UNIVERSIDAD NACIONAL DE CHIMBORAZO
FACULTAD DE INGENIERIA
CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

Desarrollo de api Gateway para el sistema de proyectos de investigación de la UNACH basado en microservicios

Trabajo de Titulación para optar al título de Ingeniería en Tecnologías de la información

Autor:
Nuñez Paguay, Jacqueline Elizabeth

Tutor:
Ing. Miryan Estela Narváez Vilema, PhD

Riobamba, Ecuador. 2024

DECLARATORIA DE AUTORÍA

Yo, Jacqueline Elizabeth Nuñez Paguay, con cédula de ciudadanía 0604867739, autora del trabajo de investigación titulado: Desarrollo de API Gateway para el sistema de proyectos de investigación de la UNACH basado en microservicios, certifico que la producción, ideas, opiniones, criterios, contenidos y conclusiones expuestas son de mí exclusiva responsabilidad.

Asimismo, cedo a la Universidad Nacional de Chimborazo, en forma no exclusiva, los derechos para su uso, comunicación pública, distribución, divulgación y/o reproducción total o parcial, por medio físico o digital; en esta cesión se entiende que el cesionario no podrá obtener beneficios económicos. La posible reclamación de terceros respecto de los derechos de autor (a) de la obra referida, será de mi entera responsabilidad; librando a la Universidad Nacional de Chimborazo de posibles obligaciones.

En Riobamba, a los 25 días del mes de noviembre del 2024

Jacqueline Elizabeth Nuñez Paguay

C.I: 0604867739

ACTA FAVORABLE - INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN

En la ciudad de Riobamba, a los 04 días del mes de octubre de 2024, luego de haber revisado el Informe Final del Trabajo de Investigación presentado por la estudiante **JACQUELINE ELIZABETH NUÑEZ PAGUAY** con CC: **0604867739**, de la carrera **INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN** y dando cumplimiento a los criterios metodológicos exigidos, se emite el **ACTA FAVORABLE DEL INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN** titulado **"DESARROLLO DE API GATEWAY PARA EL SISTEMA DE PROYECTOS DE INVESTIGACIÓN DE LA UNACH BASADO EN MICROSERVICIOS"**, por lo tanto se autoriza la presentación del mismo para los trámites pertinentes.

Firmado electrónicamente por:
MIRYAN ESTELA
NARVAEZ VILEMA

PhD. Miryan Narváez
TUTORA

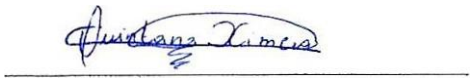
CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

Quienes suscribimos, catedráticos designados Miembros del Tribunal de Grado para la evaluación del trabajo de investigación "Desarrollo de api Gateway para el sistema de proyectos de investigación de la UNACH basado en microservicios " por Jacqueline Elizabeth Nuñez Paguay, con cédula de identidad número 0604867739, bajo la tutoría de PhD, Miryan Estela Narváez Vilema; certificamos que recomendamos la APROBACION de este con fines de titulación. Previamente se ha evaluado el trabajo de investigación y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable firmamos, en Riobamba a los 25 días del mes de noviembre del 2024

Danny Velasco, Mgs
PRESIDENTE DEL TRIBUNAL DE GRADO

Ximena Quintana, PhD
MIEMBRO DEL TRIBUNAL DE GRADO



Diego Reina , Mgs
MIEMBRO DEL TRIBUNAL DE GRADO

C E R T I F I C A C I Ó N

Que, **NUÑEZ PAGUAY JACQUELINE ELIZABETH** con CC: **0604867739**, estudiante de la Carrera **INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN** , Facultad de **INGENIERÍA**; ha trabajado bajo mi tutoría el trabajo de investigación titulado "**DESARROLLO DE API GATEWAY PARA EL SISTEMA DE PROYECTOS DE INVESTIGACIÓN DE LA UNACH BASADO EN MICROSERVICIOS**", cumple con el 6 %, de acuerdo al reporte del sistema Anti plagio **TURNITIN**, porcentaje aceptado de acuerdo a la reglamentación institucional, por consiguiente autorizo continuar con el proceso.

Riobamba, 30 de octubre de 2024

Firmado electrónicamente por:
MIRYAN ESTELA
NARVAEZ VILEMA

PhD. Miryan Narváez
TUTORA

DEDICATORIA

Agradezco primeramente a Dios por darme salud y estar conmigo en todas las circunstancias que he pasado al trascurso de mi vida estudiantil, a mis padres y hermanos por estar ahí conmigo apoyándome en no rendirme con sus palabras de aliento.

También agradezco a mi deporte que tanto amo que es el jiujitsu con el aprendí disciplina, valentía, poder, el no rendirme cuando anhelo triunfar, por cada competencia buena y mala, que me impulsa a ser mejor a levantarme cada día a esforzarme y no digo a un 100%, sino un 1% de cada día dándole todo.

AGRADECIMIENTO

Agradezco a Alex. Asitimbay, por compartir sus conocimientos hacia mi persona por los consejos dentro y fuera de lo profesional por todo el apoyo brindado en todo momento y aspecto en todo este transcurso de esta bonita etapa .

También a mi tutora por guiarme y orientarme en el desarrollo de los documentos por sus sabios conocimientos y por la paciencia que me ha tenido durante el trascurso.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El problema principal que aborda este proyecto es la falta de una infraestructura tecnológica especializada en la UNACH. Actualmente, las barreras tecnológicas limitan la eficiencia en la gestión de la investigación. El diseño y desarrollo de un API Gateway se presenta como una solución estratégica para optimizar la integración de microservicios, superando dichas barreras y potenciando la capacidad investigativa de la institución. La implementación de este proyecto no solo mejora la eficiencia en la gestión de la investigación, sino que también fortalece la posición de la universidad como líder en la convergencia de tecnología y conocimiento científico.

La justificación de este proyecto radica en la necesidad de adoptar tecnologías modernas que faciliten la gestión de datos y optimicen la colaboración entre los investigadores. La estructura de microservicios, apoyada por un API Gateway, permite una mayor flexibilidad y adaptabilidad a los cambios tecnológicos y a las demandas de la investigación académica. Este enfoque tecnológico avanzado no solo mejora la infraestructura actual, sino que también sienta las bases para futuras innovaciones y desarrollos en la universidad.

La implementación técnica del API Gateway incluye la definición de controladores que manejan las solicitudes HTTP (GET, POST, PUT, DELETE) y llaman a los servicios correspondientes en la capa de lógica de negocio, respondiendo en formatos como JSON o XML. Esta estructura facilita la gestión y el procesamiento de datos, asegurando una comunicación eficiente entre los diferentes microservicios y los clientes que interactúan con el sistema.

El proyecto también contempla una evaluación exhaustiva del rendimiento del módulo de proyectos utilizando herramientas como Blazer Meter. Se llevaron a cabo pruebas de carga con múltiples usuarios simultáneos para identificar y resolver problemas de rendimiento, asegurando que el sistema pueda manejar eficazmente las demandas de uso real. Los resultados de estas pruebas se analizaron minuciosamente y se presentaron en forma de gráficos estadísticos, permitiendo una comprensión clara del comportamiento del sistema bajo diferentes condiciones de carga.

Palabras claves: Microservicios, Arquitectura Hexagonal, Api Gateway, Gestión de datos, Azure Devops.

ABSTRACT

The main problem addressed by this project is the need for specialized technological infrastructure at UNACH. Currently, technological barriers limit the efficiency of research management. The design and development of an API Gateway is proposed as a strategic solution to optimize the integration of microservices, overcoming these barriers and enhancing the institution's research capabilities. Implementing this project not only improves efficiency in research management but also strengthens the university's position as a leader in the convergence of technology and scientific knowledge. The justification for this project lies in the need to adopt modern technologies that facilitate data management and optimize collaboration among researchers. The microservices structure, supported by an API Gateway, allows for greater flexibility and adaptability to technological changes and the demands of academic research. This advanced technological approach not only enhances the current infrastructure but also lays the foundation for future innovations and developments at the university. The technical implementation of the API Gateway includes defining controllers that handle HTTP requests (GET, POST, PUT, DELETE) and call the corresponding services in the business logic layer, responding in formats such as JSON or XML. This structure facilitates data management and processing, ensuring efficient communication between the different microservices and the clients interacting with the system. The project also included a thorough performance evaluation of the module using tools such as Blazer Meter. Load tests were conducted with multiple concurrent users to identify and resolve performance issues, ensuring that the system can effectively handle the demands of real-world usage. The results of these tests were thoroughly analyzed and presented in statistical graphs, providing a clear understanding of the system's behavior under different load conditions.

Keywords: Microservices, Hexagonal Architecture, Api Gateway, Data Management, Azure DevOps.

Firmado electrónicamente por:
MARIA FERNANDA PONCE MARCILLO

Reviewed by:
Mgs. Maria Fernanda Ponce
ENGLISH PROFESSOR
C.C. 0603818188

CAPÍTULO I. INTRODUCCIÓN

En la Universidad Nacional de Chimborazo, la excelencia académica se encuentra básicamente ligada a la investigación innovadora y la gestión eficaz de la información científica. En este contexto, surge la necesidad de abordar los desafíos tecnológicos emergentes mediante la " Desarrollo de api Gateway para el sistema de proyectos de investigación de la Unach basado en microservicios ". Este proyecto se gesta con el propósito de fortalecer la infraestructura tecnológica de la institución, aportando una solución específica optimizando la colaboración y la gestión de datos en entornos científicos.

La integración de microservicios se presenta como un componente esencial en este marco como el API Gateway asume un papel central al facilitar la comunicación fluida entre estas unidades independientes, enfocándose en diseñar un API Gateway adaptado a las características particulares de la Universidad Nacional de Chimborazo, considerando las necesidades específicas de proyectos e investigación , enfocado en la eficiencia y seguridad en la gestión de la investigación, sino que también fortalecerá la posición universitaria como líder en la convergencia de tecnología y conocimiento científico.

Este proyecto representa un plan estratégico para fortalecer la infraestructura tecnológica enfocándose en la optimización de la colaboración y la gestión de datos en el contexto de la investigación académica. La integración de microservicios se plantea como un componente esencial potencializando la flexibilidad y agilidad en el intercambio de información entre unidades independientes.

El enfoque de microservicios implica dividir una aplicación en piezas más chicas e independientes, llamadas microservicios. Cada microservicio se dedica a una tarea específica de la aplicación y puede funcionar de manera autosuficiente. A diferencia de la arquitectura monolítica, donde toda la aplicación corre en un solo servidor, los microservicios pueden ejecutarse en múltiples instancias en uno o varios servidores, permitiendo un escalado flexible de recursos según las necesidades de la carga de trabajo (Intel Corporation, 2021).

1.1. Planteamiento del problema

Ante esta problemática, es imperativo abordar la falta de una infraestructura tecnológica especializada. El diseño y desarrollo de un API Gateway adaptado a las necesidades específicas de la Universidad Nacional de Chimborazo se presenta como la solución estratégica para optimizar la integración de microservicios y superar las barreras tecnológicas que actualmente limitan la eficiencia en la gestión de la investigación. Este planteamiento del problema motiva la investigación para identificar y desarrollar un enfoque tecnológico que potencie la capacidad investigativa de la institución.

1.2. Justificación

En la Universidad Nacional de Chimborazo, el crecimiento exponencial de la información científica y la diversificación de proyectos de investigación han generado una complejidad tecnológica que requiere abordarse de manera efectiva. La gestión eficiente de los datos en este entorno demanda una arquitectura tecnológica avanzada que facilite la colaboración y asegure la integridad de los servicios, la falta de una estructura tecnológica robusta para la integración de microservicios en el Sistema de Gestión de la Investigación se presenta como un obstáculo significativo.

La ausencia de un mecanismo centralizado y eficiente para la comunicación entre microservicios dificulta la coordinación fluida entre distintas unidades de investigación. La heterogeneidad en las plataformas y la falta de un estándar para la gestión de datos científicos también contribuyen a la complejidad operativa. Además, la seguridad y la escalabilidad en la transmisión de datos entre microservicios emergen como desafíos cruciales, comprometiendo la confidencialidad y la eficiencia del proceso investigativo.

1.3.Formulación de pregunta

¿Cómo el modelo de calidad FURPS ayudaría a evaluar el rendimiento del API Gateway para el sistema de proyectos de la investigación de la UNACH?

1.4. Objetivos

Objetivos Generales

Implementar una Api Gateway para el sistema de proyectos de investigación de la Universidad Nacional de Chimborazo basado en microservicios.

Objetivos Específicos

- Analizar la arquitectura de microservicios para el desarrollo del API Gateway.
- Diseñar un API Gateway con microservicios para el sistema de investigación de la Universidad Nacional de Chimborazo.
- Evaluar el rendimiento del API Gateway utilizando el modelo de calidad FURPS.

CAPÍTULO II. MARCO TEÓRICO

2.1. Microservicios

La arquitectura de microservicios implica la fragmentación de una aplicación o sistema en unidades más pequeñas, facilita la automatización y proporciona una manera precisa de abstracción. Estas unidades, conocidas como microservicios, pueden ser fácilmente reemplazadas y modificadas, reduce los costos asociados con decisiones erróneas. Además, los microservicios permiten un desarrollo y despliegue ágil de aplicaciones compuestas por unidades independientes, autónomas, modulares y autocontenidas, con el enfoque tradicional o monolítico, la Figura 1 presenta el patrón básico de arquitectura de microservicios (IBM, s.f.).

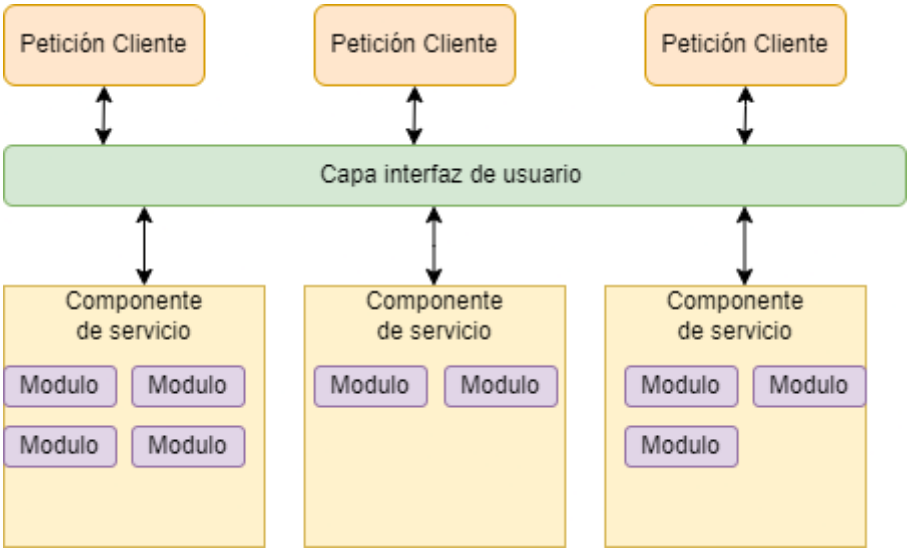


Figura 1: Patrón básico de arquitectura de Microservicio

Una de las ventajas de utilizar microservicios es la capacidad de publicar una aplicación grande como un conjunto de pequeñas aplicaciones (microservicios) que se pueden desarrollar, desplegar, escalar, manejar y visualizar de forma independiente, la Figura 2 presenta los beneficios de los microservicios.

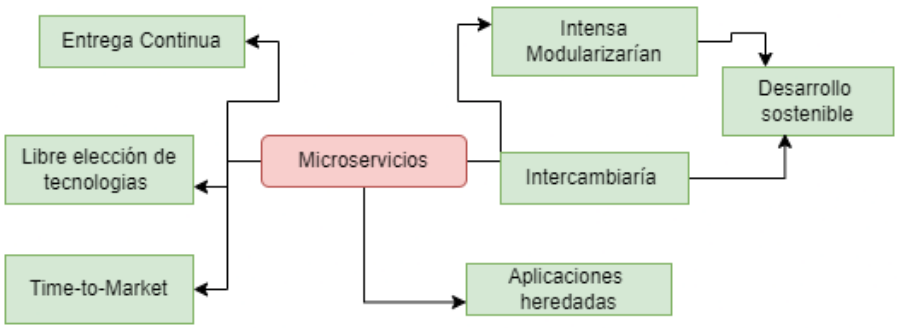


Figura 2: Beneficios de los microservicios

2.1.1. Arquitectura Hexagonal

También conocida como arquitectura de puertos y adaptadores, es un patrón de diseño de software que busca organizar un sistema a partir de un elemento central llamado "aplicación", esta arquitectura promueve la separación de preocupaciones, la flexibilidad y la estabilidad, ya que la aplicación no debe tener ninguna referencia hacia elementos fuera de su hexágono, en la figura 3 se presenta la arquitectura hexagonal (Novoseltseva, 2020).

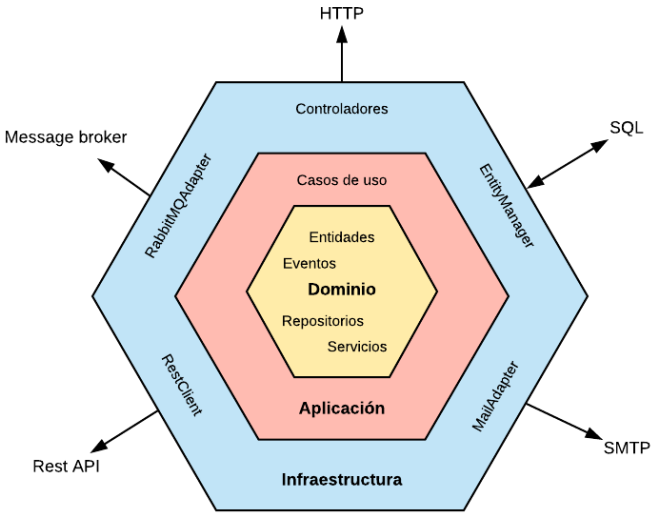


Figura 3: Arquitectura Hexagonal
Fuente: (Salguero, 2018)

La arquitectura hexagonal propone que el núcleo de un sistema de software, o su dominio, esté en el centro de todas las capas y que esté completamente independiente de cualquier elemento externos (Salgero, 2018).

2.1.2. Beneficios de la Arquitectura Hexagonal

La arquitectura hexagonal ofrece un beneficio clave al separar el núcleo del sistema de sus elementos externos, esto impulsa flexibilidad, modularidad y facilita el mantenimiento, también traduce en una mayor agilidad en el desarrollo de software.

Mantenibilidad
Dado que se está separando claramente la aplicación de otras dependencias (como un framework web), esto facilita cambiar las dependencias sin que sea necesario afectar la lógica de negocio.

Estabilidad
Separar las dependencias no solo simplifica los cambios en el código, sino que también facilita su prueba. En aplicaciones donde el código está entrelazado (por ejemplo, un método que contiene lógica se comunica con una API y guarda datos en una base de datos), resulta difícil probar una parte específica.

2.1.3. Componentes de la arquitectura hexagonal

Núcleo o dominio de la aplicación (Domain Layer)

El núcleo contiene la lógica de negocio y las reglas de negocio que definen el comportamiento de la aplicación con el funcionamiento del dominio donde se basa en la separación de responsabilidad el núcleo de la aplicación no debe preocuparse por cómo se almacenan o se recuperan los datos (Coppola, 2023).

Puertos (ports)

Son independientes de la tecnología subyacente y se definen en el Domain Layer con un objetivo principal de los puertos es permitir la separación de responsabilidades, permitiendo que el núcleo de la aplicación se centre en la lógica de negocio y las reglas de negocio, sin preocuparse por cómo se implementan las interfaces externas (Coppola, 2023).

Adaptadores (adapters)

En la arquitectura hexagonal, los adaptadores son elementos esenciales que se ocupan de llevar a cabo la funcionalidad de los puertos, facilitando la interacción entre la aplicación y su entorno externo, estos elementos específicos de la tecnología utilizada tienen la capacidad de transformar los datos de entrada y salida en formatos apropiados para que el núcleo de la aplicación pueda procesarlos (Brutti, 2023).

2.3. API Gateway

Los gateway de aplicaciones, como los API Gateways, se centran en la gestión de la comunicación entre aplicaciones y servicios, así como en la Figura 4. Es decir, proporcionan un punto de control para el acceso a APIs, optimizando la gestión de solicitudes, la seguridad y el análisis de datos. Esto los hace imprescindibles para las arquitecturas de microservicios, donde múltiples servicios de pequeño tamaño y autónomos necesitan interactuar de forma eficiente.

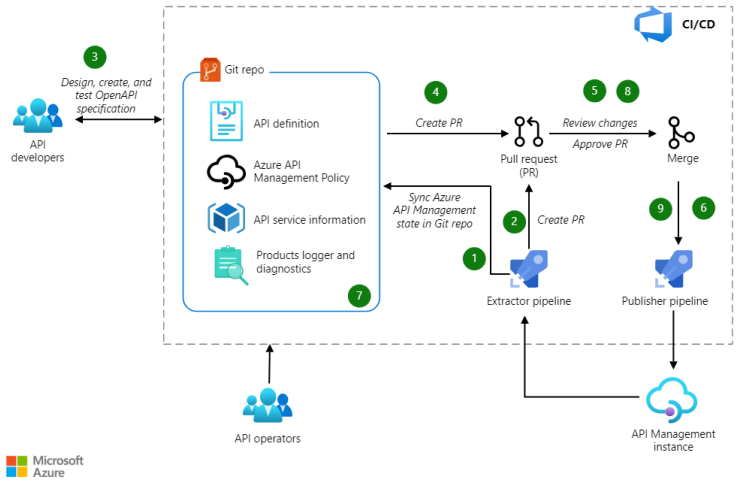


Figura 4: Api Gateway en Aplicaciones

Fuente: (microsoft, s.f.)

2.3.1 Beneficios del API Gateway

- **Mejora de la seguridad**: Los API Gateways están diseñados con características de seguridad altamente fiables para proteger los servicios expuestos externamente, incluyendo autenticación, autorización, limitación de tasa y monitoreo de tráfico.
- **Monitoreo y análisis del uso de las APIs:** Las API Gateway pueden facilitar el monitoreo y el análisis del uso de las APIs, permitiendo a los desarrolladores identificar y solucionar problemas de rendimiento y seguridad.
- **Visibilidad:** Las API Gateway proveen métricas sobre la cantidad de llamadas, latencia, tasa de errores y logs para facilitar las tareas de debugging.

2.3.2 Características

- Compatibilidad con las API con estado (WebSocket) y las API (HTTP y REST).
- Mecanismos de autenticación eficaces y flexibles, como políticas de AWS Identity and Access Management, funciones de autorizador de Lambda y grupos de usuarios de Amazon Cognito.
- Implementaciones de la versión Canary para el despliegue de cambios de forma segura.
- Registro de CloudTrail y monitoreo del uso y de los cambios en las API.

2.4. Azure DevOps

Según Hernandez (2022), Azure DevOps es un término que abarca todo el ciclo de vida del desarrollo, desde la fase de planificación y codificación hasta las de pruebas, implantación y operaciones.

2.4.1 Para qué sirve Azure DevOps

Azure DevOps ofrece una amplia gama de características y funcionalidades que permiten a los equipos de desarrollo colaborar de manera efectiva y optimizar el proceso de desarrollo de software (Martinez, 2024).

- **Gestión del código fuente:** Permite almacenar y administrar el código fuente de los proyectos en repositorios de código como Git o TFVC.
- **Seguimiento de trabajo:** Facilita la planificación, asignación y seguimiento de las tareas, problemas y requisitos del proyecto.
- **Compilación e implementación continua:** Automatiza el proceso de compilación, prueba y despliegue de aplicaciones, permite una entrega rápida y confiable.
- **Pruebas automatizadas:** Proporciona herramientas para crear y ejecutar pruebas automatizadas, mejora la calidad y la confiabilidad del software.

- **Gestión del ciclo de vida del desarrollo de software:** Permite gestionar todo el ciclo de vida del desarrollo de software, desde la planificación y seguimiento hasta el despliegue y la monitorización.

2.5. SQL Server Management Studio (SSMS)

Desarrollo integrado para administrar cualquier infraestructura SQL. Se utiliza para acceder, administrar, configurar y desarrollar todos los componentes de SQL Server y SQL Data base, así también como Microsoft lo ha optimizado a lo largo de los años y es un programa de administración de servidores y bases de datos muy popular (Softtrader, 2021).

2.6 Bibliotecas y Framework

2.6.1 .Net 7

Es una plataforma de aplicaciones que permite la creación y ejecución de servicios web y aplicaciones de Internet. En la plataforma de desarrollo se pueden utilizar una serie de lenguajes, implementaciones, herramientas y bibliotecas para el desarrollo de las aplicaciones (Aula21, 2023).

.NET 7 brinda a sus aplicaciones un mayor rendimiento y nuevas características para C# 11/F# 7, .NET MAUI, ASP.NET Core/Blazor, Web API, WinForms, WPF y más. Con .NET 7, también puede contener fácilmente sus proyectos .NET 7, configurar flujos de trabajo de CI/CD en acciones de GitHub y lograr observabilidad nativa de la nube (Douglas et al., 2022).

En la Tabla 1, se muestra varias características con respecto a sus anteriores versiones como se muestra a continuación:

Tabla 1: Características .net 7 y sus otras versiones

Característica	.NET 7	.NET 5	.NET Core 3
Rendimiento	Incremento sustancial en la velocidad y eficiencia de ejecución.	Mejoras en rendimiento respecto a .NET Core.	Focalización inicial en mejorar el rendimiento sobre versiones .NET Framework.
Versiones de C# y F#	Soporta C# 11 y F# 7, introduciendo nuevas funcionalidades y mejoras en el lenguaje.	Soporte para C# 9 y F# 5, con menos características que C# 11/F# 7.	Soportaba C# 8, con menos mejoras en el lenguaje que versiones posteriores.
.NET MAUI	Introduce o mejora significativamente el soporte para .NET MAUI.	.NET 5 no incluía soporte para .NET MAUI.	.NET Core 3 no contaba con .NET MAUI.
ASP.NET Core/Blazor	Avances importantes en ASP.NET Core y Blazor para un desarrollo web más eficiente.	Menos características y optimizaciones en ASP.NET Core y Blazor comparado con .NET 7.	Introducción de Blazor, con menos capacidades que en versiones posteriores.
APIs Web	Mejoras en la eficiencia y facilidad de desarrollo de APIs Web.	APIs Web con menos optimizaciones comparadas con .NET 7.	Soporte para APIs Web, pero sin las últimas mejoras de .NET 7.
WinForms y WPF	Actualizaciones significativas en WinForms y WPF.	Soporte para WinForms y WPF con menos actualizaciones.	Incluyó soporte para WinForms y WPF adaptados a .NET Core.
Contenerización	Facilita y optimiza la contenerización de proyectos.	La contenerización era posible pero menos optimizada.	Menos enfoque en la integración y optimización de contenerización.
CI/CD en GitHub Actions	Integración y configuración más simples para CI/CD.	Integración para CI/CD presente, pero no tan refinada.	No había integraciones específicas para CI/CD en GitHub Actions.
Observabilidad en la Nube	Mejora considerable en la observabilidad de aplicaciones en entornos de nube.	Menos capacidades para observabilidad en la nube.	Observabilidad en la nube menos desarrollada.
Contribuciones de la comunidad	Elevada participación comunitaria en el desarrollo y contribuciones.	Contribuciones comunitarias presentes, pero no al nivel de .NET 7.	Participación comunitaria en el desarrollo, pero en menor escala.

2.6.2. Bibliotecas estándares de .NET

- **Bibliotecas de clases específicas de la plataforma**
  Están concebidas para operar exclusivamente en una plataforma de .NET, como el .NET Framework en Windows. Esto implica que pueden aprovechar al máximo las funcionalidades y APIs de dicha plataforma, sin necesidad de escribir código

adicional para adaptarse a distintos entornos, han surgido otras implementaciones de .NET, estas bibliotecas continúan siendo las más empleadas en el desarrollo de software dentro del ecosistema .NET.

- **Bibliotecas de clases portables**
  Las bibliotecas portátiles son compatibles con distintas implementaciones de .NET. Pueden depender de un entorno de ejecución conocido, sin embargo, este entorno es sintético y se crea a partir de la intersección de un conjunto de implementaciones específicas de .NET.
- **Bibliotecas de clases .NET Standard**
  Las bibliotecas de .NET Standard sustituyen los conceptos de bibliotecas específicas de la plataforma y bibliotecas portátiles. Son específicas de la plataforma porque ofrecen toda la funcionalidad de la plataforma subyacente (sin necesidad de plataformas sintéticas ni intersecciones de plataformas).

2.7. Metodología Kanban

Las estrategias Kanban se implementan a través de tableros. Se trata de una estrategia de gestión de proyectos visual que permite a los equipos ver sus flujos de trabajo y la carga de trabajo. Un tablero con columnas Kanban es una excelente manera de mostrar durante todo el proceso del proyecto. La mayoría de las veces, cada columna representa una etapa del trabajo (Martins, 19).

El tablero Kanban más básico puede presentar columnas como Trabajo pendiente, en progreso y Terminado. Las tareas individuales representadas por tarjetas visuales en el tablero avanzan a través de las diferentes columnas hasta que estén finalizadas como se muestra en la Figura 5.

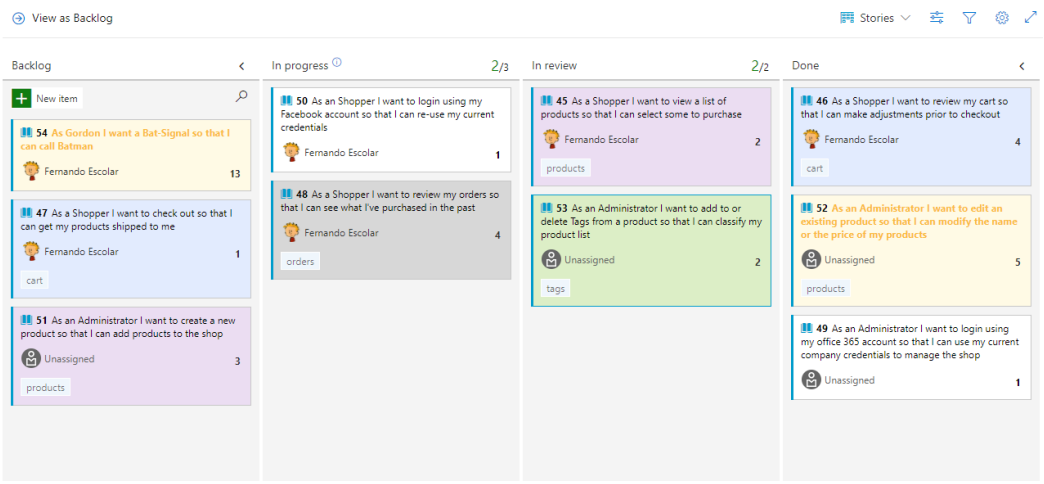


Figura 5: Tableros Kanban
Fuente: (Método Kanban con Azure DevOps, 2019)

2.8. C#

Es un lenguaje de programación moderno, basado en objetos y con seguridad de tipos. Los desarrolladores pueden crear una variedad de aplicaciones seguras y robustas que se ejecutan

en.NET con C#. C# se basa en la familia de lenguajes C, programadores que usan C, C++, Java y JavaScript lo conocerán inmediatamente (Microsoft Build, 2024).

2.9 Modelo de Calidad FURPS

Es un modelo de calidad fijo propuesto por Packard & Robert (1987) incluyen, que los factores de calidad y el atributo como las restricciones de diseño y requerimientos de implementación, físicos y de interfaz.

El modelo FURPS, creado por Hewlett-Packard (HP) en 1987, es un marco para la calidad del software que cubre diversos factores y atributos clave para evaluar y gestionar la calidad de un producto de software. FURPS es un acrónimo de Funcionalidad, Usabilidad, Fiabilidad, Rendimiento y Soporte. Cada uno de estos aspectos se centra en características específicas de calidad que son vitales para el éxito y la satisfacción del usuario final.

2.9.1 Parámetros de medición del rendimiento

Es el período que se registra desde que se accede a un sitio web o una aplicación específica hasta que se visualiza completamente en la pantalla, en este proceso mediante el Protocolo de Transferencia de Hipertexto (HTTP) pueden experimentar cambios los recursos empleados por el servidor de aplicaciones (Solvetic, 2015).

- **Eficacia:** capacidad del sistema de completar las tareas asignadas y cumplir con un objetivo propuesto.
- **Tiempo de Respuesta:** El plazo designado para finalizar una tarea, considerando las operaciones en la memoria RAM, el disco, las actividades de entrada/salida y los recursos del sistema operativo. Usualmente, el límite de tiempo aceptable para la respuesta de un sistema es de 5 segundos para garantizar su eficiencia.
- **Utilización de Recursos:** hace referencia a la cantidad de recursos software necesarios para el correcto funcionamiento del sistema.

En la Tabla 2 se distribuye en cinco atributos como criterios de calidad para todas las etapas del desarrollo de una red de calidad, de los cuales se deriva el acrónimo FURPS.

Tabla 2: Criterios asociados a factores de calidad FURPS

SIGLA	TIPO DE REQUERIMIENTO		DESCRIPCIÓN
F	Funtional	Funcional	Características, capacidades y algunos aspectos de seguridad.
U	Usability	Facilidad de uso	Factores humanos (interacción), ayuda documentación.
R	Reliability	Fiabilidad	Frecuencia de fallos, capacidad de recuperación de un fallo y grado de previsión.
P	Performance	Rendimiento	Tiempo de respuesta, productividad, precisión, disponible uso de recursos.
S	Supportabilidad	Soporte	Adaptabilidad y fortalecer el mantenimiento, internalización, facilidad de configuración.

2.9.2 Valores de la ponderación del rendimiento según el modelo FURPS

Según Vivanco (2018), establece que la dimensión de ponderación establecidas por el modelo de calidad de software FURPS son aquellas que se muestran en la Tabla 3.

Tabla 3: Valores de la ponderación del rendimiento según el modelo FURPS

Criterios Por Evaluar	Dimensión de ponderación según FURPS
Eficiencia (E)	95%
Consumo de Recurso (CR)	25%
Tiempo de Respuesta (TR)	5sg

Fuente (Alvarado, 2023)

CAPÍTULO III. METODOLOGÍA

3.1 Metodología de Investigación

En el proyecto de investigación se empleó un enfoque cuantitativo, permitiendo obtener información precisa, durante este proceso, se llevará a cabo una evaluación de rendimiento basado el modelo de calidad FURPS para medir el rendimiento del Api Gateway.

3.2 Tipo de Investigación

Se empleo la investigación en el desarrollo de un API Gateway para el sistema de proyectos de investigación de la UNACH basado en microservicios, ofrece beneficios significativos en términos de eficiencia, seguridad, rendimiento y calidad en la gestión de proyectos de investigación académica.

3.3 Diseño de Investigación

La adquisición de información y fuente de datos se empleó la investigación bibliográfica, para recabar información y lograr el objetivo propuesto. Como parte necesaria logrando una buena recolección de datos del consumo del API, permitiendo establecer los requisitos del sistema.

La investigación utilizó un método no experimental para la construcción de sus resultados, es decir no existió manipulación de variables previo el análisis de datos, así también se respaldó con un método documental debido a la búsqueda de fuentes bibliográficas correspondientes a microservicios.

3.4. Población y Muestra

El colectivo del estudio se construyó por una población infinita y ante la falta de un marco muestral delimitado no es posible el cálculo de una muestra, sin embargo, se consumió la herramienta Blazer Meter para lograr datos referentes a los indicadores específicos por el modelo de calidad FURPS.

3.5 Técnicas de Recolección de Datos

Se empleó la herramienta Blazer Meter para la recopilación de datos.

3.6. Indicación de Variables

3.6.1 Variable Independiente

Api Gateway para el sistema de proyectos de investigación.

3.6.2 Variable Dependiente

Evaluar el rendimiento del api Gateway.

3.7. Operacionalización de variables

La Tabla 4, presenta a las variables definidas y su respectiva operacionalización con el propósito de llevar a cabo una medición efectiva de los resultados de la investigación.

Tabla 4: Operacionalización de Variables

PROBLEMA	TEMA	OBJETIVOS	VARIABLES	CONCEPTUALIZACIÓN	DIMENSIÓN	INDICADORES
¿Cómo el modelo de calidad FURPS ayudaría a evaluar el rendimiento del API Gateway para el sistema de proyectos de la investigación de la UNACH?	Desarrollo de API Gateway para el sistema de proyectos de investigación de la UNACH basado en microservicios	**GENERAL** Implementar una API Gateway para el sistema de proyectos de investigación de la Universidad Nacional de Chimborazo basado en microservicios	**INDEPENDIENTE** API Gateway para el sistema de proyectos de investigación.	La arquitectura de microservicios es un método de desarrollo de aplicaciones software que funciona como un conjunto de pequeños servicios que se ejecutan de manera independiente y autónoma, proporcionando una funcionalidad de negocio completa	Arquitectura	**Independiente:** • Numero de diagramas generado • Numero de esquemas diseñados
		ESPECÍFICOS • Analizar la arquitectura de microservicios para el desarrollo del API Gateway. • Diseñar un api Gateway con microservicios para el sistema de investigación de la Universidad Nacional de Chimborazo. • Evaluar el rendimiento del api Gateway utilizando el modelo de calidad FURPS.	**DEPENDIENTE** Evaluar el rendimiento del API Gateway.	Una API Gateway desacopla la interfaz del cliente de la implementación de back-end de un entorno de aplicaciones y productos de sistemas SAP.	Rendimiento	**Dependiente** Modelo de calidad de FURPS: • % eficacia • % tiempo de respuesta • % utilización de recursos

3.8. Metodología de desarrollo de Software

En el trabajo de investigación se utilizó la metodología Kanban para el desarrollo de API Gateway para el sistema de proyectos de investigación de la UNACH basado en microservicios.

3.8.1. Inicio

En este proceso se identificó al personal involucrado, por lo tanto, se definió los requerimientos para el desarrollo del proyecto.

Equipo Kanban

Según la metodología Kanban se asigna roles específicos a los asistentes del equipo, así como se detalla en la Tabla 5.

Tabla 5: Equipo Kanban

Rol	Asistentes
Product Owner	Ing. Alex Asitimbay
Kanban Máster	Ing. Alex Buñay
Team	Jacqueline Nuñez

Requerimientos funcionales

Los requerimientos funcionales se recopilaron en el departamento de Dirección de Investigación de la UNACH (ICIT), junto al Ing. Alex Asitimbay, en la Tabla 6 se detalla los requerimientos recopilados.

Tabla 6: Requerimientos funcionales

Identificación del requerimiento	**RF01**
Nombre del requerimiento	Postulación de perfil de proyectos de investigación
Descripción del requerimiento	El usuario postulará el proyecto de investigación a funciona de la investigación que se realizará.
Prioridad del requerimiento	Alta
Identificación del requerimiento	**RF02**
Nombre del requerimiento	Evaluación de proyectos de investigación
Descripción del requerimiento	El perfil de proyectos de investigación postulado se someterá a un proceso de evaluación para determinar la pertinencia de inicio del mismo.
Prioridad del requerimiento	Alta
Identificación del requerimiento	**RF03**
Nombre del requerimiento	Seguimiento de proyectos de investigación
Descripción del requerimiento	El usuario deberá presentar avances periódicos de las actividades propuestas en el proyecto de

	investigación hasta llegar al cumplimiento total de las actividades planteadas.
Prioridad del requerimiento	Alta
Identificación del requerimiento	**RF04**
Nombre del requerimiento	Cierre de proyectos
Descripción del requerimiento	Una vez cumplido el tiempo de ejecución del proyecto de investigación el usuario debe presentar la producción académica derivada del proyecto ejecutado.
Prioridad del requerimiento	Alta

Requerimientos no funcionales

Los requerimientos no funcionales describen criterios de desempeño del sistema, como la seguridad, disponibilidad, compatibilidad, rendimiento etc., en la Tabla 7 se identifica los requisitos.

Tabla 7: Requerimientos no funcionales

Requerimiento	Descripción del requerimiento	Categoría
RNF01	la capacidad del sistema para detectar, reportar y administrar errores de manera efectiva, asegurando la integridad y el funcionamiento continuo del sistema.	Control de errores
RNF02	El diseño y la funcionalidad de las interfaces que interactúan con los usuarios, con énfasis en la accesibilidad y la usabilidad, aseguran una experiencia intuitiva, consistente y adaptable a diferentes dispositivos y necesidades.	Interfaz del usuario
RFN03	Facilidad con la que el sistema puede ser mantenido y actualizado sin comprometer su funcionalidad, del sistema para mantener un nivel consistente de rendimiento y disponibilidad a lo largo del tiempo.	Confiabilidad
RFN04	Este protocolo protege de la confidencialidad e integridad de los datos mediante el uso de algoritmos criptográficos para cifrar la información sensible.	Disponibilidad
RFN5	Se enfocan en evaluar la velocidad de ejecución, la capacidad de respuesta a diferentes niveles de trabajo y la eficiencia en la utilización de recursos para garantizar un funcionamiento fluido y sin interrupciones.	Rendimiento

3.8.2. Planificación y estimación

En esta fase, se desarrolló un plan detallado para el proyecto de investigación como se muestra en la Tabla 8.

Product Backlog

Realiza un inventario detallado de las acciones planificadas para completar el proyecto. Las tareas se dividen en Historias Técnicas (HT) e Historias de Usuario (HU), cada una con una numeración específica. Además, se incluyó una estimación del esfuerzo requerido para cada tarea calificada entre 1 (menos esfuerzo) y 5 (más esfuerzo), (ver Tabla 8).

Tabla 8: Product Backlog

Ítem	Tarea	Esfuerzo
HT-01	Análisis de los requerimientos funcionales y no funcionales del proyecto de investigación.	5
HT-02	Establecer la arquitectura del elemento.	3
HT-03	Diseño de la base de datos (Modelo relacional y físico)	5
HT-04	Instalación y configuración de herramientas para el desarrollo del proyecto.	4
HU-01	Incrementar la base de datos.	4
HU-02	Incrementar la sección de dominio para el módulo.	4
HU-03	Incrementar la sección de infraestructura para el módulo.	3
HU-04	Incrementar la sección de api Gateway para el módulo.	5
HU-05	Incrementar la sección de usuario de la aplicación para el módulo.	5
HU-06	Evaluar el proyecto mediante el modelo de calidad de FURPS.	5
HU-07	Analizar los resultados obtenidos.	4
HT-05	Desplegar el sistema en un servidor.	5
HT-06	Validar el módulo de proyectos.	4

Sprint Backlog

Se trata de un conjunto de tareas seleccionadas del inventario de productos para abordar en un sprint. La Tabla 9 contiene una lista de las actividades del Sprint Backlog.

Tabla 9: Actividades del Sprint Backlog

N°	Actividades	Semanas															
		Mes 1				Mes 2				Mes 3				Mes 4			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
	SPRINT 1																
HT-01	Análisis de los requerimientos funcionales y no funcionales del proyecto de investigación.	X															
HT-02	Establecer la arquitectura del módulo.		X														
HT-03	Diseño de la base de datos (Modelo relacional y físico).			X													
HT-04	Instalación y configuración de herramientas para el desarrollo del módulo.				X												
	SPRINT 2																
HU-01	Incrementar la base de datos.					X											
HU-02	Incrementar la sección de dominio para el módulo.						X	X									
HU-03	Incrementar la sección de infraestructura para el módulo.								X								
	SPRINT 3																
HU-04	Incrementar la sección de api Gateway para el módulo.									X	X						
HU-05	Incrementar la sección de usuario de la aplicación para el módulo.										X	X					
HU-06	Evaluar el módulo mediante el modelo de calidad de FURPS.												X				
	SPRINT 4																
HU-07	Analizar los resultados obtenidos.													X			
HT-05	Desplegar el sistema en un servidor.														X	X	
HT-06	Validar el módulo de proyectos.																X

3.8.3 Diseño del sistema

Diagrama de caso de uso

Son un elemento fundamental para el desarrollo de software hacia la presentación visual de los usuarios y el sistema, al igual que debe comportarse con facilidad de identificación, requerimientos y planificación del desarrollo hacia los usuarios con el fin de relacionarse con los roles del sistema.

La Figura 6 muestra el proceso, asegurando que el perfil de proyecto sea revisado y comparado, permitiendo al analista comunicar cualquier deficiencia y áreas de mejora al usuario.

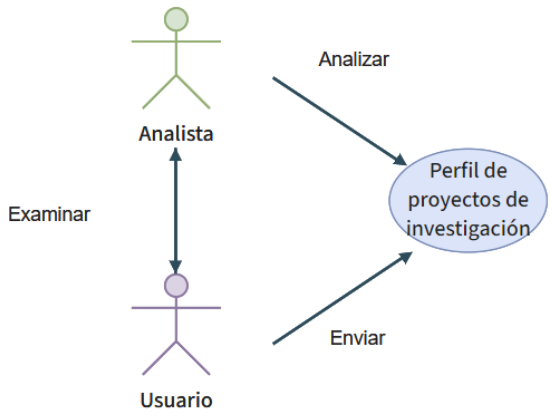


Figura 6: Casos de uso de perfil de proyectos de investigación

En la Figura 7, se muestra las actividades específicas del director a realizar en el sistema para la gestión de información de proyectos de investigación.

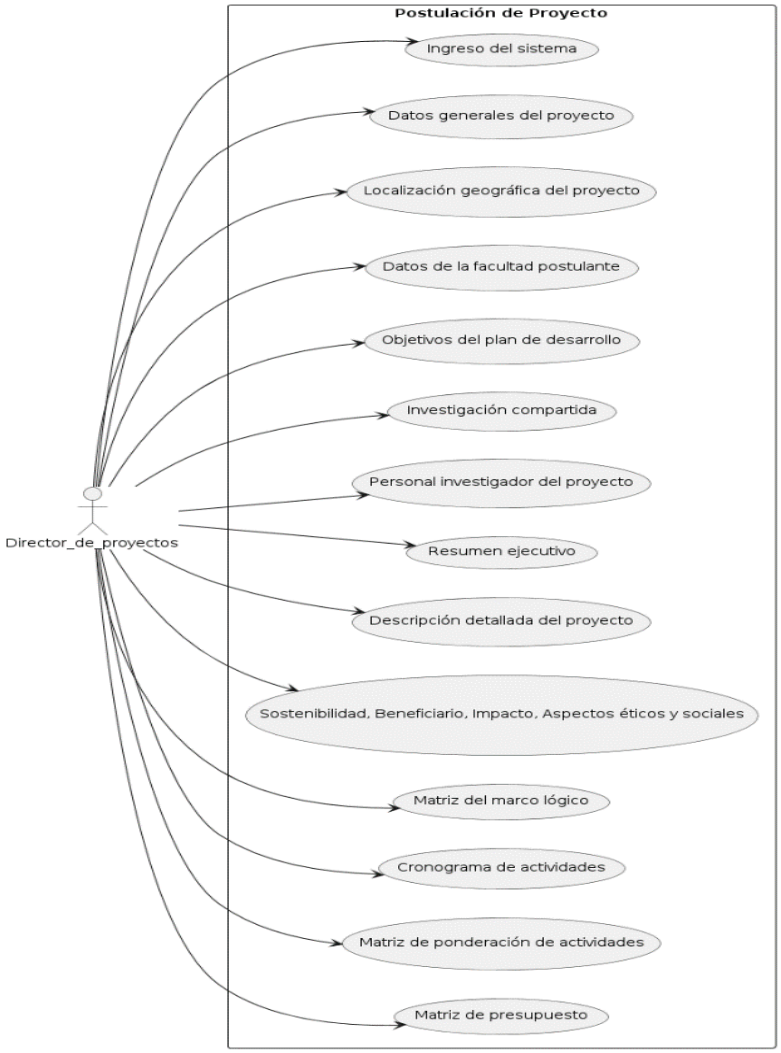


Figura 7: Casos de uso de postulación de proyectos

En la Figura 8, se muestra el proceso revisión y resolución del proyecto de investigación realizado por el departamento de investigación.

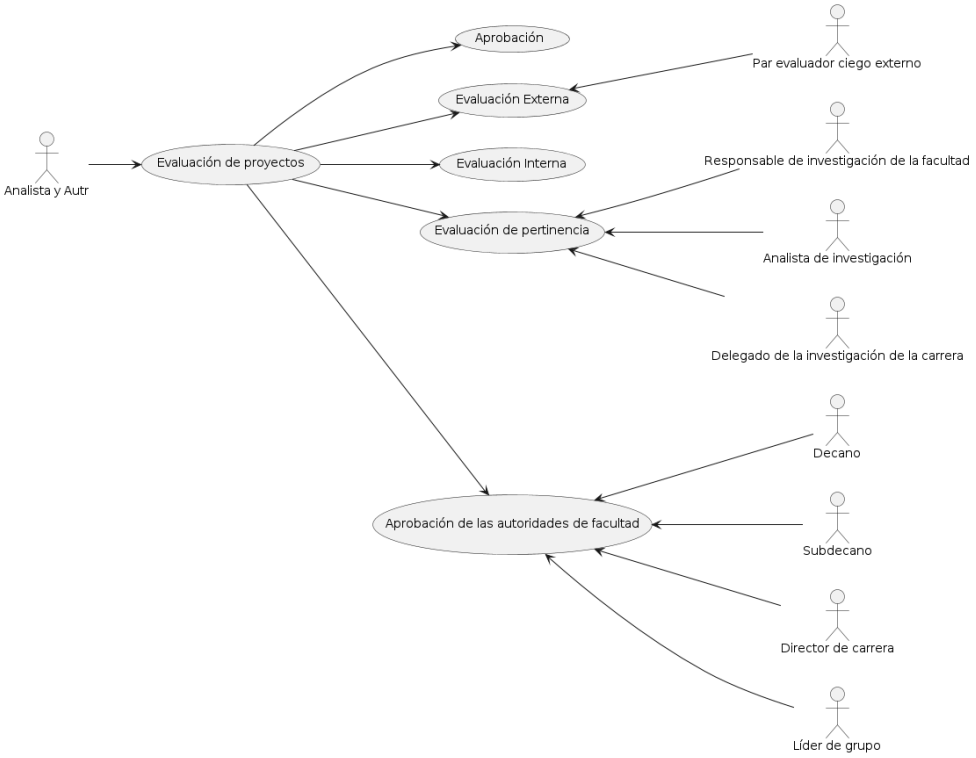


Figura 8: Casos de uso de Evaluación de proyectos

En la Figura 9, se observa el caso de uso de seguimientos de proyecto de investigación.

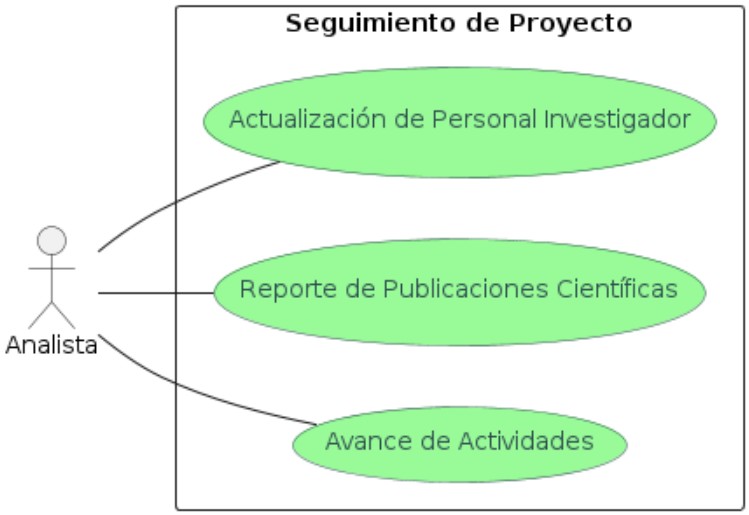


Figura 9: Caso de uso de seguimientos de proyecto

En la Figura 10, se presenta el caso de uso de cierre de proyecto de investigación.

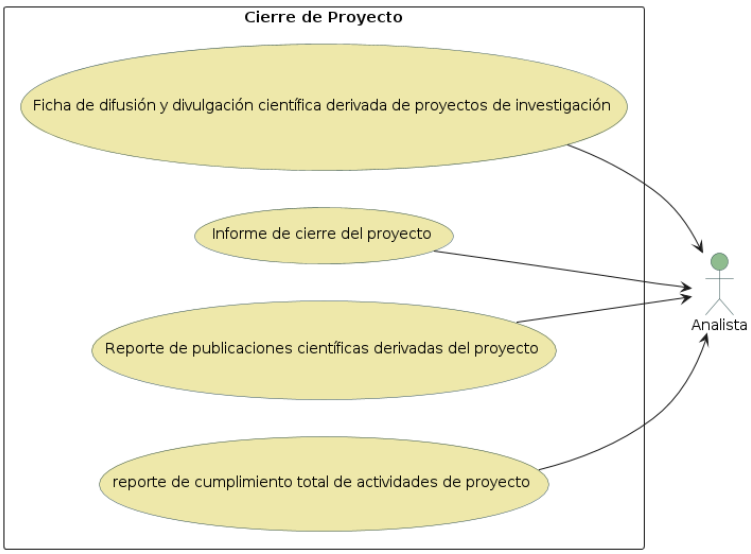


Figura 10: Caso de uso de cierre de proyecto

Diagrama relacional

Este modelo utiliza tablas para representar la información, donde cada tabla corresponde a una entidad y cada fila a una instancia específica de esa entidad. Las claves primarias y foráneas establecen las relaciones entre estas entidades, permitiendo una conexión adecuada y significativa entre los datos almacenados en diversas tablas.

La Figura 11 muestra la estructura y las relaciones importantes de la base de datos relacional que contiene el módulo de proyectos de investigación.

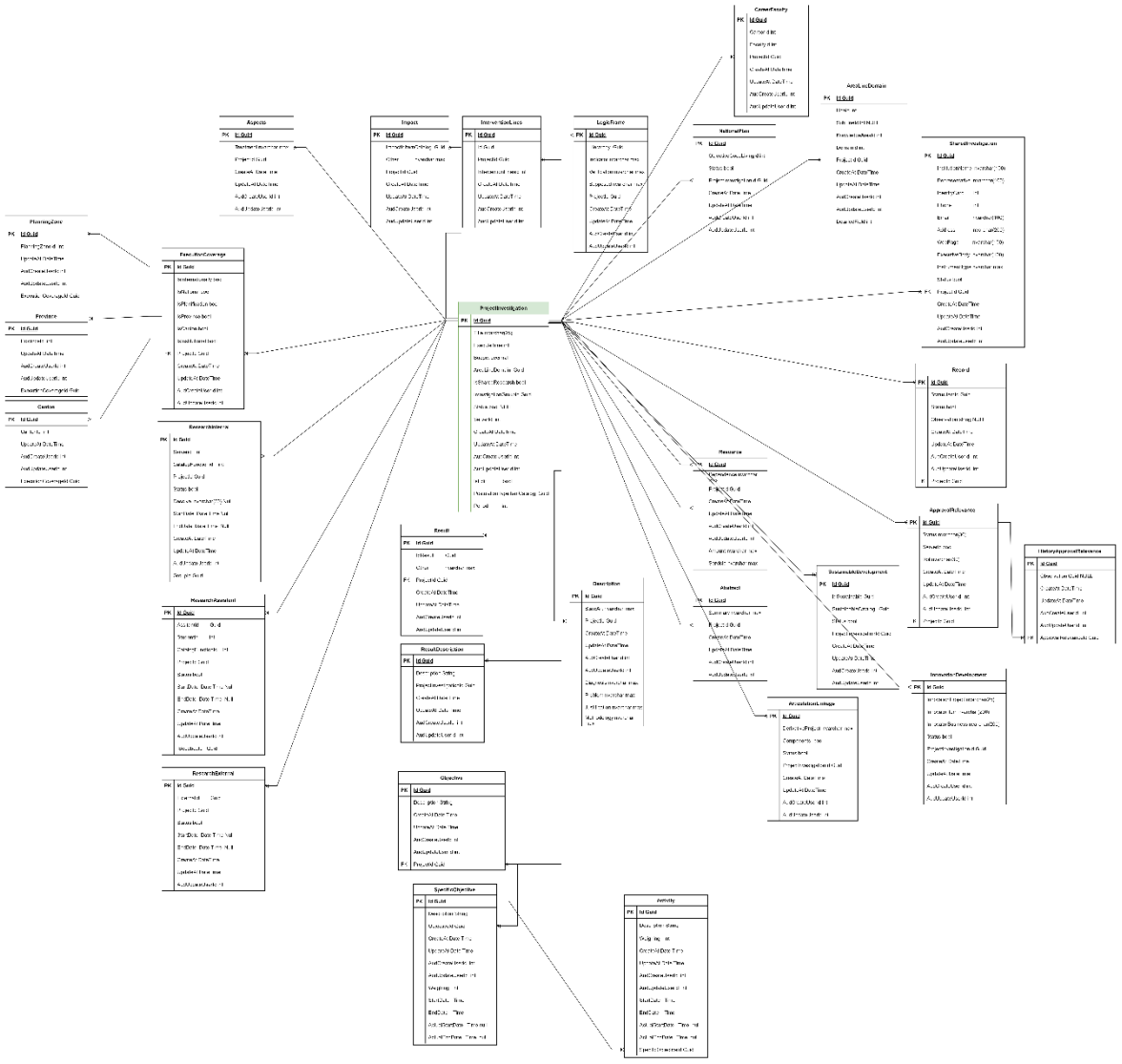


Figura 11: Diagrama Relacional

Diagrama físico de la base de datos

La visualización del diagrama físico de una base de datos proporciona una imagen detallada de cómo se traduce la estructura conceptual en términos de almacenamiento real en el sistema.

En este esquema se muestran las tablas con sus atributos, tipos de datos y las relaciones específicas entre ellas. Este tipo de diagrama es esencial para comprender la implementación técnica de la base de datos, asegurando que la estructura lógica se alinee con las necesidades de almacenamiento y rendimiento del sistema, como se visualiza en la Figura 12.

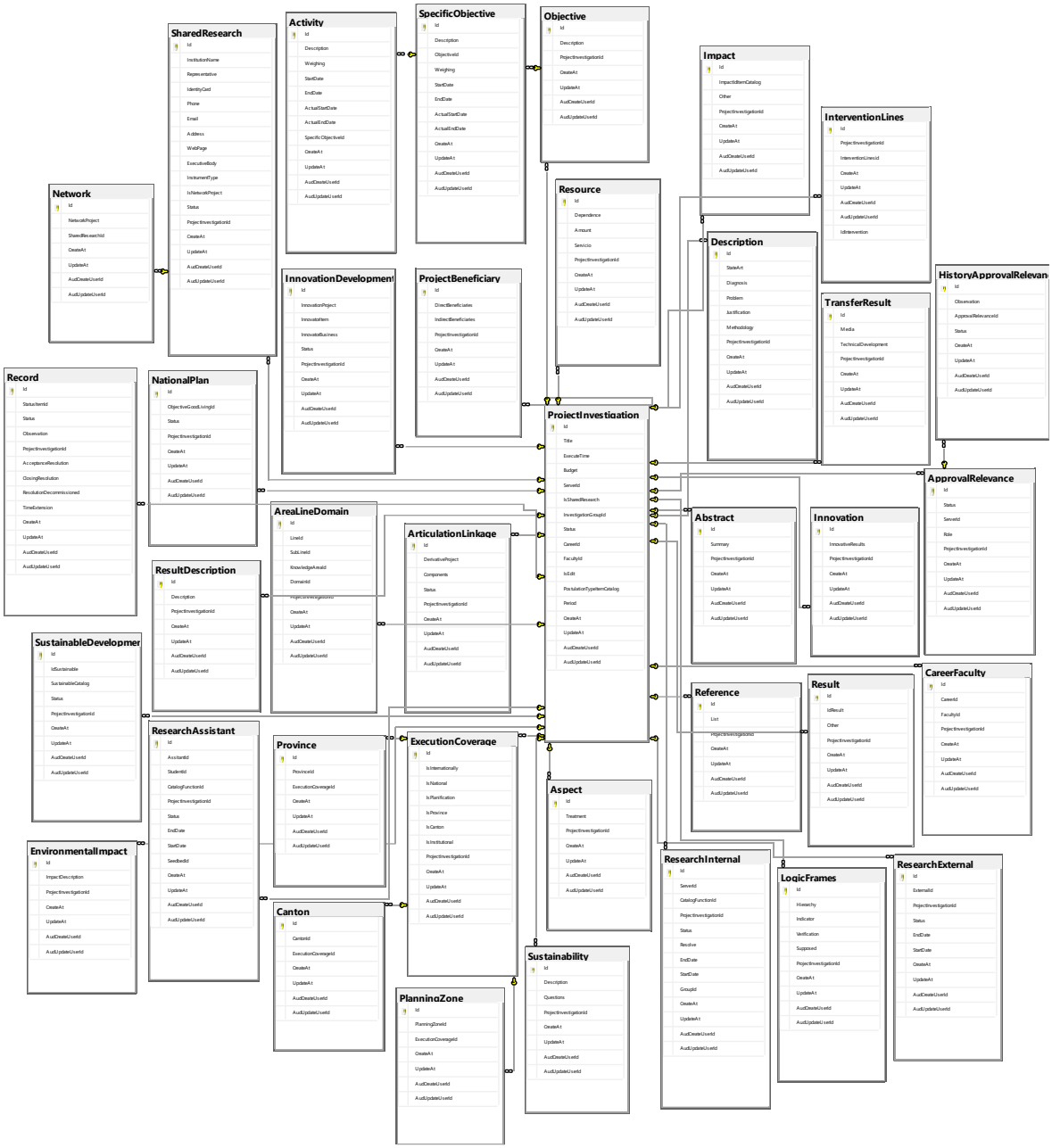


Figura 12 : Diagrama físico

Diccionario de datos

La herramienta SQL es fundamental para organizar y documentar la información sobre los datos empleados en el desarrollo de software. En la Tabla 10, se presenta la información de los proyectos almacenados en la base de datos denominada "ProjectInvestigation". En esta tabla, se detallan los atributos clave, incluyendo el nombre de cada atributo, el tipo de dato correspondiente, y una descripción que proporciona información adicional sobre cada uno de ellos.

Tabla 10: Tabla Principal del módulo de proyectos

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	Title	nvarchar (MAX)	Título del proyecto de investigación
	ExecuteTime	int	Tiempo de ejecución del proyecto
	Budget	decimal (18,2)	Presupuesto
	ServerId	int	Identificación del servidor
	IsSharedResearch	bit	Verificar si es una investigación compartida
	InvestigationGroupId	uniqueidentifier	Identificador de grupos de investigación
null	Estatus	bit	Estado del proyecto que puede ser 0 o 1 Acepta nulos
	CreateAt	datatime2(7)	Fecha de creación
	UpdateAt	datatime2(7)	Fecha de actualización
	AudCreateUserId	int	Identificación de Usuario de Creación de Auditoría
	AudUpdateUserId	int	Identificación de Usuario de Actualización de Auditoría
	IsEdit	bit	Verificar si existen cambios en el proyecto
	Period	int	Periodo del proyecto
	FacultyId	int	Identificación de la Facultad
	CareerId	int	Identificación de la Carrera
	PostulationTypeItemCatalog	uniqueidentifier	Identificador de tipo de Postulación de un proyecto

Diagrama de la arquitectura basada en microservicios

Se inició el desarrollo de la aplicación bajo la arquitectura de microservicios, utilizando JetBrains Rider como IDE.

Para implementar la arquitectura de microservicios, que abarca tanto el componente de proyectos como los microservicios asociados, ofreciendo una perspectiva minuciosa de cómo están conectados y operan dentro del sistema, tal como se evidencia en la Figura 13.

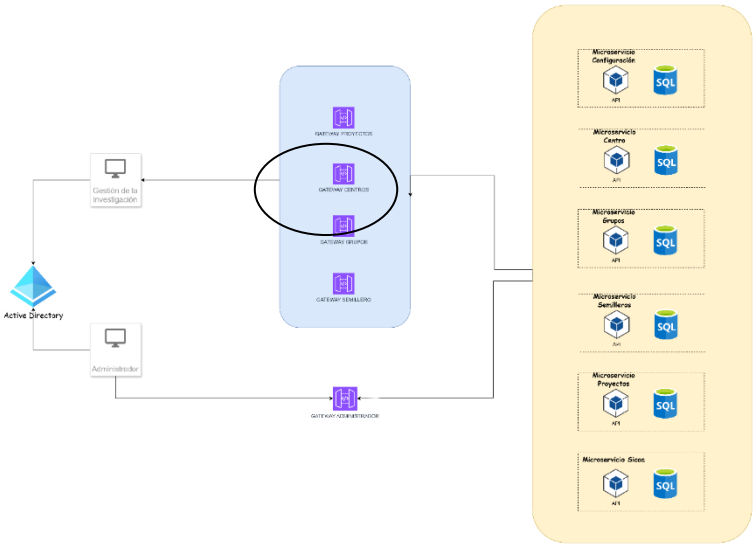


Figura 13: Arquitectura basada en microservicios

El microservicio se desarrolló utilizando .Net 7 en combinación con sus frameworks principales, ASP.Net Core y Entity Framework Core, dentro del entorno de desarrollo Jetbrains Rider. Se empleó una base de datos SQL denominada 'Proyecto Investigación' para almacenar los datos pertinentes.

Además, para establecer las API REST y ofrecer una explicación detallada de las funciones proporcionadas por el servicio para cada entidad, como en la Figura 14 se demuestra el microservicio lógico para el módulo de proyectos de Investigación.

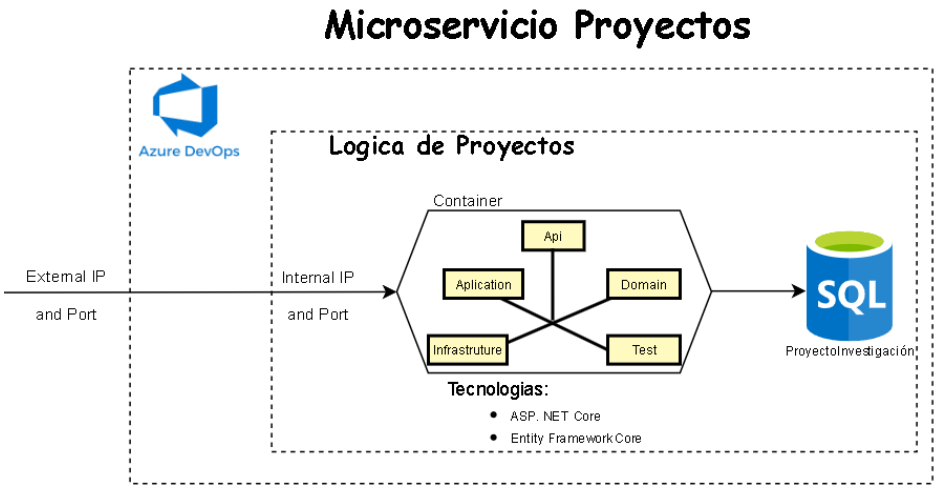


Figura 14 : Diseño lógico del microservicio Proyectos

Fuente: (Pilamunga, 2024)

La Figura 15, muestra una interfaz que le permite gestionar y clonar proyectos de investigación en un sistema de control de versiones, ofreciéndole herramientas para el desarrollo colaborativo y la administración del código fuente.

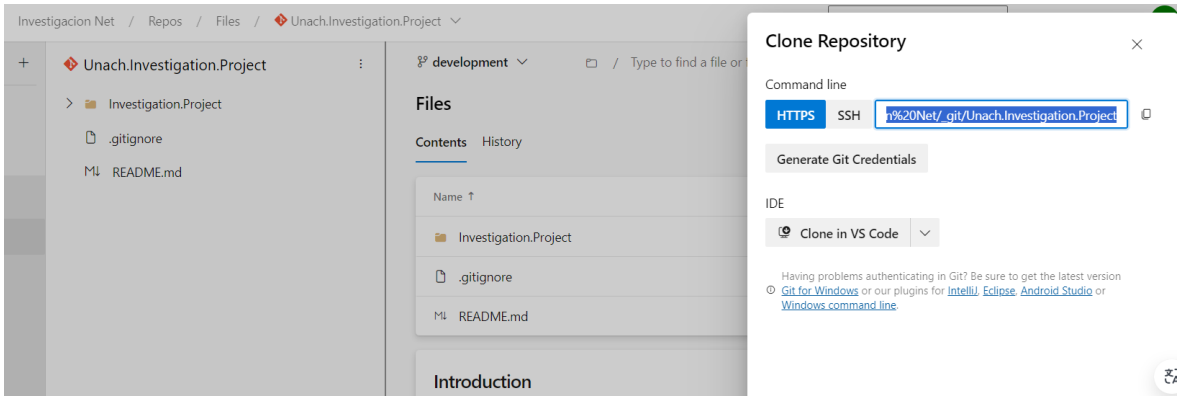


Figura 15: Clonación del proyecto

La descarga de los paquetes NuGet personalizados de Azure DevOps, suministrados por los expertos del departamento de investigación, con el fin de construir el módulo. Estos paquetes fueron ajustados de manera específica para cubrir sus requerimientos y el entorno de trabajo, simplificando la ejecución del proyecto, tal como se presenta en la Figura 16.

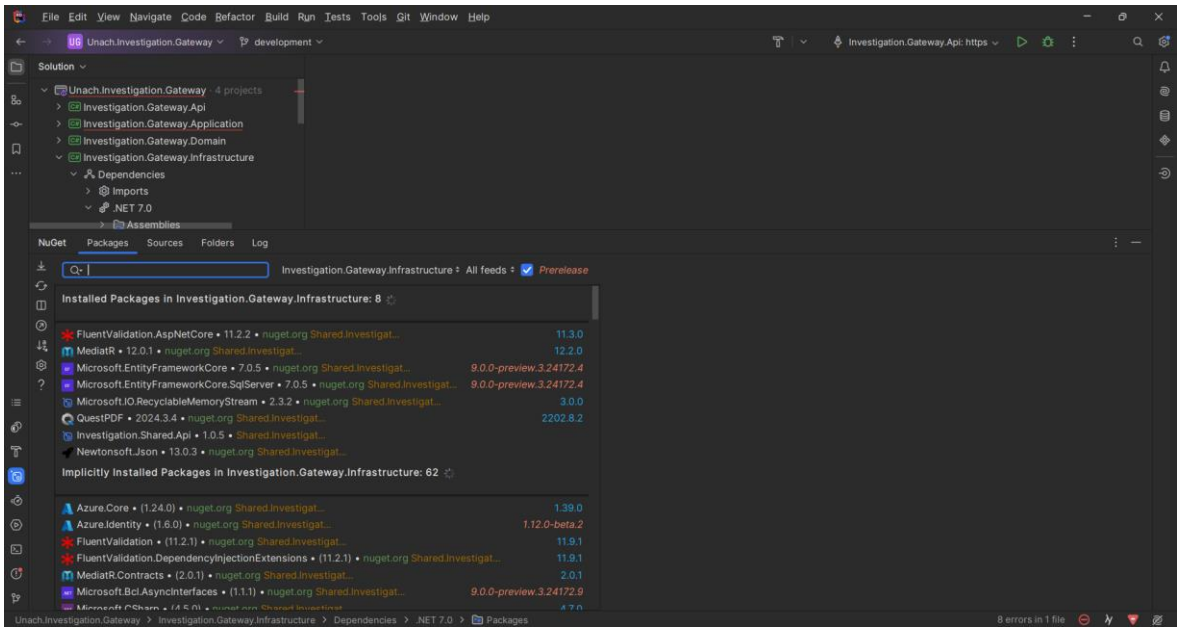


Figura 16: Paquetes NuGet

El archivo appsettings. json en la capa del API Gateway es fundamental para configurar y gestionar diversos aspectos del comportamiento del gateway, desde enrutamiento y limitación de tasa hasta autenticación y como se muestra en la Figura 17.

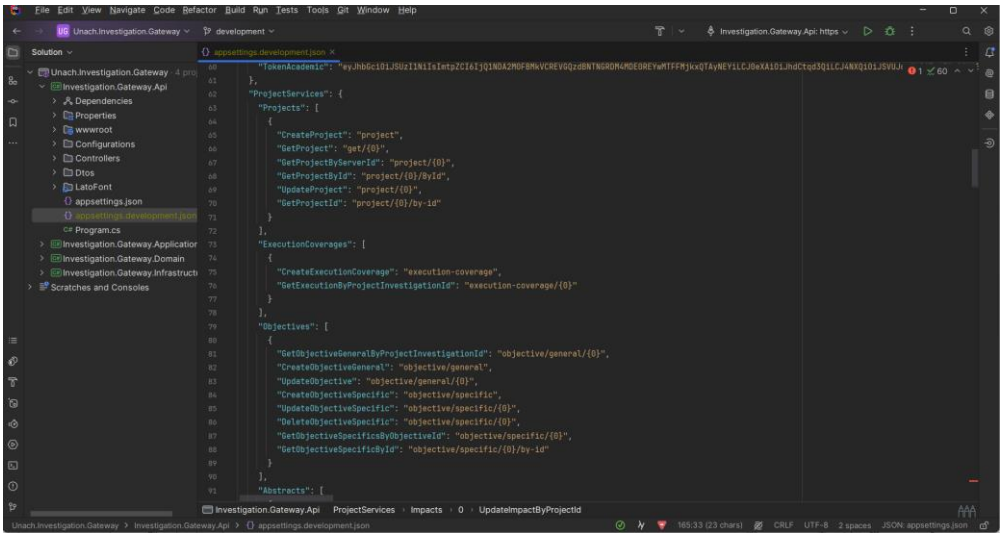


Figura 17:Archivo appsettings. Json

En la sección de la capa de dominio es una parte crucial de cualquier arquitectura de software bien diseñada, incluida la de un API Gateway. En esta capa, se encuentra la lógica de negocio central de la aplicación, los modelos de dominio y las reglas que rigen cómo los datos pueden ser manipulados y procesados, implementando las entidades de cada uno de los servicios a consumir (ver Figura18).

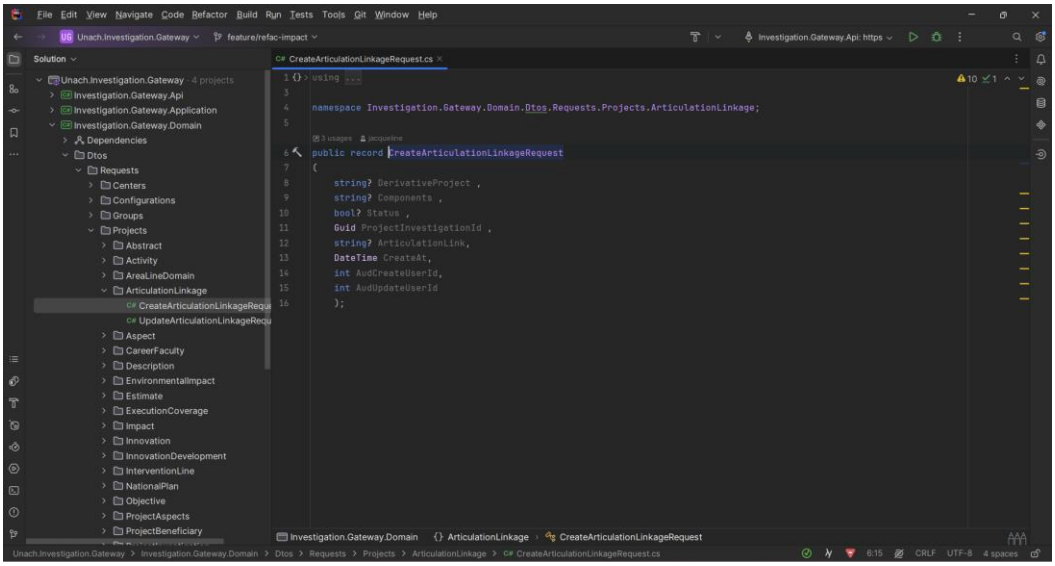


Figura 18: Capa de dominio de entidades

Estos servicios actúan como intermediarios entre los controladores (que manejan las solicitudes HTTP) y los repositorios (que interactúan con la base de datos), asegurando que la lógica de negocio se aplique de manera coherente y centralizada de cada servicio que contiene proyectos (ver Figura 19).

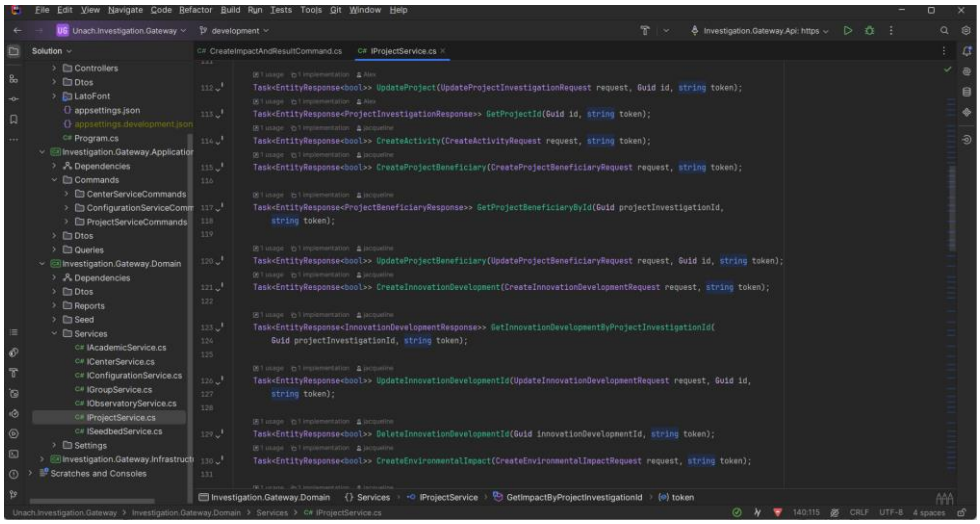


Figura 19 : Servicios de solicitudes HTTP

Los controladores definen los endpoints de la API que pueden ser invocados por los clientes los cuales manejan las solicitudes HTTP (GET, POST, PUT, DELETE) y llaman a los servicios correspondientes en la capa de lógica de negocio, realizando las respuestas en formatos como JSON o XML, como se muestra en la Figura 20.

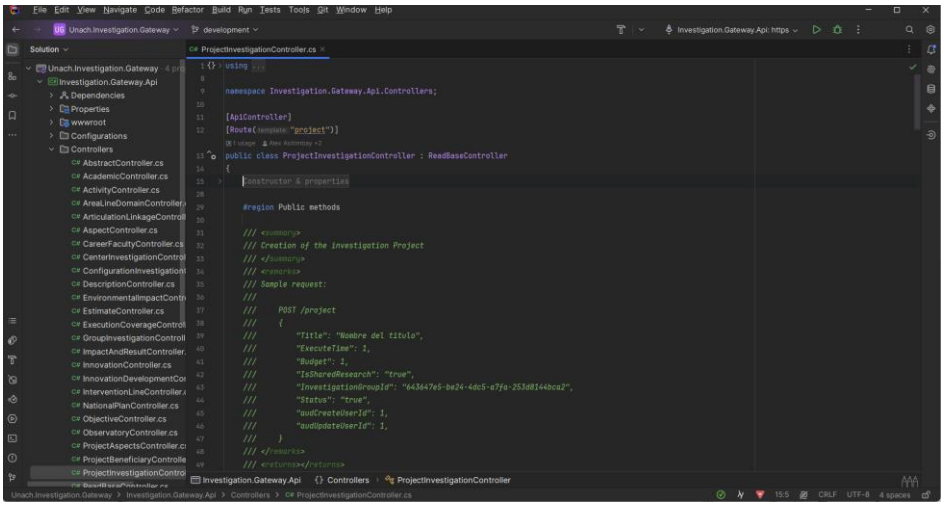


Figura 20: Controladores definen los endpoints de la API

Lanzamiento

En la figura 21, representa la plataforma digital para la gestión de actividades académicas o de investigación en la UNACH, con un enfoque en el perfil y las opciones disponibles para un usuario específico.

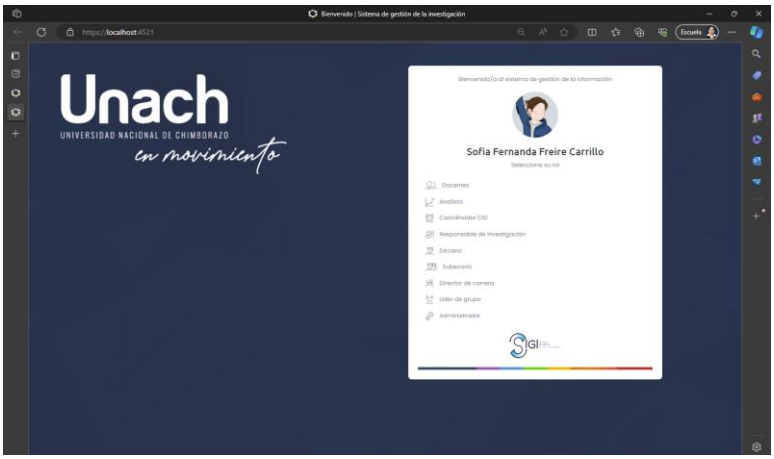


Figura 21: Inicio de Modulo

Dashboard de inicio del sistema de gestión de investigación, para facilitar la navegación y gestión de diferentes tipos de iniciativas de investigación dentro de la universidad, (Figura 22).

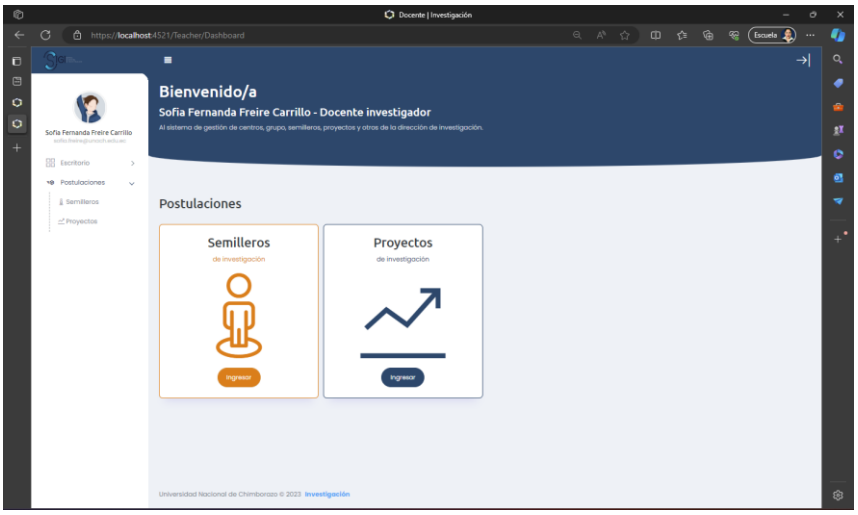


Figura 22: Vista del módulo Proyectos

A continuación, se mostrará la vista de la postulación de un proyecto con sus respectivos datos (Figura 23).

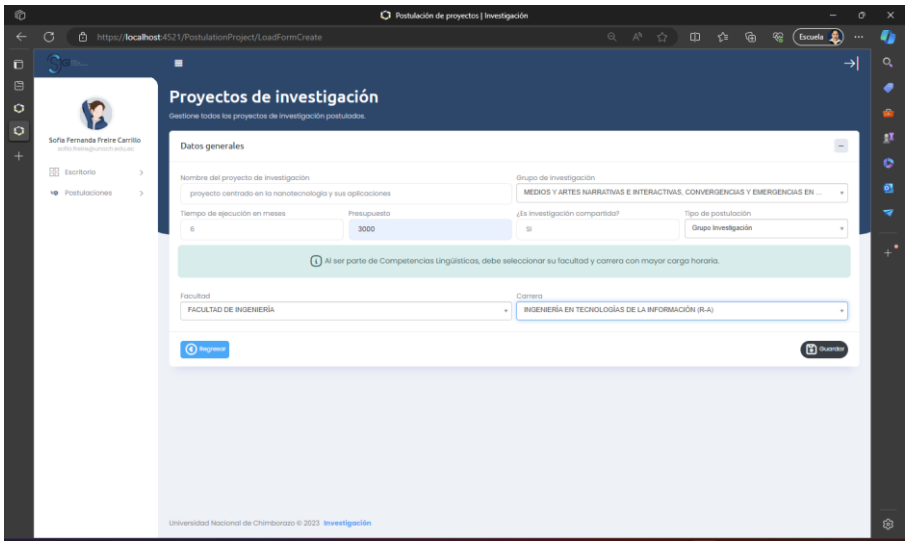


Figura 23: Postulación de un proyecto

La interfaz es clara y organizada, facilitando el acceso y la gestión de la información de los proyectos de investigación, con un menú desplegable que ofrece varias opciones de gestión y configuración, incluyendo datos generales, vinculación e innovación, objetivos, asignación estratégica e institucional, y más. También se proporciona un botón para iniciar una nueva postulación de proyecto, como se muestra en la Figura 24.

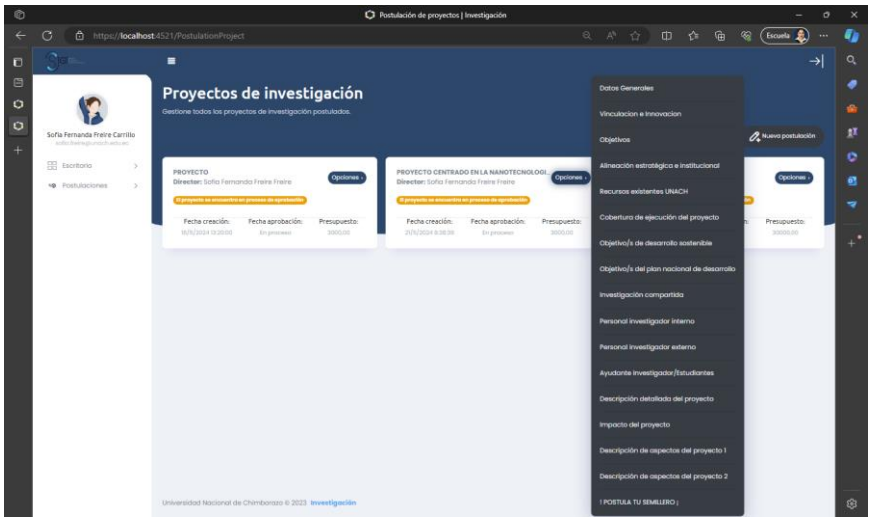


Figura 24: Listado de requisitos para la nueva postulación

En el Anexo 1, se muestra las ventanas desarrolladas para el sistema de investigación.

CAPÍTULO IV. RESULTADOS Y DISCUSIÓN

4.1. Pruebas

Se llevó a cabo una evaluación completa utilizando la herramienta Blazer Meter para abordar el tercer objetivo, que consistía en evaluar el rendimiento del módulo de proyecto. Se creó un escenario de prueba en el que se ejecutaron 300 solicitudes simultáneas para una variedad de propósitos en un lapso de un segundo. Estas pruebas se llevaron a cabo en una configuración estándar de una computadora para garantizar condiciones controladas pero representativas del entorno real de uso.

Se recopilaron datos minuciosamente del desempeño después de completar las pruebas, estos datos se analizaron minuciosamente identificando cualquier problema, anomalía o área de mejora potencial en el módulo de proyectos. Los resultados se mostraron visualmente a través de gráficos estadísticos comprensibles como se evidencia en la Figura 25.

Figura 25: Resultados generales

4.1.1. Ejecución de pruebas

La gráfica 26 muestra el comportamiento del sistema con una cantidad cada vez mayor de usuarios virtuales. Se puede ver cómo el tiempo de respuesta y los hits totales varían con la carga y cómo aumentan los errores en momentos cruciales. La identificación y la resolución de problemas de rendimiento del sistema dependen de esta información (Figura 26).

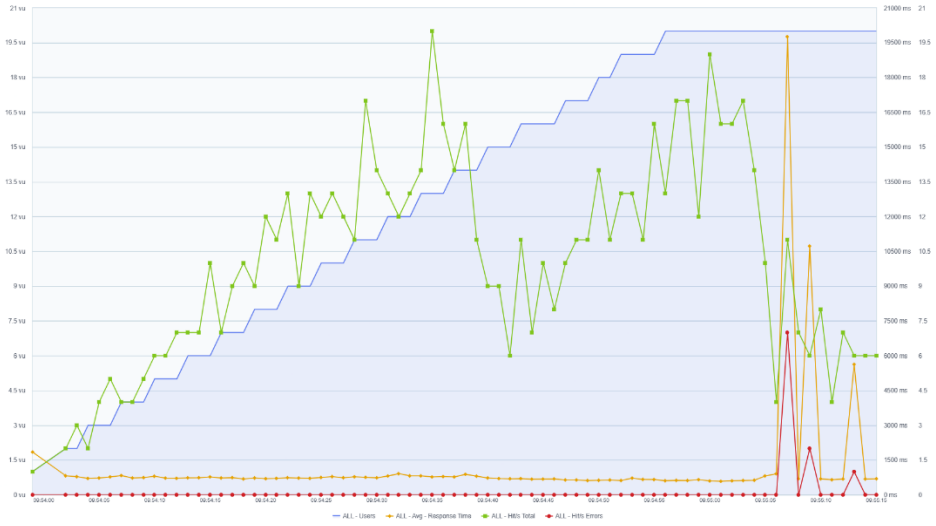


Figura 26: Gráfica de rendimiento

El comportamiento de una prueba de carga con 150 usuarios virtuales donde los hits por segundo varían periódicamente y la cantidad de errores es casi nula (Figura 27), visualizando claramente el rendimiento y la estabilidad del sistema bajo las condiciones de prueba.

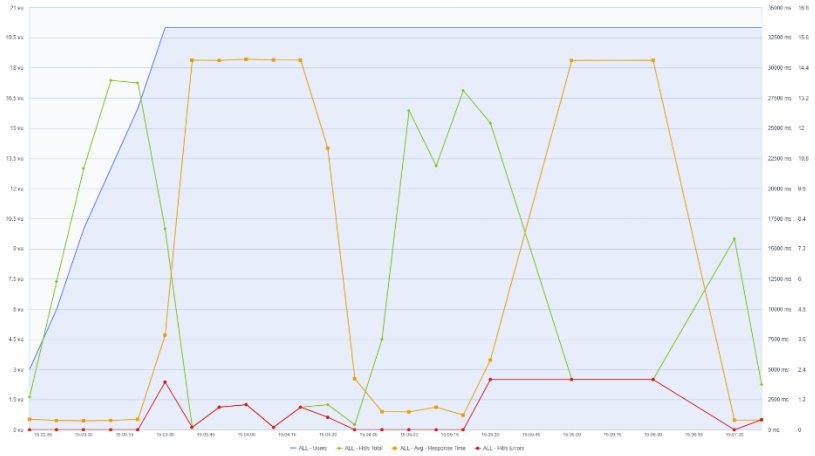


Figura 27: Gráfica de carga con 25 usuarios virtuales

Como se muestra la Figura 28, el sistema funciona correctamente, la memoria permanece estable, indica que no hay fugas o sobrecargas en la misma, no hay señales persistentes de sobrecarga y el sistema maneja adecuadamente las fluctuaciones en el uso de CPU, las conexiones y el I/O de red, estos patrones muestran un buen rendimiento del sistema porque

puede adaptarse a diversas cargas de trabajo y responder de manera efectiva a picos de demanda.

Figura 28: Monitoreo de Rendimiento del Sistema

4.2. Interpretación de resultados

El proyecto de investigación de gestión de la UNACH ha completado la aplicación web, desarrollando un API Gateway del sistema de proyectos de investigación de la UNACH basado en microservicios. Para estas evaluaciones, se utilizó la herramienta Blazer Meter (ver Anexo 2), se describen las particularidades y metodologías de las pruebas realizadas.

En la Tabla 11, se detalla las pruebas realizadas en cada proceso, reflejando 100 casos examinados.

Tabla 11: Descripción de procesos evaluados

Procesos	Cantidad de Pruebas
Registro de Perfil de Proyectos	100
Estado del proyecto	100
Matriz de Proyectos	100
Cobertura de ejecución del Proyecto	100

En la Figura 29, se muestra los resultados de las pruebas del sistema mediante un gráfico de pastel, destacando que el 100% de las pruebas realizadas resultaron exitosas.

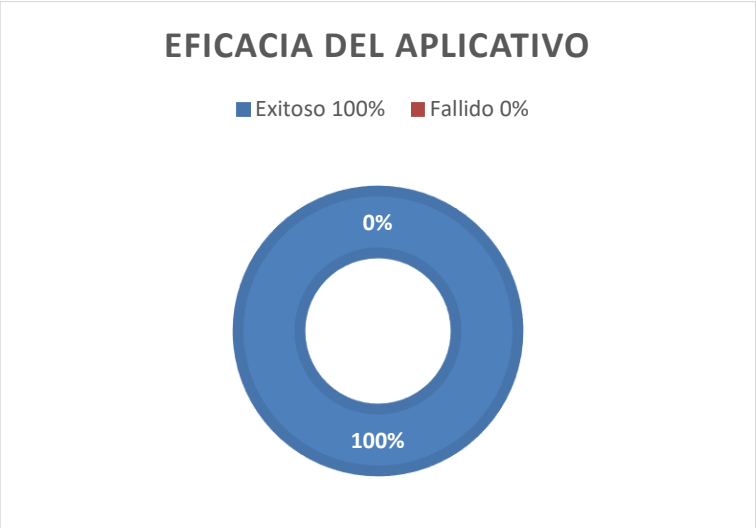


Figura 29 : Eficacia del Aplicativo

4.2.1. Valoración de indicadores

Cada indicador evaluado se representa con gráficos estadísticos en esta sección, creando un apartado específico para la valoración de estos parámetros. Cada gráfico ilustra el rendimiento del sistema según los indicadores establecidos.

Esta metodología ha permitido una comprensión más profunda y detallada de la evaluación, proporcionando información relevante sobre el desempeño del sistema en cada aspecto considerado.

Eficacia

La Tabla 12 y Figura 30, muestran el porcentaje de eficacia del módulo en relación con las 300 pruebas ejecutadas, este indicador revela que la totalidad de las solicitudes, representadas en los 300 casos analizados, lograron un rendimiento del 100%.

Tabla 12: Indicador de Eficacia

Parámetro	Indicadores	Petición
Eficacia	Requerimientos exitosos	100%
	Requerimientos fallidos	0%

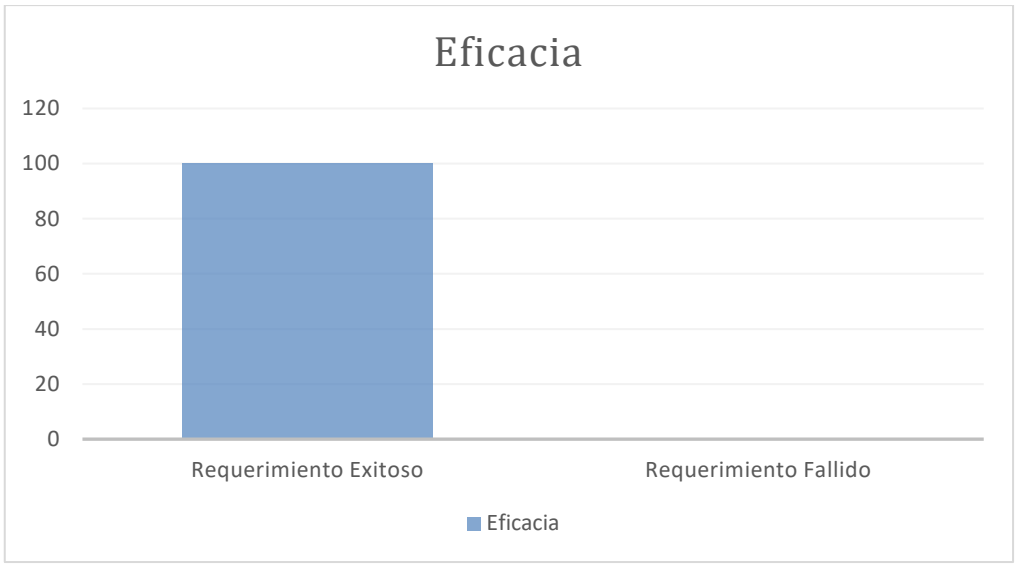


Figura 30 : Requerimiento Eficacia

Tiempo de Respuesta

En la Figura 31 se representa gráficamente los resultados obtenidos mediante la herramienta Blazer Mater, ofreciendo una visualización detallada de los datos recolectados para calcular este indicador, el tiempo de respuesta registrado en el gráfico es de un rango de 0 ms hasta un máximo de 5000 milisegundos establecidos por Blazer Meter.

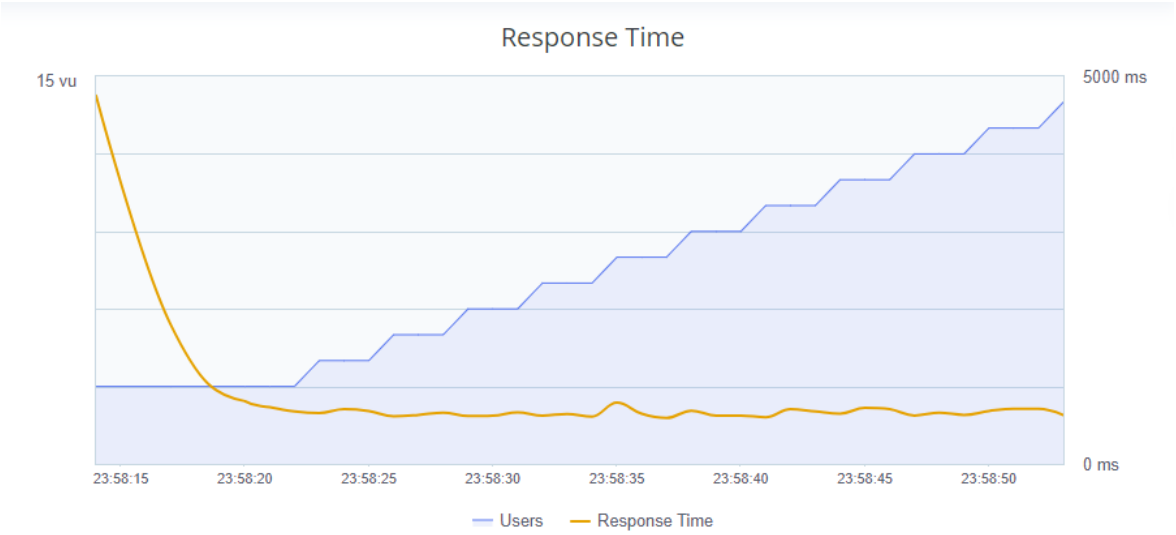


Figura 31: Tiempo de respuesta

Utilización de recursos

Evaluar el uso de recursos es fundamental para comprender como un programa o sistema afecta el rendimiento general, garantizando una operación eficiente y equilibrada del sistema (ver Anexo 3).

La Tabla 13 presenta los resultados específicos del uso de recursos, divididos en porcentajes para cada parámetro.

Tabla 13: Resultados de la utilización de recursos

Parámetro	Indicador	%Consumo
Uso de Recursos	Uso de Disco duro	1%
	Uso del CPU	2%
	Uso Memoria RAM	37%

CPU

De acuerdo con los datos mostrado en la Tabla 13, la aplicación empleó en promedio el 2% del CPU durante las pruebas.

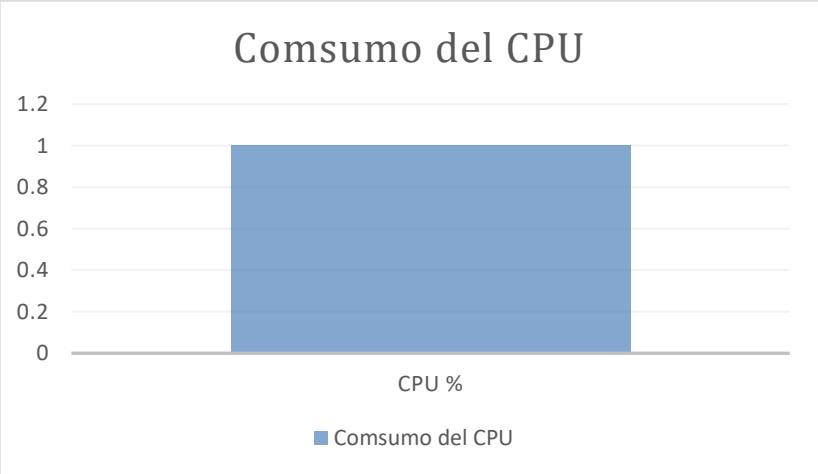


Figura 32 : Consuno del CPU

Memoria

De acuerdo con los datos mostrado en la Tabla 13, la aplicación empleó un promedio del 37% de uso de la memoria durante las pruebas.

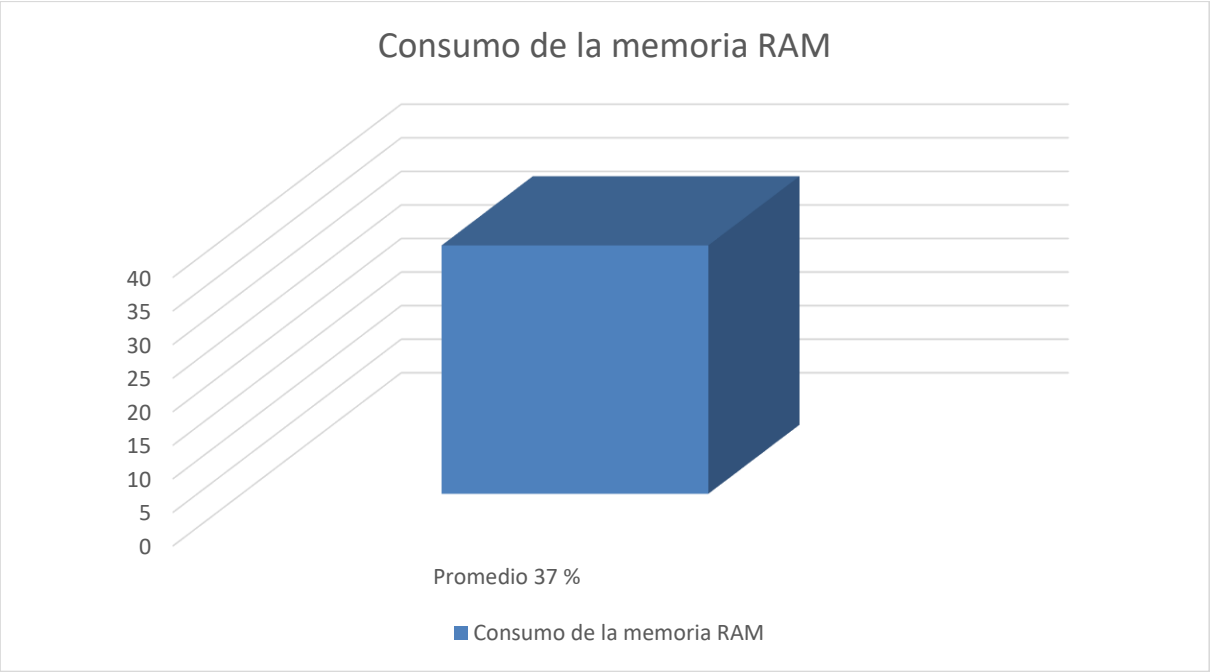


Figura 33: Consumo de Memoria RAM

Disco Duro

De acuerdo con los datos mostrado en la Tabla 13, la aplicación empleó un promedio del 1% de uso del disco duro durante las pruebas.

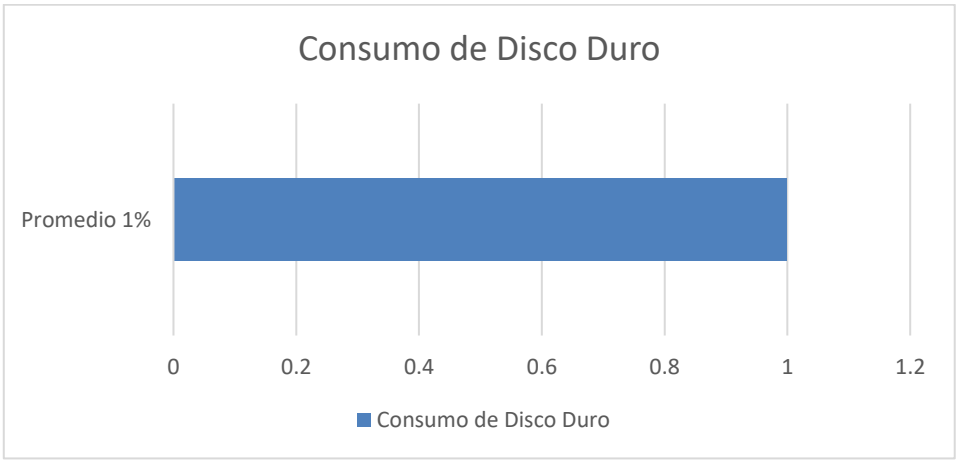


Figura 34: Consuno de Disco Duro

4.2.2. Valores obtenidos del estudio en base del modelo de FURPS

En la Tabla 14, se muestra como el módulo de proyectos de investigación exhibe un rendimiento eficiente y satisface los estándares del modelo de calidad FURPS.

Tabla 14 : Valores de estudio mediante el modelo de calidad FURPS

Parámetros	Valores Obtenidos	Modelo FURPS
Tiempo de Respuesta	5000ms	5s
Utilización de Recursos	40%	25%
Eficacia	100%	100%

En la Tabla 15, se presenta el estudio comparativo de tres solicitudes: primera realizada con 100 muestras, segunda con 200 muestras y tercera con 300 muestras.

Tabla 15 : Comparación de solicitudes

Parámetro	Primera Solicitud (100 muestras)	Segunda Solicitud (200 muestras)	Tercera Solicitud (300 muestras)
Tiempo promedio (ms)	2771	3462	4934
Tiempo Min Resp (ms)	1735	2252	4366
Tiempo Max Resp (ms)	3779	4533	5439
Error %	0.00%	0.00%	0.00%
Rendimiento	26.2/sec	43.6/sec	54.4/sec
Tamaño promedio	249.0	249.0	249.0

4.2. Discusión

La integración del API Gateway en el sistema de gestión de proyectos de investigación de la UNACH ha demostrado ser una solución efectiva para mejorar tanto el rendimiento como la escalabilidad del sistema. Los resultados obtenidos validan la adopción de arquitecturas de microservicios, proporcionando una base sólida para futuras mejoras y expansiones del sistema.

La arquitectura de microservicios se ha consolidado como una metodología eficaz para el desarrollo de aplicaciones complejas. Al descomponer una aplicación monolítica en componentes más pequeños y autónomos, cada microservicio se puede desarrollar, desplegar y escalar de manera independiente. Este enfoque permite una mayor agilidad en el desarrollo y una respuesta más rápida a los cambios y mejoras necesarias (IT, 2020).

Netflix es otro caso de estudio relevante. La compañía adoptó una arquitectura de microservicios junto con su propio API Gateway, Zuul permitió gestionar de manera eficiente la distribución del tráfico y aplicar políticas de seguridad y enrutamiento personalizado (Blog, 2026).

En el ámbito académico, la Universidad de Stanford implementó una arquitectura de microservicios y un API Gateway para mejorar la gestión de sus sistemas de información. Este enfoque permitió una integración más eficiente de diversas aplicaciones y servicios, facilitando su uso tanto por estudiantes como por personal administrativo (Services, 2021).

De manera similar, Guanoluisa y Mamani encontraron que la implementación de microservicios en entornos académicos no solo mejora la eficiencia operativa, sino también la calidad del software. Sus evaluaciones de calidad utilizando modelos similares a FURPS

mostraron aumentos en todas las métricas de calidad, con mejoras especialmente notables en la fiabilidad y el rendimiento del sistema, en línea con los resultados obtenidos en la UNACH (Tonato, 2024).

CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

Se concluye que la arquitectura de microservicios ofrece una serie de beneficios significativos en comparación con los enfoques monolíticos, tales como escalabilidad, modularidad y capacidad de despliegue independiente. Esta arquitectura permite una mayor flexibilidad y agilidad en el desarrollo y mantenimiento del API Gateway, ya que cada servicio puede ser desarrollado, implementado y actualizado de manera independiente sin afectar a otros componentes del sistema

El monitoreo del rendimiento del sistema demostró una estabilidad adecuada en cuanto a memoria, CPU, conexiones y I/O de red, indica una capacidad de adaptación a diferentes cargas de trabajo.

Los resultados obtenidos a través de la evaluación con el modelo de calidad FURPS indican que el sistema desarrollado cumple con los estándares de rendimiento establecidos, demostrando una eficacia del 100% y un tiempo de respuesta por debajo de los límites establecidos, lo que garantiza un alto nivel de calidad en la prestación de servicios de investigación en la UNACH.

5.2. Recomendaciones

Dado que la arquitectura de microservicios ha demostrado ser beneficiosa en términos de escalabilidad, modularidad y despliegue independiente, se recomienda seguir promoviendo la adopción de esta arquitectura en otros sistemas y proyectos de la UNACH. Es aconsejable establecer una estrategia de gestión de microservicios a largo plazo, incorporando herramientas para la orquestación, como Kubernetes, y sistemas de observabilidad que mejoren el monitoreo y la identificación de problemas de manera más proactiva.

A pesar del éxito en la implementación del API Gateway, se recomienda optimizar continuamente las rutas y políticas de tráfico a medida que el sistema crece y más microservicios se añaden.

Aunque el monitoreo del sistema ha demostrado estabilidad en el uso de recursos como memoria, CPU y conexiones, se recomienda realizar evaluaciones periódicas de rendimiento bajo diferentes escenarios de carga. Esto permitirá garantizar que el sistema mantenga su eficacia a medida que se incrementen los usuarios y la complejidad de las operaciones. También sería conveniente establecer mecanismos de escalado automático (auto-scaling) para adaptarse de manera más dinámica a cambios en la demanda de recursos.

Los resultados positivos obtenidos con la evaluación del modelo FURPS sugieren que el sistema está funcionando dentro de los parámetros de calidad esperados. Sin embargo, para mantener estos altos estándares de calidad, se recomienda realizar pruebas de carga y estrés de manera regular y con escenarios más variados. Además, se podrían implementar mejoras en la usabilidad del sistema para facilitar la interacción con el API Gateway tanto para desarrolladores como para usuarios finales, asegurando una experiencia de usuario fluida.

BIBLIOGRÁFIA

Alvarado, E. A. (s.f.).

Alvarado, E. A. (2023). Obtenido de http://dspace.unach.edu.ec/handle/51000/10247

Alvarado, E. A. (2023). *Aplicación web para el servicio de trámites académicos de la UNACH usando una arquitectura basada en microservicios.* Obtenido de http://dspace.unach.edu.ec/handle/51000/10247

Aula21. (2023). Obtenido de https://www.cursosaula21.com/que-es-net/

AWS. (s.f.). Obtenido de chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://docs.aws.amazon.com/es_es/prescriptive-guidance/latest/hexagonal-architectures/hexagonal-architectures.pdf

Blog, N. T. (2026). *Implementación de microservicios y Zuul en la gestión de tráfico y seguridad.* Obtenido de The API Gateway to the Cloud: https://netflixtechblog.com/zuul-edge-service-in-the-cloud-ab3af5be08ee

Brutti, F. (07 de 05 de 2023). *¿Qué es arquitectura hexagonal en programación?* Obtenido de https://thepower.education/blog/que-es-arquitectura-hexagonal-en-programacion

Chicaiza Rios, Diego Fernando. (03 de 2020). *Universidad Tecnica Salesiana*. Obtenido de https://dspace.ups.edu.ec/handle/123456789/18532

Cockburn. (19 de 06 de 2008). *Hexagonal Architecture*.

Coppola, M. (10 de 05 de 2023). Obtenido de https://blog.hubspot.es/website/que-es-arquitectura-hexagonal

Coppola, M. (s.f.). *HubSpot*. Obtenido de https://blog.hubspot.es/website/que-es-arquitectura-hexagonal

Douglas et al. (8 de 10 de 2022). Obtenido de https://devblogs.microsoft.com/dotnet/announcing-dotnet-7/

Hernandez, Y. (12 de 12 de 2022). *Dongee*. Obtenido de https://www.dongee.com/tutoriales/que-es-devops-de-microsoft/

IBM. (s.f.). *Microservisios* . Obtenido de https://www.ibm.com/mx-es/topics/microservices

Intel Corporation. (2021). *Microservicios y Arquitectura de microservicios*. Obtenido de https://www.intel.la/content/www/xl/es/cloud-computing/microservices.html

Intel Corporation. (s.f.). *Microservicos* .

IT, S. U. (2020). *Implementación de arquitectura de microservicios en sistemas académicos*. Obtenido de https://uit.stanford.edu/projects/microservices

Martinez, E. (26 de 01 de 2024). *Aodatacloud* . Obtenido de Azure DevOps: La herramienta esencial para el desarrollo de software: https://aodatacloud.es/blog/que-es-azure-devops-y-para-que-sirve/

Martins, J. (2024 de 01 de 19). *Asana*. Obtenido de https://asana.com/es/resources/what-is-kanban

Método Kanban con Azure DevOps. (05 de 03 de 2019). Obtenido de https://www.developerro.com/2019/03/05/azure-devops-kanban-ii/

microsoft. (s.f.). *Automatizar implementaciones de API con APIOps*. Obtenido de https://learn.microsoft.com/es-es/azure/architecture/example-scenario/devops/automated-api-deployments-apiops

Microsoft Build. (11 de 04 de 2024). Obtenido de https://learn.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/

Novoseltseva, E. (20 de 04 de 2020). *¿Qué Es La Arquitectura Hexagonal? Definición Y Ejemplos*. Obtenido de https://apiumhub.com/es/tech-blog-barcelona/arquitectura-hexagonal/#:~:text=La%20arquitectura%20hexagonal%20es%20un,domain%20est%C3%A1%20construido%20sobre%20ello.

Packard, H., & Robert, G. (1987). *MODELO DE CALIDAD FURPS*. Obtenido de https://modelos-de-evaluacion-red-grupo9.fandom.com/wiki/MODELO_DE_CALIDAD_FURPS

Palladino, M. (2016). *Microservicios & API gateway*. Obtenido de https://shadrin.org/nginx/blog/content/microservices-api-gateways-part-1-why-an-api-gateway.html

Pilamunga. (10 de 05 de 2024). *Microservicios para el módulo de proyectos del sistema de Gestión de Investigación de la UNACH*.

Pilamunga, J. D., & Angamarca, L. F. (27 de 05 de 2024). *Microservicios para el módulo de proyectos del sistema de gestión de investigación de la Unach.* Obtenido de http://dspace.unach.edu.ec/handle/51000/13043

Richardson, C. (2014). *API Gateway / Backends for Frontends*. Obtenido de https://microservices.io/patterns/apigateway.html

Richardson., C. (2014). *API Gateway / Backends for Frontends*. Obtenido de https://microservices.io/patterns/apigateway.html

Salgero, E. (22 de 06 de 2018). *Mediun* . Obtenido de Arquitectura Hexagonal .

Salguero, E. (22 de 06 de 2018). *Arquitectura Hexagonal*. Obtenido de https://medium.com/@edusalguero/arquitectura-hexagonal-59834bb44b7f

Services, A. W. (2021). *Implementación de microservicios en la plataforma de comercio electrónico.* Obtenido de Microservices on AWS: https://aws.amazon.com/microservices/

Softtrader. (29 de 07 de 2021). Obtenido de https://softtrader.es/que-es-sql-server-management-studio/

Solvetic. (12 de 03 de 2015). Obtenido de https://www.solvetic.com/tutoriales/article/1502-c%C3%B3mo-medir-el-rendimiento-de-una-aplicaci%C3%B3n-web/

Tonato, e. a. (2024). Implementación de microservicios en entornos académicos. Obtenido de Mejoras en la eficiencia operativa mediante microservicios.

Vivanco, G. M. (2018). *ANÁLISIS DE RENDIMIENTO DE UN SISTEMA OBTENIDO DE*. Obtenido de http://dspace.espoch.edu.ec/bitstream/123456789/8088/1/20T00978.PDF

ANEXOS

Se despliega una ventana si el proyecto de investigación cuenta o no una conexión directa; con articulación con la sociedad, continuar con el proceso de postulación como se muestra en la Figura 35.

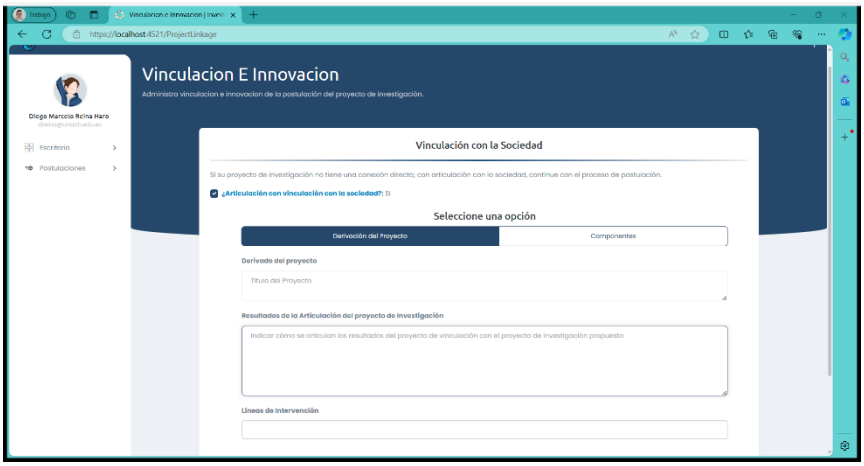


Figura 35: Vinculación e innovación de proyectos

Se despliega una ventana de objetivos de desarrollo sostenible que aporta al proyecto, como se muestra en la Figura 36.

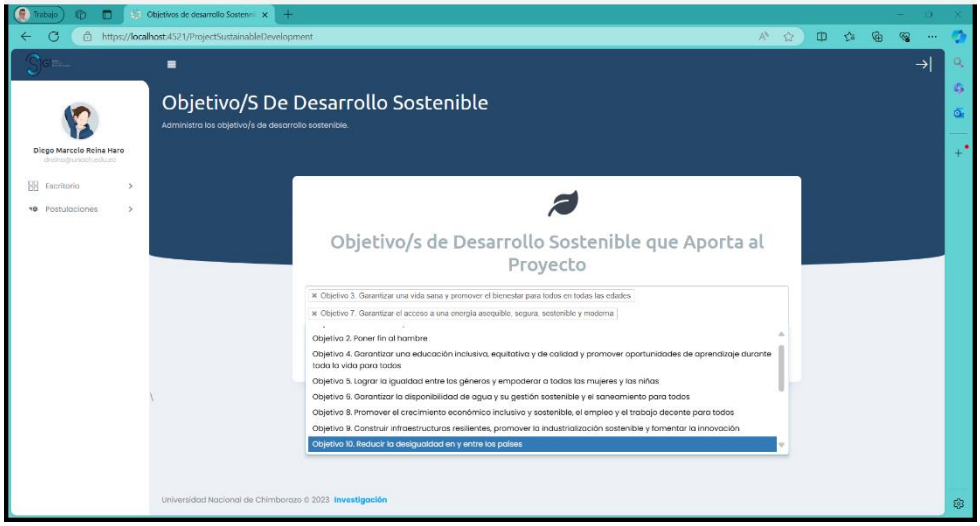


Figura 36: Objetivos del desarrollo sostenible del proyecto

Se despliega una ventana sobre la descripción del proyecto, como se muestra en la Figura 37.

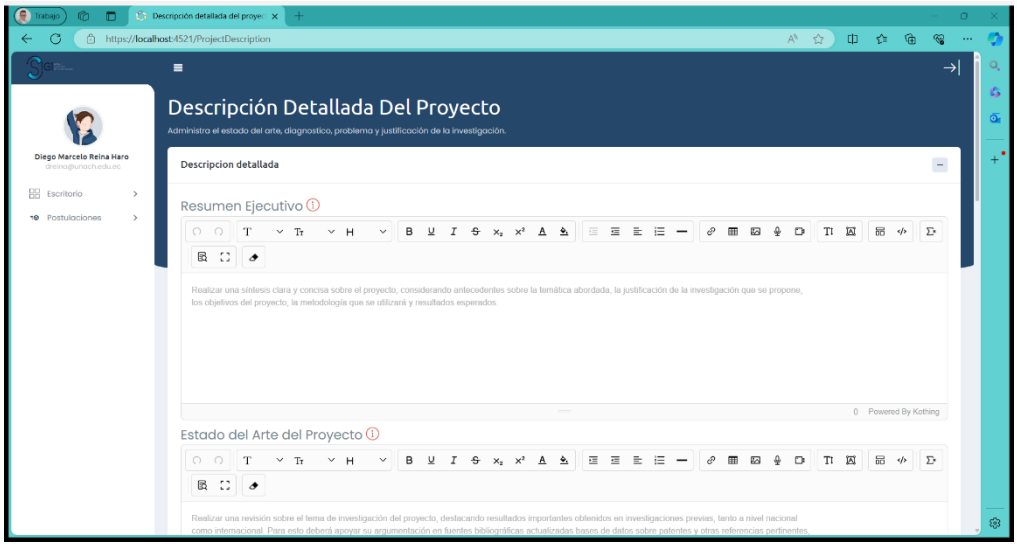
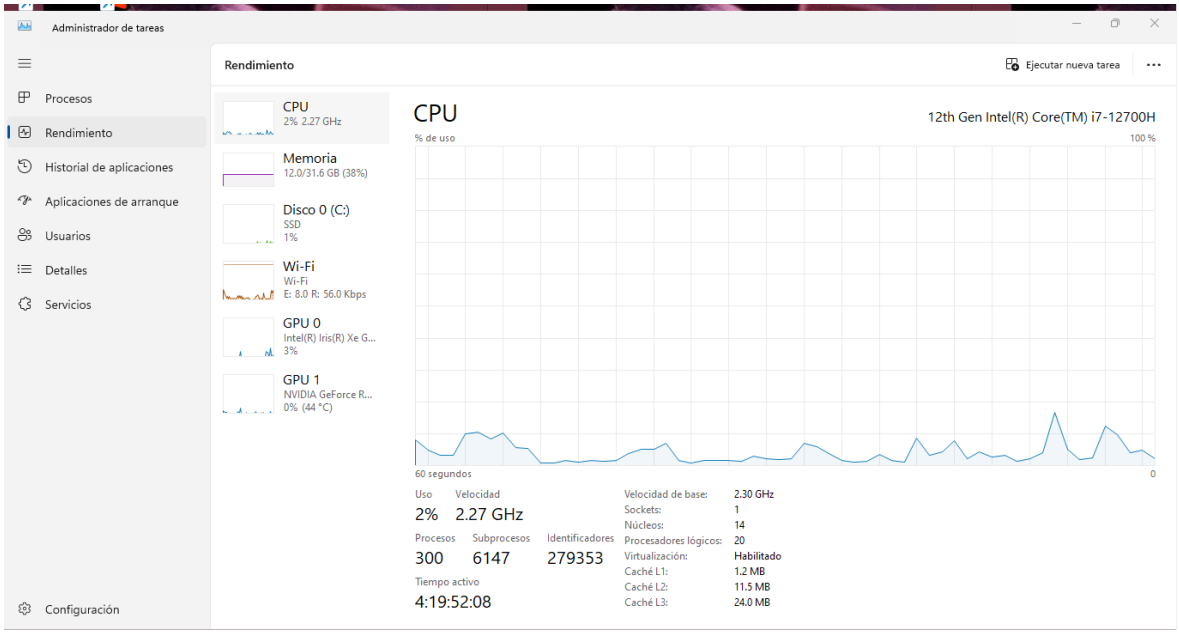


Figura 37: Descripción detalla del proyecto

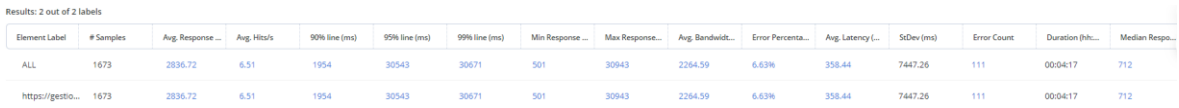
Una vez ingresado todos los componetes esta el proceso de postular el proyecto donde todas las autoridades responsables del proyecto califican y dan un seguimiento con sus respectivas observaciones.

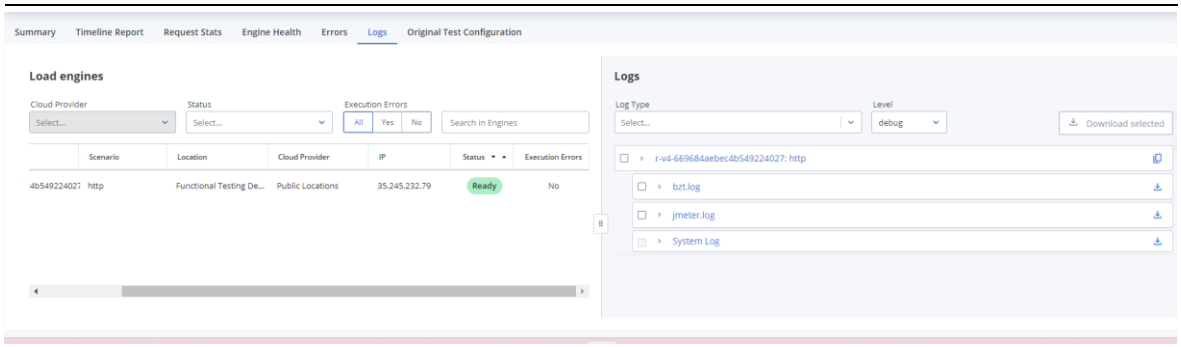
Figura 38 : Postulación de Proyectos

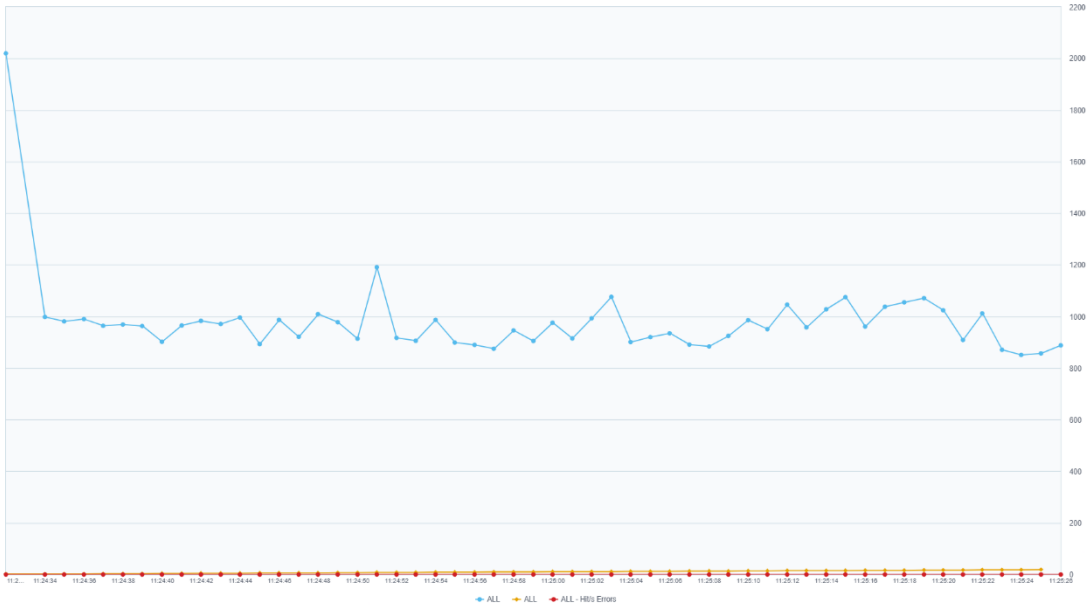
Anexo 2

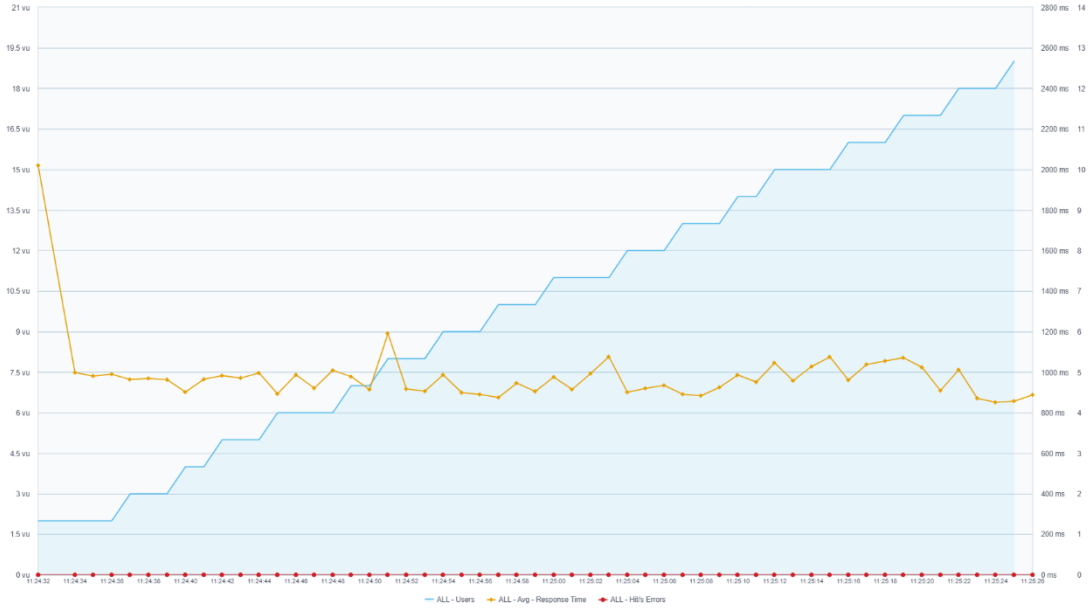

Administrador de tareas
Rendimiento
Ejecutar nueva tarea
Procesos
Rendimiento
Historial de aplicaciones
Aplicaciones de arranque
Usuarios
Detalles
Servicios
Configuración
CPU
2% 2.27 GHz
Memoria
12.0/31.6 GB (38%)
Disco 0 (C:)
SSD
1%
Wi-Fi
Wi-Fi
E: 8.0 R: 56.0 Kbps
GPU 0
Intel(R) Iris(R) Xe G...
3%
GPU 1
NVIDIA GeForce R...
0% (44 °C)
CPU
12th Gen Intel(R) Core(TM) i7-12700H
% de uso
100 %
60 segundos
0
Uso
2%
Velocidad
2.27 GHz
Procesos
300
Subprocesos
6147
Identificadores
279353
Tiempo activo
4:19:52:08
Velocidad de base: 2.30 GHz
Sockets: 1
Núcleos: 14
Procesadores lógicos: 20
Virtualización: Habilitado
Caché L1: 1.2 MB
Caché L2: 11.5 MB
Caché L3: 24.0 MB


Summary
Timeline Report
Request Stats
Engine Health
Errors
Logs
Original Test Configuration
10 VU
Max Users
7.5 Hits/s
Avg. Throughput
0 %
Errors
741.57 ms
Avg. Response Time
815 ms
90% Response Time
3.31 MiB/s
Avg. Bandwidth


Results: 2 out of 2 labels
Element Label
Samples
Avg. Response ...
Avg. Hits/s
90% line (ms)
95% line (ms)
99% line (ms)
Min Response ...
Max Response...
Avg. Bandwidt...
Error Percenta...
Avg. Latency (...
StDev (ms)
Error Count
Duration (hh:...
Median Respo...
ALL 1673 2836.72 6.51 1954 30543 30671 501 30943 2264.59 6.63% 358.44 7447.26 111 00:04:17 712
https://gestio... 1673 2836.72 6.51 1954 30543 30671 501 30943 2264.59 6.63% 358.44 7447.26 111 00:04:17 712


Summary
Timeline Report
Request Stats
Engine Health
Errors
Logs
Original Test Configuration
Load engines
Cloud Provider
Status
Execution Errors
All
Yes
No
Search in Engines
Scenario
Location
Cloud Provider
IP
Status
Execution Errors
4b549224027 http Functional Testing De... Public Locations 35.245.232.79 Ready No
Logs
Log Type
Level
debug
Download selected
r-v4-669684aebec4b549224027: http
bzt.log
jmeter.log
System Log


2200
2000
1800
1600
1400
1200
1000
800
600
400
200
0
ALL
ALL
ALL - Hit/s Errors


21 vu
19.5 vu
18 vu
16.5 vu
15 vu
13.5 vu
12 vu
10.5 vu
9 vu
7.5 vu
6 vu
4.5 vu
3 vu
1.5 vu
0 vu
2800 ms
2600 ms
2400 ms
2200 ms
2000 ms
1800 ms
1600 ms
1400 ms
1200 ms
1000 ms
800 ms
600 ms
400 ms
200 ms
0 ms
ALL - Users
ALL - Avg - Response Time
ALL - Hit/s Errors