



UNIVERSIDAD NACIONAL DE CHIMBORAZO
FACULTAD DE INGENIERIA
CARRERA DE SISTEMAS Y COMPUTACIÓN

Proyecto de Investigación previo a la obtención del título de Ingeniero en Sistemas y
Computación

TRABAJO DE TITULACIÓN

**APLICACIÓN WEB PARA EL SERVICIO DE TRÁMITES ACADÉMICOS
DE LA UNACH USANDO UNA ARQUITECTURA BASADA EN
MICROSERVICIOS**

Autor:

Alvarado Zambrano, Erick Alexis

Tutor:

MsC. Pamela Alexandra Buñay Guisñan

Riobamba – Ecuador
2022

DERECHOS DE AUTORÍA

Yo, **ERICK ALEXIS ALVARADO ZAMBRANO**, con cédula de ciudadanía **2350531055**, autor del trabajo de investigación titulado: “**APLICACIÓN WEB PARA EL SERVICIO DE TRÁMITES ACADÉMICOS DE LA UNACH USANDO UNA ARQUITECTURA BASADA EN MICROSERVICIOS**”, certifico que la producción, ideas, opiniones, criterios, contenidos y conclusiones expuestas son de mí exclusiva responsabilidad.

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

En Riobamba, 14 de diciembre del 2022.



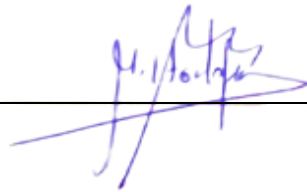
Erick Alexis Alvarado Zambrano
C.I: 2350531055

DICTAMEN FAVORABLE DEL TUTOR Y MIEMBROS DE TRIBUNAL

Quienes suscribimos, catedráticos designados Tutor y Miembros del Tribunal de Grado para la evaluación del trabajo de investigación “APLICACIÓN WEB PARA EL SERVICIO DE TRÁMITES ACADÉMICOS DE LA UNACH USANDO UNA ARQUITECTURA BASADA EN MICROSERVICIOS”, presentado por Erick Alexis Alvarado Zambrano, con cédula de identidad número 2350531055, certificamos que recomendamos la APROBACIÓN de este con fines de titulación. Previamente se ha asesorado durante el desarrollo, revisado y evaluado el trabajo de investigación escrito y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable firmamos, en Riobamba 14 de diciembre de 2022.

MsC. Milton López
PRESIDENTE DEL TRIBUNAL DE GRADO



MsC. Ana Congacha
MIEMBRO DEL TRIBUNAL DE GRADO



PhD. Ximena Quintana
MIEMBRO DEL TRIBUNAL DE GRADO



MsC. Pamela Buñay
TUTOR

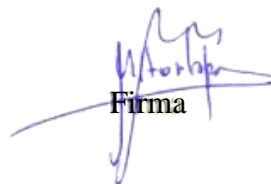


CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

Quienes suscribimos, catedráticos designados Miembros del Tribunal de Grado para la evaluación del trabajo de investigación “Aplicación web para el servicio de trámites académicos de la UNACH usando una arquitectura basada en microservicios” por Erick Alexis Alvarado Zambrano, con cédula de identidad número 2350531055, bajo la tutoría de la 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 14 de diciembre de 2022.

Presidente del Tribunal de Grado
MsC. Milton López.



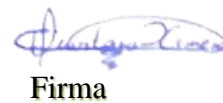
Firma

Miembro del Tribunal de Grado
MsC. Ana Congacha.



Firma

Miembro del Tribunal de Grado
PhD. Ximena Quintana.



Firma

CERTIFICADO ANTIPLAGIO

Original



Dirección
Académica
VICERRECTORADO ACADÉMICO



UNACH-RGF-01-04-02.20
VERSIÓN 02: 06-09-2021

CERTIFICACIÓN

Que, **ALVARADO ZAMBRANO ERICK ALEXIS** con CC: **2350531055**, estudiante de la Carrera **SISTEMAS Y COMPUTACIÓN, NO VIGENTE**, Facultad de **INGENIERÍA**; ha trabajado bajo mi tutoría el trabajo de investigación titulado "**APLICACIÓN WEB PARA EL SERVICIO DE TRÁMITES ACADÉMICOS DE LA UNACH USANDO UNA ARQUITECTURA BASADA EN MICROSERVICIOS**", cumple con el 4 %, de acuerdo al reporte del sistema Anti plagio **URKUND**, porcentaje aceptado de acuerdo a la reglamentación institucional, por consiguiente autorizo continuar con el proceso.

Riobamba, 24 de noviembre de 2022



Firmado electrónicamente por:
**PAMELA
ALEXANDRA BUNAY
GUISÑAN**

MsC. Pamela Alexandra Buñay Guisñan
TUTORA DE TRABAJO DE INVESTIGACIÓN

DEDICATORIA

El presente proyecto de investigación se lo dedico especialmente a mi mamá Patricia, por el apoyo incondicional que me ha brindado durante todo este largo proceso de carrera universitaria, a mi abuelito Asmen por haberme criado como su propio hijo y haberme inculcado muchos valores durante toda mi vida, a mi hermana Karen por ser mi motivo fundamental de superación, a mis amigos con los que compartimos momentos únicos y que llegaron para quedarse, gracias por todo, los llevo en mi corazón.

A mis padrinos Elisa Ortega y Pedro Gallegos quienes han sido el motor de aliento de mi ser, brindándome su amor de padres, apoyo incondicional, por todas sus enseñanzas y valores impartidos sobre mí, quienes, con sus consejos, sacrificio, y apoyo me ha permitido seguir adelante en mi carrera universitaria, contribuyendo en el logro de este paso de mi vida.

AGRADECIMIENTO

A Dios por permitirme cumplir con mi objetivo, a mi madre, a mi abuelo y mis padrinos que fueron mi apoyo incondicional, a la Universidad Nacional de Chimborazo por permitirme realizar este proyecto en sus instalaciones, y darme la oportunidad de adquirir nuevos conocimientos para mi vida profesional.

Un especial agradecimiento a la MsC. Pamela Alexandra Buñay, Tutora de tesis, por su apoyo incondicional, y de igual manera a mis tutores colaboradores, MsC. Anita Congacha y MsC. Ximena Quintana, por todo su apoyo en esta etapa de culminación.

Mi gratitud inmensa a todo el personal del departamento de CODESI de la Universidad Nacional de Chimborazo, Ing. Henry Paca, Ing. Vicente Anilema e Ing. Natalia Crespo, por brindarme su confianza, y haber impartido sus conocimientos conmigo para el desarrollo y culminación del presente trabajo de investigación.

INDICE GENERAL

CAPÍTULO I. INTRODUCCIÓN	14
1.1 Planteamiento del problema	15
1.2 Justificación	16
1.3 Objetivos	17
1.3.1 Objetivo general	17
1.3.2 Objetivos específicos	17
CAPÍTULO II. MARCO TEÓRICO	18
2.1 Arquitectura de software	18
2.2 Tipos de arquitectura	18
2.2.1 Arquitectura orientada a servicios (SOA)	18
2.2.2 Interfaz de programación de aplicación (API)	19
2.2.3 Arquitectura de microservicios	19
2.2.4 Filosofía de los microservicios	22
2.2.5 Principios de diseño de los microservicios	24
2.2.6 Desafío de los microservicios	26
2.2.7 Componentes de los microservicios	29
2.2.8 Comparación entre la arquitectura de software SOA y microservicios.	30
2.3 Aplicaciones web y sus beneficios	31
2.4 Desafío de interfaces para aplicaciones web	32
CAPÍTULO III. METODOLOGÍA	34
3.1 Tipo de investigación	34
3.2 Método de investigación	34
3.3 Técnicas de recolección de datos	34
3.4 Población de estudio y tamaño de muestra	35
3.5 Métodos de análisis y procesamiento de datos	35
3.6 Operacionalización de variables	35

3.7	Procedimiento de la investigación	38
CAPÍTULO IV. RESULTADOS		55
4.1	Resultados.....	55
4.2	Valoración de Indicadores	56
4.2.1	Eficacia	56
4.2.2	Tiempo de respuesta.....	57
4.2.3	Utilización de recursos	57
4.2.4	Comparación de los parámetros del estudio con los propuestos por el modelo FURPS	58
4.3	Discusión.....	59
5	CONCLUSIONES.....	60
6	RECOMENDACIONES.....	61
7	REFERENCIAS BIBLIOGRÁFICAS	62
8	ANEXOS	64
8.1	Anexo 1: Código HTML.....	64
8.2	Anexo 2: Arquitectura de microservicios	67
8.3	Anexo 3: Manual de Usuario	68

INDICE DE TABLAS

Tabla 1. Cuadro comparativo de monolítica y microservicio.	31
Tabla 2. Operacionalización de variables.....	36
Tabla 3. Roles y funcionalidades del personal	38
Tabla 4. Requerimientos funcionales	39
Tabla 5. Requerimientos no funcionales	41
Tabla 6. Product Backlog (lista de tareas).....	42
Tabla 7. Spring Backlog.....	43
Tabla 8. Cantidad de procedimientos realizados.....	55
Tabla 9. Requerimientos solicitados al aplicativo.....	56
Tabla 10. Uso de recursos	58
Tabla 11. Comparación de valores del modelo FURPS vs. valores del estudio	58

INDICE DE FIGURAS

Figura 1. Patrón básico de arquitectura de Microservicios	20
Figura 2. Beneficios de los microservicios.	22
Figura 3. Modelado de la Base de Datos.....	45
Figura 4. Diagrama de Base de Datos	46
Figura 5. Caso de uso del sistema de reporte académico	46
Figura 6. Caso de uso de estudiante	47
Figura 7. Diseño interno de un microservicio CRUD.....	47
Figura 8. Diseño de un microservicio CRUD.	48
Figura 9. Creación de un proyecto de API web de ASP.NET Core en Visual Studio 2019... ..	49
Figura 10. Dependencias en un microservicio API Web de CRUD.	49
Figura 11. Diseño de la clase para el récord académico.	50
Figura 12. Sesión con la base de datos.....	50
Figura 13. Uso de LINQ para recuperar los datos de la base de datos.	51
Figura 14. Configuración en el proyecto Web API.....	51
Figura 15. Creación de gestores de información.....	52
Figura 16. Estructura de la arquitectura de microservicio.	52
Figura 17. Prototipo del reporte del récord académico universitario	53
Figura 18. Módulo para el record académico universitario	54
Figura 19. Módulo para el record académico universitario	54
Figura 20. Porcentaje de éxito en la evaluación inicial de la aplicación.....	55
Figura 21. Eficacia del aplicativo.....	56
Figura 22. Tiempo de respuesta del aplicativo.....	57
Figura 23. Uso de recursos.....	58

RESUMEN

Los requerimientos constantes de actualización de procesos que manejan las secretarías de las carreras en la Universidad Nacional de Chimborazo visibilizan la necesidad de generar un aplicativo que permita la obtención de certificados académicos que minimice los tiempos de atención a los estudiantes. En función a la necesidad hallada la investigación se enfoca en la construcción de un aplicativo web para el servicio de trámites académicos de la UNACH con soporte en la metodología de desarrollo SCRUM con la arquitectura basada en microservicios, se consideró durante la fase de planificación requerimientos funcionales para el módulo y las tareas fueron asignadas mediante Product Backlog, el rendimiento del aplicativo se evaluó con la herramienta Jmeter y sus características de calidad fueron validadas con el modelo de calidad FURPS, se obtuvo una eficacia del 95%, un tiempo de respuesta de 5sg y el 25% de uso de recursos, el antecedente deja claro que el aplicativo cumple con el criterio de calidad.

Palabras Clave: Microservicios, Modelo Furps, Rendimiento.

ABSTRACT

The continuous requirements to update processes handled by the secretaries of the careers at the National University of Chimborazo make visible the need to generate an application that allows obtaining academic certificates that minimizes attention times to students. Depending on the need found, the research focuses on the construction of a web application for the service of academic procedures of the UNACH with support in the SCRUM development methodology with the architecture based on microservices, functional requirements considered during the planning phase. for the module and the tasks assigned through the Product Backlog, among the non-functional requirements, security, availability, compatibility and scalability characteristics assembled, the performance of the application evaluated with the Jmeter tool and its quality characteristics were validated with the model of FURPS quality, an efficiency of 95%, a response time of 5 seconds and 25% use of resources obtained, the background makes it clear that the application meets the quality criteria.

Keywords: Microservices, Furps Model, Performance.



firmado electrónicamente por:
MARITZA DE LOURDES
CHAVEZ AGUAGALLO

Reviewed by:

Mgs. Maritza Chávez Aguagallo

ENGLISH PROFESSOR

c.c. 0602232324

CAPÍTULO I. INTRODUCCIÓN

En los últimos años, las arquitecturas de software para la construcción de aplicaciones web referentes a procesos administrativos y de difusión de resultados han evolucionado maximizando los beneficios de la tecnología, el cambio de sistemas primitivos a sistemas de servicio reutilizables permite la reutilización de varios componentes con interfaces estándar tanto para su llamado como para la creación de funcionalidades complejas (Lopez, 2017).

En el contexto internacional que conoce que las multinacionales como Amazon Primer y Netflix utilizan software implementados con arquitectura basados en microservicios, de hecho la televisora y productora Amazon se constituyó en 1994 con la idea de promocionar y fortalecer la filosofía de microservicios, los resultados permitieron exhibir al colectivo mundial que los equipos pequeños se desenvuelven en condiciones más óptimas que los equipos grandes y entre sus líneas de producción apostaron por la creación de varias herramientas que apoyan desarrolladores en el centro de servicios Amazon Web Services; Netflix por su parte se ha posesionado como referente principal de libros y artículos científicos enfocados en el uso de microservicios a gran escala. Algunas de sus obras se titulan Building Microservices y Microservices, el reconocimiento para la empresa por parte de los usuarios radica también en el hecho de que la productora a través de su Open Source Software Center brindan herramientas gratuitas para los desarrolladores de software (Rodríguez, 2019).

En el Ecuador, las instituciones públicas y privadas demandan de forma constante software que les permitan automatizar procesos, en el año 2016 el departamento de Tecnologías de la Información y Comunicación (CGTIC) de la Asamblea Nacional del Ecuador (ANE) empleó arquitectura de microservicios para organizar y minimizar el tiempo de despacho de procesos gubernamentales constantes.

El contexto anterior resalta la importancia de la arquitectura de microservicios donde las aplicaciones creadas abarcan un conjunto de servicios más pequeños, cada uno de ellos con

su propio flujograma y ejecución de proceso, la independencia de resultados se construyen alrededor de funcionalidades de negocio y despliegue de sistemas de automatización, un adicional representa la versatilidad de su lenguaje de programación y tecnologías de almacenamiento de base de datos (Lopez, 2017), hay que destacar que los microservicios se comunican entre sí a través de APIs, y cuentan con sistemas de almacenamiento propios, lo que evita la sobrecarga y caída de la aplicación.

Con la finalidad de implementar la arquitectura de microservicios en el área académica de la Universidad Nacional de Chimborazo (UNACH) el presente proyecto implementa el servicio de gestión de tramites académicos que permite sistematizar procesos a los estudiantes, así también la creación de un aplicativo web que permite sistematizar procedimientos de almacenamiento de forma que la UNACH cuente con un repositorio de documentos en su sistema académico para la gerencia de trámites internos.

Los beneficiarios directos de la investigación son los estudiantes de la UNACH quienes a través de las ventajas tecnológicas pueden disminuir significativamente el tiempo de procesos académicos y con ello agilizar documentación de matrícula, titulación, etc.

1.1 Planteamiento del problema

En los últimos años, los microservicios han tenido un gran impacto en todas las empresas de todo el mundo trayendo muchas ventajas cuando se usan apropiadamente. Actualmente, la más importante es permitir a los equipos de trabajo realicen sus actividades de forma independiente y que los desarrolladores de software ya no necesiten comprender el contexto de un sistema completo, ayudando con la carga cognitiva. (DevOps, 2021)

En el Ecuador, la Universidad Técnica de Ambato (UTA), cuenta con un sistema académico que le ha ayudado con el proceso de la gestión de trámites académicos para cada carrera, permitiendo una mejora continua con el desarrollo institucional con el fin de que los estudiantes puedan acceder a cualquier tipo de documentación, únicamente con la autenticación

personal mediante el sistema integrado (SI), evitando acudir personalmente a la institución y así solicitar cualquier documentación que requiera. (UTA, 2022)

Las secretarías de cada carrera de la Universidad Nacional de Chimborazo (UNACH), no cuenta con un módulo de gestión de trámites académicos, esto implica que los estudiantes deben acercarse de forma presencial para solicitar cualquier tipo de trámite académico, generando una serie de inconvenientes (pérdida de documentos, trámites rechazados u oficios mal redactados) al momento de solicitar algún documento, causando un retraso al presentar dicha documentación, por tal razón, se ha planteado implementar en el Sistema Informático de Control Académico (SICOA), un módulo adicional que sistematice los procesos como: récord académico, certificado de culminación de estudios, certificado de no adeudar al departamento de TICs, certificado de matrícula y otros que puedan requerirse; generando al final un reporte del documento solicitado.

El desarrollo de la aplicación web permite integrar tecnologías de la información y comunicación para obtener un software usable que gestiona la información de servicios de trámites académicos de la UNACH utilizando la metodología de desarrollo SCRUM con la arquitectura basada en microservicios.

¿Cómo la arquitectura de microservicios influirá en el rendimiento de la aplicación web para el servicio de los trámites académicos en la UNACH?

1.2 Justificación

El estudio persigue la propuesta de implementación de una nueva arquitectura de software basada en microservicios que admite la construcción de aplicaciones web en la Universidad Nacional de Chimborazo con énfasis en el almacenamiento y organización de tramites académicos que requieren del respaldo tecnológico e innovador.

La importancia que mantiene la arquitectura de software basada en microservicios radica en múltiples beneficios, elimina desafíos asociados a las aplicaciones monolíticas, ofrece

a los operadores de redes la capacidad de implementación de tecnologías modernas a escala web, reduce el uso de recursos, añade ágilmente funciones e integrar soluciones desarrolladas por terceros.

La propuesta permite al colectivo administrativo ser eficiente al momento de dar respuesta a los distintos requerimientos solicitados por los estudiantes, así también contar con la flexibilidad de manipular dos o más procesos a la par sin saturar la eficiencia de los ordenadores.

1.3 Objetivos

1.3.1 Objetivo general

- Desarrollar una aplicación web para el servicio de trámites académicos de la UNACH usando una arquitectura basada en microservicios.

1.3.2 Objetivos específicos

- Analizar la arquitectura de microservicio para determinar las fases y actividades que se deben aplicar dentro del desarrollo de una aplicación web de servicio de trámites académicos.
- Aplicar la arquitectura de microservicios en la aplicación web para el servicio de trámites académicos de la UNACH.
- Evaluar el rendimiento de la aplicación web usando el modelo de calidad de FURPS.

CAPÍTULO II. MARCO TEÓRICO

2.1 Arquitectura de software

En la actualidad, el software está presente en gran cantidad de objetos tecnológicos tales como: televisión, PSP, teléfonos, smartwatch y otros dispositivos que actualmente lleva el hombre de forma casi permanente, hasta los sistemas que controlan las operaciones de organizaciones de toda índole o los que operan las sondas robóticas que exploran otros planetas. Uno de los factores clave del éxito de los sistemas es su buen diseño; de manera particular, el diseño de lo que se conoce como arquitectura de software. (Maceda, 2016)

Al igual que con diversos términos en ingeniería de software, no existe una definición universal del concepto de arquitectura de software. Sin embargo, la definición general siguiente que propone el Instituto de Ingeniería de Software (SEI, por sus siglas en inglés) tiende a ser aceptada ampliamente. (Maceda, 2016)

La arquitectura de software de un sistema es el conjunto de estructuras necesarias para razonar sobre el sistema. Comprende elementos de software, relaciones entre ellos, y propiedades de ambos. (Bass, Clements y Kazman, 2012)

El desarrollo de una arquitectura de software puede ser visto como un proceso de selección, adaptación y combinación de patrones. El arquitecto de software debe decidir cómo instanciar un patrón, cómo hacerlo encajar en el contexto específico y las limitaciones del problema. Mark Richards escribió un libro llamado “Software Architecture Patterns” según el que existen 5 patrones principales de arquitectura de software: microkernel, microservicios, arquitectura en capas, basada en eventos y basada en el espacio. (Richards, 2015)

2.2 Tipos de arquitectura

2.2.1 Arquitectura orientada a servicios (SOA)

SOA, es un paradigma tecnológico ampliamente difundido, que a través de los años ha apoyado a múltiples organizaciones en el logro de objetivos estratégicos cada

vez más ambiciosos. SOA, está formada por componentes disponibles a través de interfaces genéricas y protocolos estandarizados y preferentemente libres de licencias (servicios), diseñados con el menor nivel de dependencia posible con los sistemas de información que los consume y de la parte técnica. (Rosado & Jaimes, 2017)

2.2.2 Interfaz de programación de aplicación (API)

Se podría decir que, en un mundo conectado, las API son el pegamento que une todas las piezas que hacen que la vida diaria funcione. Particularmente en el sector financiero las API abiertas están transformándolo, haciéndolo menos hermético y permitiendo que distintos actores compartan recursos para crear servicios innovadores. Cualquier compañía que trabaje con datos de medios de pago o financieros de sus clientes tienen que acceder a los sistemas de otras empresas para validar y realizar operaciones y es aquí que entran a jugar las API (Bogotá, 2019).

2.2.3 Arquitectura de microservicios

Es un enfoque para el desarrollo de una aplicación única como un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y mecanismos ligeros de comunicación, a menudo un recurso de una interfaz de programación de aplicaciones (API) sobre protocolo de transferencia de hipertexto (HTTP). Estos servicios están contruidos alrededor de las capacidades del negocio y con independencia de despliegue e implementación totalmente automatizada. Existe un mínimo de gestión centralizada de estos servicios, los que pueden estar escritos en lenguajes de programación diferentes y utilizar diferentes tecnologías de almacenamiento de datos. (Lewis & Fowler, 2014).

El término microservicios no es relativamente nuevo, este estilo arquitectural fue dicho por Martin Fowler en un taller de arquitectos de software como una descripción del nuevo campo que los participantes estaban explorando. No existe una definición en

concreto para microservicio, sin embargo, una aproximación que la realiza Newman lo define como: “pequeños servicios autónomos que trabajan juntos”. (Newman, 2015)

La arquitectura de microservicio consiste en dividir una aplicación o sistema en unidades más pequeñas, la modulación facilita la automatización y proporciona medios precisos de abstracción. Los microservicios se pueden reemplazar fácilmente por módulos de sistemas monolíticos. Simplemente reemplazar microservicios reduce el costo de malas decisiones, si el microservicio está construido con una tecnología o enfoque, se puede sobrescribir si es necesario. (Newman, 2015).

Una arquitectura de microservicios promueve el desarrollo y despliegue de aplicaciones compuestas por unidades independientes, autónomas, modulares y auto-contenidas, lo cual difiere de la forma tradicional o monolítico. (Richards, 2015)



Figura 1. Patrón básico de arquitectura de Microservicios

Fuente (Richards, 2015)

2.2.3.1 Características

Algunas de las características principales de la arquitectura de microservicios según Wolff (2017) son:

Agilidad general. La agilidad general es la capacidad de responder rápidamente a un entorno en constante cambio. Las aplicaciones creadas con este patrón tienden a tener un acoplamiento muy flexible, lo que también ayuda a facilitar el cambio.

Facilidad de despliegue. Las características de implementación del patrón de microservicios tienen una tasa muy alta debido a la naturaleza detallada e independiente de los servicios remotos.

Testabilidad. Este patrón está débilmente acoplado, hay muchas menos posibilidades desde una perspectiva de desarrollo de realizar un cambio que rompa otra parte de la aplicación, lo que alivia la carga de prueba de tener que probar toda la aplicación para un pequeño cambio.

Rendimiento. Puede crear aplicaciones implementadas a partir de este patrón que funcionen muy bien, en general, este patrón no se presta naturalmente a aplicaciones de alto rendimiento debido a la naturaleza distribuida del patrón de arquitectura de microservicios.

Escalabilidad. La aplicación se divide en unidades implementadas por separado, cada componente del servicio se puede escalar individualmente, lo que permite una escalabilidad precisa de la aplicación.

Facilidad de desarrollo. Hay muchas menos posibilidades de que un desarrollador realice un cambio en un componente del servicio que afectaría a otros componentes del servicio, lo que reduciría la coordinación necesaria entre los desarrolladores o los equipos de desarrollo.

Beneficios

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

microservicios permiten a las empresas gestionar las aplicaciones de código base grande usando una metodología más práctica donde las mejoras incrementales son ejecutadas por pequeños equipos en bases de código y despliegues independientes. La agilidad, reducción de costes y la escalabilidad granular, traen algunos retos de los sistemas distribuidos y las prácticas de gestión de los equipos de desarrollo que deben ser considerados. (Maya & López, 2018)

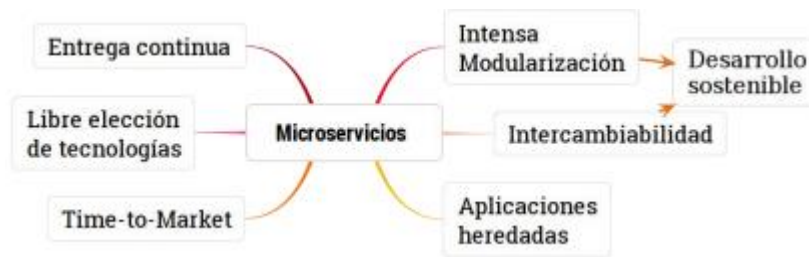


Figura 2. Beneficios de los microservicios.

Fuente: (Maya & López, 2018)

2.2.4 Filosofía de los microservicios

La revolución digital está innovando todas las industrias y esto se refleja en el vertiginoso proceso de adaptación que mantienen las empresas ante las condiciones comerciales cambiantes sin dejar de lado la misión de generar valor para los usuarios. La esencia de la transformación digital radica en una mentalidad de crecimiento, donde las organizaciones invierten en perfeccionar sus procesos, sistemas y capacidades internas para generar cambios que impulsen los negocios y resultados en la sociedad. Muchos de estos sistemas organizacionales necesitan un cambio que garantice reinventar el futuro para ofrecer experiencias emocionantes con empleados empoderados, clientes cautivados, productos innovadores y operaciones optimizadas (Ovais et al., 2019).

En las últimas décadas, la mayoría de estos sistemas fueron construidos en base a una arquitectura monolítica que conforme iba creciendo y se volvía compleja resultaba

relativamente difícil de cambiar y mantener. Posteriormente llegó la arquitectura orientada a servicios (SOA) como una gran mejora con respecto a la arquitectura monolítica (Nadareishvili et al., 2016).

SOA está más direccionada en la reutilización e integración, donde los servicios intercambian mensajes constantemente a cambio de tareas para ser ejecutadas. La plataforma de mensajería es la columna vertebral de SOA y es responsable del descubrimiento de servicios, orquestación, enrutamiento, transformación de mensajes, enriquecimiento de mensajes, seguridad y gestión de transacciones (Ovais et al., 2019).

Sin embargo, cuando la arquitectura SOA crecía se convertía en una aplicación monolítica distribuida, que nuevamente era difícil de mantener o ampliar; a esto se sumaba una de sus principales desventajas en donde la información se compartía y el conocimiento del dominio comercial estaba disperso en todo el bus de servicio empresarial (ESB), lo que dificultaba el cambio de servicio (Nadareishvili et al., 2016).

Posteriormente los microservicios surgen como una alternativa evolutiva del estilo de arquitectura que aborda los puntos débiles de otros estilos de arquitectura cambiando el paradigma de desarrollo de aplicaciones pues los esfuerzos de concentrar y segregar las capacidades comerciales en servicios individuales permitieron a las organizaciones construir sistemas que son de naturaleza modular, aislada y débilmente acoplada (Mena et al., 2019).

Estas características juegan un papel crucial que ayuda a las organizaciones a construir equipos enfocados en entregar velocidad. Esto se logra gracias a que los equipos están aislados para desarrollar e implementar microservicios de forma independiente sin ningún requerimiento de colaboración importante, en otras palabras son servicios con autonomía que aceptan cambios y que reducen el riesgo de fallas debido a esta característica de independencia (Ovais et al., 2019).

2.2.5 Principios de diseño de los microservicios

Los principios de diseño de microservicios proporcionan las directrices para evaluar las decisiones clave que pueden afectar el diseño de arquitectura basada en microservicios. Algunos principios dan soporte al acoplamiento y modularidad de los servicios mientras que otros rigen el diseño de una arquitectura de microservicios (Martínez et al., 2018).

Principio de responsabilidad única y diseño dirigido por dominio. Un microservicio debe ser responsable de brindar una característica única o un conjunto de características afines para ofrecer una herramienta empresarial. La única razón para que una interfaz de microservicio cambie debería darse por cambios en las capacidades de negocio que ofrece el microservicio. Esto asegura que los sistemas sean diseñados siguiendo los dominios del mundo real y ayuda a visualizar sistemas y arquitecturas como traducción de problemas reales (Ovais et al., 2019).

Encapsulación y segregación de interfaces. Cada microservicio posee datos y la única forma de que un servicio pueda comunicarse con otros servicios es a través de interfaces bien definidas. Estas interfaces deben diseñarse cuidadosamente teniendo en cuenta a los clientes. Una de las opciones más populares es la inserción de una puerta de enlace API la cual será responsable de la comunicación sobre las consideraciones finales de la aplicación para el cliente manteniendo los microservicios enfocados en entrega de capacidades comerciales (Mena et al., 2019).

Cultura de autonomía, propiedad y gobernanza compartida. La arquitectura de microservicios permite que las capacidades empresariales propiedad de diferentes equipos pueden trabajar de forma emancipada sin requerir mucha colaboración entre equipos. Cada equipo no requiere la asignación de una capacidad comercial exclusiva; en su lugar, pueden elegir entre un conjunto relacionado de capacidades que forman

parte de un solo dominio. La arquitectura de microservicios florece cuando se permite que los equipos tengan autonomía, ya que pueden elegir lo que creen que es correcto para entregar al negocio (Nadareishvili et al., 2016).

Sin embargo, esto no significa que los equipos puedan hacer cualquier cosa, pero ciertamente les da la libertad de tomar decisiones bajo una red de principios previamente acordados. Estos principios se denominan gobernanza compartida, que proporcionan un consenso entre los equipos con respecto a la manera de abordar diferentes preocupaciones transversales, o el medio para intentar diferentes tecnologías (López y Maya, 2017).

Desplegable de forma independiente. Cada microservicio debe poder implementarse de forma independiente permitiendo que los equipos realicen cambios sin que estos afecten a otros microservicios. En relación con la gobernanza compartida, los equipos deben seguir buscando tecnologías y prácticas que contribuyan a una mayor independencia. Este principio es extremadamente útil para operacionalizar microservicios a gran escala (Ovais et al., 2019).

Cultura de la automatización. La automatización es un concepto importante que promueve la idea de encontrar oportunidades que reemplacen pasos manuales con diferentes formas de automatización como integración e implementación continua, pruebas automatizadas y automatización de la infraestructura para lograr consistencia y reducir los gastos generales de gestionar microservicios (Valero, 2022).

Diseñando para los fracasos. Los microservicios están diseñados para ser altamente disponibles y escalables, es decir los servicios deben diseñarse para que en caso de fallos recuperen su funcionalidad tan rápido como sea posible. La idea es que no existan fallos en cascada, la falla de un microservicio no debería afectar a otros microservicios. Sin embargo, las fallas en el servicio son inevitables y requieren un enfoque pragmático que

permita la experimentación para encontrar vulnerabilidades del sistema (Ovais et al., 2019).

Diseñar servicios para el fracaso requiere tener una buena comprensión del comportamiento y las expectativas del usuario y esto se puede lograr con experimentación controlada creando caos para determinar cómo se comportaría el sistema en diferentes circunstancias. Este proceso se cataloga como ingeniería del caos (Nadareishvili et al., 2016).

La ingeniería del caos experimenta con áreas específicas del sistema introduciendo fallas intencionalmente para localizar problemas del sistema no descubiertos. Esto permite solucionarlos antes de que ocurran y afecten al negocio y a los usuarios de manera inesperada. Esto también ayuda a analizar los riesgos e impactos de condiciones turbulentas, las respuesta a incidentes, las brechas en la observabilidad, así como la capacidad del equipo para responder a los incidentes (Mena et al., 2019).

Observabilidad. Es una capacidad que se construye como parte de la arquitectura de microservicio para identificar y razonar sobre el estado interno del sistema, a través de la recopilación de registros, métricas y seguimientos ayuda con la supervisión, depuración, diagnóstico y solución de problemas de microservicios. Otro aspecto importante de la observabilidad es el rastreo distribuido, que ayuda a comprender el flujo de eventos en diferentes microservicios (Martínez et al., 2018).

2.2.6 Desafío de los microservicios

Cultura organizacional. Uno de los principales obstáculos para pasar a los microservicios es la cultura organizacional, ya que inicialmente los equipos se construyeron en torno a las capacidades técnicas en lugar de capacidades empresariales. Esto requiere una evolución en la organización, reestructuración de equipos y un cambio de prácticas heredadas (Ovais et al., 2019).

Adopción de prácticas DevOps. DevOps proporciona un conjunto de prácticas que integra el desarrollo y los equipos de operaciones para entregar valor. La adopción de estas habilidades es importante para cualquier organización ya que conlleva cambios rápidos al mercado, incrementos de frecuencia de implementación, reducción de la tasa de fallas y tiempo más rápido de recuperación dando una gran ventaja de tiempo con software de alta calidad a los usuarios finales (Valero, 2022).

Complejidad arquitectónica y operativa. La arquitectura de microservicios según Ovais (2019) es de naturaleza distribuida, hay más partes móviles y se requiere más experiencia para que los equipos las manejen esto hace que se presenten varios desafíos en comparación con una arquitectura monolítica. Algunos de estos desafíos se mencionan a continuación:

Comunicación no confiable a través de los límites del servicio. Los Microservicios dependen en gran medida de la red subyacente. Por lo tanto, la infraestructura de red tiene que estar correctamente diseñada y gobernada para atender las necesidades de comunicación, así como para proteger la infraestructura de eventos no deseados.

Congestión y latencia de la red. aunque es de manera temporal puede suscitarse un evento donde una red no tenga suficiente ancho de banda para permitir que el tráfico fluya y a causa de esta congestión de la red, diferentes cargas de trabajo pueden experimentar respuestas retrasadas o fallas parciales, lo que resulta en una alta latencia.

Integridad de datos. en una arquitectura de microservicios, la transacción de una sola empresa puede abarcar múltiples microservicios y debido a cualquier falla de la red, si algún servicio falla, puede afectar alguna parte de la transacción creando inconsistencias de datos a través de diferentes microservicios, lo que genera problemas de integridad de datos (Ovais, 2019).

Servicio de orquestación y coreografía. No existe una única forma de especificar cómo se comunican los diferentes servicios entre sí y cómo funciona el sistema en general. La orquestación presenta un único punto de falla mediante el control de la interacción de diferentes servicios, mientras que la coreografía promueve la idea de puntos finales inteligentes y tuberías de descarga, con una desventaja potencial de introducir dependencias de ciclismo. En coreografía, los microservicios publican y consumen mensajes del intermediario de mensajes, que ayuda a que la arquitectura general sea más escalable y tolerante a fallas (Mena et al., 2019; Valero, 2022).

Observabilidad. Administrar, monitorear y controlar microservicios a escala es una tarea difícil. Para comprender el interacción y comportamiento de las diferentes partes del sistema es necesario recoger métricas, registros, pilas de llamadas y generar alertas e implementar el seguimiento distribuido (Mena et al., 2019; Ovais et al., 2019).

Pruebas de extremo a extremo. La prueba de extremo a extremo de los microservicios es más desafiante ya que requiere una comunicación fiable y eficaz para atraer a todos los equipos. La configuración de una prueba es difícil y requiere coordinación entre los equipos pudiendo obstaculizar la frecuencia de lanzamiento (Nadareishvili et al., 2016; López y Maya, 2017).

Período de doble hipoteca. Si se está migrando de una aplicación monolítica a un microservicio, es necesario vivir en un mundo híbrido por un tiempo para apoyar tanto el legado como la nueva aplicación. Esto no es fácil y requiere una planificación cuidadosa cuando se trata de corregir errores y agregar nuevas funciones, lo que afecta la agilidad y productividad (Ovais et al., 2019).

Inversión en plataforma. Las organizaciones necesitan invertir en equipos de plataformas, estos equipos también construyen herramientas y marcos para ayudar a otros equipos a finalizar el trabajo (Nadareishvili et al., 2016).

2.2.7 Componentes de los microservicios

Además de los microservicios, existen otros componentes que juegan un papel vital en la arquitectura de microservicios. Estos microservicios están alojados en una plataforma de orquestación, responsable de garantizar la autocuración y la alta disponibilidad de microservicios que luego se comunican entre sí usando patrones de orquestación o coreografía (Mena et al., 2019).

En la orquestación, un microservicio es responsable de invocar otras interfaces de microservicio mientras en la coreografía, los mensajes se intercambian mediante un bus de eventos. Un cliente puede consumir un microservicio a través de una puerta de enlace API, donde los mensajes se transmiten a un microservicio específico para el procesamiento real, donde diferentes microservicios usan una base de datos SQL, una base de datos en memoria y una base de datos NoSQL para sus datos de almacenamiento (Valero, 2022 ; Ovais et al., 2019).

Mensajes. Los mensajes contienen información necesaria para que los microservicios se comuniquen entre sí. Los mensajes se clasifican como comandos o eventos. Los comandos suelen contener más información para el destinatario, con la expectativa de ser notificado sobre la entrega de mensajes, mientras que los eventos son ligeros y se utilizan principalmente como mecanismos de notificación sin ninguna expectativa por parte del consumidor además puede ser síncronos o asíncronos (Nadareishvili et al., 2016).

Persistencia y gestión de estado. El manejo de datos es un aspecto importante de los microservicios. La mayoría de los microservicios necesitan conservar el estado de los datos. En una arquitectura de microservicios, los datos son descentralizados y cada microservicio tiene la responsabilidad y autonomía de gestionar sus propios datos (Martínez et al., 2018;Ovais et al., 2019).

Orquestación. Un orquestador es responsable de colocar instancias de microservicios en infraestructura de cómputo y también cumple con la función de identificar fallas y microservicios de escala para mantener la alta disponibilidad y resiliencia de la arquitectura general (Nadareishvili et al., 2016).

El descubrimiento de servicios juega un papel importante en mantener la arquitectura de microservicio altamente detectable al permitir que se registren nuevas instancias y se conviertan disponibles para el servicio. Estas instancias se aprovisionan dinámicamente para abordar actualizaciones, fallas y demandas de escalado (Ovais et al., 2019).

Puerta de enlace API. La puerta de enlace API actúa como un proxy inverso, responsable de enrutar las solicitudes a los servicios back end apropiados para exponerlos de manera controlada al mundo exterior para el consumo. También proporciona una gestión robusta y características de seguridad que son útiles para proteger los servicios back end de actores maliciosos. (Ovais et al., 2019) (Mena et al., 2019).

2.2.8 Comparación entre la arquitectura de software SOA y microservicios.

La principal diferencia entre un enfoque SOA y los microservicios es el nivel de granularidad del cual se está hablando: mientras en un enfoque SOA se tienen servicios de granularidad gruesa centrados en funcionalidad de negocio y generalmente con protocolos de comunicación rigurosos; un enfoque de microservicios se orienta en la estructura interna de una aplicación, con servicios de granularidad fina, cada uno de ellos especializado en una funcionalidad específica y haciendo uso de protocolos livianos de comunicación como HTTP mediante APIs RESTFull. (Gómez, 2017).

Tabla 1. Cuadro comparativo de monolítica y microservicio.

	MONOLÍTICA	MICROSERVICIO
Arquitectura	Construido como un único ejecutable lógico (típicamente la parte del lado del servidor de una arquitectura de tres niveles cliente-servidor-base de datos).	Construido como un conjunto de pequeños servicios, cada uno ejecutándose por separado y comunicándose con mecanismos livianos.
Modularidad	Basado en fracturas del lenguaje.	Basado en las capacidades comerciales.
Agilidad	Los cambios en el sistema implican la creación y el despliegue de una nueva versión de toda la aplicación.	Los cambios se pueden aplicar a cada servicio de forma independiente.
Escalabilidad	Toda la aplicación escalada horizontalmente detrás de un balanceador de carga.	Cada servicio se escala de forma independiente cuando fue necesario.
Implementación	Normalmente escrito en un idioma.	Cada servicio implementado en el idioma que mejor se adapta a la necesidad.
Mantenibilidad	Gran base de código que intimida a los nuevos desarrolladores.	Base de código más pequeña y fácil de administrar.

2.3 Aplicaciones web y sus beneficios

De acuerdo con Pawan Bora (2009) las aplicaciones web son una gran tendencia en vista de los beneficios que ofrece:

Facilidad de acceso. El único software que los usuarios necesitan para acceder y utilizar aplicaciones web es un navegador como Internet Explorer, Firefox, Safari y Opera. Además, pueden acceder a las aplicaciones web desde casi cualquier lugar, siempre y cuando la computadora que se encuentre en uso tenga un navegador web y conectividad a Internet (Fowler y Stanwick, 2004).

Facilidad de despliegue. Las aplicaciones web se pueden desarrollar, actualizar y mantener de forma remota sin exigir a los usuarios que las instalen o reinstalen. Por otro lado, tiene un funcionamiento perfecto independiente del sistema operativo en las computadoras de los usuarios. Una vez construido se puede implementar para casi cualquier usuario, no es necesario crear diferentes versiones para Microsoft Windows, Macintosh OS X, GNU/Linux y otros sistemas operativos (Pawan, 2009).

Base de usuarios "entrenados". Se espera que la mayoría de los usuarios de Internet estén familiarizados con la terminología del navegador web, como inicio, atrás, adelante, marcadores, enlaces de hipertexto, botones de envío, etc. Con este conocimiento y el hecho de que el uso de aplicaciones web no requiere instalaciones complejas, las barreras para su uso son mínimas (Ferrer, 2013).

Madurez y confiabilidad de la conectividad de la red y tecnologías web. Los estándares web están mejorando y las inconsistencias del navegador que solían causar frustración para los desarrolladores web están disminuyendo. Además, la conectividad de red y el acceso de banda ancha se está volviendo más confiable, más generalizado y económico (Luján, 2002).

2.4 Desafío de interfaces para aplicaciones web

Los desafíos en la creación de apps web están relacionados con la arquitectura web subyacente, un conjunto limitado de controles interactivos admitidos de forma nativa en los navegadores web y la falta de diseño orientación sobre cómo se debe implementar las interacciones de los usuarios (Pawan, 2009).

Arquitectura web “ligeramente acoplada”. Sucede cuando cada recarga o actualización de página está marcada por retraso causados por la necesidad de establecer la conexión, el servidor para responder a la solicitud, la red para recibir la página y el navegador para recargar la página. Esto crea un experiencia irregular y discontinua para los usuarios de aplicaciones web, como caso específico se puede mencionar a un usuario que navega por una estructura de árbol jerárquico de elementos ya que por cada clic tendrá que esperar un nodo de datos para que la página se vuelva a cargar y verá la página expandida o colapsada (Nixon, 2019).

Conjunto limitado de controles o widgets para admitir diseño de aplicaciones. El soporte nativo para controles a veces es limitado a cuadros de texto, botones de radio, casillas de verificación, listas desplegadas y comando o botones de acción. No ofrece soporte para interacciones sofisticadas y en ciertos casos la falta de herramientas inherente ha llevado a una variedad de implementaciones con presentaciones e interacciones inconsistentes (pp. 1-215).

Enfoques de interacción inconsistentes. Debido a que la mayoría de los sitios web las aplicaciones están diseñadas para ser independientes del navegador, las interacciones y la apariencia no se puede simular para que coincida con todos los sistemas operativos (Pawan, 2009).

CAPÍTULO III. METODOLOGÍA

El objetivo de la presente investigación fue desarrollar una aplicación web que permitió gestionar la información de servicio de trámites académicos de la UNACH utilizando la metodología de desarrollo SCRUM con la arquitectura basada en microservicios.

3.1 Tipo de investigación

La investigación con relación al tipo de variables mantuvo un enfoque cuantitativo, se midió el rendimiento según los criterios establecidos en el modelo de calidad de software FURPS; según el nivel de profundidad en el objeto de estudio fue deductiva debido al paso inicial de búsqueda de información relacionada a las arquitecturas de software hasta llegar al estudio y análisis de la arquitectura basada en microservicios, de acuerdo con el método de investigación fue documental en fundamento a la revisión de distintas fuentes bibliográficas que discuten semejanzas y diferencias de la investigación.

3.2 Método de investigación

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

3.3 Técnicas de recolección de datos

Las técnicas de recolección de datos fueron la observación y entrevista, la observación por su parte se encargó de visibilizar el rendimiento del aplicativo web durante la ejecución de trámites académicos en tanto que la entrevista condensó las percepciones de los usuarios del aplicativo; posterior a ello la herramienta JMeter permitió la recolección de datos referentes a la medición de los criterios de rendimiento establecidos por el modelo de calidad de software FURPS.

3.4 Población de estudio y tamaño de muestra

El colectivo de estudio se constituyó por una población infinita y ante la carencia de un marco muestral delimitado no es posible el cálculo de una muestra, sin embargo, se utilizó la herramienta JMeter para obtener datos referentes a los indicadores especificados por el modelo de calidad de FURPS.

3.5 Métodos de análisis y procesamiento de datos

Se utilizó un análisis exploratorio de datos para la caracterización de variables relacionadas a la eficacia y tiempo de respuesta proporcionadas por la aplicación JMeter.

3.6 Operacionalización de variables

Tabla 2. Operacionalización de variables

Pregunta de investigación	Tema	Objetivos	Variables	Conceptualización	Dimensión	Indicadores
¿Cómo la arquitectura de microservicios influirá en el rendimiento de la aplicación web para el servicio de los trámites académicos en la UNACH?	Aplicación web para el servicio de trámites académicos de la UNACH usando una arquitectura basada en microservicios.	General:	Independiente	La arquitectura de microservicios es un método de desarrollo de aplicaciones software que funciona como un conjunto de pequeños servicios que se ejecutan de manera independiente y autónoma, proporcionando una funcionalidad de negocio completa. (Astorga, 2022)	Arquitectura.	<ul style="list-style-type: none"> ✓ Número de Fases. ✓ Número de Diagramas generados. ✓ Número de esquemas diseñados.
		Desarrollar una aplicación web para el servicio de trámites académicos de la UNACH usando una arquitectura basada en microservicios.	Arquitectura de microservicios.			
		Específicos:	Dependiente	Las aplicaciones web son aquellas herramientas donde los usuarios pueden acceder a un servidor Web a través de la red mediante un navegador determinado. (Valarezo, Honores, Gómez, & Vincés, 2018)	Rendimiento.	<p>Modelo de calidad de FURPS:</p> <ul style="list-style-type: none"> ✓ % de eficacia. ✓ % de tiempo de respuesta. ✓ % de utilización de recursos.
		<ul style="list-style-type: none"> • Analizar la arquitectura de microservicio para determinar las fases y actividades que se deben aplicar dentro del desarrollo de una aplicación web de servicio de trámites académicos. • Aplicar la arquitectura de 	Aplicación web.			

		<p>microservicios en la aplicación web para el servicio de trámites académicos de la UNACH.</p> <ul style="list-style-type: none">• Evaluar el rendimiento de la aplicación web usando el modelo de calidad de FURPS.				
--	--	---	--	--	--	--

3.7 Procedimiento de la investigación

Dentro del análisis de requisitos, se hallaron los requerimientos funcionales y no funcionales correspondientes a la aplicación web para el servicio de trámites académicos de la UNACH con soporte en la metodología de desarrollo SCRUM con la arquitectura basada en microservicios.

Las fases de ejecución fueron:

Inicio: En la fase de inicio se tomó en consideración el análisis de requerimientos, la entidad receptora emitió un listado de requisitos para el desarrollo del proyecto de investigación denominado “APLICACIÓN WEB PARA EL SERVICIO DE TRÁMITES ACADÉMICOS DE LA UNACH USANDO UNA ARQUITECTURA BASADA EN MICROSERVICIOS”.

Personal involucrado

Dentro de la metodología scrum se establece los roles y funcionalidades a cumplir del personal involucrado dentro de proyecto, la cual se detalla en la Tabla 3.

Tabla 3. Roles y funcionalidades del personal

Rol	Descripción	Personal
Product Owner	Es el encargado de comunicar los requerimientos empresariales.	Ing. Henry Paca.
Scrum Master	Es la persona que se encarga de coordinar el equipo y asignar tareas.	MsC. Pamela Buñay.
Team	Es el grupo encargado del desarrollo del sistema.	Sr. Erick Alvarado.

Planificación y estimación

En esta fase se procede a crear las historias de usuario, identificar las tareas (Product Backlog) así como la planificación (Sprint Backlog) que permitió desarrollar con éxito el sistema web.

Requerimientos funcionales

Para la toma de requisitos y especificaciones que la aplicación web debe cumplir, se organizó una entrevista dirigida al Ing. Henry Paca, director del departamento de Coordinación de Desarrollo de Sistemas Informáticos (CODESI). Los requerimientos funcionales están determinados por las funcionalidades que se requiere en la aplicación, estas se decidieron en las reuniones con el personal del CODESI.

Tabla 4. Requerimientos funcionales

Identificación del requerimiento	RF01
Nombre del requerimiento	Certificado del Récord Académico
Descripción del requerimiento	El usuario podrá solicitar mediante el SICOA, el récord académico obteniendo un reporte final de sus calificaciones, hasta el semestre en el que se encuentre.
Prioridad del requerimiento	Alta
Identificación del requerimiento	RF02
Nombre del requerimiento	Certificado de culminación de estudios
Descripción del requerimiento	El usuario podrá solicitar mediante el SICOA, el certificado de culminar sus estudios una vez que haya finalizado la malla curricular.
Prioridad del requerimiento	Alta
Identificación del requerimiento	RF03
Nombre del requerimiento	Certificado de no adeudar al departamento de TICs.
Descripción del requerimiento	El usuario podrá solicitar mediante el SICOA, el certificado de no adeudar al departamento de TICs.

Prioridad del requerimiento	Alta
Identificación del requerimiento	RF04
Nombre del requerimiento	Certificado de matrícula.
Descripción del requerimiento	El usuario podrá solicitar mediante el SICOA, el certificado de estar legalmente matriculado en la carrera.
Prioridad del requerimiento	Alta

Requerimientos no funcionales

Los requisitos no funcionales describen criterios del funcionamiento general del sistema, estos requisitos comprenden características de seguridad, disponibilidad, compatibilidad, escalabilidad, etc.

Tabla 5. Requerimientos no funcionales

Requerimiento	Descripción del requerimiento	Categoría
RNF01	Este tipo de requerimientos indica cómo la aplicación debe responder a los diferentes errores que se puedan presentar.	Control de errores
RNF02	El diseño de las interfaces de usuario a veces se considera como una tarea en la fase de requerimientos.	Interfaces de usuario
RNF03	Estos requerimientos plantean que las aplicaciones no son perfectas, pero limitan las fallas de la aplicación a determinados valores.	Confiabilidad
RNF04	Tiempo en que debe estar disponible la aplicación.	Disponibilidad
RNF05	Medidas de seguridad con relación a procedimientos que impliquen el uso de información vulnerable como, por ejemplo, las claves de acceso al software.	Seguridad

Product Backlog

Es un listado de todas las tareas que se aspira realizar durante el desarrollo del proyecto. En la Tabla 4 se puede observar la lista de tareas definidas para este proyecto, están categorizadas como HT (Historias Técnicas) y HU (Historias de Usuario) con sus respectivas numeraciones y el esfuerzo estimado para cada una en un rango de 1 (menos esfuerzo) al 5 (más esfuerzo).

Tabla 6. Product Backlog (lista de tareas)

Ítem	Tarea	Esfuerzo
HT-01	Análisis de los requerimientos funcionales y no funcionales del sistema.	5
HT-02	Generar las historias de usuario.	4
HT-03	Establecer la arquitectura del sistema	3
HT-04	Diseñar la interfaz de usuario (prototipo)	3
HT-05	Diseño de la base de datos (Modelo entidad-relación, Lógico-físico)	5
HT-06	Instalación y configuración de herramientas para el desarrollo del sistema.	4
HT-07	Generar MVC (Modelos, Vistas y Controladores) de cada entidad con sus atributos y relaciones.	5
HU-01	Desarrollar la base de datos.	5
HU-02	Desarrollar cada módulo para la aplicación web.	4
HU-03	Desarrollar la accesibilidad para la aplicación web.	4
HU-04	Desarrollar la navegabilidad para la aplicación web.	4
HU-05	Generar e imprimir reportes.	5
HU-06	Evaluar la aplicación web por medio del modelo de calidad de FURPS.	4
HU-07	Analizar los resultados obtenidos	4
HT-08	Consumir una API desde la aplicación web	5
HT-09	Realizar el manual técnico y de usuario del sistema web	5
HT-10	Desplegar el sistema en un servidor	5
HT-11	Capacitar a los usuarios y validar el Sistema web	4

Sprint Backlog

En esta sección se ha dividido el trabajo en un elemento denominado Sprint Backlog, presentando las tareas, donde el equipo debe completar en un tiempo estimado.

Tabla 7. 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 sistema	X															
HT-02	Generar las historias de usuario		X														
HT-03	Establecer la arquitectura del sistema			X													
SPRINT 2																	
HT-04	Diseñar la interfaz de usuario (prototipo)			X	X												
HT-05	Diseño de la base de datos (Modelo entidad-relación, Lógico-físico)				X												
HT-06	Instalación y configuración de herramientas para el desarrollo del sistema.					X											
HT-07	Generar MVC (Modelos, Vistas y Controladores) de cada entidad con sus atributos y relaciones.					X	X	X									
SPRINT 3																	
HU-01	Establecer los requisitos para la base de datos.					X											
HU-02	Desarrollar la base de datos.						X	X									

HU-03	Desarrollar los prototipos para cada uno de los módulos.									X							
HU-04	Desarrollar los módulos para la aplicación web.									X							
HU-05	Desarrollar la accesibilidad para la aplicación web.										X						
HU-06	Desarrollar la navegabilidad para la aplicación web.											X					
HU-07	Implementar reportes para cada uno de los procesos.												X	X			
SPRINT 4																	
HT-08	Consumir una API desde la aplicación web										X	X	X				
HT-09	Realizar el manual técnico y de usuario del sistema web													X	X		
HT-10	Desplegar el sistema en un servidor															X	
HT-11	Capacitar a los usuarios y validar el Sistema web																X

Modelado de la base de datos

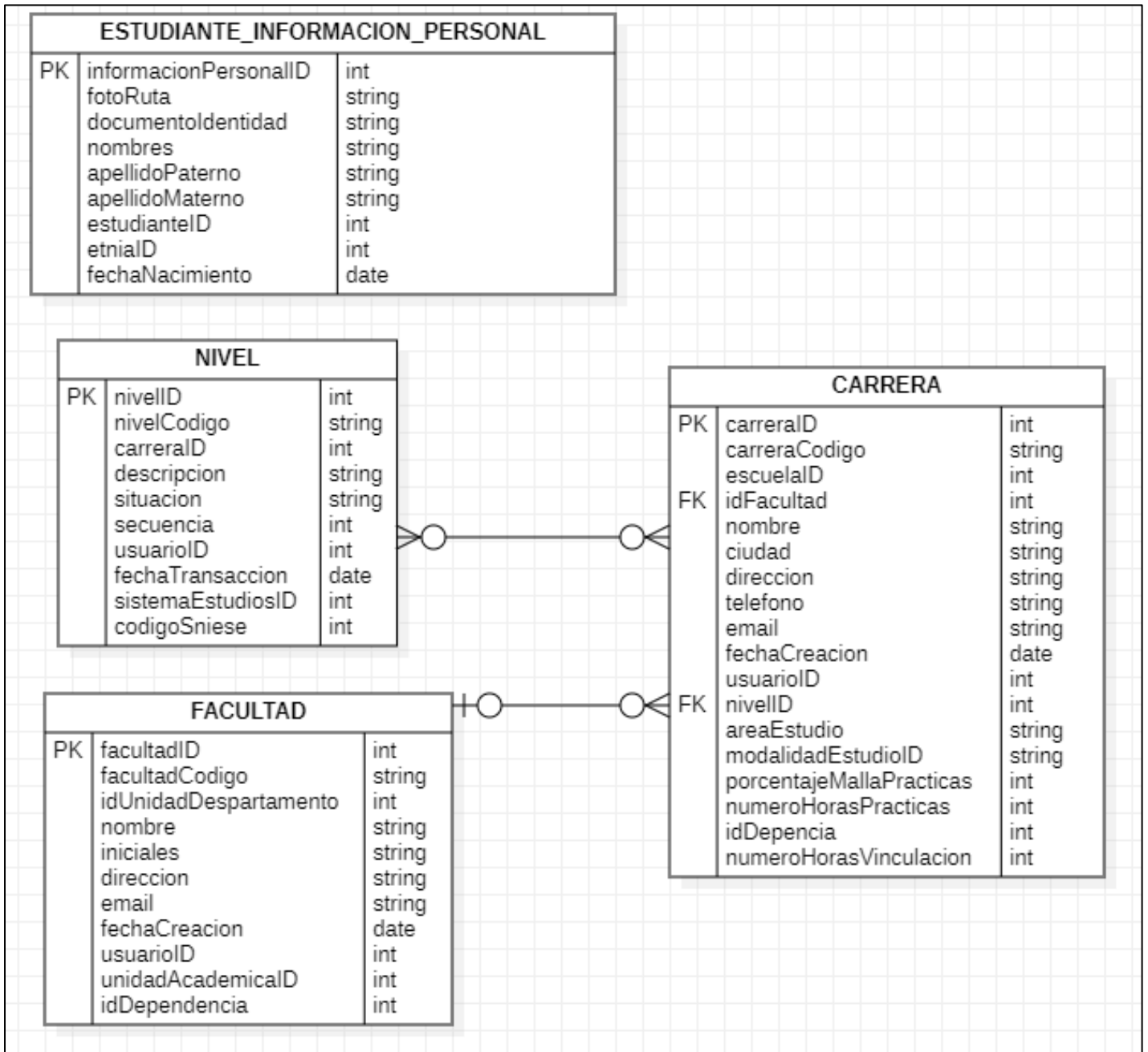


Figura 3. Modelado de la Base de Datos

Diseño

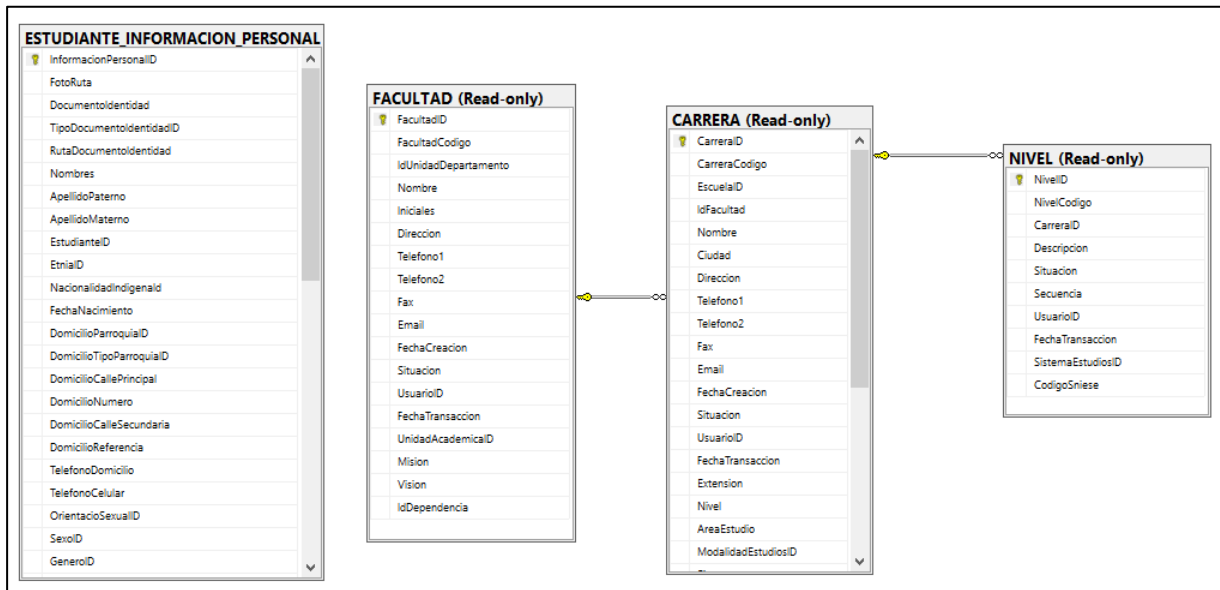


Figura 4. Diagrama de Base de Datos

Diagrama de caso de uso

Muestra las funcionalidades y usos que cada usuario tendrá sobre el sistema informático de control académico (SICOA).

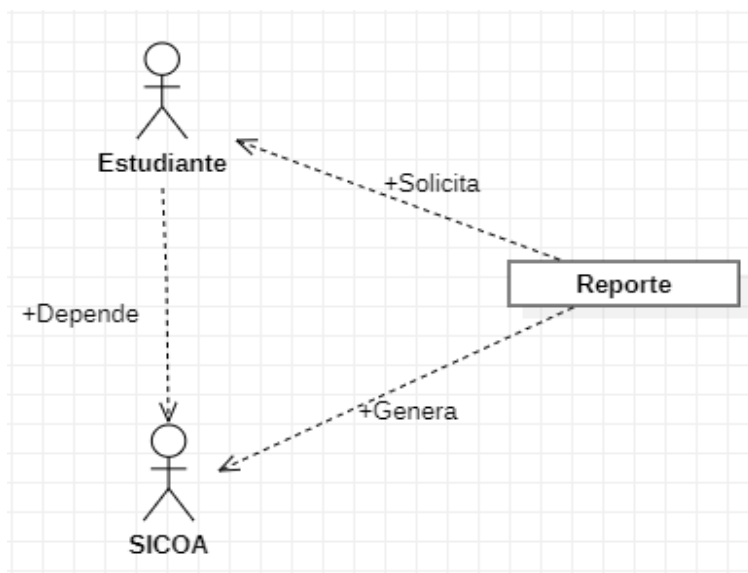


Figura 5. Caso de uso del sistema de reporte académico

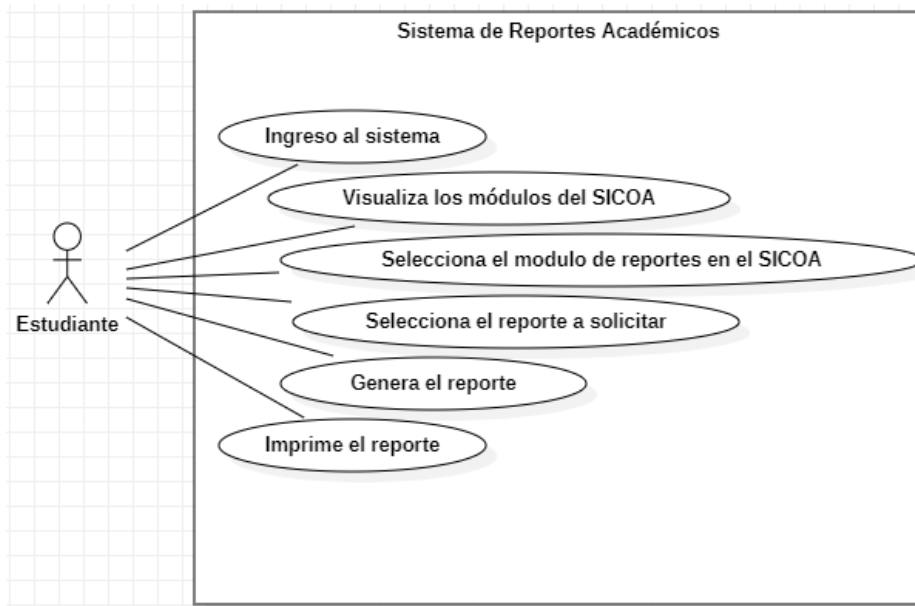


Figura 6. Caso de uso de estudiante

Implementación

Arquitectura basada en microservicio

Para aplicar la arquitectura de microservicios se diseñó el CRUD interno en el microservicio de la aplicación web, en el back end se tiene la puerta de enlace, seguido por los microservicios ya implementados como se puede observar en la figura 7.

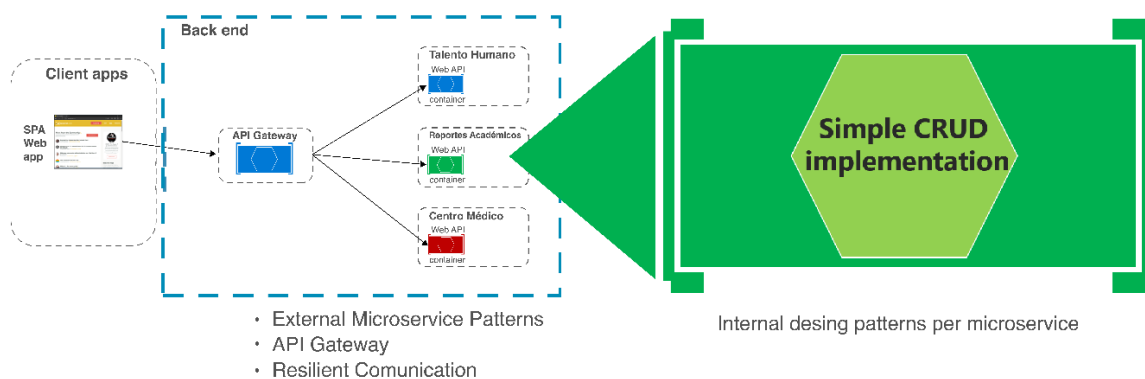


Figura 7. Diseño interno de un microservicio CRUD.

Fuente: (De La Torre et al., 2022)

Para el desarrollo del microservicio se requirió ASP.NET Core 5 y una API de acceso a datos, como Entity Framework Core. Así también, para generar automáticamente metadatos y definir las API REST, se hizo uso de Swagger a fin de proporcionar una descripción de lo que ofrece el servicio. En la figura 8, se muestra el microservicio lógico “Reportes”, que incluye su base de datos reportes.

Data-Driven/CRUD microservice container

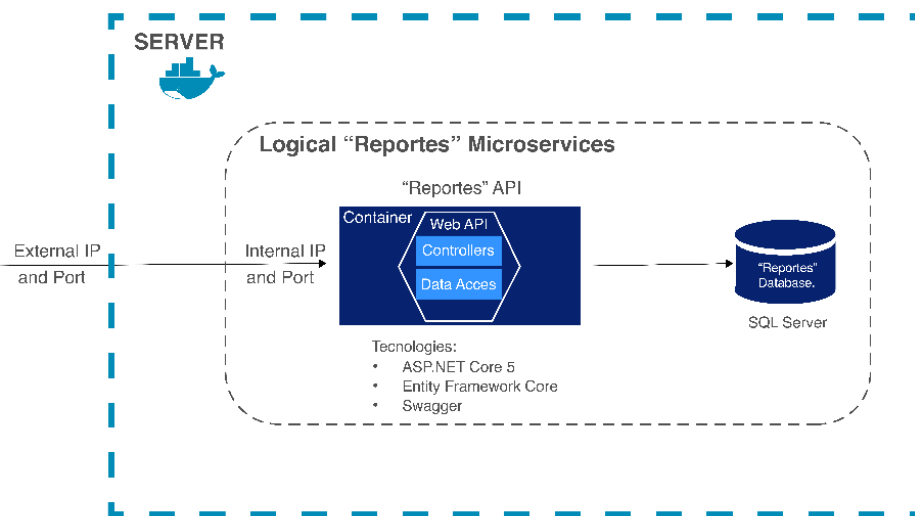


Figura 8. Diseño de un microservicio CRUD.

Fuente: (De La Torre et al., 2022)

Implementación de la arquitectura basada en microservicio

Para la implementación de un microservicio se creó un proyecto web de ASP.NET Core Web API en Visual Studio, después, se seleccionó el tipo de extensión .NET 5.0, como se evidencia en la figura 9 para la creación del proyecto.

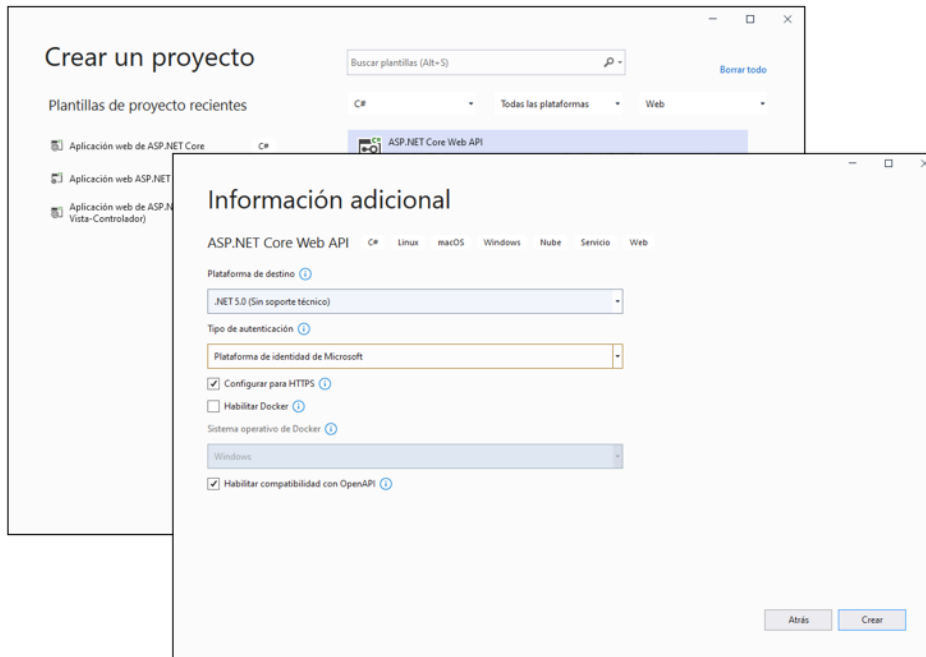


Figura 9. Creación de un proyecto de API web de ASP.NET Core en Visual Studio 2019.

Después de crear el proyecto, se implementó los controladores utilizando el patrón de arquitectura MVC, mediante la API de entity framework. Internamente, dentro de la dependencia de *Microsoft.AspNetCore.All*, hace referencia a Entity Framework y a muchos otros paquetes NuGet de .NET, como se muestra en la figura 10.

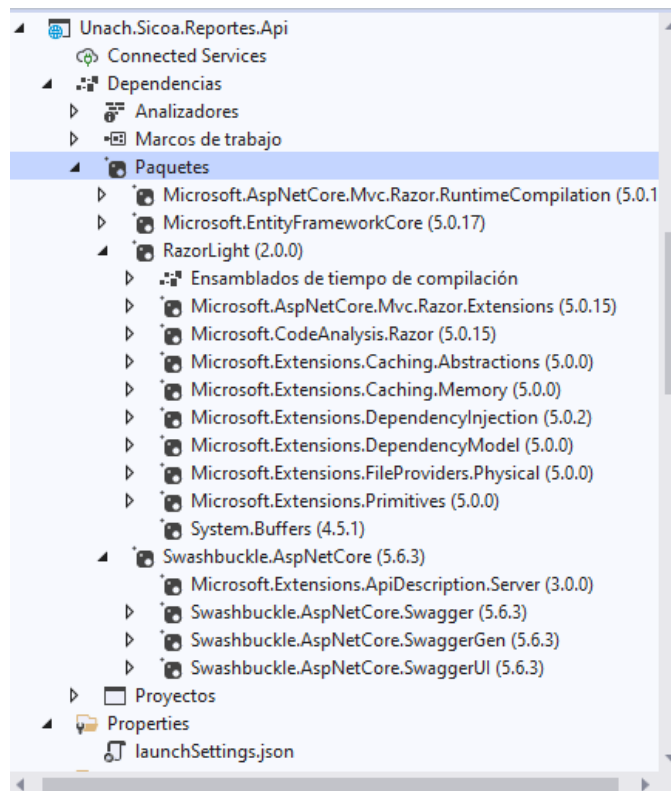
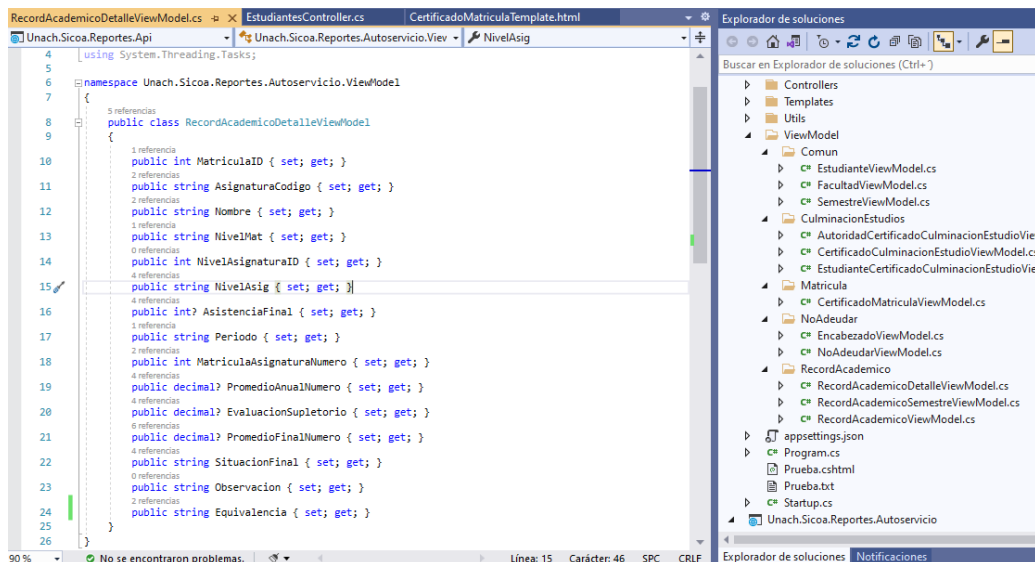


Figura 10. Dependencias en un microservicio API Web de CRUD.

Implementación de servicios API Web de CRUD con Entity Framework.

El microservicio de reportes usa Entity Framework (EF) Core y el proveedor de SQL Server de Azure. El acceso a los datos se realiza utilizando un modelo, teniendo en cuenta que un modelo se compone de entidades (modelo de dominio) y un contexto derivado (DbContext), permite consultar y guardar los datos, por medio de varias entidades como se muestra en la figura 11.

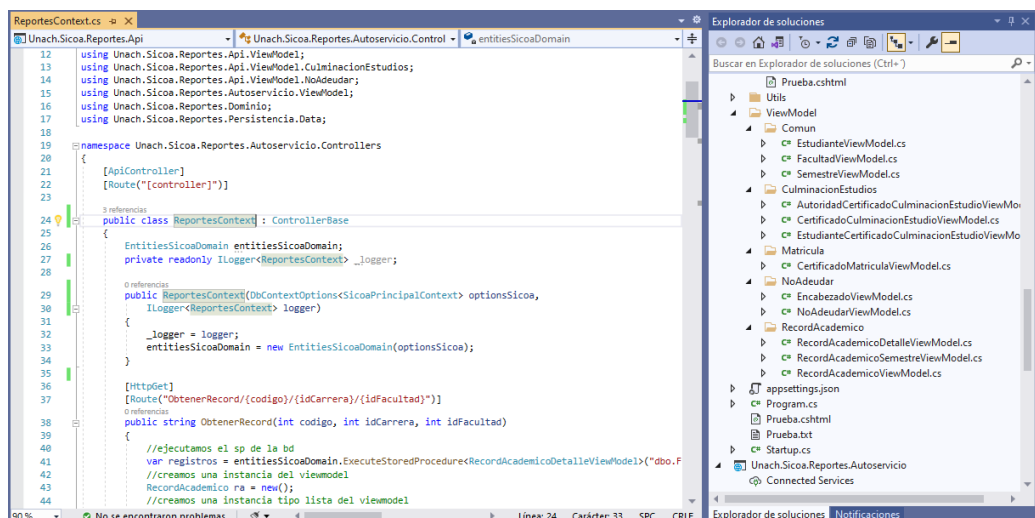


```
using System.Threading.Tasks;

namespace Unach.Sicoa.Reportes.Autoservicio.ViewModel
{
    public class RecordAcademicoDetalleViewModel
    {
        public int MatriculaID { set; get; }
        public string AsignaturaCodigo { set; get; }
        public string Nombre { set; get; }
        public string NivelMat { set; get; }
        public int NivelAsignaturaID { set; get; }
        public string NivelAsig { set; get; }
        public int? AsistenciaFinal { set; get; }
        public string Periodo { set; get; }
        public int MatriculaAsignaturaNumero { set; get; }
        public decimal? PromedioAnualNumero { set; get; }
        public decimal? EvaluacionSupletorio { set; get; }
        public decimal? PromedioFinalNumero { set; get; }
        public string SituacionFinal { set; get; }
        public string Observacion { set; get; }
        public string Equivalencia { set; get; }
    }
}
```

Figura 11. Diseño de la clase para el récord académico.

También se necesitó un ControllerBase que represente una sesión con la base de datos. Para el microservicio de reportes, la clase ReportesContext se deriva de la clase base ControllerBase, tal como se muestra en la figura 12.



```
using Unach.Sicoa.Reportes.Api.ViewModel;
using Unach.Sicoa.Reportes.Api.ViewModel.CulminacionEstudios;
using Unach.Sicoa.Reportes.Api.ViewModel.NoAdeudar;
using Unach.Sicoa.Reportes.Autoservicio.ViewModel;
using Unach.Sicoa.Reportes.Dominio;
using Unach.Sicoa.Reportes.Persistencia.Data;

namespace Unach.Sicoa.Reportes.Autoservicio.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class ReportesContext : ControllerBase
    {
        EntitiesSicoaDomain entitiesSicoaDomain;
        private readonly ILogger<ReportesContext> _logger;

        public ReportesContext(DbContextOptions<SicoaPrincipalContext> optionsSicoa,
            ILogger<ReportesContext> logger)
        {
            _logger = logger;
            entitiesSicoaDomain = new EntitiesSicoaDomain(optionsSicoa);
        }

        [HttpGet]
        [Route("ObtenerRecord/{codigo}/{idCarrera}/{idFacultad}")]
        public string ObtenerRecord(int codigo, int idCarrera, int idFacultad)
        {
            //Ejecutamos el sp de la bd
            var registros = entitiesSicoaDomain.ExecuteStoredProcedure<RecordAcademicoDetalleViewModel>("dbo.F
            //creamos una instancia del viewModel
            //creamos una instancia tipo lista del viewModel
        }
    }
}
```

Figura 12. Sesión con la base de datos.

Consulta de los datos desde controladores de API web

Para la consulta de los datos desde controladores de API, se utilizó las instancias de las clases de entidad que se recuperan de la base de datos mediante Language Integrated Query (LINQ), como se muestra en la figura 13.

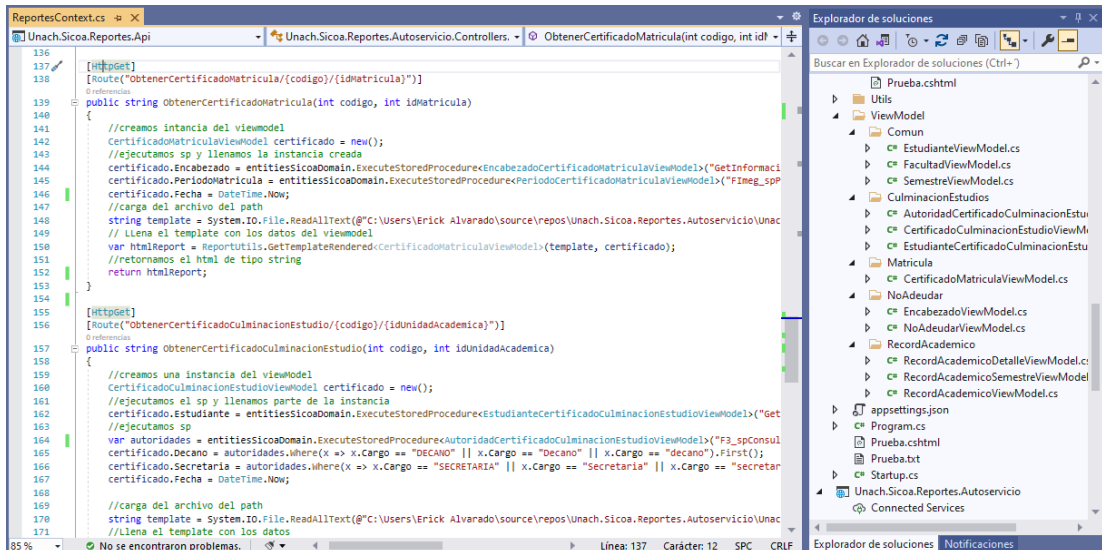


Figura 13. Uso de LINQ para recuperar los datos de la base de datos.

Finalmente, una acción que se configuró fue Swagger en el proyecto Web API, normalmente se hace en la clase de Startup como se muestra en la figura 14, haciendo un llamado al método `services.AddControllers()`, dentro del método `ConfigureServices()`.

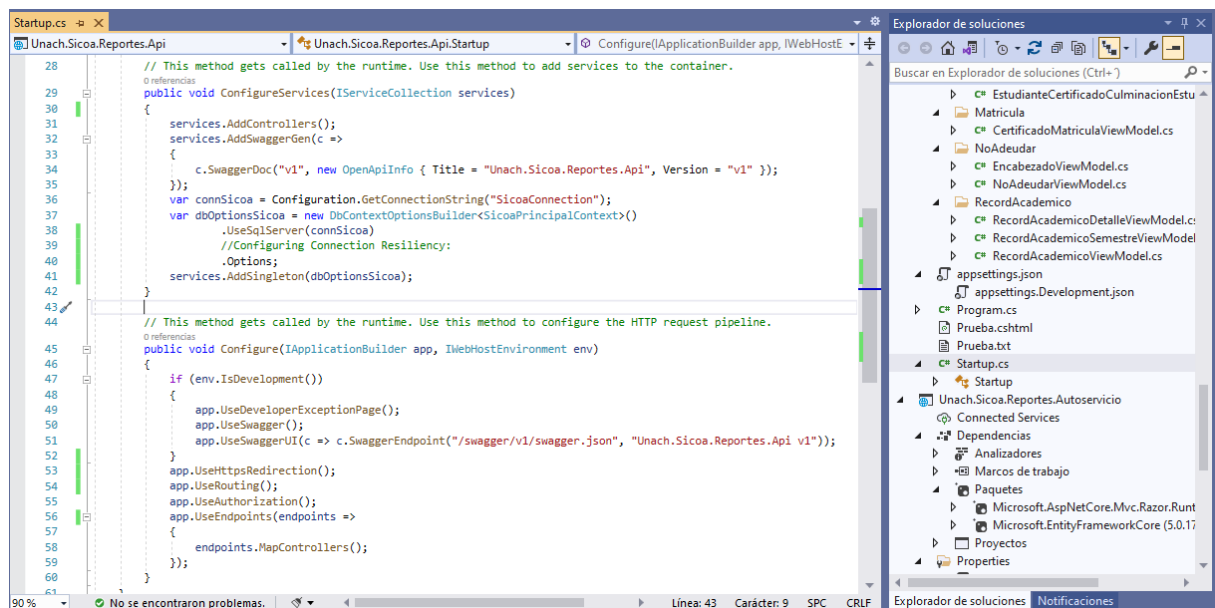


Figura 14. Configuración en el proyecto Web API.

Por otro lado, se construyó una base de datos en la herramienta Microsoft SQL Server Management Studio, para los datos de la información personal de cada estudiante, como se muestra en la figura 15.

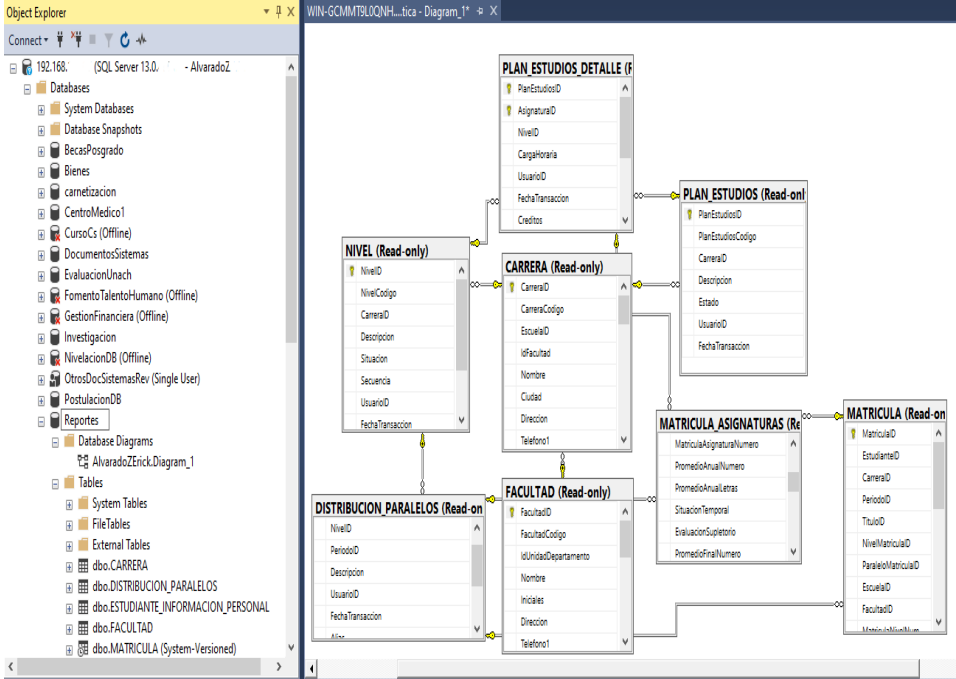


Figura 15. Creación de gestores de información

Como resultado se tiene la arquitectura externa que se puede visualizar en la figura 16, siendo la arquitectura de microservicio compuesta por varios servicios como: reportes académicos, talento humano y centro médico, siguiendo los principios descritos en la sección sobre la arquitectura basada en microservicios.

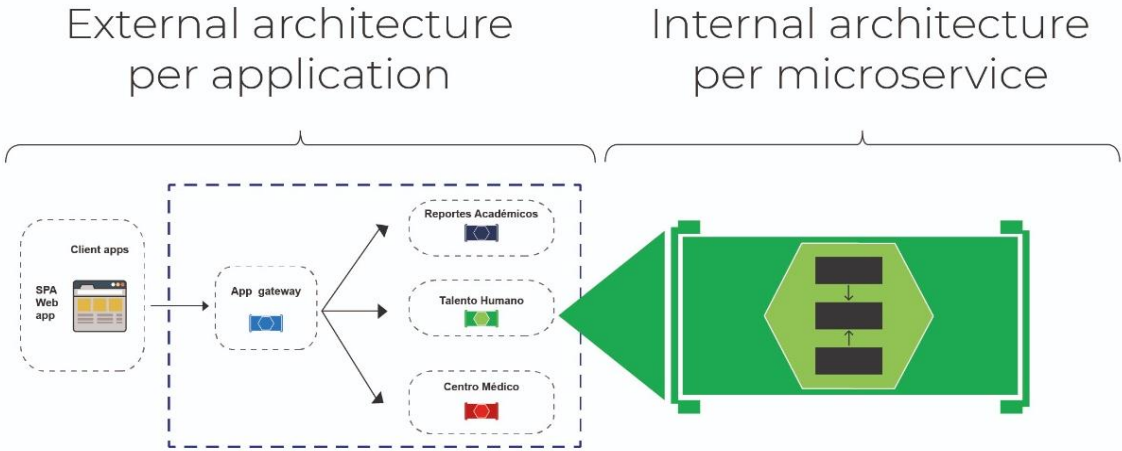


Figura 16. Estructura de la arquitectura de microservicio.

Fuente: (De La Torre et al., 2022)

Prototipo del reporte del récord académico universitario

El estudiante ingresa al SICOA donde encontrará la opción del módulo de reportes, en él se encuentran diferentes tipos de reportes como el récord académico, culminación de estudios, ser beneficiario de beca o ayuda académica y certificado de matrícula como se puede observar en la figura 10.

The screenshot shows a web browser window titled 'Sicoa' with the URL 'www.sicoaweb2.unach.edu.ec/Modulo/RecordAcademico.asp'. The page content is as follows:

UNIVERSIDAD NACIONAL DE CHIMBORAZO
FACULTAD DE @Model.FacultadEncabezado.NombreFacultad
@Model.FacultadEncabezado.Nombre
RECORD ACADÉMICO UNIVERSITARIO

DECANATO DE LA FACULTAD DE LA UNIVERSIDAD NACIONAL DE CHIMBORAZO. - @Model.FacultadEncabezado.Ciudad, el @DateTime.Now.Day de @mes del @DateTime.Now.Year .- Confírase por secretaría la solicitud.- EL DECANO.-

Revisados los documentos respectivos, consta: Estudiante: @Model.Estudiante.Nombres.- Cedula de identidad: @Model.Estudiante.DocumentoIdentidad .- Códigos: @Model.Estudiante.EstudianteID.- Modalidad: PRESENCIAL.- Sedé: @Model.FacultadEncabezado.Ciudad.- Carrera: @Model.FacultadEncabezado.Nombre

NIVEL: @recorre.NombreSemestre
 Matrícula N°: @recorre.MatriculaSemestre

COD	NOMBRE DE LA ASIGNATURA	NIVEL	%Asis	N. Matr.	Prom Sem.	Supl.	Prom Final	Equi	SIT. FINAL	OBSERVACION
@COD	@record.Nombre	@NivelAsig	@Asis	@Matr	@Prc	@Evs	@Prc	@Equ	@Situacion	@Observacion
@COD	@record.Nombre	@NivelAsig	@Asis	@Matr	@Prc	@Evs	@Prc	@Equ	@Situacion	@Observacion
@COD	@record.Nombre	@NivelAsig	@Asis	@Matr	@Prc	@Evs	@Prc	@Equ	@Situacion	@Observacion

NIVEL: @recorre.NombreSemestre
 Matrícula N°: @recorre.MatriculaSemestre

COD	NOMBRE DE LA ASIGNATURA	NIVEL	%Asis	N. Matr.	Prom Sem.	Supl.	Prom Final	Equi	SIT. FINAL	OBSERVACION
@COD	@record.Nombre	@NivelAsig	@Asis	@Matr	@Prc	@Evs	@Prc	@Equ	@Situacion	@Observacion
@COD	@record.Nombre	@NivelAsig	@Asis	@Matr	@Prc	@Evs	@Prc	@Equ	@Situacion	@Observacion
@COD	@record.Nombre	@NivelAsig	@Asis	@Matr	@Prc	@Evs	@Prc	@Equ	@Situacion	@Observacion

Figura 17. Prototipo del reporte del récord académico universitario

CAPÍTULO IV. RESULTADOS

4.1 Resultados

Al concluir el desarrollo de la aplicación web para el servicio de trámites académicos de la UNACH usando una arquitectura basada en microservicios, se realizaron pruebas de rendimiento proporcionadas por la herramienta JMeter, los valores hallados están relacionados con el modelo de calidad FURPS.

La tabla 8 condensó la información de la cantidad de procedimientos ejecutados durante la evaluación de la aplicación, en total se registraron 1250 procesos, en cantidades homogéneas de 250 casos se presentaron actividades de reconocimiento de la página de inicio de la aplicación, generación de récord académico, certificados de no adeudar a Tics, certificados de culminación de estudios y certificados de matrícula.

Tabla 8. Cantidad de procedimientos realizados

Accesibilidad	Cantidad de pruebas
Página de Inicio	250
Certificado de Récord Académico	250
Certificado No Adeudar a Tics	250
Certificado de Culminación de Estudios	250
Certificado de Matricula	250



Figura 20. Porcentaje de éxito en la evaluación inicial de la aplicación

4.2 Valoración de Indicadores

Como requerimientos solicitados a la aplicación se generaron 1000 documentos que responden a récords académicos, certificados de no adeudar a Tics, certificados de culminación de estudios y certificados de matrícula

4.2.1 Eficacia

Tabla 9. Requerimientos solicitados al aplicativo

Parámetro	Indicador	Requerimiento
Eficacia	Requerimientos exitosos del aplicativo	100%
	Requerimientos fallidos del aplicativo	0

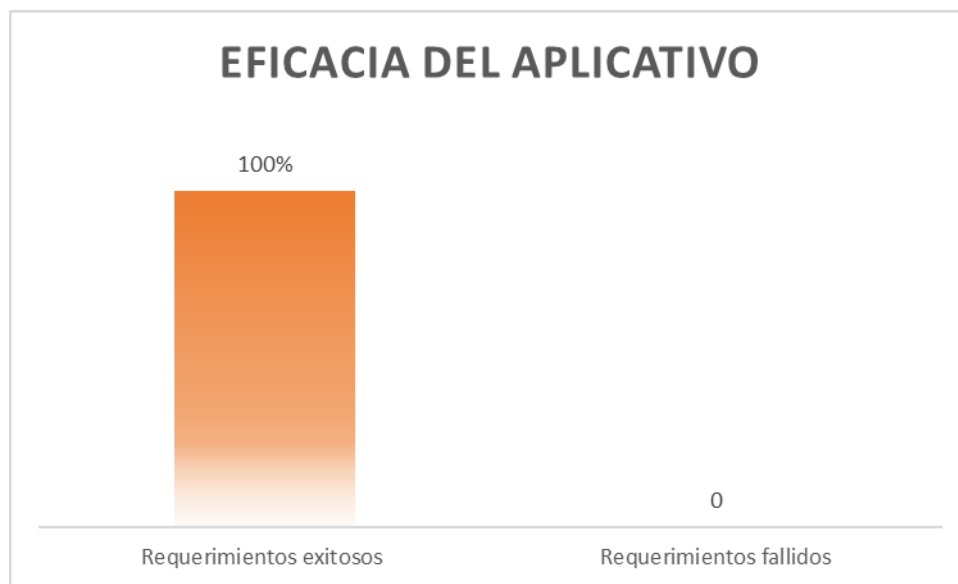


Figura 21. Eficacia del aplicativo

La Tabla 9 y Figura 21 evidenciaron el porcentaje de satisfacción del aplicativo luego de contabilizar el número de requerimientos exitosos, en este sentido la eficacia del producto fue del 100% luego de solicitar la generación de 1000 certificados de diferente categoría.

4.2.2 Tiempo de respuesta

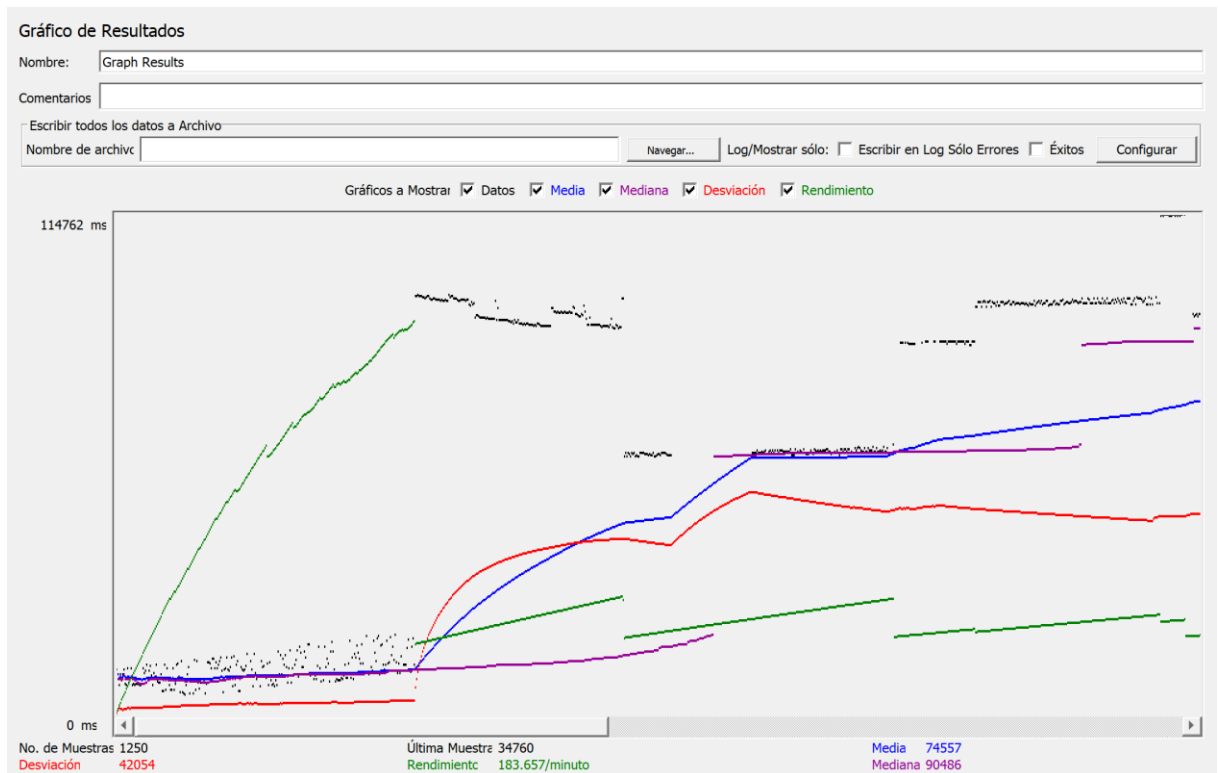


Figura 22. Tiempo de respuesta del aplicativo

El tiempo promedio de respuesta del aplicativo para la generación de los diferentes certificados académicos se encontró entre 36171 y 154945ms, el valor de la media correspondiente a 74557ms fue calculado por la herramienta JMeter basada en la fórmula

$$\bar{n} = \frac{1}{2} \sum_{i=1}^n a_i = \frac{a_1 + a_2 + a_3 + \dots + a_n}{n}$$

Ecuación 1. Fórmula del tiempo de respuesta

Fuente: Toledo, 2017

4.2.3 Utilización de recursos

Posterior al análisis de la eficacia y tiempo de respuesta del aplicativo, se verificó el tiempo de consumo de los recursos durante la generación de los documentos académicos.

Tabla 10. Uso de recursos

Parámetro	Indicador	Porcentaje de consumo
Uso de recursos	Uso de CPU	15
	Uso de la memoria RAM	75
	Uso de disco duro	5

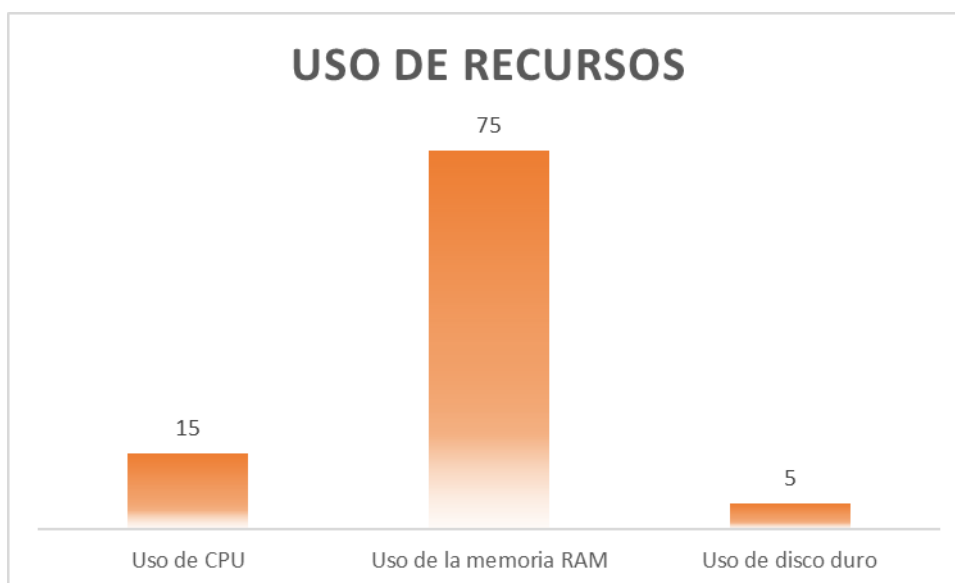


Figura 23. Uso de recursos

La Tabla 10 y Figura 23 visibilizaron que el aplicativo demandó mayoritariamente el uso de la memoria RAM con un porcentaje de 75%, seguido por el uso del 15% del CPU y apenas el 5% del espacio en el disco duro.

4.2.4 Comparación de los parámetros del estudio con los propuestos por el modelo FURPS

Tabla 11. Comparación de valores del modelo FURPS vs. valores del estudio

Parámetro	Modelo FURPS	Valores del estudio
Eficacia	95%	100%
Tiempo de respuesta	5s	74557ms
Uso de recursos	25%	15%

Una vez que los usuarios ingresan a la aplicación de manera simultánea se visibilizó que el rendimiento del estado inicial del aplicativo fue óptimo, es decir que la generación de cualquiera de los certificados antes mencionados fue exitosa, no existió ninguna falla que impidiera la generación de los documentos.

4.3 Discusión

La herramienta Jmeter fue utilizada para la medición de los indicadores de rendimiento en el aplicativo web de servicios académicos de la Universidad Nacional de Chimborazo. Los resultados hallados fueron comparados con los valores propuestos en el modelo FURPS; la eficacia del aplicativo fue del 100%, de satisfacción con relación al 95% que establece el modelo FURPS, en cuanto al tiempo de respuesta el promedio del aplicativo fue de 74557ms, cantidad significativamente inferior a los 5sg del modelo, por otro lado, el promedio de uso de recursos del aplicativo fue de 31,66% frente al 25% que establece el modelo de calidad. Los valores mencionados destacan con claridad que el aplicativo oferta calidad en rendimiento.

CONCLUSIONES

- Las constantes actualizaciones tecnológicas permiten la evolución indiscutible de mejora en la creación de sistemas web que garantizan elevados niveles de seguridad, funcionalidad, rendimiento y usabilidad. El aplicativo web desarrollado para la generación de documentos académicos usando la arquitectura de microservicios permitió construir una aplicación única como un conjunto de pequeños servicios, cada uno ejecutando su propio proceso con una dinámica ligera de comunicación e independencia de productos totalmente automatizados que minimizan la gestión centralizada de recursos y los vuelven multidisciplinarios ante cualquier sistema operativo.
- El desarrollo de la aplicación web para la generación de documentos académicos alojado el módulo de reportes de la coordinación de gestión de desarrollo de sistemas informáticos CODESI de la UNACH, se realizó bajo la metodología de desarrollo de software ágil SCRUM con una arquitectura basada en microservicios con herramientas de la plataforma .Net su framework. NetCore 5 en Visual Studio 2019 en el lenguaje de programación C#, además de Javascript, CSS para el maqo de las API y como gestor de base de datos Microsoft SQL Server Management Studio 18.
- En la evaluación del rendimiento se utilizó la herramienta JMeter y según los indicadores del Modelo FURPS se obtuvo una eficacia del 100%, un tiempo de respuesta de 74557 ms y el 25% de uso de recursos, el antecedente deja claro que el aplicativo cumple con el criterio de calidad.

RECOMENDACIONES

- Trabajar con tecnologías de la plataforma .Net para el desarrollo de sistemas web de manera que se aprovechen las librerías disponibles como el framework .NetCore, de forma que se eleven los niveles de seguridad.
- Utilizar metodologías de desarrollo de software cuyas etapas que forman parte de sus procesos garanticen la agilidad de los requerimientos y la facilidad de modificación de procesos.
- Evaluar los aplicativos basados en arquitectura de microservicios con diferentes modelos de calidad que permitan corregir errores y maximicen los niveles de calidad.

REFERENCIAS BIBLIOGRÁFICAS

- Astorga, P. P. (2022). *Arquitectura de microservicios: qué es, ventajas y desventajas*. Obtenido de <https://decidesoluciones.es/arquitectura-de-microservicios/#:~:text=La%20arquitectura%20de%20microservicios%20es%20un%20m%C3%A9todo%20de%20desarrollo%20de,una%20funcionalidad%20de%20negocio%20completa.>
- Bogotá, C. d. (2019). El mundo conectado por las API. *Cámara de Comercio de Bogotá*, 14.
- De La Torre, C., Wagner, B., & Rousos, M. (2022). *.NET Microservices: Architecture for Containerized .NET Applications*. Redmond, Washington 98052-6399: EDITION v6.0.
- DevOps. (2021). *IT Digital Media Group*. Obtenido de Los beneficios de la arquitectura de microservicios: <https://discoverthenew.ituser.es/devops/2021/06/los-beneficios-de-la-arquitectura-de-microservicios>
- Gómez. (2017). Un acercamiento a los microservicios. *Revistas UNAC*, 11.
- Lewis, J., & Fowler, M. (2014). *Microservices*. Viittattu.
- Lopez, D. (2017). Arquitectura de software basada en microservicios para el desarrollo de aplicaciones web de la Asamblea Nacional. *Universidad Técnica del Norte*, 18-20.
- Maceda, H. C. (2016). *Arquitectura de Software conceptos y ciclos de desarrollo*. Ciudad de México: Compañía de Cengage Learning.
- Maya , E., & López, D. (2018). Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web. *ResearchGate*, 13.
- Newman, S. (2015). *Creación de microservicios*. 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, Inc.
- Ovais, A. (2019). Aplicativos web y características de funcionalidad. *Universidad Técnica Particular de Loja*, 1(1), 1-75.

- Richards, M. (2015). *Software Architecture Patterns*. Gravenstein Highway North, Sebastopol, CA 95472.: Copyright © 2015 O'Reilly Media, Inc.
- Rodríguez, Á. P. (2019). Arquitectura basada en micro-servicios para aplicaciones web. *Revista Tecnología, Investigación y Academia*, 12-14.
- Rosado, A., & Jaimés, J. (2017). REVISIÓN DE LA INCORPORACIÓN DE LA ARQUITECTURA ORIENTADA A SERVICIOS EN LAS ORGANIZACIONES. 12.
- UTA. (2022). *Universidad Técnica de Ambato*. Obtenido de Servicios en línea: https://deadv.uta.edu.ec/#servicios_en_linea
- Valarezo, M. R., Honores, J. A., Gómez, A. S., & Vincés, L. F. (2018). COMPARACIÓN DE TENDENCIAS TECNOLÓGICAS EN APLICACIONES WEB. *3C Tecnología. Glosas de Innovación aplicadas a la pyme.*, 22.

ANEXOS

8.1 Anexo 1: Código HTML

Al ejecutar el código se mostrará una interfaz como la figura 3 representando el récord académico.

```
@model RecordAcademico

<html>

<head>

  <title>HTML Editor - Full Version</title>

</head>

<body>

  <h1 style="text-align: center;"><span style="font-size:20px;">UNIVERSIDAD NACIONAL DE
CHIMBORAZO</span></h1>

  <p style="text-align: center;"><strong><span style="font-
size:16px;">@Model.FacultadEncabezado.NombreFacultad</span></strong></p>

  <p style="text-align: center;"><strong><span style="font-
size:16px;">@Model.FacultadEncabezado.Nombre</span></strong></p>

  <table align="center" border="2" cellpadding="1" cellspacing="1" style="width:500px;">

    <tbody>

      <tr>

        <td style="text-align: center;">RECORD ACAD&Eacute;MICO UNIVERSITARIO</td>

      </tr>

    </tbody>

  </table>

  <p>&nbsp;</p>

  <p><span style="font-size:11px;">DECANATO DE LA @Model.FacultadEncabezado.NombreFacultad DE LA
UNIVERSIDAD NACIONAL DE CHIMBORAZO.- @Model.FacultadEncabezado.Ciudad, @DateTime.Now.Day de @
string mes = DateTime.Now.ToString("MMMM"); } @mes del @DateTime.Now.Year .- Confi&eacute;rase por
secretar&iacute;a lo solicitado.- EL DECANO.-</span></p>
```



```

        <td style="text-align: center; width: 50px"><strong><span style="font-size:11px;">Prom.
Final.</span></strong></td>
        <td style="text-align: center; width: 50px"><strong><span style="font-
size:11px;">Equiv.&nbsp;</span></strong></td>
        <td style="text-align: center; width: 90px"><strong><span style="font-size:11px;">SIT.
FINAL.</span></strong></td>
        <td style="text-align: center; width: 250px"><strong><span style="font-
size:11px;">OBSERVACI&Oacute;N</span></strong></td>
    </tr>
</thead>
<tbody>
    @foreach (var record in recorre.DetalleRecord)
    {
        <tr>
            <td style="text-align: left;"><span style="font-size: 11px;"> @record.AsignaturaCodigo</span></td>
            <td style="text-align: left;"><span style="font-size: 11px;"> @record.Nombre </span></td>
            <td style="text-align: center;"><span style="font-size:11px;"> @record.NivelAsig</span></td>
            <td style="text-align: center;"><span style="font-size:11px;"> @if (record.AsistenciaFinal !=
null)@record.AsistenciaFinal</span></td>
            <td style="text-align: center;"><span style="font-size:11px;"> @record.MatriculaAsignaturaNumero
</span></td>
            <td style="text-align: center;"><span style="font-size:11px;"> @if (record.PromedioAnualNumero !=
null)@record.PromedioAnualNumero </span></td>
            <td style="text-align: center;">
                <span style="font-size:11px;">
                    @if (record.EvaluacionSupletorio != null)
                    @record.EvaluacionSupletorio
                </span>
            </td>
            <td style="text-align: center;">
                <span style="font-size:11px;">
                    @if (record.PromedioFinalNumero != null)
                    @record.PromedioFinalNumero
                </span>
            </td>
            <td style="text-align: center;"><span style="font-size:11px;">@record.Equivalencia</span></td>
            <td style="text-align: center;"><span style="font-size:11px;">@record.SituacionFinal</span></td>
            <td style="text-align: center;">&nbsp;</td>

```

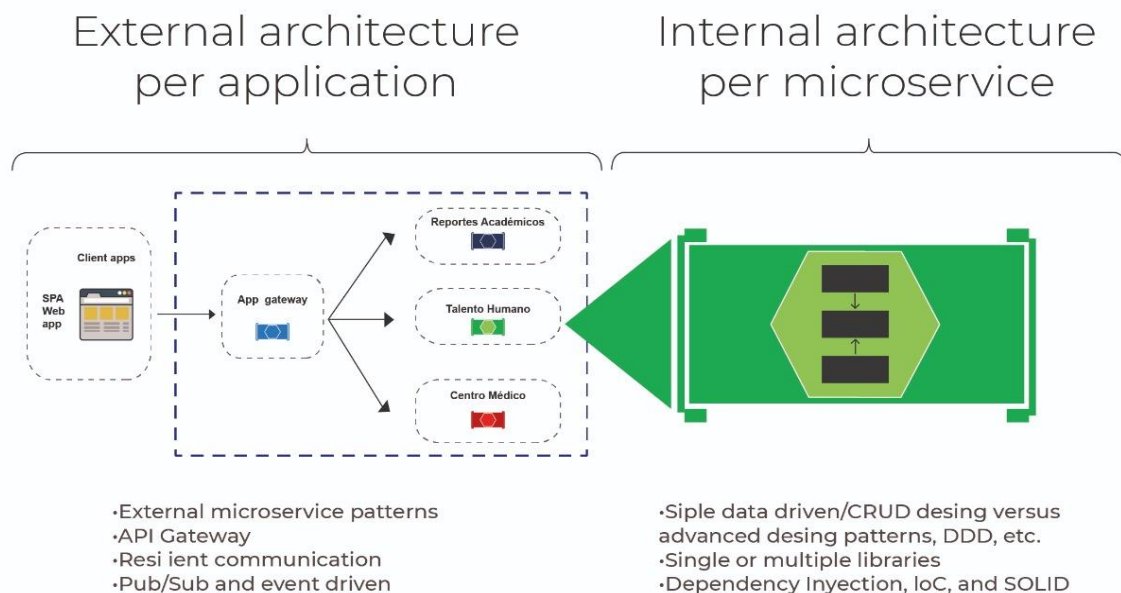
```

        </tr>
    </tbody>
    <tfoot>
        <tr>
            <td style="text-align: left;" colspan="11"> Promedio de asignaturas aprobadas:
                @recorre.PromedioSemestre.ToString("0.##") </td>
        </tr>
    </tfoot>
</table>
<br />
}
<h6>Promedio general de asignaturas aprobadas: @Model.PromedioGeneral </h6>

<p style="text-align: right;">@Model.FacultadEncabezado.Ciudad, @DateTime.Now.Day de @string mes2 =
DateTime.Now.ToString("MMMM"); } @mes2 del @DateTime.Now.Year </p>
</body>
</html>

```

8.2 Anexo 2: Arquitectura de microservicios



8.3 Anexo 3: Manual de Usuario



DIRECCIÓN DE TECNOLOGÍAS DE LA
INFORMACIÓN Y COMUNICACIÓN
COORDINACIÓN DE GESTIÓN DE
DESARROLLO DE SISTEMAS INFORMÁTICOS

ESPECIFICACIONES DE REQUISITOS DEL SOFTWARE

Reportes Académicos

Versión 1.0



Control del Documento

TÍTULO: MANUAL DE USUARIO MÓDULO DE REPORTES ACADÉMICOS

VERSIÓN: 1

CÓDIGO DEL FORMATO:

DEPENDENCIA: COORDINACIÓN DE GESTIÓN DE DESARROLLO DE SISTEMAS INFORMÁTICOS

Firmas y Aprobaciones

ELABORADO POR: Erick Alexis Alvarado Zambrano
Egresado Investigador

FECHA: 11/10/2022 Firma: _____

REVISADO POR: Ing. Pamela Buñay
Tutor

FECHA: 15/11/2022 Firma: _____

Lista de Cambios

VERSIÓN	FECHA	AUTOR	DESCRIPCIÓN
1.0	15-11-2022	Erick Alvarado	Emisión Inicial



ESPECIFICACIONES DE REQUISITOS DE SOFTWARE

Sistema de Información:	Sistema Informático de control académico (SICOA).
Módulo /Funcionalidad:	Reportes académicos
Unidad Requirente:	Estudiantes
Titular de la unidad:	Ing. Henry Paca

Nombre de requisito: Certificado de récord académico						
No	Prioridad	Como	Necesito	Para	Criterios de aceptación	Estimación:
1	Alta	Alumno	Generar un reporte del récord académico en el sistema SICOA.	Para conocer las calificaciones finales obtenidas de cada asignatura en el semestre aprobada durante el proceso académico del estudiante.	Mostrará el detalle de la fecha y hora de la ejecución de dicho proceso, para posteriormente imprimirlo.	<20 días>
2	Alta	Desarrollador	Establecer procedimientos en la base de datos.	Para obtener la información de cada estudiante como: nombres, ciclo académico, C.I., facultad, carrera, nivel, asignatura y situación final de cada semestre. Así también como el nombre del decano y la secretaria de la facultad.	Asegurar el buen funcionamiento de la base de datos.	<8 horas>
3	Alta	Desarrollador	Conexión de la base de datos hacia la aplicación web.	Para extraer la información del estudiante hacia la aplicación web.	Encriptación de datos.	<8 horas>
4	Alta	Desarrollador	Interfaz de la aplicación web.	Para visualizar la interfaz de la aplicación web.	Creación de la plantilla para el encabezado con el formato de tabla establecida para el certificado.	<8 horas>
5	Alta	Desarrollador	Interfaz de programación de aplicaciones (API).	Consumir de forma genérica cualquier consulta en la interfaz del usuario.	El estudiante mediante el sistema SICOA, con su código de estudiante y código de carrera genera el reporte.	<8 horas>
6	Alta	Desarrollador	Generar reporte en el sistema SICOA.	Crear un documento final en el que se visualice el récord académico universitario.	El documento se descargará en formato PDF, en el que se visualice el código QR único, con el detalle de la fecha y hora de la ejecución de dicho proceso, con el nombre de cada decano y secretaria de la facultad.	<8 horas>



ANEXOS

UNIVERSIDAD NACIONAL DE CHIMBORAZO
FACULTAD DE INGENIERÍA
INGENIERÍA EN SISTEMAS Y COMPUTACIÓN

RECORD ACADÉMICO UNIVERSITARIO

DECANATO DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD NACIONAL DE CHIMBORAZO.- Riobamba, 29 de abril de 2022.- Confiérase por secretaría lo solicitado.- EL DECANO.-

[Firma]
Ing. Patricio Villacres, Dr.

Revisados los documentos respectivos, consta: Estudiante: ALVARADO ZAMBRANO ERICK ALEXIS.- Cédula de identidad: 2350531055.- Código: 41610.- Modalidad: PRESENCIAL.- Sede: RIOBAMBA.- Carrera: INGENIERÍA EN SISTEMAS Y COMPUTACIÓN .

NIVEL: INGENIERÍA EN SISTEMAS Y COMPUTACIÓN - PRIMER SEMESTRE

Matrícula N°.: 218466.- Ciclo Académico: OCTUBRE 2017 - MARZO 2018										
CÓD.	NOMBRE DE LA ASIGNATURA	NIVEL	% Asist.	Nro. Matr.	Prom. Sem.	Supl.	Prom. Final	Equiv.	SIT. FINAL	OBSERVACION
SIC101	FISICA I	PRIMER SEMESTRE		1	7.00		7.00	-	APROBADO	
SIC102	MATEMATICA I	PRIMER SEMESTRE		1	8.00		8.00	-	APROBADO	
SIC103	LOGICA DE PROGRAMACION	PRIMER SEMESTRE		1	7.00		7.00	-	APROBADO	
SIC104	QUIMICA	PRIMER SEMESTRE		1	7.00		7.00	-	APROBADO	
SIC105	EDUCACION FISICA I	PRIMER SEMESTRE		1	9.00		9.00	-	APROBADO	
SIC106	LENGUAJE COMUNICACION Y DESARROLLO DEL PENSAMIENTO	PRIMER SEMESTRE		1	8.00		8.00	-	APROBADO	
Promedio de asignaturas aprobadas: 7,67										

NIVEL: INGENIERÍA EN SISTEMAS Y COMPUTACIÓN - SEGUNDO SEMESTRE

Matrícula N°.: 240920.- Ciclo Académico: MARZO 2018 - AGOSTO 2018										
CÓD.	NOMBRE DE LA ASIGNATURA	NIVEL	% Asist.	Nro. Matr.	Prom. Sem.	Supl.	Prom. Final	Equiv.	SIT. FINAL	OBSERVACION
SIC201	FISICA II	SEGUNDO SEMESTRE		1	7.00		7.00	-	APROBADO	
SIC202	CALCULO DIFERENCIAL E INTEGRAL	SEGUNDO SEMESTRE		1	8.00		8.00	-	APROBADO	
SIC203	ALGEBRA LINEAL	SEGUNDO SEMESTRE		1	9.00		9.00	-	APROBADO	
SIC204	CIRCUITOS Y SIMULACION ELECTRONICA	SEGUNDO SEMESTRE		1	7.00		7.00	-	APROBADO	
SIC205	METODOS DE INVESTIGACION Y TECNICAS DE ESTUDIO	SEGUNDO SEMESTRE		1	7.00		7.00	-	APROBADO	
SIC206	PROGRAMACION I	SEGUNDO SEMESTRE		1	7.00		7.00	-	APROBADO	
SIC208	EDUCACION FISICA II	SEGUNDO SEMESTRE		1	10.00		10.00	-	APROBADO	
Promedio de asignaturas aprobadas: 7,86										



Nombre de requisito: Certificado de culminación de estudios						
No	Prioridad	Como	Necesito	Para	Criterios de aceptación	Estimación:
1	Alta	Alumno	Generar un reporte de culminación de estudios.	Para comprobar la constancia de la aprobación de todos los semestres que comprende el plan de estudios de la carrera.	Mostrará el detalle de la fecha y hora de la ejecución de dicho proceso, para posteriormente imprimirlo.	<20 días>
2	Alta	Desarrollador	Establecer procedimientos en la base de datos.	Para obtener la información de cada estudiante como: nombre completo, facultad. Así también como el nombre del decano y la secretaria de la facultad.	Asegurar el buen funcionamiento de la base de datos.	<8 horas>
3	Alta	Desarrollador	Conexión de la base de datos hacia la aplicación web.	Para extraer la información del estudiante hacia la aplicación web.	Encriptación de datos.	<8 horas>
4	Alta	Desarrollador	Interfaz de la aplicación web.	Para visualizar la interfaz de la aplicación web.	Creación de la plantilla para el encabezado con el formato establecido para el certificado.	<8 horas>
5	Alta	Desarrollador	Interfaz de programación de aplicaciones (API).	Consumir de forma genérica cualquier consulta en la interfaz del usuario.	El estudiante mediante el sistema SICOA, con su código de estudiante y código de unidad académica genera el reporte.	<8 horas>
6	Alta	Desarrollador	Generar reporte en el sistema SICOA.	Crear un documento final en el que se visualice el certificado de culminación de estudios.	El documento se descargará en formato PDF, se visualizará el código QR único, con el detalle de la fecha y hora de la ejecución de dicho proceso, con el nombre de cada decano y secretaria de la facultad.	<8 horas>



Coordinación de Gestión de
Desarrollo de Sistemas Informáticos
en movimiento



ANEXOS



UNIVERSIDAD NACIONAL DE CHIMBORAZO
FACULTAD DE CIENCIAS DE LA SALUD

Confieren el presente

**CERTIFICADO DE CULMINACIÓN
DE ESTUDIOS DE GRADO**

Al(a) señor(ita) **JOSELYN NICOLE GRANIZO CANDO**

Al haber culminado sus estudios de grado y cumplido con los requisitos exigidos para el proceso de Graduación y Titulación, tales como: Suficiencia en el Idioma, Prácticas Preprofesionales y Vinculación con la Sociedad, previo a la obtención del Título de **MEDICO GENERAL**.

Riobamba, 26 de octubre de 2021

Dr. Gonzalo Bonilla P.
DECANO

MARÍA FERNANDA CEPEDA
SECRETARIA

Elab: MCEPEDA



Nombre de requisito: Certificado de matricula						
No	Prioridad	Como	Necesito	Para	Criterios de aceptación	Estimación:
1	Alta	Alumno	Generar un reporte de la matricula académica en el sistema SICOA.	Para conocer en detalle cuando el alumno se encuentre cursando el ciclo académico.	Mostrará el detalle de la fecha y hora de la ejecución de dicho proceso, para posteriormente imprimirlo.	<20 días>
2	Alta	Desarrollador	Establecer procedimientos en la base de datos.	Para obtener la información de cada estudiante como: nombres, C.I., código de estudiante, ciclo académico, facultad, carrera, nivel y matrícula de cualquier semestre. Así también como el nombre de la secretaria de la carrera.	Asegurar el buen funcionamiento de la base de datos.	<8 horas>
3	Alta	Desarrollador	Conexión de la base de datos hacia la aplicación web.	Para extraer la información del estudiante hacia la aplicación web.	Encriptación de datos.	<8 horas>
4	Alta	Desarrollador	Interfaz de la aplicación web.	Para visualizar la interfaz de la aplicación web.	Creación de la plantilla para el encabezado con el formato establecido para el certificado.	<8 horas>
5	Alta	Desarrollador	Interfaz de programación de aplicaciones (API).	Consumir de forma genérica cualquier consulta en la interfaz del usuario.	El estudiante mediante el sistema SICOA, con el código de estudiante y código de matrícula o a través de un menú desplegable generar el reporte.	<8 horas>
6	Alta	Desarrollador	Generar reporte en el sistema SICOA.	Crear un documento final en el que se visualice el récord académico universitario.	El documento se descargará en formato PDF, en el que se visualice el código QR único, con el detalle de la fecha y hora de la ejecución de dicho proceso, con el nombre de la secretaria de la facultad.	<8 horas>



ANEXOS

UNIVERSIDAD NACIONAL DE CHIMBORAZO

FACULTAD DE CIENCIAS DE LA SALUD

ESCUELA DE MEDICINA (R)

CERTIFICADO DE MATRICULA

Certifico que: **GUEVARA VALDIVIEZO JORGE FERNANDO** con
cédula de identidad: 0604741090, código: 44458, se encuentra
legalmente matriculado(a) en la carrera de: **MEDICINA (R)** en el
nivel: **SEXTO SEMESTRE** del Ciclo Académico: **Periodo 2021-2S**,
Fecha de Inicio: **15/11/2021**, Fecha de Finalización: **25/03/2022** con
matrícula No.: **458323** como consta en los registros de ésta
secretaría.

Riobamba, 03 de agosto de 2022

Mgs. Ligia Viteri N.
SECRETARIA

Elab: MCEPEDA



Nombre de requisito: Certificado no adeudar al departamento de Tics						
No	Prioridad	Como	Necesito	Para	Criterios de aceptación	Estimación:
1	Alta	Alumno	Generar un reporte de no adeudar al departamento de Tics en el sistema SICOA.	Para conocer que el estudiante no ha realizado daños en el departamento de tecnologías de la institución.	Mostrará el detalle de la fecha y hora de la ejecución de dicho proceso, para posteriormente imprimirlo.	<20 días>
2	Alta	Desarrollador	Establecer procedimientos en la base de datos.	Para obtener la información de cada estudiante como: nombres, C.I., facultad y carrera. Así también como el nombre del director de tecnologías de la información y comunicación.	Asegurar el buen funcionamiento de la base de datos.	<8 horas>
3	Alta	Desarrollador	Conexión de la base de datos hacia la aplicación web.	Para extraer la información del estudiante hacia la aplicación web.	Encriptación de datos.	<8 horas>
4	Alta	Desarrollador	Interfaz de la aplicación web.	Para visualizar la interfaz de la aplicación web.	Creación de la plantilla para el encabezado con el formato de establecido para el certificado.	<8 horas>
5	Alta	Desarrollador	Interfaz de programación de aplicaciones (API).	Consumir de forma genérica cualquier consulta en la interfaz del usuario.	El estudiante mediante el sistema SICOA, con el código de estudiante y código de matrícula o a través de un menú desplegable generar el reporte.	<8 horas>
6	Alta	Desarrollador	Generar reporte en el sistema SICOA.	Crear un documento final en el que se visualice el récord académico universitario.	El documento se descargará en formato PDF, en el que se visualice el código QR único, con el detalle de la fecha y hora de la ejecución de dicho proceso, con el nombre del director de tecnologías de la información y comunicación.	<8 horas>



ANEXOS

Coordinación de Gestión de
Desarrollo de Sistemas Informáticos
en movimiento



**CERTIFICADO PARA
TRAMITE DE FIN DE CARRERA**

El (la) señor (ita): Millán Ramos Daniel Hernán
Cédula de Identidad No. 0603568585
Alumno (a) de la Facultad de: Ciencias de la Educación, Humanas y Tecnologías
Carrera: Pedagogía de los Idiomas Nacionales y Extranjeros

No adeuda materiales, componentes o reparación en los equipos de la Dirección de Tecnologías de la Información y Comunicación, salas y laboratorios de internet.

Riobamba, 28 de septiembre, 2022

Mgs. José Javier Haro Mendoza
DIRECTOR DE TECNOLOGÍAS DE LA INFORMACIÓN y COMUNICACIÓN

Generado: Riobamba 29/09/2022 16:19:38/