UNIVERSIDAD NACIONAL DEL SANTA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas e Informática

“Análisis comparativo entre las arquitecturas de software para aplicaciones web aplicado a la empresa 365 Engineering & Consulting”

TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS E INFORMÁTICA

AUTORES:

- Bach. Huamán Ricse, Daniel Anthony
- Bach. Quenhua Vásquez, Elvis Arnol

ASESOR:

Ms. Macedo Alcántara, Dayán Fernando

Código ORCID: 0000-0003-1190-4032

Nuevo Chimbote – Perú

2023-10-10

UNIVERSIDAD NACIONAL DEL SANTA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas e Informática

"Análisis comparativo entre las arquitecturas de software para aplicaciones web aplicado a la empresa 365 Engineering & Consulting"

Tesis para Obtener el Título Profesional de Ingeniero de Sistemas e Informática

Revisado y Aprobado por Asesor:

Ms. Dayán Fernando Macedo Alcántara
DNI : 32941877
Código ORCID: 0000-0003-1190-4032

UNIVERSIDAD NACIONAL DEL SANTA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas e Informática

"Análisis comparativo entre las arquitecturas de software para aplicaciones web aplicado a la empresa 365 Engineering & Consulting"

Tesis para Obtener el Título Profesional de Ingeniero de Sistemas e Informática

Revisado y Aprobado por el Jurado Evaluador:

Ms. Camilo Ernesto Suárez Rebaza
Presidente
DNI : 32978627
Código ORCID: 0000-0002-6870-4296

Ms. Pedro Glicerio Manco Pulido
Secretario
DNI: 32953190
Código ORCID: 0000-0002-8542-2119

Ms. Dayán Fernando Macedo Alcántara
Integrante
DNI : 32941877
Código ORCID: 0000-0003-1190-4032

FACULTAD DE INGENIERIA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

ACTA DE EVALUACIÓN PARA SUSTENTACIÓN DE TESIS

A los 10 días del mes de octubre del año dos mil veintitrés, siendo las 11: 00 horas, en el Aula S2 de la Escuela Profesional de Ingeniería de Sistemas e Informática- (Nuevo Pabellón), se instaló el Jurado Evaluador designado mediante T. Resolución Nº 531-2023-UNS-CFI, con fecha con fecha 23.08.2023, integrado por los siguientes docentes: MS. CAMILO ERNESTO SUAREZ REBASA (Presidente), MS. PEDRO GLICERIO MANCO PULIDO (Secretario), MS. DAYAN FERNANDO MACEDO ALCÁNTARA (Integrante), en base a la Resolución Decanal Nº 725-2023-UNS-FI se da inicio la sustentación de la Tesis intitulada: "**Análisis comparativo entre las Arquitecturas de Software para Aplicaciones Web aplicado a la empresa 365 Engineering & Consulting**", presentado por las Bachilleres Huaman Ricse Daniel Anthony, con cód. Nº 0200914007 y Quenhua Vasquez Elvis Arnol, con cód. Nº 0200914046, quienes fueron asesorados por el docente Ms. Dayan Fernando Macedo Alcántara, según lo establece la Resolución Decanal Nº 114-2020-UNS-FI, de fecha 20.07.2020.

Terminada la sustentación, el tesista respondió a las preguntas

El Jurado Evaluador, después de deliberar sobre aspectos relacionados con el trabajo, contenido y sustentación del mismo, y con las sugerencias pertinentes en concordancia con el Art. 71° y 111° del Reglamento General de Grados y Títulos, vigente de la Universidad Nacional del Santa (Resolución N° 580-2022-CU-R-UNS del 22.08.2022); consideran la siguiente nota final de evaluación:

BACHILLER	CALIFICACIÓN	CONDICIÓN
DANIEL ANTHONY HUAMÁN RICSE	17	BUENO

Siendo la 11: 00 a.m. se dio por terminado el Acto de Sustentación y en señal de conformidad, firma el Jurado la presente Acta.

Nuevo Chimbote, 10 de octubre de 2023

MS. CAMILO ERNESTO SUAREZ REBASA
PRESIDENTE

MS. PEDRO GLICERIO MANCO PULIDO
SECRETARIO

MS. DAYAN FERNANDO MACEDO ALCÁNTARA
INTEGRANTE

FACULTAD DE INGENIERIA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

ACTA DE EVALUACIÓN PARA SUSTENTACIÓN DE TESIS

A los 10 días del mes de octubre del año dos mil veintitrés, siendo las 11: 00 horas, en el Aula S2 de la Escuela Profesional de Ingeniería de Sistemas e Informática- (Nuevo Pabellón), se instaló el Jurado Evaluador designado mediante T. Resolución Nº 531-2023-UNS-CFI, con fecha con fecha 23.08.2023, integrado por los siguientes docentes: MS. CAMILO ERNESTO SUAREZ REBASA (Presidente), MS. PEDRO GLICERIO MANCO PULIDO (Secretario), MS. DAYAN FERNANDO MACEDO ALCÁNTARA (Integrante), en base a la Resolución Decanal Nº 725-2023-UNS-FI se da inicio la sustentación de la Tesis intitulada: "**Análisis comparativo entre las Arquitecturas de Software para Aplicaciones Web aplicado a la empresa 365 Engineering & Consulting**", presentado por las Bachilleres: Huaman Ricse Daniel Anthony, con cód. Nº 0200914007 y Quenhua Vasquez Elvis Arnol, con cód. Nº 0200914046, quienes fueron asesorados por el docente Ms. Dayan Fernando Macedo Alcántara, según lo establece la Resolución Decanal Nº 114-2020-UNS-FI, de fecha 20.07.2020

Terminada la sustentación, el tesista respondió a las preguntas

El Jurado Evaluador, después de deliberar sobre aspectos relacionados con el trabajo, contenido y sustentación del mismo, y con las sugerencias pertinentes en concordancia con el Art. 71° y 111° del Reglamento General de Grados y Títulos, vigente de la Universidad Nacional del Santa (Resolución N° 580-2022-CU-R-UNS del 22.08.2022); consideran la siguiente nota final de evaluación:

BACHILLER	CALIFICACIÓN	CONDICIÓN
ELVIS ARNOL QUENHUA VÁSQUEZ	17	BUENO

Siendo la 11: 00 a.m. se dio por terminado el Acto de Sustentación y en señal de conformidad, firma el Jurado la presente Acta.

Nuevo Chimbote, 10 de octubre de 2023

MS. CAMILO ERNESTO SUAREZ REBASA
PRESIDENTE

MS. PEDRO GLICERIO MANCO PULIDO
SECRETARIO

MS. DAYAN FERNANDO MACEDO ALCÁNTARA
INTEGRANTE

turnitin

Recibo digital

Este recibo confirma quesu trabajo ha sido recibido por Turnitin. A continuación podrá ver la información del recibo con respecto a su entrega.

La primera página de tus entregas se muestra abajo.

Autor de la entrega:	Elvis Quenhua Vásquez
Título del ejercicio:	Trabajo
Título de la entrega:	Análisis comparativo entre las Arquitecturas de Software pa...
Nombre del archivo:	INFORME_TESIS_HUAMAN_-_QUENHUA.pdf
Tamaño del archivo:	4.98M
Total páginas:	116
Total de palabras:	15,544
Total de caracteres:	96,212
Fecha de entrega:	16-oct.-2023 03:58p. m. (UTC-0500)
Identificador de la entre...	2197874731

UNIVERSIDAD NACIONAL DEL SANTA

FACULTAD DE INGENIERÍA

UNS

"Análisis comparativo entre las Arquitecturas de Software para Aplicaciones Web aplicado a la empresa 365 Engineering & Consulting"

TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS E INFORMÁTICA

AUTORES:

ASESOR:

Análisis comparativo entre las Arquitecturas de Software para Aplicaciones Web aplicado a la empresa 365 Engineering & Consulting

INFORME DE ORIGINALIDAD

20%	19%	3%	8%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	repositorio.uns.edu.pe Fuente de Internet	5%
2	repositorio.ucv.edu.pe Fuente de Internet	1%
3	Submitted to Universidad Cesar Vallejo Trabajo del estudiante	1%
4	core.ac.uk Fuente de Internet	1%
5	1library.co Fuente de Internet	1%
6	hdl.handle.net Fuente de Internet	1%
7	Submitted to Universitat Politècnica de València Trabajo del estudiante	1%
8	www.coursehero.com Fuente de Internet	1%

DEDICATORIA

Quiero dedicar mi tesis principalmente a Dios, ya que él me ha brindado la fuerza necesaria para alcanzar esta meta en mi vida.

Dedicado a mis padres y familia, por su inquebrantable apoyo, amor y paciencia a lo largo de todos estos años.

Dedicado a mis amigos, con quienes compartí esta travesía, brindándome su compañía y apoyo en esta etapa de mi vida.

Bach. Elvis Arnol Quenhua Vásquez

DEDICATORIA

A mi madre, quien es mi principal fuente de apoyo incondicional y cuando más la necesitaba, estuvo siempre presente. Gracias a ella estoy cumpliendo este objetivo importante.

A mis amigos y compañeros de estudios de la Universidad Nacional del Santa, por sus motivaciones y el haber compartido muchas experiencias fructíferas durante mi carrera profesional.

Bach. Daniel Anthony Huamán Ricse

AGRADECIMIENTOS

Deseamos expresar nuestra gratitud hacia algunas personas que nos apoyaron a realizar este proyecto de investigación:

En primer lugar, queremos expresar nuestra gratitud a nuestros padres, quienes siempre nos han brindado su apoyo incondicional para lograr nuestros objetivos académicos y personales.

A los docentes de la Universidad Nacional del Santa, por impartir sus conocimientos y consejos para llegar ser buenos profesionales.

A nuestro asesor Ms. Dayán Fernando Macedo Alcántara, por darnos sus orientaciones, tolerancia y apoyo, que permitieron la realización de nuestra tesis.

A la empresa 365 ENGINEERING & CONSULTING, por facilitarnos los recursos necesarios para nuestra investigación.

A las siguientes profesionales también: ISC. Carlos Pérez Domínguez, MGTI. Miguel Ángel de León, MSC. Ernesto Solano León y Mg. Duval Neri Luján, por compartirnos sus experiencias y amplio conocimiento sobre diversos temas de gran importancia para el desarrollo del proyecto .

Bach. Daniel Anthony Huamán Ricse.

Bach. Elvis Arnol Quenhua Vásquez.

INDICE

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN

El presente proyecto de investigación tiene como objetivo principal el análisis comparativo entre la Arquitecturas de Software Multicapa, Arquitectura orientada a servicios (SOA) y de Microservicios, que son cruciales para el desarrollo de aplicaciones web, teniendo en cuenta los atributos de calidad que se han revisado en varias fuentes bibliográficas. La arquitectura del software se refiere a cómo se estructura y organiza el sistema para cumplir con los requisitos funcionales y no funcionales, asegurando así la calidad del software. estructura es un ejemplo de un diseño de sistema de alto nivel que cumple dos propósitos: satisface los requisitos de calidad y actúa como punto de referencia para el desarrollo.

En este proyecto de investigación se desarrollarán tres aplicaciones web propuestas, según el tipo de arquitectura y serán desplegadas dentro de una red local. Estas Aplicaciones llevarán el mismo proceso de negocio, acerca de la Gestión de Informes de Proyectos de la empresa 365 ENGINEERING & CONSULTING, que se dedica a la Consultoría de Sistemas de Gestión, servicios de equipos y de seguridad para el Trabajo. Posteriormente se realizará el análisis comparativo de las Arquitecturas de Software, para descubrir la Arquitectura de software que tiene mejor impacto en el rendimiento de Aplicaciones web para la empresa 365 ENGINEERING & CONSULTING.

Palabras Claves: Arquitectura de Software, Aplicaciones Web, Consultoría de Sistemas de Gestión, Seguridad para el trabajo, Atributos de Calidad de Software

ABSTRACT

The main objective of the current research project is to analyze the three main software architectures: 3-layer, Service Oriented Architecture (SOA) and Microservices, which are crucial for the development of web applications, taking into account the quality attributes that have been reviewed in various literature sources. Software architecture refers to how the system is structured and organized to meet functional and non-functional requirements, thus ensuring software quality. Structure is an example of a high-level system design that serves two purposes: it satisfies quality requirements and acts as a reference point for development.

In this research project, three proposed web applications will be developed, according to the type of architecture and will be deployed within a local network. These Applications will carry the same business process, about the Project Report Management of the company 365 ENGINEERING & CONSULTING, which is dedicated to the Management Systems Consulting, equipment and safety services for the Work.

Subsequently, a comparative analysis of the Software Architectures will be carried out, in order to discover the most optimal Architecture for the development of Web Applications of the company 365 ENGINEERING & CONSULTING.

Keywords: Software Architecture, Web Applications, Management Systems Consulting, Security for the job, Software Quality Attributes

PRESENTACIÓN

Señores miembros del jurado:

Conforme a lo decretado en el Reglamento General de Grados y Títulos de la Universidad Nacional del Santa, presentamos a su consideración el Proyecto de Trabajo de Investigación titulado: "**ANALISIS COMPARATIVO ENTRE LAS ARQUITECTURAS DE SOFTWARE PARA APLICACIONES WEB APLICADO A LA EMPRESA 365 ENGINEERING & CONSULTING"** con la finalidad de obtener el Título Profesional de Ingeniero de Sistemas e Informática.

El presente Proyecto de Investigación posee como propósito realizar un Análisis Comparativo entre las principales Arquitecturas de Software: Multicapas, Arquitectura orientada a servicios (SOA) y la Arquitectura de Microservicios, con el fin de demostrar cual es la arquitectura más óptima, para el desarrollo de aplicaciones web en la empresa 365 ENGINEERING & CONSULTING, ubicada en el distrito de Nuevo Chimbote.

En virtud de lo expuesto, queremos someter a su consideración el presente proyecto, con el objetivo de que sea evaluado y revisado, esperamos se cumpla con los requisitos mínimos para su aprobación.

Atentamente,

Los Autores

INTRODUCCIÓN

Actualmente el Internet es un mercado con mucha demanda, y por ello existe la necesidad de desarrollar las aplicaciones web cada vez más complejas, que satisfagan los requisitos tanto funcionales como no funcionales, que garanticen la calidad del software. Vale mencionar algunos Atributos de calidad de gran importancia, como por ejemplo el Rendimiento, la Disponibilidad, Escalabilidad, Seguridad, etc.

Es muy importante la elección de una Arquitectura de Software factible y adecuada, que incorpore varios componentes y puedan integrarse: ya sea a corto o largo plazo, con otras aplicaciones, para que las funcionalidades sean completas y efectivas.

Por lo tanto, la tesis actual se centra en realizar el Análisis Comparativo entre las principales Arquitecturas de Software: Multicapa, Arquitectura orientada a servicios (SOA) y la Arquitectura de Microservicios, con el fin de demostrar cual es la arquitectura más óptima para el desarrollo de aplicaciones web, teniendo como caso práctico el desarrollo de aplicaciones para la empresa 365 ENGINEERING & CONSULTING, que serán desplegadas dentro de una red local.

El informe actual está estructurado en siete capítulos, que son los siguientes:

En el **CAPITULO I,** se proporciona una descripción general del espacio de estudio (la empresa).

En el **CAPITULO II**, se describe la realidad problemática que da origen al trabajo de investigación, así como la hipótesis, los objetivos y las justificaciones del trabajo de investigación.

En el **CAPITULO III**, incluye el marco teórico, describiendo Las Arquitecturas de software Multicapa, Orientada a Servicios(SOA), Microservicios y las herramientas tecnológicas utilizadas en el desarrollo y sus respectivas pruebas de carga.

En el **CAPITULO IV**, se aplica la metodología hibrida: ICONIX, describiendo los pasos necesarios para desarrollar las aplicaciones web que se proponen.

En el **CAPITULO V**, se especifica la población y la muestra, así como los métodos utilizados para la recopilación y análisis de los datos.

En el **CAPITULO VI,** se exponen los resultados obtenidos de las pruebas de carga, efectuadas a las aplicaciones web. La comprobación de la hipótesis general se realizará utilizando las pruebas estadísticas respectivas por cada indicador cuantitativo.

En el **CAPITULO VII,** se especifican las conclusiones y recomendaciones de los resultados obtenidos anteriormente, acompañadas de la bibliografía y sus anexos.

DATOS GENERALES DEL ESTUDIO

- **TITULO DEL PROYECTO**

  "ANALISIS COMPARATIVO ENTRE LAS ARQUITECTURAS DE SOFTWARE PARA APLICACIONES WEB APLICADO A LA EMPRESA 365 ENGINEERING & CONSULTING."

- **TESISTAS**

  Bach. Daniel Anthony Huamán Ricse.

  Bach. Elvis Arnol Quenhua Vásquez.

- **ASESOR**

  Ms. Dayán Fernando Macedo Alcántara

- **TIPO DE INVESTIGACION**

  a) Según su fin o propósito

  Aplicada Tecnológica, porque se llevó a cabo la creación de una solución práctica que permita abordar la problemática planteada en la elección de una Arquitectura de Software óptima para el desarrollo de aplicaciones web en la empresa 365 ENGINEERING & CONSULTING.

  b) Por el nivel de comprensión que se obtiene:

  Descriptiva, porque la recolección de información se llevará a cabo mediante el diagnóstico de la problemática en la empresa 365 ENGINEERING & CONSULTING.

- **METODO DE INVESTIGACION**

  Después de precisar la situación problemática, se propuso una hipótesis y se realizaron varias observaciones para realizar un análisis comparativo de las arquitecturas de software para el desarrollo de aplicaciones web. El método se ajustaría al del tipo **inductivo-deductivo**.

CAPITULO I: DATOS GENERALES DE LA EMPRESA

1.1. Descripción de la Empresa

1.1.1. Razón Social

365 ENGINEERING & CONSULTING E.I.R.L.

1.1.2. R.U.C

20603639830

1.1.3. Tipo de Institución

Empresa en Consultoría de Sistemas de Gestión y Servicios de Equipos de Seguridad para empresas.

1.1.4. Dirección Legal

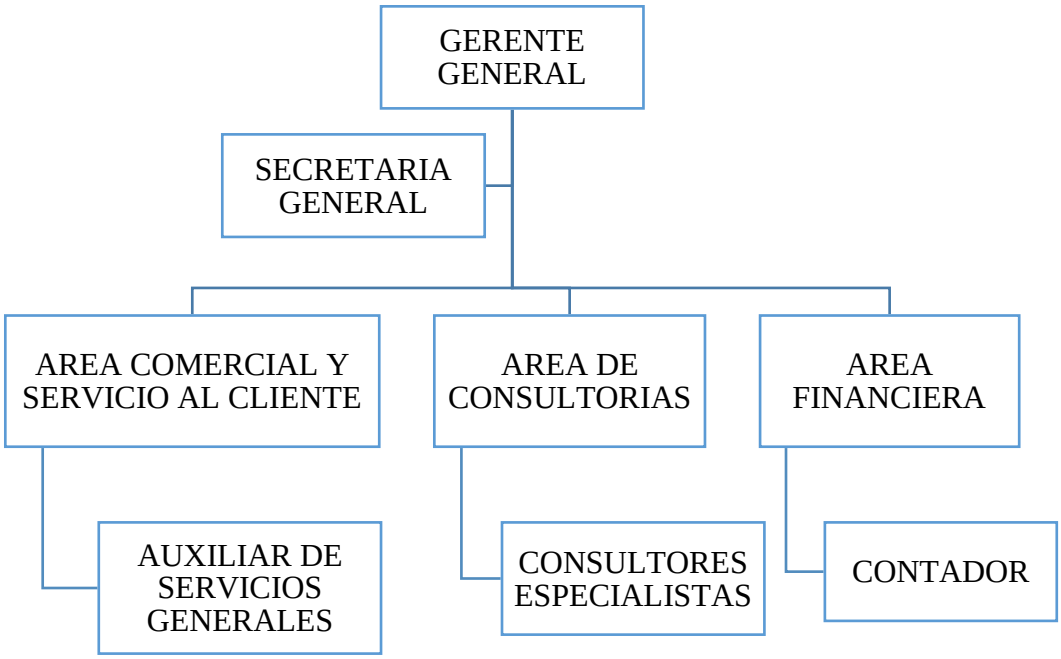
Calle Los Nogales, Manzana E5, Lote 15, Urb. Bellamar II Etapa – Nuevo Chimbote, provincia del Santa, región Ancash, Perú.

1.1.5. Objetivos

- Brindar a sus clientes asesoramiento y consultoría de sistemas de gestión. Esto implica brindar asesoramiento especializado en temas como medio ambiente, salud y seguridad ocupacional en las empresas.
- Ayudar en la implementación de normas y estándares: muchas organizaciones buscan cumplir con normas y estándares internacionales (ISOS). Nuestro objetivo es guiar a los clientes en el proceso de implementación de estos estándares, asegurando el cumplimiento.

1.1.6. Estructura Orgánica

Figura 1: Organigrama General



Fuente: *Elaboración propia*

1.2. Actividad de la Empresa

La empresa ofrece una variedad de servicios para apoyar a las empresas a crear un entorno de trabajo seguro. Algunos de los servicios disponibles incluyen:

- ✓ Elaboración de riesgos laborales: se realizan inspecciones y análisis de los lugares de trabajo para identificar riesgos potenciales para la seguridad y la salud de los trabajadores.
- ✓ Desarrollo de programas de seguridad y salud: se desarrolla e implementa programas y políticas de seguridad que permitan cumplir el desempeño de los empleados de manera correcta.
- ✓ Capacitación en seguridad y salud: impartir capacitaciones y talleres para los empleados y supervisores sobre temas como manejo de sustancias peligrosa, primeros auxilios, prevención de accidentes, etc.

- ✓ Auditorias de cumplimiento: realizar auditorías que nos permita comprobar el cumplimiento de normativas y reglamentos en materia de seguridad y salud en el ambiente laboral.

1.3. Direccionamiento Estratégico

- **Misión**

  Nuestra misión es promover el desarrollo de medidas estratégicas de seguridad y prevención de riesgos, con calidad y eficiencia, para fomentar un equilibrio en los procesos industriales de las empresas. Asimismo, buscamos controlar y mitigar los impactos generados por las acciones propias de la empresa.

- **Visión**

  Nuestra visión es llevar a cabo una intervención integral en los procesos industriales, basados en principios éticos y en la responsabilidad social empresarial y ambiental. Buscaremos posicionarnos como líder y competir tanto a nivel nacional, destacándose por los más altos estándares en seguridad y salud industrial.

1.4. Valores

- **Seguridad y Salud**

  Nuestro compromiso radica en garantizar la seguridad, prevenir riesgos y promover la salud laboral. Fomentamos una cultura de prevención y estamos preparados para responder a cualquier emergencia que pueda surgir en los entornos laborales de nuestra empresa.

- **Comunidad y Medio ambiente**

  Nuestro compromiso social y cultural radica en colaborar con la comunidad y contribuir para que las estrategias empresariales se enfoquen de manera consciente en proteger a las personas y preservar el medio ambiente.

- **Trabajo en Equipo**

  Promover la participación de todo los involucrados en los procesos, con el fin de alcanzar un objetivo compartido. Compartir información y conocimientos para generar confianza y sostenibilidad de dichos procesos.

- **Orientación a la empresa**

  Garantizar que nuestras actividades se enfoquen en cumplir con las necesidades y expectativas del cliente, ofreciendo soluciones completas, competitivas y de excelencia.

- **Conducta ética**

  Nuestras acciones se fundamentan en la profesionalidad, la integridad moral, la lealtad y el respeto hacia los individuos.

CAPITULO II: PLANTEAMIENTO DEL PROBLEMA

2.1. Problema

2.1.1. Realidad Problemática

En la empresa 365 Engineering & Consulting E.I.R.L.; especializada en Consultoría de Sistemas de Gestión y proveedor de equipos y accesorios de seguridad para empresas. Esta firma surge con la convicción de incrementar rentabilidad, crear soporte y potenciar la formalidad de los clientes, brindando servicios de auditoría, capacitación y entre otros, garantizando una buena salud y seguridad de sus colaboradores, utilizando las herramientas de gestión obligatorias. Además, esta empresa trabaja con criterios legales, según las normativas de la ley de Seguridad y Salud en el Trabajo (ley 29783).

En la actualidad, los usuarios de la empresa 365 Engineering & Consulting, elaboran un informe al final del Proyecto, con el objeto de presentar y entregar el informe impreso y almacenado en un Disco Compacto(CD) a los clientes pertinentes, pero puede darse el caso de la pérdida o el deterioro del documento en físico, lo cual contiene información muy importante de los proyectos ejecutados.

Ante esta situación actual problemática, requiere de una aplicación web para la gestión de los informes elaborados por los usuarios de la empresa y de que los clientes puedan descargarlos desde dicha plataforma en cualquier instante, mediante un clave de acceso otorgado por un usuario responsable de la empresa.

Cabe mencionar que la mayoría de las empresas y/u organizaciones cuentan con sistemas o aplicaciones web, desarrollados bajo una Arquitectura

tradicional centralizada. Esto supone una problemática considerable en la calidad del software a futuro, porque involucrará grandes costos para que el sistema subsista como un todo. Por ejemplo, actualmente las grandes empresas tecnológicas como Amazon, Netflix, EBay, PayPal, etc., son quienes manejan algunos de los productos y servicios más utilizados en todo el mundo, han migrado sus sistemas bajo una Arquitectura Tradicional o monolítica, a las arquitecturas modernas más demandadas.

Se pueden encontrar las siguientes problemáticas en relación con lo mencionado anteriormente, de acuerdo a las Arquitecturas tradicionales centralizadas:

- ✓ **No existe un escalamiento óptimo.** Para escalar un sistema bajo una Arquitectura tradicional, se debe desplegar toda la aplicación, con lo cual requerimos más infraestructura aun cuando la escalabilidad solo sea requerida por unos pocos casos de uso.
- ✓ **Complejidad del mantenimiento del código fuente**. Cuando un sistema carece de un diseño modular sólido y no está lo suficientemente desacoplado, a medida que la aplicación crece, puede repercutir un aumento significativo en los costos, debido a la complejidad que implica el mantenimiento del código fuente.
- ✓ **Complejidad en la reutilización de código fuente.** Esto supone un gran reto y por lo usual, las otras aplicaciones acaban teniendo las propias copias del código fuente.
- ✓ **Restricción para adoptar nuevas tecnologías.** Ante Cualquier modificación en el marco de trabajo o lenguajes de programación

afectará a todo el sistema de software, lo que resultará más costoso y lento los cambios pertinentes.

- ✓ **Punto único de fallo**. En caso de que falle algún componente de la arquitectura, podría llegar a afectar a toda la aplicación.

2.1.2. Análisis del Problema

Teniendo en cuenta la problemática actual de la empresa, se llevará a cabo un análisis exhaustivo de cada uno de los problemas identificados para profundizar en los detalles y proponer posibles soluciones:

- ✓ **Existe el problema de escalamiento.** Podemos decir que ante los inconvenientes que existe en el escalamiento de las aplicaciones web, es necesario elegir una arquitectura indicada que permita escalar cada componente del software de forma independiente.
- ✓ **Existe la complejidad en el mantenimiento del software**. Podemos decir que las aplicaciones dentro de una Arquitectura tradicional tendrán un gran reto a futuro, ante el crecimiento del Proyecto de software, el mantenimiento del código tenderá a ser más complejo. Con una arquitectura moderna, se logrará reducir las complicaciones y el costo en el mantenimiento del software, esto acompañado de los recursos indispensables para cumplir los objetivos.
- ✓ **Existe la complejidad para la reutilización del código**. La reutilización del código tiene como propósito en utilizar existente y que estará disponible para su uso. Con una arquitectura moderna; como el de Microservicios, facilitará notablemente esta estrategia porque, parte de la premisa de la diversificación general de sus servicios. Por tanto, el desarrollador podrá reutilizar el código de otros servicios contiguos

en su propio software, ampliando y potenciando enormemente su desarrollo.

- ✓ **Existe la limitante para adoptar nuevas tecnologías**. Por lo general, la arquitectura tradicional centralizada utiliza la misma colección de tecnología en todo momento. Los cambios en cierta tecnología tendrán un costo significativo, tanto en términos de tiempos de respuesta como de inversión financiera. El grupo de desarrolladores de software, podrán elegir libremente las herramientas con la arquitectura moderna.
- ✓ **Existe el punto único de fallo.** Dado que, si falla algún componente de la arquitectura, podría repercutir en el funcionamiento total de la aplicación. Se estará optando por una Arquitectura de software, donde prevalezca la modularidad y disponibilidad del sistema, ante una alta demanda de usuarios del sistema.

2.2. Antecedentes

A continuación, se enumeran algunos proyectos de investigación recientes; cada uno con aportes valiosos para el tema de estudio de este proyecto de investigación.

2.2.1. Antecedentes a Nivel Internacional

- ➢ **<u>Antecedente 01:</u>**

  Huerta (2020) en su investigación "**Transición del Monolito a los Microservicios: cambio de paradigma en el desarrollo de aplicaciones**."

  Este proyecto ha tenido como objetivo general el estudio detallado de las Arquitectura de Microservicios y sus diferencias respecto a la Arquitectura monolítica, con la certidumbre de que pueda servir como modelo para futuros estudiantes o desarrolladores. Se logró diseñar una

aplicación utilizando la Arquitectura de Microservicios para compararla con una aplicación monolítica desarrollada a la par. Tuvo como conclusión que el proyecto cumplió exitosamente los objetivos trazados en su propuesta. (Huerta, 2020)

Correlación: Este proyecto nos proporcionará el conocimiento que necesitamos sobre el proceso de transición de la arquitectura monolítica a la arquitectura de Microservicios, ya que el proyecto consistirá en la implementación de cuatro aplicaciones REST.

- **Antecedente 02:**

  Pacheco (2018) en su investigación "**Estudio Comparativo entre una Arquitectura con Microservicios y Contenedores Dockers y una Arquitectura Tradicional (Monolítica) con Comprobación Aplicativa**". El objetivo de esta investigación es examinar y comparar una arquitectura de microservicios en contenedores Dockers con una arquitectura de software tradicional utilizando un caso práctico para comprobar el rendimiento y los recursos utilizados en la aplicación. En lo metodológico, tuvo un enfoque documental y bibliográfico para ampliar, estudiar y profundizar el problema.

  Se concluyó que la Arquitectura de Microservicios es mejor en término de tiempo de respuesta y es tolerante a fallos, ya que retorna los datos solicitados de manera exitosa, sin importar que otro servicio se detenga por una actualización o falla. (Pacheco, 2018)

  Correlación: En este antecedente, nos proporcionará la orientación para llevar a cabo una comparación entre las arquitecturas tradicional y Microservicios. Durante el proceso del proyecto se emplearán

herramientas de pruebas para validar el correcto funcionamiento del sistema.

- **<u>Antecedente 03</u>:**

  Barahona (2023) en su investigación "**Propuesta de Implementación de Microservicios usando Metodología Ágil para la Optimización de Procesos en Banhcafe**."

  El objetivo principal de esta investigación es conocer el valor que se lograría al implementar los Microservicios mediante una metodología ágil, con tal de lograr la disminución de retrasos en la entrega de los aplicativos producidos por el equipo, ocasionado por dos problemas potenciales: La existencia de una aplicación monolítica y la ausencia de una metodología ágil para el desarrollo.

  Se concluye en esta investigación que entre agosto 2021 y julio 2022, los programadores emplearon en promedio uno a cuatro meses para desarrollar nuevas características. Así también, el porcentaje de mejora fue de un 92%, después de la implementación de los microservicios con una metodología ágil . (Barahona, 2023)

  Correlación: Dicho antecedente, nos aportará en nuestra investigación sobre cómo implementar los Microservicios en una empresa, ya que sugiere mejoras en la planificación y entrega de proyectos. El objetivo es reducir el tamaño del cogido fuente al desarrollar nuevos proyectos y evitar demoras al implementar nuevos servicios en producción. Esto se logrará evitar la necesidad de compilar todo el sistema centralizado cada vez que se debe implementar un nuevo servicio, lo que a menudo detiene el funcionamiento del sistema del negocio.

2.2.2. Antecedentes a Nivel Nacional

- **<u>Antecedente 04:</u>**

  Ruelas (2017) en su investigación "**Modelo de Composición de Microservicios para la Implementación de una Aplicación Web de Comercio Electrónico utilizando Kubernetes**."

  Tuvo como objetivo general presentar un modelo de composición de Microservicios para la implementación de una aplicación web de e-commerce utilizando la herramienta Kubernetes. Se descubrió que un modelo basado en la arquitectura de Microservicios, que utiliza Kubernetes para implementar una aplicación Web de comercio electrónico, funciona mejor que una arquitectura monolítica en un 104% en los indicadores cuantitativos de disponibilidad , rendimiento y tiempo de respuesta. (Ruelas , 2017)

  Correlación: Este Proyecto nos brindará una comprensión sobre la implementación de una aplicación bajo el modelo de Microservicios en el ámbito del comercio electrónico. Asimismo, se utilizó el servicio de autenticación JWT para asegurar el acceso seguro a los Microservicios, y se hizo uso de herramientas tecnológicas Kubernetes para la orquestación de los contenedores donde se pusieron en marcha los Microservicios.

- **<u>Antecedente 05</u>:**

  Según Nishimura y Ramírez (2022) en su investigación "**Implementación de una Arquitectura de Microservicios con Plataforma de integración y automatización de flujos de trabajo**

para la Gestión de Servicios en la Empresa automotriz HONDA PERÚ SAC."

El objetivo de este proyecto es crear una aplicación bajo la Arquitectura de Microservicios que incluya una plataforma de integración y automatización de flujos de trabajo en la nube para su uso en la gestión de servicios para la empresa automotriz HONDA PERU SAC. Esta investigación analiza las razones por las que las empresas automotrices deberían implementar estos componentes para la toma de decisiones en la gestión de servicios. (Nishimura & Ramirez, 2022)

Correlación: Este proyecto nos enseñará el desarrollo y la implementación de trabajo estandarizado, lo que ayudará a mejorar los procesos de la empresa utilizando los servicios de computación en la nube.

- **Antecedente 06:**

  Villaizán (2019) en su investigación "**Arquitectura de software basada en Microservicios para implementación de la aplicación web de cobranza digital en Financial Systems Company SAC**."

  El objetivode este proyecto de investigación fue desarrollar una aplicación de acuerdo a la Arquitectura de Microservicios, que pudiera ser utilizada en el módulo de cobranza digital de la empresa de sistemas financieros de Perú SAC. El diseño es transversal y no experimental, ya que se llevó a cabo una investigación sobre los componentes, las interacciones y los criterios de calidad para diseñar el software bajo la arquitectura de Microservicios.

Se concluyó que la Arquitectura de Microservicios puede soportar envíos de mensajes a gran escala sin afectar el rendimiento y con alta disponibilidad, según la evaluación de uso de canales digitales. (Villaizán, 2019)**.**

Correlación: Este proyecto buscará soluciones que permitan la escalabilidad del sistema y detallará el uso de estándares de la industria durante la implementación. Estas soluciones están respaldas por empresas líderes como Google y Amazon, que desempeñaron un papel fundamental como componentes integradores en esta investigación.

2.3. Formulación del Problema

¿Qué arquitectura de software del análisis comparativo, tendrá mejor impacto en el rendimiento de las aplicaciones web para la empresa 365 ENGINEERING & CONSULTING.?

2.4. Operacionalización de Variables

Tabla 1*: Las Variables y sus Indicadores*

Variables	**Indicadores**
V.I: Tipos de Arquitecturas de Software	● Multicapa ● SOA ● Microservicios
V.D: Atributos de calidad de las Arquitecturas de software	● Tiempo de respuesta. ● Disponibilidad. ● Tasa de error.

Fuente: *Elaboración propia*

2.5. Objetivo General

Realizar un análisis comparativo de las arquitecturas de software, mediante una comprobación aplicativa y seleccionar la arquitectura con mejor rendimiento para las aplicaciones web de la empresa 365 ENGINEERING & CONSULTING.

2.6. Objetivos Específicos

- ✓ Determinar los promedios de los tiempos de respuesta, por cada grupo de peticiones concurrentes y Arquitectura de software. Los tiempos se obtendrán de la herramienta Progress Telerik Fiddler Classic.
- ✓ Determinar los porcentajes de disponibilidad por cada grupo de peticiones concurrentes y Arquitectura de software. Se hallará mediante la división entre la cantidad de peticiones con respuesta satisfactoria y el total de peticiones concurrentes. Las peticiones satisfactorias se obtendrán de la herramienta Progress Telerik Fiddler Classic.
- ✓ Determinar la tasa de error por cada grupo de peticiones concurrentes y Arquitectura de software. Se hallará mediante la división entre la cantidad de peticiones erróneas o fallidas y el total de peticiones concurrentes. Las peticiones erróneas se obtendrán de la herramienta Progress Telerik Fiddler Classic.

2.7. Hipótesis

La Arquitectura de Microservicios es la que presenta mejor rendimiento que la Arquitectura Multicapa y Arquitectura Orientado a Servicios(SOA), para las aplicaciones web de la empresa 365 ENGINEERING & CONSULTING.

2.8. Justificación de la Investigación

2.8.1. Justificación Teórica

Los diversos fundamentos y teorías examinados en este proyecto de investigación desempeñaron un papel crucial al proporcionar una solidad base científica y teórica para respaldar el estudio. Estos elementos se convirtieron en fuentes vitales para futuras investigaciones, contribuyendo significativamente a la rigurosidad científica del trabajo.

2.8.2. Justificación Práctica

En esta investigación tiene como propósito de hallar la arquitectura más apropiada para aplicar en la empresa 365 ENGINEERING & CONSULTING, así como en otras empresas que están en proceso de expansión a nivel local o nacional, utilizando indicadores cuantitativos para la comparación de las arquitecturas de softwares en este estudio.

2.9. Importancia de la Investigación

Este proyecto de investigación es importante porque permite a la empresa 365 Engineering & Consulting, elegir la Arquitectura más idónea para sus aplicaciones, y de esa manera se logrará mejorar el proceso de la gestión de informes y archivos de los proyectos con sus clientes. También esta investigación, dará a conocer a la comunidad de Ingenieros y/o desarrolladores de software, sobre el análisis comparativo de las arquitecturas de software seleccionadas en este proyecto, como son la Arquitectura Multicapa, Orientada a servicios y de Microservicios. De esta manera, sea un motivo para que otros investigadores puedan realizar nuevos análisis comparativos de otras arquitecturas de software, con el objetivo de hallar la arquitectura más óptima que otorgue beneficios para el desarrollo de software y a mejorar los procesos de la empresa interesada.

CAPITULO III: MARCO TEÓRICO REFERENCIAL

3.1. Arquitectura de Software

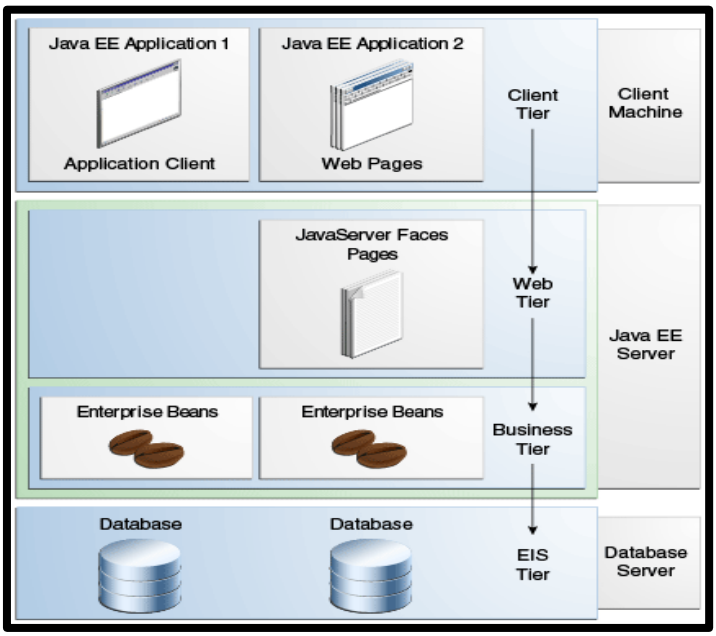
3.1.1. Definición

De acuerdo con **Len Bass, Paul Clements y Rick Kazman (2015)** mencionan lo siguiente: "La arquitectura de software es el conjunto de estructuras de soporte necesarias para razonar sobre un sistema, que incluye componentes de software, relaciones entre ellos y sus atributos." (Blass, Clements, & Kazman, 2015, págs. 4-5)

3.1.2. Arquitectura Multicapa

Según la página de Oracle(2023), menciona que "Las aplicaciones Multicapa se consideran generalmente aplicaciones de tres niveles porque están distribuidas en tres lugares: las máquinas cliente, la máquina servidor de Java EE y la base de datos en el extremo posterior. Aunque una aplicación Java EE puede consistir en todos los niveles mostrados en la figura:" (Oracle, 2023)

Figura 2: Aplicaciones de Varios Niveles



Fuente: https://docs.oracle.com/javaee/7/tutorial/overview003.htm

3.1.3. Arquitectura Single Page Application (SPA).

Según **Emmit A. Scott, Jr. (2015)** afirma que: “En una arquitectura SPA, toda la aplicación se ejecuta como una sola página web. En este enfoque, la capa de presentación de toda la aplicación se ha factorizado fuera del servidor y se gestiona desde el navegador.

El concepto de SPA lleva el desarrollo de la web a un nivel completamente nuevo al expandir las técnicas de manipulación a nivel de página de AJAX a toda la aplicación. Además, los patrones y prácticas comúnmente utilizados en la creación de un SPA pueden llevar a una eficiencia general en el diseño de la aplicación, el mantenimiento del código y el tiempo de desarrollo. Sin embargo, el éxito de la implementación de una aplicación de una sola página se verá muy afectado por su comprensión de la arquitectura del SPA.” (Scott, 2015)

3.1.4. Arquitectura Orientada a Servicios (SOA).

Según la página de Red Hat (2023), menciona que “La arquitectura orientada a servicios (SOA) es un tipo de diseño de software que permite reutilizar sus elementos a través de interfaces de servicios que se comunican a través de una red con un lenguaje común. Un servicio es una unidad de software autónoma que realiza una o más funciones y está destinada a completar una tarea.” (Red Hat, 2023).

3.1.5. Arquitectura de Microservicios.

Según Martin Fowler (2014), afirma que “Los Microservicios son un estilo arquitectónico que organiza una aplicación como un grupo de servicios más pequeños en lugar de un único servicio voluminoso. además, estos servicios más pequeños tienen el siguiente comportamiento: débilmente acoplados,

desplegables independientemente como una sola entidad y altamente mantenibles y comprobables.

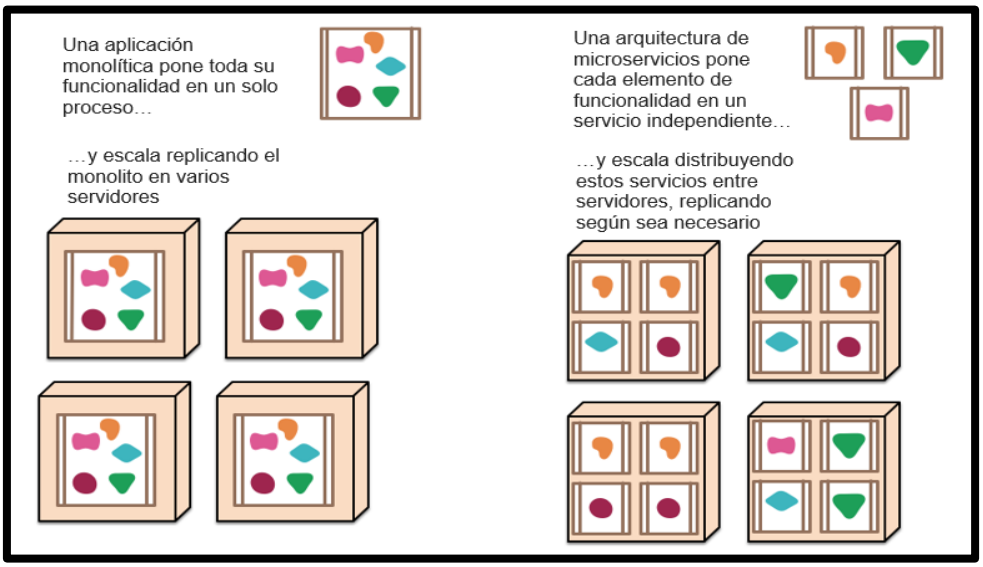
En la práctica, dividimos una aplicación grande en módulos más pequeños(servicios). sin embargo, la implementación de cada módulo se realiza como una única aplicación. cuando se produce la integración de todos los servicios (módulos), la arquitectura completa de la aplicación se conoce como Microservicios." (Fowler, 2014)

3.1.6. Arquitectura Monolítica Vs Arquitectura de Microservicios.

Según Martin Fowler (2014) menciona que "A medida que se despliegan más aplicaciones en la nube, las aplicaciones monolíticas pueden tener éxito, pero cada vez se sienten más frustradas con ellas. Los cambios realizados en una pequeña parte de la aplicación, requiere que todo el monolito sea reconstruido y desplegado. Con el tiempo, a menudo es difícil mantener una buena estructura modular y también mantener los cambios que sólo deberían afectar a un módulo dentro de ese módulo. El escalado implica a toda la aplicación en lugar de partes de ella que requieren mayores recursos." (Fowler, 2014)

Debido a estas frustraciones, se ha desarrollado el estilo arquitectónico de Microservicios, que consiste en construir aplicaciones como conjuntos de servicios. Los servicios no solo tienen la capacidad de desplegarse y escalar de forma independiente, sino que también tienen un límite modular sólido que permite incluso crear servicios distintos en diferentes lenguajes de programación. Varios equipos también pueden supervisarlos.

Figura 3: Monolitos y Microservicios



Fuente: https://martinfowler.com/articles/microservices.html

3.2. Aplicaciones Web

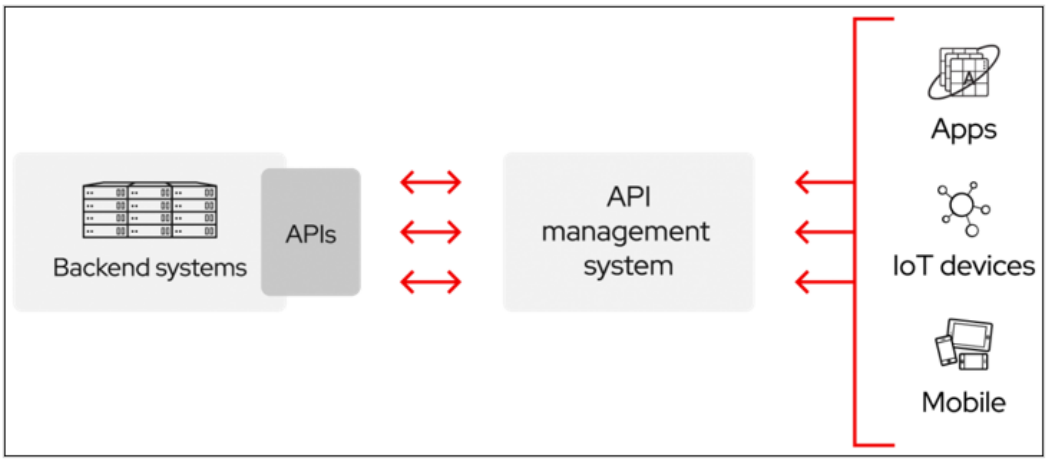
Según Amazon Web Service (2023) define que: "Una aplicación web es un software que se ejecuta en un navegador web. Dado que las empresas tienen que intercambiar información y proporcionar servicios de forma remota. Utilizan aplicaciones web para comunicarse con los clientes cuando lo necesiten y de una forma segura. Las funciones más comunes de los sitios web, como los carros de compra, la búsqueda y el filtrado de productos. Le permiten acceder a funcionalidades complejas sin la necesidad de instalar un software." (AWS, 2023)

Según Microsoft (2023) afirma que: "Las aplicaciones web modernas tienen expectativas más altas por parte del usuario y mayores demandas que nunca. Las aplicaciones web deben ser seguras, flexibles y escalables para satisfacer los picos de demanda. Cada vez más, los escenarios complejos deben controlarse mediante experiencias de usuario enriquecidas creadas en el cliente con JavaScript y comunicarse de forma eficaz a través de las API web." (Microsoft, 2023)

3.3. Interfaz de Programación de Aplicaciones (API)

Según la página de Red Hat (2023) describe que: “El término API es una abreviatura de Application Programming Interfaces, que en español significa interfaz de programación de aplicaciones. Las APIs permiten que sus servicios se comuniquen con otros, sin necesidad de conocer cómo están desplegados. Esto simplifica el desarrollo de las aplicaciones y permite ahorrar tiempo y dinero.” (Hat, 2023)

Figura 4: ¿Cómo Trabaja una API?



Fuente: https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

3.4. Servicios Restful

Según la página de Oracle (2013), mencionan que “Los servicios web de RESTful están construidos para funcionar mejor en la web. La transferencia de estado de representación (REST) es un estilo arquitectónico que especifica las restricciones, como la interfaz uniforme, que si se aplican a un servicio web inducen propiedades deseables, como el rendimiento, la escalabilidad y la modificabilidad, que permiten que los servicios funcionen mejor en la Web.” (ORACLE, 2013)

3.5. Pruebas de Software

3.5.1. Definición.

De acuerdo a la página oficial de IBM (2023), afirman que “La prueba de software es el proceso de evaluar que una aplicación de software hace lo

que debería hacer. Los resultados positivos de las pruebas integran la prevención de errores, la reducción de los costos de desarrollo y la optimización del rendimiento.” (IBM, 2023)

3.5.2. Pruebas de Rendimiento.

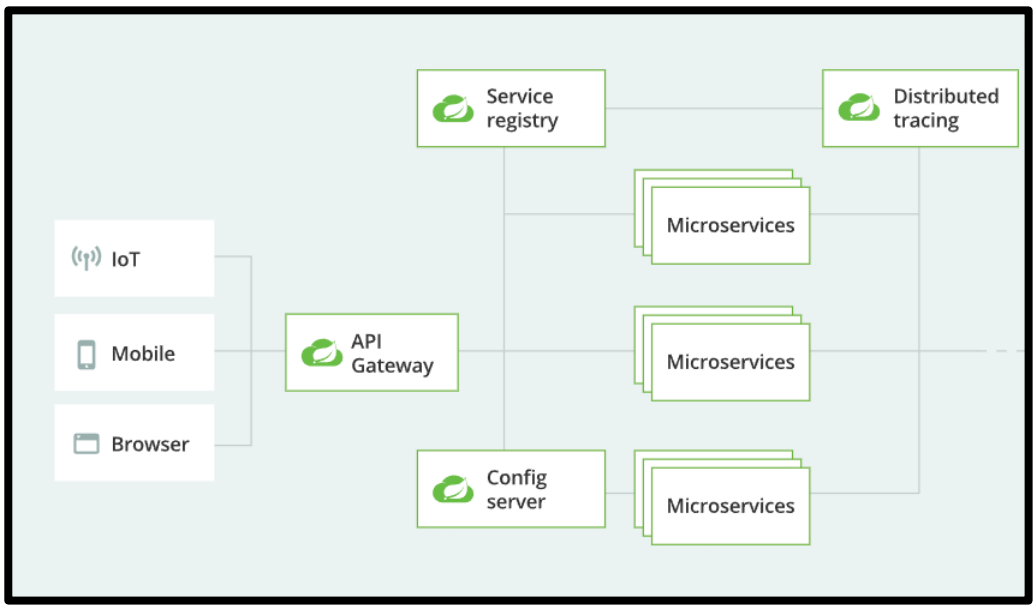
De acuerdo a la página oficial de IBM (2021), afirman que “El objetivo de las pruebas de rendimiento es evaluar la capacidad del sistema bajo una carga de trabajo establecida, utilizando diferentes tipos de pruebas de rendimiento tales como las pruebas de carga y de estabilidad. Las pruebas de carga, por ejemplo, se utilizan para evaluar el rendimiento en condiciones de carga reales. (IBM, 2021)

3.6. Herramientas Tecnológicas Empleadas

3.6.1. Spring Cloud.

De acuerdo a la página de Spring, afirman que “Spring Cloud ofrece herramientas para permitir que los desarrolladores construyan inmediatamente algunos patrones comunes en los sistemas distribuidos (ejemplo: gestión de configuración, descubrimiento de servicios, microproxy, sesiones distribuidas, etc.). Utilizando Spring Cloud los desarrolladores pueden poner en marcha rápidamente servicios y aplicaciones que implementen esos patrones.” (VMWARE, 2023)

Figura 5: Arquitectura de Spring Cloud



Fuente: https://spring.io/cloud

3.6.2. Spring Framework.

De acuerdo a la página de Spring, afirman que "Spring Framework es un marco Java de código abierto que se utiliza para desarrollar aplicaciones de nivel empresarial. Proporciona una amplia gama de características y funcionalidades como la inversión de control (IoC), la inyección de dependencias (DI), la programación orientada a aspectos (AOP) y mucho más. Estas características ayudan a los desarrolladores a crear aplicaciones robustas, escalables y mantenibles." (VMWARE, 2023)

3.6.3. Spring Boot

De acuerdo a la página de Spring, afirman que "Spring Boot facilita la creación de aplicaciones independientes basadas en Spring de nivel de producción que se pueden ejecutar simplemente.

Se adopta un punto de vista crítico sobre la plataforma Spring y las librerías de terceros para que pueda empezar con un mínimo de complicaciones. La

mayoría de las aplicaciones Spring Boot necesitan una configuración mínima de Spring.” (VMware, 2023)

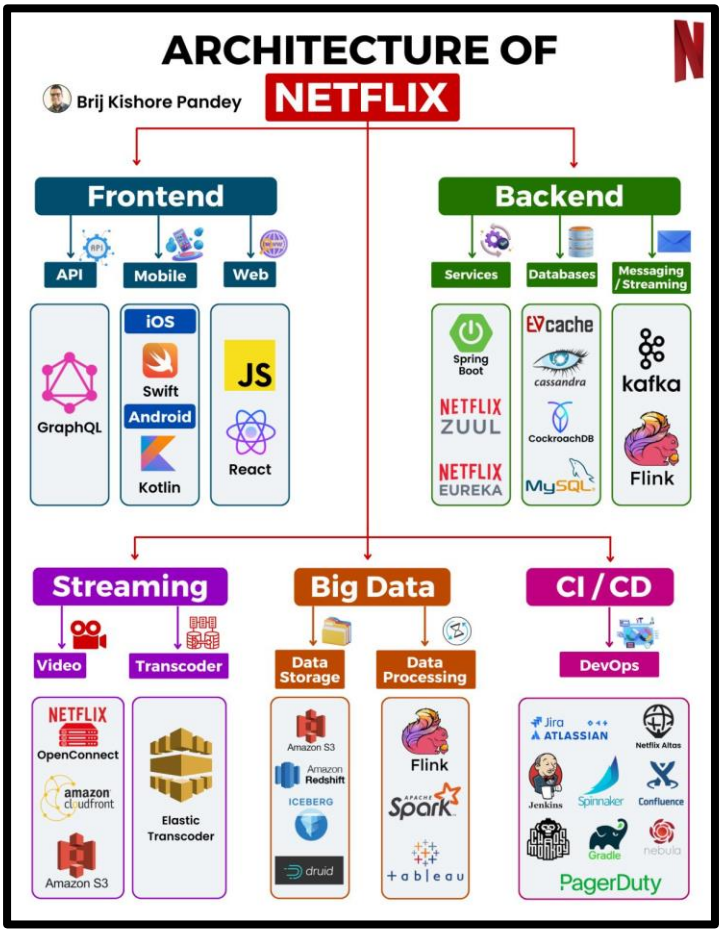
3.6.4. Netflix OSS

De acuerdo a la página de Spring, afirman que “Spring Cloud Netflix proporciona integraciones OSS de Netflix para aplicaciones de Spring Boot a través de la autoconfiguración y la vinculación con el entorno de Spring y otros modismos del modelo de programación de Spring. Con unas simples anotaciones, puede habilitar y configurar rápidamente los patrones comunes dentro de su aplicación y construir grandes sistemas distribuidos con componentes de Netflix. Los patrones proporcionados incluyen:

- Configuración distribuida.
- Registro y auto reconocimiento de servicios.
- Enrutador.
- Llamadas servicio a servicio.
- Balanceo de carga.
- Control de ruptura de comunicación con los servicios.
- Clusterización.
- Mensajería distribuida.” (VMWARE, 2023)

Netflix utiliza servicios en la nube para alojar su plataforma y la mayoría de sus aplicaciones. En particular, utilizan Amazon Web Services (AWS) como su principal proveedor de la nube. Se muestra en la siguiente imagen la Arquitectura de Netflix, que permite escalar rápidamente su plataforma, adaptándose al crecimiento del tráfico y al número de usuarios de sus servicios.

Figura 6: Arquitectura de NETFLIX



Fuente: https://www.linkedin.com/posts/brijpandeyji_systemdesign-coding-softwareengineering-activity-7081232542984511489-pJFv/

3.6.5. Spring Cloud Config

De acuerdo a la página de Spring, afirman que "Spring Cloud Config Server proporciona una API basada en recursos HTTP para la configuración externa (pares nombre-valor o contenido YAML equivalente). Como todas las aplicaciones de Spring Boot, se ejecuta en el puerto 8080 por defecto, pero se puede cambiar al puerto más convencional 8888 de varias maneras." (VMWARE, 2023).

3.6.6. Spring Boot Admin

Según la página de Spring Boot Admin Docs, definen que "Spring Boot Admin es una herramienta de monitorización que pretende visualizar la

información proporcionada por los Actuadores de Spring Boot de una forma agradable y accesible. Consta de dos partes principales:

- Un servidor que proporciona una interfaz de usuario para interactuar con los Actuadores Spring Boot.
- Un cliente que se utiliza para registrarse en el servidor y permitir el acceso a las URLs de los actuadores.” (Codecentric, 2023)

3.6.7. Eureka

Según la página de Spring (2023), mencionan que “El descubrimiento de servicios es uno de los principios clave de una arquitectura basada en microservicios. Tratar de configurar a mano cada cliente o alguna forma de convención puede ser difícil de hacer y puede ser frágil. Eureka es el servidor y el cliente de Netflix Service Discovery. El servidor puede ser configurado y desplegado para estar altamente disponible, con cada servidor replicando el estado sobre los servicios registrados a los demás.” (VMWARE, 2023).

3.6.8. Zuul

Según la página de Spring (2023), mencionan que “El enrutamiento es una sección integral de una arquitectura de microservicios. Zuul es un enrutador con base en JVM y un equilibrador de carga del lado del servidor de Netflix. El motor de reglas de Zuul permite escribir reglas y filtros en prácticamente cualquier lenguaje JVM, con soporte incorporado para Java y Groovy.” (VMWARE, 2023)

3.6.9. Circuit Breaker: Hystrix Clients

Según la página de Spring (2023), mencionan que “Netflix ha creado una biblioteca llamada Hystrix que implementa el patrón de interruptores. En

una arquitectura de microservicios, es habitual tener varias capas de llamadas a servicios.” (VMWARE, 2023)

El principio es análogo a la electrónica: Hystrix vigila los métodos para detectar fallos en las llamadas a servicios relacionados. Si se produce un fallo, abrirá el circuito y reenviará la llamada a un método alternativo.

3.6.10. Java Server Faces.

De acuerdo a la página de Oracle (2023), mencionan que “JavaServer Faces (JSF) es una especificación estandarizada para crear interfaces de usuario (UI) para aplicaciones del lado del servidor. Antes de JavaServer Faces, los desarrolladores de aplicaciones web solían recurrir a la creación de componentes de interfaz de usuario HTML con servelets o páginas JavaServer (páginas JSP). Esto se debe principalmente a que los componentes de interfaz de usuario HTML son el mínimo común denominador que admiten los navegadores web.” (ORACLE, 2023)

JavaServer Faces es una tecnología liderada por Sun Microsystems como JSR 127 en el marco del Java Community Process (JCP). Su objetivo es crear un marco estándar para componentes de interfaz de usuario para aplicaciones web.

3.6.11. Angular

Según la página de Angular (2023), mencionan que “es un framework de diseño de aplicaciones y plataformas de desarrollo para generar aplicaciones de una sola página eficientes y sofisticadas o basado en la arquitectura Single Page Application(SPA**)**. Angular utiliza el enfoque de programación declarativa y se centra en la manipulación del DOM (Document Object

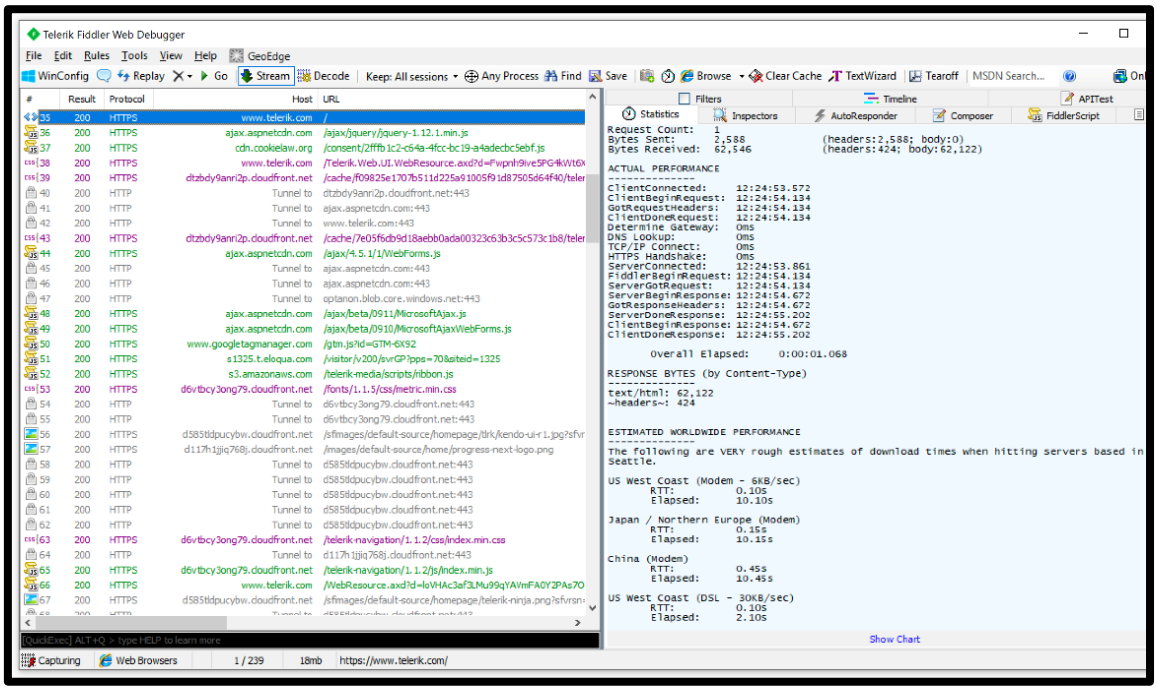
Model) para lograr la interactividad en las aplicaciones web”.” (ANGULAR, 2023)

3.6.12. Herramienta de Prueba de Carga: Fiddler Classic.

Según la página de Fiddler Classic (2023), definen que “Fiddler Classic es una herramienta de servidor proxy para depurar el tráfico web de aplicaciones como los navegadores. Se utilizan para capturar y grabar este tráfico web y luego reenviarlo a un servidor web. Las respuestas del servidor se devuelven a la herramienta Fiddler y luego al cliente. El tráfico web grabado se presenta a través de una lista de sesiones en la interfaz de usuario de depuración web de Fiddler” (Progress, 2023).

Se observa un ejemplo del mapeo de las peticiones, en la figura 7:

Figura 7: Fiddler Classic



Fuente: https://www.telerik.com/blogs/understanding-telerik-fiddler-as-a-proxy

CAPITULO IV: METODOLOGÍA DE DESARROLLO

4.1. Análisis de Requisitos

En la metodología ICONIX inicia con el "modelo del dominio", para el cual es necesario generar un listado de requisitos. En esta sección, se mostrarán los requisitos funcionales y los no funcionales, previo al desarrollo de las aplicaciones web.

4.1.1. Requisitos Funcionales

- R01: El usuario debe iniciar sesión (con su nombre de usuario y password previamente creado).
- R02: El sistema tendrá la capacidad de cerrar sesión, que permitirá al usuario salir de la aplicación.
- R03: El sistema tendrá la capacidad de crear usuario.
- R04: El sistema tendrá la capacidad de editar usuario.
- R05: El sistema tendrá la capacidad de dar de baja a un usuario.
- R06: El sistema tendrá la capacidad de crear un proyecto.
- R07: El sistema tendrá la capacidad de editar el proyecto.
- R08: El sistema tendrá la capacidad de dar de baja un proyecto.
- R09: El sistema tendrá la capacidad de agregar el archivo del proyecto.
- R10: El sistema permitirá descargar el archivo de proyecto.

4.1.2. Requisitos No Funcionales

a) Requisitos de las Computadoras

- NF01: Sistema Operativo Windows 10.
- NF02: Servidor Web Glassfish server (para la aplicación JSF) y Tomcat embebido (para las aplicaciones Spring Boot).

- NF03: Motores de Base de Datos Relacionales: PostgreSQL 12 y Mysql server 5.7.
- NF04: Memoria RAM de 32GB y 12 GB, para la PC 01 y PC02 respectivamente.
- NF05: CPU 16 núcleos lógicos con 2.5 GHz y 4 núcleos lógicos con 2.3GHz, para la PC 01 y PC02 respectivamente.

b) Requisitos de Usabilidad

- Las interfaces de usuario de las aplicaciones web, serán muy sencillas.
- Las aplicaciones serán muy fáciles de adaptar y utilizar para los usuarios.

c) Requisitos de escalabilidad

- La aplicación web basada en la Arquitectura de Microservicios, deberá ser capaz de escalar horizontalmente, esto significa manejar un aumento en la distribución de manera eficiente en la carga de trabajo entre múltiples instancias o nodos.

4.2. Modelado del Dominio

4.2.1. Listado De Objetos o Clases de Dominio

a) Usuario: Es la persona que se conecta al sistema para hacer uso de los servicios que este les proporciona.

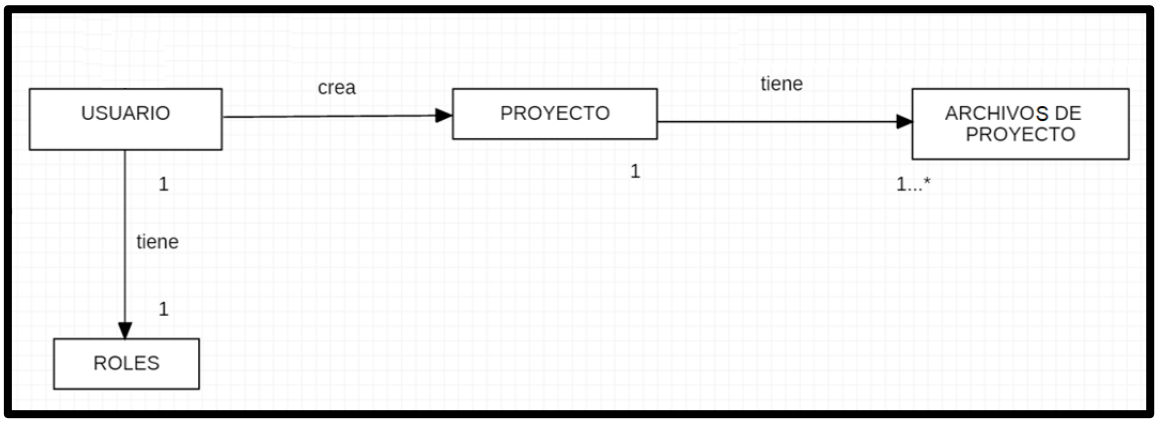
b) Proyecto: Es la entidad donde se registra los datos pertinentes de los proyectos contratados con los clientes de la empresa 365 ENGINEERING & CONSULTING.

c) Archivos de Proyecto: Los archivos contienen la información del Proyecto en curso o finalizado.

d) Rol: Es un conjunto de privilegios que se otorga a un usuario o grupo de usuarios para que puedan realizan ciertas tareas en el sistema.

4.2.2. Modelo de Dominio Inicial

Figura 8: Modelo de Dominio Inicial

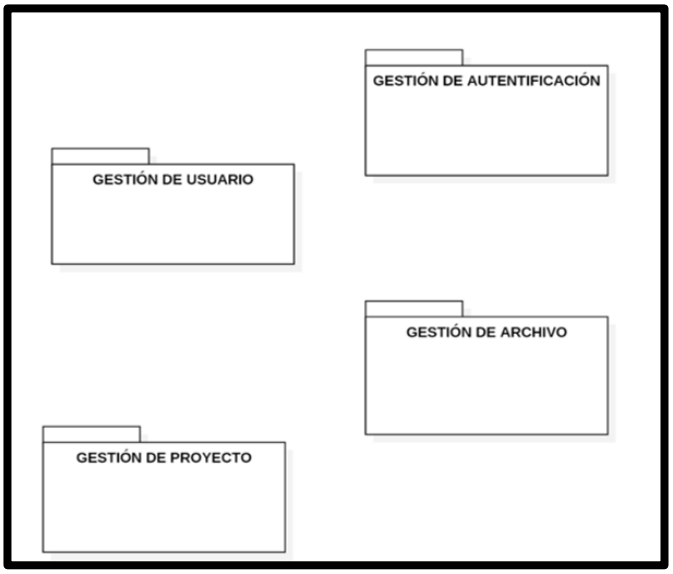


Fuente: *Elaboración propia.*

4.2.3. Modelado de Casos de Uso

4.2.3.1. Diagrama de Paquetes de Casos de Uso

Figura 9: Diagrama de Paquetes de Casos de Uso

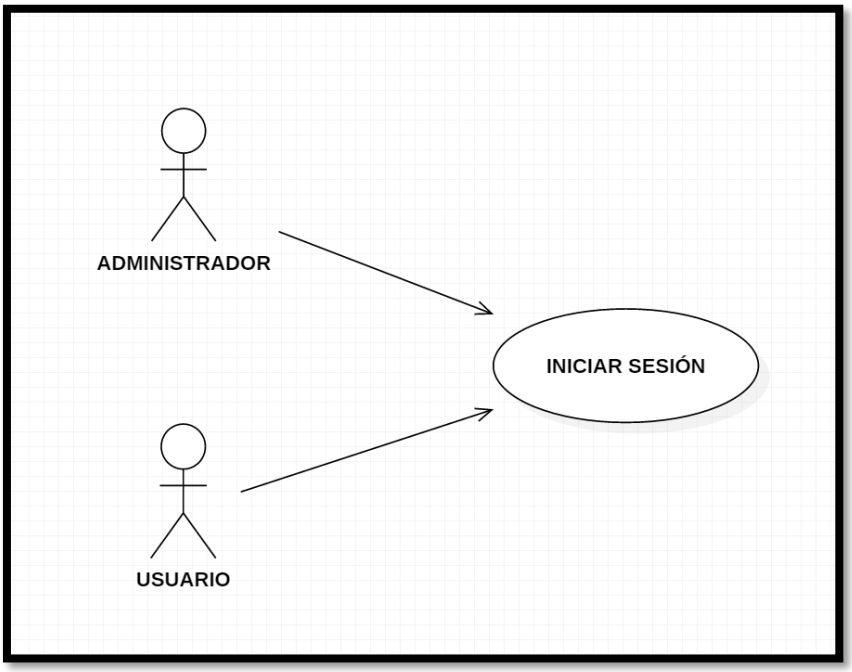


Fuente: *Elaboración propia*

4.2.3.2. Diagrama de Casos de Uso

a) Gestión de Autentificación

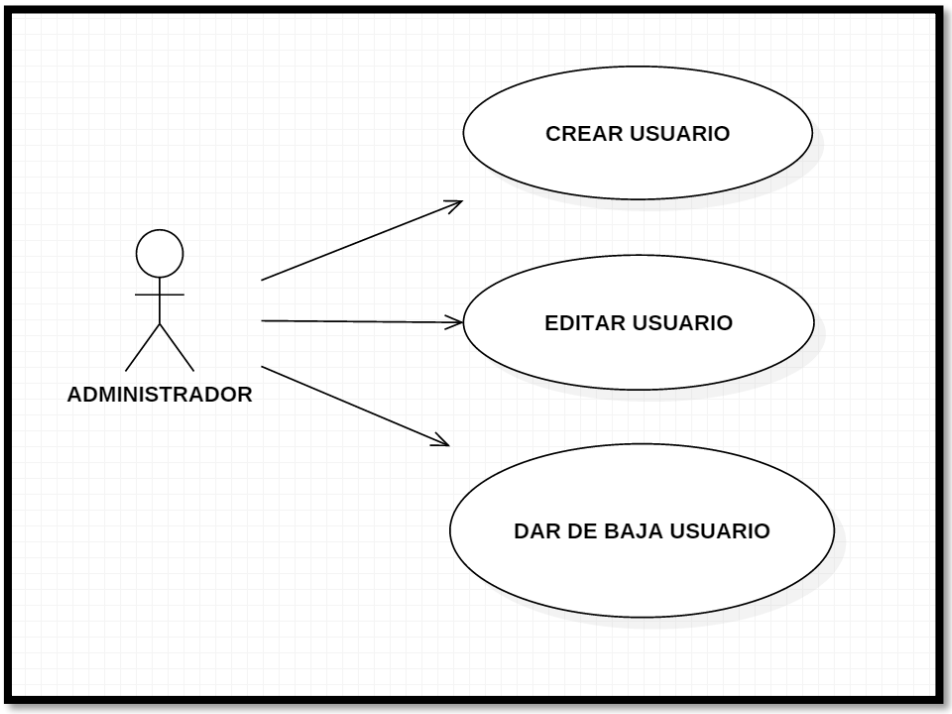
Figura 10: Gestión de Autentificación



Fuente: *Elaboración propia*

b) Gestión de Cuentas

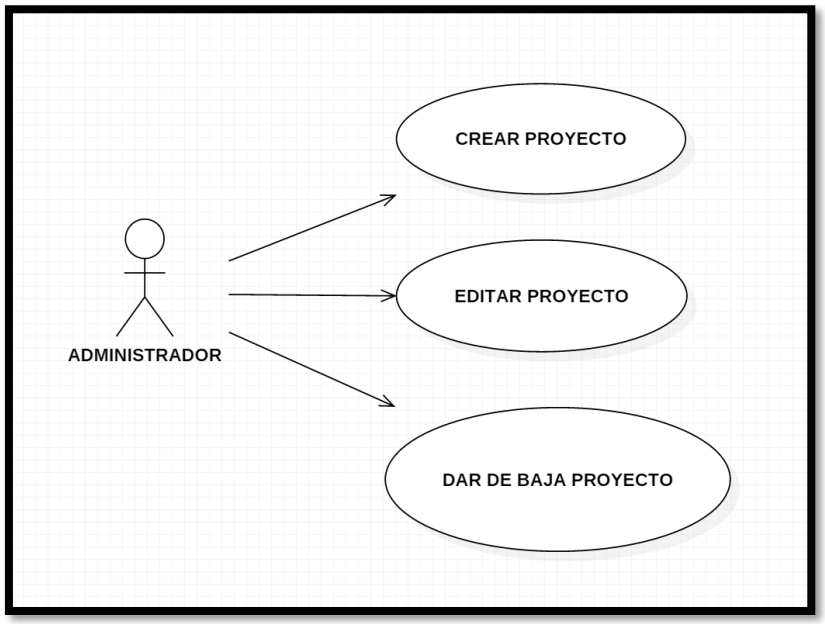
Figura 11: Gestión de Cuentas



Fuente: *Elaboración propia*

c) Gestión de Proyecto

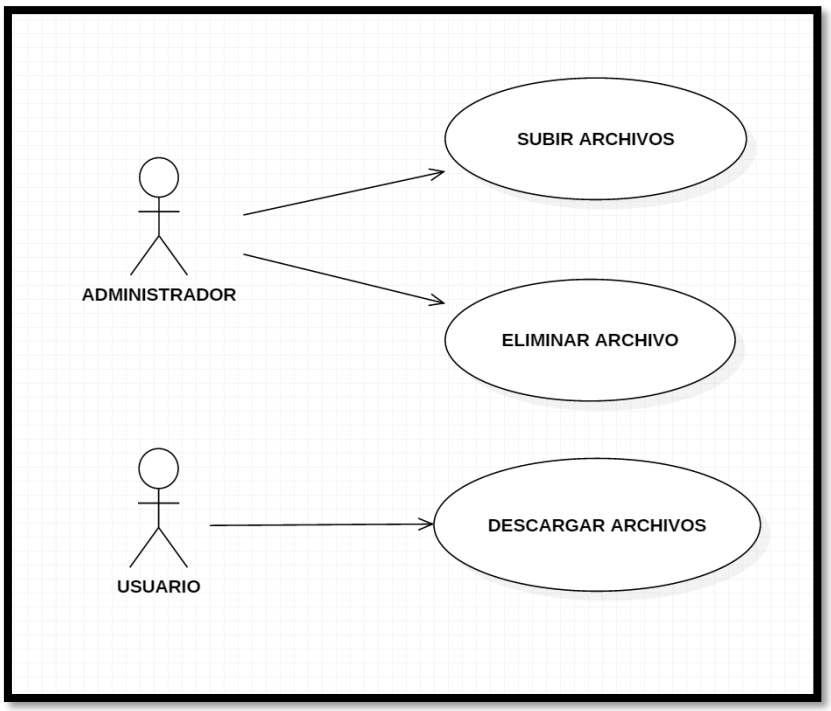
Figura 12: Gestión de Proyecto



Fuente: *Elaboración propia*

d) Gestión de Archivo de Proyectos

Figura 13: Gestión de Archivos del Proyecto



Fuente: *Elaboración propia*

4.3. Análisis y Diseño Preliminar

4.3.1. Especificación de Casos de Uso

Durante esta etapa, se elaborará una descripción textual minuciosa de cada caso de uso.

Tabla 2*: Caso de uso Iniciar Sesión*

Especificación de Caso de Uso	
Caso de Uso	Iniciar Sesión
Precondiciones	Estar conectado a la intranet/internet
Flujo de Eventos	**Básico** ● Cuando el usuario hace clic en el enlace "Iniciar sesión" en la Página Principal, el sistema mostrara la Página de Inicio de sesión. A continuación, el usuario introduce el "Usuario" y "Password" y seleccionara la opción "INGRESAR". ● El sistema comprobara la existencia de los datos ingresados por el usuario. Si los datos son válidos, el sistema autenticara al usuario en la sesión y lo enviara a la página de listado de usuarios del sistema.
	Alternativos • **Datos inválidos:** El sistema redirige nuevamente a la pantalla de Inicio de Sesión y mostrará un mensaje indicando que el "El nombre de usuario y/o password es incorrecto".

	• **Usuario cancela inicio de sesión:** cuando el usuario seleccione la opción "Salir", el sistema muestra nuevamente la Página Principal de Inicio de Sesión.
Post condiciones	Se observa el nombre del usuario en la sesión

Tabla 3: *Caso de uso Crear Usuario*

Especificación de Caso de Uso	
Caso de Uso	Crear Usuario
Precondiciones	Haber ingresado al Sistema
Flujo de Eventos	**Básico** ● El usuario deberá dirigirse al menú "Usuarios" y el sistema le mostrará la ventana de "Usuarios", para crear un usuario, el usuario deberá de dirigirse al botón "Nuevo", el sistema le mostrará la ventana de "Nuevo Usuario". El usuario deberá loguear su "Usuario", "Password", "Estado (activo/inactivo)", "Rol (ADMIN/USER)". ● El sistema guardará los datos registrados por el usuario satisfactoriamente, mostrando un mensaje "Usuario añadido correctamente".
	Alternativos • **Datos inválidos:** El sistema mostrará un mensaje si se ingresó datos incorrectos o

		duplicados, mostrando siguiente mensaje: "El nombre de usuario y/o password es incorrecto o duplicado".
Post condiciones	Los datos del usuario se registrarán en la base de datos	

Tabla 4: *Caso de uso Editar Usuario*

Especificación de Caso de Uso		
Caso de Uso	Editar Usuario	
Precondiciones	Haber ingresado al Sistema	
Flujo de Eventos	**Básico**	● El usuario deberá entrar al menú "Usuario" y el sistema le mostrara la ventana de "Usuario", para modificar los datos de un usuario, el usuario deberá de dirigirse al botón "Modificar", el sistema la mostrara la ventana de "Modificar Usuario". El usuario podrá modificar su "Usuario", "Password", "Estado (activo/inactivo)", "Rol (ADMIN/USER)". ● El sistema guardará los datos modificados por el usuario satisfactoriamente, mostrando un mensaje "Usuario modificado correctamente".
	Alternativos	• **Datos inválidos:** El sistema mostrará un mensaje si se ingresó datos incorrectos o duplicados, mostrando el siguiente mensaje:

	"El nombre de usuario y/o password es incorrecto o duplicado".
Post condiciones	Los datos del usuario se actualizarán en la base de datos

Tabla 5: *Caso de uso Dar de baja Usuario*

Especificación de Caso de Uso		
Caso de Uso	Dar de baja Usuario	
Precondiciones	Haber ingresado al Sistema	
Flujo de Eventos	**Básico**	● El usuario deberá dirigirse al menú "Usuarios" y el sistema muestra el listado de Usuarios. Para dar de baja a un usuario, deberá de dirigirse a la columna "Modificar". ● El sistema le mostrará la ventana de Editar Usuario, donde se observa el botón switch, para cambiar el estado. ● El usuario deberá elegir la opción "No", con lo cual el sistema dará de baja satisfactoriamente al usuario.
	Alternativos	Ninguno
Post condiciones	Se muestra la página de Listado de Usuarios.	

Tabla 6*: Caso de uso Crear Proyecto*

Especificación de Caso de Uso		
Caso de Uso	Crear Proyecto	
Precondiciones	Haber ingresado al Sistema	
Flujo de Eventos	**Básico**	● El usuario deberá dirigirse al menú "Proyectos" y dar clic en el botón "Nuevo", el sistema le mostrará la pantalla de "Nuevo". El usuario deberá ingresar su nombre de "Proyecto", "Descripción", "Estado (activo/inactivo)", "Usuario", "Fecha de Expiración". ● El sistema guardará los datos registrados por el usuario satisfactoriamente.
	Alternativos	• **Datos inválidos o faltantes:** El sistema resaltará los campos en rojo , si hay datos que faltasen ingresar en el formulario , por ende no podrá guardar los cambios.
Post condiciones	Se muestra la página de Listado de Proyectos.	

Tabla 7*: Caso de uso Editar Proyecto*

Especificación de Caso de Uso		
Caso de Uso	Editar Proyecto	
Precondiciones	Haber ingresado al Sistema	
Flujo de Eventos	**Básico**	● El usuario deberá dirigirse al menú "Proyecto" y el sistema le mostrará la

		ventana de “Proyecto”, para modificar los datos de un proyecto, el usuario deberá de dirigirse al botón “Modificar”. ● El sistema mostrará la pantalla de “Modificar Proyecto”. El usuario podrá modificar el nombre de “Proyecto”, “Descripción”, “Estado (activo/inactivo)”, “Usuario”, “Fecha de Expiración”. ● El sistema guardará los datos modificados por el usuario satisfactoriamente, mostrando un mensaje “Proyecto modificado correctamente”.
	Alternativos	• **Datos inválidos o faltantes:** El sistema resaltará los campos en rojo , si hay datos que faltasen ingresar en el formulario , por ende no podrá guardar los cambios.
Post condiciones	Se muestra la página de Listado de Proyectos.	

Tabla 8*:* *Caso de uso Dar de baja Proyecto*

Especificación de Caso de Uso		
Caso de Uso	Dar de baja Proyecto	
Precondiciones	Haber ingresado al Sistema	
Flujo de Eventos	**Básico**	● El usuario deberá dirigirse al menú “Proyecto”, luego el sistema le mostrará el Listado de proyectos. El usuario deberá dar

		clic en **Modificar**, lo cual el sistema le mostrará una ventana encima, para que el usuario pueda dar de baja al Proyecto respectivo. ● El usuario deberá dar clic en botón “estado”, cambiando a inactivo y luego dar clic en guardar. ● Por último, el sistema le mostrará un mensaje : “se actualizó correctamente”, luego se dirigirá a la página de Proyectos.
	Alternativos	Ninguno
Post condiciones	Se muestra la página de Listado de Proyectos.	

Tabla 9*:* *Caso de uso Agregar Archivo del Proyecto*

Especificación de Caso de Uso		
Caso de Uso	Agregar Archivo del Proyecto	
Precondiciones	Haber ingresado al Sistema	
Flujo de Eventos	**Básico**	● El usuario deberá dirigirse al menú “Proyecto” y el sistema le mostrará la ventana de “Proyecto”, para añadir un archivo el usuario deberá dar clic en el icono del archivo; en la columna “Archivo”. ● El sistema le mostrara a continuación la ventana de “Archivos de Proyecto”.

	● El usuario deberá dar clic al botón “Nuevo Archivo”. ● el sistema le mostrara la ventana de “Nuevo Archivo”, esta ventana tiene la opción de un botón “Seleccionar Archivo”, lo cual el usuario podrá dar clic en esa opción y agregar el archivo que corresponda. ● El sistema guardará el archivo añadido por el usuario satisfactoriamente mostrando un mensaje “Correcto, Se subió el archivo”.
Alternativos	• **Datos inválidos:** El sistema mostrara un mensaje "Incorrecto, Archivo debe ser menor o igual a 3 MB” si el archivo no es del formato indicando anteriormente o si su peso es mayor a lo solicitado. • **Corrección de archivo:** El sistema permitirá al usuario de ser necesario “Modificar” el archivo subido anteriormente.
Post condiciones	Se muestra la página de Listado de Archivos del Proyecto.

Tabla 10: *Caso de uso Descargar Archivo de Proyecto*

Especificación de Caso de Uso	
Caso de Uso	Descargar Archivo de Proyecto
Precondiciones	Haber ingresado al Sistema

Flujo de Eventos	**Básico**	● El usuario deberá dar clic en la opción del menú: “Proyectos”. ● El sistema le mostrará una tabla del listado de Proyectos, donde cada registro tiene el icono, en la columna Archivo, lo cual el usuario debe dar clic en ese icono. ● El sistema le mostrara el Listado de archivos del proyecto. ● El usuario deberá dar clic en el icono de descarga, del nombre del archivo que necesite descargar. ● El sistema le permitirá ver el archivo, descargado en formato PDF.
	Alternativos	• **Descarga inválida:** El sistema no dejará descargar el archivo si la fecha actual supera a la fecha de expiración.
Post condiciones	Se muestra la página de Listado de Archivos del Proyecto.	

4.4. Arquitectura Técnica

El propósito de la arquitectura técnica es tener una visión de la arquitectura tecnología para dar entender cómo los componentes de las aplicaciones web se comunican e interactúan entre sí.

4.4.1. Arquitecturas de las aplicaciones web de la empresa

- **Arquitectura de Software Multicapa**

La Aplicación Web será desarrollada con el Framework Java Server Faces 2.1

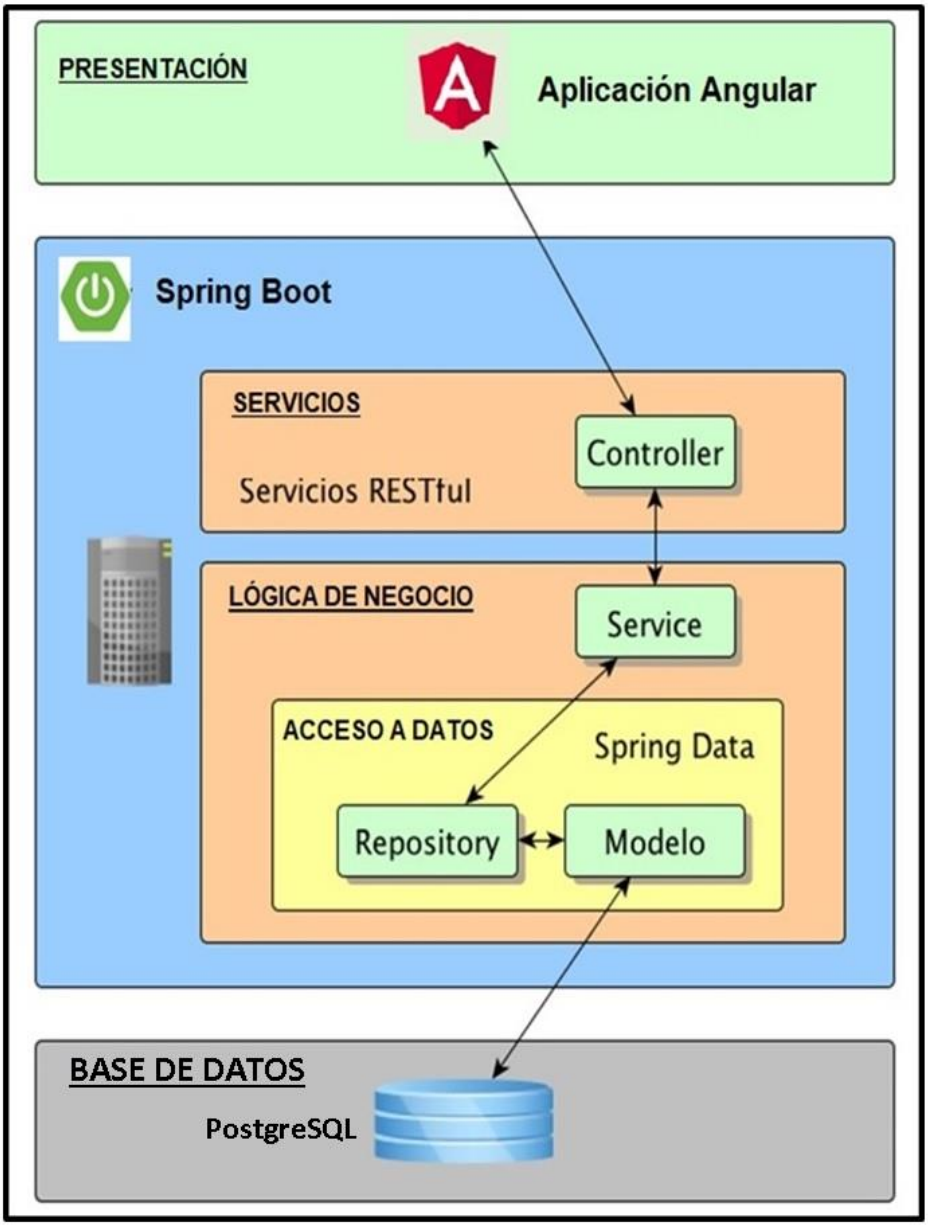
Figura 14: Arquitectura Multicapa

Fuente: Elaboración propia

- **Arquitectura Orientada a Servicios (SOA)**

  Para esta Arquitectura, el sistema será desarrollado con el framework Spring, tanto para la capa de lógica de Negocio y Acceso a Datos. Del lado de Capa presentación, será desarrollado con el framework Angular

Figura 15: Arquitectura S.O.A

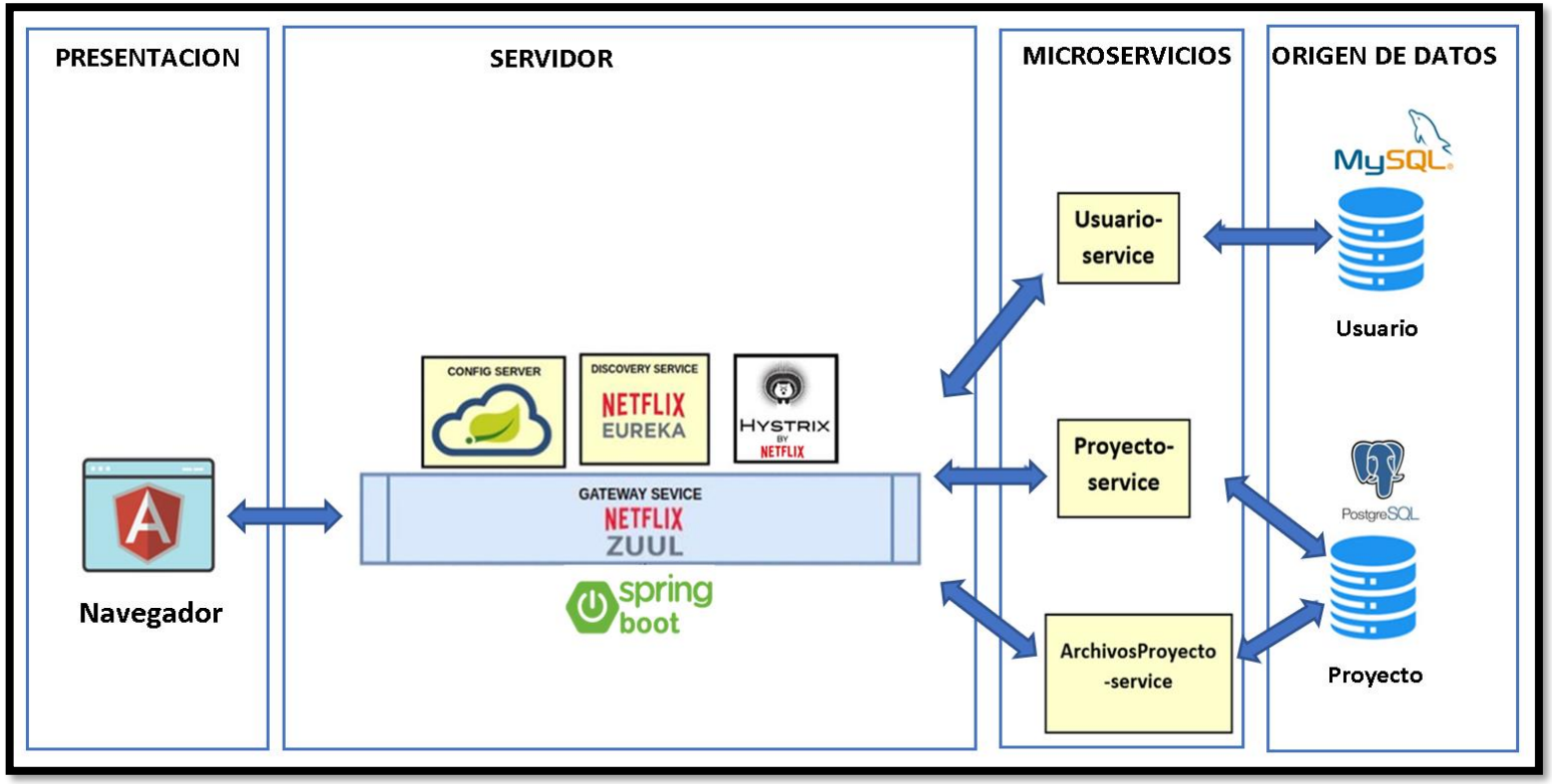


Fuente: Elaboración propia

- **Arquitectura de Microservicios**

  En esta arquitectura, el sistema será desarrollado con el framework Angular; del lado de la capa de Presentación. Del lado del servidor será con el framework Spring, para la capa de servicios, lógica de Negocio y Acceso a Datos.

Figura 16: Arquitectura de Microservicios

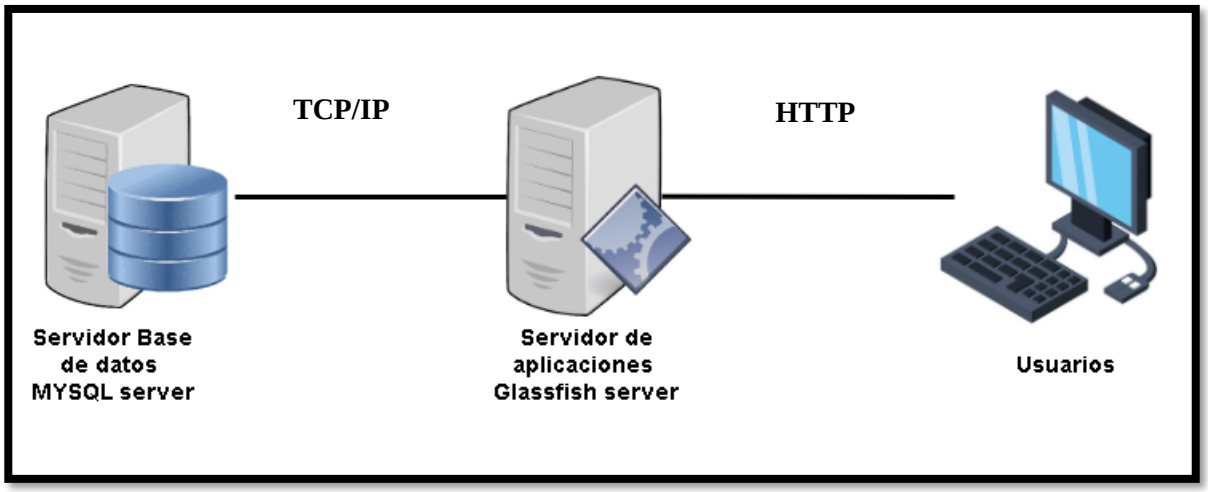


Fuente: Elaboración propia

4.4.2. Diagramas de Despliegue de las Arquitecturas Propuestas

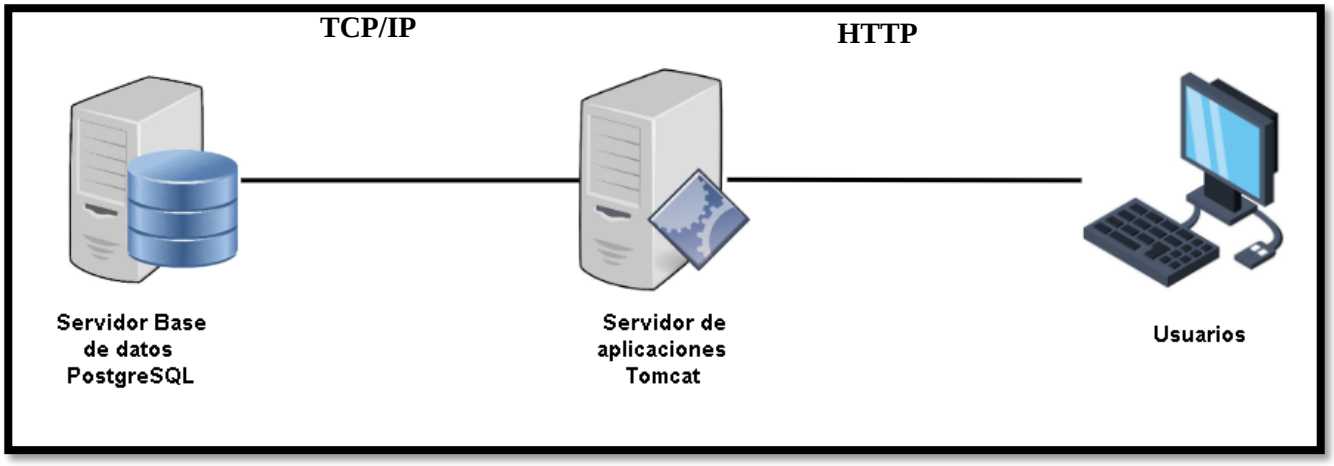
Los diagramas de despliegue muestran las relaciones físicas entre los nodos que componen un sistema (nodos del servidor, el servidor de aplicaciones, gestor de base de datos, etc.).

Figura 17: Diagrama de Despliegue de la Arquitectura Multicapa



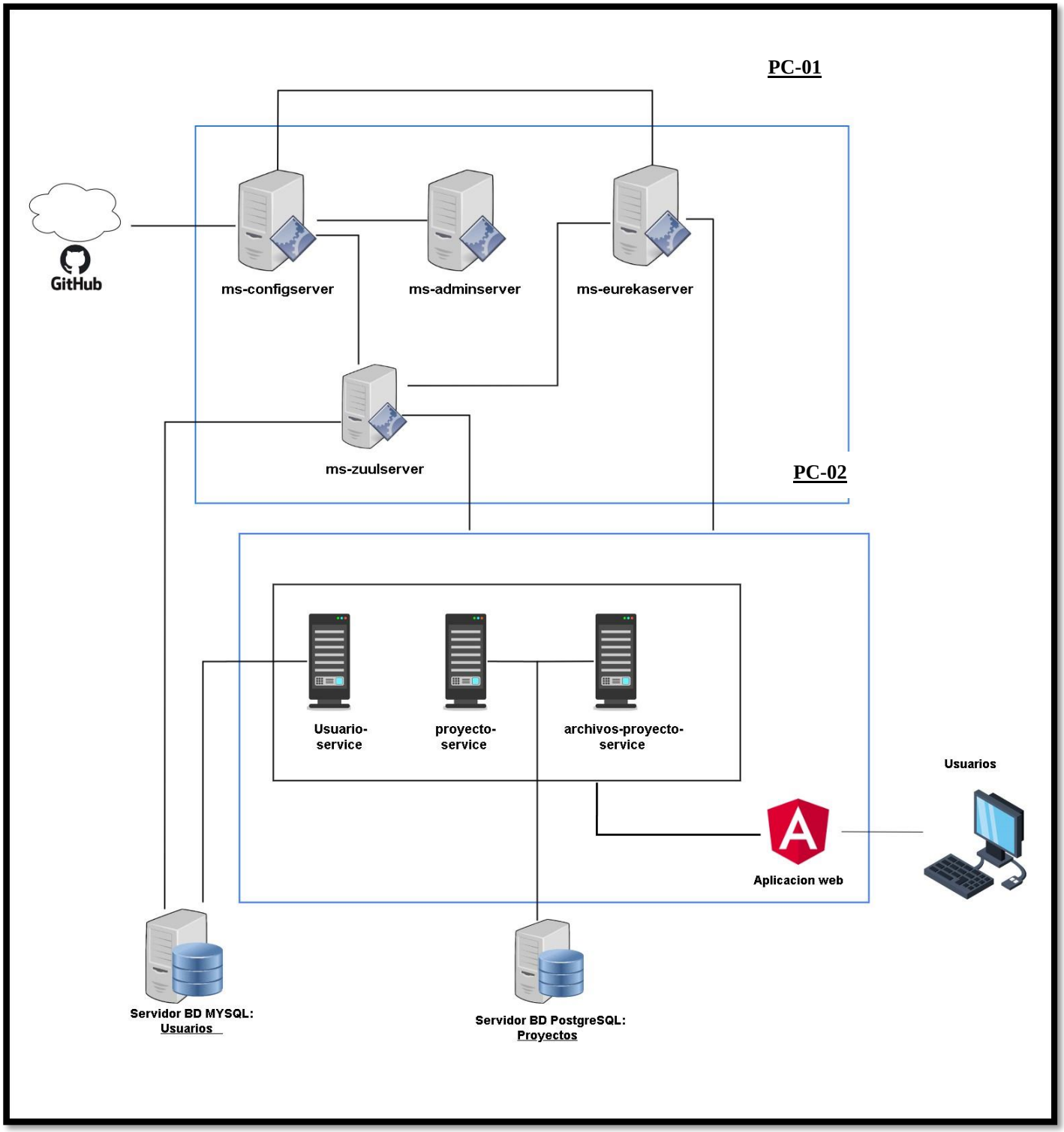
Fuente: Elaboración propia

Figura 18: Diagrama de Despliegue de la Arquitectura S.O.A.



Fuente: Elaboración propia

Figura 19:Diagrama de Despliegue de la Arquitectura de Microservicios

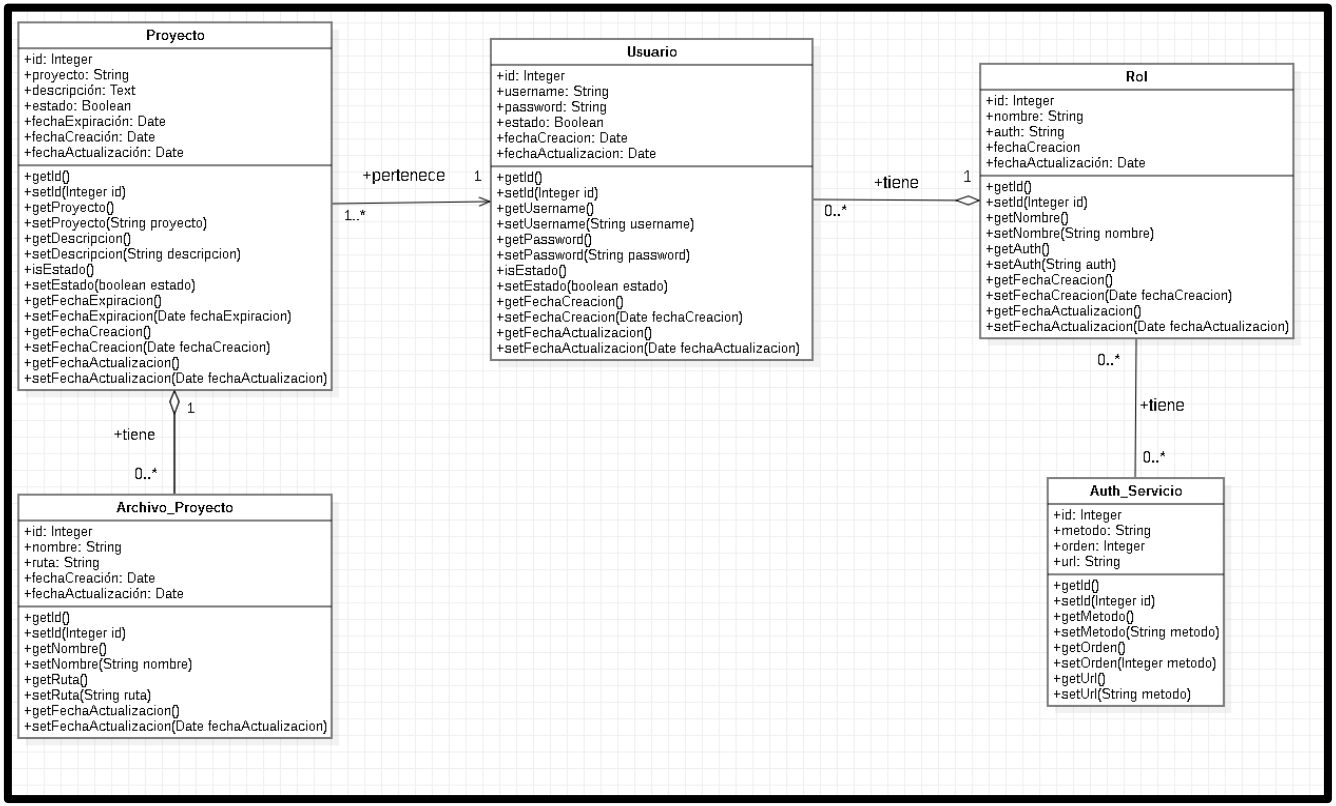


Fuente: Elaboración propia

4.5. Diseño Detallado

4.5.1. Diagrama de Clases detallado

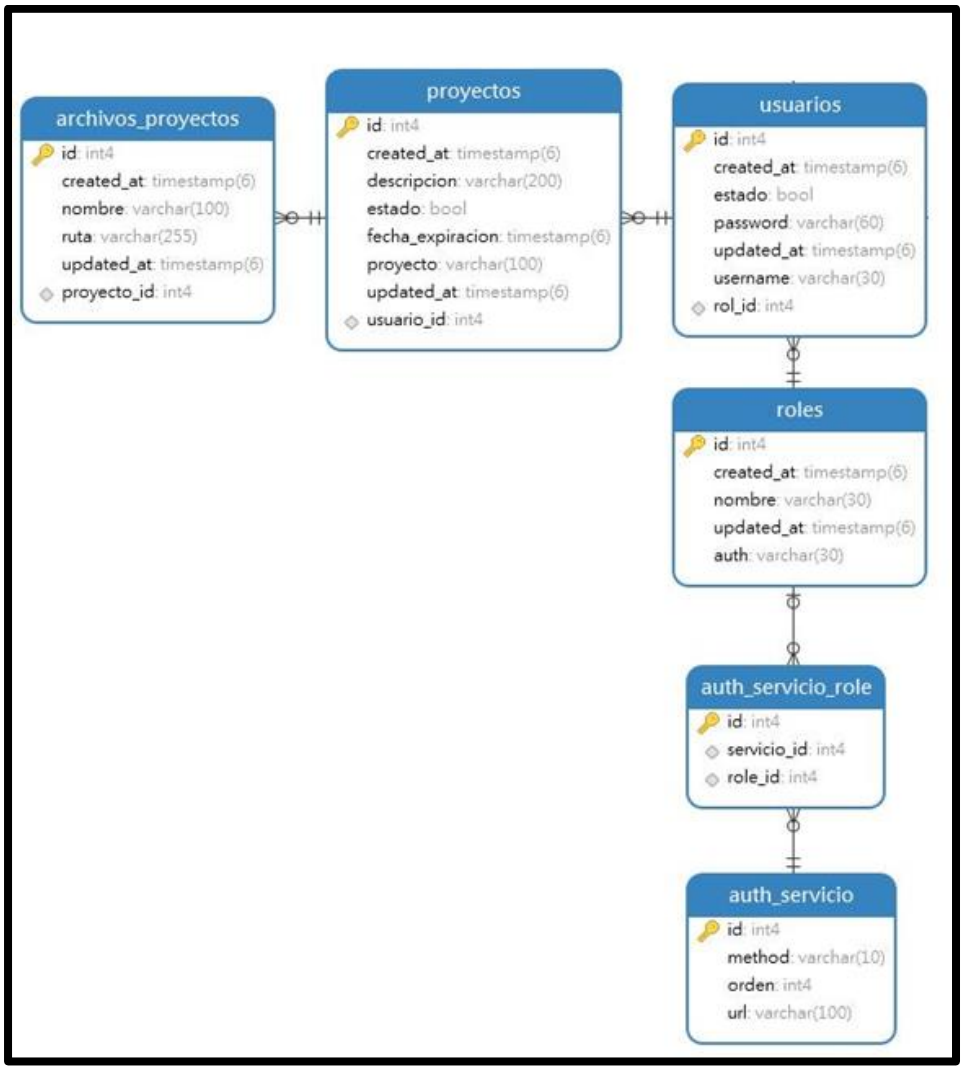
Figura 20: Diagrama de Clases UML



Fuente: Elaboración propia

4.5.2. Diagrama de Base de Datos de las Aplicaciones Web

Figura 21: Diagrama de Base de Datos del Sistema



Fuente: Elaboración propia

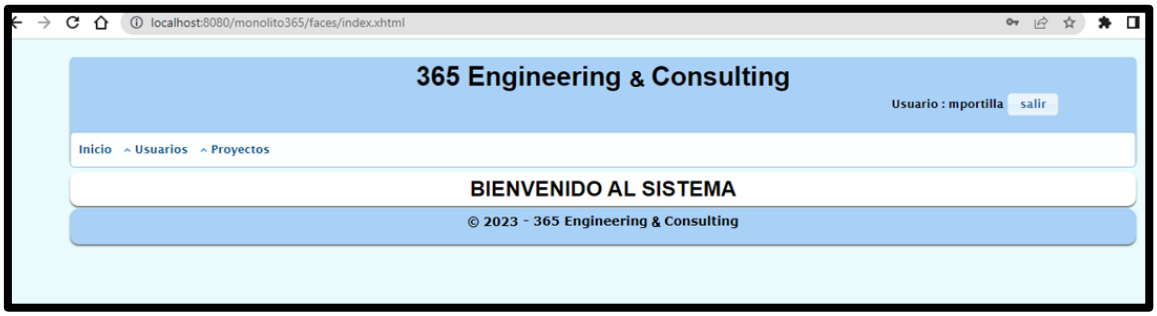
4.5.3. Interfaces Finales

4.5.3.1. Arquitectura Multicapa:

Figura 22: Pantalla de Inicio de Sesión

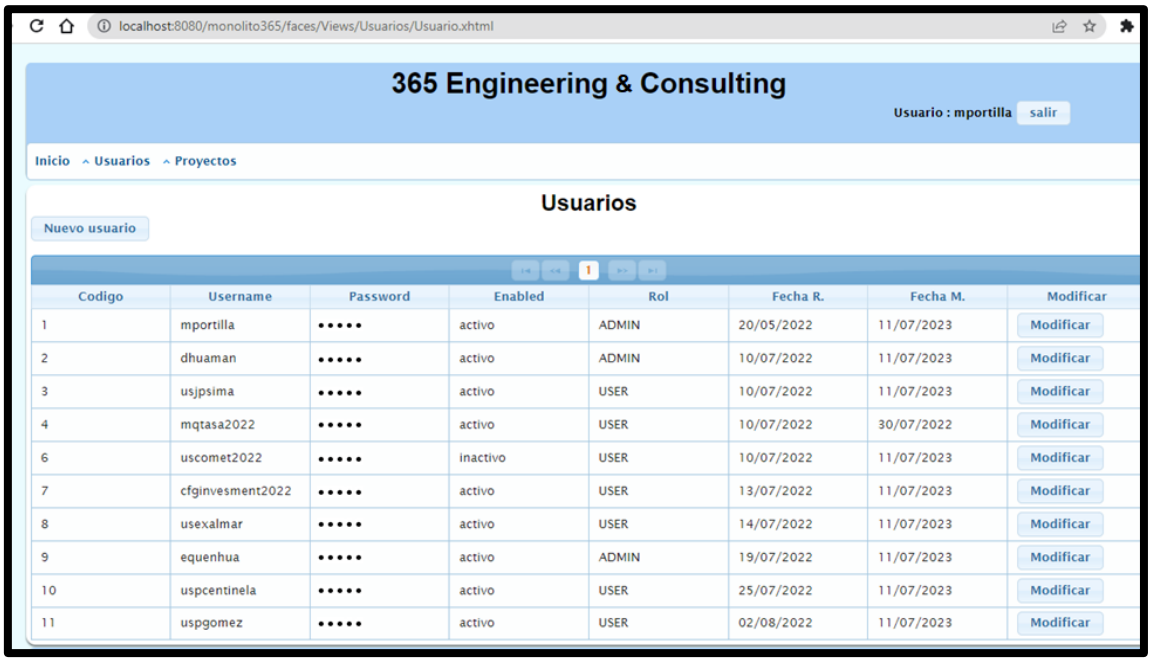
Fuente: Elaboración propia

Figura 23: Interfaz de Bienvenida al Sistema



Fuente: Elaboración propia

Figura 24: Interfaz del listado de usuarios del sistema en la web

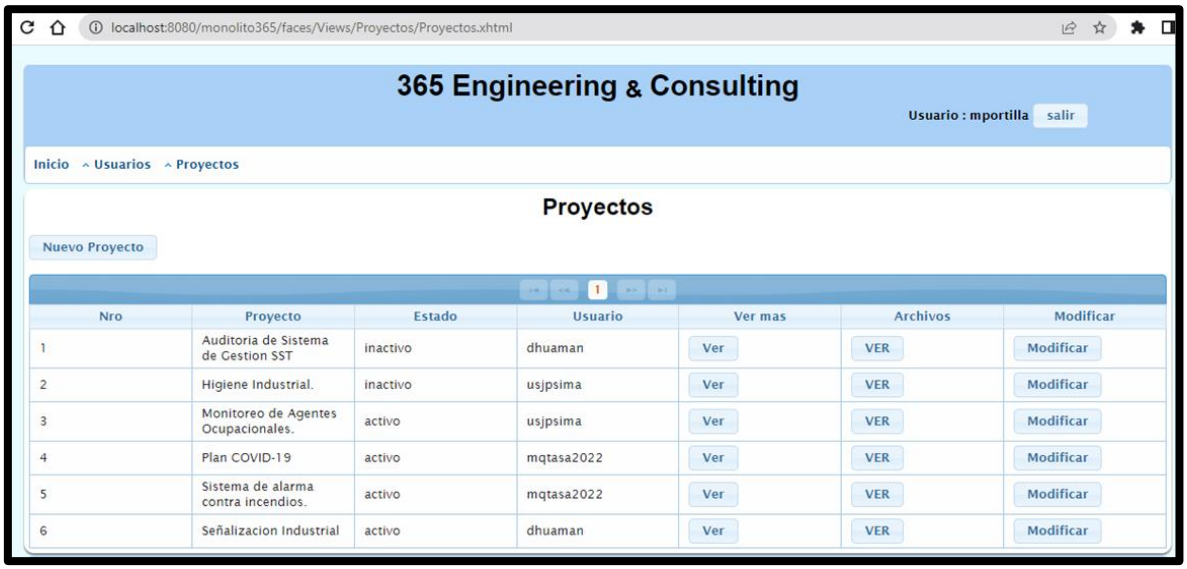


Fuente: Elaboración propia

Figura 25: Interfaz de registro del sistema en la web

Fuente: Elaboración propia

Figura 26: Interfaz del listado de proyectos de la empresa



Fuente: Elaboración propia

Figura 27: Interfaz del listado de archivos de los proyectos de la empresa

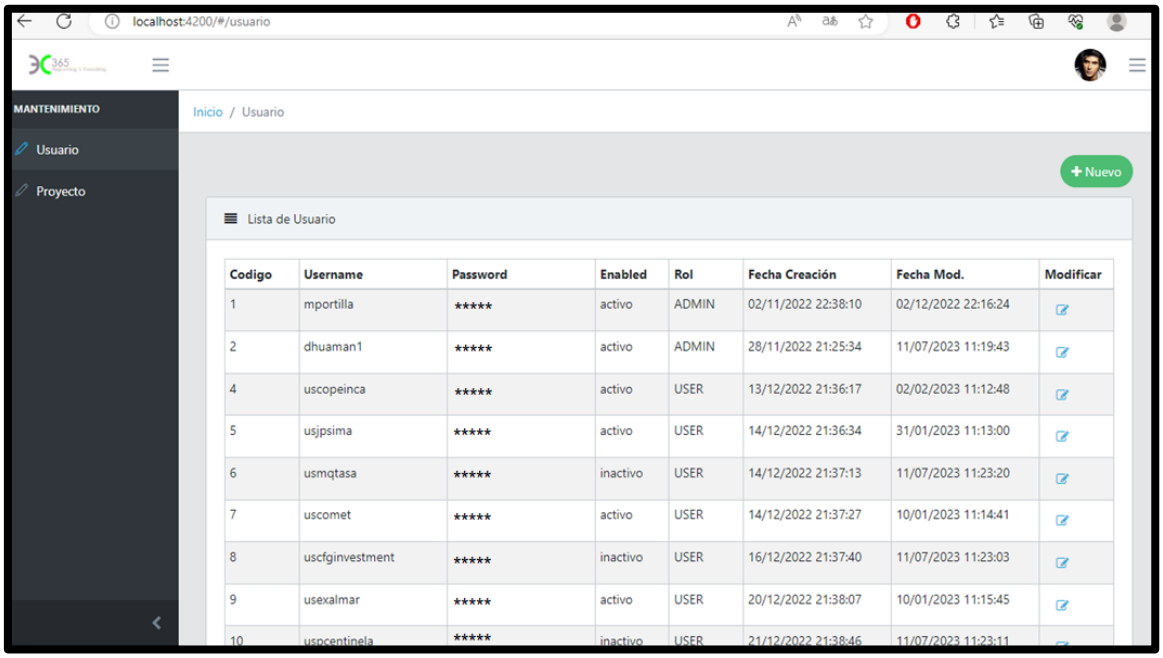
Fuente: Elaboración propia

4.5.3.2. Arquitectura SOA y de Microservicios

Figura 28: Pantalla de Inicio de Sesión

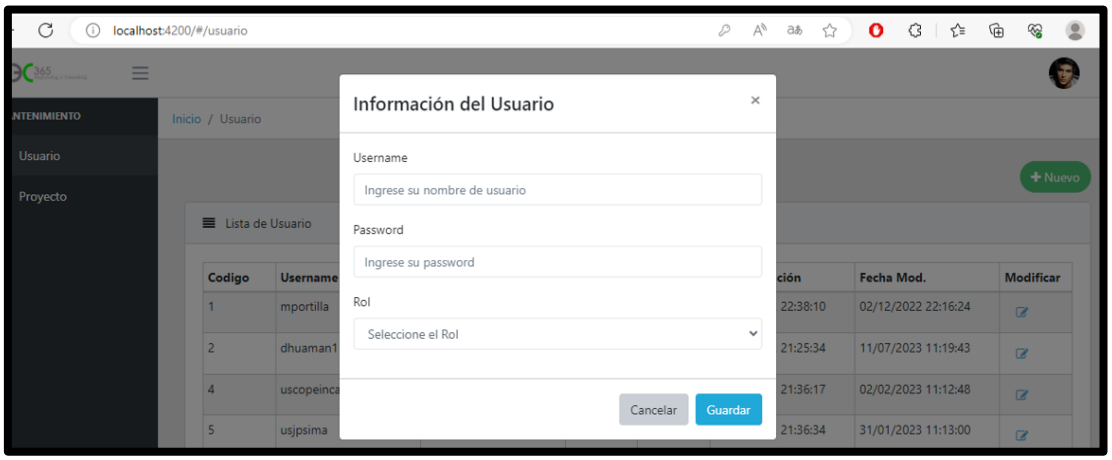
Fuente: Elaboración propia

Figura 29: Interfaz del listado de usuarios del sistema



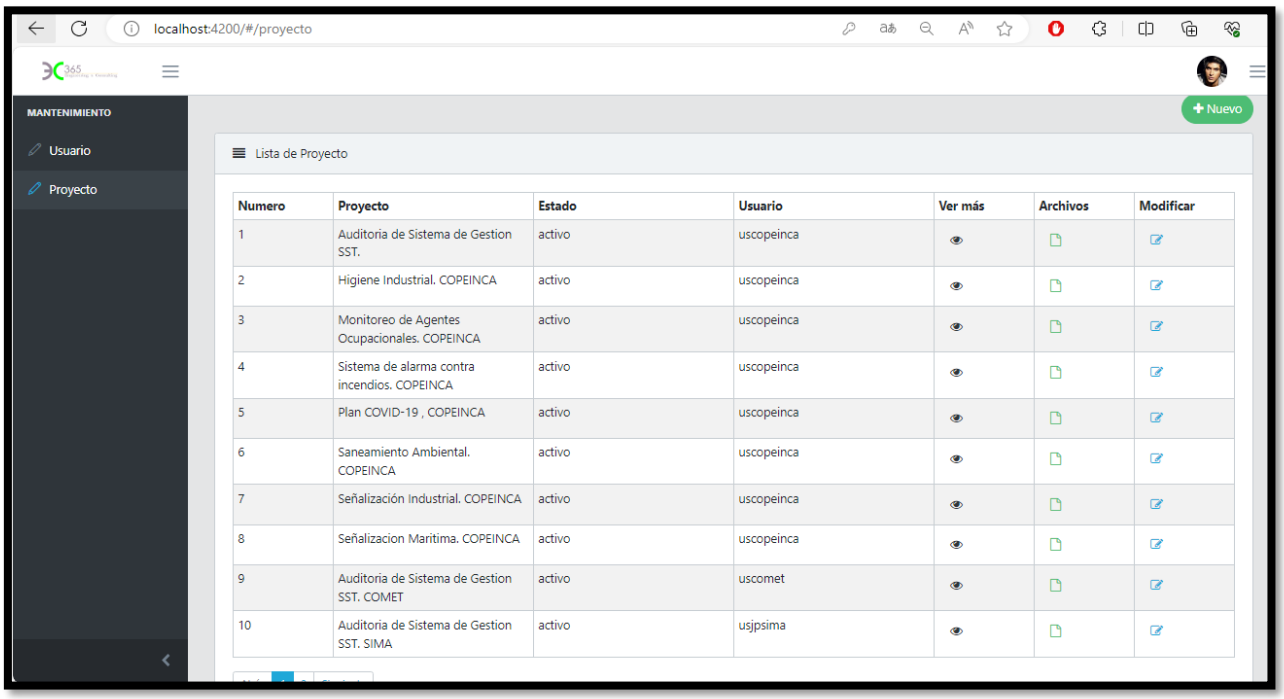
Fuente: Elaboración propia

Figura 30: Interfaz del registro de usuarios del sistema



Fuente: Elaboración propia

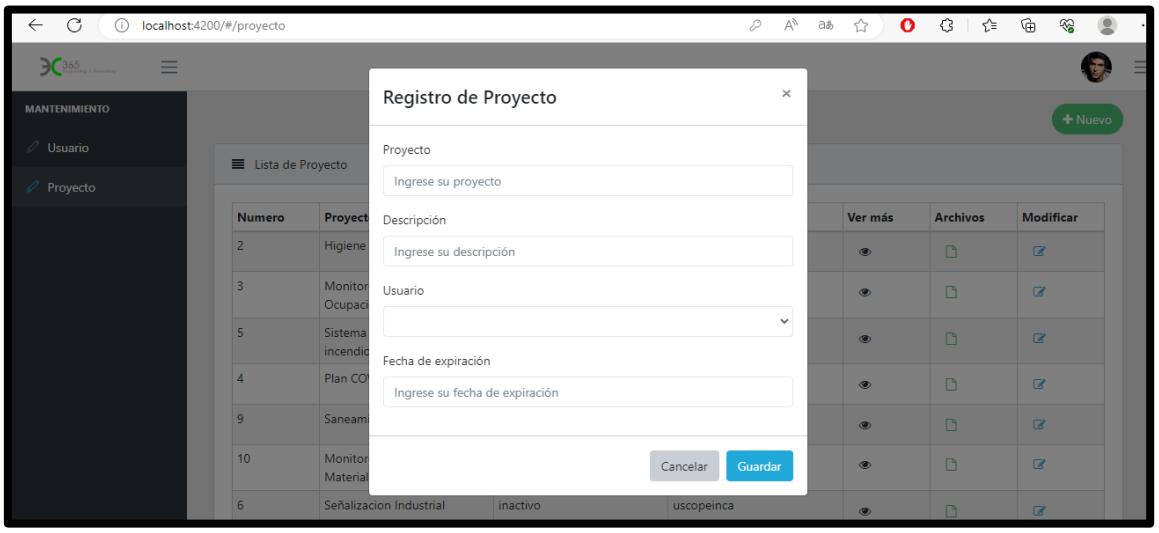
Figura 31: Interfaz del listado de proyectos de la empresa



Numero	Proyecto	Estado	Usuario	Ver más	Archivos	Modificar
1	Auditoria de Sistema de Gestion SST.	activo	uscopeinca			
2	Higiene Industrial. COPEINCA	activo	uscopeinca			
3	Monitoreo de Agentes Ocupacionales. COPEINCA	activo	uscopeinca			
4	Sistema de alarma contra incendios. COPEINCA	activo	uscopeinca			
5	Plan COVID-19 , COPEINCA	activo	uscopeinca			
6	Saneamiento Ambiental. COPEINCA	activo	uscopeinca			
7	Señalización Industrial. COPEINCA	activo	uscopeinca			
8	Señalizacion Maritima. COPEINCA	activo	uscopeinca			
9	Auditoria de Sistema de Gestion SST. COMET	activo	uscomet			
10	Auditoria de Sistema de Gestion SST. SIMA	activo	usjpsima			

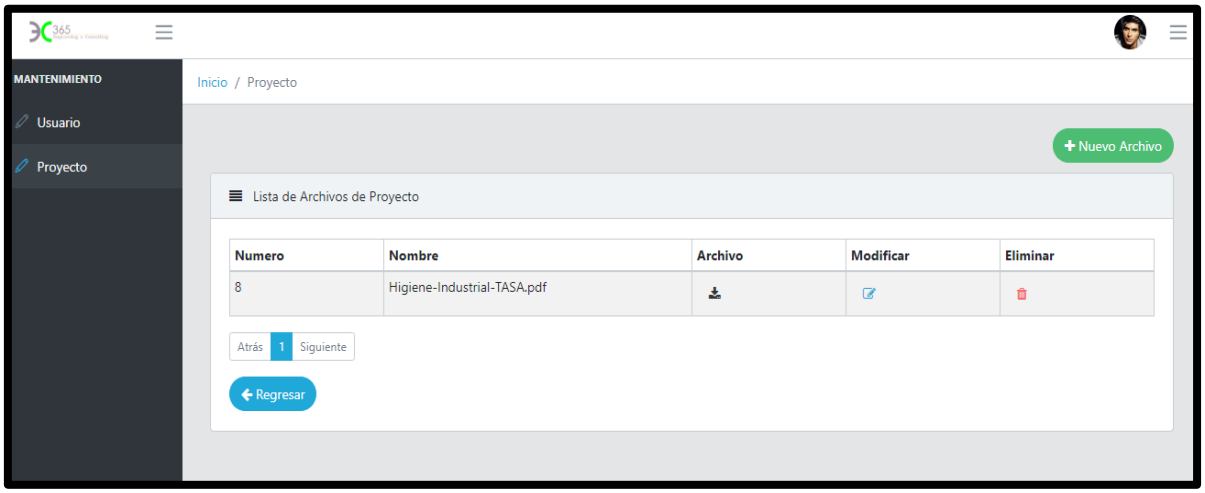
Fuente: Elaboración propia

Figura 32: Interfaz de registro de proyectos de la empresa



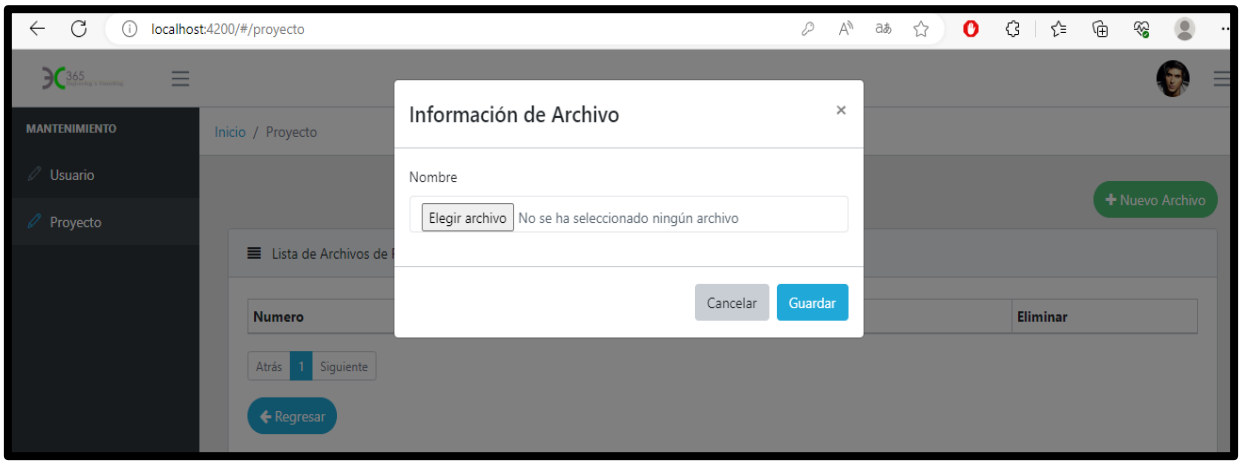
Fuente: Elaboración propia

Figura 33: Interfaz del listado de archivos de los proyectos de la empresa



Fuente: Elaboración propia

Figura 34: Interfaz de la carga de archivos de proyectos



Fuente: Elaboración propia

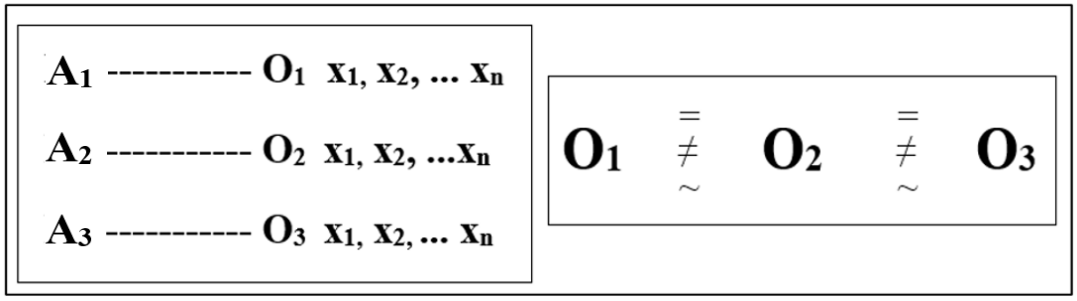
CAPITULO V: MATERIALES Y MÉTODOS

5.1. Diseño de la Investigación

El diseño del estudio fue no experimental, ya que se realizó un análisis comparativo de las tres Arquitecturas: Multicapa, Orientada a Servicios y Microservicios, que permitió medir su tiempo de respuesta, disponibilidad y tasa de error.

Según Radhakrishnan (2013) menciona que: "el diseño de investigación no experimental es una de las categorías de diseños de investigación, en la que el investigador observa los fenómenos a medida que ocurren de forma natural y no se introducen variables externas y que es un diseño de investigación en el que las variables no se manipulan deliberadamente ni se controla el ajuste" (Radhakrishnan, 2013, pág. 25)

Figura 35: Diseño de la Investigación



Fuente: Elaboración propia

Dónde:

A_1 = Arquitectura Multicapa O_1 = observación 1

A_2 = Arquitectura Orientada a Servicios O_2 = observación 2

A_3 = Arquitectura de Microservicios O_3 = observación 3

La A1, A2 y A3, representan las arquitecturas de software que utilizaremos para comparar, la O significa las observaciones o mediciones realizadas en base a los indicadores **(X_1, X_2, … X_n)** que se tendrá en cuenta, para realizar las pruebas de carga de las aplicaciones según las Arquitecturas de software. Esto nos permitirá

saber cuál será la Arquitectura más óptima para el desarrollo de aplicaciones web, como caso práctico para la empresa 365 ENGINEERING & CONSULTING.

Las observaciones **O_1, O_2 y O_3**; de la parte derecha del diagrama, nos indica la comparación que se lleva a cabo entre las arquitecturas, las cuales pueden ser: iguales (=), diferentes (≠) o semejantes (~) con respecto a la otra.

5.2. Población y Muestra

5.2.1. Población

La población está conformada por el número de peticiones HTTP realizadas a las aplicaciones web, como caso práctico a la Empresa 365 ENGINEERING & CONSULTING, lo cual se considera que la población es infinita.

5.2.2. Muestra

En nuestro estudio, el procedimiento para la determinación de la muestra será de tipo No Probabilística, donde se utilizó el muestreo por conveniencia, esto para crear muestras de acuerdo a la facilidad de acceso y para su uso. Se tendrá en cuenta como muestra 20 grupos de peticiones concurrentes (50, 100, 150, 200, 250, 300, 350, 450, 500, 550, 600, 650, 700, 750, 800, 850, 900, 950, 1000), para realizar las pruebas de carga a las aplicaciones web. Dichas pruebas se realizaron con la herramienta Fiddler Classic**.**

5.3. Técnicas e Instrumentos de recolección de datos

5.3.1. Técnicas de Recolección de Datos

Las técnicas de recolección de datos son un conjunto de actividades y procedimientos que permiten al investigador obtener los datos necesarios para llevar a cabo la investigación. En este proyecto de tesis se emplearán las siguientes técnicas:

5.3.1.1. Investigación Bibliográfica

Se recurrió a fuentes primarias y secundarias de información (Páginas web, artículos, tesis y otros) que permitieron orientar el desarrollo de la presente investigación y enriquecer el marco teórico.

5.3.1.2. Pruebas de carga

Esto permite verificar del funcionamiento de las aplicaciones bajo una gran cantidad de usuarios simultáneos o peticiones concurrentes, ya sea cuando se navega por el sitio web o se realizan transacciones durante un cierto período de tiempo.

CAPITULO VI: COMPROBACIÓN DE LA HIPÓTESIS Y RESULTADOS

El objetivo en este capítulo es realizar las pruebas de carga a las aplicaciones web desarrolladas y desplegadas en una red local, para después obtener los resultados estadísticos. Con dichas pruebas se conseguirán los datos de acuerdo al tiempo de respuesta, disponibilidad y tasa de error.

Se utilizó la herramienta Fiddler Classic para las pruebas a las aplicaciones web, donde el programa realiza la captura del tráfico web (sea HTTP o HTTPS) según la cantidad de peticiones concurrentes, dichas pruebas de carga se muestran en los anexos 7, 8 y 9. Se consideró una muestra de 20 grupos de peticiones concurrentes, los cuales son: 50, 100, 150, 200, 250, 300, 350, 450, 500, 550, 600, 650, 700, 750, 800, 850, 900, 950 y 1000. Además, teniendo en cuenta el escenario de las pruebas, el caso de Consultar el listado de proyectos dentro de la Aplicación web. Las ejecuciones de las aplicaciones se observan en los anexos 1,2,3 y 4.

A continuación, se presentan los resultados de las pruebas realizadas en esta investigación:

6.1. Prueba de Hipótesis para los Indicadores Cuantitativos

6.1.1. Tiempo de Respuesta

En la Tabla 11 se muestra los resultados de las pruebas de carga, según el indicador de Tiempo de Respuesta expresado en segundos, en el cual se considera los tiempos promedio por cada grupo de peticiones concurrentes, por cada Arquitectura de software, con el fin de encontrar la arquitectura que tenga el menor tiempo promedio de respuesta.

Tabla 11: *Tiempos de Respuesta según el tipo de Arquitectura de Software*

TIEMPOS DE RESPUESTAS (segundos)			
GRUPO DE PETICIONES CONCURRENTES	**Multicapa**	**SOA**	**MICROSERVICIOS**
50	0.257	1.255	0.314
100	0.182	1.157	0.145
150	1.580	1.143	0.266
200	0.829	1.037	0.140
250	0.779	1.171	0.257
300	1.324	1.934	0.171
350	0.367	1.284	0.125
400	0.268	1.818	0.219
450	0.327	1.669	0.166
500	2.038	1.478	0.137
550	0.414	11.882	0.145
600	0.286	2.154	0.149
650	0.317	3.123	0.140
700	0.281	7.267	0.147
750	0.203	2.312	0.163
800	0.251	1.716	0.162
850	0.170	7.686	0.156
900	0.220	2.185	0.164
950	0.225	2.040	0.155
1000	2.790	1.820	0.151

Fuente: Elaboración propia

a) Estadísticos descriptivos de la muestra

En la Tabla 12 se muestran los estadísticos descriptivos, tales como: La Media, Mediana, Desviación Estándar, Varianza, el Mínimo y Máximo Tiempo de respuesta por cada una de las Arquitecturas de Software.

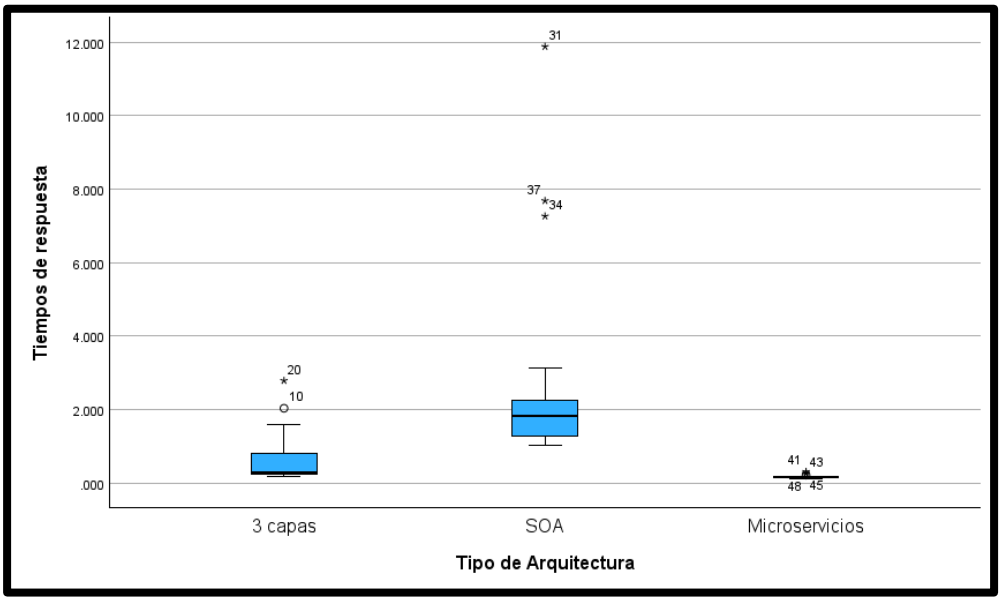
Tabla 12: *Estadísticos descriptivos de las Arquitecturas según el Tiempo de respuesta*

	Arquitecturas	N	Media	Mediana	DE	Mínimo	Máximo
Tiempo de respuesta	Multicapa	20	0.655	0.301	0.7243	0.170	2.790
	Microservicios	20	0.174	0.155	0.0501	0.125	0.314
	SOA	20	2.807	1.819	2.8172	1.037	11.882

Fuente: Elaboración propia

En la Figura 36 se observa que los tiempos de respuestas de las Arquitecturas Multicapa y SOA son superiores a los tiempos de la Arquitectura de Microservicios.

Figura 36: Diagrama de Cajas del Tiempo de respuesta de las Arquitecturas de Software



Fuente: Elaboración propia

b) Análisis de normalidad del conjunto de datos

Se utilizó el programa estadístico SPSS para validar el supuesto de normalidad. Dado que la muestra es menor a 30, se usará la prueba Shapiro-Wilk para validar la normalidad de estos datos.

Tabla 13*: Pruebas de Normalidad de Shapiro-Wilk*

	Tipo de Arquitectura	**Shapiro-Wilk**		
		Estadístico	**gl**	**Sig.**
Tiempos de respuesta	Multicapa	0.687	20	<.001
	SOA	0.613	20	<.001
	Microservicios	0.738	20	<.001

Fuente: Elaboración propia

Se observa en la Tabla 13, que el valor de significancia (Sig.) es menor que 0.05, por lo tanto, los datos no provienen de una distribución normal y procedemos a aplicar la prueba no paramétrica **Kruskal-Wallis.**

c) Hipótesis Estadísticas

Hipótesis H0: La Arquitectura de Microservicios, Multicapa y SOA son iguales en el tiempo de respuesta.

Hipótesis H_1: La Arquitecturas de software son diferentes significativamente en sus tiempos de respuestas.

d) Prueba de Kruskal-Wallis

- **Nivel de significancia**

  En la prueba de Kruskal-Wallis con 2 grados de libertad, se utilizará un nivel de significación estadística del 5%, por lo que el nivel de confianza ($1-\alpha = 0,95$) será del 95%.

Tabla 14*: Resultado de la significancia con la prueba Kruskal-Wallis.*

Hipótesis nula	Prueba	Sig.	Decisión
La distribución de Tiempos de respuesta es la misma entre categorías de Tipo de Arquitectura.	Prueba de Kruskal-Wallis para muestras independientes	<.001	Rechace la hipótesis nula.

Fuente: Elaboración propia

Se observa en la tabla 14; el valor de significancia resulta ser menor a 0.05, entonces se rechaza el H0 y por consiguiente se acepta el H1. Esto quiere decir que existe diferencias significativas entre las Arquitecturas de Software, de acuerdo al indicador tiempo de respuesta, esto con un nivel de error del 5% y un nivel de confianza del 95%.

e) Conclusión

Dado la prueba estadística de Kruskal-Wallis y el grafico estadístico (en la figura 36), se concluye entonces que la Arquitectura de Microservicios presenta los tiempos de respuesta más bajos que el resto de arquitecturas, y la Arquitectura SOA presenta los tiempos de respuesta mayores a la Arquitectura Multicapa y de Microservicios.

6.1.2. Disponibilidad

Se presenta el reporte del indicador cuantitativo de Disponibilidad, que consiste en la capacidad de la aplicación web para brindar un servicio completo 24/7 a los usuarios de los sistemas. Esto fue calculado al contar el número de peticiones con status HTTP de tipo 200.

En la Tabla 16, se muestra los datos promedios de la disponibilidad por cada Arquitectura de software, estos están expresados en porcentajes.

Tabla 15*: Distribución de la Disponibilidad según el tipo de Arquitectura de Software*

Disponibilidad (porcentaje)			
GRUPO DE PETICIONES CONCURRENTES	**Multicapa**	**SOA**	**MICROSERVICIOS**
50	100.00%	100.00%	100.00%
100	99.00%	100.00%	100.00%
150	100.00%	100.00%	100.00%
200	99.00%	100.00%	100.00%
250	99.60%	100.00%	100.00%
300	99.33%	100.00%	100.00%
350	99.43%	100.00%	100.00%
400	99.50%	100.00%	100.00%
450	99.78%	100.00%	100.00%
500	99.60%	100.00%	100.00%
550	99.64%	100.00%	100.00%
600	100.00%	100.00%	100.00%
650	99.08%	100.00%	100.00%
700	99.29%	100.00%	100.00%
750	99.33%	100.00%	100.00%
800	99.50%	100.00%	100.00%
850	99.53%	100.00%	100.00%
900	99.67%	100.00%	100.00%
950	99.37%	100.00%	100.00%
1000	99.30%	100.00%	100.00%

Fuente: Elaboración propia

a) Estadísticos descriptivos de la muestra

En la Tabla 16 se muestran los estadísticos descriptivos, tales como: La Media, Mediana, Desviación Estándar, Varianza, el Mínimo y Máximo de la Disponibilidad por cada una de las Arquitecturas de Software.

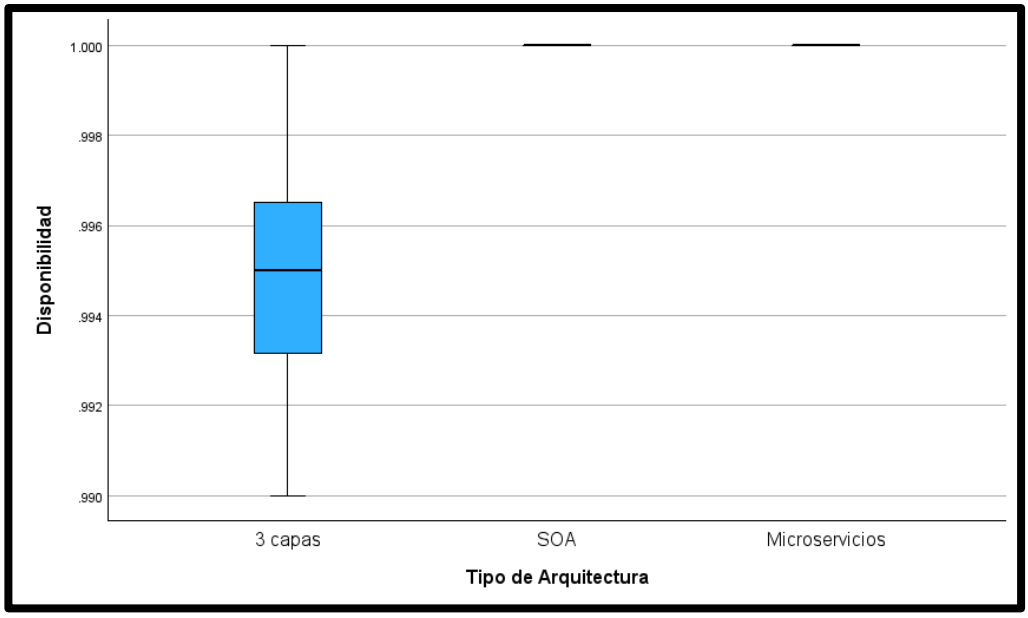
Tabla 16*: Estadísticos descriptivos de las Arquitecturas según la Disponibilidad*

	Arquitecturas	N	Media	Mediana	DE	Mínimo	Máximo
disponibilidad	Multicapa	20	0.995	0.995	0.00302	0.990	1.00
	Microservicios	20	1.000	1.000	0.00000	1.000	1.00
	SOA	20	1.000	1.000	0.00000	1.000	1.00

Fuente: Elaboración propia

Si se observa la Figura 37**,** la disponibilidad de las Arquitectura de Microservicios y SOA son superiores a la Arquitectura Multicapa.

Figura 37: Diagrama de Cajas de la Disponibilidad de las Arquitecturas de Software



Fuente: Elaboración propia

b) Análisis de normalidad del conjunto de datos

Se utilizará el programa estadístico SPSS para validar el supuesto de normalidad. Dado que la muestra es menor a 30, se usará la prueba Shapiro-Wilk para validar la normalidad de estos datos.

Tabla 17: Pruebas de Normalidad de Shapiro-Wilk

	Tipo de Arquitectura	Shapiro-Wilk		
		Estadístico	gl	Sig.
Disponibilidad	Multicapa	0.949	20	0.348
	SOA	-	20	-
	Microservicios	-	20	-

Fuente: Elaboración propia

Se observa en la Tabla 17, que el valor de significancia (Sig.) de la Arquitectura Multicapa es mayor que 0.05, pero en las Arquitecturas SOA y Microservicios no presenta variabilidad, porque son constantes. En general los datos no provienen de una distribución normal y procedemos a aplicar la prueba no paramétrica Kruskal-Wallis.

c) Hipótesis Estadísticas

Hipótesis H_0: La Arquitectura de Microservicios, Multicapa y SOA son iguales en la disponibilidad.

Hipótesis H_1: La Arquitecturas de software son diferentes significativamente en su disponibilidad.

d) Prueba Kruskal-Wallis

- **Nivel de significancia**

  En la prueba de Kruskal-Wallis con 2 grados de libertad, se utilizará un nivel de significación estadística del 5%, por lo que el nivel de confianza ($1-\alpha = 0,95$) será del 95%.

Tabla 18: *Resultado de la significancia con la prueba Kruskal-Wallis.*

Hipótesis nula	Prueba	Sig.	Decisión
La distribución de Disponibilidad es la misma entre categorías de Tipo de Arquitectura.	Prueba de Kruskal-Wallis para muestras independientes	<.001	Rechace la hipótesis nula.

Fuente: Elaboración propia

Se observa en la tabla 18; el valor de significancia resulta ser menor a 0.05, entonces se rechaza el H0 y por consiguiente se acepta el H1. Esto quiere decir que existe diferencias significativas entre las Arquitecturas de Software, de acuerdo al indicador disponibilidad, esto con un nivel de error del 5% y un nivel de confianza del 95%.

e) Conclusión

De acuerdo a la prueba estadística Kruskal-Wallis y al Diagrama de cajas (en la figura 37), se concluye que la Arquitectura SOA y de Microservicios, resultaron ser mejores en cuanto al indicador Disponibilidad; ya que son constantes al 100%, y la Arquitectura SOA presenta una disponibilidad mayor a las otras Arquitecturas.

6.1.3. Tasa de Error

Se presenta el reporte del indicador cuantitativo de Tasa de Error, que ha sido calculado al contar el número de peticiones fallidas o cuyo status HTTP es distinto al código 200, como por ejemplo el código 500, 404 y otros códigos relacionados.

En la Tabla 19, se muestra la comparativa de las arquitecturas de software en materia de la Tasa de error expresados en porcentajes, esta información fue obtenida a través del programa Fiddler Classic.

Tabla 19*: Distribución de la Tasa de error según el tipo de Arquitectura de Software*

	Tasa de Error (porcentaje)		
GRUPO DE PETICIONES CONCURRENTES	**Multicapa**	**SOA**	**MICROSERVICIOS**
50	0.00%	0.00%	0.00%
100	1.00%	0.00%	0.00%
150	0.00%	0.00%	0.00%
200	1.00%	0.00%	0.00%
250	0.40%	0.00%	0.00%
300	0.67%	0.00%	0.00%
350	0.57%	0.00%	0.00%
400	0.50%	0.00%	0.00%
450	0.22%	0.00%	0.00%
500	0.40%	0.00%	0.00%
550	0.36%	0.00%	0.00%
600	0.00%	0.00%	0.00%
650	0.92%	0.00%	0.00%
700	0.71%	0.00%	0.00%
750	0.67%	0.00%	0.00%
800	0.50%	0.00%	0.00%

850	0.47%	0.00%	0.00%
900	0.33%	0.00%	0.00%
950	0.63%	0.00%	0.00%
1000	0.70%	0.00%	0.00%

Fuente: Elaboración propia

a) Estadísticos descriptivos de la muestra

En la Tabla 20 se muestran los estadísticos descriptivos, tales como: La Media, Mediana, Desviación Estándar, Varianza, el Mínimo y Máximo de la Tasa de error por cada una de las Arquitecturas de Software.

Tabla 20*: Estadísticos descriptivos de las Arquitecturas según la Tasa de Error*

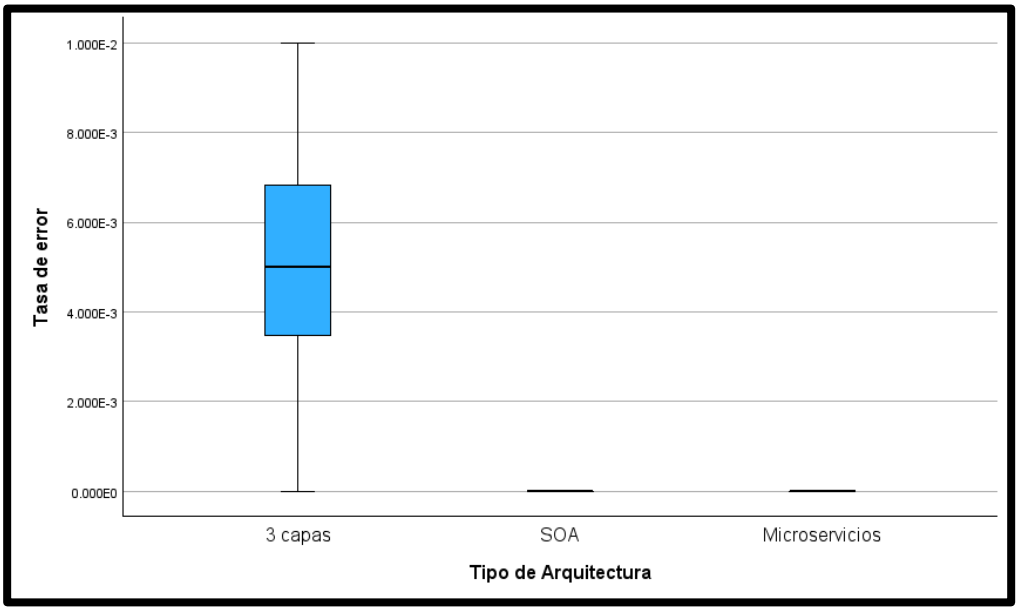
	Arquitecturas	N	Media	Mediana	DE	Mínimo	Máximo
Tasa de error	**Multicapa**	20	0.00503	0.00500	0.00302	0.00	0.0100
	Microservicios	20	0.00000	0.00000	0.00000	0.00	0.0000
	SOA	20	0.00000	0.00000	0.00000	0.00	0.0000

Fuente: Elaboración propia

En la Figura 38 se observa que la Tasa de error de la Arquitectura Multicapa es mayor que las Arquitecturas SOA y de Microservicios.

Figura 38:

Figura 38: Diagrama de Cajas de la Tasa de Error de las Arquitecturas de Software



Fuente: Elaboración propia

b) Análisis de normalidad del conjunto de datos

Se utilizará el programa estadístico SPSS para validar el supuesto de normalidad. Dado que la muestra es menor a 30, se usará la prueba Shapiro-Wilk para validar la normalidad de estos datos.

Tabla 21*: Pruebas de Normalidad de Shapiro-Wilk*

	Tipo de Arquitectura	Shapiro-Wilk		
		Estadístico	gl	Sig.
Tasa de error	Multicapa	0.949	20	0.348
	SOA	-	20	-
	Microservicios	-	20	-

Fuente: Elaboración propia

Se observa en la Tabla 21, que el valor de significancia (Sig.) de la Arquitectura Multicapa es mayor que 0.05, pero en las Arquitecturas SOA y Microservicios no presenta variabilidad, porque son constantes.

En general los datos no provienen de una distribución normal y procedemos a aplicar la prueba no paramétrica Kruskal-Wallis.

c) Hipótesis Estadísticas

Hipótesis H_0: La Arquitectura de Microservicios, Multicapa y SOA son iguales en la tasa de error.

Hipótesis H_1: La Arquitectura de Microservicios, Multicapa y SOA son diferentes significativamente en su tasa de error.

d) Prueba Kruskal-Wallis

- **Nivel de significancia**

  En la prueba de Kruskal-Wallis con 2 grados de libertad, se utilizará un nivel de significación estadística del 5%, por lo que el nivel de confianza ($1-\alpha = 0,95$) será del 95%.

Tabla 22: *Resultado de la significancia con la prueba Kruskal-Wallis.*

Hipótesis nula	Prueba	Sig.	Decisión
La distribución de Tasa de error es la misma entre categorías de Tipo de Arquitectura.	Prueba de Kruskal-Wallis para muestras independientes	<.001	Rechace la hipótesis nula.

Se observa en la tabla 22; el valor de significancia resulta ser menor a 0.05, entonces se rechaza el H0 y por consiguiente se acepta el H1. Esto quiere decir que existe diferencias significativas entre las Arquitecturas de Software, de acuerdo al indicador tasa de error, esto con un nivel de error del 5% y un nivel de confianza del 95%.

e) Conclusión

De acuerdo a la prueba estadística Kruskal-Wallis y al Diagrama de cajas (en la figura 38), se concluye entonces que la Arquitectura SOA y de Microservicios presentan las mejores tasas de error, y la Arquitectura Multicapa presenta la tasa de error mayores a la Arquitectura SOA y de Microservicios.

CAPITULO VII: CONCLUSIONES Y RECOMENDACIONES

7.1. Conclusiones

- Se cumplió el objetivo de hallar los promedios de los tiempos de respuesta por cada Arquitectura de software y grupo de peticiones concurrentes. Se realizaron las pruebas de rendimiento mediante el programa Progress Telerik Fiddler Classic, para obtener los tiempos de respuestas.
- Se cumplió el objetivo de determinar los porcentajes de disponibilidad por cada grupo de peticiones concurrentes y Arquitectura de software. A través del programa Progress Telerik Fiddler Classic se obtuvo la cantidad de peticiones con respuestas satisfactorias, cuyos códigos de respuesta oscilan entre 200 y 299.
- Se cumplió el objetivo de determinar la tasa de error por cada Arquitectura de software en estudio. A través del programa Progress Telerik Fiddler Classic se obtuvo la cantidad de peticiones con respuestas erróneas, cuyos códigos de respuesta oscilan entre 500 y 599.
- S realizó el análisis comparativo entre las Arquitecturas de software que cumple con el objetivo general en dar validez a la hipótesis planteada. En base a las pruebas estadísticas efectuadas, se puede concluir que la Arquitectura de Microservicios es la que presenta mejor rendimiento según los tiempos de respuesta, disponibilidad y tasa de error. Se debe tener en cuenta que las pruebas de rendimiento, se realizaron hasta un máximo de 1000 peticiones concurrentes.

7.2. Recomendaciones

- Se recomienda en realizar nuevas investigaciones con otras Arquitecturas y atributos de calidad del software, como por ejemplo: Modularidad, interoperabilidad, seguridad, y entre otros, donde al realizar el análisis comparativo se logre obtener diferentes resultados y demostrar cuales son las arquitecturas que tengan mejor impacto en el rendimiento de las aplicaciones web.
- Al utilizar la Arquitectura de Microservicios, se observó que presentan mejor rendimiento que las otras, ante un alto tráfico de consumo de las aplicaciones web, pero también vale mencionar, su gran complejidad para su gestión e integración, por ello se requiere contar con un equipo de desarrolladores experimentados. Por ello, se recomienda realizar un análisis exhaustivo, antes de elegir dicha arquitectura, si se adecua a los requerimientos funcionales, no funcionales, y a los recursos que presenta la Empresa u Organización, como los recursos económicos, humanos y tecnológicos.
- Se recomienda utilizar contenedores para las aplicaciones, de acuerdo a la Arquitectura de Microservicios, que contiene todo lo necesario para que el programa se ejecute, incluso las librerías y código fuente. Esto también ayudará a los desarrolladores a acelerar la implementación de aplicaciones a la nube.
- En este contexto de contenedores, se recomiendan dos herramientas fundamentales: Docker y Kubernetes. Docker es un sistema operativo para contenedores que automatiza su uso, funcionando como un ejecutable que incluye todo lo necesario para que una aplicación se ejecute correctamente. Por otro lado, Kubernetes actúa como un orquestador de contenedores,

gestionando su coordinación y se ha convertido prácticamente en un estándar en esta área.

- Actualmente en el mercado de cloud computing, está liderado por tres proveedores: Amazon Web Services(AWS), Azure y Google Cloud Platform(GCP). Es importante que los interesados conozcan y evalúen los beneficios y desventajas de cada proveedor, por ello se recomienda que la implementación de la arquitectura de microservicios sea en la nube, debido a que es una tendencia tecnológica en el mercado actual, que brindará grandes beneficios a largo plazo en la empresa interesada.
- Si el nivel de los recursos es relativamente bajo, si el sistema no tiene altas tasas de consumo o peticiones concurrentes por segundo, se recomienda optar por una Arquitectura Monolítica o Tradicional, con el uso adecuado de patrones arquitectónicos y de diseño, que pueden aportar soluciones a diversos problemas en distintos ámbitos.

REFERENCIAS BIBLIOGRAFICAS

ANGULAR. (2023). Obtenido de https://docs.angular.lat/docs

AWS. (2023). *amazon*. Obtenido de amazon: https://aws.amazon.com/es/what-is/web-application/

Barahona, C. (2023). *Propuesta de implementación de microservicios usando metodología ágil para la optimización de procesos en Banhcafe.* Obtenido de Repositorio de la Universidad Tecnologica Centroamericana: https://repositorio.unitec.edu/xmlui/handle/123456789/12487

Blass, L., Clements, P., & Kazman, R. (2015). *Software Architecture in Practice Third Edition.* Estados Unidos: Addison-Wesley Professional; 3er edición (25 Septiembre 2012).

Codecentric. (2023). *Spring Boot Admin Docs*. Obtenido de Spring Boot Admin Docs: http://docs.spring-boot-admin.com/current/getting-started.html

Fowler, M. (25 de Marzo de 2014). *Microservices*. Obtenido de Microservices: https://martinfowler.com/articles/microservices.html

Hat, R. (2023). *El concepto de las interfaces de programación de aplicaciones* . Obtenido de https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

Huerta, A. (2020). *Transición del monolito a los microservicios: cambio de paradigma en el desarrollo de aplicaciones.* Obtenido de Repositorio Insitucional de la Universidad Pontificia Comillas: http://hdl.handle.net/11531/43412

IBM. (2021). *Pruebas de rendimiento*. Obtenido de https://www.ibm.com/docs/es/rtw/9.0.0?topic=phases-performance-testing

IBM. (2023). *¿Qué es la prueba de software?* Obtenido de https://www.ibm.com/es-es/topics/software-testing

Microsoft. (2023). *microsoft*. Obtenido de microsoft: https://learn.microsoft.com/es-es/dotnet/architecture/modern-web-apps-azure/modern-web-applications-characteristics

Nishimura , C., & Ramirez, R. (2022). *Implementación de una arquitectura de microservicios con plataforma de integración y automatización de flujos de trabajo para la gestión de servicios en la empresa automotriz Honda Perú SAC – Lima 2022.* Obtenido de Repositorio Institucional de la Universidad Tecnológica del Perú : https://hdl.handle.net/20.500.12867/6663

ORACLE. (2013). *TThe Java EE 6 Tutorial*. Obtenido de The Java EE 6 Tutorial: https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html

Oracle. (2023). *1.3 Aplicaciones multinivel distribuidas*. Obtenido de Oracle.com: https://docs.oracle.com/javaee/7/tutorial/overview003.htm

ORACLE. (2023). *Developing Web Applications with JavaServer Faces*. Obtenido de Developing Web Applications with JavaServer Faces: https://www.oracle.com/technical-resources/articles/javase/javaserverfaces.html

Pacheco, J. (2018). *Estudio Comparativo entre una Arquitectura con Microservicios y Contenedores Dockers y una Arquitectura Tradicional(Monolítica) con Comprobación Aplicativa.* Obtenido de Repositorio Insitucional de la Universidad de Guayaquil : http://repositorio.ug.edu.ec/handle/redug/32755

Progress. (2023). *Progress Telerik.* Obtenido de Progress Telerik: https://www.telerik.com/blogs/understanding-telerik-fiddler-as-a-proxy

Radhakrishnan, D. G. (2013). *anvpublication.* Obtenido de Www.anvpublication.org

Red Hat, I. (2023). *Red Hat*. Obtenido de https://www.redhat.com/es/topics/cloud-native-apps/what-is-service-oriented-architecture

Ruelas , D. (2017). *MODELO DE COMPOSICIÓN DE MICROSERVICIOS PARA LA IMPLEMENTACIÓN DE UNA APLICACIÓN WEB DE COMERCIO ELECTRÓNICO UTILIZANDO KUBERNETES.* Obtenido de Revista De Investigaciones , 7(3).: https://doi.org/10.26788/riepg.v7i3.371

Scott, E. (2015). *SPA Design and Architecture.* Estados Unidos: Manning Publications; N.º 1 edición (31 octubre 2015).

Villaizán, H. (2019). *Arquitectura de software basada en microservicios para implementación de la aplicación web de cobranza digital en Financial Systems Company SAC.* Obtenido de Repositorio Institucional Continental: https://hdl.handle.net/20.500.12394/6387

VMWARE. (2023). *Circuit Breaker: Hystrix Clients*. Obtenido de Circuit Breaker: Hystrix Clients: https://cloud.spring.io/spring-cloud-netflix/multi/multi__circuit_breaker_hystrix_clients.html

VMWARE. (2023). *SPRING*. Obtenido de SPRING: https://spring.io/projects/spring-cloud-netflix

VMWARE. (2023). *SPRING*. Obtenido de SPRING: https://cloud.spring.io/spring-cloud-config/multi/multi__spring_cloud_config_server.html

VMWARE. (2023). *SPRING*. Obtenido de SPRING: https://cloud.spring.io/spring-cloud-netflix/reference/html/

VMWARE. (2023). *SPRING*. Obtenido de SPRING: https://cloud.spring.io/spring-cloud-netflix/multi/multi__router_and_filter_zuul.html

VMware. (2023). *Spring Boot*. Obtenido de https://spring.io/projects/spring-boot

VMWARE. (2023). *Spring Cloud*. Obtenido de Spring Cloud: https://spring.io/projects/spring-cloud

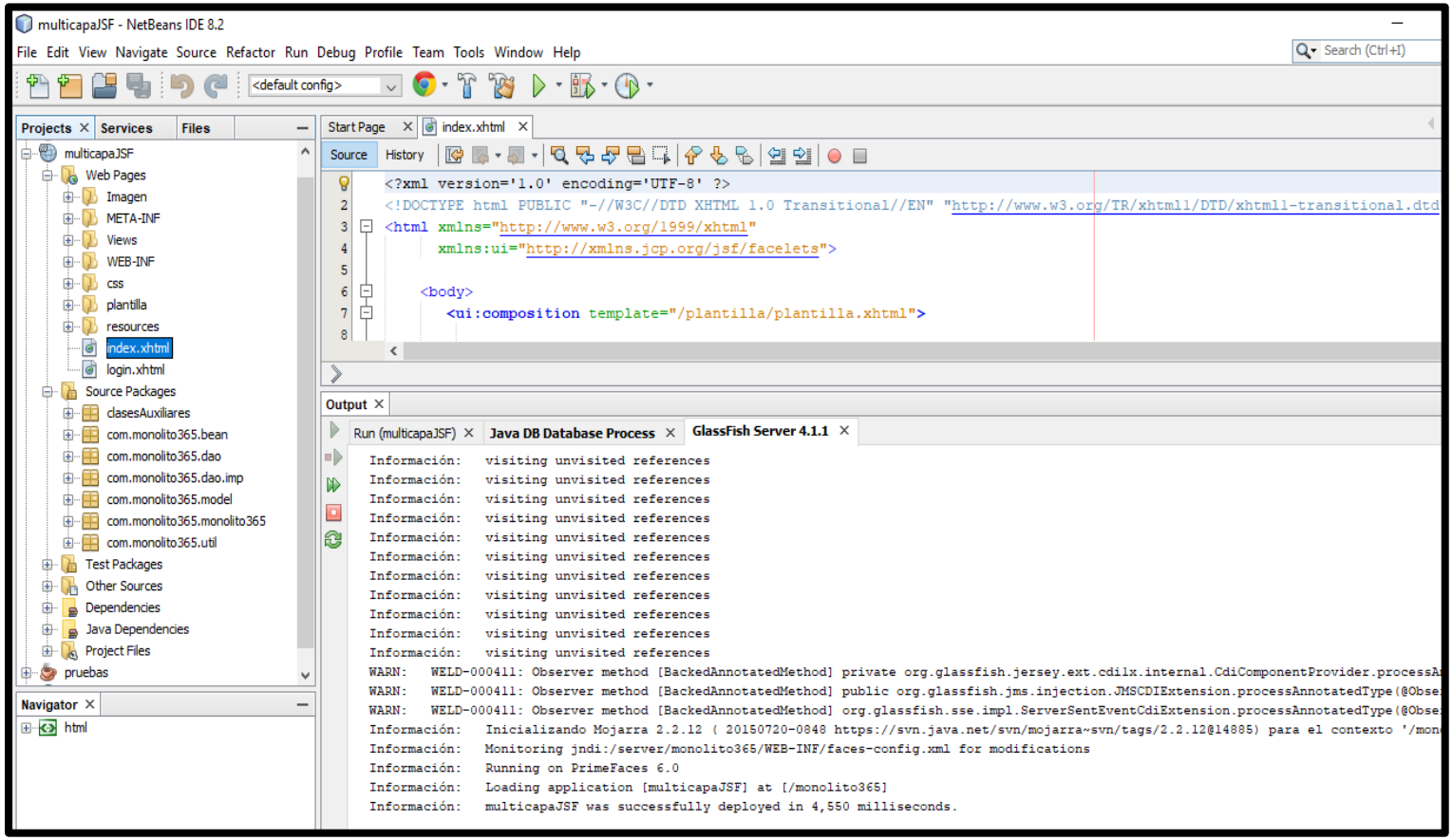
VMWARE. (2023). *Spring Framework*. Obtenido de Spring Framework6.0.11: https://spring.io/projects/spring-framework

ANEXOS

ANEXO 1

Ejecución de la aplicación web, con la Arquitectura Multicapa: Desarrollado con el framework Java server Faces 2.0

Figura 39: Ejecución de la aplicación web con JSF con el servidor Glassfish, en el editor NetBeans

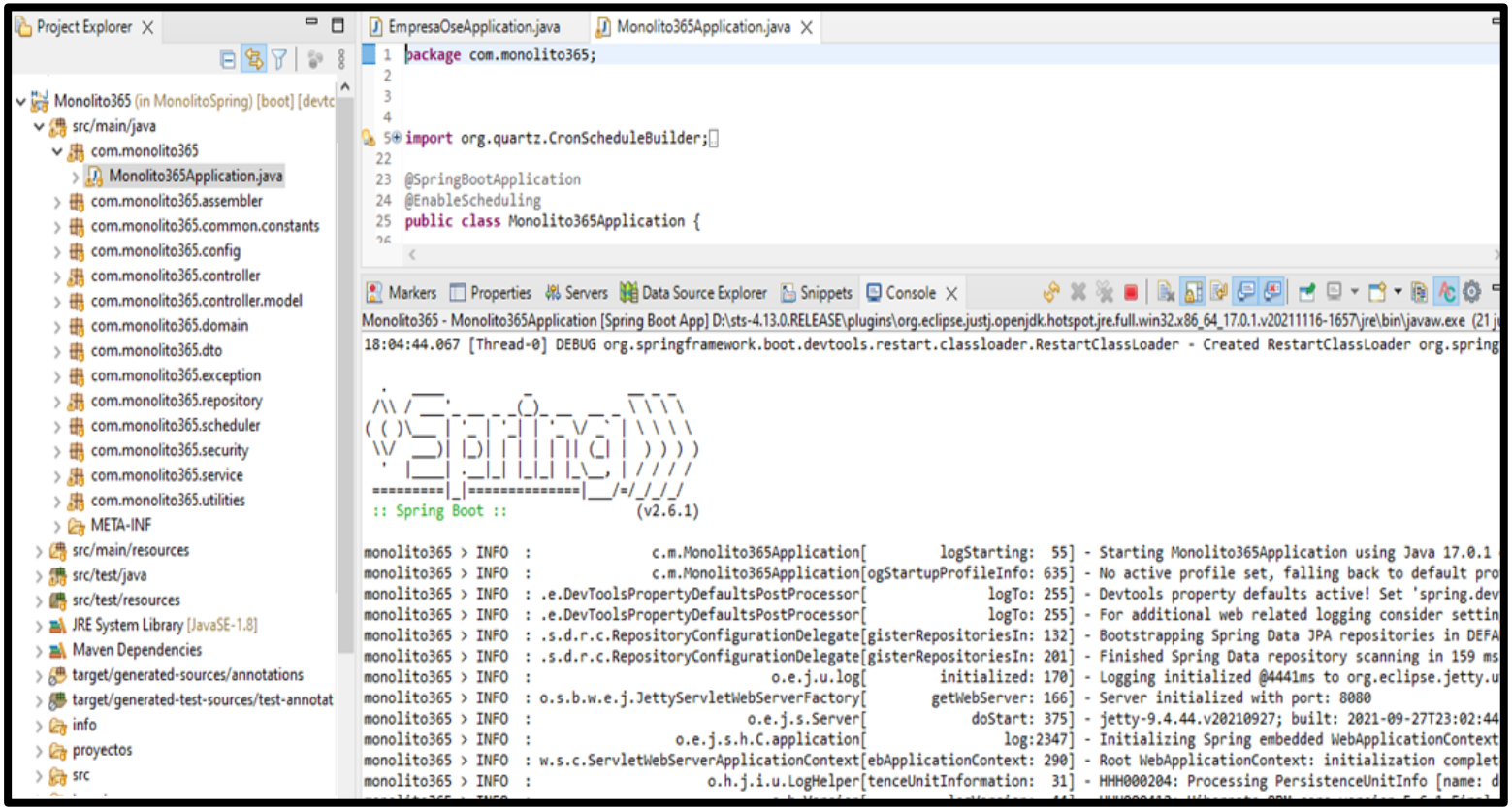


Fuente: Elaboración propia

ANEXO 2

Ejecución del proyecto SpringBoot(Monolito365), basado en la Arquitectura Orientada a Servicios(SOA)

Figura 40: Ejecución de la aplicación Spring Boot, en el IDE Spring Tool Suite

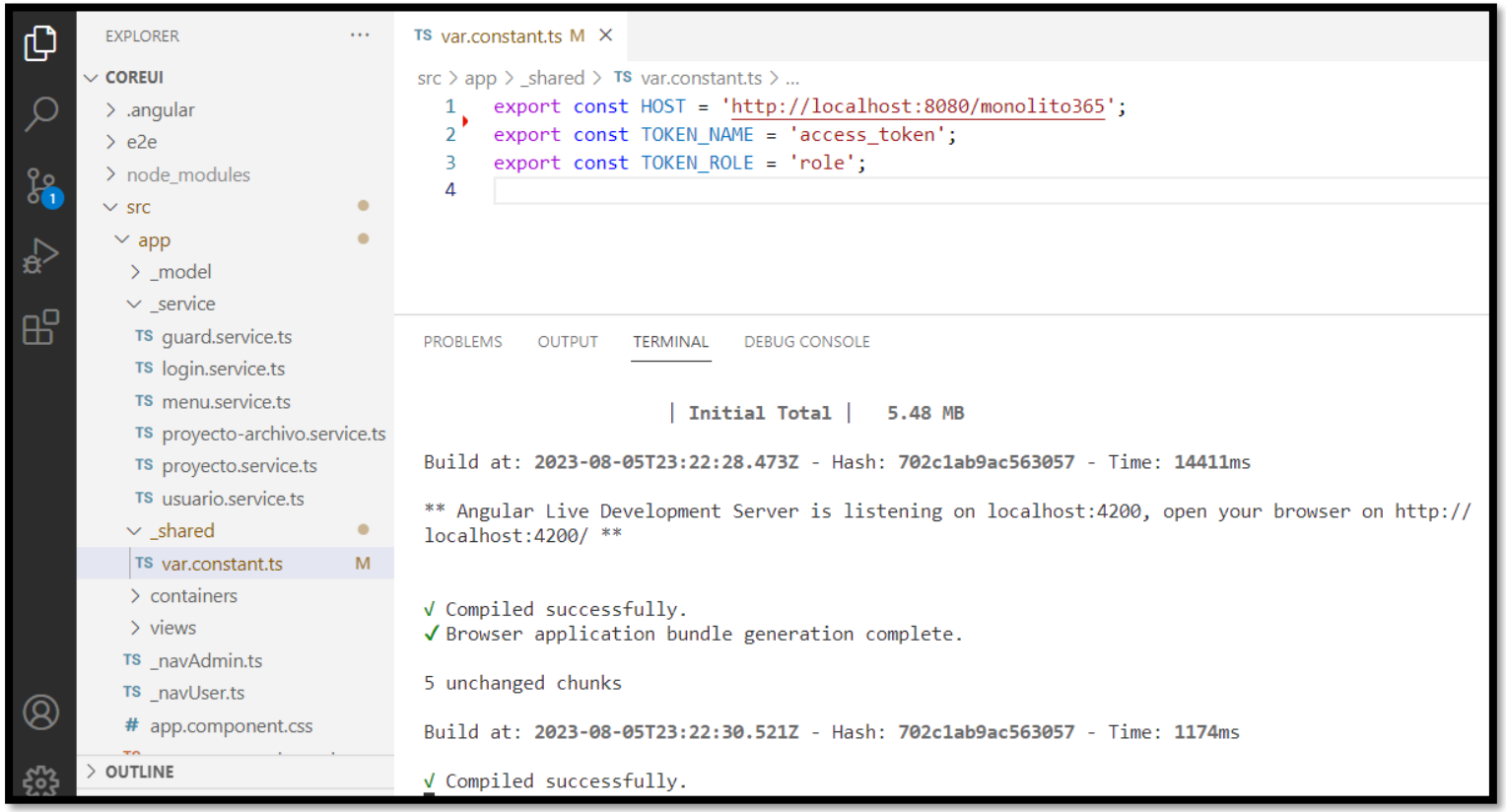


Fuente: *Elaboración propia.*

Ejecución del proyecto Angular, basado en la Arquitectura Orientada a Servicios(SOA).

Se utilizó el siguiente comando de Angular: **ng serve**

Figura 41: Ejecución de la aplicación Angular, en el editor Visual Studio Code



Fuente: *Elaboración propia*

ANEXO 3

Despliegue de los componentes Netflix, para la Arquitectura de Microservicios:

a) Ejecución del Spring Cloud Config.: Se utilizó el siguiente comando:

java -jar target/ms-configserver-0.0.1-SNAPSHOT.jar, como se visualiza en la siguiente imagen, el seguimiento de su ejecución:

Figura 42: Ejecución del Spring Cloud Config, dentro de la línea de comandos Git Bash

Fuente: *Elaboración propia.*

b) Ejecución del Eureka Server. Se utilizó el siguiente comando:

java -jar target/ms-eurekaserver-0.0.1-SNAPSHOT.jar, como se visualiza en la siguiente imagen, el seguimiento de su ejecución:

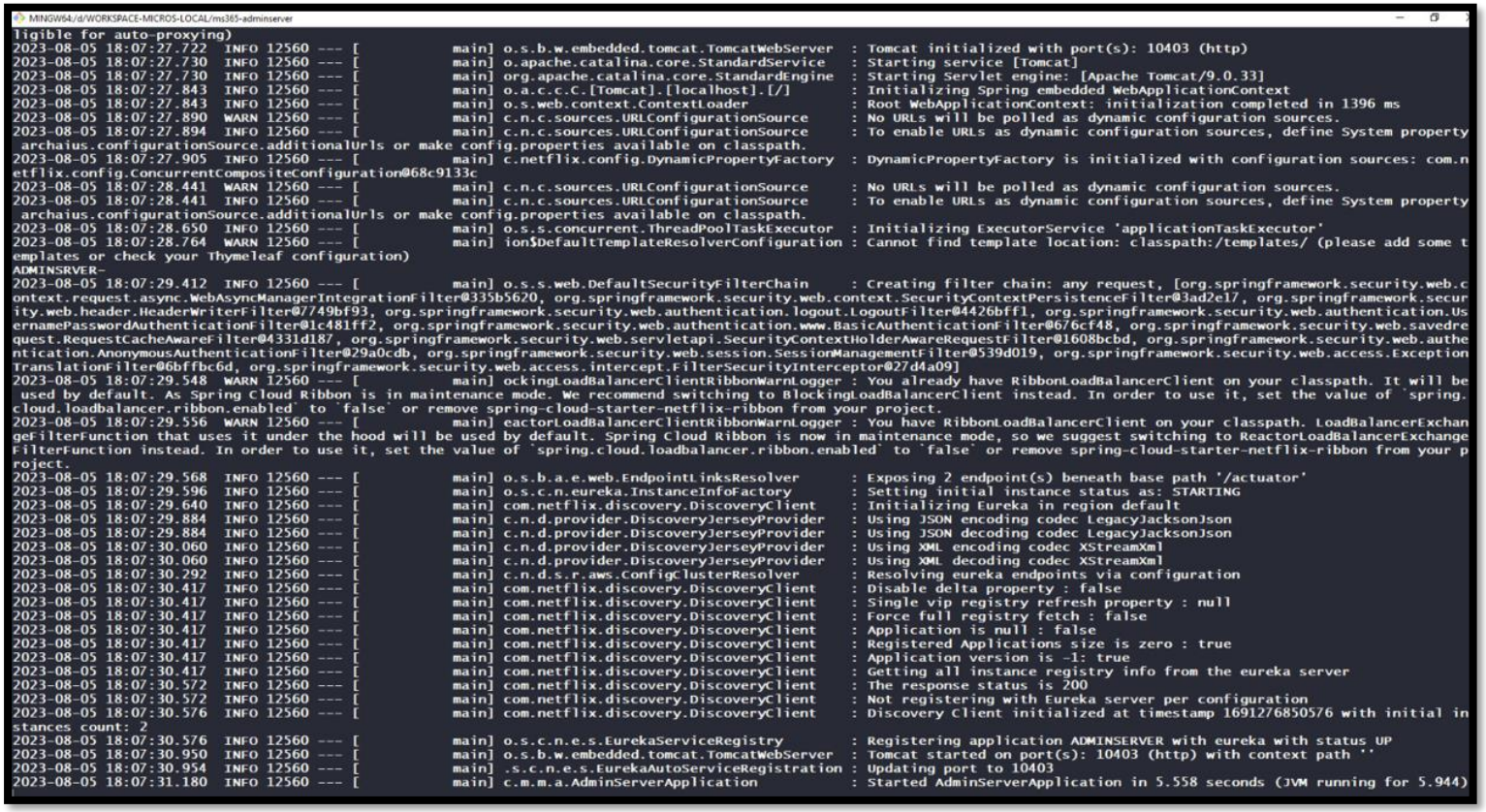
Figura 43: Ejecución del EurekaServer en la línea de comandos Git Bash

Fuente: *Elaboración propia*

c) Ejecución del Spring Boot Admin Server

Se utilizó el siguiente comando: **java -jar target/ms-adminserver-0.0.1-SNAPSHOT.jar**, como se visualiza en la siguiente imagen, el seguimiento de su ejecución:

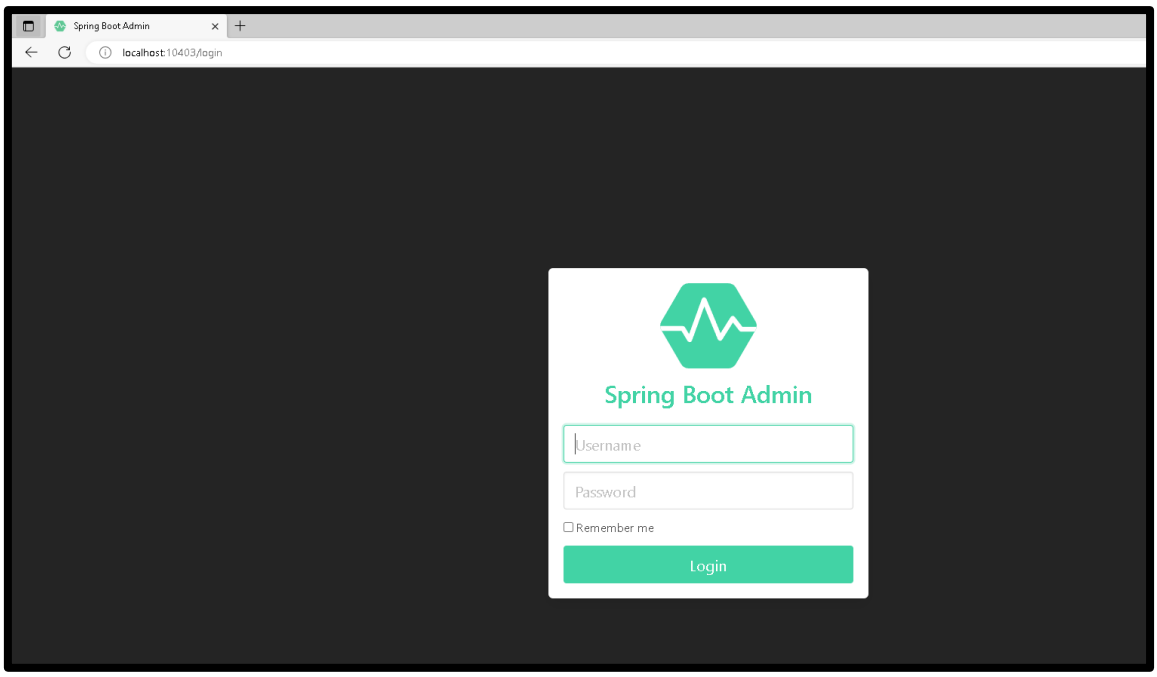
Figura 44: Ejecución del Admin Server en la línea de comandos Git Bash



Fuente: *Elaboración propia.*

Después de ejecutar el componente, se puede observar el panel de control del Spring Boot Admin.

Figura 45: Inicio de sesión al Spring Boot Admin



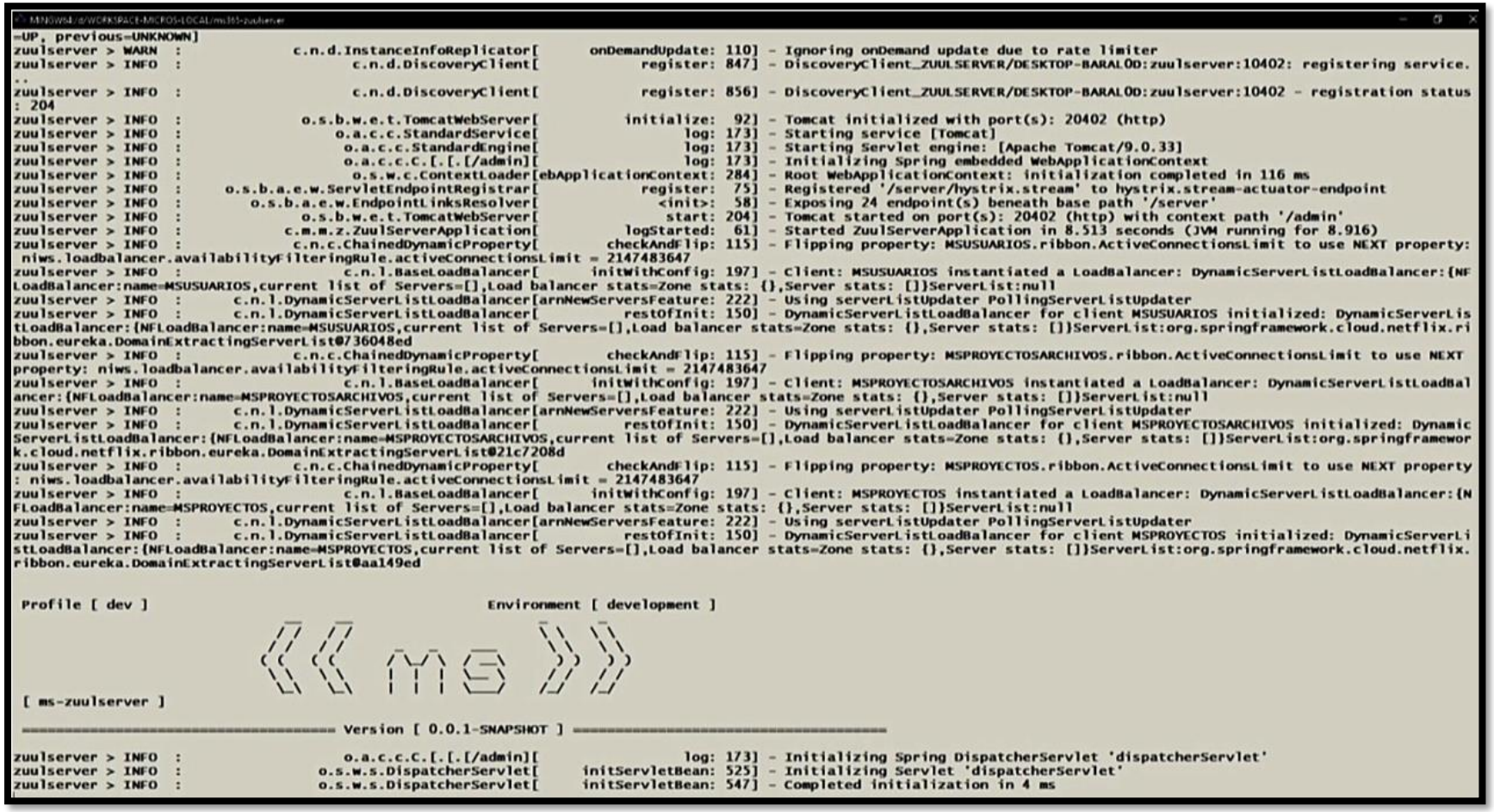
Fuente: *Elaboración propia.*

d) Ejecución del Zuul Server

Se utilizó el siguiente comando:

java -jar target/ms-zuulserver-0.0.1-SNAPSHOT.jar, como se visualiza en la siguiente imagen, el seguimiento de la ejecución de este servicio:

Figura 46: Ejecución del ZuulServer en la línea de comandos Git Bash.



Fuente: *Elaboración propia.*

e) Ejecución del microservicios: usuario-service

Se utilizó el siguiente comando: **java -jar target/usuario-service -0.0.1-SNAPSHOT.jar**, como se visualiza en la siguiente imagen, el seguimiento de su ejecución.

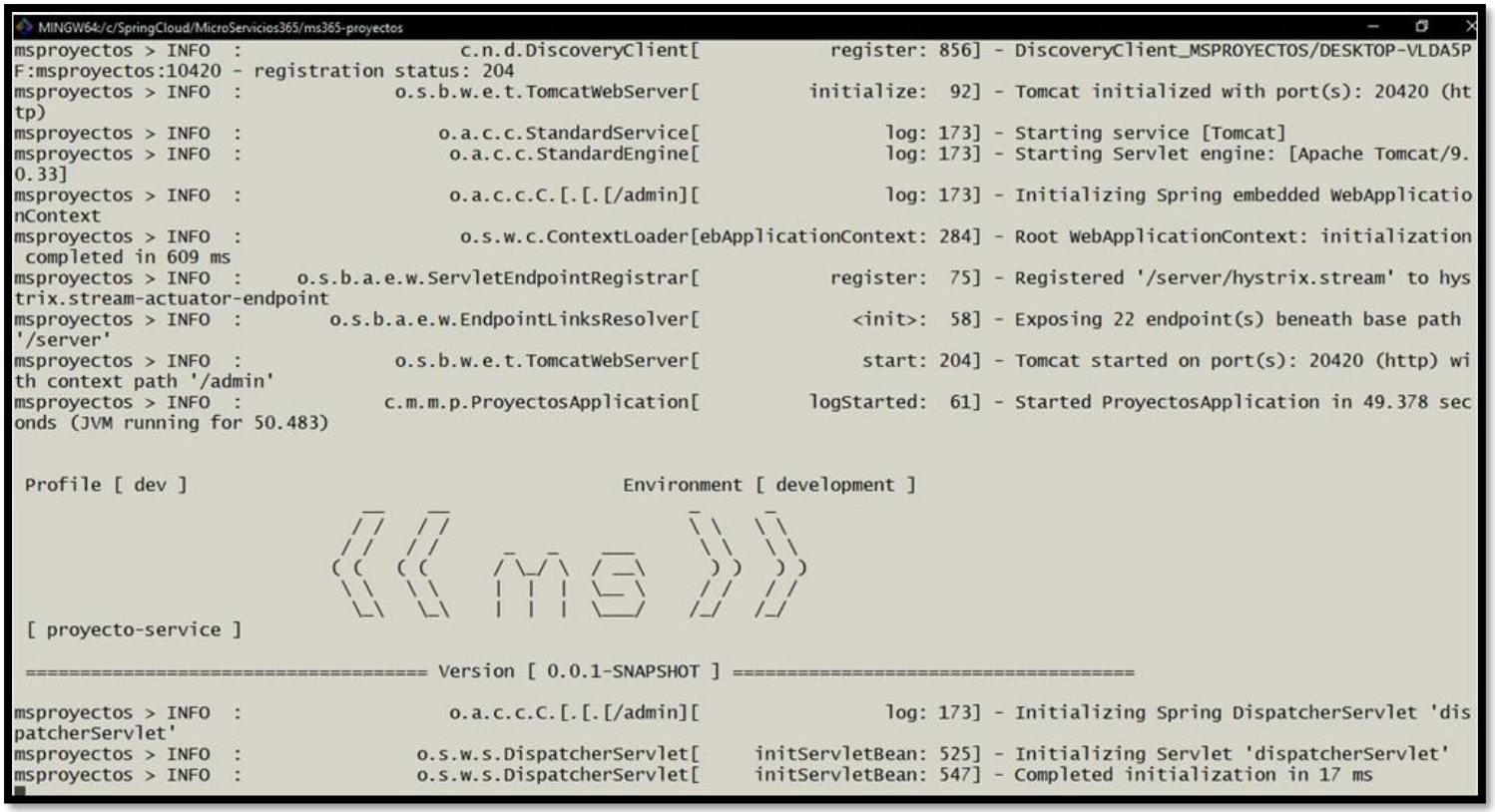
Figura 47: Ejecución del usuario-service, dentro de la línea de comandos Git Bash

Fuente: *Elaboración propia.*

f) Ejecución del microservicios: proyecto-service

Se utilizó el siguiente comando: **java -jar target/proyecto-service -0.0.1-SNAPSHOT.jar**, como se visualiza en la siguiente imagen, el seguimiento de su ejecución:

Figura 48: Ejecución del proyecto-service, dentro de la línea de comandos Git Bash

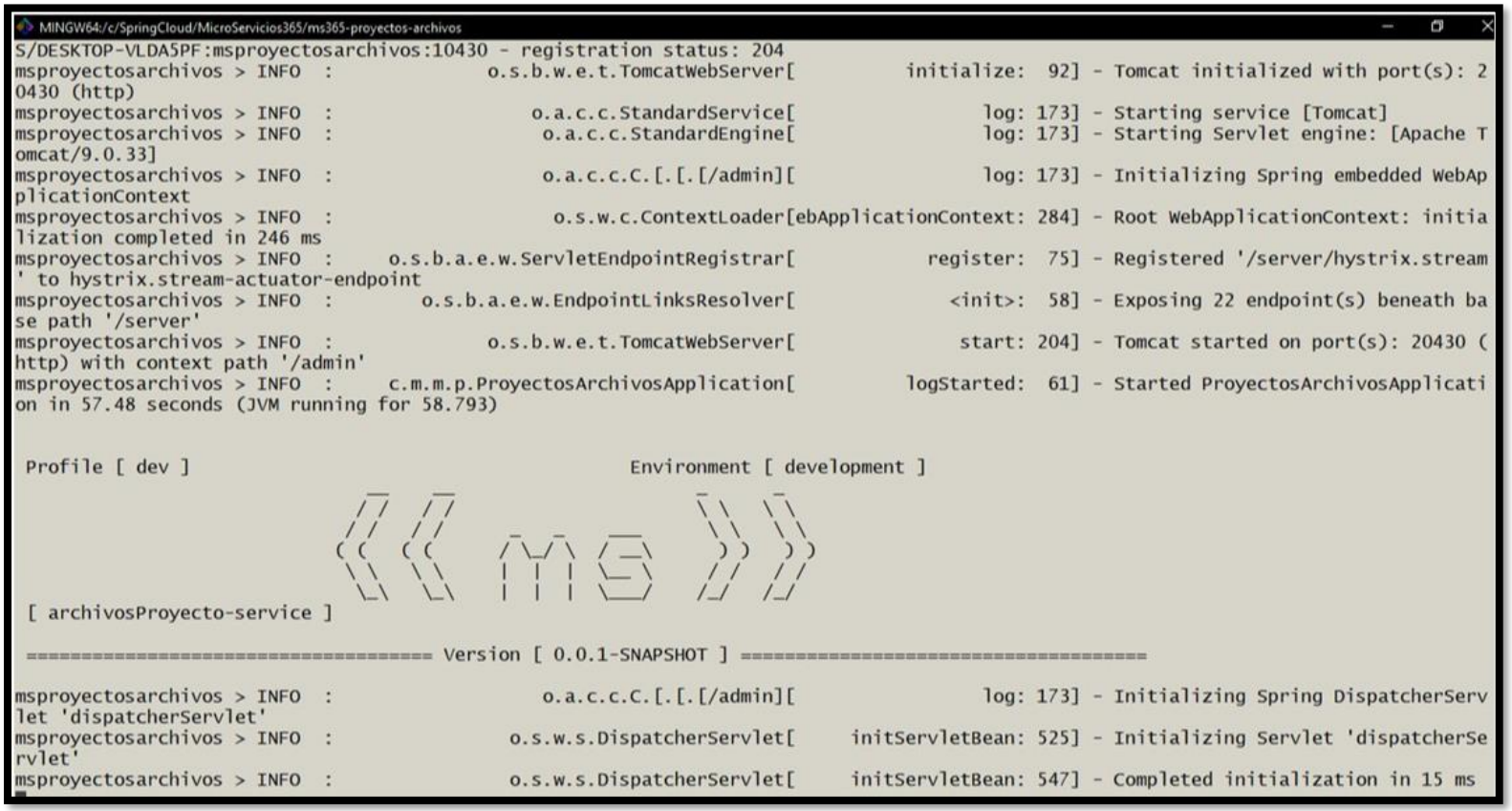


Fuente: *Elaboración propia.*

g) Ejecución del microservicios archivosProyecto-service

Se utilizó el siguiente comando : **java -jar target/archivosProyecto-service-0.0.1-SNAPSHOT.jar**, como se visualiza en la siguiente imagen, el seguimiento de su ejecución.

Figura 49: Ejecución del archivosProyecto-service, dentro de la línea de comandos Git Bash



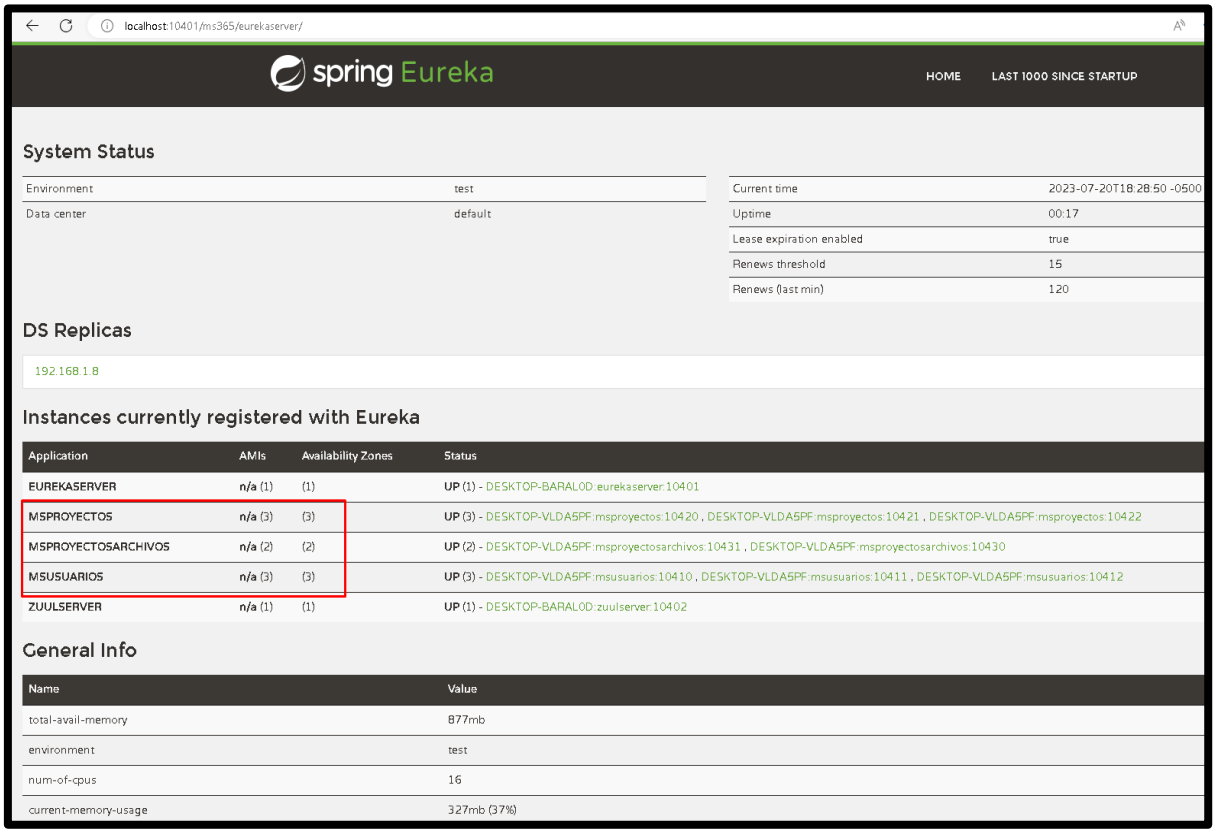
Fuente: *Elaboración propia.*

ANEXO 4

Panel de control del Eureka Server

En este escenario hay 5 microservicios incluyendo el servicio de administración como se muestra en la Figura 50, el panel de control del servicio Eureka.

Figura 50: Panel de control del Eureka Server

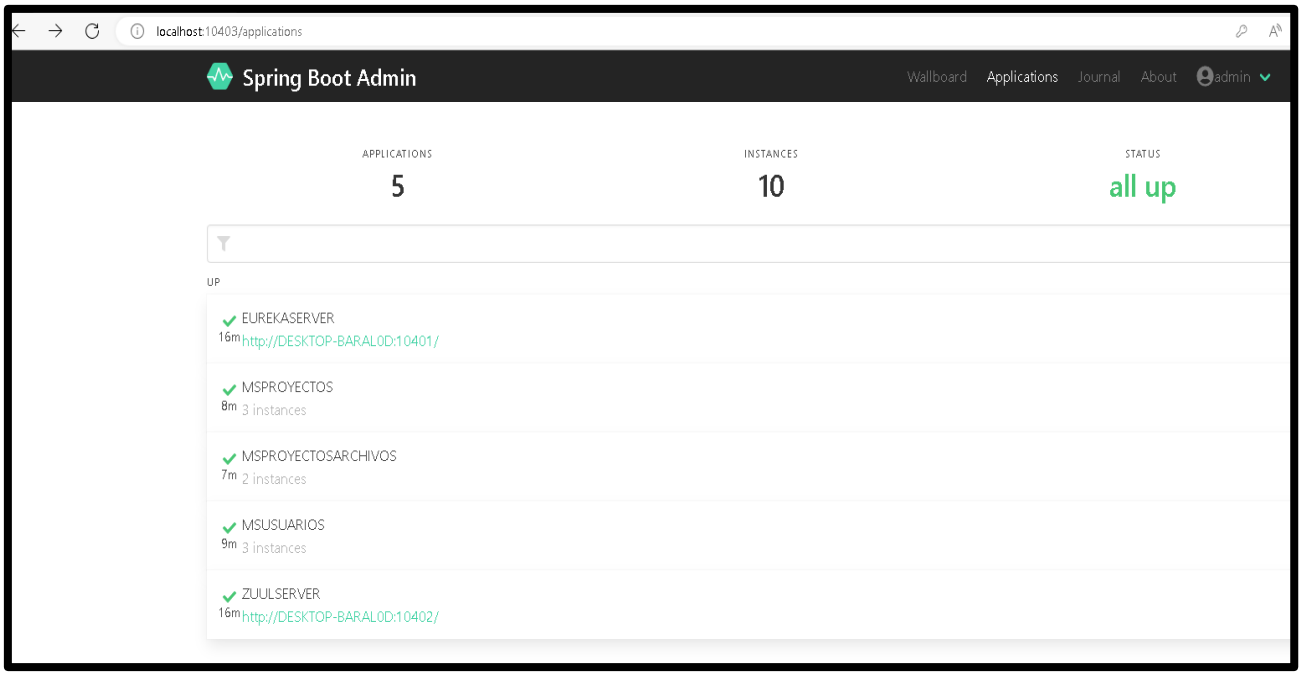


Fuente: *Elaboración propia.*

Aplicación Spring Boot Admin.

Todos los microservicios se registran mediante el servicio de administración. La interfaz de usuario del Admin server se visualiza en la Figura 51.

Figura 51: Panel de control del Spring Boot Admin



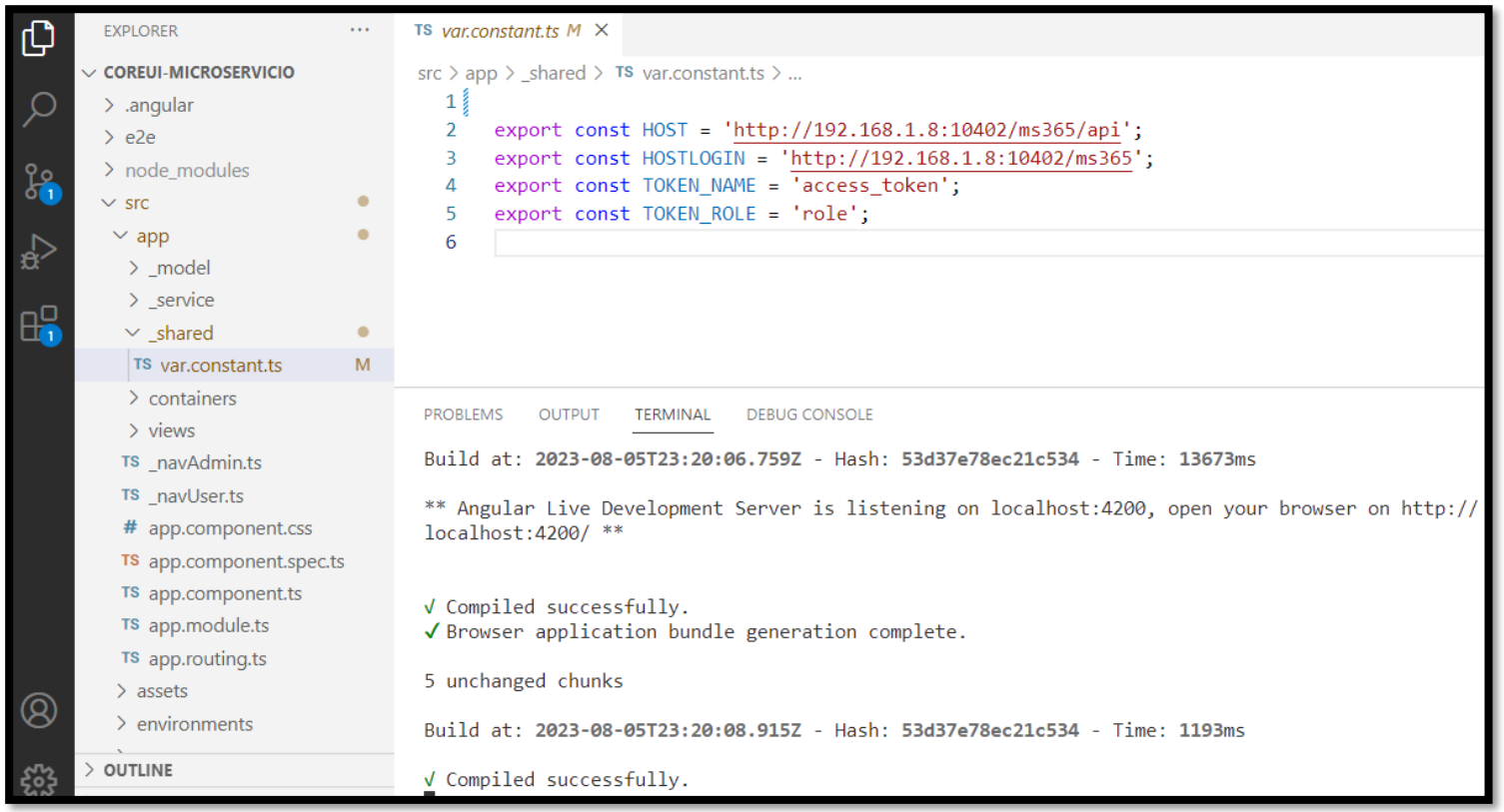
Fuente: *Elaboración propia.*

ANEXO 5

Despliegue de la aplicación Angular, de la Arquitectura Microservicios:

Se utilizó el siguiente comando de Angular: **ng serve**

Figura 52: Ejecución de la aplicación Angular



Fuente: *Elaboración propia.*

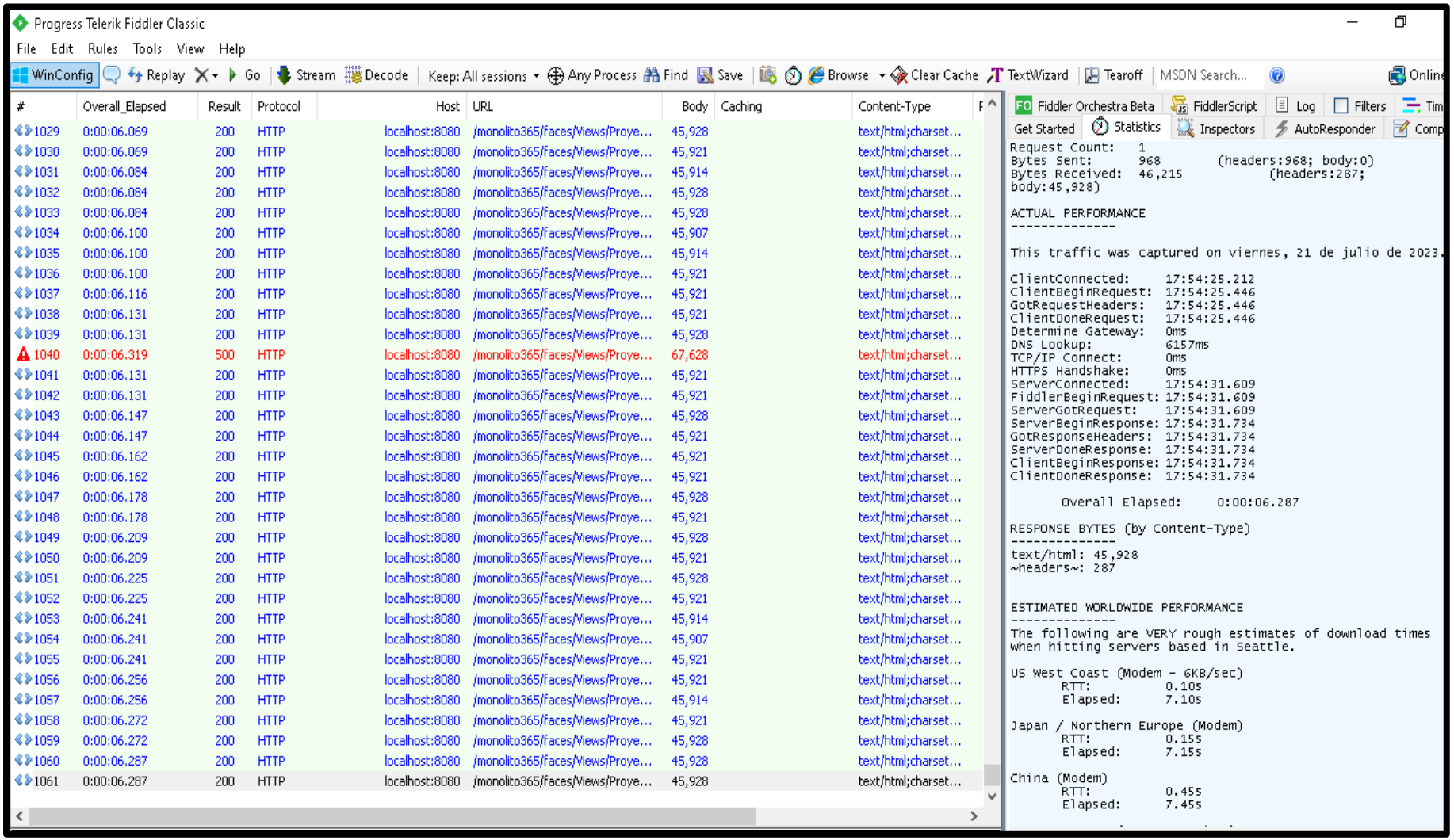
Como se observa en la Figura 55, Las URLs definidas en el archivo var.constant.ts, marcan el punto de entrada hacia el API Gateway(Zuul Server)

ANEXO 6

Resultados de los tiempos de respuesta y códigos de respuesta HTTP de la Arquitectura Multicapa.

Mediante el programa Fiddler Classic, se probó la aplicación Java Server Faces. En este caso, se ingresó mil peticiones concurrentes, donde se observa en las columnas **Overall_Elapsed** (el tiempo de respuesta) y **Result** (el status HTTP), los datos necesarios para hallar los indicadores cuantitativos: Tiempo de respuesta, la disponibilidad y tasa de error.

Figura 53: Reporte de la prueba de carga a la aplicación JSF(Arquitectura Multicapa)



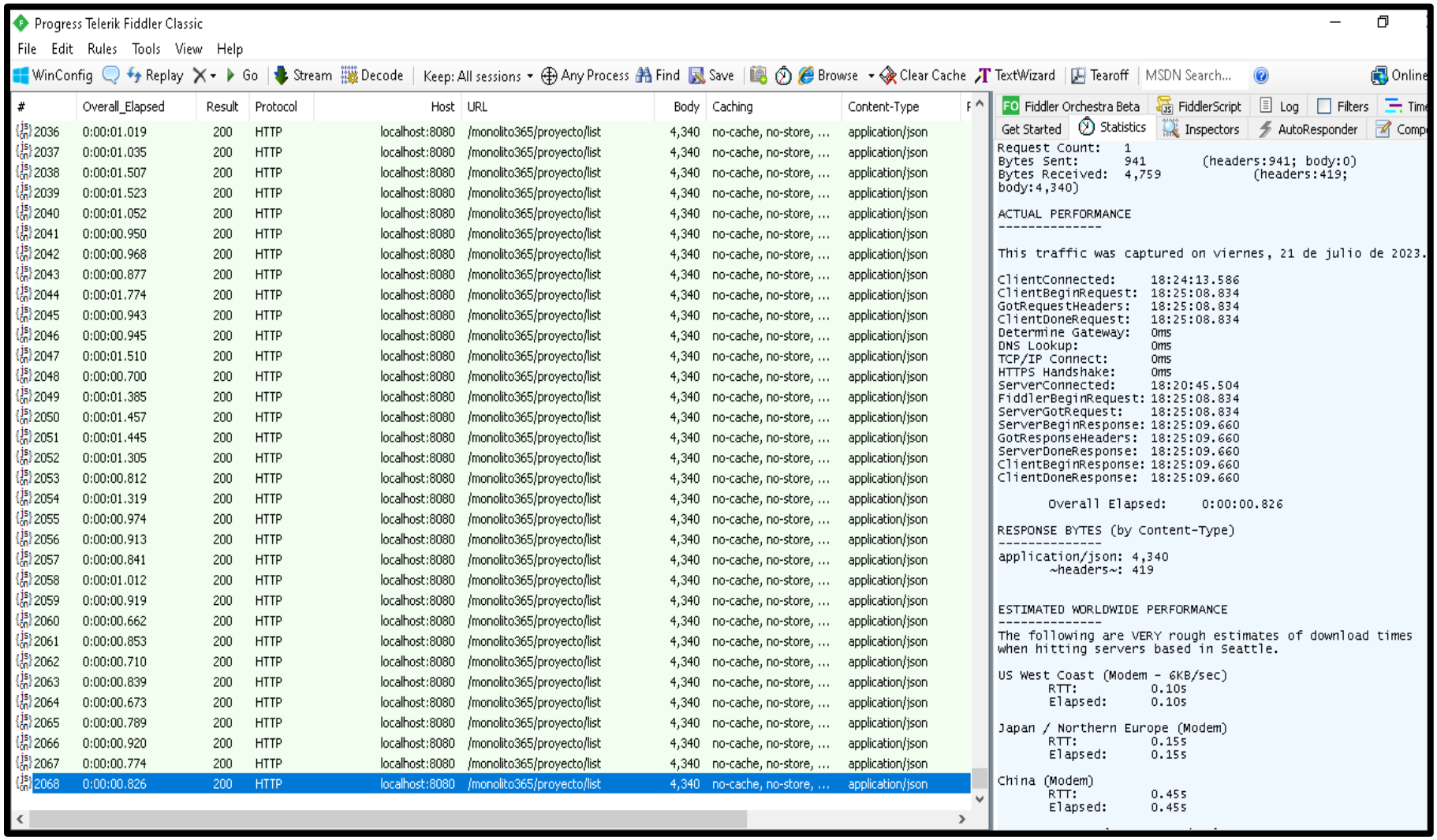
Fuente: *Elaboración propia.*

ANEXO 7

Resultados de los tiempos de respuesta y códigos de estado de respuesta HTTP de la Arquitectura SOA.

Mediante el programa Fiddler Classic, se probó la aplicación Angular (con la Arquitectura SOA). En este caso, se ingresó mil peticiones concurrentes, donde se observa en las columnas **Overall_Elapsed** (el tiempo de respuesta) y **Result** (el status HTTP), los datos necesarios para hallar los indicadores cuantitativos: Tiempo de respuesta, Disponibilidad y Tasa de error.

Figura 54: Reporte de la prueba de carga a la aplicación Angular (Arquitectura SOA).



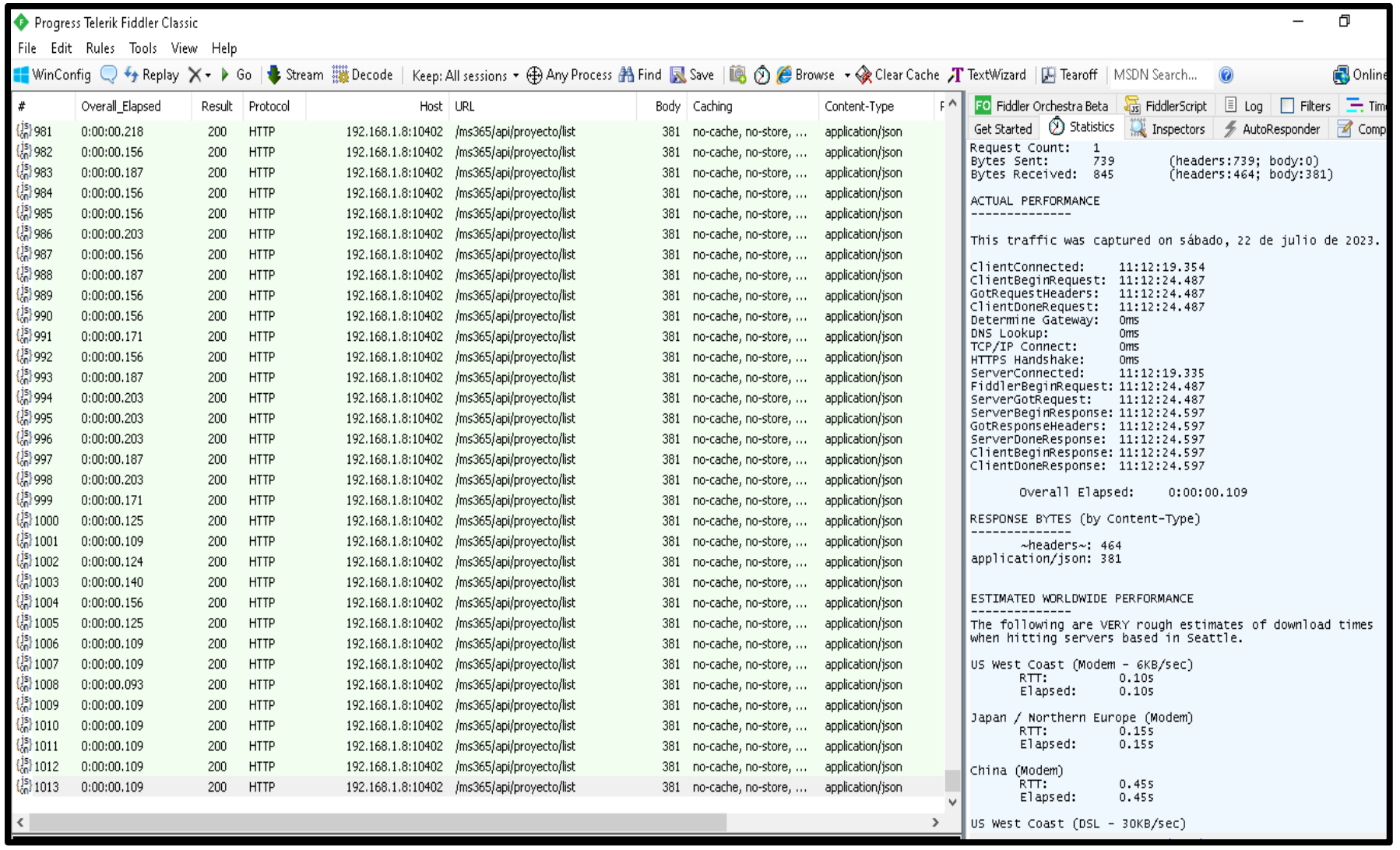
Fuente: *Elaboración propia.*

ANEXO 8

Resultados de los tiempos de respuesta y códigos de estado de respuesta HTTP de la Arquitectura de Microservicios.

Mediante el programa Fiddler Classic, se probó la aplicación Angular (con la Arquitectura de Microservicios). En este caso, se ingresó mil peticiones concurrentes, donde se observa en las columnas Overall_Elapsed (el tiempo de respuesta) y Result (el status HTTP), los datos necesarios para hallar los indicadores cuantitativos: Tiempo de respuesta, Disponibilidad y Tasa de error.

Figura 55: Reporte de la prueba de carga a la aplicación Angular (Arquitectura de Microservicios).



Fuente: *Elaboración propia.*