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

Microservicios para el módulo de proyectos del sistema de Gestión de Investigación de la UNACH

Trabajo de Titulación para optar al título de Ingeniero en Tecnologías de la Información

Autores:
Angamarca Murquincho Leidy Francisca
Pilamunga Valla Jonathan David

Tutor:
MsC. Pamela Alexandra Buñay Guisñan

Riobamba, Ecuador. 2024

DECLARATORIA DE AUTORÍA

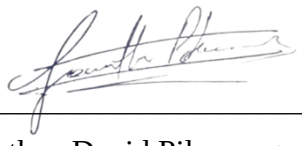
Nosotros, Leidy Francisca Angamarca Murquincho, con cédula de ciudadanía 1150016499 y Jonathan David Pilamunga Valla, con cédula de ciudadanía 0606218816, autores del trabajo de investigación titulado: Microservicios para el módulo de proyectos del sistema de Gestión de Investigación de la UNACH, certificamos que la producción, ideas, opiniones, criterios, contenidos y conclusiones expuestas son de nuestra exclusiva responsabilidad.

Asimismo, cedemos 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 de la obra referida será de nuestra entera responsabilidad; librando a la Universidad Nacional de Chimborazo de posibles obligaciones.

En Riobamba, 22 de mayo de 2024.

Leidy Francisca Angamarca Murquincho

C.I: 1150016499



Jonathan David Pilamunga Valla

C.I: 0606218816

DICTAMEN FAVORABLE DEL PROFESOR TUTOR

ACTA FAVORABLE - INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN

En la Ciudad de Riobamba, a los 17 días del mes de abril de 2024, luego de haber revisado el Informe Final del Trabajo de Investigación presentado por los estudiantes **LEIDY FRANCISCA ANGAMARCA MURQUINCHO** con CC: **1150016499** y **JONATHAN DAVID PILAMUNGA VALLA** con CC: **0606218816**, de la carrera de **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 **"MICROSERVICIOS PARA EL MÓDULO DE PROYECTOS DEL SISTEMA DE GESTIÓN DE INVESTIGACIÓN DE LA UNACH"**, por lo tanto se autoriza la presentación del mismo para los trámites pertinentes.

PAMELA ALEXANDRA BUNAY GUISNAN

MsC. Pamela Alexandra Buñay Guisñan
TUTORA

CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

Quienes suscribimos, catedráticos designados Miembros del Tribunal de Grado para la evaluación del trabajo de investigación Microservicios para el módulo de proyectos del sistema de Gestión de Investigación de la UNACH, presentado por Leidy Francisca Angamarca Murquincho, con cédula de identidad número 1150016499 y Jonathan David Pilamunga Valla, con cédula de identidad número 0606218816, bajo la tutoría de MsC. Pamela Alexandra Buñay Guisñan; certificamos que recomendamos la APROBACIÓN 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 22 de mayo de 2024.

Estela Narváez, PhD.
PRESIDENTE DEL TRIBUNAL DE GRADO

Johanna Moyano, Mgs.
MIEMBRO DEL TRIBUNAL DE GRADO

Ximena Quintana, PhD.
MIEMBRO DEL TRIBUNAL DE GRADO

CERTIFICADO ANTIPLAGIO
Original

Dirección
Académica
VICERRECTORADO ACADÉMICO

en movimiento
SGC
SISTEMA DE GESTIÓN DE LA CALIDAD
UNACH-RGF-01-04-08.15
VERSIÓN 01: 06-09-2021

CERTIFICACIÓN

Que, **ANGAMARCA MURQUINCHO LEIDY FRANCISCA** con CC: **1150016499** y **PILAMUNGA VALLA JONATHAN DAVID** con CC: **0606218816**, estudiantes de la Carrera de **INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN**, Facultad de **INGENIERÍA**; han trabajado bajo mi tutoría el trabajo de investigación titulado "**MICROSERVICIOS PARA EL MÓDULO DE PROYECTOS DEL SISTEMA DE GESTIÓN DE INVESTIGACIÓN DE LA UNACH**", cumple con el 9 %, 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, 13 de mayo de 2024

PAMELA ALEXANDRA BUÑAY GUISÑAN

Mgs. Pamela Alexandra Buñay Guisñan
TUTORA

DEDICATORIA

Dedico este trabajo de investigación a mis padres, Rosa Murquincho y Manuel Angamarca, quienes han sido la principal fuente de inspiración y apoyo para culminar mi carrera universitaria. A mis hermanas y sobrinos por sus palabras de aliento en momentos de dificultad.

Leidy Angamarca

Dedico este proyecto de investigación a mi familia, su amor y apoyo constante han sido una fuente inagotable de fortaleza a lo largo de mi travesía universitaria. A mis amigos, quienes compartieron conmigo tanto las alegrías como los desafíos de este recorrido.

Jonathan Pilamunga

AGRADECIMIENTO

Primeramente, agradecemos a Dios por darnos la fortaleza de seguir adelante, a pesar de los desafíos que se presentaron en el camino. A nuestros padres, quienes han sido nuestros motores incondicionales durante todo este proceso. Su amor, apoyo y sacrificio han sido la brújula que nos ha guiado hasta este punto.

Expresamos nuestro sincero agradecimiento a la Universidad Nacional de Chimborazo por abrirnos sus puertas y permitirnos formarnos como profesionales, al igual que los docentes, quienes compartieron sus conocimientos con dedicación y paciencia, desempeñando un papel fundamental en nuestro crecimiento académico.

Un reconocimiento especial a nuestra tutora, MsC. Pamela Buñay, cuya orientación experta han sido invaluables durante el proceso de desarrollo de nuestra tesis. A su vez, agradecemos al personal del área de Investigación de la UNACH, especialmente al Ing. Alex Asitimbay y Ing. Alex Buñay, por confiar en nuestro trabajo y solventar nuestras dudas.

A nuestros amigos y compañeros de la universidad, les agradecemos por su amistad y compañerismo. Sus palabras de ánimo y presencia constante en nuestra vida universitaria nos motivaron a esforzarnos al máximo para culminar este importante capítulo de nuestras vidas.

Leidy Angamarca y Jonathan Pilamunga

ÍNDICE GENERAL

DECLARATORIA DE AUTORÍA
DICTAMEN FAVORABLE DEL PROFESOR TUTOR
CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL
CERTIFICADO ANTIPLAGIO
DEDICATORIA
AGRADECIMIENTO
ÍNDICE GENERAL
RESUMEN
ABSTRACT
CAPÍTULO I. INTRODUCCION. 17
1.1. Antecedentes 17
1.2. Planteamiento del Problema 18
1.2.1. Problema y Justificación 18
1.2.2. Formulación del Problema 19
1.3. Objetivos 19
1.3.1. Objetivo General 19
1.3.2. Objetivos Específicos 19
CAPÍTULO II. MARCO TEÓRICO. 20
2.1. Arquitectura de microservicios 20
2.1.1. Características 20
2.1.2. Ventajas 21
2.1.3. Desventajas 21
2.1.4. Comparación con la arquitectura monolítica 21
2.1.5. Pruebas unitarias 22
2.2. Estrategias de Diseño 23
2.2.1. Principios de diseño 23
2.2.2. Patrones de diseño 24
2.3. IDE de Desarrollo 25
2.3.1. JeatBrains Rider 25
2.3.2. Funcionalidades 26
2.4. Lenguaje de programación 27

2.4.1. C# 27
2.5. Framework y Bibliotecas 28
2.5.1. .NET 7 28
2.5.2. Bibliotecas estándares en C# y .NET 30
2.6. Paquete Nuget 31
2.7. Azure Devops 31
2.8. Metodología SCRUM 32
2.8.1. Roles 32
2.8.2. Artefactos 32
2.8.3. Eventos 32
2.8.4. Impacto de la metodología ágil Scrum en el proceso de desarrollo de software ... 33
2.9. Modelo de Calidad FURPS 34
2.9.1. Parámetros de medición del rendimiento 34
2.9.2. Valores de ponderación del rendimiento según el modelo FURPS 34
2.10. Apache JMeter 34
CAPÍTULO III. METODOLOGIA. 36
3.1. Tipo y diseño de Investigación 36
3.1.1. Según la fuente de la investigación 36
3.1.2. Según el objeto de estudio 36
3.2. Instrumentos de recolección de Datos 36
3.3. Población de estudio y tamaño de muestra 36
3.4. Identificación de variables 36
3.4.1. Variable Dependiente 36
3.4.2. Variable Independiente 36
3.5. Operacionalización de Variables 36
3.6. Metodología de desarrollo de software 38
3.6.1. Inicio 38
3.6.2. Planificación y estimación 40
3.6.3. Implementación 43
3.6.4. Revisión y retrospectiva 53
3.6.5. Lanzamiento 54
3.7. Pruebas 54
3.7.1. Parámetros de evaluación 54

3.7.2. Ejecución de pruebas 55
CAPÍTULO IV. RESULTADOS Y DISCUSIÓN 57
4.1. Interpretación de Resultados 57
4.1.1. Valoración de indicadores 57
4.1.2. Comparación entre los valores obtenidos y valores establecidos en el modelo de calidad FURPS 61
4.2 Discusión 61
CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES 63
5.1 Conclusiones 63
5.2 Recomendaciones 64
BIBLIOGRÁFIA 65
ANEXOS 67

ÍNDICE DE TABLAS

Tabla 1. Arquitectura monolítica vs microservicios 22
Tabla 2. Descripción de las funcionalidades 26
Tabla 3. Características .net 7 y sus otras versiones 29
Tabla 4. Valores de ponderación del rendimiento según el modelo FURPS 34
Tabla 5. Operacionalización de Variables 37
Tabla 6. Equipo Scrum 38
Tabla 7. Requerimientos funcionales 38
Tabla 8. Requerimientos no funcionales 39
Tabla 9. Product Backlog 40
Tabla 10. Spring Backlog 42
Tabla 11. Tabla Principal de Proyectos 45
Tabla 12. Parámetros de evaluación 54
Tabla 13. Procedimientos Realizados 57
Tabla 14. Resultados del indicador Eficacia 58
Tabla 15. Resultados de utilización de recursos 59
Tabla 16. Comparación de los valores obtenidos y los de FURPS 61
Tabla 17. Tabla ProjectInvestigation 67
Tabla 18. Tabla Abstract 67
Tabla 19. Tabla Activity 68
Tabla 20. Tabla ApprovalRelevance 68
Tabla 21. Tabla AreaLineDomain 68
Tabla 22. Tabla ArticulationLinkage 69
Tabla 23. Tabla Aspect 69
Tabla 24. Tabla Canton 70
Tabla 25. Tabla CareerFaculty 70
Tabla 26. Tabla Description 70
Tabla 27. Tabla EnvironmentalImpact 71
Tabla 28. Tabla ExecutionCoverage 71
Tabla 29. Tabla HistoryApprovalRelevance 72
Tabla 30. Tabla Impact 72
Tabla 31. Tabla Innovation 73
Tabla 32. Tabla InnovationDevelopment 73

Tabla 33. Tabla InterventionLines 73
Tabla 34. Tabla LogicFrames 74
Tabla 35. Tabla NationalPlan 74
Tabla 36. Tabla Network 74
Tabla 37. Tabla Objective 75
Tabla 38. Tabla PlanningZone 75
Tabla 39. Tabla ProjectBeneficiary 75
Tabla 40. Tabla Province 76
Tabla 41. Tabla Record 76
Tabla 42. Tabla Reference 77
Tabla 43. Tabla ResearchAssistant 77
Tabla 44. Tabla ResearchExternal 78
Tabla 45. Tabla ResearchInternal 78
Tabla 46. Tabla Resource 78
Tabla 47. Tabla Result 79
Tabla 48. Tabla ResultDescription 79
Tabla 49. Tabla SharedResearch 80
Tabla 50. Tabla SpecificObjective 80
Tabla 51. Tabla Sustainability 81
Tabla 52. SustainableDevelopment 81
Tabla 53. Tabla TransferResult 82

ÍNDICE DE FIGURAS

Figura 1. Diseño de arquitectura de microservicios 20
Figura 2. Principios SOLID 24
Figura 3. Entorno de desarrollo en JetBrains Rider 26
Figura 4. C# 28
Figura 5. Plataformas de .NET 29
Figura 6. Proceso SCRUM 33
Figura 7. Marcos de trabajo ágil más usados 33
Figura 8. Diagrama relacional 43
Figura 9. Diagrama físico de la base de datos 44
Figura 10. Caso de uso de análisis de perfil de proyecto 46
Figura 11. Caso de uso para la Gestión de proyectos 46
Figura 12. Caso de uso del proceso de revisión y resolución para proyectos 47
Figura 13. Diseño general del sistema 47
Figura 14. Diseño lógico del microservicio Proyectos 48
Figura 15. Descarga de plantilla 48
Figura 16. Paquetes Nuget 49
Figura 17. Sección Dominio 49
Figura 18. Clase ProjectDbContext 50
Figura 19. Configuración de cada entidad 50
Figura 20. Sección Aplicación 51
Figura 21. Sección Api 51
Figura 22. Dtos de las entidades 52
Figura 23. Sección Pruebas unitarias 52
Figura 24. Validación de atributos en las pruebas 53
Figura 25. Documentación de Api Rest en JetBrains Rider con Swagger 53
Figura 26. Microservicio publicado 54
Figura 27. Configuración del escenario 55
Figura 28. Configuración de hilos 55
Figura 29. Resultados generales 56
Figura 30. Porcentaje de éxito del módulo 57
Figura 31. Indicador Eficacia 58
Figura 32. Tiempo de respuesta 59

Figura 33. Uso de CPU 60
Figura 34. Uso de Memoria RAM 60
Figura 35. Uso de Disco Duro 61

RESUMEN

Este trabajo de investigación aborda la implementación de microservicios en el módulo de proyectos del sistema de Gestión de Investigación de la Universidad Nacional de Chimborazo (UNACH). Esta iniciativa surge como parte de la migración del sistema hacia una arquitectura moderna que fragmenta la aplicación en servicios más pequeños e independientes, con el fin de mejorar la escalabilidad, la integración y la seguridad del sistema. La investigación fue de tipo cuantitativa. Para el desarrollo del módulo, se emplearon diversas tecnologías, entre las que se incluyen el lenguaje de programación C# junto con el Framework .NET7, el entorno de desarrollo integrado JetBrains Rider, y para la gestión de bases de datos, SQL Server. Además, se aplicó la metodología de desarrollo ágil Scrum, que permitió optimizar el proceso de implementación de los microservicios. Se evaluó el rendimiento del módulo utilizando la herramienta JMeter y se compararon sus métricas de calidad con los estándares establecidos por el modelo FURPS. Se logró una eficacia del 100%, un tiempo de respuesta de 0,32 segundos y un uso de recursos del 21%. Estos resultados confirman que el módulo de proyectos de investigación desarrollado cumple con los criterios de calidad establecidos.

Palabras claves: Microservicios, Scrum, Rendimiento, Modelo FURPS, JMeter

ABSTRACT

This research focuses on implementing microservices in the Research Management System project module at the National University of Chimborazo (UNACH). This initiative is part of a system migration towards a modern architecture that fragments the application into more minor, independent services, enhancing the system's scalability, integration, and security. The research, conducted quantitatively, utilized several technologies, such as the C# programming language.NET7 Framework, the JetBrains Rider integrated development environment, and SQL Server for database management. Notably, the agile development methodology Scrum was applied, which was pivotal in optimizing the microservices' implementation process. The module's performance was evaluated using the JMeter tool, and its quality metrics were compared against the standards set by the FURPS model. The module achieved an efficiency of 100%, a response time of 0.32 seconds, and a resource usage of 21%. These outstanding results affirm that the research project module meets and surpasses the established quality criteria.

Keywords: Microservices, Scrum, Performance, FURPS Model, JMeter

Reviewed by
Mgs. Vanessa Palacios.
ENGLISH PROFFESOR
C.C. 0603247487

CAPÍTULO I. INTRODUCCION

La gestión eficaz de la información en aplicaciones es crucial en la era digital, en particular en sistemas que manejan un volumen considerable de transacciones y usuarios concurrentes. Este escenario desafía la eficiencia de los recursos computacionales y la calidad de los servicios, impulsando la necesidad de administrar estos aspectos de manera que las aplicaciones no solo alcancen, sino también superen sus objetivos y requerimientos. En respuesta a estos retos, la arquitectura de microservicios se ha revelado como una solución moderna y eficaz para mitigar la complejidad, ofreciendo ventajas como escalabilidad automática y detallada, pruebas automatizadas, integración y despliegue continuo, seguridad mejorada y mayor tolerancia a fallos. Además, el desarrollo de aplicaciones basadas en microservicios, integrando prácticas de DevOps, facilita actualizaciones constantes, rápidas y automatizadas, resultando en entregas más ágiles y probadas (Vera-Rivera et al., 2019).

La arquitectura de microservicios se caracteriza por descomponer una aplicación en componentes más pequeños e independientes llamados microservicios. Cada microservicio se encarga de una función específica de la aplicación y puede operar de forma autónoma. En contraste con la arquitectura monolítica, donde toda la aplicación se ejecuta en un solo servidor, los microservicios pueden ejecutarse en múltiples instancias en uno o varios servidores, permitiendo un escalado dinámico de recursos según las demandas de la carga de trabajo (Intel Corporation, 2021).

La adopción de la metodología Scrum en proyectos basados en microservicios es esencial. Scrum, con su enfoque ágil y adaptable, agiliza la gestión de la complejidad de los microservicios, permitiendo una rápida adaptación a cambios y fomentando la colaboración, lo que resulta fundamental para el éxito del trabajo.

Ante la creciente complejidad y volumen de datos en los proyectos de investigación, el CODESI de la Universidad Nacional de Chimborazo (UNACH) reconoció la necesidad de evolucionar y mejorar sus procesos de gestión. Este cambio resultó fundamental para abordar de manera más eficiente los desafíos derivados del crecimiento de datos en proyectos de investigación, facilitando la gestión de cargas variables y la adaptación a cambios en la demanda.

El trabajo se centró en las necesidades específicas del área de investigación de la UNACH, buscando ofrecer una solución tecnológica sólida y adaptable que contribuya al avance y excelencia en la gestión de proyectos con el desarrollo del módulo aplicando la metodología ágil Scrum, se midió el rendimiento mediante el modelo de calidad FURPS.

1.1. Antecedentes

En el año 2020, en la Universidad Politécnica de Madrid, desarrolló un framework computacional orientado a la docencia, se basó en la arquitectura de microservicios. Esta herramienta se convirtió en la base tecnológica para la creación de una plataforma que

facilitó la labor innovadora de los profesores, permitiendo la revisión en línea y automática de actividades de aprendizaje compartidas con los estudiantes. Además, estableció un marco de referencia para la integración de componentes que habilitaron la evaluación en línea, automática y con capacidad de aprendizaje automático, brindando retroalimentación a los estudiantes (Martín & García, 2020).

Por otro lado, en el trabajo titulado "Aplicación Web para el Servicio de Trámites Académicos de la UNACH usando una Arquitectura basada en Microservicios", se logró mejorar la eficiencia en la respuesta a los requerimientos de los estudiantes como la generación de récords académicos, certificados de culminación de estudios, constancias de no adeudo al departamento de TICs y certificados de matrícula por parte del colectivo administrativo de la Universidad Nacional de Chimborazo (Alvarado, 2023). Este trabajo destacó la flexibilidad de manipular múltiples procesos simultáneamente sin afectar la eficiencia del sistema académico universitario.

Así también, en el ámbito de la automatización de la industria de extracción de hidrocarburos se aplicó esta arquitectura. Un ejemplo de esta situación es el proyecto llevado a cabo en la empresa BLC Venezuela, que implementó el desarrollo de un sistema informático con una arquitectura basada en los conceptos de microservicios y computación en la nube. Esto permitió mejorar aspectos en el sistema como el rendimiento, escalabilidad, tolerancia a fallas, independencia de los datos, etc. (León, 2021).

Además, cabe recalcar la importancia de ejecutar la metodología DevOps para la automatización en la construcción y despliegue de microservicios. La utilización de esta metodología ha permitido el despliegue eficiente de microservicios a través de contenedores y su gestión mediante Kubernetes. La adopción de esta estrategia demostró ser efectiva para lograr la escalabilidad y asegurar la continuidad operativa de las aplicaciones, incluso en situaciones de caídas o fallo (Cusco, 2022).

Los antecedentes proporcionados fueron relevantes y respaldaron la importancia de la arquitectura de microservicios en diferentes contextos, incluyendo la educación, la administración universitaria, la industria de extracción de hidrocarburos y la automatización. Estos casos de estudio demostraron cómo la adopción de microservicios mejoró la eficiencia, escalabilidad y capacidad de adaptación en una variedad de aplicaciones y sectores.

1.2. Planteamiento del Problema

1.2.1. Problema y Justificación

La Dirección de Investigación de la Universidad Nacional de Chimborazo ha estado comprometida con el desarrollo de investigaciones avanzadas en ciencia y tecnología. Su objetivo siempre ha sido abordar y resolver problemas cruciales que afectan a las comunidades locales, regionales y nacionales (UNACH, 2023). Para lograr esto de manera

eficaz, fue esencial contar con un sistema tecnológico que se alineara con sus necesidades pasadas y futuras, que facilitan la gestión y el análisis de datos.

En este contexto, el CODESI de la UNACH tomó la decisión estratégica de migrar de una arquitectura monolítica existente a una arquitectura basada en microservicios. Esta transición se justificó por las ventajas que ofrecía, incluyendo mayor escalabilidad, rendimiento, flexibilidad y eficiencia. Estas características fueron cruciales para el desarrollo y mantenimiento de sistemas complejos, permitiendo una mejor adaptación a las demandas cambiantes del entorno de investigación.

Por tal motivo, se desarrolló el módulo de proyectos del sistema de Gestión de Investigación de la UNACH con la arquitectura de microservicios, utilizando la metodología de desarrollo SCRUM y se evaluó el rendimiento del módulo mediante los indicadores del modelo de calidad FURPS.

1.2.2. Formulación del Problema

¿Cómo los microservicios influirán en el rendimiento del módulo de proyectos del sistema de Gestión de Investigación de la UNACH?

1.3. Objetivos

1.3.1. Objetivo General

Implementar microservicios para el módulo de proyectos del sistema de Gestión de Investigación de la UNACH.

1.3.2. Objetivos Específicos

- Investigar las herramientas adecuadas para el desarrollo microservicios para el módulo de proyectos del sistema de Gestión de Investigación de la UNACH.
- Desarrollar el módulo de proyectos del sistema de Gestión de Investigación de la UNACH.
- Evaluar el rendimiento del módulo de proyectos mediante el modelo de calidad FURPS.

CAPÍTULO II. MARCO TEÓRICO

2.1. Arquitectura de microservicios

Durante mucho tiempo, la visión predominante en el desarrollo de software fue la arquitectura monolítica, pero su crecimiento conllevó dificultades de escalabilidad y otras funcionalidades. En cambio, los microservicios surgieron como una alternativa transformadora y ágil.

La arquitectura de microservicios representó una estrategia arquitectónica y organizativa en el desarrollo de software, caracterizada por la creación de pequeños servicios autónomos que se comunican mediante APIs claramente definidas. Estos servicios son gestionados por equipos reducidos e independientes. Esta arquitectura facilita la escalabilidad y acelera el proceso de desarrollo de proyectos de software (AWS, 2023).

Además, se conoce la arquitectura de microservicios como una colección de servicios independientes y debe implementar una funcionalidad de negocio individual dentro de un contexto delimitado. Un contexto delimitado es una división natural de una empresa y proporciona un límite explícito dentro del cual existe un modelo de dominio (Azure, 2020).

La Figura 1 muestra la configuración completa de la arquitectura de microservicios, ofreciendo una perspectiva esencial para comprender la disposición y la interconexión de los elementos clave de este estilo arquitectónico.

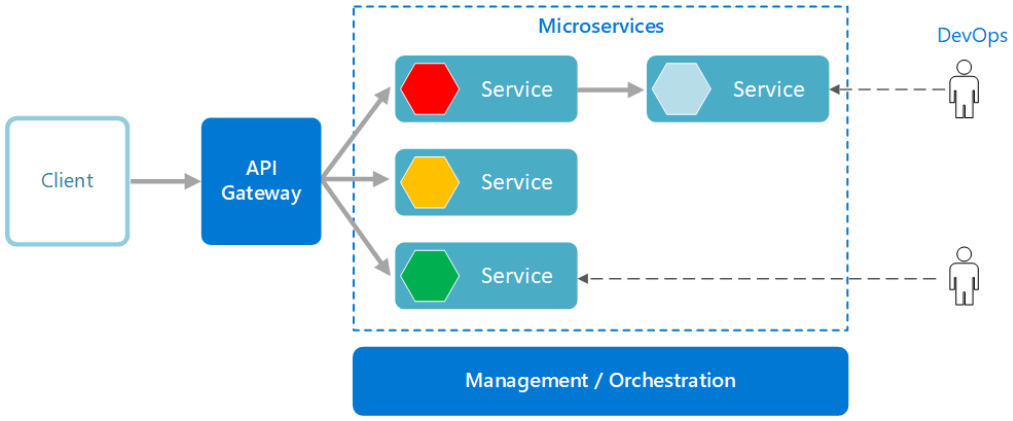


Figura 1. Diseño de arquitectura de microservicios
Fuente: (Azure, 2020)

2.1.1. Características

A continuación, se exploran las características distintivas de los microservicios, proporcionando una comprensión más profunda de este enfoque arquitectónico innovador (Fernández, 2021):

- Alto nivel de desacoplamiento: Cada microservicio opera en su propio dominio, facilitando la comunicación y el intercambio de datos con otros servicios sin generar dependencias significativas.

- Nombres únicos: Cada microservicio se distingue por un nombre exclusivo, permitiendo una fácil identificación y ubicación dentro del entorno arquitectónico.
- Resiliencia: Ante posibles problemas, los microservicios tienen la capacidad de reiniciarse en otra máquina, asegurando la integridad de los datos y la información.
- Independencia: La implementación, escalado y actualización de cada microservicio se lleva a cabo de manera independiente, ofreciendo flexibilidad y agilidad en el desarrollo.

2.1.2. Ventajas

Estas ventajas no solo optimizan la operación y el desarrollo, sino que también destacan la versatilidad inherente de esta arquitectura avanzada (Fernández, 2021):

- Escalabilidad: La arquitectura de microservicios permite una escalabilidad eficiente, cada servicio puede escalarse independientemente para satisfacer la demanda específica de funcionalidades.
- Módulos independientes: Cada microservicio opera como un módulo independiente, reduciendo las interdependencias y facilitando la gestión individual de servicios.
- Total libertad del desarrollador: La capacidad de desarrollar y desplegar servicios de manera independiente proporcionó una libertad significativa a los desarrolladores, permitiéndoles adaptarse rápidamente a las necesidades cambiantes y a las innovaciones tecnológicas.

2.1.3. Desventajas

La arquitectura de microservicios, aunque presenta beneficios, también conlleva desafíos significativos. El alto consumo de memoria es una preocupación, el manejo de múltiples servicios puede requerir recursos considerables. La fragmentación de distintos microservicios demanda tiempo y cuidado en la planificación. La complejidad aumenta con la gestión de un gran número de servicios, puede dar lugar a dificultades operativas. Además, la ejecución de pruebas o testeos en un entorno de despliegue distribuido añade una capa adicional de complicación, cada microservicio debe ser evaluado en términos de su interdependencia y funcionalidad global. Estos desafíos resaltan la importancia de un enfoque cuidadoso y estratégico al implementar arquitecturas basadas en microservicios (Chakray, 2019).

2.1.4. Comparación con la arquitectura monolítica

En una arquitectura monolítica, todos los procesos están estrechamente interrelacionados, lo que implica que, al enfrentar una demanda inesperada en un proceso, es necesario escalar toda la estructura. La complejidad creciente al agregar o mejorar características limita la experimentación y dificulta la implementación de nuevas ideas. Además, la dependencia entre procesos aumenta el riesgo de errores. Por otro lado, en una arquitectura de microservicios, la aplicación se compone de servicios independientes que se comunican a través de interfaces definidas por API (AWS, 2023). En síntesis, la adopción de microservicios plantea una perspectiva alentadora para optimizar el desarrollo y la operación de sistemas, particularmente en entornos académicos.

En la Tabla 1, se presenta una comparación detallada entre dos paradigmas arquitectónicos prominentes: la arquitectura monolítica y la arquitectura de microservicios. La tabla destaca las diferencias clave en aspectos como la estructura, el desarrollo, el mantenimiento, disponibilidad y otros elementos esenciales.

Tabla 1. Arquitectura monolítica vs microservicios

	Monolítica	**Microservicios**
Estructura	Un solo bloque de código y base de datos.	Compuesta por servicios independientes y bases de datos específicas para cada servicio.
Desarrollo	Desarrollo y despliegue simple y menos flexible.	Mayor flexibilidad en el desarrollo y despliegue de servicios independientes.
Comunicación entre Componentes	Comunicación interna directa.	Comunicación mediante interfaces bien definidas y API ligeras.
Adaptabilidad	Menos adaptable a cambios y a nuevas tecnologías.	Mayor adaptabilidad y capacidad para incorporar nuevas tecnologías de forma modular.
Disponibilidad	Mayor riesgo de disponibilidad debido a la interconexión.	Mayor robustez y disponibilidad al ser servicios independientes.
Mantenimiento	Actualizaciones más complejas debido a la interdependencia.	Facilita el mantenimiento y actualización de servicios de manera independiente.

2.1.5. Pruebas unitarias

Las pruebas desempeñan un papel fundamental en el desarrollo de software, garantizando la calidad y funcionalidad del producto en diversas etapas del proceso (Laura, 2023):

- Pruebas de Unidad: Estas pruebas validan que los métodos y clases desarrollados funcionen según lo previsto. Aunque son tareas técnicas para los desarrolladores, constituyen una red de seguridad crucial para detectar posibles consecuencias no deseadas al realizar cambios en el código. Sin embargo, las pruebas unitarias por sí solas no son suficientes para garantizar el rendimiento del sistema.

2.2. Estrategias de Diseño

2.2.1. Principios de diseño

Los principios de diseño en el ámbito de las arquitecturas de microservicios constituyen enfoques comprobados para enfrentar desafíos particulares y mejorar aspectos específicos durante el diseño, desarrollo, implementación y operación de sistemas que utilizan la arquitectura de microservicios (Richardson, 2023).

Entre estos principios se halla SOLID, un acrónimo que engloba los cinco principios de diseño con el propósito de mejorar la claridad, adaptabilidad y mantenimiento de los sistemas de software (Caqui, 2022). A continuación, se describen en detalle cada uno de estos principios:

- SRP: Principio de Responsabilidad Única

  Busca que cada entidad sea responsable de proporcionar una característica única de funcionalidad dentro del programa, encapsulando completamente esa responsabilidad dentro de la clase.

- OCP: Principio Abierto/Cerrado

  Una clase se considera abierta si es posible crear su propia subclase y modificarla según sea necesario, permitiendo agregar nuevos métodos o campos, o anular el comportamiento base.

- LSP: Principio de Sustitución de Liskov

  Predice si una subclase sigue siendo compatible con el código que puede haber funcionado con objetos en la superclase mediante pruebas conocidas como el principio de reemplazo. Es esencial al crear bibliotecas y marcos utilizados por otros.

- ISP: Principio de Segregación de Interfaces

  Dicta que las interfaces "gruesas" deben dividirse en partes más pequeñas para diseñar experiencias de usuario más detalladas. Los clientes solo deben utilizar los servicios que realmente necesitan para evitar que actualizaciones afecten a clientes que no utilizan las operaciones actualizadas.

- DIP: Principio de Inversión de Dependencia

  Las dependencias entre los niveles deben ser evitadas por completo. Es esencial contar con abstracciones en cada uno de ellos. Los detalles no deben servir como fundamento para una síntesis. La utilización de abstracciones es fundamental para manejar los detalles.

En la Figura 2 se visualizan los principios de diseño SOLID, representando de manera clara los componentes fundamentales de esta sigla.

Figura 2. Principios SOLID

Estos principios proporcionan una base sólida para la organización del código en la arquitectura de microservicios, facilitando una estructura coherente y modular. Además, se destacan otros principios de diseño para las arquitecturas de microservicios, tales como la promoción de una cultura de autonomía, propiedad y gobernanza compartida. Además, se enfatiza la automatización de procesos clave como la integración continua, implementación continua y pruebas automatizadas para mejorar la eficiencia y reducir los costos operativos (Alvarado, 2023).

2.2.2. Patrones de diseño

Los patrones de diseño son soluciones probadas y documentadas para problemas comunes que surgen durante el desarrollo de software. Proporcionan un enfoque estructurado y reutilizable para resolver problemas de diseño y arquitectura, permite a los desarrolladores tomar decisiones informadas y eficientes en diferentes contextos.

- El propósito fundamental del patrón Unit of Work es gestionar la colección de objetos que están relacionados con operaciones en la base de datos. Su función principal radica en supervisar los objetos afectados por cambios, facilitar la escritura de dichos cambios en la base de datos y manejar cualquier problema de concurrencia que pueda surgir durante el proceso. Este patrón se implementa de diversas formas, siendo común encontrar su aplicación en clases como DataContext, ObjectContext o ISession. Por lo tanto, para establecer una abstracción adecuada, es fundamental definir una interfaz como IUnitOfWork que permita conceptualizar y estandarizar esta funcionalidad en diferentes contextos de aplicación (Tamayo & Katrib, 2011).

- La adopción del patrón Repository fomenta la reutilización del código, al encapsular la lógica de acceso a datos dentro del repositorio, permite una fácil sustitución de las implementaciones de este último sin afectar otras áreas del sistema. Es importante destacar que el concepto de repositorio no se limita a una única fuente de datos,

pudiendo estar respaldado por múltiples fuentes, como bases de datos relacionales, servicios web, archivos locales, entre otros (Tamayo & Katrib, 2011).

- CQRS, que significa Segregación de Responsabilidades de Comandos y Consultas, es un patrón de diseño que busca separar las operaciones de lectura y escritura en un sistema de almacenamiento de datos. Esta segregación facilita la optimización del rendimiento, la escalabilidad y la seguridad de la aplicación. La adopción de CQRS puede generar una mayor flexibilidad en el sistema, permitiendo una evolución más fluida con el tiempo y evitando posibles conflictos de combinación al nivel del dominio provocados por comandos de actualización (Microsoft, 2023).

- El patrón de diseño DTO, también conocido como Objeto de Transferencia de Datos (en inglés, Data Transfer Object), constituye una técnica comúnmente empleada en el desarrollo de software para la transferencia eficiente de datos entre distintos componentes de una aplicación. Un DTO se presenta como una clase simple que típicamente alberga exclusivamente campos de datos y métodos de acceso, tales como getters y setters, sin contener lógica de negocio adicional. Su propósito fundamental radica en agilizar el intercambio de información entre diferentes partes del sistema, especialmente en entornos distribuidos o cuando es necesario compartir datos entre diversas capas de una aplicación (Blancarte, 2021).

2.3. IDE de Desarrollo

2.3.1. JeatBrains Rider

Es una herramienta destacada en el entorno .NET, brinda a los desarrolladores la capacidad de crear aplicaciones de manera rápida y efectiva en múltiples plataformas. Su versatilidad permite el desarrollo para .NET, ASP.NET, .NET Core, Xamarin y Unity, sin importar el sistema operativo utilizado ya sea Windows, macOS o Linux. Rider se distingue por ofrecer una experiencia de edición y acceso al código amplia y precisa, cubriendo una variedad de lenguajes esenciales en el desarrollo en .NET, desde C#, VB.NET y F# hasta ASP.NET Razor syntax, JavaScript, TypeScript, XAML, XML, HTML, CSS, SCSS, JSON y SQL (Jetbrains, 2023).

En la Figura 3, se puede observar claramente el distintivo representativo de JetBrains Rider. Este IDE emerge como una solución integral dentro del conjunto de productos de JetBrains, diseñado para optimizar el proceso de desarrollo de software y ofrecer una experiencia eficiente y colaborativa a los desarrolladores.

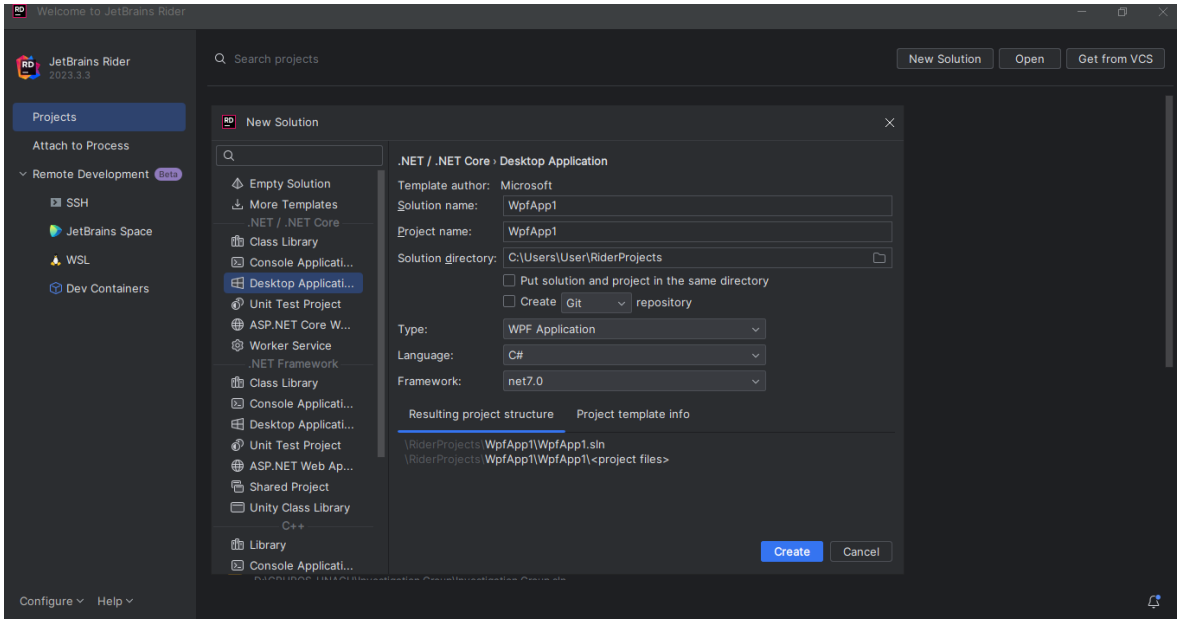


Figura 3. Entorno de desarrollo en JetBrains Rider
Fuente: (Jetbrains, 2023)

2.3.2. Funcionalidades

En la Tabla 2 se detalla las múltiples funcionalidades del IDE Rider, una herramienta esencial en el ámbito del desarrollo de software.

Tabla 2. Descripción de las funcionalidades

Funcionalidades	Descripción
Análisis del código	Rider lleva a cabo un análisis en tiempo real de más de 2200 aspectos del código, ofreciendo soluciones automáticas para corregir problemas a nivel individual o global.
Edición de código	Proporciona diversas funcionalidades de edición de código inteligente, como completado de código, importación automática de espacios de nombres, inserción automática de llaves, resaltado de delimitadores coincidentes, reorganización de código, plantillas en vivo y más.
Refactorización	Con más de 60 refactorizaciones de ReSharper integradas, Rider permite cambiar nombres, extraer métodos, interfaces y clases, mover y copiar tipos, utilizar sintaxis alternativas, entre otras acciones contextuales disponibles a través del menú AltIntro.
Ejecutor de pruebas de unidad	Facilita la ejecución y depuración de pruebas de unidad dirigidas a .NET Framework, .NET Core y Mono. Marca clases y métodos de prueba con un icono en el

	medianil del editor, permitiendo gestionar pruebas, explorarlas, agruparlas y visualizar sus resultados.
Navegación y búsqueda	Ofrece accesos rápidos para buscar y navegar hacia archivos, tipos o miembros en la base de código. El acceso directo Search Everywhere permite encontrar usos de símbolos y realizar búsquedas en configuraciones y acciones.
Depuración	Incluye un depurador compatible con aplicaciones .NET Framework, Mono y .NET Core. Permite crear configuraciones de depuración, gestionar puntos de interrupción, evaluar expresiones y explorar subprocesos.
Control de la versión	Compatibilidad integrada para Git, Subversion, Mercurial, Perforce y TF, Rider facilita el control de versiones sin configuración adicional y admite otros sistemas a través de complementos.
Desarrollo web	Soporte para JavaScript, TypeScript, HTML, CSS, y Sass, Rider permite el desarrollo de aplicaciones web, móviles y de escritorio. Compatible con Node.js, React, Angular, Vue.js y diversas herramientas de desarrollo web.
Bases de datos y SQL	Permite trabajar con SQL y bases de datos directamente, conectándose a bases de datos, editando esquemas y datos tabulares, realizando consultas y analizando esquemas con diagramas UML.
Complementos	Es compatible con una amplia gama de complementos desarrollados para la plataforma IntelliJ y ReSharper. Además de los complementos incluidos, ofrece la opción de instalar más según las necesidades.

Fuente: Adaptado de (Jetbrains, 2023).

2.4. Lenguaje de programación

2.4.1. C#

Microsoft desarrolló C# como una variante del lenguaje C, manteniendo una sintaxis muy similar pero enfocada en la programación orientada a objetos. Este lenguaje es ampliamente utilizado en sectores como el desarrollo de videojuegos, la impresión 3D, la robótica, y la creación de aplicaciones para plataformas web y móviles, incluyendo sistemas operativos como Microsoft, iOS y Android. C# es un lenguaje multiplataforma concebido para trabajar con la infraestructura del lenguaje común, una norma estandarizada que define un entorno virtual para ejecutar aplicaciones (Larrea et al., 2022) .

La Figura 4 ilustra un ejemplo de código común y fácil de comprender del lenguaje de programación C#, destacando su sencillez y eficacia en el desarrollo de software.

Figura 4. C#
Fuente: (Microsoft, 2023)

Su sintaxis es similar a Java y actualmente se ha convertido en un lenguaje tremendamente flexible, entre sus características (Dongee, 2023) destaca las siguientes:

- Programación orientada a objetos: C# es un lenguaje orientado a objetos, significa que ayuda a crear código reutilizable asignando objetos a clases relacionadas y creando relaciones entre ellos. Esto facilita la estructuración lógica de los programas y su mantenimiento a lo largo del tiempo.
- Fácil mantenimiento: La sintaxis del lenguaje es fácil de aprender y comprensible, facilita el mantenimiento y la depuración del código existente. Esto facilita la vida a los desarrolladores que necesitan hacer cambios en aplicaciones existentes.
- Funciones de seguridad integradas: C# ofrece potentes funciones de seguridad integradas, como el control de acceso, la criptografía y la comprobación robusta de tipos. Éstas ayudan a mantener la aplicación a salvo de ataques maliciosos garantizando que sólo los usuarios autorizados puedan acceder a la aplicación.
- Alto rendimiento: C# admite una gestión eficaz de la memoria, lo que lo hace adecuado para desarrollar aplicaciones de alto rendimiento con limitaciones de tiempo críticas.
- Desarrollo multiplataforma: .NET Framework permite a los desarrolladores crear aplicaciones que funcionen en varias plataformas sin tener que reescribir el código para cada una de ellas.

2.5. Framework y Bibliotecas

2.5.1. .NET 7

La versión 7 de .NET presentada por Microsoft añade nuevas características tanto para aplicaciones escritas en C# como en F#. Además, ofrece mejoras para tecnologías como .NET MAUI, ASP.NET Core/Blazor, Web API, entre otros. Esta actualización facilita la contenerización de proyectos .NET 7, la configuración de secuencias de trabajo de CI/CD en GitHub Actions (Douglas et al., 2022).

Además, .NET 7 representa una versión importante que mejora la calidad de vida del desarrollador al potenciar aspectos fundamentales como el rendimiento, la funcionalidad y la usabilidad. Se espera que estas nuevas características continúen enriqueciendo la

experiencia de uso de la Plataforma .NET. La Figura 5 muestra las plataformas adaptables del framework.

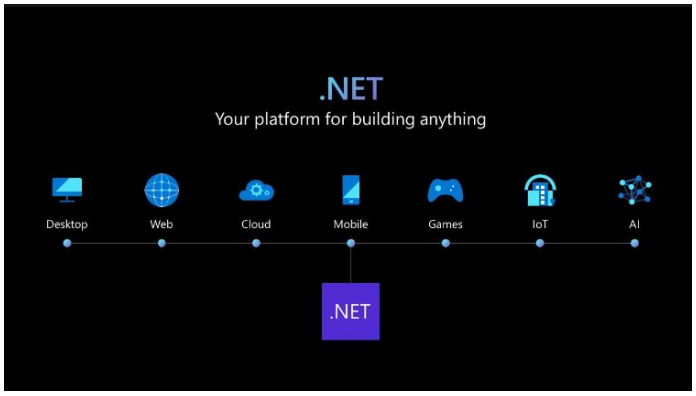


Figura 5. Plataformas de .NET
Fuente: (Douglas et al., 2022)

En la Tabla 3 se distingue varias características con respecto a sus anteriores versiones:

Tabla 3. 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	APIs Web con menos optimizaciones	Soporte para APIs Web, pero sin las

	facilidad de desarrollo de APIs Web.	comparadas con .NET 7.	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.

Fuente: (Douglas et al., 2022)

2.5.2. Bibliotecas estándares de .NET

Estas bibliotecas, comúnmente conocidas como bibliotecas de clases, permiten la segmentación de funcionalidades valiosas en módulos que pueden ser aprovechados por diversas aplicaciones. También son útiles para cargar funcionalidades innecesarias o desconocidas al inicio de la aplicación. Existen tres tipos de bibliotecas de clases disponibles (Microsoft Build, 2023):

- **Bibliotecas de clases específicas de la plataforma**

Las bibliotecas específicas de la plataforma se enlazan a una única plataforma de .NET y, por tanto, pueden tomar dependencias significativas de un entorno de ejecución conocido. Este entorno expone un conjunto conocido de API (API de .NET y SO) y gestiona el estado esperado, por ejemplo, el Registro de Windows.

Las bibliotecas específicas de la plataforma han sido el tipo de biblioteca de clases principal de .NET Framework. Incluso con la aparición de otras implementaciones de .NET, las bibliotecas específicas de la plataforma continúan siendo el tipo de biblioteca dominante.

- **Bibliotecas de clases portables**

Las bibliotecas portables son compatibles con varias implementaciones de .NET. Pueden tomar dependencias en un entorno de ejecución conocido; en cambio, el entorno es sintético y está generado por la intersección de un conjunto de implementaciones concretas de .NET. Las hipótesis de plataforma y API expuestas son un subconjunto de lo que estaría disponible para una biblioteca específica de la plataforma.

- **Bibliotecas de clases .NET Standard**

Las bibliotecas de .NET Standard representan una evolución respecto a las bibliotecas específicas de la plataforma y portables. Y están diseñadas para ser específicas de la plataforma, aprovechando todas las capacidades de la plataforma subyacente, también son portables, lo que significa que pueden funcionar en todas las plataformas compatibles.

2.6. Paquete Nuget

Es un administrador de paquetes que posibilita a los desarrolladores crear, compartir y aprovechar código valioso. Con frecuencia, este código se encapsula en "paquetes" que comprenden tanto código compilado, como archivos DLL, así como otros elementos esenciales para los proyectos que incorporan estos paquetes. En el contexto de .NET, el mecanismo respaldado por Microsoft para la compartición de código se denomina NuGet (Microsoft, 2023).

Un paquete NuGet se presenta como un archivo comprimido en formato ZIP con la extensión .nupkg, que alberga tanto el código compilado en forma de archivos DLL como otros archivos asociados a dicho código. Además, incluye un manifiesto descriptivo que abarca información crucial, como el número de versión del paquete. Los desarrolladores que deseen compartir su código crean estos paquetes y los distribuyen en un servidor público o privado. Los usuarios que deseen incorporar estos paquetes en sus proyectos pueden obtenerlos del host correspondiente, integrarlos en sus proyectos y luego acceder a la funcionalidad del paquete en el código del proyecto. En este proceso, NuGet se encarga de gestionar todos los detalles intermedios. NuGet simplifica la capacidad de alojar paquetes de forma privada en diversos entornos, ya sea en la nube, como en Azure DevOps, en una red privada o incluso en el sistema de archivos local (Microsoft, 2023).

2.7. Azure Devops

DevOps se define como una perspectiva en el desarrollo de software que promueve una colaboración estrecha entre los departamentos de desarrollo, operaciones y otros equipos afines, con el fin de lograr objetivos comerciales compartidos. Esta metodología surge de la fusión de los conceptos "desarrollo" y "operaciones", concentrándose en la cooperación entre diversas partes interesadas en el proceso de desarrollo de software. La finalidad principal de DevOps radica en proporcionar de manera continua software de alta calidad a los clientes, aprovechando los sistemas más eficientes disponibles (Lotriet & Mudadi, 2023).

2.8. Metodología SCRUM

Scrum es un enfoque ágil de desarrollo que se inspiró en el deporte del rugby, en donde equipos autónomos colaboran de manera autogestionada, priorizando la entrega incremental de productos y valor. Se basa en el conocimiento tácito y la creatividad de las personas, fomentando la colaboración abierta y el aprendizaje continuo, mientras superpone las distintas fases del desarrollo para maximizar la eficiencia y la adaptabilidad (Palacio, 2022). A continuación, se detalla los componentes de Scrum:

2.8.1. Roles

En Scrum, se identifican tres roles principales:

- Scrum Team: compuesto por los miembros colaboradores que buscan mejorar la eficacia en las tareas y procesos.
- Product Owner: actúa como representante del cliente y se encarga de asegurar el cumplimiento de los objetivos del proyecto.
- Scrum Master: Responsable de apoyar al equipo Scrum en el logro de las metas del proyecto, facilitando la resolución de problemas y asegurando una comunicación fluida entre los participantes durante cada sprint.

2.8.2. Artefactos

Los artefactos que forma a Scrum son:

- Producto Backlog: una lista que registra y prioriza los requerimientos del cliente, generalmente conocidos como historias de usuario, que se desglosan en tareas más pequeñas.
- Sprint backlog: es una lista de tareas a realizar durante un sprint para lograr un incremento previsto.
- Incremento: el resultado final de cada sprint.

2.8.3. Eventos

Los eventos dentro de Scrum incluyen:

- Sprint: un período de tiempo fijo, típicamente de 1 a 4 semanas, durante el cual el equipo se enfoca en las actividades necesarias para lograr los objetivos del sprint.
- Sprint Planning; una reunión en la que se establecen los objetivos y las tareas para el sprint en curso.
- Daily Scrum: reunión breve que se lleva a cabo diariamente, aproximadamente durante 15 minutos, para sincronizar al equipo y actualizar sobre el progreso del sprint, así como para planificar las actividades del día.
- Sprint Review: una reunión al finalizar cada sprint, que suele durar una o dos horas, donde se examina el progreso y se evalúa si se han alcanzado los objetivos establecidos.

- Sprint Restrospective: una reunión que tiene lugar después de la revisión de cada sprint, donde se discuten los desafíos enfrentados durante el sprint y se proponen soluciones para mejorar el proceso en el futuro.

En la Figura 6 se demuestra el proceso de la metodología SCRUM.

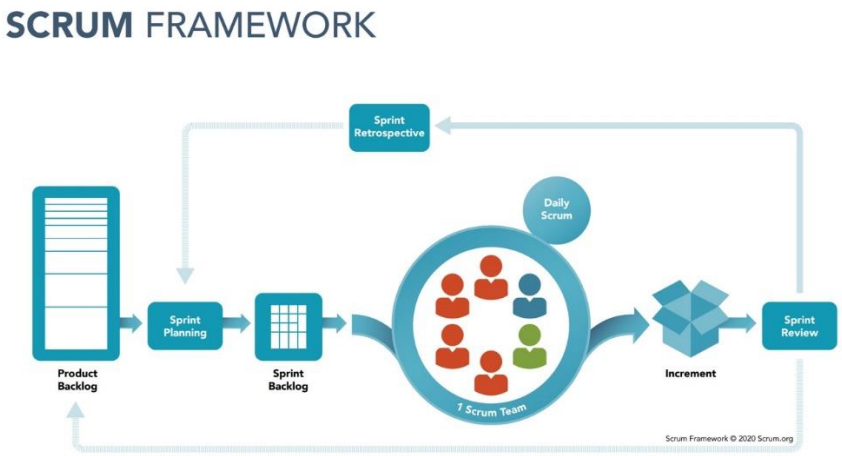


Figura 6. Proceso SCRUM
Fuente: (Francia, 2017)

2.8.4. Impacto de la metodología ágil Scrum en el proceso de desarrollo de software

El empleo de Scrum se hace notorio al examinar el panorama del desarrollo ágil del año 2022, destacándose como una metodología líder y ampliamente preferida dentro de la comunidad Agile (State of Agile, 2022). La comparación entre Scrum y otros marcos de trabajo ágil se ilustra en la Figura 7.

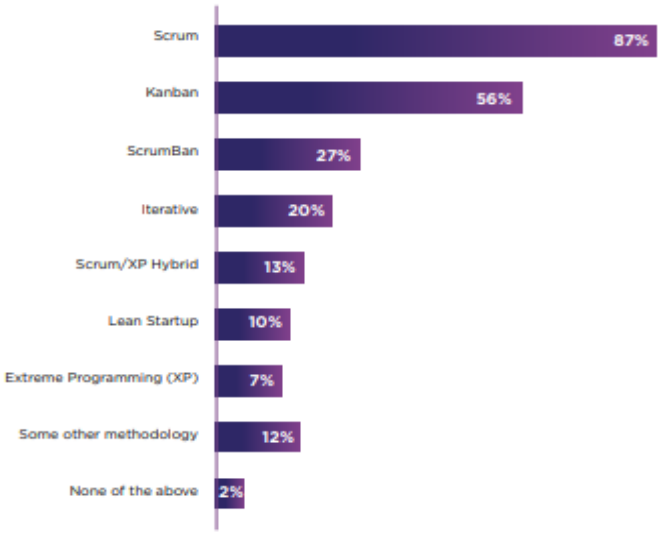


Figura 7. Marcos de trabajo ágil más usados
Fuente: (State of Agile, 2022)

2.9. Modelo de Calidad FURPS

La calidad del software se relaciona con el nivel de cumplimiento de las características fundamentales que un sistema computacional debe tener a lo largo de su ciclo de vida. Estas características juegan un papel crucial en asegurar la confiabilidad del sistema para el cliente, a su vez mejora su satisfacción respecto a la funcionalidad y eficiencia del sistema desarrollado (Callejas-Cuervo et al., 2017).

El modelo FURPS, desarrollado por Hewlett-Packard (HP) en 1987, es un marco de calidad de software que abarca una variedad de factores y atributos esenciales para evaluar y gestionar la calidad de un producto de software a lo largo de su ciclo de vida. FURPS es un acrónimo que representa Funcionalidad, Usabilidad, Confiabilidad, Rendimiento y Soporte. Cada uno de estos aspectos se enfoca en características específicas de calidad que son críticas para el éxito y la satisfacción del usuario final (Callejas-Cuervo et al., 2017).

2.9.1. Parámetros de medición del rendimiento

En términos de rendimiento, según el modelo FURPS, es necesario evaluar indicadores como la velocidad de procesamiento, el tiempo de respuesta y el consumo de recursos (Manobanda, 2020):

- Eficacia: La eficacia se refiere a la capacidad del sistema para completar tareas asignadas y alcanzar objetivos establecidos.
- Tiempo de Respuesta: indica el tiempo necesario para completar una tarea, considerando aspectos como el acceso a la memoria RAM, el disco, las operaciones de entrada/salida y los recursos del sistema operativo.
- Utilización de Recursos: se refiere a la cantidad de recursos software requeridos para el funcionamiento adecuado del sistema.

2.9.2. Valores de ponderación del rendimiento según el modelo FURPS

En la Tabla 4 se detallan las dimensiones de ponderación establecidas por el modelo de calidad de software FURPS para los parámetros del indicador rendimiento:

Tabla 4. Valores de ponderación del rendimiento según el modelo FURPS

Parámetros	Ponderación
Eficacia	95%
Tiempo de Respuesta	5sg
Utilización de Recursos	25%

Fuente: (Alvarado, 2023)

2.10. Apache JMeter

Es una herramienta de código abierto desarrollada por la Apache Software Foundation para realizar pruebas de rendimiento y carga en aplicaciones web. Permite simular el comportamiento de múltiples usuarios concurrentes, facilita la evaluación del rendimiento y

la estabilidad de aplicaciones bajo diferentes condiciones de carga. JMeter es altamente configurable y puede utilizarse para probar una variedad de protocolos y tecnologías web, convirtiéndose en una herramienta versátil para desarrolladores, ingenieros de rendimiento y equipos de calidad de software (Sheti, 2022).

CAPÍTULO III. METODOLOGÍA

Para esta investigación, se adoptó un enfoque cuantitativo con el objetivo de obtener resultados objetivos y medibles sobre el rendimiento del módulo de proyectos de investigación, basándose en los criterios establecidos por el modelo de calidad de software FURPS.

3.1. Tipo y diseño de Investigación

3.1.1. Según la fuente de la investigación

- Investigación documental: Esta investigación es de tipo documental o bibliográfica, la búsqueda implicó la exploración de varias fuentes, como libros, publicaciones científicas, artículos y otros recursos pertinentes para respaldar la investigación sobre los microservicios con información actualizada.

3.1.2. Según el objeto de estudio

- Investigación aplicada: Esta investigación se caracteriza como aplicada, dado que se emplearon conocimientos teóricos previamente investigados para implementar el módulo de proyectos de investigación del sistema de Gestión de Investigación de la UNACH.
- Investigación descriptiva: Esta investigación adoptó una naturaleza descriptiva al examinar y detallar los resultados derivados de la evaluación del rendimiento del módulo mediante el uso del modelo de calidad FURPS.

3.2. Instrumentos de recolección de Datos

Para la recolección de datos se utilizó como herramienta apache JMeter.

3.3. Población de estudio y tamaño de muestra

La población de este estudio se consideró infinita, debido que se recopilaron diversos datos sobre los indicadores de rendimiento del modelo de calidad FURPS utilizando la herramienta JMeter, lo que eliminó la necesidad de calcular un tamaño de muestra específico.

3.4. Identificación de variables

3.4.1. Variable Independiente

Microservicios

3.4.2. Variable Dependiente

Rendimiento del Módulo de proyectos del sistema de Gestión de Investigación de la UNACH

3.5. Operacionalización de Variables

En la Tabla 5 se observa a detalle la operacionalización de cada una de las variables.

Tabla 5. Operacionalización de Variables

Pregunta de investigación	Tema	Objetivos	Variables	Conceptualización	Dimensión	Indicadores
¿Cómo los microservicios influirán en el rendimiento del módulo de proyectos del sistema de Gestión de Investigación de la UNACH?	Microservicios para el módulo de proyectos del sistema de Gestión de Investigación de la UNACH.	**General:** Implementar microservicios para el módulo de proyectos del sistema de Gestión de Investigación de la UNACH.	**Independiente:** Microservicios	La arquitectura de microservicios se emplea para construir una aplicación mediante elementos autónomos que operan individualmente conectados con una interfaz, llevando a cabo cada función de la aplicación como si fueran servicios separados (Ortega, 2020).	Arquitectura	**Independiente:** ✓ Número de diagramas generados. ✓ Número de esquemas diseñados.
		Específicos: • Investigar las herramientas adecuadas para el desarrollo microservicios para el módulo de proyectos del sistema de Gestión de Investigación de la UNACH. • Desarrollar el módulo de proyectos del sistema de Gestión de Investigación de la UNACH. • Evaluar el rendimiento del módulo de proyectos mediante el modelo de calidad FURPS.	**Dependiente:** Rendimiento del Módulo de proyectos del sistema de Gestión de Investigación de la UNACH	Se refiere al conjunto de acciones y procesos involucrados en la supervisión y administración de los proyectos de investigación en la UNACH. Esta variable está vinculada a la organización, seguimiento y control de proyectos de investigación en una institución académica.	Rendimiento	**Dependiente:** Modelo de calidad de FURPS: ✓ % de eficacia. ✓ % de tiempo de respuesta. ✓ % de utilización de recursos.

3.6. Metodología de desarrollo de software

Dentro de este trabajo se utilizó la metodología SCRUM en la planificación de actividades del desarrollo de microservicios para el módulo de proyectos del sistema de Gestión de Investigación de la UNACH.

3.6.1. Inicio

En esta fase se identificó el personal involucrado. Por otro lado, se definió el análisis de requerimientos para el desarrollo del módulo.

Equipo Scrum

Según la metodología Scrum, se asignan roles específicos a los miembros del equipo, detallados en la Tabla 6.

Tabla 6. Equipo Scrum

Rol	Responsable
Product Owner	Ing. Alex Asitimbay
Scrum Master	Ing. Alex Buñay
Team	Leidy Angamarca
	Jonathan Pilamunga

Requerimientos funcionales

Los requisitos funcionales fueron determinados por la Dirección de Investigación de la UNACH, en la Tabla 7 se describe los requisitos recopilados.

Tabla 7. Requerimientos funcionales

Identificación del requerimiento	**RF01**
Nombre del requerimiento	Perfil de proyectos de investigación.
Descripción del requerimiento	El usuario podrá llenar la información requerida de acuerdo con su proyecto de investigación.
Prioridad del requerimiento	Alta
Identificación del requerimiento	**RF02**
Nombre del requerimiento	Seguimiento de proyectos de investigación
Descripción del requerimiento	El analista determinara si el proyecto es vigente o no

Prioridad del requerimiento	Alta
Identificación del requerimiento	**RF03**
Nombre del requerimiento	Estructura del equipo proyectos de investigación
Descripción del requerimiento	Mediante el reglamento de investigación se determina los roles de equipo
Prioridad del requerimiento	Alta
Identificación del requerimiento	**RF04**
Nombre del requerimiento	Matrices de proyectos de investigación
Descripción del requerimiento	El usuario podrá llenar las matrices de proyectos mediante el módulo.
Prioridad del requerimiento	Alta

Requerimientos no funcionales

Los requisitos no funcionales se enfocan en los aspectos generales del desempeño del sistema. Incluyen especificaciones como seguridad, disponibilidad, compatibilidad y escalabilidad, etc. En la Tabla 8 se identifica los requisitos.

Tabla 8. Requerimientos no funcionales

Requerimiento	Descripción del requerimiento	Categoría
RNF01	Estas especificaciones describen el método de gestión y respuesta de la aplicación ante posibles errores, resaltando la importancia de un manejo de errores constante y preciso para garantizar un funcionamiento eficiente.	Control de errores
RNF02	El desarrollo de interfaces de usuario suele ser visto como un componente de la fase de especificación de requerimientos.	Interfaces de usuario
RNF03	Estos criterios establecen que, aunque las aplicaciones no son completamente libres de errores, sus fallos deben mantenerse dentro de ciertos márgenes preestablecidos.	Confiabilidad
RNF04	Se detalla la necesidad de que la aplicación se mantenga activa y accesible durante un tiempo específico, subrayando la relevancia de asegurar una disponibilidad continua y efectiva para los usuarios.	Disponibilidad

RNF05	Este requisito destaca la necesidad de implementar protocolos de seguridad para procedimientos que utilizan información crítica, como las contraseñas de acceso al software, enfatizando la protección rigurosa de tales datos.	Seguridad
RNF06	Estos criterios definen parámetros para analizar cómo la aplicación desempeña sus tareas de manera eficiente y efectiva. Se enfocan en evaluar la velocidad de ejecución, la capacidad de respuesta frente a diferentes niveles de trabajo y la eficiencia en la utilización de recursos, todo ello con el objetivo de asegurar un funcionamiento fluido y sin interrupciones.	Rendimiento

3.6.2. Planificación y estimación

En esta fase, se elaboró un plan detallado para el desarrollo del proyecto de investigación.

Product Backlog

La Tabla 9 establece un inventario completo de las actividades planeadas para la ejecución del proyecto. Las tareas están clasificadas como Historias Técnicas (HT) e Historias de Usuario (HU), cada una con su propia numeración. Además, se incluyó una estimación del esfuerzo requerido para cada tarea calificado de 1 (menos esfuerzo) y al 5 (más esfuerzo).

Tabla 9. Product Backlog

Ítem	Tarea	Esfuerzo
HT-01	Análisis de los requerimientos funcionales y no funcionales del módulo.	5
HT-02	Establecer la arquitectura del módulo.	3
HT-03	Diseño de la base de datos (Modelo entidad-relación, Modelo relacional y físico)	5
HT-04	Instalación y configuración de herramientas para el desarrollo del módulo.	4
HU-01	Desarrollar la base de datos.	4
HU-02	Desarrollar la sección de dominio para el módulo.	4
HU-03	Desarrollar la sección de infraestructura para el módulo.	3
HU-04	Desarrollar la sección de api para el módulo.	5
HU-05	Desarrollar la sección de aplicación para el módulo.	5
HU-06	Evaluar el módulo 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

Es una lista de elementos de trabajo seleccionados del Product Backlog para ser abordados durante un sprint. En la Tabla 10 se especifican las actividades del Sprint Backlog.

Tabla 10. Spring 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 módulo.	X															
HT-02	Establecer la arquitectura del módulo.		X														
HT-03	Diseño de la base de datos (Modelo entidad-relación, Modelo relacional y físico).			X													
HT-04	Instalación y configuración de herramientas para el desarrollo del módulo.				X												
	SPRINT 2																
HU-01	Desarrollar la base de datos.					X											
HU-02	Desarrollar la sección de dominio para el módulo.						X	X									
HU-03	Desarrollar la sección de infraestructura para el módulo.								X								
	SPRINT 3																
HU-04	Desarrollar la sección de api para el módulo.									X	X						
HU-05	Desarrollar la sección de 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

3.6.3. Implementación

Durante esta etapa, se inició el desarrollo de la aplicación bajo la arquitectura de microservicios usando como IDE a JetBrains Rider.

Diagrama Relacional de la base de datos

Este modelo se fundamenta en la representación de la información a través de tablas, donde cada tabla representa una entidad específica, y cada fila en dicha tabla representa una instancia única de esa entidad. Las relaciones entre estas entidades se establecen mediante claves primarias y foráneas, posibilitando una conexión lógica y significativa entre los datos almacenados en diversas tablas.

La Figura 8 ofrece una visualización de la estructura y las relaciones clave presentes en la base de datos relacional que sustenta el módulo de proyectos.

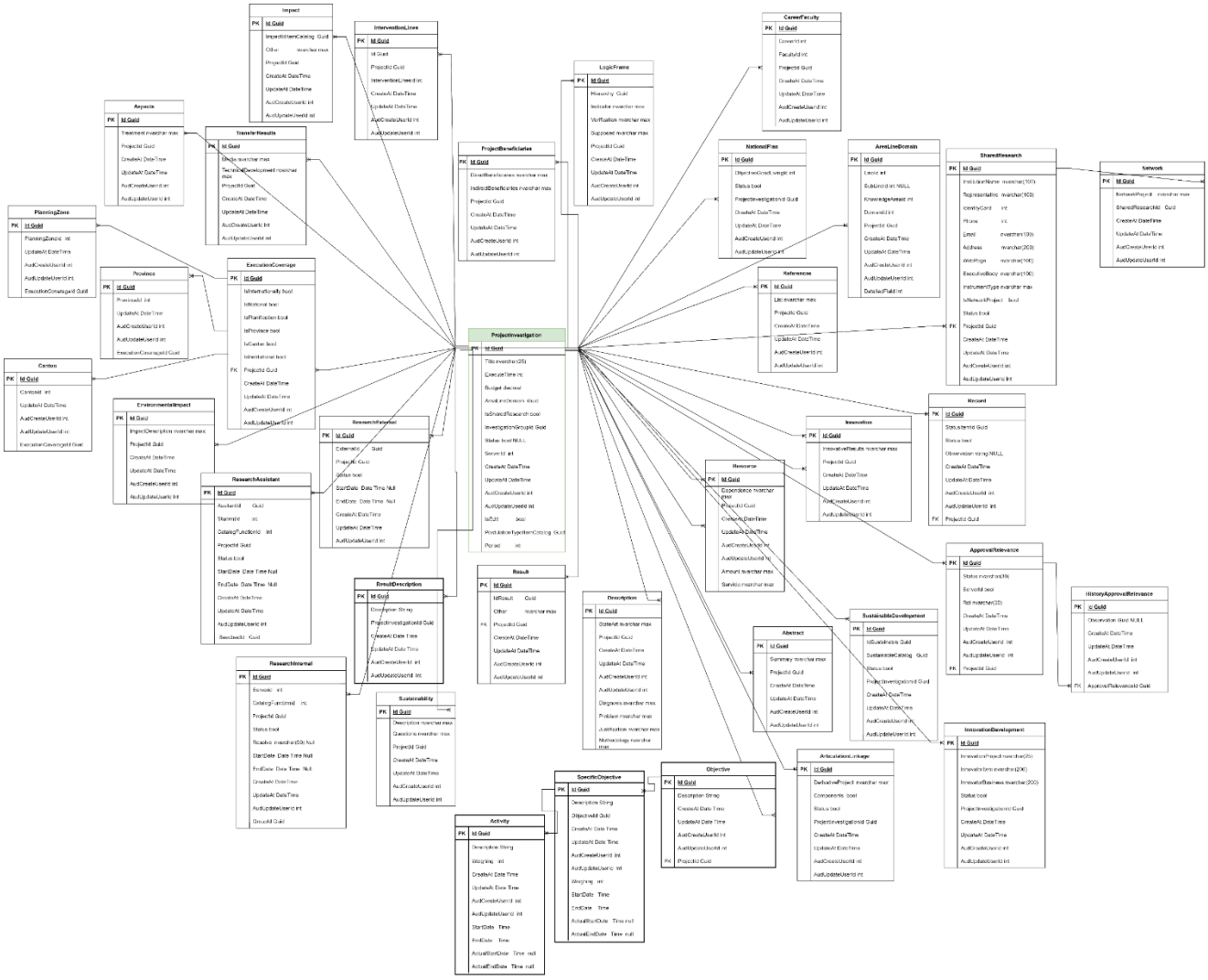


Figura 8. Diagrama relacional

Diagrama Físico de la base de datos

La representación visual del diagrama físico de la base de datos ofrece una visión detallada de cómo la estructura conceptual se concreta en términos de almacenamiento real en el sistema. En este esquema se representan las tablas, junto con sus atributos, tipos de datos y las relaciones específicas entre ellas, como se visualiza en la Figura 9.

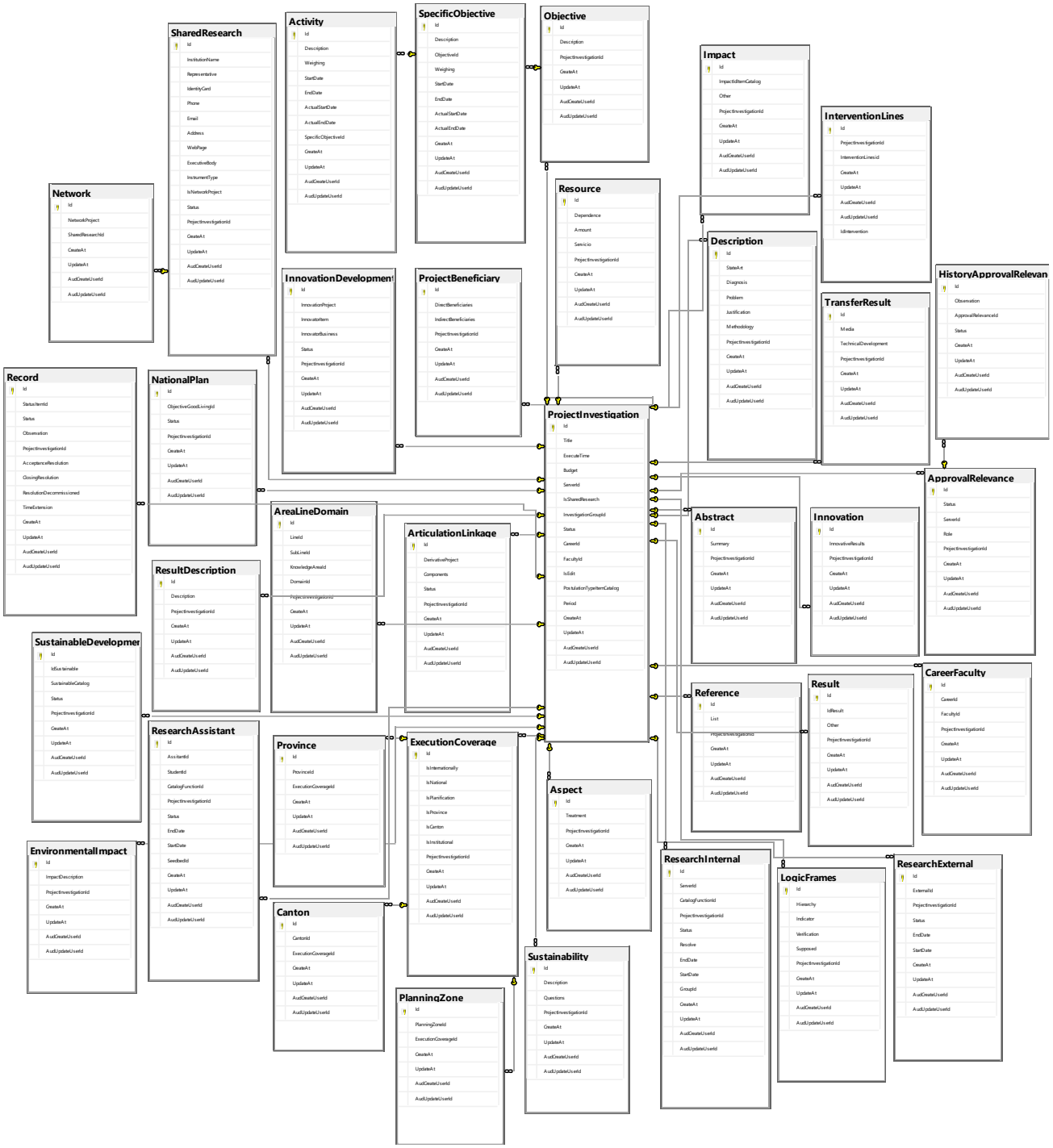


Figura 9. Diagrama físico de la base de datos

Diccionario de datos

Es una herramienta que sirve para organizar y documentar información detallada sobre los datos utilizados en el desarrollo del producto de software. A continuación, en la Tabla 11 se distingue a la tabla principal de proyectos, identificada en la base de datos como "ProjectInvestigation". En esta tabla los atributos clave, incluyendo el nombre, el tipo de dato correspondiente, y una descripción que proporciona información adicional sobre cada uno de ellos.

Tabla 11. Tabla Principal 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	Status	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

En el Anexo 1, se incluye el diccionario de datos que comprende la base de datos abordada en este trabajo de investigación.

Diagramas de casos de uso

Son de ayuda en el desarrollo de software al ofrecer una representación visual de las interacciones entre usuarios y el producto que se desarrolla. Su utilidad radica en la

identificación y documentación de requisitos funcionales clave al mostrar gráficamente cómo los usuarios se relacionan con las diversas funciones del sistema.

En la Figura 10 se ilustra el caso de uso que comienza cuando el usuario envía su perfil, el cual pasa por un proceso de análisis. El analista examina la información e interactúa con el usuario para afinar detalles del perfil del proyecto.

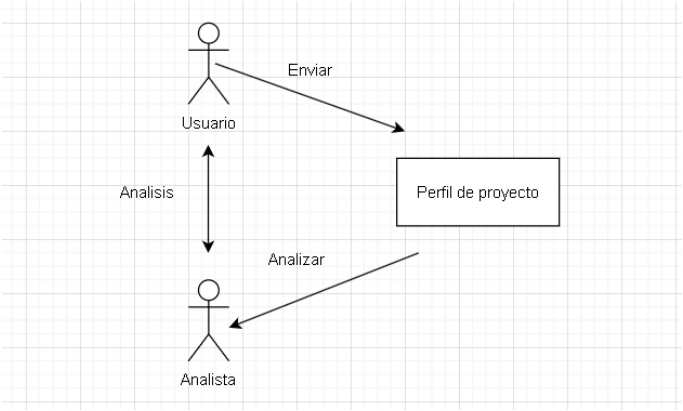


Figura 10. Caso de uso de análisis de perfil de proyecto

El caso de uso de la Figura 11, indica las acciones específicas que el director debe realizar en el sistema para gestionar la información de proyectos de investigación.

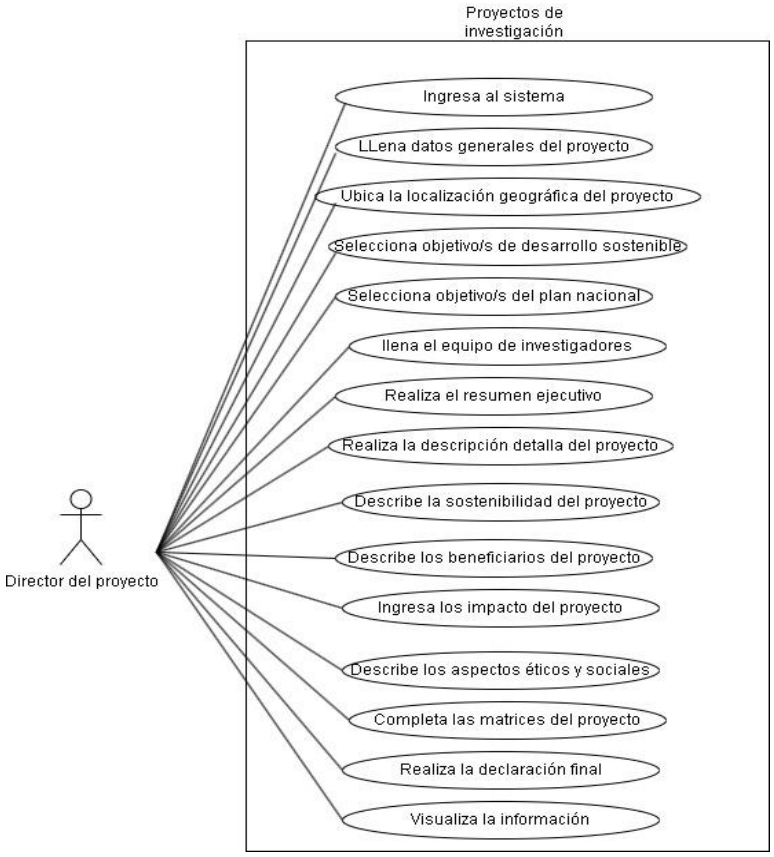


Figura 11. Caso de uso para la Gestión de proyectos

El caso de uso de la Figura 12, se visualiza el proceso de revisión y resolución para el proyecto de investigación que realiza el analista.

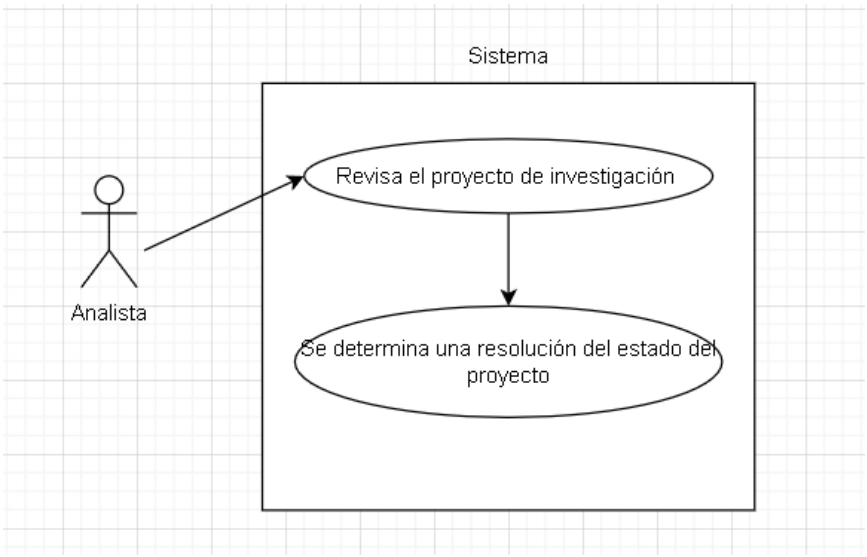


Figura 12. Caso de uso del proceso de revisión y resolución para proyectos

Arquitectura basada en microservicio

Para aplicar la arquitectura de microservicios se elaboró un esquema general que incluye tanto el módulo de proyectos como los microservicios correspondientes, proporcionando una visión detallada de sus conexiones y funcionamiento dentro del sistema como se observa en la Figura 13.

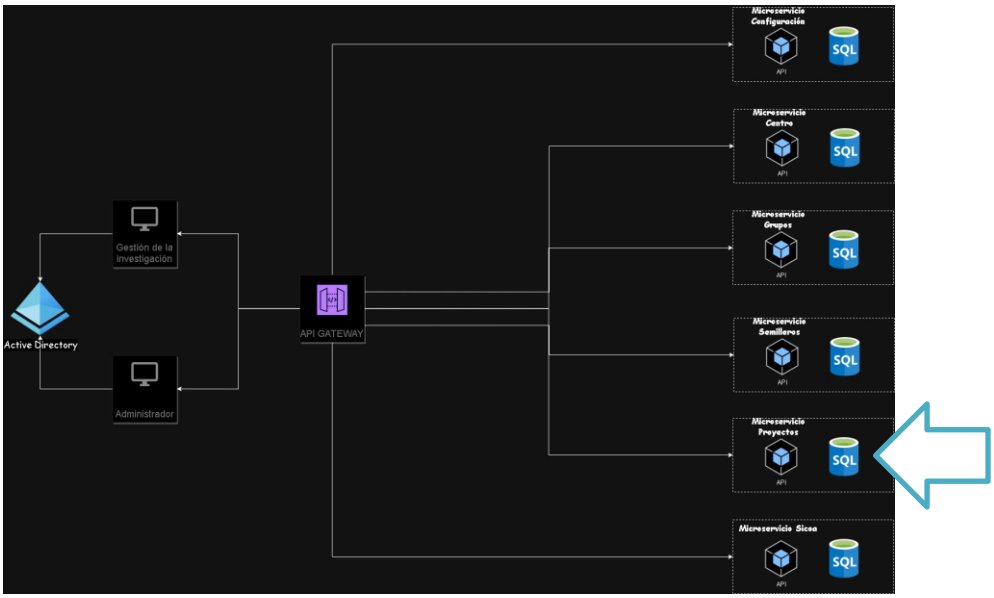


Figura 13. Diseño general del sistema

La implementación del microservicio se llevó a cabo utilizando .Net 7 junto con sus frameworks principales, ASP.Net Core y Entity Framework Core en el IDE Jetbrains Rider. Se utilizó una base de datos SQL llamada 'ProyectoInvestigación' para almacenar la

información relevante. Además, para definir las API REST y proporcionar una descripción clara de las funcionalidades ofrecidas por el servicio en relación con cada entidad, se utilizó Swagger. En la Figura 14 se demuestra el microservicio lógico para el módulo de proyectos de Investigación.

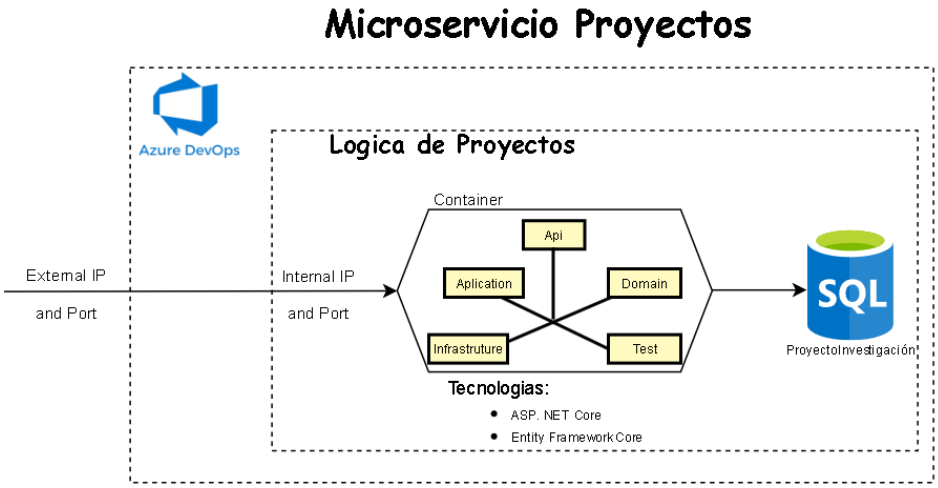


Figura 14. Diseño lógico del microservicio Proyectos

Para iniciar el desarrollo del microservicio, el primer paso fue descargar una plantilla preconfigurada de Azure DevOps. Esta plantilla está específicamente diseñada para proyectos de microservicios con dependencias .Net 7 y equipada con una arquitectura hexagonal, lo que facilita la implementación y asegura una estructura coherente desde el inicio. En la Figura 15 se presenta el procedimiento realizado.

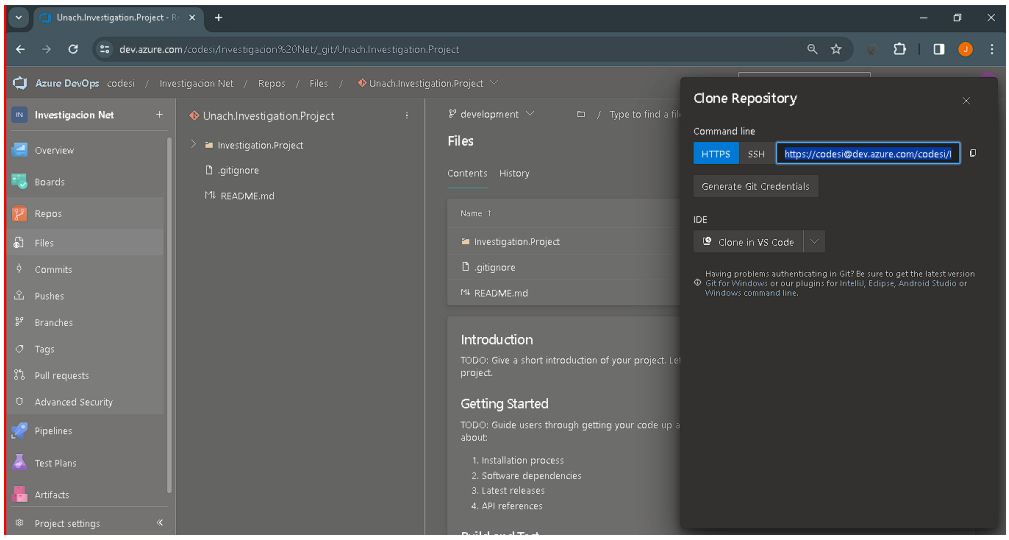


Figura 15. Descarga de plantilla

Luego, se procedió a descargar los paquetes NuGet personalizados de Azure DevOps, proporcionados por los técnicos del área de investigación, para el desarrollo del módulo. Estos paquetes fueron adaptados específicamente para sus necesidades y entorno de desarrollo, facilitando la implementación del proyecto como se refleja en la Figura 16.

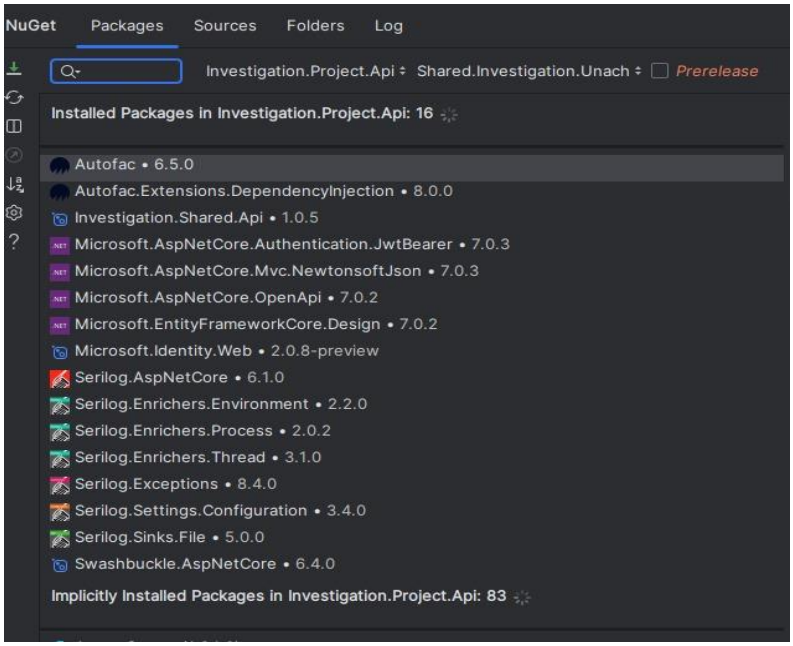


Figura 16. Paquetes Nuget

Luego, se crearon cada entidad junto con sus respectivos parámetros. Además, se diseñó las interfaces para los repositorios de cada entidad, siguiendo el patrón de diseño Repositorio. Estas interfaces definieron las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) según la lógica de negocios de cada entidad, lo que permitió un desarrollo más centrado en los requisitos específicos del proyecto tal como se aprecia en la Figura 17.

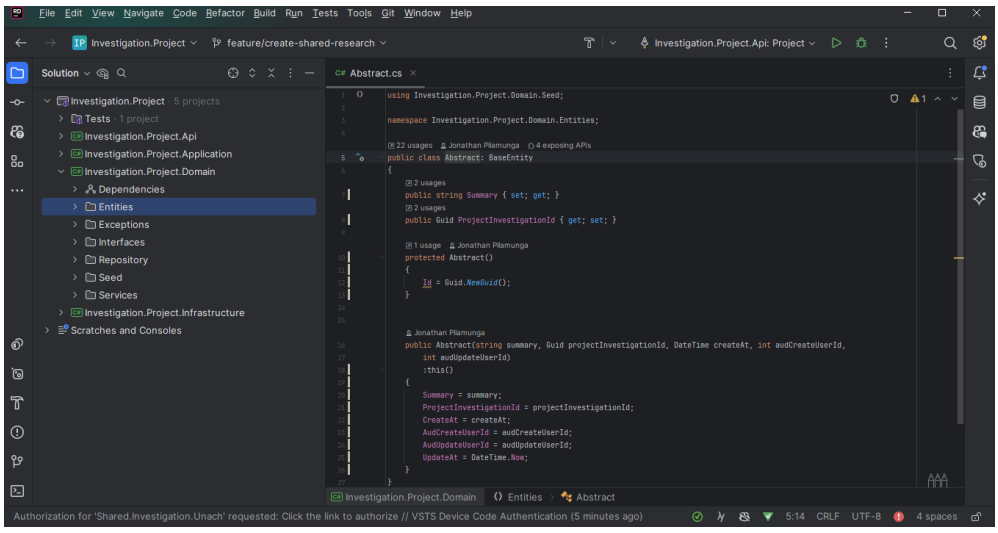


Figura 17. Sección Dominio

La sección de infraestructura, donde se establece la estructura para interactuar con la base de datos utilizando el enfoque de Code-First. Esto simplifica la gestión de la base de datos sin necesidad de escribir directamente sentencias SQL para la creación o actualización de la base de datos, así como para otras funciones relacionadas.

Para ello se creó un contexto de base de datos, clase que hereda de DbContext, siguiendo el patrón de diseño Unit Of Work. Este contexto, proporcionado por Entity Framework Core,

sirve como puerta de entrada a la base de datos y gestiona la configuración y las operaciones de acceso a datos. En la Figura 18 se contempla lo mencionado.

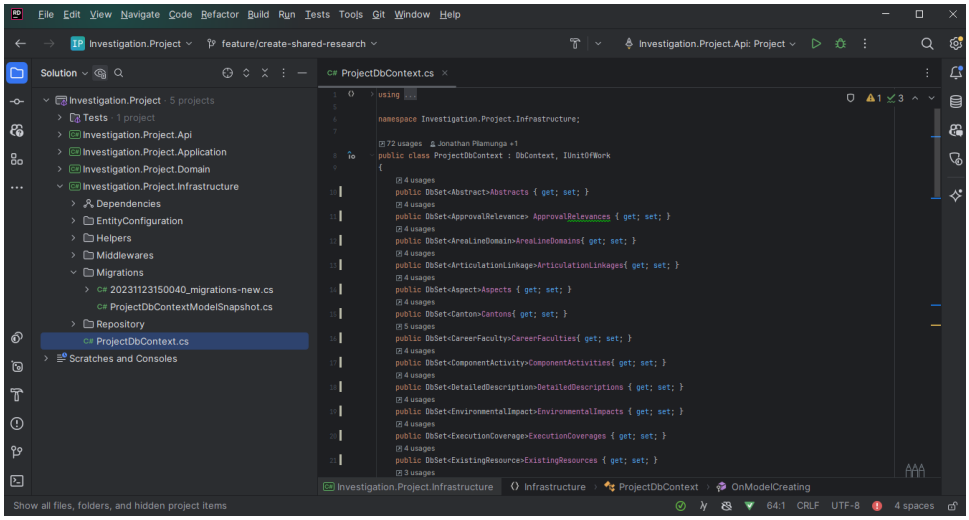


Figura 18. Clase ProjectDbContext

Adicionalmente, se definen las configuraciones específicas de cada entidad en el método OnModelCreating. En este método, se aplica una instancia de configuración para cada entidad utilizando clases que implementan IEntityTypeConfiguration<TEntity>. Estas clases definen las relaciones y restricciones de cada entidad utilizando el API de configuración de Entity Framework Core como se ilustra en la Figura 19.

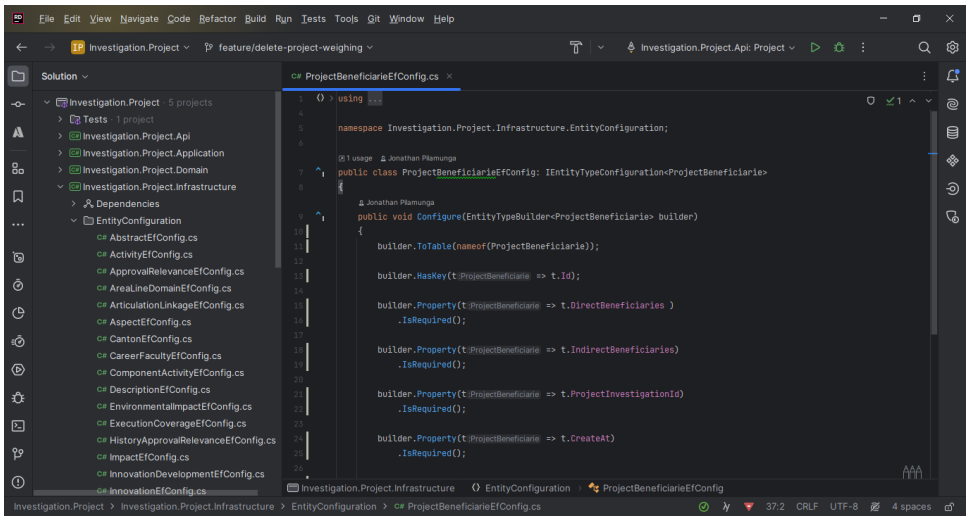


Figura 19. Configuración de cada entidad

En la sección de aplicación, se desarrollaron comandos y consultas que fueron diseñadas con el patrón de diseño CQRS (Command Query Responsibility Segregation), separando las responsabilidades de lectura y escritura. En esta sección se estableció como la aplicación interactúa con los datos y ejecuta las operaciones necesarias para cumplir con los requisitos del negocio. En la Figura 20 se puede ver un manejador de un comando.

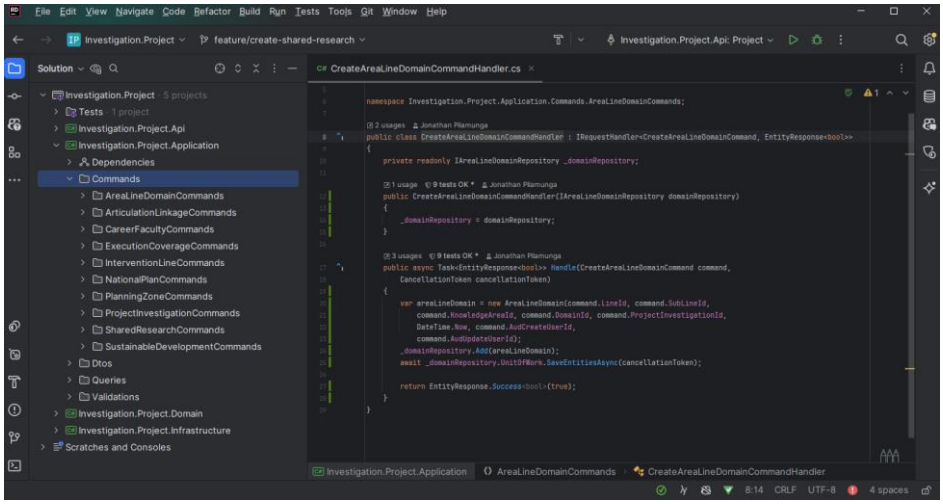


Figura 20. Sección Aplicación

En la sección API, se crearon controladores para manejar las solicitudes HTTP relacionadas con las entidades del proyecto. Cada controlador incluyó métodos correspondientes a estas operaciones, donde las solicitudes HTTP fueron convertidas en comandos o consultas de aplicación y enviadas al mediador para su procesamiento, siguiendo el patrón de diseño Mediador. Una vez completado el proceso, se devuelve el resultado junto con el código de estado HTTP correspondiente. En la Figura 21 se visualiza la configuración de un controlador.

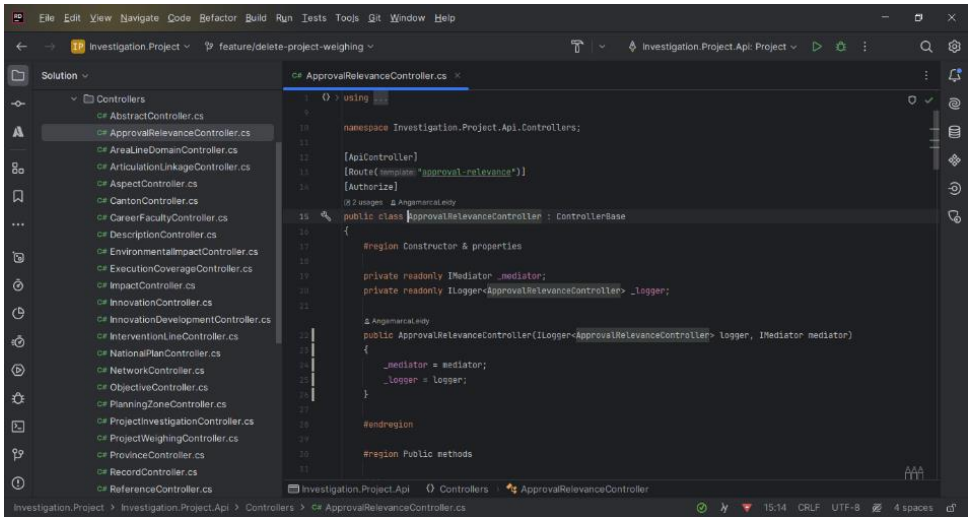


Figura 21. Sección Api

En la sección de DTOs (Data Transfer Objects), se crearon clases específicas para facilitar la transferencia de datos entre las capas de aplicación y API. Estas clases fueron diseñadas para contener la información requerida para cada operación CRUD, eliminando la transferencia de datos redundantes o innecesarios. Cada DTO fue adaptado a los requisitos de las operaciones en las que se utilizó, tal como se muestra en la Figura 22.

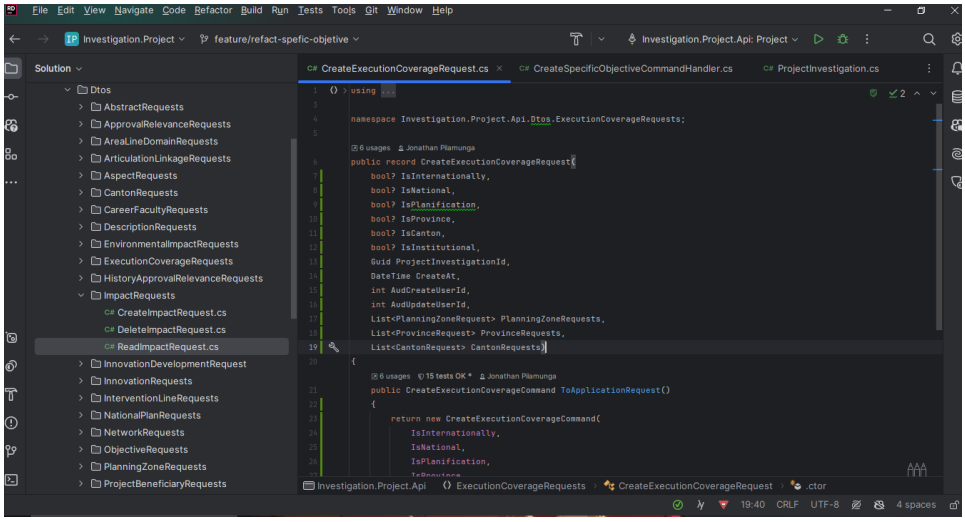


Figura 22. Dtos de las entidades

Para las pruebas unitarias, se desarrollaron escenarios de prueba para validar el comportamiento del módulo en diferentes situaciones. Estas pruebas se centraron en simular casos de uso y escenarios específicos, asegurando que el módulo reaccionará correctamente según lo esperado. El objetivo principal fue garantizar la integridad y la robustez del módulo mediante la detección temprana de posibles problemas y la validación de su comportamiento en diferentes contextos como se refleja en la Figura 23.

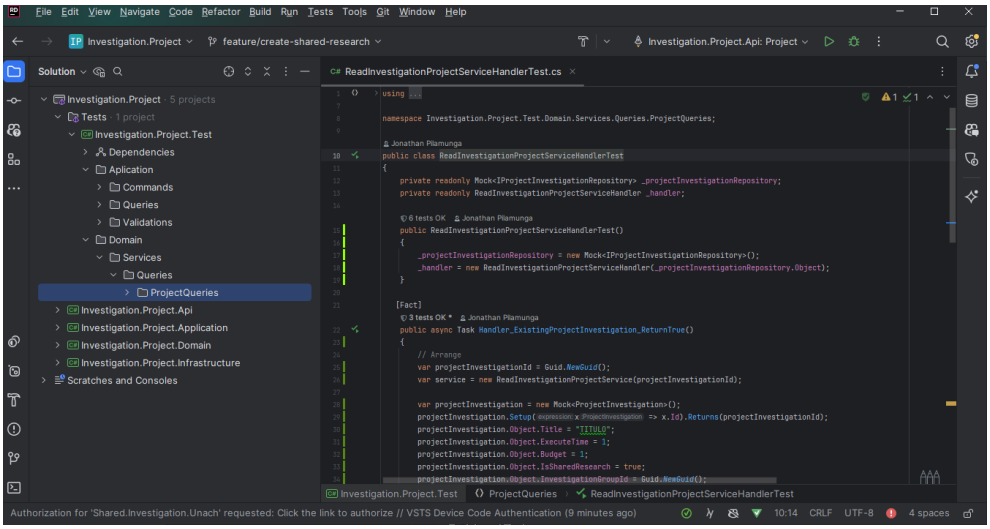


Figura 23. Sección Pruebas unitarias

Además, se realizaron validaciones de los parámetros utilizados en las pruebas para asegurar que cumplan con los requisitos establecidos. Se garantizó que los datos de entrada sean adecuados para cada escenario de prueba como la Figura 24.

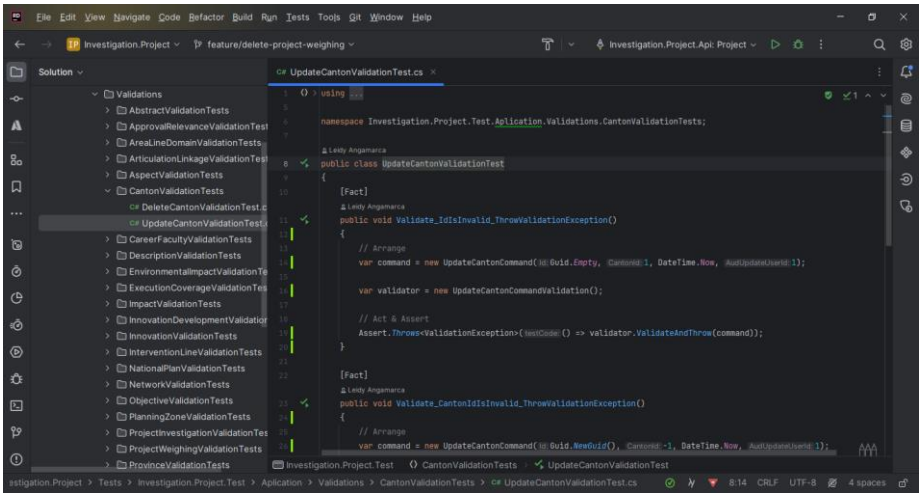


Figura 24. Validación de atributos en las pruebas

3.6.4. Revisión y retrospectiva

Durante la fase de Revisión y Retrospectiva en el marco de la metodología Scrum, se llevaron a cabo diversas actividades destinadas a asegurar la calidad y comprensión de las APIs generadas. Se realizaron pruebas para garantizar su correcto funcionamiento y su conformidad con los requisitos establecidos. Para facilitar su uso y comprensión, se utilizó Swagger como herramienta de documentación para la API REST, proporcionando una documentación clara y accesible.

En la Figura 25, indica la documentación de las APIs implementadas en JetBrains Rider a través de la interfaz de Swagger. Esta documentación brinda a los usuarios una visión detallada de los endpoints disponibles, los parámetros requeridos y las respuestas esperadas, facilita su integración y uso.

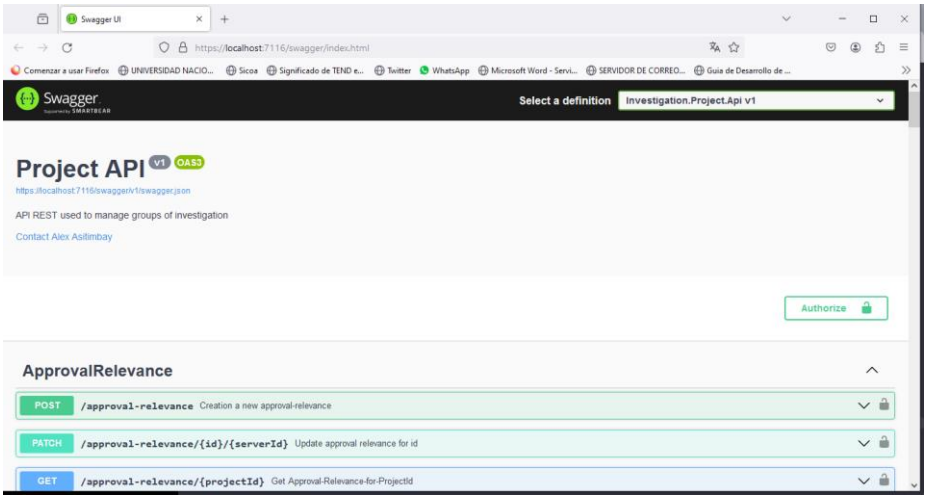


Figura 25. Documentación de Api Rest en JetBrains Rider con Swagger

3.6.5. Lanzamiento

La Figura 26, indica el lanzamiento del módulo de proyectos de investigación a través de su API correspondiente, ofreciendo una interfaz clara para la interacción con el sistema permitiendo que los usuarios puedan acceder a descripciones precisas.

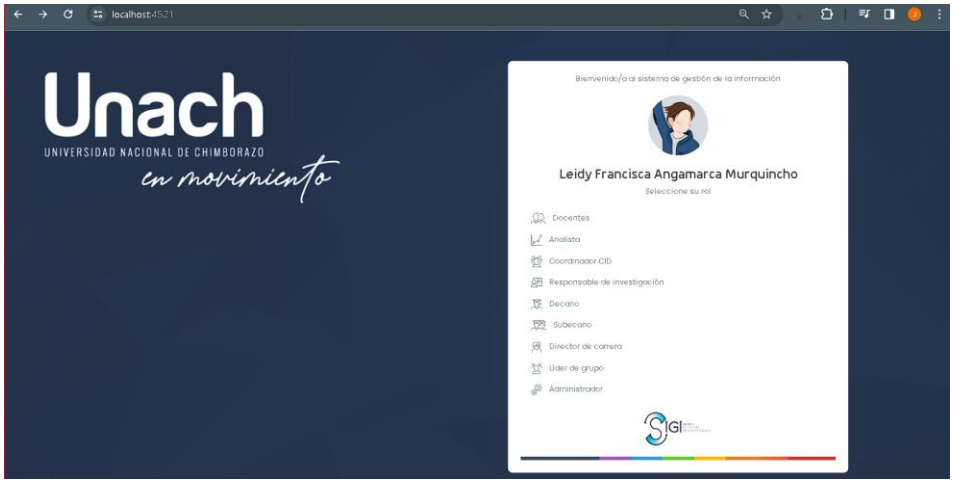


Figura 26. Microservicio publicado

3.7. Pruebas

Para dar respuesta al tercer objetivo orientado a evaluar el rendimiento del módulo de proyectos, se realizó la evaluación utilizando la herramienta Apache JMeter. Se ejecutó 400 solicitudes con distintos propósitos en un segundo. Dichas pruebas se realizaron en una computadora portátil HP Pavilon. Seguidamente, se organizó y analizó la información recabada, presentando a través de gráficos estadísticos. Este análisis fue contrastado con los datos estándares del modelo de calidad FURPS de cada indicador.

3.7.1. Parámetros de evaluación

El análisis de los indicadores es requerido para evaluar el rendimiento del módulo. En la Tabla 12 se presentan los indicadores clave que se consideró para realizar las pruebas.

Tabla 12. Parámetros de evaluación

Medida	**Indicadores**
Rendimiento	Eficiencia Tiempo de respuesta Utilización de recursos: • Uso de CPU • Uso de memoria RAM • Uso de almacenamiento

3.7.2. Ejecución de pruebas

En la Figura 27, se aprecia la configuración detallada de la petición, donde se estableció los parámetros necesarios para cada requerimiento.

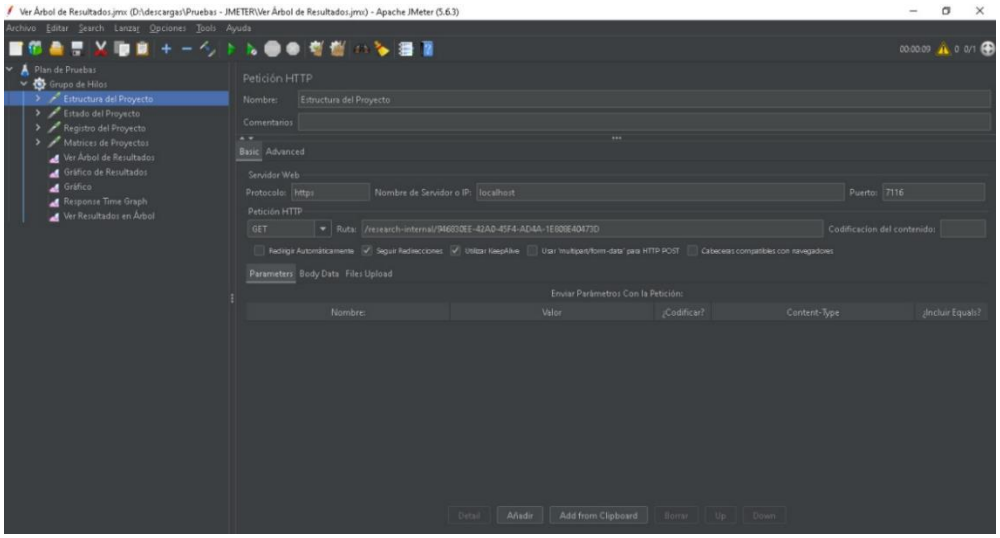


Figura 27. Configuración del escenario

En la Figura 28, se observa la configuración de los hilos, donde se especificó el tiempo de las pruebas.

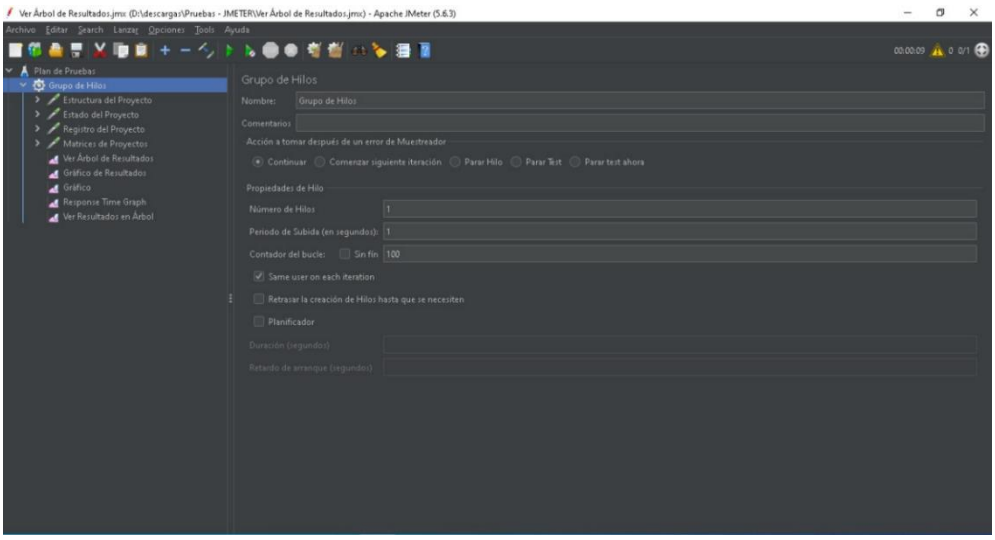


Figura 28. Configuración de hilos

La Figura 29, representa los principales resultados de las pruebas realizadas. Estos resultados proporcionan una visión detallada del rendimiento del módulo bajo diferentes condiciones.

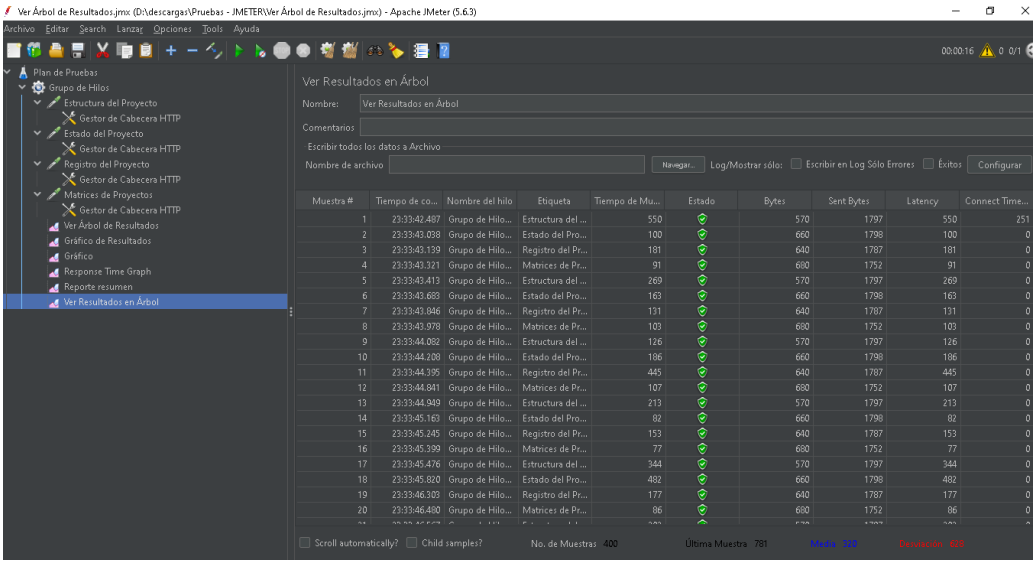


Figura 29. Resultados generales

CAPÍTULO IV. RESULTADOS Y DISCUSIÓN

4.1. Interpretación de Resultados

A continuación, se exponen los resultados derivados de las pruebas de rendimiento realizadas en el módulo de proyectos del Sistema de Gestión de la UNACH. La herramienta utilizada para llevar a cabo estas evaluaciones fue JMeter (ver Anexo 2). Inicialmente, se detallan las características y enfoques específicos de las pruebas ejecutadas.

En la Tabla 13, se detalla las pruebas realizadas en cada proceso, reflejando 400 casos examinados.

Tabla 13. Procedimientos Realizados

Procesos	Cantidad de pruebas
Estructura del Proyecto	100
Estado del Proyecto	100
Registro de perfil del Proyecto	100
Matrices del Proyecto	100

En la Figura 30, se visualiza los resultados de las pruebas del módulo mediante un gráfico de pastel, destacando que el 100% de las pruebas realizadas resultaron exitosas.

Figura 30. Porcentaje de éxito del módulo

4.1.1. Valoración de indicadores

Para esta sección, se presenta los gráficos estadísticos correspondientes a cada indicador evaluado, constituyendo así un apartado específico dedicado a la valoración de estos parámetros. Cada gráfico proporciona una representación visual detallada del rendimiento del módulo en relación con los indicadores establecidos. Este enfoque permitió una

comprensión más profunda y segmentada de la evaluación, brindando información clave sobre el desempeño del sistema en cada aspecto considerado.

Eficacia

La Tabla 14 y la Figura 31, demuestra el porcentaje de eficacia del módulo en relación con las 400 pruebas realizadas. Este indicador revela que la totalidad de las solicitudes, representadas en los 400 casos examinados, obtuvieron un rendimiento del 100%.

Tabla 14. Resultados del indicador Eficacia

Parámetro	Indicador	Petición
Eficacia	Número de peticiones realizadas correctamente	400
	Promedio (%)	100%

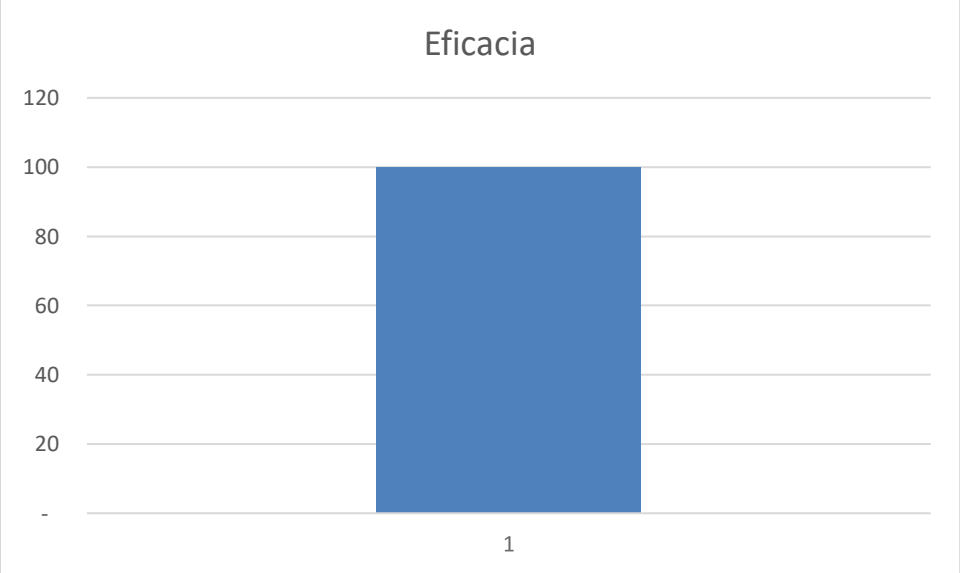


Figura 31. Indicador Eficacia

Tiempo de respuesta

En la Figura 32, se evidencia de manera gráfica los resultados generados por la herramienta JMeter, proporcionando una visualización detallada de los datos recabados para calcular este indicador.

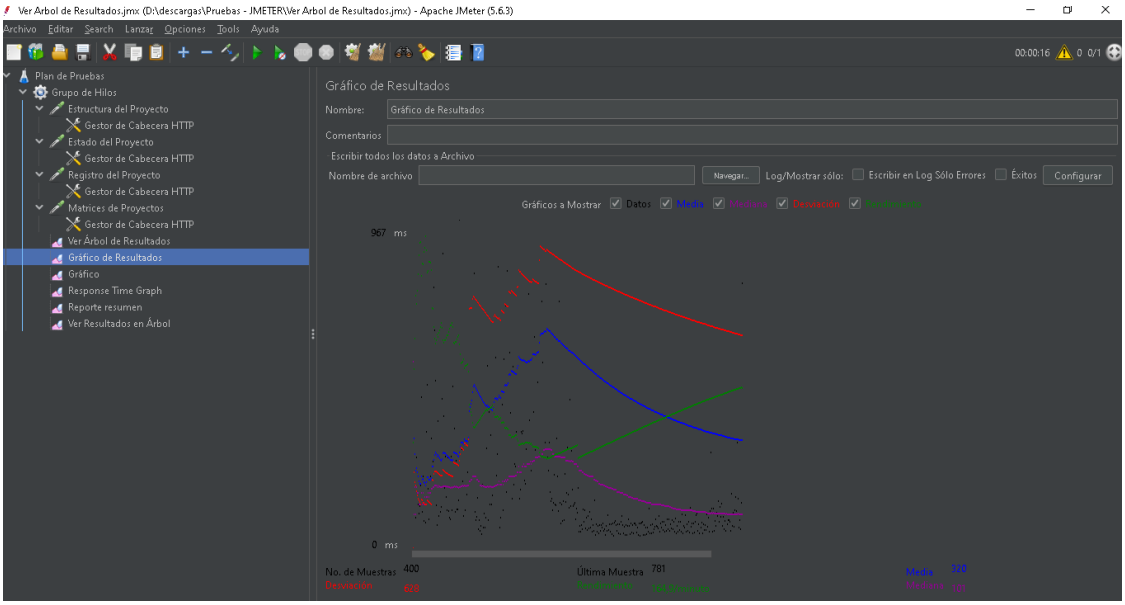


Figura 32. Tiempo de respuesta

El módulo de proyectos de investigación presenta un tiempo de respuesta promedio de 0,32 segundos según JMETER.

Utilización de recursos

La evaluación de la utilización de recursos resulta fundamental para comprender como un programa o sistema impacta en el rendimiento global, asegurando una operación eficiente y equilibrada del sistema (véase en el Anexo 3).

La Tabla 15 son los resultados detallados del indicador de utilización de recursos, desglosando los porcentajes correspondientes a cada parámetro.

Tabla 15. Resultados de utilización de recursos

Parámetro	Indicador	Porcentaje
Consumo de Recursos	Uso de CPU	11%
	Uso de Memoria RAM %	51%
	Uso de Disco Duro %	1%

CPU

Según los datos presentados en la Figura 33, la aplicación utilizó en promedio el 11% de la CPU durante las pruebas.

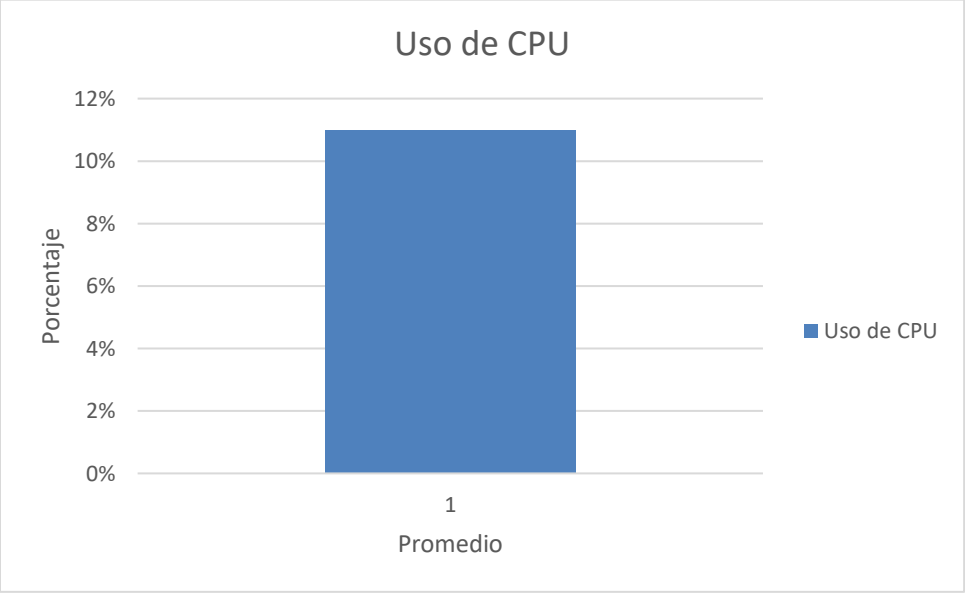


Figura 33. Uso de CPU

Uso de Memoria RAM

En la Figura 34, el módulo alcanzó un promedio de 58% de la Memoria RAM durante las pruebas.

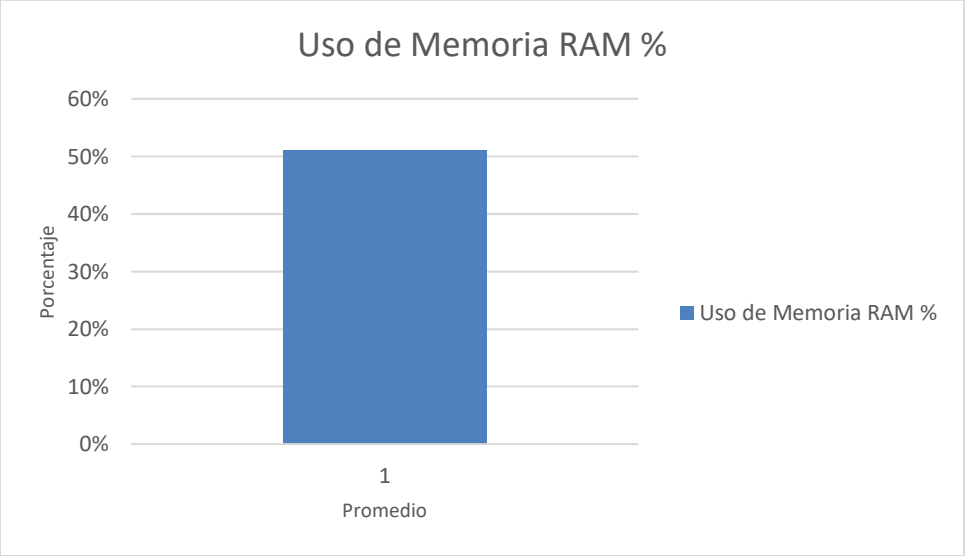


Figura 34. Uso de Memoria RAM

Disco Duro

En la Figura 35, el módulo alcanzo un promedio de 1% del uso del disco duro durante las pruebas.

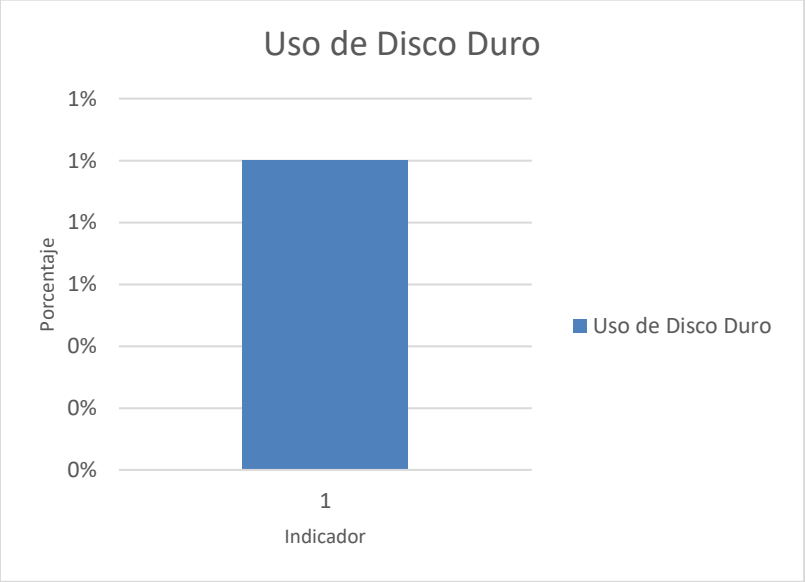


Figura 35. Uso de Disco Duro

4.1.2. Comparación entre los valores obtenidos y valores establecidos en el modelo de calidad FURPS

En la Tabla 16, se muestra como el módulo de proyectos de investigación cumple con los estándares del modelo de calidad FURPS, teniendo un rendimiento eficaz.

Tabla 16. Comparación de los valores obtenidos y los de FURPS

Parámetro	Modelo FURPS	Valores obtenidos
Eficacia	95%	100%
Tiempo de respuesta	5s	0,32s
Utilización de Recursos	25%	21%

4.2. Discusión

En conjunto, estos resultados indican que la implementación de microservicios en el módulo de proyectos del sistema de Gestión de Investigación de la UNACH contribuyó positivamente al rendimiento del sistema, ofreciendo una eficacia del 100%, tiempos de respuesta bajos y un uso de recursos eficientes en comparación de las métricas del modelo de calidad FURPS. Estos aspectos resaltan la calidad en rendimiento del sistema, demostrando su capacidad para satisfacer los requisitos operativos y las expectativas de los usuarios.

La implementación de microservicios en el Módulo de Proyectos muestra un desempeño ligeramente superior en comparación con la investigación realizada previamente por (Alvarado, 2023). Si bien el estudio previo logró resultados satisfactorios, el módulo implementado alcanzó métricas aún más favorables en cada uno de los aspectos evaluados. Mientras que el trabajo anterior obtuvo una muy alta eficacia, tiempos de respuesta

razonables y un buen aprovechamiento de recursos, el módulo de proyectos superó esos niveles al alcanzar una eficacia máxima, tiempos de respuesta significativamente más rápidos y una utilización de recursos más eficiente.

CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

La investigación y selección de herramientas adecuadas permitieron el desarrollo de una arquitectura de microservicios para el módulo de proyectos del sistema de Gestión de Investigación de la UNACH. Se identificaron tecnologías como .NET7, C#, JetBrains Rider, SQL server y Azure DevOps que proporcionan un conjunto completo de funcionalidades para la implementación eficiente de microservicios.

El desarrollo del módulo de proyectos de investigación permitió la implementación de funcionalidades clave en el sistema, tales como: el seguimiento de proyectos, estructuras, perfiles y matrices. Se optó por seguir la metodología SCRUM, aprovechando sus sprints para realizar entregas constantes. Para el almacenamiento de datos, se utilizó SQL Server, mientras que JetBrains Rider fue el IDE junto con .NET7 como framework, que simplificó el acceso a todas las librerías necesarias. Además, la cultura de Azure DevOps garantizó un mejor control en todo el proceso.

En la evaluación del rendimiento del módulo de proyectos, se empleó la herramienta JMETER, con el objetivo de medir los indicadores de calidad según el modelo FURPS. Los resultados obtenidos demostraron una eficacia del 100%, con un tiempo de respuesta de 0,32 segundos y una utilización de recursos del 21%. Estos antecedentes indican que se logró cumplir con los parámetros de calidad establecidos, validando el rendimiento del módulo de proyectos para el sistema de Gestión de Investigación de la UNACH.

5.2. Recomendaciones

Investigar herramientas tecnológicas actuales ha demostrado ser altamente beneficioso para el desarrollo de diversas arquitecturas, entre ellas la de microservicios. JetBrains y Azure Devops suelen incorporar características avanzadas que pueden mejorar significativamente al desarrollo de software.

Adoptar metodologías ágiles en la gestión del desarrollo de software para garantizar la agilidad en la gestión de requerimientos y la flexibilidad en la adaptación de procesos. Esto permite una entrega más rápida y continua del software en desarrollo, promoviendo una colaboración efectiva entre equipos y clientes, una respuesta ágil a los cambios durante el ciclo de desarrollo. La aplicación de metodologías como SCRUM optimiza la eficiencia del equipo de desarrollo y asegura una mayor satisfacción del cliente al recibir productos de software iterativos y ajustados a sus necesidades en el menor tiempo posible.

Emplear herramientas de evaluación de rendimiento, como JMETER, para medir indicadores de calidad garantiza la entrega de un software más sólido y eficiente.

BIBLIOGRÁFIA

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

AWS. (2023). *¿Qué son los microservicios? | AWS*. https://aws.amazon.com/es/microservices/

Azure. (2020). *Diseño de arquitectura de microservicios - Azure Architecture Center.* https://learn.microsoft.com/es-es/azure/architecture/microservices/

Blancarte, O. (2021). *Datalake una nueva forma de análisis de datos.* https://www.oscarblancarteblog.com/author/oblancarte/

Callejas-Cuervo, M., Alarcón-Aldana, A. C., y Álvarez-Carreño, A. M. (2017). Modelos de calidad del software, un estado del arte. *Scielo, 13*(1). https://doi.org/https://doi.org/10.18041/entramado.2017v13n1.25125

Caqui, D. N. (2022). *Arquitectura de microservicios para mejorar la calidad de software en una entidad bancaria de Lima metropolitana, 2022.* http://repositorio.unasam.edu.pe/handle/UNASAM/5337

Chakray. (2019). *¿Qué son los microservicios?: Definición, características y ventajas y desventajas.* https://www.chakray.com/es/que-son-los-microservicios-definicion-caracteristicas-y-ventajas-y-desventajas/

Cusco, B. F. (2022). *Desarrollo e implementación de una arquitectura DevOps para un sistema web basado en microservicios en infraestructuras basadas en código.* http://dspace.ups.edu.ec/handle/123456789/21675

Dongee. (2023). *¿Qué es C#? Guía completa para principiantes*. https://www.dongee.com/tutoriales/que-es-c/

Douglas, J., Likness, J., & Petropoulos, A. (2022). *.NET 7 is Available Today*. https://devblogs.microsoft.com/dotnet/announcing-dotnet-7/

Fernández, E. (2021). *Que son los Microservicios. Características y Ventajas.* https://www.incentro.com/es-ES/blog/que-son-microservicios

Francia, J. (2017). *¿Qué es Scrum?* https://www.scrum.org/resources/blog/que-es-scrum

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

Jetbrains. (2023). *Rider: El IDE .NET multiplataforma de JetBrains*. https://www.jetbrains.com/es-es/rider/

Larrea, N. P., Valencia, M. V., Cazco, S. A., & Hermida, B. A. (2022). Anólisis de los lenguajes de programación mós utilizados en el desarrollo de aplicaciones web y móviles. *8*(3), 1601–1625. https://doi.org/https://www.dominiodelasciencias.com/ojs/index.php/es/article/view/2889

Laura, C. A. (2023). Pruebas de Software para Microservicios. *4*(1), 151-160. https://doi.org/https://revistas.ulasalle.edu.pe/innosoft/article/view/86

León, J. R. (2021). *Arquitectura basada en microservicios para un sistema de producción de hidrocarburos en la empresa BLC Venezuela, C.A.* Universidad Católica Andrés Bello: http://catalogo-gy.ucab.edu.ve/documentos/tesis/36236.pdf

Lotriet, H., y Mudadi, A. (2023). AN ANALYSIS OF DEVOPS' IMPACT ON INFORMATION TECHNOLOGY ORGANISATIONS: A CASE STUDY. *34*(1), 155-167. https://doi.org/https://www.proquest.com/docview/2821425108/F8142C0FFADB4EA2PQ/2?accountid=36757&sourcetype=Scholarly%20Journals

Manobanda, A. B. (2020). *MODELO FURPS PARA EVALUAR EL SISTEMA DE RECAUDACIÓN DE PATENTES GADM PENIPE.* https://doi.org/http://dspace.unach.edu.ec/handle/51000/7080

Martín, J., y García, S. (2020). *Arquitectura de microservicios aplicada al desarrollo de un Ecosistema Educativo basado en aplicaciones software de respuesta automática inteligente.* https://oa.upm.es/64511/

Microsoft. (2023). *Lenguaje C#.* https://learn.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/

Microsoft. (2023). *Una introducción a NuGet*. https://learn.microsoft.com/es-es/nuget/what-is-nuget

Microsoft Build. (2023). *Bibliotecas de clases de .NET - .NET.* https://learn.microsoft.com/es-es/dotnet/standard/class-libraries

Ortega, J. M. (2020). Tecnologías para arquitecturas basadas en microservicios: Patrones y soluciones para aplicaciones desplegadas en contenedores. https://doi.org/https://www.researchgate.net/publication/342702922_Tecnologias_para_arquitecturas_basadas_en_microservicios_Patrones_y_soluciones_para_aplicaciones_desplegadas_en_contenedores

Palacio, M. (2022). *Scrum Master.* Scrum Manager: https://www.scrummanager.com/files/scrum_master.pdf

Richardson, C. (2023). *Pattern: Microservice Architecture.* https://doi.org/http://microservices.io/patterns/microservices.html

Sheti, S. (2022). *Apache Jmeter: Todo lo que necesitas saber.* Geekflare: https://geekflare.com/es/apache-jmeter-guide/

State of Agile. (2022). *State of Agile Report.* https://stateofagile.com/

Tamayo, A., y Katrib, M. (Enero de 2011). *Los patrones Inversión de Control (IoC) e Inyección de Dependencias (DI).* https://www.researchgate.net/publication/240609917_Los_patrones_Inversion_de_Control_IoC_e_Inyeccion_de_Dependencias_DI

UNACH. (2023). *DIRECCIÓN DE INVESTIGACIÓN*. https://investigacion.unach.edu.ec/bienvenida.php

Vera-Rivera, F. H., Gaona, C., & Astudillo, H. (2019). Desarrollo de aplicaciones basadas en microservicios: tendencias y desafíos de investigación. *Revista Ibérica De Sistemas e Tecnologias De Informação*, 107-120. https://doi.org/https://www.proquest.com/scholarly-journals/desarrollo-de-aplicaciones-basadas-en/docview/2348878316/se-2

ANEXOS

Anexo 1: Diccionario de datos

Tabla 17. Tabla ProjectInvestigation

	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	Status	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

Tabla 18. Tabla Abstract

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla.
	Summary	nvarchar (MAX)	Resumen del proyecto.
FK	ProjectInvestigactionId	uniqueidentifier	Llave foránea de la tabla principal.
	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.

Tabla 19. Tabla Activity

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla.
	Description	nvarchar (MAX)	Describir la actividad.
	Weighing	int	Identificador de la ponderación
	StartDate	datatime2(7)	Fecha de inicio del proyecto.
	EndDate	datatime2(7)	Fecha de finalización del proyecto.
	ActualStartDate	datatime2(7)	Fecha de actual de inicio del proyecto.
	ActualEndDate	datatime2(7)	Fecha de actual de finalización del proyecto.
FK	EspecificObjectiveId	uniqueidentifier	Llave foránea de la tabla SpecificObjective. 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.

Tabla 20. Tabla ApprovalRelevance

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla.
	Status	nvarchar (30)	Indicador si hay aprobaciones relevantes.
	ServerId	int	Identificación del servidor.
	Role	nvarchar (30)	Rol.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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.

Tabla 21. Tabla AreaLineDomain

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla.
	LineId	int	Identificar la línea de intervención.
	SubLineId	int	Identificar la sub línea de intervención.
	KnowledgeAreaId	int	Especificar el área de conocimiento.

	DomainId	int	Identificar el dominio del proyecto.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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.

Tabla 22. Tabla ArticulationLinkage

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla.
	DerivativeProject	nvarchar (MAX)	Nombre de proyecto de vinculación del cual se deriva.
	Components	bit	Escoger los nombres de componentes.
	Status	bit	Verificar si tiene el proyecto articulación con un proyecto de vinculación.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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.

Tabla 23. Tabla Aspect

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla.
	Treatment	nvarchar (MAX)	Describir el tratamiento que se dará a los aspectos éticos o sociales durante el desarrollo de la investigación.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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.

Tabla 24. Tabla Canton

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla.
	CantonId	int	Permite identificar a los cantones en las que se ejecutará el proyecto.
FK	ExecutionCoverageId	uniqueidentifier	Llave foránea de la tabla ExecutionCoverage.
	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.

Tabla 25. Tabla CareerFaculty

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla.
	CareerId	int	Carreras involucradas.
	FacultyId	int	Facultades involucradas.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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.

Tabla 26. Tabla Description

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	StateOfTheArt	Nvarchar (MAX)	Estado del arte del proyecto, máximo 2 páginas.
	Diagnosis	Nvarchar (MAX)	Abordar de forma precisa el diagnóstico del proyecto.
	Problem	nvarchar (MAX)	Definir de forma clara y concisa el problema o necesidad que abordará el proyecto de investigación.
	Justification	nvarchar (MAX)	Justificar cómo el desarrollo de los objetivos del proyecto contribuirá a

			solucionar el problema de investigación planteado.
	Methodology	nvarchar (MAX)	Exponer el proceso metodológico a seguir para desarrollar la investigación (la metodología, tipo de investigación; técnicas e instrumentos, etc.)
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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.

Tabla 27. Tabla EnvironmentalImpact

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla.
	ImpactDescription	nvarchar (MAX)	Describir los impactos ambientales positivos y negativos.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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.

Tabla 28. Tabla ExecutionCoverage

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla.
	IsInternationally	bit	Identificar si es proyecto va a tener cobertura internacional.
	IsNational	bit	Identificar si es proyecto va a tener cobertura nacional.
	IsPlanification	bit	Identificar si es proyecto va a tener cobertura zonal.
	IsProvince	bit	Identificar si es proyecto va a tener cobertura provincial.
null	IsCanton	bit	Identificar si es proyecto va a tener cobertura cantonal.

	IsInstitutional	bit	Identificar si es proyecto va a tener cobertura institucional.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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.

Tabla 29. Tabla HistoryApprovalRelevance

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	Observation	nvarchar (MAX)	Observaciones hechas al proyecto.
FK	ApprovalRelevanceId	uniqueidentifier	Llave foránea de la tabla ApprovalRelevance.
	Status	nvarchar (MAX)	Indicador si existe el historial
	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

Tabla 30. Tabla Impact

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	ImpactIdItemCatalog	uniqueidentifier	Identificador del impacto existente en el catálogo.
	Other	nvarchar (MAX)	Describir otros impactos que no existen en el catálogo.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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

Tabla 31. Tabla Innovation

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	InnovativeResults	nvarchar (MAX)	Describir los resultados del proyecto que contengan innovación.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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

Tabla 32. Tabla InnovationDevelopment

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla.
	InnovationProject	nvarchar (MAX)	Derivar en un Proyecto de innovación.
	InnovatorItem	nvarchar (MAX)	Definir el Elemento Innovador.
	InnovatorBusiness	nvarchar (MAX)	Describir el Negocio Innovador.
	Status	bit	Estado para saber si el proyecto tiene previsto el desarrollo de innovación.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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.

Tabla 33. Tabla InterventionLines

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla.
	IdIntervention	uniqueidentifier	Identificador de un servicio externo
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	InterventionLinesid	int	Identificación de una línea de intervención externo.
	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.

Tabla 34. Tabla LogicFrames

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla.
	Hierarchy	uniqueidentifier	Identificador de la jerarquía de objetivos en la matriz.
	Indicator	nvarchar (MAX)	Campo denominado Indicador de la matriz.
	Verification	nvarchar (MAX)	Campo de Fuentes de Verificación de la matriz.
	Supposed	nvarchar (MAX)	Campo de Supuesto de la matriz del marco lógico.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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.

Tabla 35. Tabla NationalPlan

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	ObjectGoodLivingId	int	Objetivos del plan nacional de desarrollo al que aporta el proyecto.
	Status	bit	Ayudar a determinar si existe los objetivos del plan nacional.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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

Tabla 36. Tabla Network

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla

	NetworkProject	nvarchar (MAX)	Nombre del proyecto de la red.
FK	SharedResearchId	uniqueidentifier	Llave foránea de la tabla SharedResearch.
	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

Tabla 37. Tabla Objective

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	Description	Nvarchar (MAX)	Escribir el objetivo general
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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

Tabla 38. Tabla PlanningZone

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	PlanningZoneId	int	Permite identificar a las zonas en las que se ejecutará el proyecto.
FK	ExecutionCoverageId	uniqueidentifier	Llave foránea de la tabla ExecutionCoverage.
	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

Tabla 39. Tabla ProjectBeneficiary

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla

	DirectBeneficiaries	nvarchar (MAX)	Descripción de los beneficiarios directos del proyecto.
	IndirectBeneficiaries	nvarchar (MAX)	Descripción de los beneficiarios indirectos del proyecto.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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

Tabla 40. Tabla Province

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	ProvinceId	int	Permite identificar a las provincias en las que se ejecutará el proyecto.
FK	ExecutionCoverageId	uniqueidentifier	Llave foránea de la tabla ExecutionCoverage.
	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

Tabla 41. Tabla Record

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	StatusItemId	uniqueidentifier	Identificación del estado del Ítem.
	Status	bit	Indicador si aplica record
null	Observation	nvarchar (MAX)	Observaciones
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
null	AcceptanceResolution	nvarchar (MAX)	Resolución de aceptación
null	ClosingResolution	nvarchar (MAX)	Resolución de cierre
null	ResolutionDecommissioned	nvarchar (MAX)	Resolución dado de baja
null	TimeExtension	nvarchar (MAX)	Resolución de extensión de tiempo
	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

Tabla 42. Tabla Reference

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	List	nvarchar (MAX)	Realizar un listado de los documentos (libros, artículos de revistas, memorias de congresos, etc.) que fueron utilizados como referencia para el desarrollo de la propuesta del proyecto
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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

Tabla 43. Tabla ResearchAssistant

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	AssitanId	uniqueidentifier	Identificación del Asistente
	StudentId	int	Identificación de Estudiante
	CatalogFunctionId	int	Identificación de Función en el Catálogo.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	Status	bit	Indicador si existe un asistente
null	EndDate	datatime2(7)	Fecha de finalización del proyecto
null	StartDate	datatime2(7)	Fecha de inicio del Proyecto
	SeedbedId	uniqueidentifier	Identificador de un estudiante que pertenece a un semillero.
	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

Tabla 44. Tabla ResearchExternal

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	ExternalId	uniqueidentifier	Identificación del investigador externo.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	Status	bit	Sirve para verificar si es una investigación externa.
null	EndDate	datatime2(7)	Fecha de finalización del proyecto
null	StartDate	datatime2(7)	Fecha de inicio del proyecto
	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

Tabla 45. Tabla ResearchInternal

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	ServerId	int	Identificación del servidor
	CatalogFunctionId	int	Identificación de la Función del Catálogo.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	Status	bit	Sirve para verificar si es una investigación interna.
	Resolve	nvarchar (50)	Describir si está pendiente en resolver.
null	EndDate	datatime2(7)	Fecha de finalización del proyecto
null	StartDate	datatime2(7)	Fecha de inicio del proyecto
	CreateAt	datatime2(7)	Fecha de creación
	GroupId	uniqueidentifier	Identificador del grupo al que pertenece el investigador interno.
	UpdateAt	datatime2(7)	Fecha de actualización
	AudCreateUserId	int	Identificación de Usuario de Creación de Auditoría
	AudUpdateUserId	int	Identificación de Usuario de Actualización de Auditoría

Tabla 46. Tabla Resource

Nombre	Tipo de dato	Descripción

PK	Id	uniqueidentifier	Identificador de la tabla
	Dependence	nvarchar (MAX)	Dependencia que facilita el requerimiento.
	Servicio	nvarchar (MAX)	Bien/Servicio
	Amount	nvarchar (MAX)	Cantidad
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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

Tabla 47. Tabla Result

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	IdResult	uniqueidentifier	Identificador del resultado existente en el catálogo.
	Other	nvarchar (MAX)	Describir otros resultados que no existen en el catálogo.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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

Tabla 48. Tabla ResultDescription

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	Description	nvarchar (MAX)	Describir el resultado
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla Principal.
	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

Tabla 49. Tabla SharedResearch

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	InstitutionName	nvarchar (100)	Nombre de Institución co-ejecutora.
	Representative	nvarchar (100)	Nombres y apellidos
	IndentityCard	int	Guarda la cédula de identidad
	Phone	int	Teléfono del representante
	Email	nvarchar (100)	Correo electrónico del representante.
	Address	nvarchar (200)	Dirección
	WebPage	nvarchar (100)	Página web institucional
	ExecutiveBody	nvarchar (100)	Órgano Ejecutor que puede ser un departamento o unidad de investigación.
	InstrumentType	nvarchar (MAX)	Definir el tipo de instrumento de cooperación
	IsNetworkProject	bit	Identificar si es un proyecto de una red.
	Status	bit	Indicador si aplica recursos compartidos
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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

Tabla 50. Tabla SpecificObjective

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	Description	nvarchar (MAX)	Mencionar los objetivos específicos
FK	ObjectiveId	uniqueidentifier	Llave foránea de la tabla Objective.
	Weighing	int	Identificador de la ponderación
	StartDate	datatime2(7)	Fecha de inicio del proyecto.
	EndDate	datatime2(7)	Fecha de finalización del proyecto.
	ActualStartDate	datatime2(7)	Fecha de actual de inicio del proyecto.
	ActualEndDate	datatime2(7)	Fecha de actual de finalización del proyecto.
	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

Tabla 51. Tabla Sustainability

	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	Description	nvarchar (MAX)	Describir los mecanismos que se proponen para dar sostenibilidad al proyecto, una vez que haya concluido el plazo de ejecución del proyecto.
	Questions	nvarchar (MAX)	Respuesta a las preguntas como: ¿la institución beneficiaria tiene interés y la capacidad de brindar el apoyo para la continuación del proyecto?, ¿existe la posibilidad de involucrar a otras instituciones de forma que se pueda dar continuidad al proyecto?, ¿se han identificado otras fuentes de cofinanciamiento podrían estar interesadas en apoyar la continuación del proyecto?
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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.

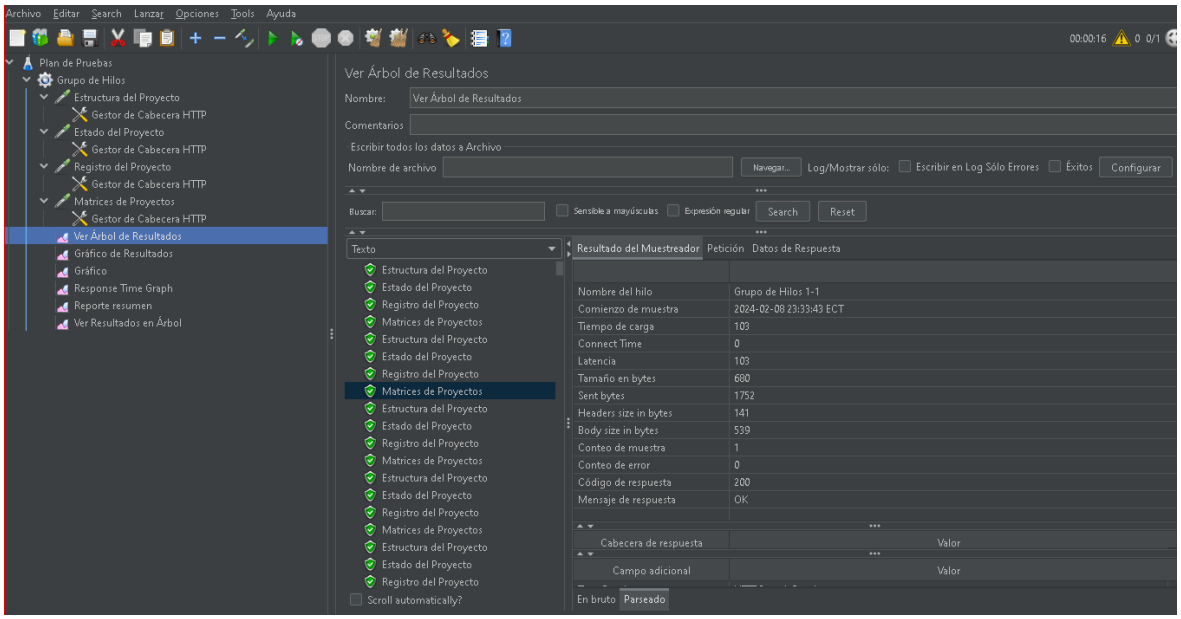
Tabla 52. SustainableDevelopment

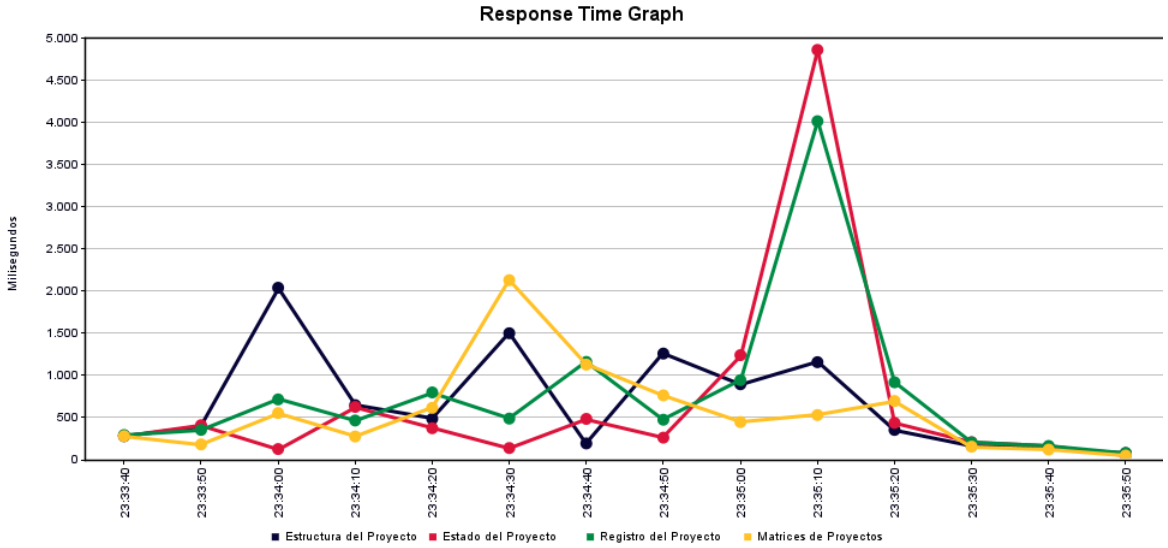
	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	IdSustainable	uniqueidentifier	Escoger objetivos de desarrollo sostenible.
	SustainableCatalog	uniqueidentifier	Catálogo de objetivos sostenibles.
	Status	bit	Determinador si existe objetivos del desarrollo sostenible.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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

Tabla 53. Tabla TransferResult

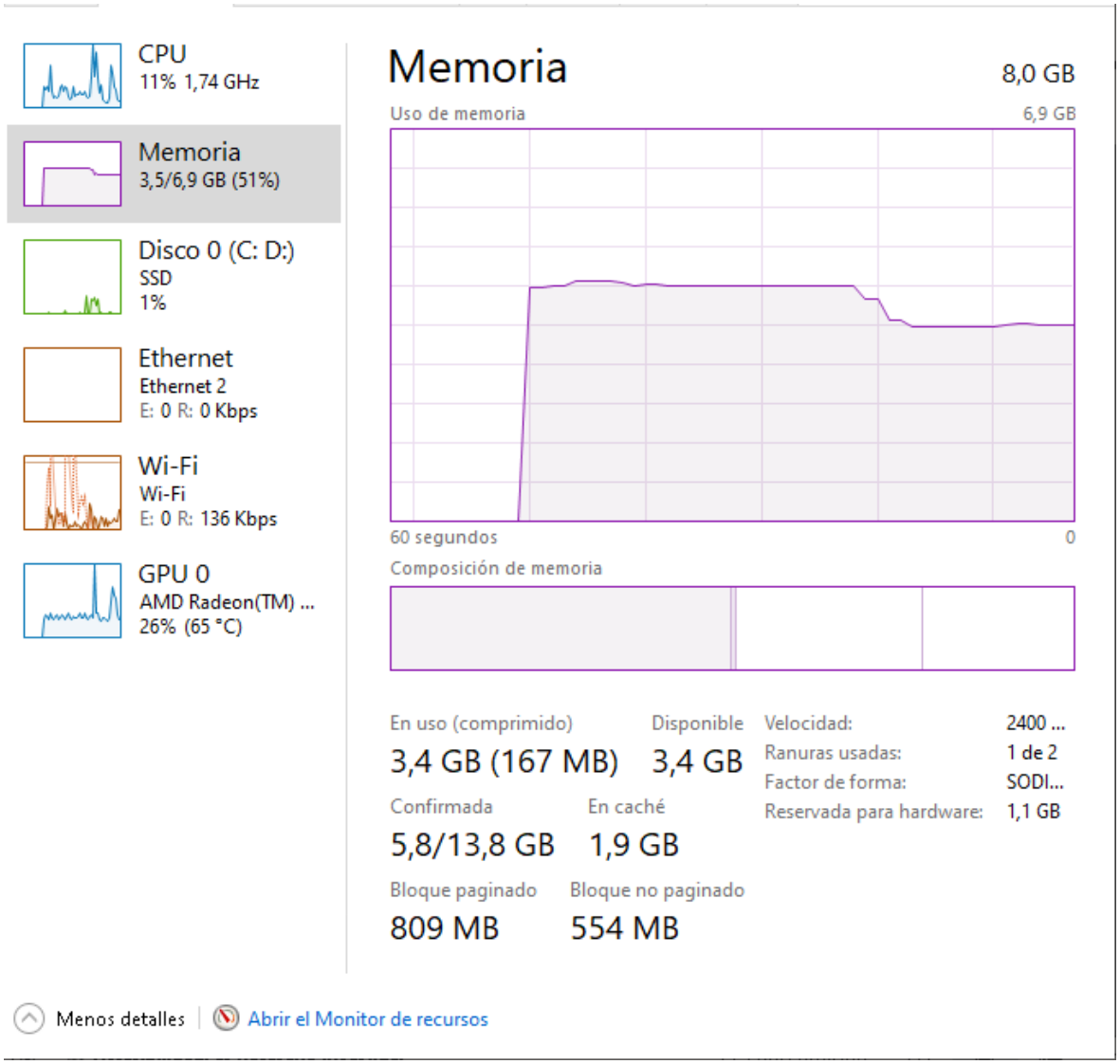
	Nombre	Tipo de dato	Descripción
PK	Id	uniqueidentifier	Identificador de la tabla
	Media	nvarchar (MAX)	Exponer cuáles serán los medios para realizar la transferencia de los resultados del proyecto que pueden ser: publicaciones científicas, publicaciones técnicas, organización de talleres con participación de los beneficiarios del proyecto, participación de los investigadores en congresos nacionales e internacionales, etc.
	TechnicalDevelopment	nvarchar (MAX)	Detallar si el proyecto incluye algún tipo de desarrollo tecnológico, ya sea un producto o un proceso.
FK	ProjectInvestigationId	uniqueidentifier	Llave foránea de la tabla principal.
	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

Anexo 2: Pruebas en la herramienta JMeter





Anexo 3: Utilización de recursos



Anexo 4: Acta de entrega y recepción

Dirección de Investigación
VICERRECTORADO DE INVESTIGACIÓN, VINCULACIÓN Y POSTGRADO

en movimiento
SGC

CERTIFICACIÓN

A petición de las partes interesadas: yo Ing. Alex Fabricio Asitimbay Chamba. Técnico de Investigación de la Universidad Nacional de Chimborazo. Certifico:

Que, la **Srta. LEIDY FRANCISCA ANGAMARCA MURQUINCHO**, portador del número de cédula de ciudadanía No. 1150016499 y el **Sr. JONATHAN DAVID PILAMUNGA VALLA**, portador del número de cédula de ciudadanía No. 0606218816, estudiantes de la carrea de Ingeniería en Tecnologías de la Información, **entregaron el Módulo de proyectos del Sistema de Gestión de Investigación de la UNACH**, como parte de su Tesis de Grado denominado: "**MICROSERVICIOS PARA EL MÓDULO DE PROYECTOS DEL SISTEMA DE GESTIÓN DE INVESTIGACIÓN DE LA UNACH**".

Es todo cuanto puedo certificar en honor a la verdad. Facultando a los interesados hacer uso del presente, para los fines que crea conveniente.

Riobamba. 20 de marzo de 2024

Firmado electrónicamente por:
ALEX FABRICIO ASITIMBAY CHAMBA

Ing. Alex Fabricio Asitimbay Chamba

Técnico de Investigación de la Universidad Nacional de Chimborazo

Anexo 5: Manual De Usuario

MANUAL DE USUARIO DEL MICROSERVICIO DE PROYECTOS DE INVESTIGACIÓN

Versión

1.0.0

Fecha de creación

18/03/2024

Realizado por

Leidy Angamarca y Jonathan Pilamunga

Índice

1. Introducción 3
2. Consideración previas 3
3. Modulo del Proyectos de Investigación 3
3.1 Autorización 3
3.2 Descripción del API 4
3.3 Respuesta del API 5
4. EndPoints Generados 5

1. Introducción

Este documento orienta a los desarrolladores en el uso de la API del módulo de proyectos de investigación. Ofrece detalles sobre los endpoints y métodos disponibles, simplificando la interacción con la API desde la autenticación hasta su utilización.

2. Consideración previas

El módulo de proyectos de investigación fue diseñado específicamente para satisfacer las necesidades de la Dirección de Investigación de la Universidad Nacional de Chimborazo (UNACH). En consecuencia, el acceso y uso de la API están restringidos únicamente al personal autorizado.

3. Módulo del Proyectos de Investigación

Puede encontrar el módulo en la siguiente ruta:

- Api Proyectos https://pruebas.unach.edu.ec:4462/swagger/index.html
- Producción http://pruebas.unach.edu.ec/

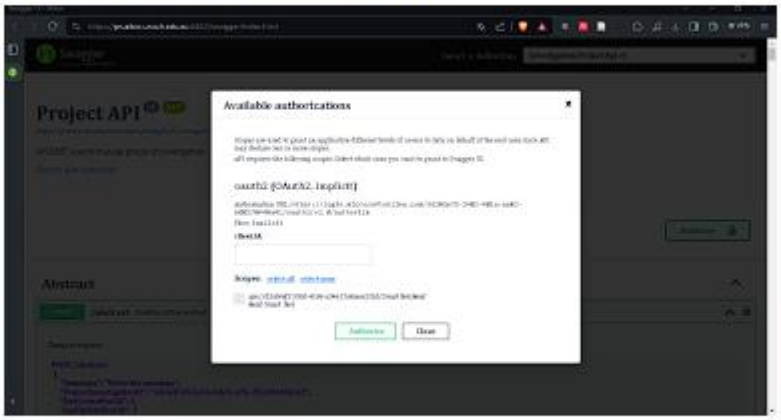
En su versión V1 Swagger gestiona proyectos de investigación

mailto:alex.asitimbay@unach.edu.ec

El API REST consume y trabaja con los datos accesibles por la entidad de CODESI.

3.1 Autorización

Para acceder a la API, necesitas autorización mediante OAuth2 con un flujo implícito y además pertenecer al grupo de desarrollo. Se debe proporcionar un ID de cliente, especificados en la URL de autorización. En este caso, el alcance es api://d3cd46f2-595d-4506-a346-f1c8aece335d/UnachTest.Read.



3.2 Descripción del API

Para el método POST, Swagger envía el método seguido de "/" y la entidad correspondiente, tal como se muestra en la imagen.

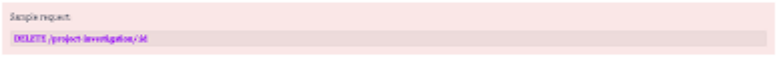
Luego se presenta un ejemplo que incluye los campos correspondientes de esa entidad, junto con los tipos de valores válidos para cada uno.

Para el método PATCH, que representa una actualización, se envía el método seguido de "/" y la entidad a actualizar. Después de la entidad, se requiere proporcionar el ID del elemento que se va a actualizar.

También se encontrará un ejemplo que ilustra la estructura de esa entidad junto con los valores correspondientes, en concordancia con su ID.

El método DELETE opera utilizando el ID de la entidad que se va a eliminar.

En el ejemplo de solicitud, se muestra cómo será la ruta para eliminar la entidad, incluyendo su ID.

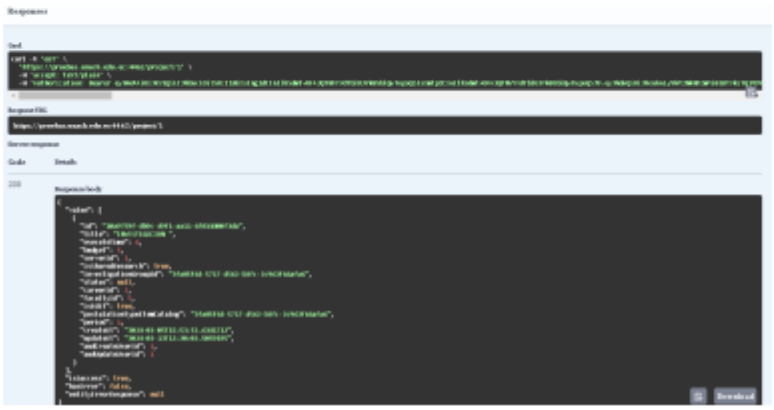


Para el método GET, se solicitará un parámetro que identifique las entidades existentes, o la única entidad según el caso.

En el ejemplo, se mostrará el parámetro correspondiente seguido de su entidad.

3.3 Respuesta del API

La respuesta se proporciona a través de Curl, que incluye el método enviado, la dirección, el tipo de aceptación ("text/plain"), y la autorización mediante un token. Luego, se presenta detalles sobre los valores enviados a través de la API.



4. EndPoints Generados

Los endpoints generados fueron creados de acuerdo con los requisitos para satisfacer las necesidades de la Dirección de Investigación de la Universidad Nacional de Chimborazo.