procesan los valores asociados al mismo.
- **Process Validations** (Validaciones de procesos): se validan todos los datos que ingresan, en caso de haber un error se termina el ciclo y salta a la última fase.
- **Update Model Values** (Actualizar los valores del modelo): una vez que los valores se han procesado y validado, se actualizan las propiedades de los beans gestionados asociados a los componentes.

- **Invoke Application** (Invocar la aplicación): al completar la actualización de los valores se llama a esta fase, la cual ejecuta el código de acción.
- **Render Response** (Respuesta de render): muestra los resultados finales al usuario.

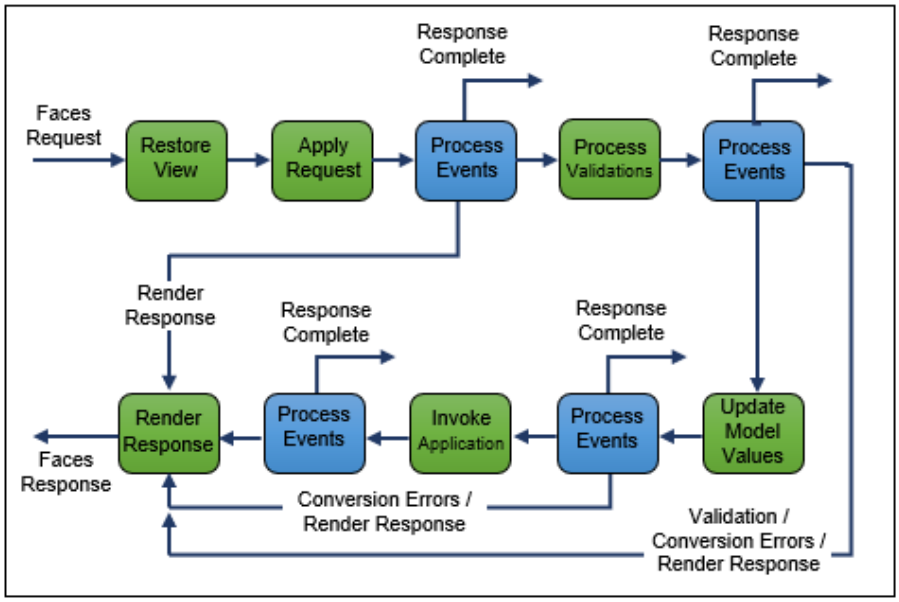


Fig. 3: Ciclo de vida de JSF
Fuente: (Stojanov, Simoncev, Pesov, & Mitreski, 2009)

- **Integración JSF:** Facelets está centrada en JSF, es compatible con todos sus componentes básicos y funciona dentro del ciclo de vida.
- **Composición:** Facelets compila un árbol de componentes JSF marcando uno o varios archivos en un solo árbol.
- **Plantillas:** Facelets admite la reutilización de los diseños de vistas en archivos de plantilla, utilizándolos en diferentes páginas.

1.3. Framework frontend Angular

Es un framework para la creación de aplicaciones web basadas en SPA (Single Page App) que son dinámicas, escalables y reactivas.

Angular es una plataforma que facilita la creación de aplicaciones web. Angular combina plantillas declarativas, inyección de dependencia, herramientas de extremo a extremo y mejores prácticas integradas para resolver los desafíos de desarrollo. Angular permite a los desarrolladores crear aplicaciones que se ejecutan en la web, el dispositivo móvil o el escritorio. La mayoría del código angular se puede escribir con el JavaScript más reciente, con tipos de inyección de dependencia y con decoradores para metadatos (Google, 2010-2020).

1.3.1. Elementos

- Componentes. - Controla un trozo de la pantalla o de la vista.
- Plantillas Two-way data binding. - Cuentan con un sistema en el que la vista y el controlador están en constante relación lo que simplifica el desarrollo, este sistema permite que todo cambio que realice en la vista se actualice en tiempo real en el modelo y viceversa (Arizmendi, 2018).
- Decoradores. – Patrones de diseño que configuran dinámicamente los atributos o metadatos de las clase y componentes.
- Metadatos. – Describen las clases y sus relaciones.
- Servicios. – Facilitan la reutilización de código,
- Providers. – Son servicios que brindan funcionalidades desde los métodos.
- Directivas. - Consisten en marcadores en un elemento de DOM (Modelo de Objetos del Documento) que indican al compilador de Angular que dicho elemento tiene un comportamiento especifico, gracias a esto, se puede trabajar fácilmente a nivel de componentes, siendo estos componentes reutilizables en toda la aplicación (CreativeCommons, 2014).

1.4. Programación Reactiva

1.4.1. Librería Reactive eXtensions

ReactiveX es una biblioteca para componer programas asíncronos y basados en eventos mediante el uso de secuencias observables.

Extiende el patrón observador para admitir secuencias de datos y/o eventos y agrega operadores que le permiten componer secuencias de manera declarativa al tiempo que abstrae las preocupaciones sobre cosas como subprocesos de bajo nivel, sincronización, seguridad de subprocesos, estructuras de datos concurrentes y no bloqueo de dispositivos de entrada y salida (ReativeX, 2019).

1.5. Normas ISO/ICE 25010

Este modelo de calidad establece el sistema para la evaluación de la calidad del producto. Detalla las características de calidad que se van a tener en cuenta a la hora de evaluar las propiedades de un producto software determinado.

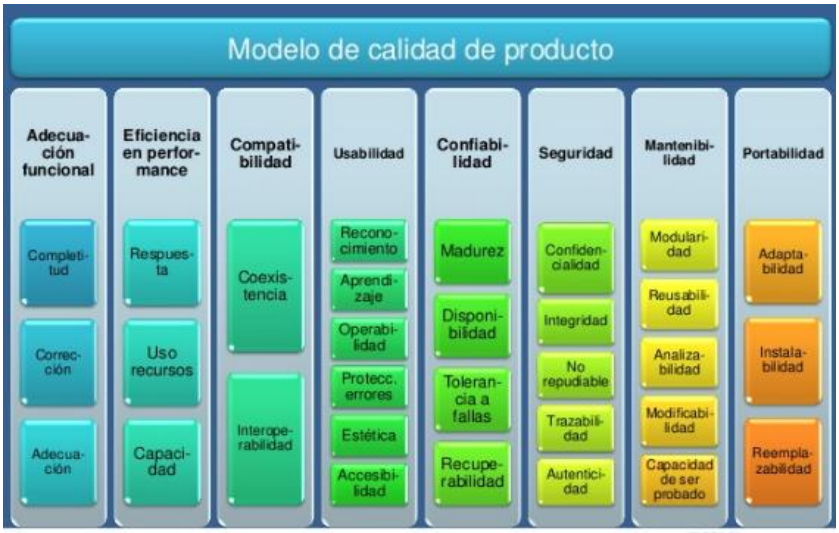


Fig. 4: Modelo de calidad de producto
Fuente: (Martínez, 2015)

1.5.1. Característica de eficiencia de desempeño

Es la característica que permitirá evaluar la eficiencia de la utilización del framework Angular para el desarrollo de las pantallas en el proyecto, ya que representa el desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones. Esta se divide en las siguientes subcaracterísticas (ISO/IEC 25000, 2019):

- **Comportamiento temporal. -** Los tiempos de respuesta y procesamiento de un sistema cuando lleva a cabo sus funciones bajo condiciones determinadas en relación con un banco de pruebas (benchmark) establecido.
- **Utilización de recursos. -** Las cantidades y tipos de recursos utilizados cuando el software lleva a cabo su función bajo condiciones determinadas.
- **Capacidad. -** Grado en que los límites máximos de un parámetro de un producto o sistema software cumplen con los requisitos (ISO/IEC 25000, 2019).

1.6. Marco de trabajo SCRUM

Scrum es un marco de trabajo que implementa valores y principios ágiles, es utilizado para desarrollar, entregar y realizar el mantenimiento de productos de software complejos, se centra en la colaboración en equipo para el cumplimiento de proyectos que ofrecen valor a los clientes de manera rápida, para lo cual tiene como base la creación de ciclos breves para el desarrollo, reduciendo el riesgo y costo, obteniendo una pronta retroalimentación por parte de los usuarios. (Drumond, Cook, & West, 2018).

Como uno de los principales modelos de gestión ágil, Scrum, cumple con los objetivos de brindar valor, reducir el tiempo de salida al mercado, agilidad, flexibilidad y resultados fiables. Además, recorre las fases del ciclo de desarrollo ágil (Palacio & Ruata, 2009):

- **Concepto:** Se crea la visión y las características del producto, informando al equipo que se encargará de su desarrollo.
- **Especulación:** en esta fase se realizan hipótesis con la información obtenida de la visión del proyecto y se establecen los límites que marcaran su desarrollo, tales como costes y agendas, se comprobaran las partes realizadas y su impacto en el entorno. Esta fase se repite en cada iteración y consiste en:
  - Desarrollar y revisar los requisitos generales.
  - Mantener la lista de las funcionalidades que se esperan.
  - Mantener el Plan de entrega, estableciendo las fechas para las versiones, hitos e iteraciones.
- **Exploración:** Se incrementa el producto en el que se añaden las funcionalidades de la fase de especulación.
- **Revisión:** El equipo revisa todo lo que se ha construido y se contrasta con el objetivo deseado.
- **Cierre:** Al final de las fechas especificadas se obtiene el producto esperado, el cual puede aún necesitar ciclos incrementales para alcanzar la visión original.

En el desarrollo del proyecto intervienen varias personas que cumplen un rol que describe sus funciones principales (Palacio M. , 2021):

- **Propietario del producto** (Product Owner)

  Es la persona que tiene conocimiento experto sobre el entorno de negocio, toma las decisiones del cliente referentes al valor del producto, se encarga de desarrollar y administrar la lista del producto (Product Backlog) manteniendo una interacción frecuente con el equipo

- **Equipo de Desarrollo** (Development Team)

  Es el grupo de profesionales que se encargan del desarrollo del proyecto, poseen las habilidades necesarias para entregar cada incremento del producto final cumpliendo la meta en cada sprint.

- **Scrum Master**

  Se encarga de supervisar el proceso del proyecto, aplicando el marco de trabajo SCRUM, actúa como moderador en las reuniones diarias entre el propietario del producto y el equipo, asesorando y brindando la formación.

Las herramientas que permiten el desarrollo del proyecto se llaman Artefactos, destacan tres, que son clave para el funcionamiento del marco de trabajo (Palacio M. , 2021):

- **Pila de producto** (Product Backlog)

  Está compuesto por el conjunto de requisitos expresados por el cliente, quien también designa la prioridad de cada uno. Cada requisito se denomina **historia de usuario**, la cual se subdivide en tareas de menor tamaño.

- **Pila de Sprint** (Sprint Backlog)

  Son los requisitos desde el punto de vista del equipo, las tareas a realizar durante cada sprint, son la representación visual de avance, ya que contiene el nombre del responsable y el estado de las tareas.

- **Incremento**

  Es el resultado de cada sprint que se encuentra en condiciones de ser entregado al cliente.

Dinámica del Marco de trabajo SCRUM

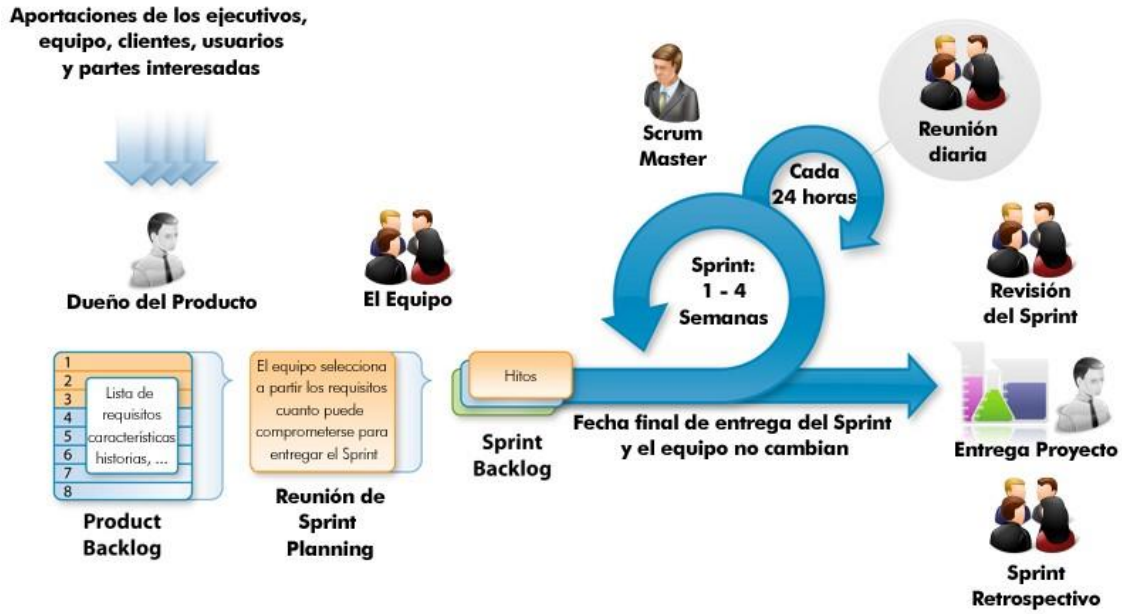


Fig. 5: Dinámica del Marco de trabajo SCRUM
Fuente: (ISLA VISUAL, 2012)

CAPÍTULO 2

El módulo Proyectos permite tener acceso a la información relacionada a los proyectos vinculados al usuario que inicio sesión en el sistema, también los detalles de los objetivos y actividades relacionados con dichos proyectos, además los usuarios que forman parte de cada proyecto.

2. DESARROLLO

Implementación de pantallas Angular
en el Módulo de Proyectos del Sistema SIAD

Para la integración de las pantallas utilizando el framework frontend Angular se crearon varios DTOS en la aplicación backend y el Web Service REST que será consumido por la aplicación frontend en Angular. Para la obtención de datos estadísticos se realizó un monitoreo al servidor a través de la configuración de JAVA Melody.

2.1. Requisitos del proyecto

Las nuevas pantallas frontend basadas en Angular para el módulo de proyectos se enfocan en la presentación de los listados de proyectos del usuario; objetivos, actividades e integrantes a través del uso de web services REST disminuyendo el uso de recursos en el lado de cliente, al acceder en una única consulta no solo a la información de un proyecto sino de todos a la vez, por lo que comparado con las pantalla de JavaEE, representa una mejora del tiempo en la presentación de detalles del proyecto.

2.1.1. Historias de usuario

HISTORIA DE USUARIO	
Número: 1	**Usuario:** Cliente
Nombre historia: DTOs en la aplicación backend.	
Prioridad: Alta	**Riesgo:** Alta
Programadora: Srta. Tamia Maldonado	**Estimación:** 8
Descripción: Como cliente requiero que se mapeen los datos de las entidades involucradas en la presentación de la información de los proyectos del sistema SIAD.	
Pruebas de aceptación:	
Los DTO permitirán retornar los datos de los proyectos, objetivos y actividades según el usuario especificado.	

Tabla. 2: Historia de Usuario Nro. 1
Fuente: Propia

Se recolectó la historia de usuario DTOs en la aplicación backend (Tabla 2) para logras la transmisión de datos entre el cliente y el servidor, mediante datos serializables que no contienen operaciones de negocio u operaciones sobre los datos. Lo que permitirá una fácil manipulación por parte de web service en la aplicación web de Angular.

HISTORIA DE USUARIO	
Número: 2	**Usuario:** Cliente
Nombre historia: Web services REST en el backend.	
Prioridad: Alta	**Riesgo:** Alta
Programadora: Srta. Tamia Maldonado	**Estimación:** 8
Descripción: Como cliente requiero que un servicio GET permita buscar la información correspondiente a los proyectos vinculados con un usuario específico.	
Pruebas de aceptación:	
El web service recibirá un identificador de usuario para desplegar la información requerida.	

Tabla. 3: Historia de Usuario Nro. 2
Fuente: Propia

Se recolectó la historia de usuario Web services REST en el backend (Tabla 3) para que invoque al método que realiza la consulta de los datos asociados a usuario autenticado en el sistema. El servicio get proporciona los datos serializados en el formato de texto estándar JSON.

HISTORIA DE USUARIO	
Número: 3	**Usuario:** Cliente
Nombre historia: Aplicación Frontend Angular	
Prioridad: Alta	**Riesgo:** Alta
Programadora: Tamia Maldonado	**Estimación:** 8
Descripción: Como cliente requiero que una aplicación Angular consuma la información generada por el web service de proyectos.	
Pruebas de aceptación:	
La aplicación mostrará la información en forma de listas.	

Tabla. 4: Historia de Usuario Nro. 3
Fuente: Propia

Se recolecto la historia de usuario Aplicación Frontend Angular (Tabla 4) para la visualización ordenada de los datos contenidos en el archivo JSON, es decir los proyectos en marcha de los que el usuario es responsable, actúa como validador o es integrante, también los proyectos finalizados en los que participó.

HISTORIA DE USUARIO	
Número: 4	**Usuario:** Cliente
Nombre historia: Configuración de Java Melody	
Prioridad: Alta	**Riesgo:** Alta
Programadora: Tamia Maldonado	**Estimación:** 8
Descripción: Como cliente requiero obtener datos estadísticos que muestren el tiempo de respuesta al usuario.	
Pruebas de aceptación: En la ejecución de la aplicación se mostrarán los datos generados por Java Melody, así como también en la dirección raíz de la aplicación + /monitoring	

Tabla. 5: Historia de Usuario Nro. 4
Fuente: Propia

Se recolectó la historia de usuario Configuración de Java Melody (Tabla 5) para contar con una herramienta que permita la recolección estadística de la ejecución del sistema web.

HISTORIA DE USUARIO	
Número: 5	**Usuario:** Cliente
Nombre historia: Tiempos de ejecución y uso de memoria	
Prioridad: Alta	**Riesgo:** Alta
Programadora: Tamia Maldonado	**Estimación: 8**
Descripción: Como cliente requiero que se verifiquen los tiempos de ejecución y el uso de memoria utilizados por la aplicación.	
Pruebas de aceptación: Los datos estadísticos obtenidos durante la utilización de las pantallas Angular se compararán con los obtenidos con la aplicación JEE y sus pantallas JSF.	

Tabla. 6: Historia de Usuario Nro. 5
Fuente: Propia

Se recolectó la historia de usuario Tiempos de ejecución y uso de memoria (Tabla 6) para visualizar la diferencia entre los dos tipos de pantalla durante la invocación y presentación de la información.

2.1.2. Product Backlog

ID	PRIORIDAD	HISTORIA	ESTIMACIÓN
HU1	ALTA	DTOs en la aplicación backend.	8 horas
HU2	ALTA	Web services REST en el backend.	8 horas
HU3	ALTA	Aplicación Frontend Angular	8 horas
HU4	ALTA	Configuración de Java Melody	4 horas
HU5	ALTA	Tiempos de ejecución y uso de memoria	8 horas

Tabla. 7: Product Backlog
Fuente: Propia

2.1.3. Roles del Proyecto

Persona	Descripción	Rol
Msc. Mauricio Rea	Director del presente trabajo de tesis, Docente de la Universidad Técnica del Norte	Jefe Proyecto (Scrum Master)
Tamia Maldonado	Tesista	Equipo de desarrollo (Development Team)

Tabla. 8: Roles del Proyecto
Fuente: Propia

2.2. Planificación de desarrollo

2.2.1. Desarrollo de Sprints

Sprint: 0
Total, horas: 21:00
Fecha Inicio SP1: 09/Diciembre
Fecha Final SP1: 20/Diciembre

Historia de usuario	Desarrollador	Fase Desarrollo	Tarea	Tipo	Tiempo Estimado (Horas)	Tiempo Real (Horas)	Estado
Matriz de planificación	Tamia Maldonado	Planificación	Formalización de la matriz de planificación	Nueva	2:00	1:00	HECHO
			Organización y análisis de los documentos para los Sprint 0 - 1 - 2	Nueva	1:00	1:00	HECHO
Acta de constitución	Tamia Maldonado	Desarrollo	Recolección de información del proyecto	Nueva	0:20	0:10	HECHO
			Identificación de Propósito y Justificación del proyecto	Nueva	0:30	0:20	HECHO
			Descripción del Proyecto y Entregable	Nueva	0:30	0:30	HECHO
			Descripción de Requerimientos de alto nivel (Requisitos de producto y proyecto)	Nueva	1:00	1:00	HECHO
			Cronograma de hitos principales	Nueva	0:30	0:20	HECHO
			Definición de presupuesto estimado	Nueva	0:30	0:20	HECHO
			Lista de Interesados (stakeholders)	Nueva	0:20	0:10	HECHO
			Identificación de Requisitos de aprobación del proyecto	Nueva	0:30	0:30	HECHO
			Asignación de Personal y recursos preasignados	Nueva	0:10	0:10	HECHO

Especificación de requerimientos	Tamia Maldonado	Desarrollo	Desarrollo de la parte introductoria (Propósito - Alcance - Personal Involucrado - Definiciones, Abreviaturas - Resumen)	Nueva	1:00	0:30	HECHO
			Desarrollo de la descripción general (Perspectiva del Producto - Funcionalidad del Producto - Características de los usuarios - Restricciones - Suposiciones y dependencias - Evolución del Sistema)	Nueva	1:00	0:40	HECHO
			Análisis de los requisitos funcionales	Nueva	0:30	0:30	HECHO
			Análisis de los requisitos no funcionales	Nueva	0:20	0:20	HECHO
			Análisis de los requerimientos específicos y requisitos de Interfaz (De Usuario, De Hardware, De Software, De Comunicación)	Nueva	1:00	1:25	HECHO
Cartillas de historias de usuario	Tamia Maldonado	Desarrollo	Creación de las historias de usuario: Integración de Angular con Proyecto JavaEE	Nueva	5:00	4:00	HECHO
Backlog de historias de usuario	Tamia Maldonado	Desarrollo	Llenado de la matriz de Historias de Usuario: Integración de Angular con Proyecto JavaEE	Nueva	5:00	4:00	HECHO
			TOTAL, SPRINT		21:00	16:55	

Tabla. 9: Planificación de desarrollo Sprint 0
Fuente: Propia

Sprint:	1
Total, horas:	15:00
Fecha Inicio SP2:	23/Diciembre
Fecha Final SP2:	27/Diciembre

Historia de usuario	Desarrollador	Fase Desarrollo	Tarea	Tipo	Tiempo Estimado (Horas)	Tiempo Real (Horas)	Estado
Diagrama conceptual	Tamia Maldonado	Desarrollo	Análisis del Diagrama Conceptual: Modulo de Proyectos	Nueva	2:00	1:00	HECHO
Arquitectura de software	Tamia Maldonado	Desarrollo	Desarrollo de la parte introductoria y descripción de la parte arquitectónica.	Nueva	1:00	0:10	HECHO
			Análisis de la arquitectura y definición de las herramientas y tecnologías a utilizarse		1:00	0:20	HECHO
Instalación y configuración de herramientas a usar	Tamia Maldonado	Desarrollo	Instalación de PostgreSQL, Servidor de Aplicaciones WildFly, IDE de desarrollo Eclipse, entre otros	Nueva	3:00	2:00	HECHO
			Configuración de WildFly y JBoss Tools en eclipse	Nueva	1:00	0:30	HECHO
			Configuración de WildFly 14 en eclipse IDE	Nueva	1:00	0:30	HECHO
Creación y configuración del proyecto	Tamia Maldonado	Desarrollo	Creación del proyecto en el Workspace	Nueva	1:00	0:30	HECHO
			Configuración de JPA y JSF	Nueva	1:00	0:30	HECHO
			Estructuración de carpetas para el proyecto	Nueva	1:00	0:45	HECHO
			Importación de librerías a utilizar	Nueva	1:00	0:15	HECHO
			Mapeo de modelo de Base de Datos al proyecto	Nueva	1:00	1:00	HECHO
			TOTAL, SPRINT		14:00	7:00	

Tabla. 10: Planificación de desarrollo Sprint 1

Fuente: Propia

Sprint:	2
Total, horas:	36:00
Fecha Inicio SP3:	30/Diciembre
Fecha Final SP3:	31/Enero

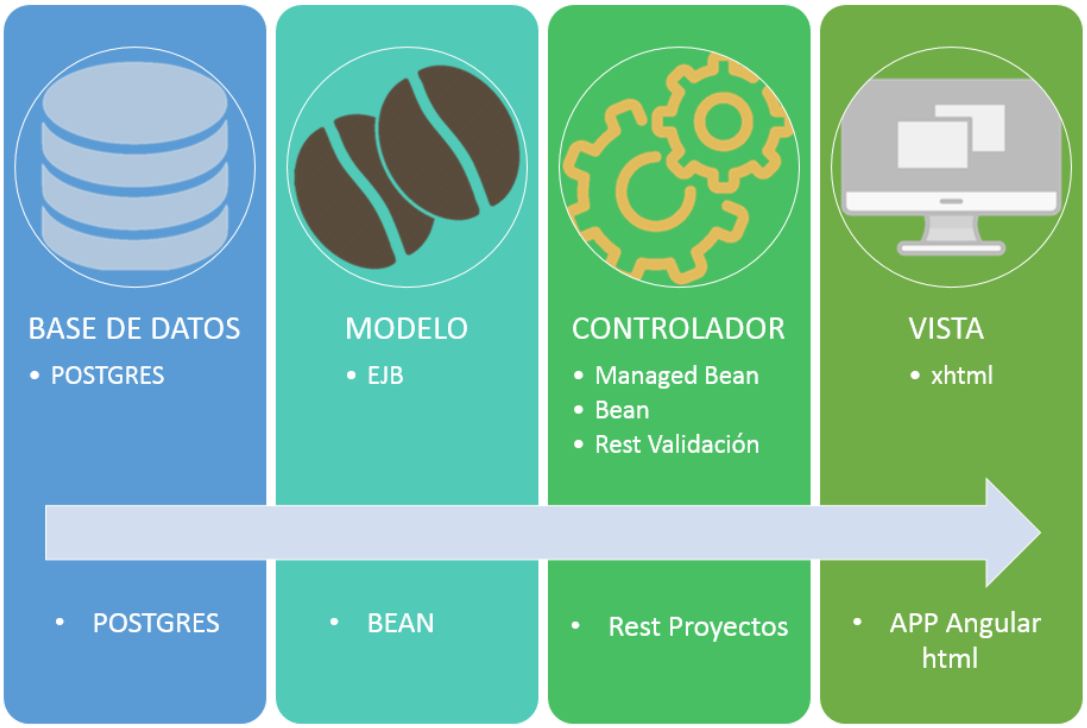
Historia de usuario	Desarrollador	Fase Desarrollo	Tarea	Tipo	Tiempo Estimado (Horas)	Tiempo Real (Horas)	Estado
Diseño y codificación general	Tamia Maldonado	Desarrollo	Creación de DTOs en la aplicación backend	Nueva	8:00	5:00	HECHO
			Creación de web services REST en el backend		8:00	2:00	HECHO
			Creación de aplicación frontend en Angular para consumir servicios		15:00	10:00	HECHO
Instalación y configuración de herramientas a usar	Tamia Maldonado	Desarrollo	Configuración de Javamelody para obtener datos estadísticos	Nueva	4:00	1:00	HECHO
Plan de pruebas	Tamia Maldonado	Desarrollo	Verificación de tiempos de ejecución y uso de memoria	Nueva	8:00	8:00	HECHO
Informe de Plan de pruebas	Tamia Maldonado	Desarrollo	Presentación de resultados	Nueva	3:00	1:00	HECHO
			TOTAL, SPRINT		36:00	27:00	

Tabla. 11: Planificación de desarrollo Sprint 2
Fuente: Propia

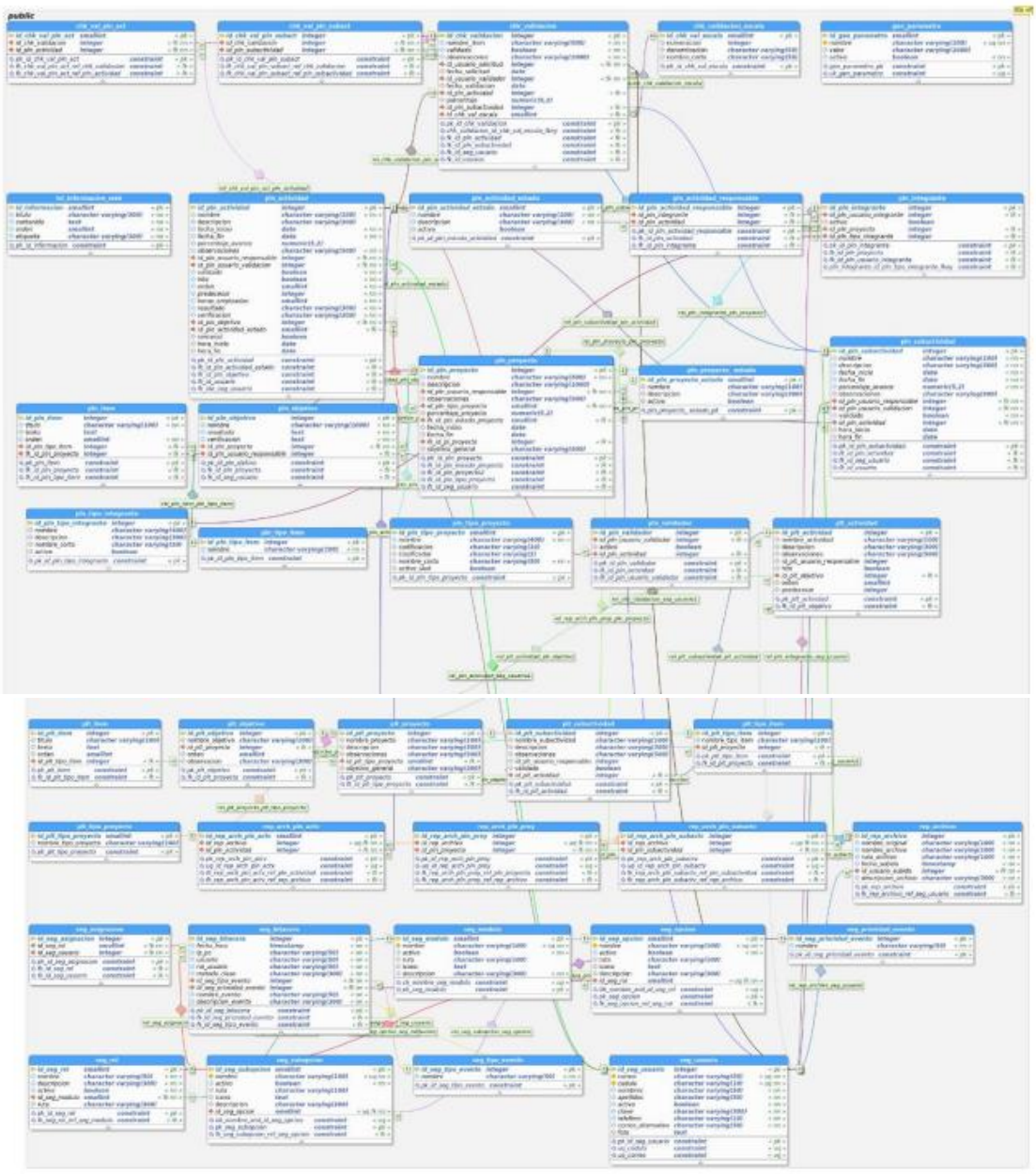
2.2.2. Seguimiento del producto potencialmente entregable.

- **Arquitectura**

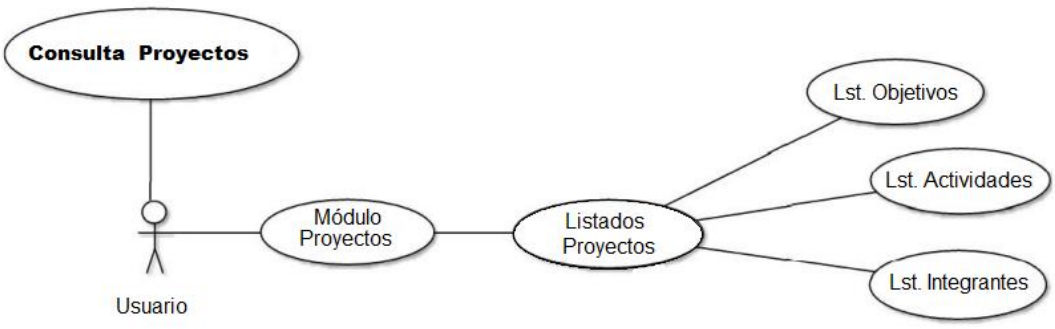
El desarrollo del sistema se encuentra enfocado en la arquitectura de software MVC (Modelo, Vista, Controlador) donde Modelo contiene los datos, Vista se encarga de la presentación al cliente de los datos requeridos y Controlador se encarga de lógica de programación.



- **Base de datos**



- **Casos de uso**



El usuario del sistema tiene acceso para visualizar los proyectos en los que participa y los objetivos, actividades e integrantes de cada uno.

2.3. Codificación

2.3.1. Creación de DTOs en la aplicación backend

Patrón DTO aplicado a las entidades que intervienen en la Planificación de Proyectos:

Un patrón de diseño proporciona técnicas probadas que ayudan a resolver problemas comunes en el desarrollo web. El patrón DTO, (Data Transfer Object), es decir el objeto de transferencia de datos es un objeto plano que contiene información de una o más entidades de la aplicación del servidor para ser transportada hacia otra aplicación o la capa de presentación, disminuyendo el número de llamadas realizadas al servidor (Monday, 2003).

ENTIDADES **DTOs**

Fig. 6: Comparación entre entidades y DTOs
Fuente: Propia

Existe un mayor número de clases mapeadas desde las tablas de la base de datos en comparación al número de clases DTOs, porque estos son creados con los atributos requeridos por el web service agrupando la información contenida en varias entidades, por ejemplo:

DTOs	**Incluye datos de:**
PlnActividadDTO	PlnActividad, PlnActividadEstado, PlnActividadResponsable
PlnIntegranteDTO	PlnIntegrante, SegUsuario1, SegUsuario2

PnObjetivoDTO	PlnObjetivo
PlnProyectoDTO	PlnProyecto, PlnTipoProyecto, PlnProyecto_estado

Tabla. 122: Relaciones entre datos
Fuente: Propia

De acuerdo con el esquema de la base de datos del proyecto se puede visualizar las relaciones existentes entre las tablas:

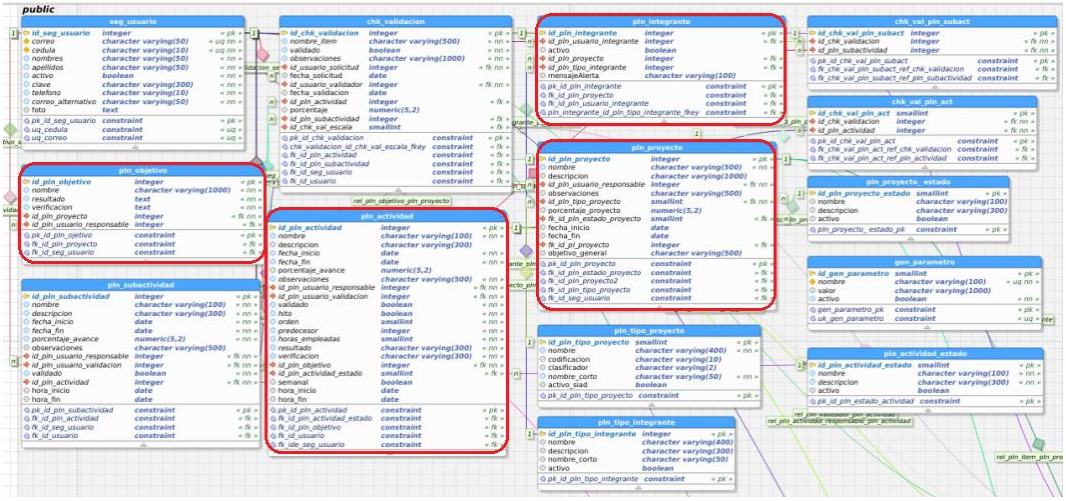


Fig.7: Base de Datos SIAD
Fuente: Propia

En las entidades estas relaciones se representan mediante las **asociaciones bidireccionales uno-a-muchos** (OneToMany), entre la entidad independiente a las entidades dependientes.

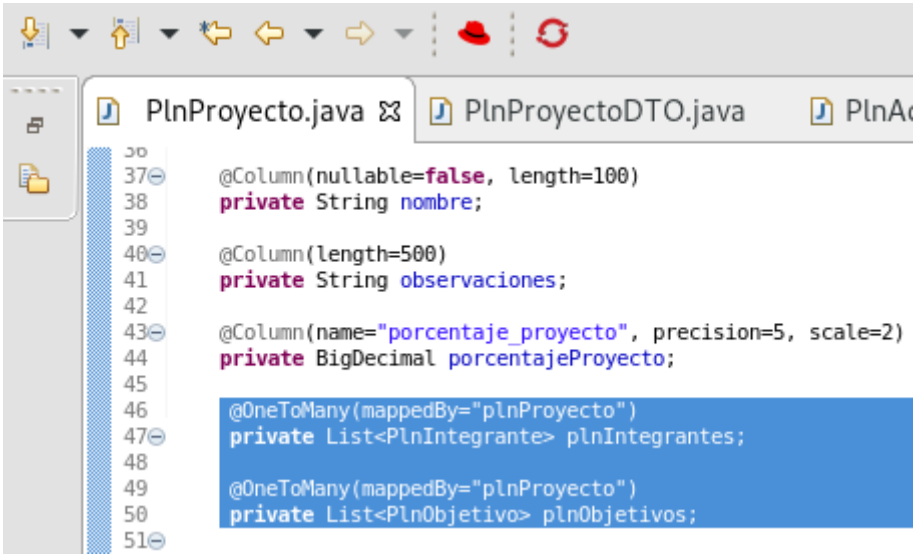


Fig.8: Relación Entidades: Proyecto-Integrante y Proyecto-Objetivo
Fuente: Propia

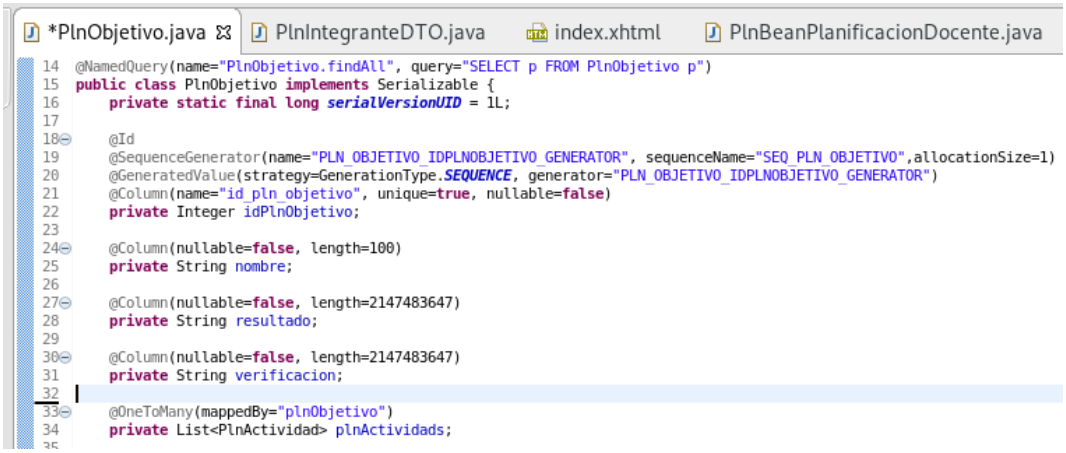


Fig.9: Relación Entidades: Objetivo-Actividad
Fuente: Propia

Y **asociación bidireccional muchos a uno** a la entidad independiente.

Fig.10: Relación Entidades: Integrante-Proyecto
Fuente: Propia

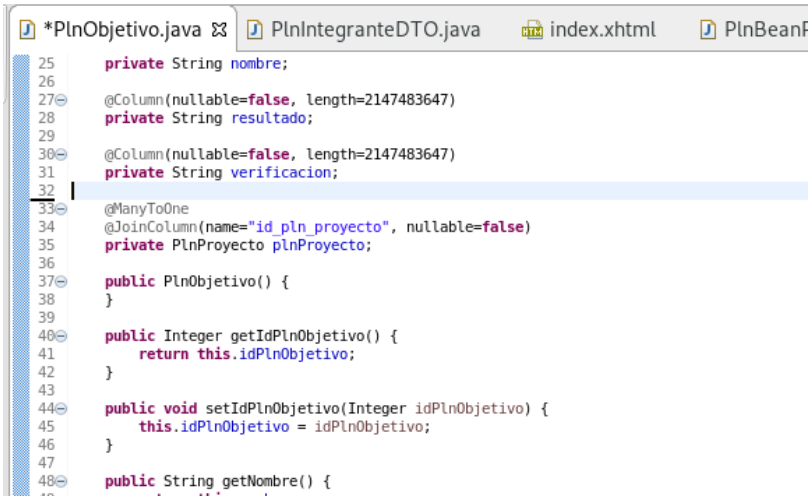


Fig.11: Relación Entidades: Objetivo-Proyecto
Fuente: Propia

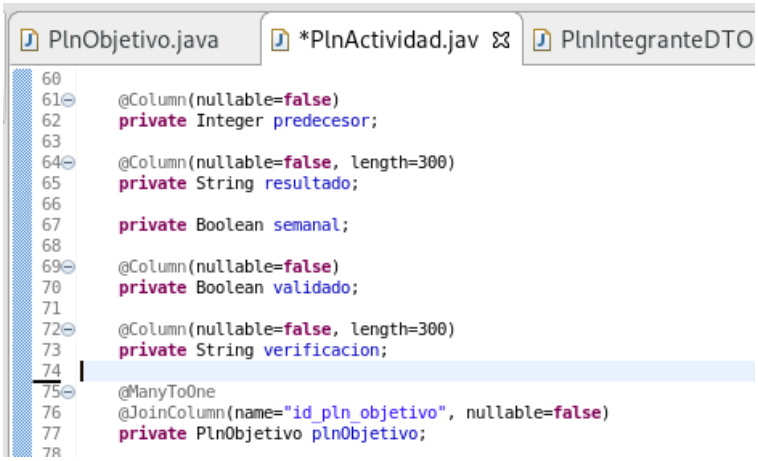


Fig.12: Relación Entidades: Actividad-Objetivo
Fuente: Propia

En los DTO se representa mediante una colección de datos tipo List.

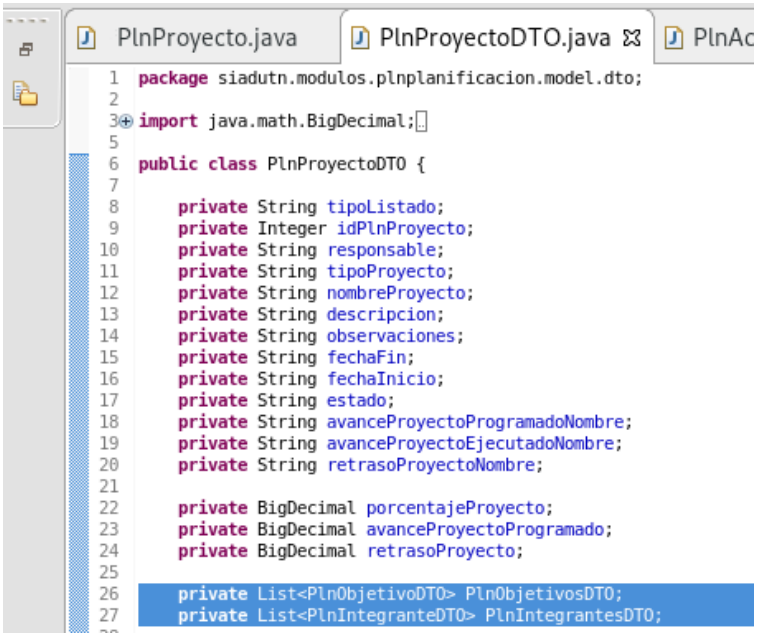


Fig.13: Relación DTO Proyecto-Integrante y Proyecto-Objetivo
Fuente: Propia

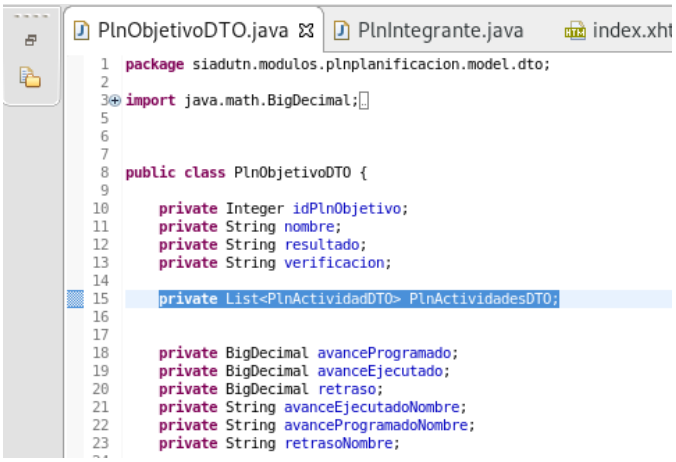


Fig.14: Relación DTO Objetivo-Actividad
Fuente: Propia

Un DTO debe ser de solo lectura, evitando métodos u operaciones distintas a get y set sobre sus atributos y serializable para evitar discrepancias con los tipos de datos.

ENTIDAD	DTO
	

*Tabla. 13: Atributos, tipo de atributos y métodos: Entidad P**lnProyecto** vs. DTO **PlnProyectoDTO***
Fuente: Propia

2.3.2. Creación de web services REST en el backend

El web service creado en la clase **servicioREST.java**, produce un **archivo json** que contiene una lista completa de los proyectos, objetivos y actividades relacionadas con el usuario actual del sistema, a través de la ruta especificada en las anotaciones @Path: **"…/consulta/proyectos"**

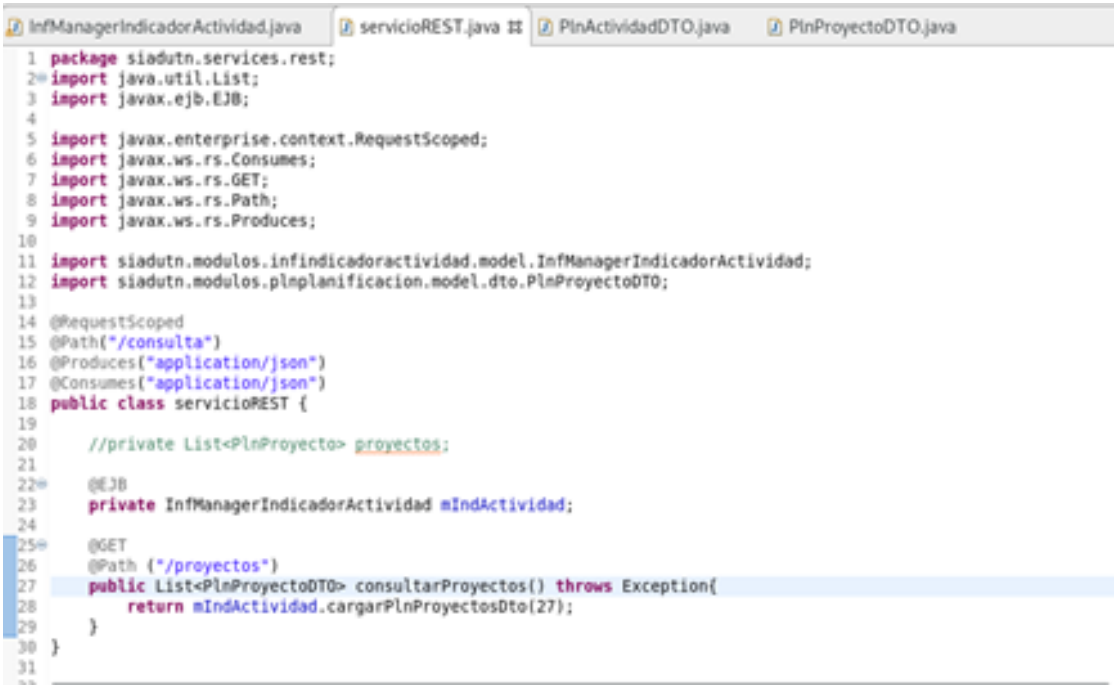


Fig. 15: Creación de Web Service REST
Fuente: Propia

Archivo JSON resultante.

Tabla: 14: Archivo Json como resultado de la consulta realizada
Fuente: Propia

2.3.3. Creación de aplicación frontend en Angular para consumir servicios Frontales JavaEE

- **Listados de Proyectos**

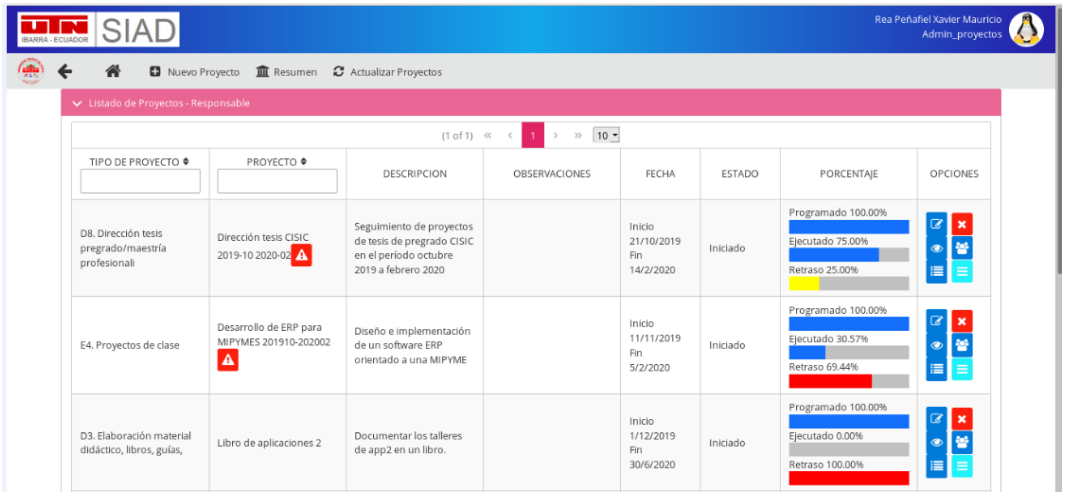


Fig. 16: Frontal "Proyectos" del Sistema SIAD
Fuente:

- **Integrantes del proyecto seleccionado**

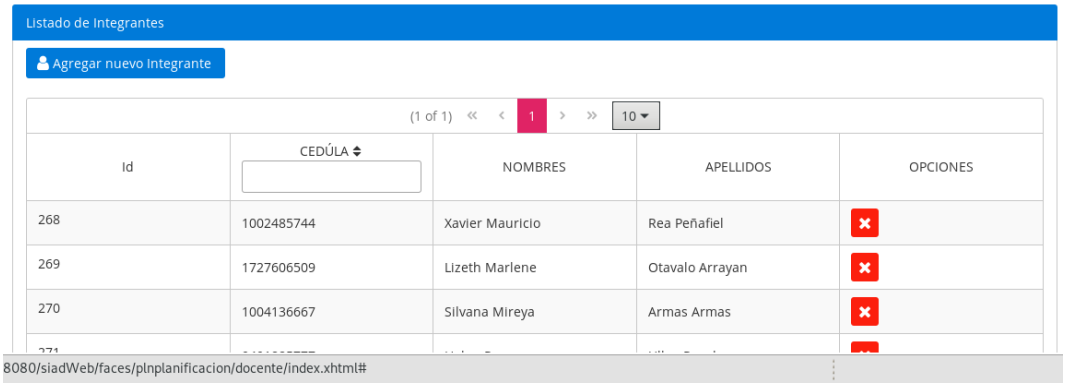


Fig. 17: Frontal "Integrantes" del Sistema SIAD
Fuente:

- **Objetivos del proyecto seleccionado**

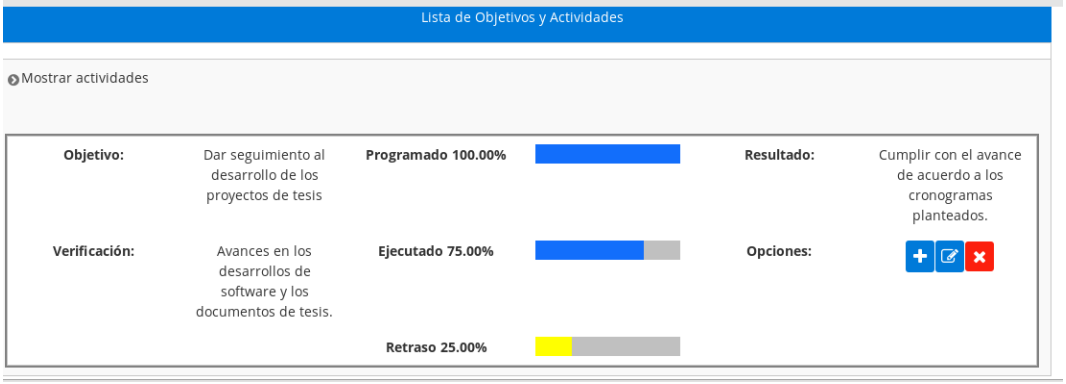


Fig. 18: Frontal "Objetivos" del Sistema SIAD
Fuente:

- **Actividades del objetivo seleccionado**

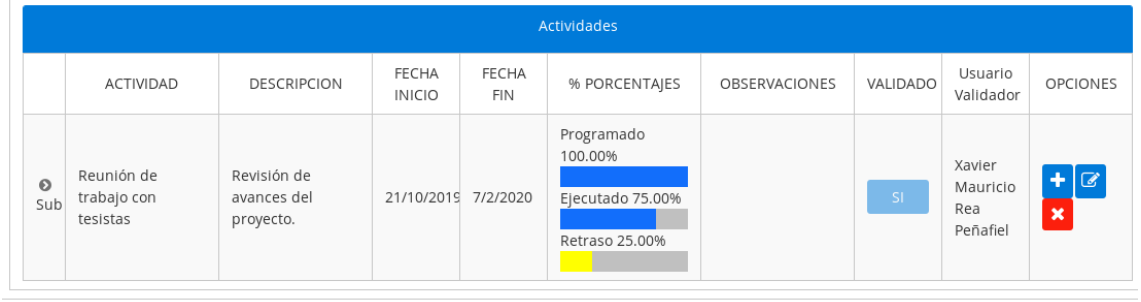


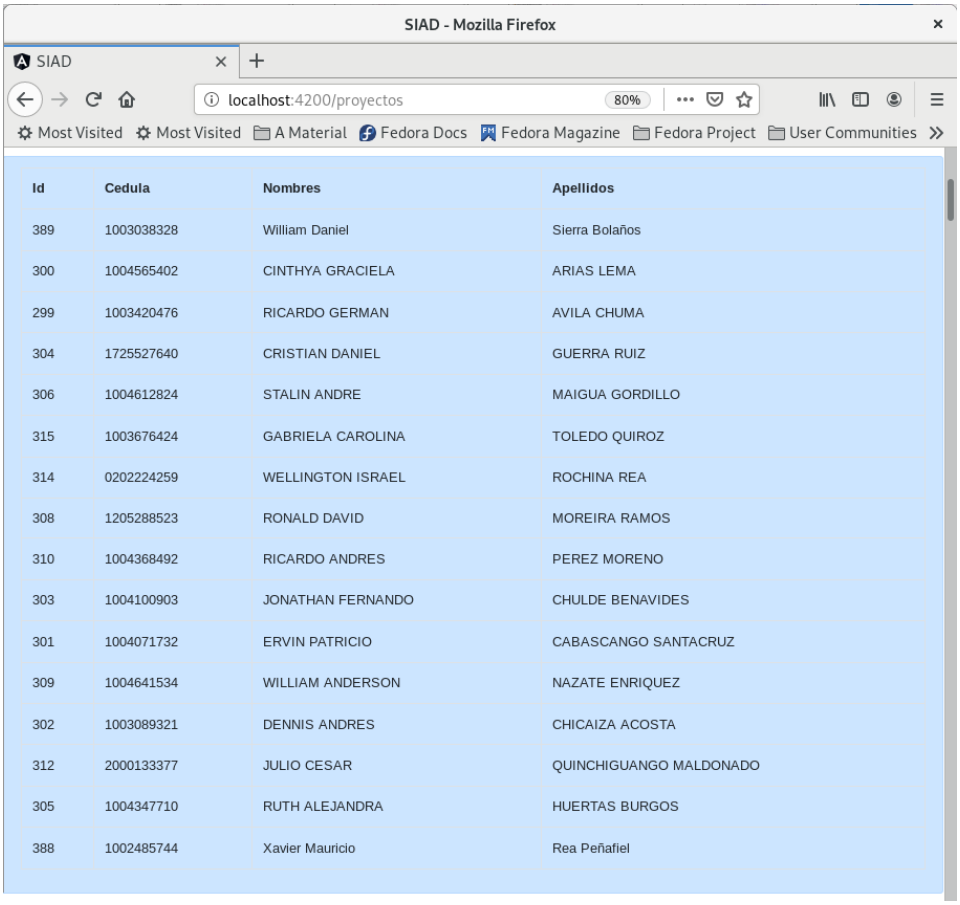
Fig. 19: Frontal "Actividades" del Sistema SIAD
Fuente:

Frontales Angular

- **Listados de Proyectos**

Fig. 20: Frontal "Proyectos" – Aplicación Angular
Fuente: Propia

- **Integrantes del proyecto seleccionado**



Id	Cedula	Nombres	Apellidos
389	1003038328	William Daniel	Sierra Bolaños
300	1004565402	CINTHYA GRACIELA	ARIAS LEMA
299	1003420476	RICARDO GERMAN	AVILA CHUMA
304	1725527640	CRISTIAN DANIEL	GUERRA RUIZ
306	1004612824	STALIN ANDRE	MAIGUA GORDILLO
315	1003676424	GABRIELA CAROLINA	TOLEDO QUIROZ
314	0202224259	WELLINGTON ISRAEL	ROCHINA REA
308	1205288523	RONALD DAVID	MOREIRA RAMOS
310	1004368492	RICARDO ANDRES	PEREZ MORENO
303	1004100903	JONATHAN FERNANDO	CHULDE BENAVIDES
301	1004071732	ERVIN PATRICIO	CABASCANGO SANTACRUZ
309	1004641534	WILLIAM ANDERSON	NAZATE ENRIQUEZ
302	1003089321	DENNIS ANDRES	CHICAIZA ACOSTA
312	2000133377	JULIO CESAR	QUINCHIGUANGO MALDONADO
305	1004347710	RUTH ALEJANDRA	HUERTAS BURGOS
388	1002485744	Xavier Mauricio	Rea Peñafiel

Fig. 21: Frontal "Integrantes" – Aplicación Angular
Fuente: Propia

- **Objetivos del proyecto seleccionado**

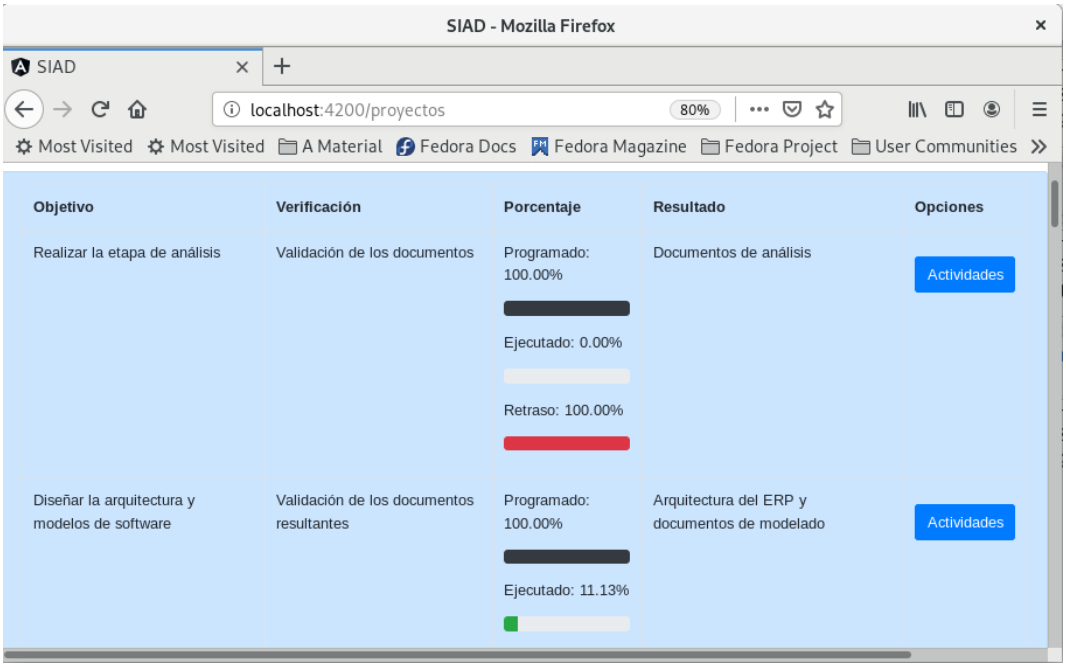


Fig. 22: Frontal "Objetivos" – Aplicación Angular
Fuente: Propia

- **Actividades del objetivo seleccionado**

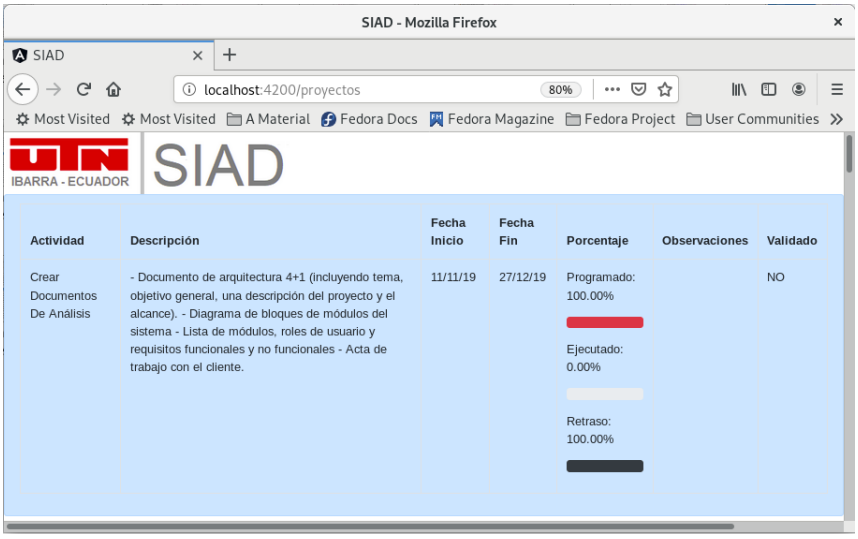


Fig. 23: Frontal "Actividades" – Aplicación Angular
Fuente: Propia

2.4. Configuración de Javamelody para obtener datos estadísticos

JavaMelody permite medir y calcular estadísticas de una aplicación para realizar optimizaciones y detección temprana de tendencias malas alertando al desarrollador acerca de la toma de decisiones, además gestiona estadísticas de solicitudes y genera gráficas relacionadas a la utilización de la aplicación por parte de los usuarios (GitHub, Inc., 2020).

Su instalación se realiza mediante la copia de dos archivos: **javamelody.jar** y **jrobin-1.5.9.jar** en el directorio **WEB-INF/lib** de la aplicación web que va a ser supervisada.

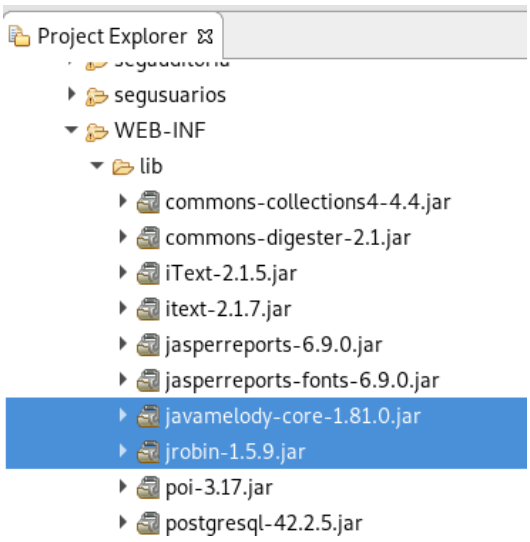


Fig. 24: Archivos "jar" de JavaMelody
Fuente: Propia

Y la modificación del archivo **web.xml,**

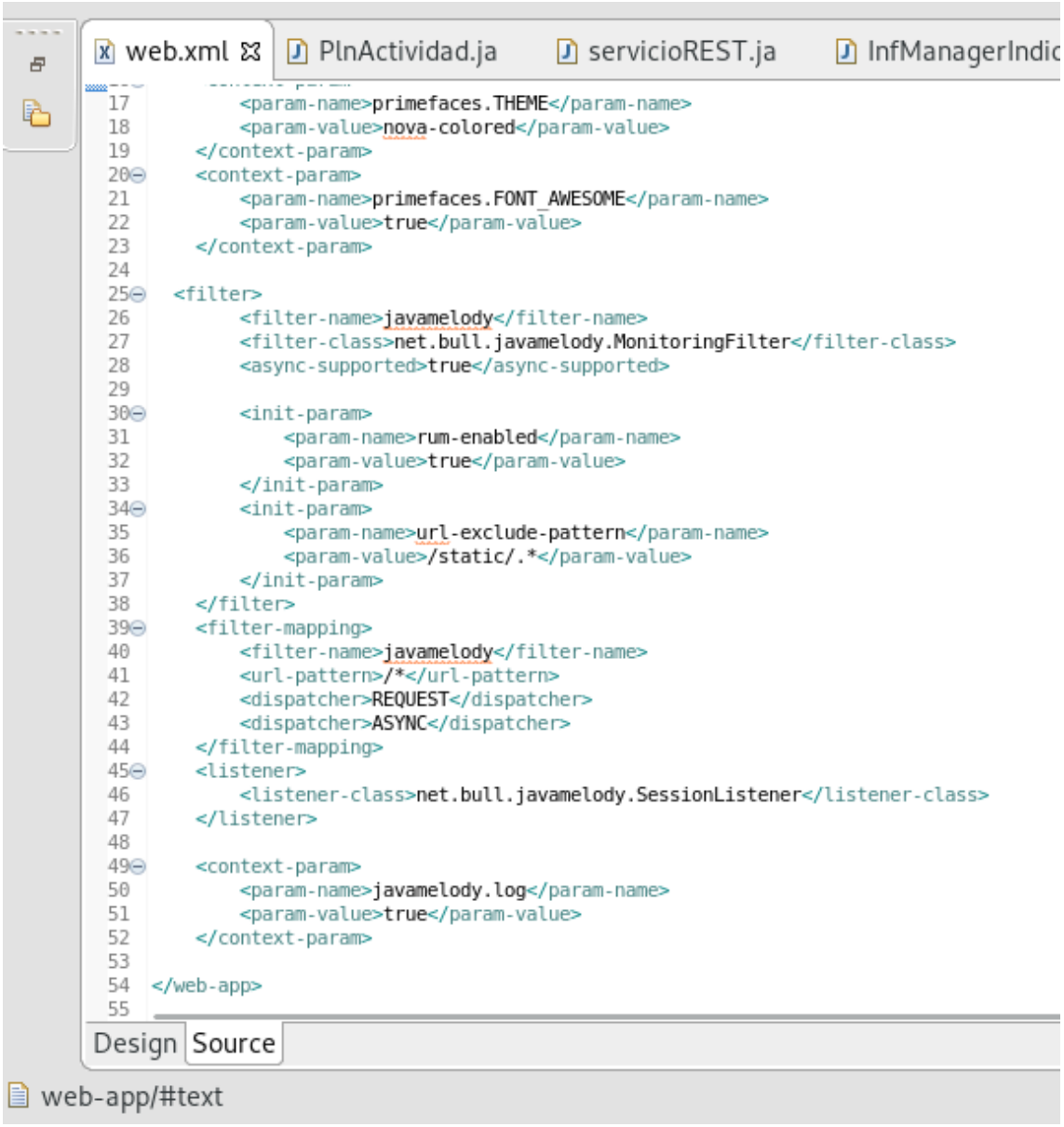


Fig. 25: Archivo "web-xml"
Fuente: Propia

2.5. Pruebas/Evaluación

2.5.1. Verificación de uso de memoria y CPU

Durante la ejecución de las peticiones GET para la consulta de la información a desplegar a las pantallas.

Para visualizar los datos obtenidos con Javamelody se accede a la dirección de la aplicación /**../siadWeb/monitoring**:

Fig. 26: Url de la barra de navegación – Aplicación JAVA EE
Fuente: Propia

- **Utilización de la memoria**

Las estadísticas del consumo de memoria durante la ejecución de las consultas para ambos frontales: un promedio de 248 M, con resultado entre 230 M a 271 M.

Fig. 27: Uso de memoria – Java Melody
Fuente: Propia

- **Uso del procesador**

Las estadísticas del consumo de CPU durante la ejecución de las consultas para ambos frontales: un promedio de 942 ms, con resultado entre 595 ms a 2437 ms

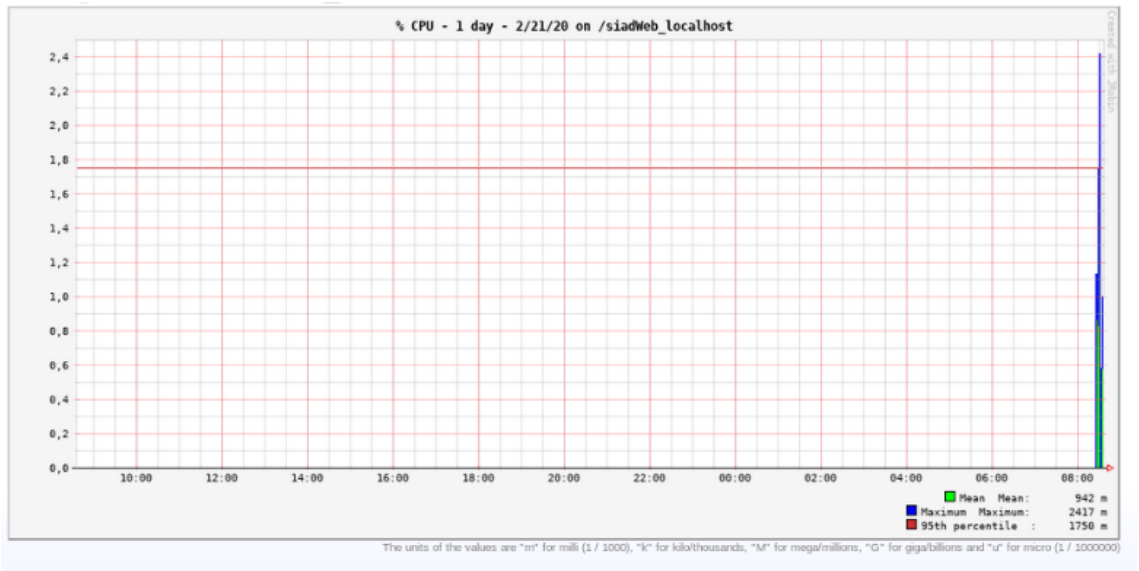
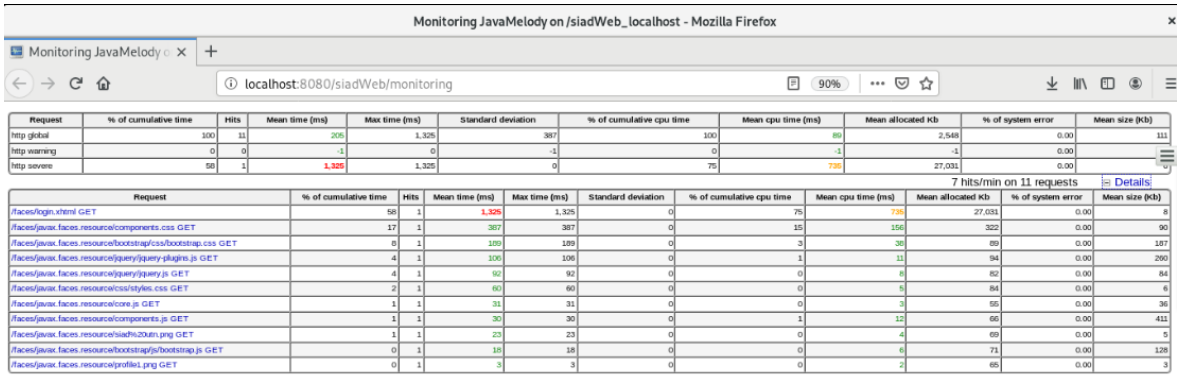


Fig. 28: Consumo de Procesador – Java Melody
Fuente: Propia

2.5.2. Verificación de uso de tiempos de ejecución

Las estadísticas del tiempo de espera y tiempo de respuesta de las peticiones GET de proyectos del Sistema SIAD.



Request	% of cumulative time	Hits	Mean time (ms)	Max time (ms)	Standard deviation	% of cumulative cpu time	Mean cpu time (ms)	Mean allocated Kb	% of system error	Mean size (Kb)
http global	100	11	205	1,325	387	100	89	2,548	0.00	111
http warning	0	0	-1	0	-1	0	-1	-1	0.00	
http severe	58	1	1,325	1,325	0	75	736	27,031	0.00	

Request	% of cumulative time	Hits	Mean time (ms)	Max time (ms)	Standard deviation	% of cumulative cpu time	Mean cpu time (ms)	Mean allocated Kb	% of system error	Mean size (Kb)
/faces/login.xhtml GET	58	1	1,325	1,325	0	75	736	27,031	0.00	8
/faces/javax.faces.resource/components.css GET	17	1	387	387	0	15	156	322	0.00	90
/faces/javax.faces.resource/bootstrap/css/bootstrap.css GET	8	1	189	189	0	3	38	89	0.00	187
/faces/javax.faces.resource/jquery/jquery-plugins.js GET	4	1	106	106	0	1	11	94	0.00	260
/faces/javax.faces.resource/jquery/jquery.js GET	4	1	92	92	0	0	8	82	0.00	84
/faces/javax.faces.resource/css/styles.css GET	2	1	60	60	0	0	5	84	0.00	6
/faces/javax.faces.resource/core.js GET	1	1	31	31	0	0	3	55	0.00	36
/faces/javax.faces.resource/components.js GET	1	1	30	30	0	1	12	66	0.00	411
/faces/javax.faces.resource/siad%20utn.png GET	1	1	23	23	0	0	4	69	0.00	5
/faces/javax.faces.resource/bootstrap/js/bootstrap.js GET	0	1	18	18	0	0	6	71	0.00	128
/faces/javax.faces.resource/profile1.png GET	0	1	3	3	0	0	2	65	0.00	3

Fig. 29: Resumen de estadísticas – Java Melody
Fuente: Propia

La información se muestra también en la consola de JavaEE:

08:30:34,556 INFO [javamelody] (default task-4) remoteAddr = 127.0.0.1, request = /faces/segaccesos/menu.xhtml POST: 194 ms, 0 Ko
08:30:34,734 INFO [javamelody] (default task-14) remoteAddr = 127.0.0.1, request = /faces/javax.faces.resource/moment/moment.js?ln=primefaces&v=7.0 GET: 5 ms, 58 Ko
08:30:34,744 INFO [javamelody] (default task-15) remoteAddr = 127.0.0.1, request = /faces/javax.faces.resource/charts/charts.css?ln=primefaces&v=7.0 GET: 15 ms, 3 Ko
08:30:34,749 INFO [javamelody] (default task-13) remoteAddr = 127.0.0.1, request = /faces/javax.faces.resource/chartjs/chartjs.js?ln=primefaces&v=7.0 GET: 20 ms, 395 Ko
08:30:34,755 INFO [javamelody] (default task-16) remoteAddr = 127.0.0.1, request = /faces/javax.faces.resource/charts/charts.js?ln=primefaces&v=7.0 GET: 14 ms, 328 Ko
08:30:34,758 INFO [javamelody] (default task-17) remoteAddr = 127.0.0.1, request = /faces/javax.faces.resource/headerR.png?ln=images GET: 3 ms, 30 Ko
08:30:34,760 INFO [javamelody] (default task-18) remoteAddr = 127.0.0.1, request = /faces/javax.faces.resource/spacer/dot_clear.gif?ln=primefaces&v=7.0 GET: 2 ms, 0 Ko
08:30:35,653 INFO [javamelody] (default task-4) remoteAddr = 127.0.0.1, request = /faces/segaccesos/opciones.xhtml GET: 1091 ms, 41 Ko
08:30:57,220 INFO [javamelody] (default task-4) remoteAddr = 127.0.0.1, request = /faces/plnplanificacion/docente/index.xhtml GET: 1322 ms, 154 Ko
08:30:34,758 INFO [javamelody] (default task-17) remoteAddr = 127.0.0.1, request = /rest: 1 ms
08:30:34,760 INFO [javamelody] (default task-18) remoteAddr = 127.0.0.1, request = / rest/consulta GET: 2 ms, 0 Ko
08:30:35,653 INFO [javamelody] (default task-4) remoteAddr = 127.0.0.1, request = / **rest/consulta/proyectos** GET: 845 ms, 31 Ko08:30:57,220 INFO [javamelody] (default task-4) remoteAddr = 127.0.0.1, request = / GET: 900 ms, 124 Ko

La petición "/**rest/consulta/proyectos**" proporciona la información necesaria para la aplicación Angular.

CAPÍTULO 3

3. Validación de Resultados

A continuación, se muestra las pruebas realizadas a cada lenguaje de programación, se realizaron en base a las métricas establecidas anteriormente sobre la ISO-25010.

3.1. Diseño de evaluación de Eficiencia de desempeño

Comportamiento en el tiempo

- Métrica: Tiempo de respuesta.

Tiempo estimado para completar una tarea, se toma el tiempo desde que se envía la petición hasta obtener la respuesta.

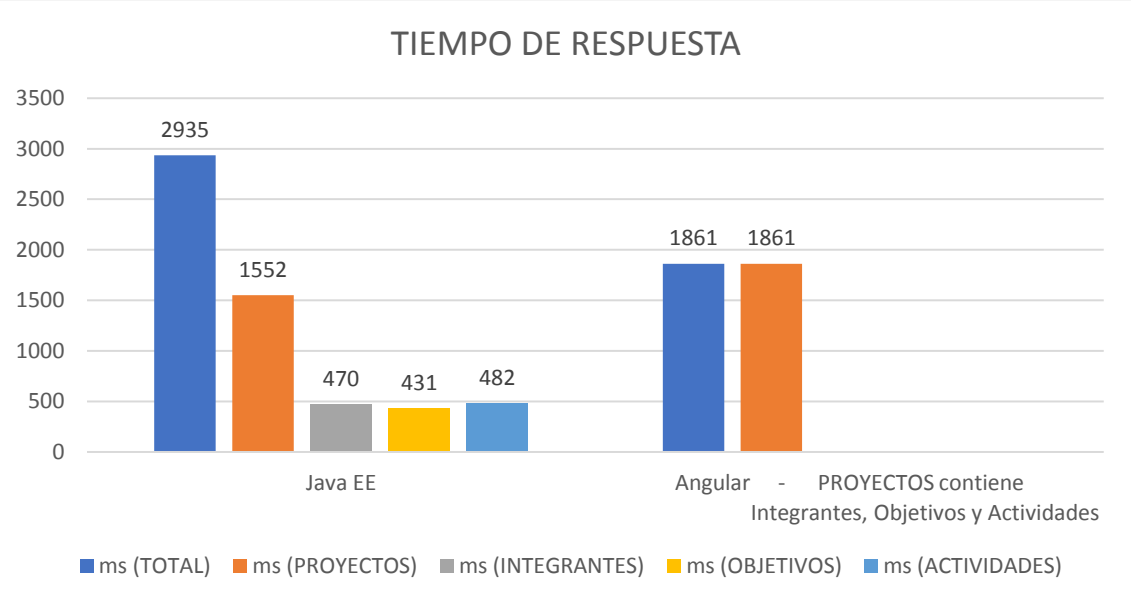


Fig. 30: Comparación de Tiempo de respuesta – Sistema SIAD
Fuente: Propia

Como se muestra en Fig. 30, las pantallas creadas a partir de Java EE y sus componentes propios tarda más tiempo en devolver la respuesta, ya que la consulta inicial busca primero los proyectos y dependiendo de la siguiente petición del usuario se consultan los integrantes o los objetivos y actividades; en el caso de la pantalla de Angular, la consulta inicial genera un archivo .json donde se incluye la información total de proyectos del usuario, con los objetivos, las actividades y los integrantes cada proyecto.

- Métrica: Tiempo de espera.

Tiempo desde que se envía una instrucción, para que inicie un trabajo, hasta que lo completa, tomando el tiempo cuando se inicia un trabajo y el tiempo en completar el trabajo.

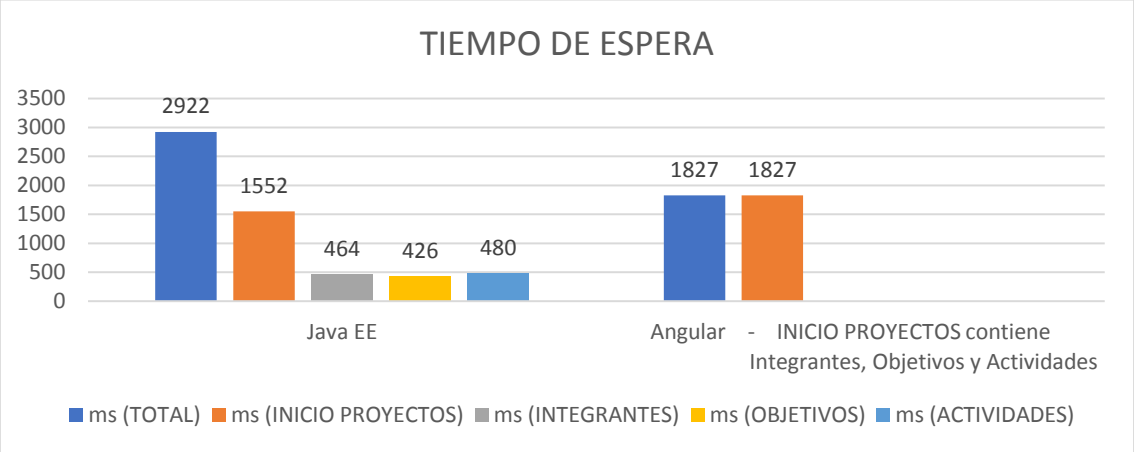


Fig. 31: Comparación de Tiempo de espera – Sistema SIAD
Fuente: Propia

En Fig. 31 se muestra que el tiempo de espera para la pantalla de Angular en relación a las consultas realizadas es menor al presentado por las de JavaEE, debido a que en la respuesta a la consulta para Angular se encuentran todos los datos necesarios para su presentación.

- Métrica: Rendimiento

Cantidad de unidades de datos que procesa un sistema en un cierto período de tiempo.

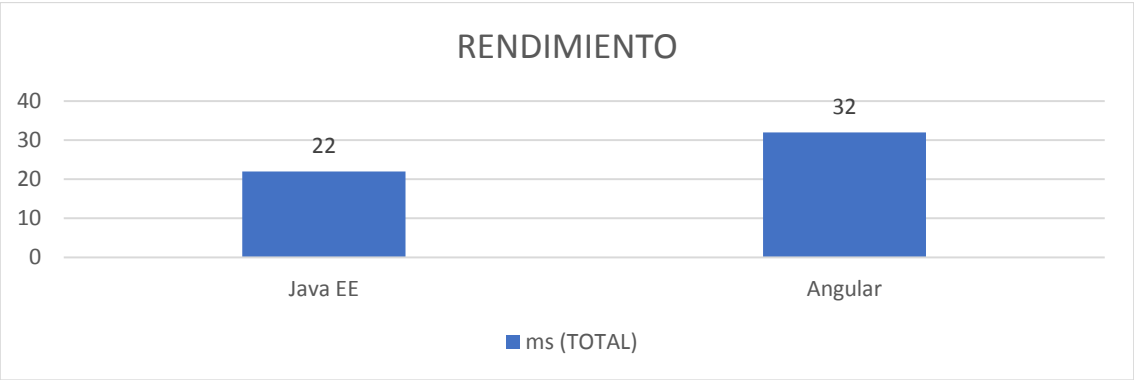


Fig. 32: Comparación de Rendimiento – Sistema SIAD
Fuente: Propia

En Fig.32 se muestra que el número de tareas realizadas para Angular es mayor que el número de tareas ejecutadas para JavaEE en el mismo periodo de tiempo.

Utilización de recursos

- Métrica: Utilización de CPU.

Cantidad de tiempo de CPU en milisegundos (ms) que se usa para realizar una tarea.

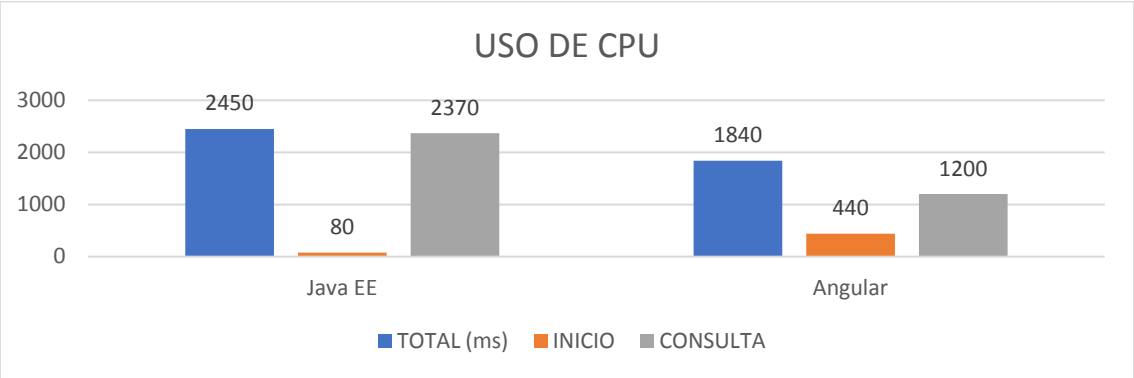


Fig. 33: Comparación de Uso de Procesador – Sistema SIAD
Fuente: Propia

El consumo de tiempo de CPU es menor para la presentación de las pantallas Angular en contraste a las pantallas JavaEE.

- Métrica: Utilización de la memoria.

La memoria en megabytes (M) que es usada para realizar una tarea dada la cantidad total de espacio de memoria y la cantidad de espacio de memoria que realmente es usado para realizar una tarea.

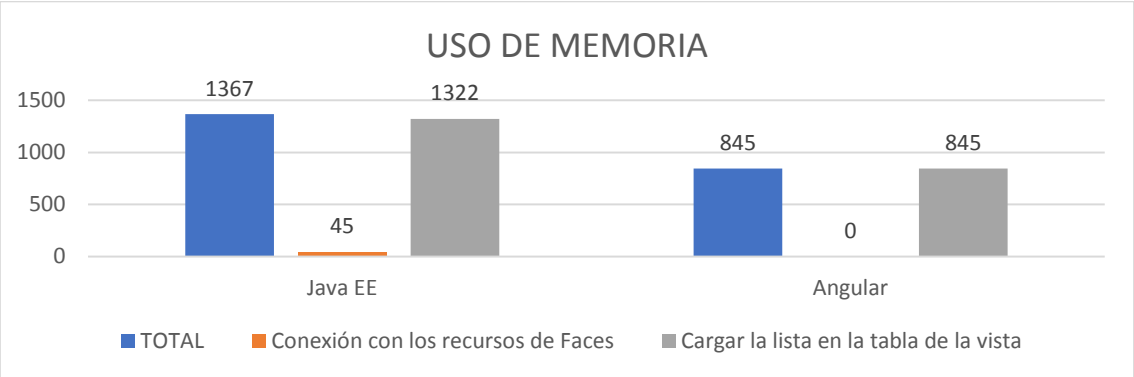


Fig. 34: Comparación de Uso de memoria – Sistema SIAD
Fuente: Propia

El uso de memoria para la presentación de la información en Angular es menor comparado con el usado en JavaEE debido a que este último precisa además conectar con los recursos asociados a la creación de las páginas web de la tecnología Java Server Faces.

Capacidad

- Métrica: Número de peticiones online

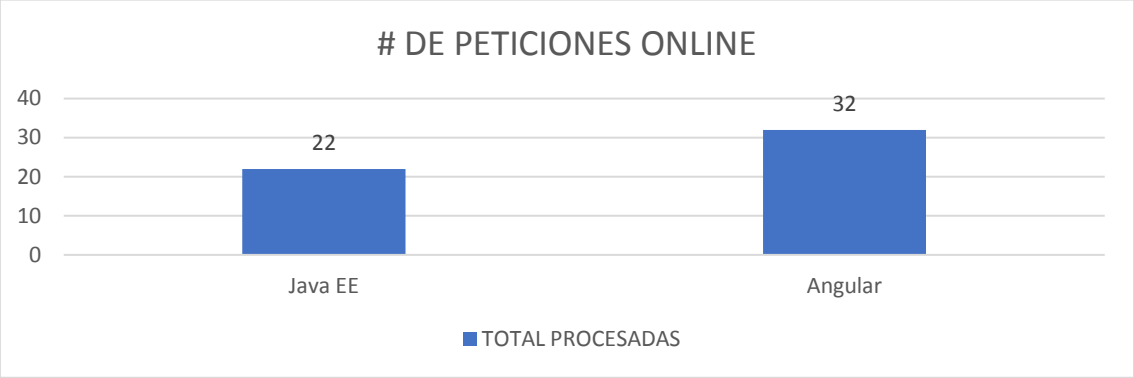


Fig. 35: Comparación de # de peticiones online– Sistema SIAD
Fuente: Propia

Las pantallas de Angular permiten realizar un mayor número de peticiones online en comparación con Java EE.

3.2. Validación de resultados

Para obtener los datos se evaluaron las tareas de obtención de información de los proyectos y su presentación al usuario, en correspondencia al estándar ISO/IEC 25023:2016 que provee medidas de calidad para evaluar cuantitativamente la calidad del producto de software en un rango de 1 a 10, dependiendo de los valores obtenidos durante la ejecución del sistema, como complemento, para demostrar el grado de satisfacción se tomó en cuenta los niveles de puntuación propuestos por la Norma ISO/IEC 25040:

Escala de medición	Niveles de Puntuación	Grado de Satisfacción
8,00 – 10,00	Cumple con los requisitos	Muy Satisfactorio
5,00 – 7,95	Aceptable	Satisfactorio
1,00 – 4,95	Inaceptable	Insatisfactorio

Tabla. 15: Escala de medición ISO/ICE 25040
Fuente: iso25000.com

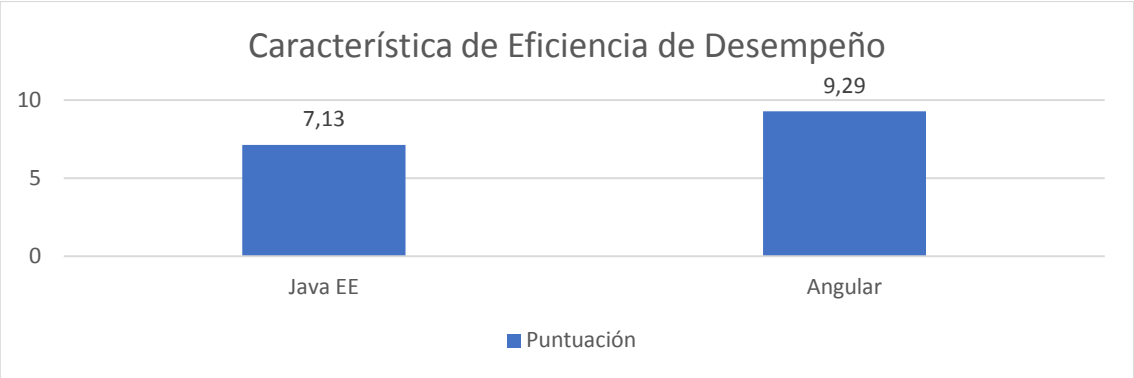


Fig. 36: Comparación Puntuación de Característica Evaluada– Sistema SIAD
Fuente: Propia

La Fig. 36 muestra la puntuación obtenida por los dos frontales probados durante la evaluación, a continuación, se detallan las subcaracterísticas, métricas y el rango de valores utilizados en la evaluación (deseado y peor caso) tomando los datos obtenidos en las pruebas realizadas:

Datos Java EE

Característica	Subcaracterísticas	Métrica	Peor caso	Valor Deseado	Variables			Valor Obtenido	Valor Métrica / 10	Final Subcaracterística	Total Característica
					A	B	T	X			
Eficiencia en el desempeño	Comportamiento del tiempo	Tiempo de respuesta	>=5 seg	<=3 seg	0	2,9350		2,94	7,07	7,16	7,13
		Tiempo de espera	>5 seg	<=3 seg	0,0030	2,9250		2,92	7,08		
		Rendimiento	0	>=4	22		3	7,33	7,33		
	Utilización de recursos	Utilización de CPU	>=10 seg	<=3 seg	0,0181	2,4500		2,43	7,57	6,90	
		Utilización de la memoria	>=1500 mb	<=900mb	45,00	1322,00		1277,00	6,23		
	Capacidad	Número de peticiones online	0	>=10	22		3	7,33	7,33	7,33	

Fig. 37: Evaluación del Sistema SIAD – JavaEE
Fuente: Propia

Datos Angular

Característica	Subcaracterísticas	Métrica	Peor caso	Valor Deseado	Variables			Valor Obtenido	Valor Métrica / 10	Final Subcaracterística	Total Característica
					A	B	T	X			
Eficiencia en el desempeño	Comportamiento del tiempo	Tiempo de respuesta	>=5 seg	<=3 seg	0	1,8610		1,86	8,14	8,77	9,29
		Tiempo de espera	>5 seg	<=3 seg	0,0030	1,8270		1,82	8,18		
		Rendimiento	0	>=4	32		3	10,67	10,00		
	Utilización de recursos	Utilización de CPU	>=10 seg	<=3 seg	0,0181	1,8500		1,83	8,17	9,08	
		Utilización de la memoria	>=1500 mb	<=900mb	0,00	845,00		845,00	10,00		
	Capacidad	Número de peticiones online	0	>=10	32		3	10,67	10,00	10,00	

Fig. 36 Evaluación del Sistema SIAD - Angular
Fuente: Propia

CONCLUSIONES

- La documentación generada brinda información concerniente a la plataforma de desarrollo y framework de aplicaciones Angular, la plataforma de Java: Edición Empresarial, la arquitectura rest, la API de desarrollo REST y la programación reactiva.

- Las pantallas creadas en Angular a partir de los datos obtenidos del servicio web del sistema SIAD permiten que la información se muestre al usuario en un tiempo significativamente menor debido a que el servidor envía datos planos para sean visualizados por el frontend y con un menor consumo de recursos en el lado del cliente en relación con sus símiles en JavaEE.

- Las subcaracterísticas de la característica de eficiencia de desempeño de la norma ISO/ICE 25010 establecen un criterio de evaluación del funcionamiento del sistema actual en comparación con las nuevas pantallas permitiendo notar la mejora en el tiempo de respuesta, así como la visible optimización en aspectos como uso de CPU y memoria.

RECOMENDACIONES

- Combinar plataformas de desarrollo de aplicaciones utilizando las características que mejor se adapten a las necesidades de las tareas a realizar para el correcto funcionamiento de los sistemas.

- Tomar en cuenta las guías existentes para las distintas plataformas de desarrollo y las comunidades que brindan apoyo para la investigación de temas relacionados.

- Evaluar la calidad los sistemas de acuerdo a las normas probadas para poder realizar las mejoras necesarias previo a la puesta en marcha garantizando un alto grado de satisfacción en el usuario y un eficiente consumo de recursos.

REFERENCIAS

Álvarez Caules, C. (22 de 03 de 2018). *Blog sobre JavaEE*. Obtenido de arquitecturajava: arquitecturajava.com/arquitectura-rest-y-sus-niveles/

Amstrong, E., Ball, J., Bodoff, S., Bode Carson, D., Evans, I., Green, D., . . . Jendrock, E. (07 de 12 de 2005). *Documentation.* Obtenido de Oracle Help Center Learn: docs.oracle.com/javaee/0.4/tutorial/doc/J2EETutorial.pdf

Arizmendi, P. (2018). *AngularJS: Conviértete en el profesional que las compañías de software necesitan* (Kindle ed.). Amazon Mexico Services, Inc.

Blokehead, T. (2016). *Scrum - ¡Guía definitiva de prácticas ágiles esenciales de Scrum!* Babelcube.

Chapaval, N. (2018). *Platzi*. Obtenido de platzi.com/blog/que-es-frontend-y-backend/

Drumond, C., Cook, M., & West, D. (9 de 11 de 2018). *ATLASSIAN Agile Coach*. Obtenido de atlassian: www.atlassian.com/es/agile/scrum

Echazú, E., & Rodríguez, R. (02 de 2018). *FundéuRAE.* Obtenido de Fundeu: fundeu.es/wp-content/uploads/2018/02/Glosario-de-Comunicación-Estratégica-Fundéu.pdf

Garzás, P., & Piattini, V. (2015). *Fábricas de software: Experiencias, tecnologías y organización.* Obtenido de https://ebookcentral.proquest.com

GitHub, Inc. (10 de 02 de 2020). *GitHub*. Obtenido de www.github.com: https://github.com/javamelody/wiki/UserGuide#javamelody-setup

Google. (2010-2020). *Angular*. Obtenido de https://angular.io

ISLA VISUAL. (13 de 11 de 2012). *islavisual.com.* Obtenido de ISLA VISUAL: https://islavisual.com/articulos/desarrollo_web/scrum.jpg

ISO/IEC 25000. (2019). Obtenido de Portal ISO 25000: https://iso25000.com/index.php/normas-iso-25000/iso-25010/21-eficiencia-de-desempeno

Leiva, Geancarlo. (31 de 01 de 2014). *Sw de fábrica Blog*. Obtenido de Blogger: swdefabrica.blogspot.com/2014/01/nota-de-arquitecto-jee-el-estandar-java.html

Limburg, A. (3 de Septiembre de 2019). *BLOG DE JAX LONDON*. Obtenido de PROGRAMACIÓN REACTIVA EN JAVA EE TUTORIAL: https://jaxlondon.com/blog/java-core-languages/future-enterprise-java-reactive/

Martínez, R. (18 de 04 de 2015). *LinkedIn Corporation.* Obtenido de LinkedIn SlideShare: https://es.slideshare.net/rmartinez582/sistemas-actuales-y-calidad-de-producto

Monday, P. B. (2003). Implementing the Data Transfer Object Pattern. En *Web Services Patterns: Java™ Platform Edition.* Berkeley: Apress. doi:https://doi.org/10.1007/978-1-4302-0776-4_16

Naciones Unidas. (2019). *La Agenda 2030 y los Objetivos de Desarrollo Sostenible: una oportunidad para América Latina y el Caribe.* Santiago. Obtenido de https://repositorio.cepal.org/bitstream/handle/11362/40155/24/S1801141_es.pdf

Nieto Lemus, A. C. (2015). Arquitectura por componentes JEE, un caso práctico. *Revista Gerencia Tecnológica Informática*, 31-41.

Ollivier, S. M. (2016). *AngularJS Desarrolle hoy las aplicaciones web de mañana.* Ediciones ENI.

Oracle. (2014). *Java Platform, Enterprise Edition: The Java EE Tutorial*. Obtenido de Oracle | Hardware and Software, Engineered to Work Together: https://docs.oracle.com/javaee/7/tutorial/overview.htm#BNAAW

Oracle and/or its affiliates. (2017). *Java Platform, Enterprise Edition (Java EE) 8*. Obtenido de Java Platform, Enterprise Edition The Java EE Tutorial Java Platform, Enterprise Edition, Release 8: https://javaee.github.io/tutorial/toc.html

Palacio, J., & Ruata, C. (07 de 2009). *Scrum Manager: Proyectos - apuntes de formación.* Obtenido de scrummanager.net/files/sm_proyecto_apuntes_12.pdf

Palacio, M. (06 de 2021). *Scrum Manager.* Obtenido de srummanager: www.scrummanager.net/files/scrum_master.pdf

ReativeX. (2019). *ReaxtivoX*. Obtenido de reactivex.io/intro.html

Thompson, M., Bóner, J., Farley, D., & Kuhn, R. (16 de Septiembre de 2014). El Manifiesto de Sistemas Reactivos. Obtenido de https://www.reactivemanifesto.org/es