



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

**Módulo de matrícula para el sistema de titulación de la  
Universidad Nacional de Chimborazo utilizando Linq y  
Expresiones Lambda**

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

**Autores:**

Ligia Adriana Cepeda Pataron  
Celio Gustavo Paguay Quiroz

**Tutor:**

Ing. Ana Elizabeth Congacha Aushay. MsC

**Riobamba, Ecuador. 2024**

## DECLARATORIA DE AUTORÍA

Nosotros, Ligia Adriana Cepeda Pataron con cédula de ciudadanía 0650530975 y Celio Gustavo Paguay Quiroz, con cédula de ciudadanía 1600908006, autores del trabajo de investigación titulado: Módulo de matrícula para el sistema de titulación de la Universidad Nacional de Chimborazo utilizando Linq y expresiones Lambda, certifico que la producción, ideas, opiniones, criterios, contenidos y conclusiones expuestas son de mí exclusiva responsabilidad.

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

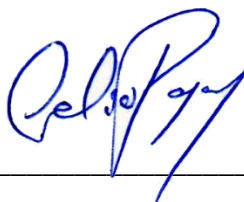
En Riobamba, 12 de julio de 2024.



---

Ligia Adriana Cepeda Pataron

C.I: 0650530975



---

Celio Gustavo Paguay Quiroz

C.I: 1600908006

# DICTAMEN FAVORABLE DEL PROFESOR TUTOR



Dirección  
Académica  
VICERRECTORADO ACADÉMICO

*en movimiento*



SISTEMA DE GESTIÓN DE LA CALIDAD  
UNACH-RGF-01-04-08.11  
VERSIÓN 01 : 06-09-2021

## ACTA FAVORABLE - INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN

En la Ciudad de Riobamba, a los 17 días del mes de julio de 2024, luego de haber revisado el Informe Final del Trabajo de Investigación presentado por los estudiantes: **LIGIA ADRIANA CEPEDA PATARON** con CC: **0650530975** y **CELIO GUSTAVO PAGUAY QUIROZ** con CC: **1600908006**, de la carrera **TECNOLOGÍAS DE LA INFORMACIÓN** y dando cumplimiento a los criterios metodológicos exigidos, se emite el **ACTA FAVORABLE DEL INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN** titulado "MÓDULO DE MATRÍCULA PARA EL SISTEMA DE TITULACIÓN DE LA UNIVERSIDAD NACIONAL DE CHIMBORAZO UTILIZANDO LINQ Y EXPRESIONES LAMBDA", por lo tanto se autoriza la presentación del mismo para los trámites pertinentes.



Ing. Ana Elizabeth Congacha  
**TUTORA**

# CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

## CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

Quienes suscribimos, catedráticos designados Miembros del Tribunal de Grado para la evaluación del trabajo de investigación Módulo de matrícula para el sistema de titulación de la Universidad Nacional de Chimborazo utilizando Linq y expresiones Lambda, presentado por Ligia Adriana Cepeda Pataron con cédula de ciudadanía 0650530975 y Celio Gustavo Paguay Quiroz, con cédula de ciudadanía 1600908006, autores del trabajo de investigación titulado: Módulo de matrícula para el sistema de titulación de la Universidad Nacional de Chimborazo utilizando Linq y expresiones Lambda, bajo la tutoría de Mgs. Ana Elizabeth Congacha Aushay; 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 a los 8 días del mes de agosto de 2024.

Danny Velasco, Mgs.  
**PRESIDENTE DEL TRIBUNAL DE GRADO**



Diego Reina, Mgs.  
**MIEMBRO DEL TRIBUNAL DE GRADO**



Jorge Delgado, Mgs.  
**MIEMBRO DEL TRIBUNAL DE GRADO**



# CERTIFICADO ANTIPLAGIO



Dirección  
Académica  
VICERRECTORADO ACADÉMICO



UNACH-RGF-01-04-08.1.7  
VERSIÓN 01 : 06-09-2021

## CERTIFICACIÓN

Que, **LIGIA ADRIANA CEPEDA PATARON** con CC: **0650530975** y **CELIO GUSTAVO PAGUAY QUIROZ** con CC: **1600908006**, estudiantes de la Carrera de **INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN**, Facultad de **INGENIERÍA**; ha trabajado bajo mi tutoría el trabajo de investigación titulado **"MÓDULO DE MATRÍCULA PARA EL SISTEMA DE TITULACIÓN DE LA UNIVERSIDAD NACIONAL DE CHIMBORAZO UTILIZANDO LINQ Y EXPRESIONES LAMBDA"**, cumple con el 1%, de acuerdo al reporte del sistema Anti plagio **TURNITIN**, porcentaje aceptado de acuerdo a la reglamentación institucional, por consiguiente autorizo continuar con el proceso.

Riobamba, 29 de julio de 2024



Ing. Ana Elizabeth Congacha Aushay  
CONGRACHA AVSHAY

Ing. Ana Elizabeth Congacha Aushay  
**TUTORA PROYECTO DE INVESTIGACIÓN**

## **DEDICATORIA**

A nuestras queridas familias Cepeda Pataron y Paguay Quiroz,

Con inmenso agradecimiento y profundo cariño, les dedicamos esta tesis. Cada logro alcanzado y cada obstáculo superado en este camino académico ha sido posible gracias a su amor incondicional y apoyo constante. Ustedes han sido nuestra fortaleza en los momentos difíciles y nuestra motivación en los momentos de triunfo. Sus palabras de aliento y su fe inquebrantable en nosotros han sido el cimiento de este logro.

Esta tesis no solo es el fruto de nuestro esfuerzo, sino también un testimonio de su dedicación y sacrificio. Con todo nuestro amor, les dedicamos este trabajo como una muestra de gratitud por su incondicional respaldo.

Con amor y gratitud eternos,

Ligia Adriana Cepeda Pataron y Celio Gustavo Paguay Quiroz

## **AGRADECIMIENTO**

En el trayecto de realizar esta tesis, hemos contado con el apoyo esencial de diversas personas que han sido claves en nuestra experiencia académica. Hoy queremos expresar nuestro más sincero agradecimiento a quienes han contribuido a la realización de este trabajo.

Primero, agradecemos profundamente a nuestra tutora de tesis, Ing. Ana Elizabeth Congacha Aushay. Su orientación, conocimiento y dedicación fueron indispensables para la culminación de este proyecto. Sus valiosos consejos y su compromiso nos guiaron a lo largo de este proceso. Estamos muy agradecidos por su paciencia y apoyo constante.

Queremos también extender nuestro agradecimiento al Ing. Hernán Darío Centeno Aulla por su desinteresado apoyo a lo largo de nuestra carrera. Su ayuda y generosidad han sido invaluable, y su constante disposición para asistirnos es algo que valoramos profundamente. Siempre estaremos en deuda por su generosidad y amistad.

A nuestros amigos, gracias por ser compañeros incondicionales durante este recorrido. Sus palabras de ánimo, solidaridad y amistad sincera han sido una fuente de motivación y alegría que ha enriquecido nuestra vida universitaria.

Finalmente, nuestro agradecimiento más profundo es para nuestros seres queridos. Su constante ánimo, comprensión y amor incondicional han sido el pilar sobre el cual hemos construido este logro. Sin su fe en nosotros y su inquebrantable apoyo, este éxito no habría sido posible.

Gracias a todos ustedes, nuestra tesis ha cobrado vida. Cada uno de ustedes ha dejado una marca imborrable en nuestra trayectoria académica, y les estamos agradecidos más allá de lo que las palabras pueden expresar. Este logro también es suyo.

Con eterna gratitud,

Ligia Adriana Cepeda Pataron y Celio Gustavo Paguay Quiroz

## ÍNDICE GENERAL

DICTAMEN FAVORABLE DEL PROFESOR TUTOR.....	3
CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL.....	4
CERTIFICADO ANTIPLAGIO.....	5
DEDICATORIA.....	6
AGRADECIMIENTO .....	7
RESUMEN .....	13
ABSTRACT .....	14
CAPÍTULO I. INTRODUCCIÓN.....	15
1.1 Planteamiento del Problema.....	16
1.2 Justificación.....	16
1.3 Formulación del Problema .....	17
1.4 Objetivos .....	17
1.4.1 Objetivo General.....	17
1.4.2 Objetivos Específicos.....	17
CAPÍTULO II. MARCO TEÓRICO.....	18
2.1 Aplicación Web.....	18
2.2 Visual Studio Community .....	18
2.2.2 .NET7.....	18
2.3 Language-Integrated Query (LINQ) .....	19
2.4 Expresiones Lambda .....	21
2.5 LINQ Y Expresiones Lambda.....	22
2.6 Microsoft SQL Server .....	22
2.6.1 Seguridad de base de datos Microsoft SQL Server .....	23
2.7 Repositorio de base de datos .....	23
2.8 API .....	23



2.9 Angular.....	23
2.10 Organización Internacional de Normalización (ISO) 25010.....	24
2.10.1 Eficiencia de desempeño.....	24
2.11 Apache JMeter.....	25
2.12 Metodología Scrum.....	25
CAPÍTULO III. METODOLOGÍA.....	27
3.1 Tipo y diseño de la Investigación.....	27
3.1.1 Según la fuente de investigación.....	27
3.1.2 Según el tipo de variable.....	27
3.2 Técnicas de Recolección de Datos.....	27
3.3 Población y Muestra.....	27
3.4 Técnicas de análisis e interpretación de la información.....	27
3.5 Identificación de variables.....	27
3.5.1 Variable dependiente.....	27
3.5.2 Variable independiente.....	28
3.6 Operacionalización de variables.....	28
3.7 Desarrollo del módulo de matrícula.....	29
3.7.1 Fase de Inicio.....	29
3.7.2 Fase de Planificación y estimación.....	31
3.7.3 Fase de Implementación.....	35
3.7.4 Fase de Revisión y retrospectiva.....	38
CAPÍTULO IV. RESULTADOS Y DISCUSIÓN.....	40
4.1 Tiempo de Respuesta.....	40
4.2 Tiempo de Procesamiento.....	42
4.3 Discusión.....	43
4.3.1 Tiempos de respuesta.....	43

4.3.2 Tiempos de procesamiento .....	43
CAPÍTULO V. CONCLUSIONES y RECOMENDACIONES .....	44
BIBLIOGRAFÍA .....	46
ANEXOS .....	49
ANEXO 1. INSTALACION DE HERRAMIENTAS .....	50
ANEXO 2. ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE .....	53
ANEXO 3. CONFIGURACIÓN BACKEND .....	60
ANEXO 4. CONFIGURACIÓN FRONTEND .....	64
ANEXO 5. ACTA DE ENTREGA .....	67
ANEXO 6. MANUAL DE USUARIO .....	70
ANEXO 7. IMPLEMENTACIÓN FRONTEND.....	86
ANEXO 8. RESULTADOS EVALUACIÓN CON JMETER .....	92

## ÍNDICE DE TABLAS

Tabla 1: Fases de la metodología SCRUM .....	26
Tabla 2: Operacionalización de Variables .....	28
Tabla 3: Roles de la metodología SCRUM .....	29
Tabla 4: Requerimientos no funcionales .....	30
Tabla 5: Planificación Sprint 1 .....	31
Tabla 6: Planificación Sprint 2 .....	31
Tabla 7: Planificación Sprint 3 .....	32
Tabla 8: Planificación Sprint 4 .....	32
Tabla 9: Planificación Sprint 5 .....	33
Tabla 10: Planificación Sprint 6 .....	33
Tabla 11: Planificación Sprint 7 .....	33
Tabla 12: Comparación tiempos de respuesta operación 1 .....	40
Tabla 13: Comparación tiempos de respuesta operación 2 .....	41
Tabla 14: Comparación tiempos de procesamiento operación 1 .....	42
Tabla 15: Comparación tiempos de procesamiento operación 2 .....	42

## ÍNDICE DE FIGURAS

Figura 1: Operación de consulta con LINQ.....	21
Figura 2: Expresión Lambda .....	21
Figura 3: Operación de consulta con LINQ.....	22
Figura 4: Clasificación de la norma ISO 25010 .....	24
Figura 5: Apache JMeter .....	25
Figura 6: Requerimientos funcionales Estudiante .....	29
Figura 7: Diagrama de casos de uso .....	34
Figura 8: Diagrama de procesos .....	34
Figura 9: Diagrama de arquitectura .....	35
Figura 10: Base de datos del módulo de matrícula.....	35
Figura 11: API Creado para el sistema de titulación .....	36
Figura 12: GetNivelesCarreraByIdCarreraApi.....	37
Figura 13: Vista del Estudiante del Sistema de Matrículas .....	38
Figura 14: Inicio de sesión correcto .....	38
Figura 15: Mensaje de confirmación .....	39
Figura 16: Matrícula generada correctamente .....	39
Figura 17: Gráfica de tiempos de respuesta operación 1 .....	41
Figura 18: Gráfica de tiempos de respuesta operación 2.....	41

## RESUMEN

La investigación se basa en el desarrollo del módulo de matrícula para la unidad de titulación de la Universidad Nacional de Chimborazo utilizando Linq y expresiones Lambda. Asimismo, se realiza una evaluación del comportamiento temporal del sistema. Se identificó que el principal problema se presenta en la gestión manual de los procesos de titulación y la ausencia de un sistema que permita a los estudiantes realizar la matriculación en línea. La solución propuesta fue desarrollar un sistema con .NET7 e integrar el uso de nuevas tecnologías que permiten la reutilización de código y escalabilidad del sistema. Para tener un desarrollo eficiente y ágil se empleó la metodología SCRUM. La investigación adoptó un enfoque cuantitativo para evaluar el tiempo de respuesta y el tiempo de procesamiento del sistema desarrollado para la gestión de matrículas. Utilizando la herramienta JMeter, se realizaron pruebas de carga que revelaron que el uso de LINQ y expresiones lambda resulta significativamente más eficaz en comparación con sistemas tradicionales. Se obtuvieron promedios de tiempo de aproximadamente 1 milisegundo para la operación de Guardar Tipo de Matrícula y 2,35 milisegundos para Generar Matrícula. Además, los tiempos de procesamiento mostraron una clara ventaja para LINQ/Lambda con promedios de 2,1917 milisegundos y 2,35 milisegundos respectivamente, en contraste con aproximadamente 6,0833 milisegundos y 6,1975 milisegundos para el sistema tradicional en estas operaciones. Estos resultados subrayan la eficiencia mejorada del sistema con LINQ y Lambda, evidenciando su capacidad para optimizar el rendimiento y la velocidad de procesamiento en aplicaciones web.

**Palabras claves:** Linq, Lambda, .Net, Aplicación Web, Módulo de matrícula, SCRUM.

# ABSTRACT

## ABSTRACT

This research is based on the development of the enrollment module for the degree unit of the Universidad Nacional de Chimborazo using Linq and Lambda expressions and an evaluation of the temporal behavior of the system. The main problem was identified as the manual management of the degree processes and the absence of a system that allows students to enroll online. The proposed solution was to develop a system with .NET7 and integrate new technologies that would enable code reuse and system scalability. The Scrum methodology was used to achieve efficient and agile development. The research adopted a quantitative approach to evaluate the system's response time and processing time developed for license plate management. Load tests were performed using the JMeter tool, which revealed that LINQ and lambda expressions are significantly more efficient than traditional systems. Time averages of approximately one millisecond were obtained for the Save License Plate Type operation and 2.35 milliseconds for Generate License Plate. In addition, processing times showed a clear advantage for LINQ/Lambda with averages of 2.1917 milliseconds and 2.35 milliseconds, respectively, in contrast to approximately 6.0833 milliseconds and 6.1975 milliseconds for the traditional system for these operations. These results underscore the system's improved efficiency with LINQ and Lambda, evidencing its ability to optimize performance and processing speed in web applications.

**Keywords:** Linq, Lambda, .Net, Web Application, Enrollment Module, SCRUM.



Reviewed by:  
Ms.C. Ana Maldonado León  
ENGLISH PROFESSOR  
C.I.0601975980

## CAPÍTULO I. INTRODUCCIÓN

Los avances tecnológicos están transformando constantemente nuestra percepción del mundo, de forma que las actividades empresariales se adaptan rápidamente a las nuevas innovaciones automatizando procesos y mejorando sus productos y servicios obteniendo una gestión más eficiente y efectiva, lo que se traduce en mayor productividad y satisfacción para los usuarios.

Las aplicaciones web son herramientas accesibles y flexibles para los usuarios de internet que no necesitan software adicional y funcionan en cualquier dispositivo con conexión a internet. La uniformidad de la comunicación entre cliente y servidor facilita su uso, permitiendo compartir información en tiempo real y almacenar copias de seguridad en servidores remotos, lo que mejora la seguridad de los datos (Zea Ordóñez y otros, 2018).

Una herramienta clave para la integración de procesos y desarrollo de aplicaciones web es .NET 7, esta plataforma de Microsoft permite crear aplicaciones web modernas y escalables, convirtiéndose en una opción para organizaciones y empresas que buscan mejorar sus procesos y la experiencia de los usuarios. Net 7 utiliza tecnologías como Language Integrated Query (Linq) y expresiones Lambda (Microsoft, 2023).

Según Hector Godínez (2024) afirma que LINQ ofrece numerosas ventajas sobre el uso tradicional de SQL, especialmente en el entorno de desarrollo de .NET. Estas ventajas incluyen una mejor legibilidad y expresividad del código, verificación en tiempo de compilación, y facilidad para realizar operaciones sobre objetos y colecciones.

Las expresiones lambda son valiosas en programación por su capacidad para reducir el código repetitivo, mejorando la concisión y legibilidad del código base. También facilitan la reutilización del código al permitir que los desarrolladores pasen comportamientos como parámetros a métodos, lo que mejora la flexibilidad y mantenibilidad del software (FasterCapital, 2024).

Por otra parte, un sistema de matrículas en general ayuda a las instituciones educativas a optimizar recursos y agilizar los procesos relacionados. Estos sistemas permiten ahorrar tiempo, reducir errores, mejorar la eficiencia y organización, y facilitar el acceso de los estudiantes al proceso de registro (Lema Romero & Hernandez Castillo, 2018).

En este contexto, en la Universidad Nacional de Chimborazo (UNACH) se gestiona manualmente la matrícula de los estudiantes en la unidad de titulación lo que ocasiona retrasos en el proceso de matriculación y requiere de la asistencia presencial de los alumnos generando inconformidades en la comunidad académica.

Esta investigación permitió la automatización del módulo de matrícula mejorando procesos, realización de consultas y reducción de tiempos de espera utilizando LINQ y expresiones Lambda.

## **1.1 Planteamiento del Problema**

El impacto de la tecnología en la sociedad actual es innegable, y una de las áreas donde se ha notado más su influencia es en el uso de aplicaciones web revolucionado la forma en que las empresas y organizaciones llevan a cabo sus actividades, permitiendo una mayor eficiencia y rapidez en los procesos.

Las aplicaciones web permiten a los usuarios acceder a un servidor web por medio de internet desde un determinado navegador, teniendo como aspecto positivo la aceptación y usabilidad por parte de los usuarios efectuando peticiones y permitiendo el acceso simultáneo a operaciones por parte de múltiples usuarios (Molina Ríos et al., 2017). “Para desarrollar estas aplicaciones se pueden utilizar tecnologías como Linq y expresiones Lambda, las cuales facilitan la gestión y consulta de datos desde múltiples fuentes para filtrar, ordenar y transformar colecciones de objetos de forma eficiente” (Microsoft, 2023).

Por otra parte, los sistemas de matrículas de titulación permiten una gestión eficiente y efectiva de la inscripción y registro de estudiantes en instituciones de educación superior. Estos sistemas permiten el registro en línea, procesamiento de solicitudes, disponibilidad de información y gestión de registros que contribuyen a la agilización de procesos administrativos (Valdivieso, 2019).

## **1.2 Justificación**

En la UNACH la Unidad de Titulación consta de cuatro módulos como es: matriculación, planificación e integración curricular, proyecto de titulación y examen complejo, esta investigación se centra específicamente en el módulo de matrícula para las carreras vigentes donde se maneja las opciones de titulación, revisión de tiempos transcurridos entre matrículas, aprobación de matrículas y pagos de aranceles, cabe recalcar que actualmente el proceso se realiza de forma manual generando inconformidad en la comunidad académica, debido a que los estudiantes deben llenar los oficios de matrícula y entregarlos de forma física en las secretarías de sus respectivas carreras. Esta situación provoca dificultades en la toma de decisiones, aumenta la posibilidad de cometer errores, y puede resultar en la pérdida de documentación o informes, así como en la acumulación excesiva de papeles.

Por lo antes mencionado la presente investigación desarrolla una aplicación web con el uso de Linq y expresiones Lambda para gestionar los procesos de matrícula de titulación de forma automatizada, reducir recursos, optimizar tiempos y mejorar la integración con otras plataformas utilizadas en la gestión educativa, lo que contribuiría a la obtención de una mejor gestión educativa dentro de la universidad.



### **1.3 Formulación del Problema**

¿Cómo el uso de Linq y expresiones Lambda mejorará el comportamiento temporal del módulo de matrícula?

### **1.4 Objetivos**

#### **1.4.1 Objetivo General**

Implementar el módulo de matrícula para el sistema de titulación de la Universidad Nacional de Chimborazo utilizando Linq y expresiones Lambda.

#### **1.4.2 Objetivos Específicos**

- Analizar Linq y expresiones Lambda en el desarrollo de sistemas.
- Desarrollar el módulo de matrícula para el sistema de titulación de la Universidad Nacional de Chimborazo utilizando Linq y expresiones Lambda.
- Evaluar el comportamiento temporal del módulo de matrícula utilizando la norma ISO 25010.

## **CAPÍTULO II. MARCO TEÓRICO**

### **2.1 Aplicación Web**

El IONOS (2023) menciona que las aplicaciones web consisten en la creación de un software de aplicación que se guarda y se conserva en un servidor externo, al cual el usuario puede acceder por medio del navegador. La noción de aplicación web está estrechamente relacionada con el alojamiento en la nube. Por consiguiente, la información es almacenada de forma permanente en extensos servidores en línea. Una aplicación web (también conocida como web app) utiliza HTML, JavaScript y CSS como base. Debido a que se carga en el servidor web y se ejecuta en el navegador, de parte del usuario final no es necesario realizar una instalación previa de alguna herramienta para su funcionamiento.

### **2.2 Visual Studio Community**

El entorno de Visual Studio Community es muy popular entre los creadores, ya que es gratuito y se puede ampliar para desarrollar aplicaciones actualizadas en diferentes plataformas, como Windows, Android, IOS, aplicaciones web y servicios en la nube. Un factor clave de uso de Visual Studio Community es su gratuidad y la capacidad de crear aplicaciones en una amplia variedad de plataformas (Contreras Sánchez y Suárez Jácome, 2022).

#### **2.2.1 ASP.NET MVC**

Microsoft ha creado el framework ASP.NET MVC con el fin de desarrollar aplicaciones web que utilicen el patrón MVC (Modelo-Vista-Controlador). Este framework es gratuito y de código abierto, y viene integrado en Visual Studio 2022. ASP.NET MVC se basa en el marco de trabajo ASP.NET y utiliza el lenguaje de programación C# (Armas Navarrete, 2019).

Según lo descrito por Armas Navarrete (2019) mediante el uso de ASP.NET MVC se pueden crear sitios web utilizando HTML6, CSS7 y JavaScript8. Es posible implementar el código de programación utilizando lenguajes como C# o VB.NET. El patrón arquitectónico MVC, que divide una aplicación en tres componentes principales, se emplea en este framework. La lógica así como los datos se representarían en el modelo, el diseño y presentación sería la vista y finalmente el controlador que sería el intermediario entre los dos anteriores y que además se encarga de que el usuario pueda interactuar con el sistema.

#### **2.2.2 .NET7**

Microsoft ha creado una plataforma de desarrollo de software llamada .NET, que es empleada para la producción de una amplia variedad de software, que incluye aplicaciones de escritorio, aplicaciones web, servicios en la nube y otros programas

informáticos. .NET 7 es una versión de esta plataforma de desarrollo de software de Microsoft, que fue lanzada en noviembre de 2021. Esta versión incluye mejoras en el rendimiento y la eficiencia, así como nuevas características y funcionalidades (Freire, 2022).

Una característica sobresaliente en .NET 7 es la capacidad de ejecutar operaciones matemáticas con alta precisión, gracias a la inclusión de cálculos de punto flotante con mayor resolución. Asimismo, se han llevado a cabo mejoras que optimizan el rendimiento y eficiencia, se ha mejorado el soporte para la programación asíncrona, se ha ampliado la compatibilidad con diversas plataformas y sistemas operativos, y se han integrado nuevas funcionalidades que facilitan el desarrollo de aplicaciones de escritorio, aplicaciones web y servicios en la nube (Freire, 2022).

Según Santos Garrido (2020) manifiesta que se ha aprovechado el uso del framework .NET debido a que, además de ser la plataforma principal para compilar y ejecutar código C#, ofrece un conjunto de herramientas y bibliotecas que aceleran el desarrollo y mejoran la eficiencia del código. Un ejemplo destacado es LINQ, el cual brinda métodos para trabajar con operaciones lambda y colecciones, permitiendo aprovechar todos los núcleos del procesador y optimizar el rendimiento del sistema.

### **2.3 Language-Integrated Query (LINQ)**

LINQ es un modelo de programación que proporciona una forma completamente nueva de consultar y manipular datos en los proyectos .NET de Microsoft. LINQ pretende superar las limitaciones del lenguaje SQL estándar incrustado y, al mismo tiempo, aumentar la productividad del trabajo de programación proporcionando una sintaxis más corta, significativa y expresiva para el procesamiento de datos (Turrado, 2019).

LINQ es un conjunto de tecnologías que permite la integración de funciones de consulta directamente en el lenguaje C#. Antes, las consultas de datos se realizaban como simples cadenas sin la verificación de tipos en tiempo de compilación y sin compatibilidad con IntelliSense, lo que obligaba a aprender un lenguaje de consulta diferente para cada tipo de origen de datos. Con LINQ, una consulta se convierte en una construcción de lenguaje de primera clase, como clases, métodos y eventos, lo que permite escribir consultas en colecciones de objetos fuertemente tipados con palabras clave del lenguaje y operadores familiares. Además, la familia de tecnologías de LINQ proporciona una experiencia de consulta uniforme para objetos, bases de datos relacionales y XML a través de LINQ to Objects, LINQ to SQL y LINQ to XML, respectivamente (Microsoft, 2023).

La tecnología LINQ fue desarrollada por Microsoft con el fin de incorporar consultas de manera directa en el código de programación, utilizando una sintaxis parecida a SQL. Es posible utilizar LINQ con varios lenguajes de programación, tales como C#, Visual Basic.NET y F#, y se puede emplear para realizar consultas en diversas fuentes de

información, tales como colecciones de objetos en memoria, archivos XML, servicios web y bases de datos, entre otras.

El autor Armas Navarrete (2019) manifiestan que se trata de un conjunto de herramientas que proporcionan una estructura de búsqueda consistente para recuperar información de diversas fuentes y formatos. La estructura de búsqueda de LINQ posibilita la realización de tareas de filtrado, ordenación y agrupación de datos con poco código. Además, las mismas estructuras de búsqueda se pueden aplicar a distintas fuentes de datos, como colecciones, matrices, documentos XML y bases de datos SQL, entre otras. Por lo tanto, una de las ventajas principales de LINQ es su capacidad para llevar a cabo búsquedas con tipos de datos fuertes, lo que permite detectar errores de consulta durante la compilación en lugar de la ejecución, lo que reduce los errores en el código. Además, LINQ es altamente adaptable y fácil de usar, lo que permite a los programadores generar búsquedas complejas de manera rápida y sencilla.

Ventajas de LINQ:

- **Legibilidad:** Utiliza una sintaxis clara y legible que facilita la lectura y comprensión del código, lo que reduce la posibilidad de errores y hace que el código sea más fácil de mantener y depurar.
- **Productividad:** Permite realizar tareas comunes de consulta y manipulación de datos en menos líneas de código que las soluciones convencionales, lo que puede reducir el tiempo de desarrollo y aumentar la productividad.
- **Composición:** Es muy flexible y permite combinar varias operaciones de consulta y manipulación de datos en una sola consulta, lo que puede mejorar la eficiencia y reducir la complejidad del código.
- **Tipado fuerte:** Utiliza el sistema de tipos fuertemente tipados de .NET, lo que garantiza que los errores de tipo se detecten en tiempo de compilación en lugar de en tiempo de ejecución, lo que mejora la calidad y fiabilidad del código.
- **Interoperabilidad:** Es compatible con muchos tipos de datos y proveedores de datos, lo que permite integrar fácilmente los datos de diferentes fuentes y plataformas.

En general, LINQ es una herramienta muy útil para la manipulación y consulta de datos en .NET, que puede mejorar significativamente la eficiencia, la calidad y la legibilidad del código.

En la figura 1 se muestra un ejemplo de una operación de consulta, incluyendo la creación del origen de datos, la definición de la expresión de consulta y su ejecución mediante una instrucción foreach utilizando LINQ.

```

C#

// Specify the data source.
int[] scores = { 97, 92, 81, 60 };

// Define the query expression.
IEnumerable<int> scoreQuery =
    from score in scores
    where score > 80
    select score;

// Execute the query.
foreach (int i in scoreQuery)
{
    Console.Write(i + " ");
}

// Output: 97 92 81

```

**Figura 1:** Operación de consulta con LINQ

## 2.4 Expresiones Lambda

Según Contreras Sánchez y Suárez Jácome (2022) afirman que una función anónima que devuelve un único valor se conoce como expresión lambda. Siempre que el delegado sea válido, estas expresiones pueden ser utilizadas.

Las funciones anónimas o sin nombre son comúnmente conocidas como funciones lambda y son altamente utilizados con LINQ para crear delegados, dichas funciones no poseen modificadores de acceso y devuelven el resultado de la evaluación de una condición especificada (Armas Navarrete, 2019).

Las funciones lambda son altamente convenientes en lenguajes de programación que soportan la programación funcional, como C#, Python o Java, debido a que permiten la escritura de código más clara y expresiva y posibilitan el uso de técnicas avanzadas de programación como LINQ, la programación asíncrona o la programación reactiva.

Una expresión lambda generalmente se lo reconoce por situarse después del operador => es decir, al lado derecho. Una expresión lambda devuelve el resultado de evaluar la condición y tiene la forma que se visualiza en la figura 2.

```

C#

(input-parameters) => expression

```

**Figura 2:** Expresión Lambda

Las expresiones lambda tienen varias ventajas, entre las que se incluyen:

- **Concisión:** permiten escribir funciones anónimas de manera muy concisa, lo que puede hacer que el código sea más legible y fácil de entender.
- **Flexibilidad:** son muy flexibles, ya que pueden ser usadas en una variedad de situaciones y contextos.
- **Funciones de orden superior:**  
Estas funciones pueden tomar otras funciones como argumentos y/o devolver funciones como resultado, lo que aumenta la flexibilidad y la capacidad de reutilización del código.
- **Eficiencia:** pueden ser más eficientes que otras formas de crear funciones, especialmente cuando se trata de operaciones simples y repetitivas.
- **Reducción de errores:** Al escribir funciones de manera más concisa y legible, se reduce la posibilidad de cometer errores, lo que puede mejorar la calidad y fiabilidad del código.

En general, las expresiones lambda son una herramienta poderosa y versátil que puede mejorar significativamente la calidad y la eficiencia del código.

## 2.5 LINQ Y Expresiones Lambda

En la figura 3, se muestra un ejemplo de cómo se puede utilizar la combinación de LINQ y expresiones lambda para realizar consultas entre diferentes tablas de la base de datos directamente desde el código en c#.

```
// Usar LINQ con expresiones lambda para obtener los datos
var estudiantesConOpcionesTitulacion = _context.SeleccionOpcionEstudiante
    .Include(se => se.Usuario) // Incluir la entidad Usuario
    .Include(se => se.ConfiguracionTitulacion)
    .ThenInclude(ct => ct.OpcionTitulacion) // Incluir la entidad OpcionTitulacion
    .Where(se => se.ConfiguracionTitulacion.IdOpcionTitulacion != null) // Filtro
    .OrderBy(se => se.Usuario.ApellidoPaterno) // Ordenar por ApellidoPaterno
    .Select(se => new
    {
        IdEstudiante = se.IdEstudiante,
        IdUsuario = se.Usuario.IdUsuario,
        IdOpcionTitulacion = se.ConfiguracionTitulacion.IdOpcionTitulacion,
        IdConfiguracionTitulacion = se.ConfiguracionTitulacion.IdConfiguracionTitulacion
    })
    .Take(10) // Tomar solo los primeros 10 resultados
    .ToList();

return Ok(estudiantesConOpcionesTitulacion);
```

Figura 3: Operación de consulta con LINQ

## 2.6 Microsoft SQL Server

Según el criterio de Parinango y Alfredo (2022), “SQL Server es un producto de Microsoft el cual es utilizado para almacenar y recuperar datos según sea solicitado de las diferentes aplicaciones de software la cuales se relacionan con bases de datos relacionales”.

## **2.6.1 Seguridad de base de datos Microsoft SQL Server**

SQL Server ofrece dos opciones de autenticación: la de Windows, que se integra con usuarios de dominio, y el modo mixto, que admite tanto la autenticación de Windows como la de SQL Server. La segmentación de roles incluye roles fijos del servidor y de la base de datos para simplificar la administración de permisos. Los propietarios de objetos tienen permisos irrevocables, evitando la eliminación de usuarios con objetos de otros propietarios. La estrategia de seguridad se basa en privilegios mínimos, con SQL Server proporcionando funciones avanzadas de cifrado y descifrado mediante certificados y claves simétricas y asimétricas (Parinango y Alfredo, 2022).

## **2.7 Repositorio de base de datos**

El autor López González (2020) destaca que el patrón de diseño conocido como "repositorio" es ampliamente adoptado en la industria del software para abstraer los detalles de la capa de datos. En esencia, se trata de una interfaz que proporciona métodos estandarizados para recuperar y actualizar entidades específicas. Un repositorio de base de datos actúa como un punto central donde se gestionan los datos de una base de datos, actuando como una capa intermedia entre la lógica de la aplicación y la base de datos subyacente. Su objetivo principal es ofrecer una interfaz coherente y simplificada para interactuar con los datos almacenados, independientemente del tipo de base de datos o la complejidad de su estructura.

El repositorio de base de datos se encarga de realizar todas las operaciones necesarias para acceder, manipular y actualizar los datos de la base de datos, incluyendo la validación de la información y la gestión de la transacción. Además, puede incluir funcionalidades como el caché de datos para mejorar el rendimiento de la aplicación, la gestión de versiones de los datos o la implementación de políticas de seguridad (López González, 2020).

## **2.8 API**

Una interfaz de programación de aplicaciones (API) se define como un conjunto de reglas que permite la comunicación entre programas informáticos, facilitando la transmisión de datos entre ellos. Las API ayudan a los desarrolladores a evitar trabajos redundantes permitiéndoles integrar funciones existentes en nuevas aplicaciones mediante solicitudes adecuadamente formateadas. De manera similar a cómo un cajero automático permite a los clientes interactuar con su banco, una API permite a un software interactuar con otro para acceder a servicios necesarios (Cloudflare, 2024).

## **2.9 Angular**

Según el autor Deyimar A (2023) angular manifiesta que Angular es un framework de código abierto desarrollado por Google para crear aplicaciones web de una sola página y

menús animados para páginas HTML y utiliza la arquitectura Modelo-Vista-Controlador (MVC) comúnmente utilizado en el desarrollo de aplicaciones web. Este framework facilita la vinculación de JavaScript y HTML, sincronizando el código y reduciendo la cantidad de código necesario para los desarrolladores. Entre sus ventajas destacan el enlace bidireccional de datos, directivas que amplían la funcionalidad de HTML, estructura de código limpia, soporte para pruebas unitarias y de integración, y compatibilidad con navegadores de escritorio y móviles, lo que garantiza un futuro prometedor debido a su creciente popularidad y extensa documentación.

## 2.10 Organización Internacional de Normalización (ISO) 25010

El modelo de calidad sirve como base para evaluar la calidad de un producto de software. El modelo define las características de calidad consideradas necesarias para evaluar las propiedades del software.

La calidad de un software como producto se refiere al nivel de satisfacción de los requisitos de usuario y aporte de valor. Estos requisitos, tales como funcionalidad, rendimiento, seguridad y mantenibilidad, son categorizados en el modelo de calidad en características y subcaracterísticas.

De esta forma, se puede evaluar la calidad del software y establecer estándares y criterios de evaluación para su mejora continua (Organización Internacional de Normalización [ISO 25000], 2022).

La figura 4 muestra la clasificación de las normas ISO 25010 para la evaluar la calidad del producto del software.



Figura 4: Clasificación de la norma ISO 25010

Fuente: (ISO 25000, 2022).

### 2.10.1 Eficiencia de desempeño

Para medir esta eficiencia, se consideran aspectos como:

- **Comportamiento temporal.** Evalúa los tiempos de respuesta y procesamiento y las ratios de throughput del sistema en función de un benchmark establecido.

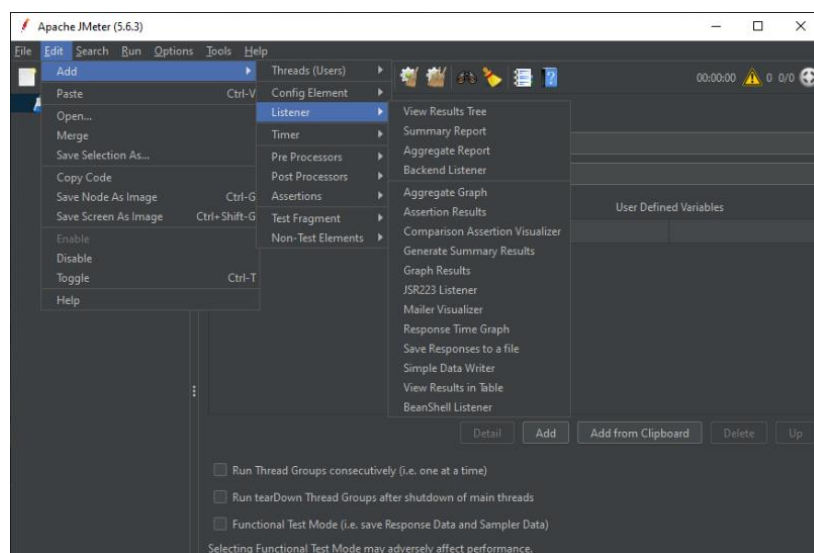


- **Utilización de recursos.** Evalúa las cantidades y tipos de recursos empleados por el software bajo condiciones determinadas.
- **Capacidad.** Evalúa el grado en que los límites máximos de un parámetro del software cumplen con los requisitos establecidos.

## 2.11 Apache JMeter

Se trata de una herramienta de código abierto que consiste en una aplicación Java completamente pura. Esta aplicación tiene como objetivo principal cargar, evaluar el comportamiento funcional y medir el rendimiento de distintos sistemas. Aunque su uso inicial se enfocaba en la evaluación de aplicaciones web, ha evolucionado para poder realizar otras funciones de prueba. Permite medir el rendimiento de una aplicación, tanto de recursos estáticos como dinámicos. Además, es capaz de evaluar aplicaciones web dinámicas, simulando el comportamiento de varios usuarios a la vez para evaluar su capacidad de respuesta (Apache Software Foundation, 2023).

En la figura 5 tenemos la página principal del programa en la que se puede observar la interfaz de Apache JMeter y una breve lista de herramientas que ofrece. Entre estas se incluyen opciones para configurar y ejecutar pruebas, como samplers para distintos protocolos (HTTP, HTTPS, etc.), listeners para recolectar y visualizar datos, entre otras. Estas características hacen de JMeter una herramienta flexible y poderosa para evaluar y mejorar el rendimiento de aplicaciones en entornos de prueba y producción.



**Figura 5:** Apache JMeter

**Fuente:** (Apache Software Foundation, 2023).

## 2.12 Metodología Scrum

La metodología Scrum resulta muy adecuada para el desarrollo de aplicaciones web, debido a su enfoque iterativo e incremental que permite gestionar y controlar de manera más eficiente proyectos complejos de software.

Al dividir el proceso de desarrollo en sprints cortos, se logra una mayor adaptabilidad a los cambios en las necesidades del cliente, lo que se traduce en un mayor valor para el producto final. Además, Scrum fomenta la colaboración y el trabajo en equipo, lo que puede mejorar la comunicación entre los miembros del equipo y fomentar la creatividad y la innovación (Sáez Hurtado, 2021).

La tabla 1 describe las fases de la metodología SCRUM.

**Tabla 1:** Fases de la metodología SCRUM

<b>Fases</b>	<b>Descripción</b>
<b>Inicio</b>	Se analiza el proyecto para identificar las necesidades básicas de cada sprint para llegar al objetivo del proyecto general.
<b>Planificación y estimación</b>	Se delega tareas y se estima tiempos de entrega, de forma que se logra tener un orden de trabajo que se puede clasificar según la prioridad.
<b>Implementación</b>	En esta fase de desarrollo no se debería permitir la realización de cambios repentinos que no son necesarios.
<b>Revisión y retrospectiva</b>	La revisión del proceso es donde se realiza una autoevaluación en el equipo evaluando el trabajo realizado y recabando críticas internas.
<b>Lanzamiento</b>	Desenlace del proyecto y entrega del producto

**Fuente:** (Jaimes Nava, 2023)

## **CAPÍTULO III. METODOLOGÍA**

### **3.1 Tipo y diseño de la Investigación**

La investigación es de tipo aplicada debido a que se desarrolló una aplicación web para el control del módulo de matrículas de la unidad de titulación, incorporando los conocimientos adquiridos durante la carrera.

#### **3.1.1 Según la fuente de investigación**

La fuente teórica se basó en investigaciones bibliográficas, puesto que se recolectó información de varios artículos científicos, sitios web, libros y revistas relacionadas con el tema de investigación.

#### **3.1.2 Según el tipo de variable**

El tipo de variable fue cuantitativa, debido a que el objetivo principal fue verificar los tiempos de respuesta y procesamiento mediante una comparativa en dos escenarios planteados: con la aplicación Linq y Lambda y otro sin estas.

### **3.2 Técnicas de Recolección de Datos**

Para la obtención de resultados, se emplearon métodos específicos de recolección de datos utilizando la herramienta JMeter.

### **3.3 Población y Muestra**

De acuerdo con la investigación planteada se evaluó el comportamiento temporal en términos de tiempos de respuesta y procesamiento por tal razón se trata de una población infinita puesto que se obtuvieron datos de diferentes mediciones utilizando JMeter.

### **3.4 Técnicas de análisis e interpretación de la información**

En la investigación se realizó un análisis comparativo entre dos escenarios planteados: una aplicación con las tecnologías Linq y expresiones Lambda y otro sin estas. Posteriormente con la herramienta de evaluación JMeter y la aplicación web del módulo de matrículas se realizó la medición de criterios de desempeño con enfoque al comportamiento temporal establecidos por la norma ISO 25010.

### **3.5 Identificación de variables**

#### **3.5.1 Variable dependiente**

Comportamiento temporal del Módulo de matrícula.

### 3.5.2 Variable independiente

Uso de Linq y expresiones Lambda.

### 3.6 Operacionalización de variables

En la Tabla 2 se describen el problema, objetivos y variables del tema, además de los indicadores para cada variable.

**Tabla 2:** Operacionalización de Variables

<b>Pregunta de investigación</b>	<b>Tema</b>	<b>Objetivos</b>	<b>Variables</b>	<b>Indicadores</b>
¿Cómo el uso de Linq y expresiones Lambda mejorará el comportamiento temporal del módulo de matrícula?	Módulo de matrícula para el sistema de titulación de la Universidad Nacional de Chimborazo utilizando Linq y expresiones Lambda	<b>General</b> Implementar el módulo de matrícula para el sistema de titulación de la Universidad Nacional de Chimborazo utilizando Linq y expresiones Lambda.	<b>Independiente:</b> Uso de Linq y expresiones Lambda.	Reducción de líneas de código Métodos Linq y Lambda: <ul style="list-style-type: none"> <li>• Take</li> <li>• Where</li> <li>• OrderBy</li> <li>• Select</li> <li>• Include</li> </ul>
		<b>Específicos:</b> <ul style="list-style-type: none"> <li>• Analizar Linq y expresiones Lambda en el desarrollo de sistemas.</li> <li>• Desarrollar el módulo de matrícula para el sistema de titulación de la Universidad Nacional de Chimborazo utilizando Linq y expresiones Lambda.</li> <li>• Evaluar el comportamiento temporal del módulo de matrícula utilizando la norma ISO 25010.</li> </ul>	<b>Dependiente:</b> Comportamiento temporal del Módulo de matrícula.	Norma ISO 25010: Eficiencia de desempeño. <ul style="list-style-type: none"> <li>• Tiempo de respuesta</li> <li>• Tiempo de procesamiento</li> </ul>

### 3.7 Desarrollo del módulo de matrícula

La aplicación web se desarrolló siguiendo la metodología Scrum, la cual se caracteriza por su enfoque en la flexibilidad y la colaboración constante entre el equipo de desarrollo y los stakeholders. Esto permitió adaptarse ágilmente a los cambios en los requisitos que surgieron a lo largo del proyecto. A continuación, se describen los tres roles importantes que participaron en el desarrollo del proyecto, detallados en la Tabla 3:

**Tabla 3:** Roles de la metodología SCRUM

Rol	Competencia	Descripción
Product Owner	Ing. Danny Velasco	Profesor de la carrera.
Scrum Master	Ing. Stefano Aguayo	Encargado de la orientación y control del proyecto
Scrum Team	Adriana Cepeda, Celio Paguay	Encargados del diseño y desarrollo del módulo de matrícula.

La metodología Scrum consta de 5 fases, las cuales son desarrolladas a continuación:

#### 3.7.1 Fase de Inicio

En esta fase, se realizó un análisis inicial del proyecto para identificar las necesidades básicas y definir los objetivos generales del módulo de matrícula. Además, se realizó la instalación de herramientas para el desarrollo del sistema. Para más detalle ver anexo 1.

##### 3.7.1.1 Requerimientos Funcionales

Los requerimientos funcionales fueron recopilados a partir de las peticiones de la Dirección Académica a CODESI. Se realizaron reuniones y entrevistas con los stakeholders para comprender sus necesidades y expectativas sobre el sistema de matrícula.

Nombre de requisito: <ESTUDIANTE>					
No	Prioridad	Como	Necesito	Para	Criterios de aceptación
1	ALTA	Estudiante	Registrarse en el sistema	Ingresar con una cuenta válida al sistema	Los estudiantes deben registrarse para poder ingresar al sistema.
2	ALTA	Estudiante	Ingresar al sistema	Postular a una opción de titulación	El estudiante puede postular a una opción de titulación, aceptando los términos requeridos.
3	ALTA	Estudiante	Seleccionar opción de titulación	Seleccionar una opción de titulación disponible para su carrera	El estudiante puede seleccionar una opción de titulación disponible para su carrera.
4	ALTA	Estudiante	Ingresar al apartado de matriculación	Verificar la situación actual del estudiante en su proceso de titulación	El sistema verifica la situación actual del estudiante en su proceso de titulación.
5	ALTA	Estudiante	Ver matriculas en titulación	Ver sus matriculas actuales y futuras en el sistema	El estudiante puede ver sus matriculas actuales y futuras en el sistema.
6	ALTA	Estudiante	Aceptar nueva matrícula	Aceptar la matrícula generada para el periodo en curso	El estudiante puede aceptar la matrícula generada para el periodo en curso.

**Figura 6:** Requerimientos funcionales Estudiante

En la figura 6 se muestra uno de los requerimientos funcionales elaborados. Estos requerimientos fueron documentados y priorizados según su importancia. Para ver los requerimientos funcionales completos, consulte el Anexo 2.

### 3.7.1.2 Requerimientos No Funcionales

En la tabla 4, se describen los requisitos no funcionales que establecen los criterios del funcionamiento general del sistema, destacando la importancia del tiempo de respuesta y tiempo de procesamiento como objetivo central de la investigación.

**Tabla 4:** Requerimientos no funcionales

<b>Rol</b>	<b>Competencia</b>
Seguridad	La aplicación web debe cumplir con altos estándares de seguridad para proteger los datos de los usuarios.
Escalabilidad	La aplicación debe ser capaz de manejar incrementos en la carga de usuarios sin degradar el rendimiento.
Usabilidad	La interfaz de usuario debe ser intuitiva y de fácil manejo para los usuarios finales.
Disponibilidad	La aplicación debe estar disponible en todo momento, minimizando el tiempo de mantenimiento planificado.
Rendimiento	La aplicación web debe procesar operaciones de manera rápida y eficiente.
Adaptabilidad	La aplicación debe ser compatible con diversos dispositivos y navegadores web, garantizando una experiencia de usuario consistente.
Tiempo de respuesta	El sistema debe responder a las solicitudes de los usuarios dentro de tiempos aceptables.
Tiempo de procesamiento	El sistema debe procesar operaciones internas de manera rápida y eficiente.

### 3.7.1.3 Análisis de funcionalidades y caracterización

Se realizó un análisis detallado de la documentación proporcionada por la Dirección Académica de la UNACH. Este análisis permitió identificar y caracterizar las funcionalidades necesarias para el desarrollo del módulo de matrícula. Se examinaron los procesos actuales, se identificaron las áreas de mejora y se definieron las características clave que debía incluir la nueva aplicación web. Este análisis fue fundamental para asegurar que el sistema desarrollado cumpliera con las expectativas y necesidades de los usuarios finales.

### 3.7.2 Fase de Planificación y estimación

#### 3.7.2.1 Planificación de Sprints

Se realizó la planificación detallada de los sprints de desarrollo del módulo de matrícula. Cada sprint se diseñó meticulosamente para abordar una serie específica de funcionalidades y tareas críticas, las cuales fueron priorizadas según los requisitos detallados en el backlog del producto. Este proceso garantizó una distribución efectiva de los esfuerzos de desarrollo, asegurando que las características más importantes y demandadas por los usuarios fueran implementadas de manera iterativa y en ciclos de desarrollo claramente definidos.

#### SPRINT 1

Para el Sprint 1 se estimó un tiempo de 6 semanas de desarrollo para los ítems descritos en la tabla 5.

Tabla 5: Planificación Sprint 1

Tiempo	Actividades	Responsable
6 semanas	Levantamiento de información	Scrum Team Product Owner
	Análisis de requerimientos	Scrum Master Scrum Team Product Owner
	Análisis de la caracterización de titulación	Scrum Team Product Owner Scrum Master

#### SPRINT 2

Para el Sprint 2 se estimó un tiempo de 2 semanas de desarrollo para los ítems descritos en la tabla 6.

Tabla 6: Planificación Sprint 2

Tiempo	Actividades	Responsable
2 semanas	Planificación de actividades y estimación de tiempos	Scrum Team
	Elaboración del cronograma de Planificación de la investigación	Scrum Master Scrum Team

### SPRINT 3

Para el Sprint 3 se estimó un tiempo de 7 semanas de desarrollo para los ítems descritos en la tabla 7.

**Tabla 7:** Planificación Sprint 3

<b>Tiempo</b>	<b>Actividades</b>	<b>Responsable</b>
7 semanas	Planificación del diseño del sistema de matrícula	Scrum Team Scrum Master
	Elaboración del diseño del sistema de matrícula	Scrum Team Scrum Master
	Elaboración del documento “Especificaciones de requisitos del software”	Scrum Team
	Esquematización del diseño de interfaz	Scrum Team Scrum Master

### SPRINT 4

Para el Sprint 4 se estimó un tiempo de 5 semanas de desarrollo para los ítems descritos en la tabla 8.

**Tabla 8:** Planificación Sprint 4

<b>Tiempo</b>	<b>Actividades</b>	<b>Responsable</b>
5 semanas	Creación de la base de datos	Scrum Team Scrum Master
	Normalización de tablas en la base de datos	Scrum Team Scrum Master
	Generar diagrama de base de datos.	Scrum Team
	Presentación de diseño de base de datos a CODESI	Scrum Team Scrum Master



## SPRINT 5

Para el Sprint 5 se estimó un tiempo de 8 semanas de desarrollo para los ítems descritos en la tabla 9.

**Tabla 9:** Planificación Sprint 5

<b>Tiempo</b>	<b>Actividades</b>	<b>Responsable</b>
8 semanas	Desarrollo del sistema	Scrum Team Scrum Master
	Creación de DTOs	Scrum Team
	Creación de endpoints para los diferentes roles	Scrum Team
	Implementación de controladores	Scrum Team

## SPRINT 6

Para el Sprint 6 se estimó un tiempo de 6 semanas de desarrollo para los ítems descritos en la tabla 10.

**Tabla 10:** Planificación Sprint 6

<b>Tiempo</b>	<b>Actividades</b>	<b>Responsable</b>
6 semanas	Creación del proyecto con Angular	Scrum Team Scrum Master
	Creación de componentes	Scrum Team
	Creación de servicios	Scrum Team

## SPRINT 7

Para el Sprint 7 se estimó un tiempo de 2 semanas de desarrollo para los ítems descritos en la tabla 11.

**Tabla 11:** Planificación Sprint 7

<b>Tiempo</b>	<b>Actividades</b>	<b>Responsable</b>
2 semanas	Socialización del sistema con CODESI	Scrum Team Product Owner Scrum Master
	Socialización del sistema con dirección académica	Scrum Team Product Owner Scrum Master

### 3.7.2.3 Diagrama de caso de uso

El diagrama de casos de uso muestra de manera clara cómo las personas o sistemas interactúan con el software. En la figura 7 se observa las acciones que realizan los actores en el sistema de matrícula de titulación.

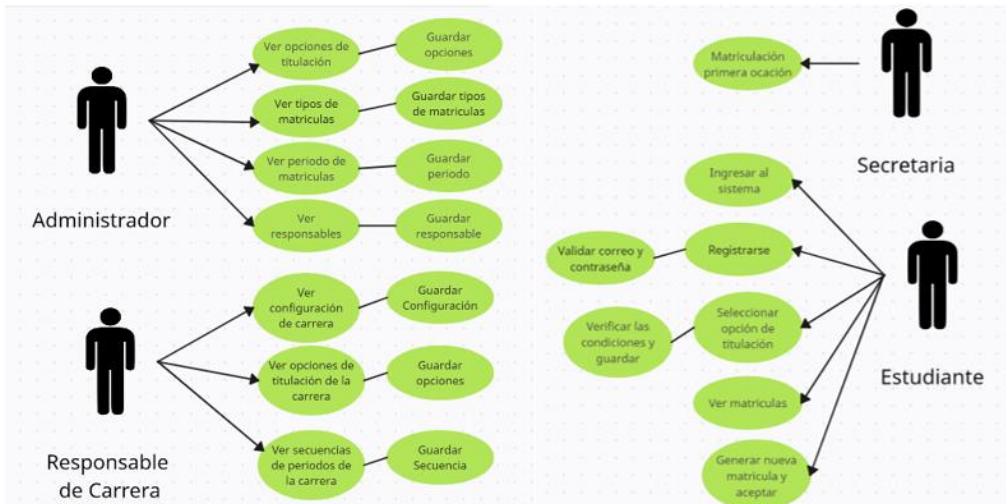


Figura 7: Diagrama de casos de uso

### 3.7.2.4 Diagrama de procesos

En el siguiente diagrama se representa gráficamente las diferentes etapas o pasos involucrados en la ejecución de un proceso o actividad. La figura 8 se representa secuencialmente las etapas o pasos del proceso de matrícula en la unidad de titulación.

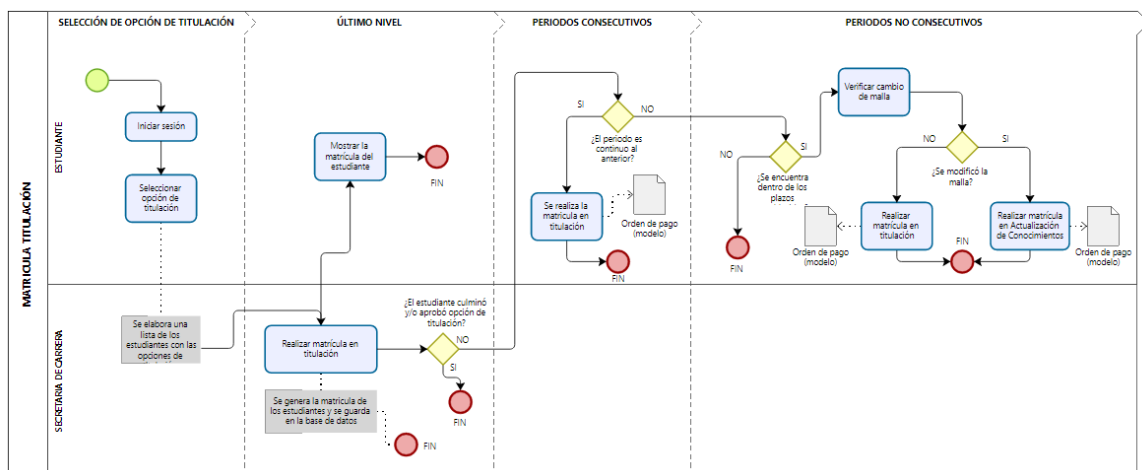


Figura 8: Diagrama de procesos

### 3.7.2.5 Diagrama de arquitectura

En el diagrama de arquitectura representa visualmente la estructura, componentes y relaciones de un sistema o aplicación de software. En la figura 9 se observa los

componentes principales de la aplicación web del módulo de matrícula, sus relaciones y cómo interactúan entre sí.

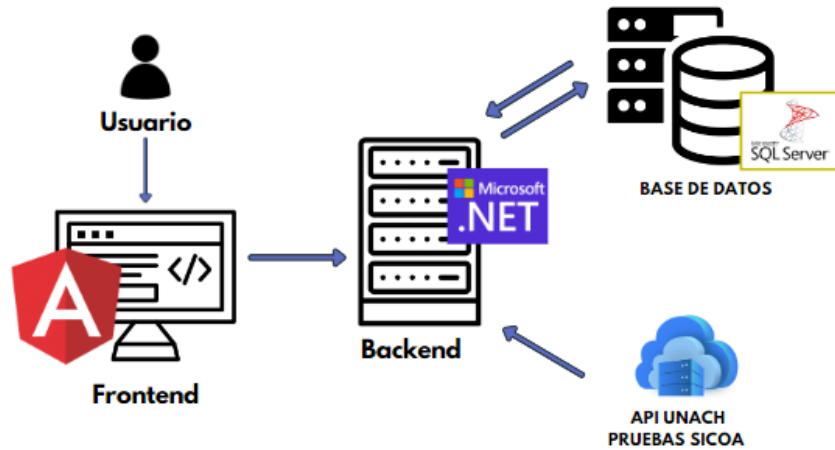


Figura 9: Diagrama de arquitectura

### 3.7.3 Fase de Implementación

#### 3.7.3.1 Base de datos

En la figura 10 se observa el esquema de la base de datos, su estructura y las relaciones diseñada específicamente para el módulo de matrícula del sistema de titulación de la Universidad Nacional de Chimborazo.



Figura 10: Base de datos del módulo de matrícula

#### 3.7.3.2 BackEnd

Para la implementación del backend se consideró la creación de controladores, perfiles de mapeo, configuraciones para el consumo de API, DTOs, librerías, clases y configuraciones que fueron necesarias para la correcta creación de los endpoints del sistema desarrollado. Para más detalle ver anexo 3.

Como resultado, se obtiene los métodos GET y POST como se evidencia en la figura 11.

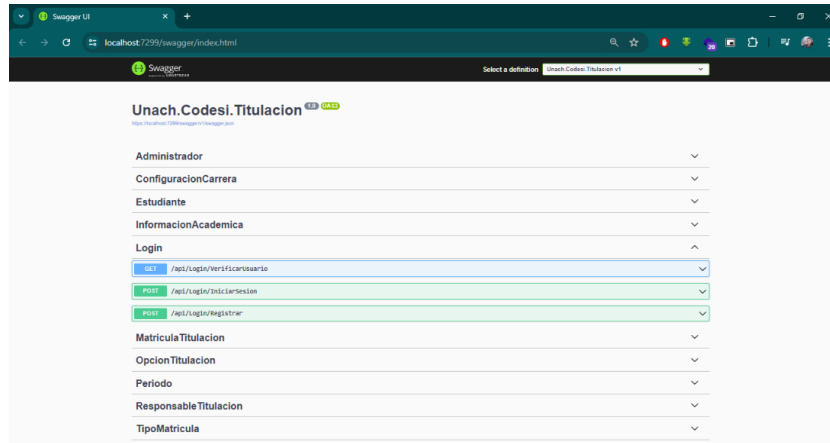


Figura 11: API Creado para el sistema de titulación

### Uso de LINQ y Expresiones Lamba:

En la figura 12 se presenta el método GetNivelesCarreraByIdCarreraApi, que utiliza LINQ y expresiones lambda en C# para filtrar y seleccionar datos de manera eficiente.

A continuación, se detallan las partes clave de este proceso:

#### Filtrado con LINQ:

**Where:** Se utiliza para seleccionar solo los niveles de carrera que pertenecen al sistema de estudios semestral.

```
nivelesCarrera.Where(n => n.SistemaEstudiosID == (int)CatalogoSistemaEstudios.SEMESTRAL)
```

Esto asegura que solo se incluyan los niveles que cumplen con la condición.

#### Proyección con Expresiones Lambda:

**Select:** Se usa para crear una nueva estructura de datos con solo los campos necesarios (NivelID y Nivel).

```
.Select(n => new EntidadNivelesCarreraBySistemaEstudios  
{  
    NivelID = n.NivelID,  
    Nivel = n.Nivel  
}).ToList();
```

Esta parte transforma cada elemento en una nueva instancia de: *EntidadNivelesCarreraBySistemaEstudios*.

### Encapsulamiento de la Respuesta:

Los resultados filtrados y proyectados se encapsulan en un objeto: *EntidadRespuesta<List<EntidadNivelesCarreraBySistemaEstudios>>*,

que incluye la lista de niveles de carrera y un mensaje descriptivo.

```
var respuesta = new EntidadRespuesta<List<EntidadNivelesCarreraBySistemaEstudios>>()  
{  
    Entidad = nivelesFiltrados,  
    Mensaje = " Niveles de carrera obtenidos correctamente "  
};
```

### Devolución de los Resultados:

El método devuelve un *OkResult* con la respuesta encapsulada, indicando que la operación se completó correctamente.

```
return Ok(respuesta);
```

Este enfoque demuestra cómo usar LINQ y expresiones lambda de manera efectiva en aplicaciones web API desarrolladas en C#, logrando un código más claro y eficiente para manipular y transformar datos.

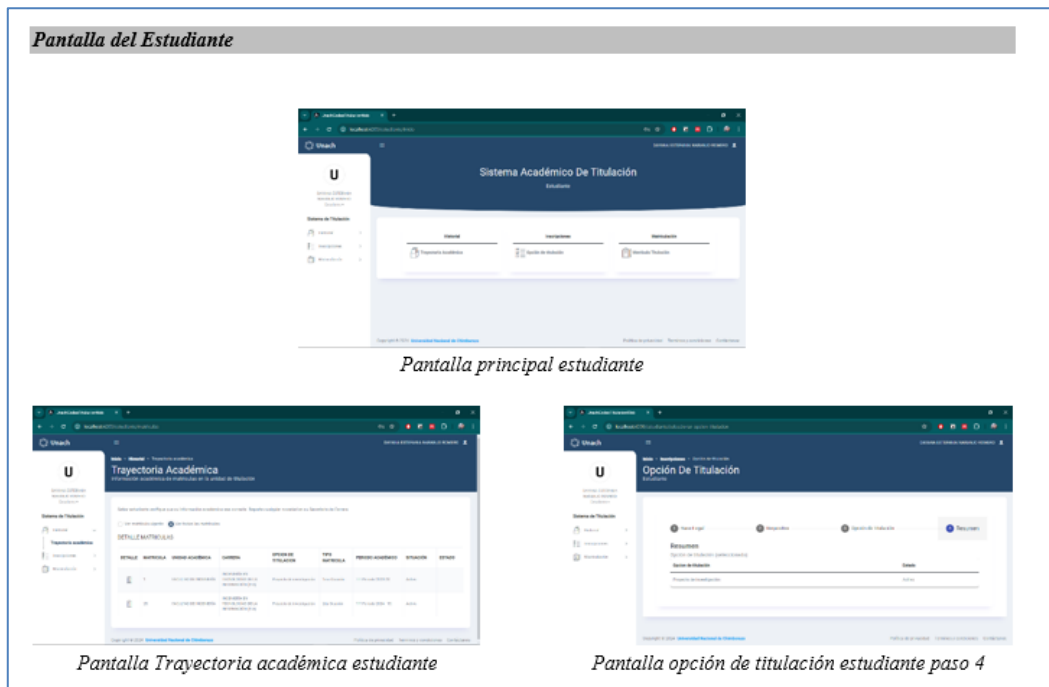
```
[HttpGet("GetNivelesCarreraByIdCarreraApi/{idCarrera}")]  
[EnableCors]  
0 references  
public async Task<IActionResult> GetNivelesCarreraByIdCarreraApi(int idCarrera)  
{  
    var nivelesCarrera = _informacionAcademicaApiClient.GetNiveles(idCarrera);  
  
    var nivelesFiltrados = nivelesCarrera  
        .Where(n => n.SistemaEstudiosID == (int)CatalogoSistemaEstudios.SEMESTRAL)  
        .Select(n => new EntidadNivelesCarreraBySistemaEstudios  
            {  
                NivelID = n.NivelID,  
                Nivel = n.Nivel  
            }).ToList();  
  
    var respuesta = new EntidadRespuesta<List<EntidadNivelesCarreraBySistemaEstudios>>()  
    {  
        Entidad = nivelesFiltrados,  
        Mensaje = "Niveles de carrera obtenidos correctamente"  
    };  
    return Ok(respuesta);  
}
```

Figura 12: GetNivelesCarreraByIdCarreraApi

### 3.7.3.3 FrontEnd

Para la implementación del frontend, se consideró la creación de componentes, servicios e instalación de librerías, entre otras configuraciones, que se detallan en el Anexo 4. Este enfoque permitió obtener una interfaz intuitiva para el usuario.

En la Figura 13, se muestra una vista de las pantallas finales que el estudiante utiliza, desarrolladas en Angular.

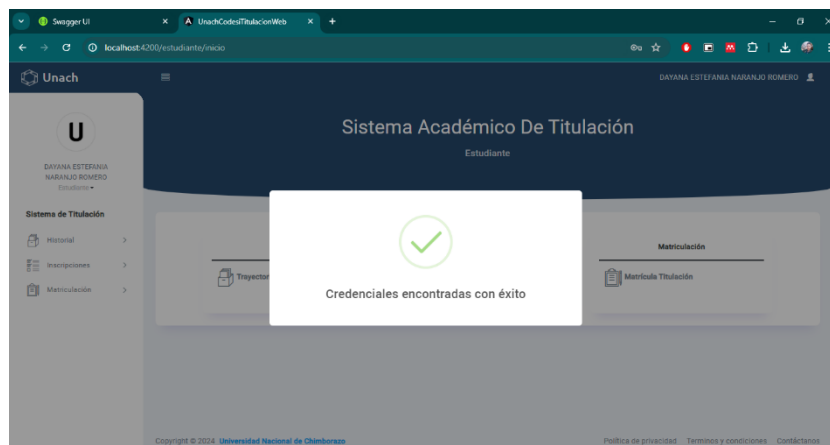


**Figura 13:** Vista del Estudiante del Sistema de Matrículas

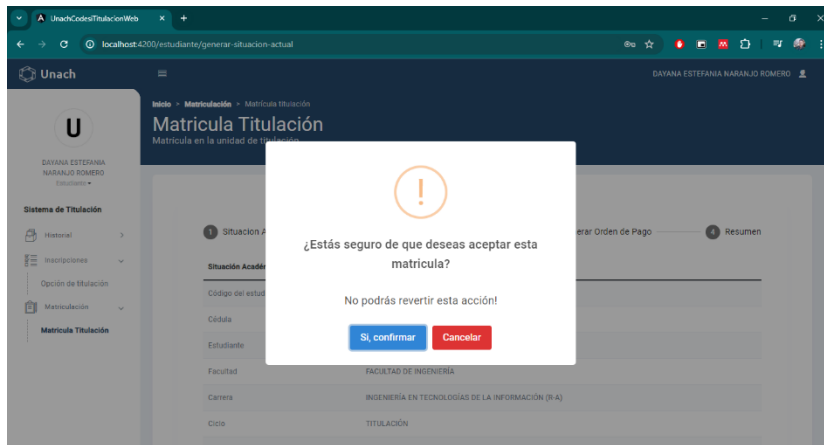
### 3.7.4 Fase de Revisión y retrospectiva

Durante esta fase, se realizaron pruebas exhaustivas para comprobar el correcto funcionamiento del sistema, asegurando que todas las funcionalidades desarrolladas cumplan con los requisitos especificados y operen sin errores.

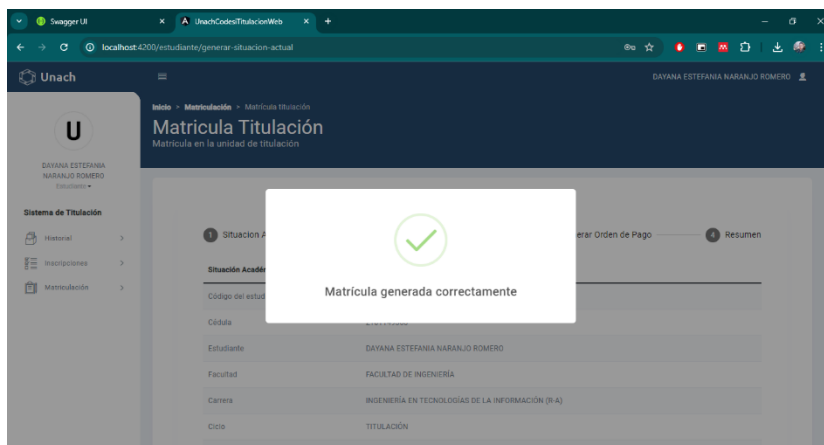
Las pruebas incluyeron casos de uso reales y simulaciones de distintos escenarios para garantizar la robustez del sistema.



**Figura 14:** Inicio de sesión correcto



**Figura 15:** Mensaje de confirmación



**Figura 16:** Matrícula generada correctamente

### 3.7.5 Fase de Lanzamiento

Durante esta fase crucial del proyecto, se llevó a cabo la presentación oficial del sistema desarrollado para la gestión del módulo de matrícula en la Universidad Nacional de Chimborazo, seguida de la firma del acta de entrega, disponible en el anexo 5.

La presentación incluyó una explicación detallada de las funcionalidades implementadas, destacando la integración de tecnologías innovadoras como LINQ y expresiones Lambda para optimizar el rendimiento y la eficiencia del sistema. Además, se completó la entrega formal del sistema completo, asegurando la inclusión de todos los archivos relevantes y la documentación esencial, incluyendo el detallado manual de usuario disponible en el anexo 6.

## CAPÍTULO IV. RESULTADOS Y DISCUSIÓN

En el estudio, se propuso evaluar el comportamiento temporal del módulo de matrícula mediante el uso de Apache JMeter. Se realizaron 1200 simulaciones divididas en 4 ciclos de pruebas de carga, cada uno con 100 peticiones durante 10 segundos para cada sistema evaluado. Esta configuración permitió obtener información detallada sobre el tiempo de respuesta y tiempo de procesamiento de ambos sistemas: uno implementado con LINQ y expresiones Lambda, y otro sin estas tecnologías al que se denominó Sistema Tradicional. Mediante JMeter, se simuló la carga de múltiples usuarios concurrentes para medir y comparar eficazmente los tiempos de respuesta y procesamiento entre los sistemas.

Para ambos escenarios evaluados, se utilizará el método POST, que facilita la transferencia y guardado de datos entre el usuario y el sistema. La primera actividad consiste en guardar el tipo de matrícula, permitiendo registrar y categorizar los distintos tipos de inscripción. La segunda actividad implica generar una nueva matrícula, operación que crea registros individuales en el sistema para nuevos usuarios o estudiantes. Ambos métodos son cruciales para la gestión eficiente y precisa de datos dentro del contexto académico evaluado.

### 4.1 Tiempo de Respuesta

#### 4.1.1 Comparativa Operación 1: Guardar Tipo de Matrícula

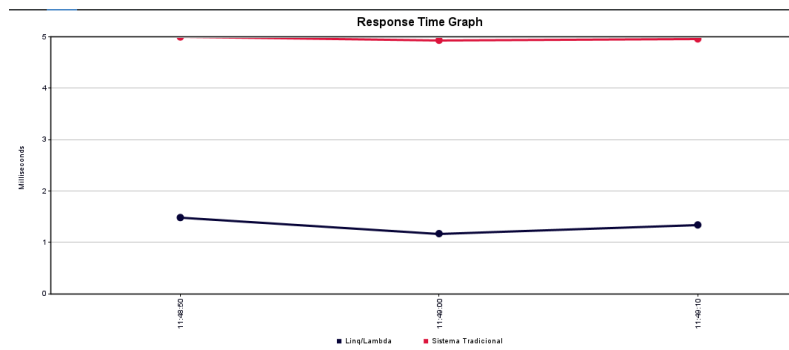
En la tabla 12 se presentan los resultados de los tiempos de respuesta obtenidos durante la evaluación de los dos sistemas. Se utilizó la configuración anteriormente mencionada, simulando el método POST de Guardar Tipo de Matrícula.

**Tabla 12:** Comparación tiempos de respuesta operación 1

	<b>Sistema con Linq y Lambda</b>	<b>Sistema Tradicional</b>
Muestras	1200	1200
Media	1	5
Mínimo	0	3
Máximo	76	165
Desviación estándar	4.41	6.99
Rendimiento(req/s)	38.3	38.3
KB/s recibidos	6.76	40.68
KB/s enviados	10.50	11.36
%Error	0%	0%

En la Figura 17, se puede apreciar de forma gráfica los tiempos de respuesta que tuvieron cada sistema en realizar la misma operación.





**Figura 17:** Gráfica de tiempos de respuesta operación 1

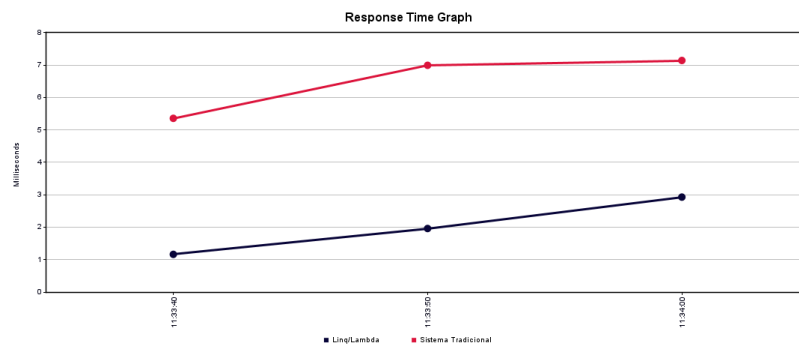
#### 4.1.2 Comparativa Operación 2: Generar Matrícula

En la tabla 13 se presentan los resultados de los tiempos de respuesta obtenidos durante la evaluación de los dos sistemas. Se empleó la misma configuración previamente mencionada, simulando el método POST de Generar Matrícula.

**Tabla 13:** Comparación tiempos de respuesta operación 2

	<b>Sistema con Linq y Lambda</b>	<b>Sistema Tradicional</b>
Muestras	1200	1200
Media	2	6
Mínimo	0	4
Máximo	124	87
Desviación estándar	6.32	2.46
Rendimiento(req/s)	38.38	38.41
KB/s recibidos	7.01	76.53
KB/s enviados	19.15	20.03
%Error	0%	0%

En la Figura 18, se puede apreciar de forma gráfica los tiempos de respuesta que tuvieron cada sistema en realizar la misma operación, pero con diferentes tecnologías.



**Figura 18:** Gráfica de tiempos de respuesta operación 2

## 4.2 Tiempo de Procesamiento

### 4.2.1 Comparativa Operación 1: Guardar Tipo de Matrícula

En la tabla 14 se presentan los resultados del tiempo de procesamiento obtenidos durante la evaluación de los dos sistemas para la operación de guardar tipo de matrícula. Se utilizó la misma configuración previamente mencionada, simulando el método POST correspondiente.

Tabla 14: Comparación tiempos de procesamiento operación 1

Sistema	SampleTime(ms)	Latency (ms)	ConnectTime (ms)	Tiempo de Procesamiento (ms)
Linq/Lambda	2,365	2,365	0.144166667	2,220833333
Sistema Tradicional	4,800833333	4,795	0.111666667	4,683333333

#### Cálculo del Tiempo de Procesamiento:

Tiempo de Procesamiento = Latency(ms) - ConnectTime(ms)

- **Linq/Lambda:**  
Tiempo de Procesamiento =  $2,365 - 0.144166667 = 2,220833333$  ms
- **Sistema Tradicional:**  
Tiempo de Procesamiento =  $4,795 - 0.111666667 = 4,683333333$  ms.

### 4.2.2 Comparativa Operación 2: Generar Matrícula

Continuando con la evaluación, en la tabla 15 se muestran los resultados del tiempo de procesamiento para la operación de generar matrícula. Se ejecutaron pruebas bajo la misma configuración, simulando el método POST correspondiente.

Tabla 15: Comparación tiempos de procesamiento operación 2

Sistema	SampleTime(ms)	Latency (ms)	ConnectTime (ms)	Tiempo de Procesamiento (ms)
Linq/Lambda	2,350833333	2,35	0.158333333	2,191666667
Sistema Tradicional	6,200833333	6,1975	0.114166667	6,083333333

#### Cálculo del Tiempo de Procesamiento:

Tiempo de Procesamiento = Latency(ms) - ConnectTime(ms)

- **Linq/Lambda:**

Tiempo de Procesamiento =  $2,35 - 0.158333333 = 2,191666667$  ms

- **Sistema Tradicional:**

Tiempo de Procesamiento =  $6,1975 - 0.114166667 = 6,083333333$  ms

### **4.3 Discusión**

En esta sección, se analizan y discuten los resultados obtenidos durante la evaluación de los tiempos de respuesta y tiempos de procesamiento del módulo de matrícula, comparando el sistema desarrollado con LINQ y expresiones Lambda frente al sistema tradicional.

#### **4.3.1 Tiempos de respuesta**

Durante las pruebas de carga simuladas, se observaron tiempos de respuesta significativamente diferentes entre el sistema basado en LINQ y Lambda y el sistema tradicional. El sistema LINQ/Lambda mostró tiempos de respuesta más bajos en ambos escenarios evaluados (Guardar Tipo de Matrícula y Generar Nueva Matrícula), con medias de 2 ms y 2,35 ms respectivamente, en comparación con el sistema tradicional que registró medias de 4,8 ms y 6,2 ms para las mismas operaciones. Esta diferencia puede atribuirse a la optimización y eficiencia que proporcionan las consultas LINQ y Lambda en la manipulación de datos en comparación con los métodos tradicionales utilizados en el sistema que no utiliza estas tecnologías.

#### **4.3.2 Tiempos de procesamiento**

En cuanto a los tiempos de procesamiento, que se calculan como la diferencia entre la latencia y el tiempo de conexión, también se observaron diferencias marcadas. El sistema LINQ/Lambda mostró tiempos de procesamiento promedio de 2,191666667 ms y 2,35 ms para las operaciones evaluadas, mientras que el sistema tradicional tuvo tiempos promedio de 6,083333333 ms y 6,1975 ms. Esto sugiere una eficiencia superior en el manejo y procesamiento de las operaciones de matrícula en el sistema desarrollado con LINQ y Lambda.

## **CAPÍTULO V. CONCLUSIONES y RECOMENDACIONES**

### **CONCLUSIONES**

- El análisis de Linq y expresiones Lambda en el desarrollo de sistemas confirma su utilidad para manipular datos con eficiencia, mejorar la legibilidad del código y crear repositorios genéricos. Estas habilidades optimizan aplicaciones, disminuyen costos y recursos, garantizando un software sólido y mantenible.
- El desarrollo del módulo de matrícula para el sistema de titulación de la Universidad Nacional de Chimborazo utilizando LINQ y expresiones Lambda ha sido fundamental debido a su capacidad para manejar y manipular datos de manera ágil y estructurada. Estas tecnologías han optimizado significativamente la gestión de información y los procesos relacionados con la matrícula de estudiantes. Además, han permitido la creación de un software robusto y fácilmente mantenible, mejorando la eficiencia en el manejo de datos y la legibilidad del código, esenciales para desarrollar sistemas modernos y eficaces.
- La evaluación del comportamiento temporal del módulo de matrícula utilizando la norma ISO 25010 ha revelado mejoras significativas tanto en los tiempos de respuesta como en el tiempo de procesamiento. Con el uso de LINQ y expresiones Lambda, se logró reducir los tiempos promedio de respuesta a aproximadamente 1 milisegundo para la operación de Guardar Tipo Matrícula y 2 milisegundos para la operación de Generar Matrícula. Además, se observó una mejora en el tiempo de procesamiento, destacando la eficiencia mejorada que estas tecnologías ofrecen en comparación con métodos tradicionales. Estos resultados validan su efectividad en la optimización del rendimiento del sistema en términos de tiempos de respuesta y procesamiento.

## RECOMENDACIONES

- Mantenerse al tanto de las actualizaciones y nuevas funcionalidades en LINQ y expresiones Lambda mediante la investigación constante. Esto permitirá aprovechar al máximo estas herramientas en el desarrollo de sistemas, mejorando la eficiencia y la calidad del software.
- Capacitar de forma continua al equipo de desarrollo en las mejores prácticas y actualizaciones relacionadas con LINQ y expresiones Lambda, asegurando así un mantenimiento óptimo y adaptación a futuras necesidades del sistema.
- Se recomienda evaluar y seleccionar las tecnologías más adecuadas según la naturaleza y requerimientos específicos del proyecto o sistema a desarrollar. Considerar aspectos como la escalabilidad, la integración con otras tecnologías existentes, y el soporte de la comunidad y actualizaciones continuas son clave para el éxito a largo plazo.

## BIBLIOGRAFÍA

- Amaya Lozano, E. D., & Juez Candell, C. S. (2016). *Análisis, diseño, desarrollo e implementación de un sistema de control para registros y cobro de matrícula y pensiones para la Unidad Educativa Particular Mixta Mercedes de Jesús Molina Mediante un aplicativo Web.* dspace.ups.edu.ec: <https://dspace.ups.edu.ec/bitstream/123456789/12298/1/UPS-GT001626.pdf>
- Apache Software Foundation. (2023). *Apache JMeter™.* jmeter.apache.org: <https://jmeter.apache.org/>
- Armas Navarrete, J. A. (2019). *Desarrollo de un sistema de gestión de datos para el monitoreo integral de glaciares.* bibdigital.epn.edu.ec: <https://bibdigital.epn.edu.ec/handle/15000/20360>
- Cloudflare. (2024). *¿Qué es una API?* www.cloudflare.com: <https://www.cloudflare.com/es-es/learning/security/api/what-is-an-api/>
- Contreras Sánchez, C. I., & Suáres Jácome, R. (2022). *SISTEMA WEB PARA EL REGISTRO Y SEGUIMIENTO DE LOS PROYECTOS ACTIVOS DEL DEPARTAMENTO DE VINCULACIÓN CON LA SOCIEDAD EN LA CARRERA DE INGENIERIA DE SOFTWARE EN LA UNIVERSIDAD DE GUAYAQUIL.* repositorio.ug.edu.ec: <http://repositorio.ug.edu.ec/handle/redug/65254>
- Deyimar, A. (11 de 01 de 2023). *¿Qué es Angular y cuáles son sus ventajas?* www.hostinger.e: <https://www.hostinger.es/tutoriales/que-es-angular>
- Escuela de Programación en Madrid [AEPI]. (03 de Junio de 2020). *¿Qué es LINQ?* asociacionaepi.es: <https://asociacionaepi.es/que-es-linq/>
- FasterCapital. (2024). *Beneficios De Utilizar Expresiones Lambda.* fastercapital.com: <https://fastercapital.com/es/tema/beneficios-de-utilizar-expresiones-lambda.html#:~:text=Las%20expresiones%20Lambda%20promueven%20la,y%20la%20mantenibilidad%20del%20c%C3%B3digo>
- Freire, A. (2022). *.NET 7 YA EN NUESTROS SERVIDORES. ¡CONOCE SUS NOVEDADES!* dinahosting.com: [https://dinahosting.com/blog/novedades-net-7-0/#Que\\_es\\_NET](https://dinahosting.com/blog/novedades-net-7-0/#Que_es_NET)
- Godinez, H. G. (06 de 03 de 2024). *Linq to dataset permite realizar consultas sobre.* www.coursehero.com: <https://www.coursehero.com/file/p6o90mmt/LINQ-to-DataSet-Permite-realizar-consultas-sobre-conjuntos-de-datos-ADONET/>

- IONOS. (2023). *Conceptos básicos: definición de web app y ejemplos*. www.ionos.es:  
<https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-una-web-app-y-que-clases-hay/>
- ISO 25000. (2022). *ISO/IEC 25010*. iso25000.com:  
<https://iso25000.com/index.php/normas-iso-25000/iso-25010>
- Jaimes Nava, O. (04 de 2023). *Diseño de APIs para la lectura de variables empleando dispositivos de IoT*. reini.utcv.edu.mx:  
<http://reini.utcv.edu.mx/bitstream/123456789/1431/1/1%20-%2020193L00101.pdf#page=19&zoom=100,109,212>
- Lema Romero, C. X., & Hernandez Castillo, V. S. (2018). *Sistema web de gestión de matriculación y notas para la escuela "Pan y Vida"*. dspace.ups.edu.ec:  
<https://dspace.ups.edu.ec/bitstream/123456789/16114/1/UPS-GT002316.pdf>
- López Gonzáles, J. (2020). *Diseño y formalización de lenguajes de consultas inspirados en ópticas*. burjcdigital.urjc.es:  
<https://burjcdigital.urjc.es/bitstream/handle/10115/17540/thesis.pdf?sequence=1&isAllowed=y>
- Microsoft. (09 de Septiembre de 2022). *Tutorial: Escritura de consultas en C# (LINQ)*. learn.microsoft.com:  
<https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/walkthrough-writing-queries-linq>
- Microsoft. (2023). *Expresiones lambda y funciones anónimas*. learn.microsoft.com:  
<https://learn.microsoft.com/es-es/dotnet/csharp/language-reference/operators/lambda-expressions>
- Microsoft. (2023). *Language Integrated Query (LINQ) (C#)*. learn.microsoft.com:  
<https://learn.microsoft.com/es-es/dotnet/csharp/programming-guide/concepts/linq/>
- Microsoft. (12 de Marzo de 2023). *Novedades de .NET 7*. learn.microsoft.com:  
<https://learn.microsoft.com/es-es/dotnet/core/whats-new/dotnet-7>
- Microsoft. (11 de 04 de 2024). *Language-Integrated Query (LINQ)*. learn.microsoft.com:  
<https://learn.microsoft.com/es-es/dotnet/csharp/linq/>
- Molina Ríos, J. R., Zea Ordóñez, M. P., Contento Segarra, M. J., & García Zerda, F. G. (2017). *Metodologías de desarrollo en aplicaciones web*. dialnet.unirioja.es:  
<https://dialnet.unirioja.es/servlet/articulo?codigo=6143045>

- Parinango, J., & Alfredo, E. (2022). *Análisis de los sistemas de gestión de base de datos relacionales con marcos de trabajo para procesamiento de datos masivos*. repositorio.uss.edu.pe: <https://repositorio.uss.edu.pe/handle/20.500.12802/10234>
- Sáez Hurtado, J. (2021). *Cómo funciona la Metodología Scrum: Qué es y cómo utilizarla*. www.iebschool.com: <https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/>
- Santos Garrido, J. (2020). *Desarrollo de sistema para caracterizar ruidos urbanos y transformarlos a otro tipo de sonidos agradables al oído humano*. riuma.uma.es: <https://riuma.uma.es/xmlui/bitstream/handle/10630/19181/SantosgarridojoseMemoria.pdf?sequence=1&isAllowed=y>
- Turrado, J. (2019). *Introducción rápida a LINQ con C#: manejar información en memoria nunca fue tan sencillo*. www.campusmvp.es: <https://www.campusmvp.es/recursos/post/introduccion-rapida-a-linq-con-c-sharp.aspx>
- Valdivieso, Y. (19 de Febrero de 2019). *¿Por qué es importante el Registro de Matrícula para la prestación del servicio educativo?* www.cali.gov.co: <https://www.cali.gov.co/educacion/publicaciones/145981/por-que-es-importante-el-registro-de-matricula-para-la-prestacion-del-servicio-educativo/>
- Zea Ordóñez, M. P., Molina Ríos, J. R., Contento Segarra, M. J., & García Zerda, F. G. (2018). *Comparación de metodologías en aplicaciones web*. dialnet.unirioja.es: <https://dialnet.unirioja.es/servlet/articulo?codigo=6415697>

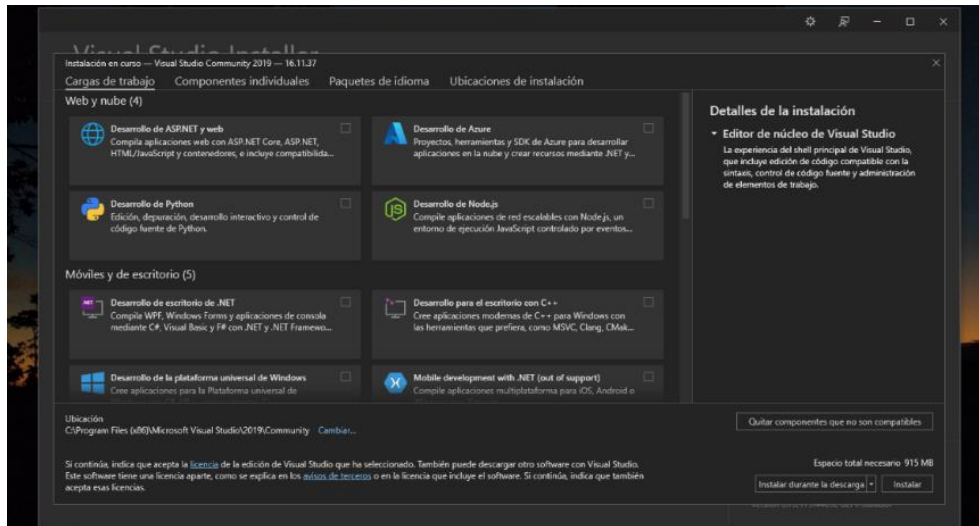


## **ANEXOS**

# **ANEXO 1. INSTALACION DE HERRAMIENTAS**

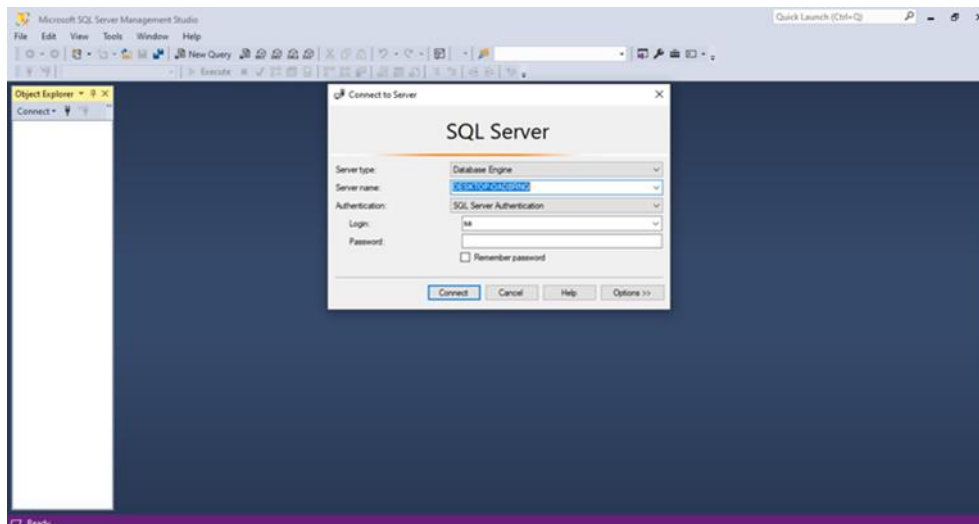
## Backend: .NET con Visual Studio Community

Para implementar el código backend del módulo de matrícula del sistema de titulación, se utilizó Visual Studio Community de Microsoft. Esta herramienta proporciona un entorno robusto y flexible para el desarrollo de aplicaciones .NET. La Figura 6 ilustra el proceso de instalación y configuración de Visual Studio 2022, esencial para el desarrollo de la lógica de negocio y la integración con el frontend.



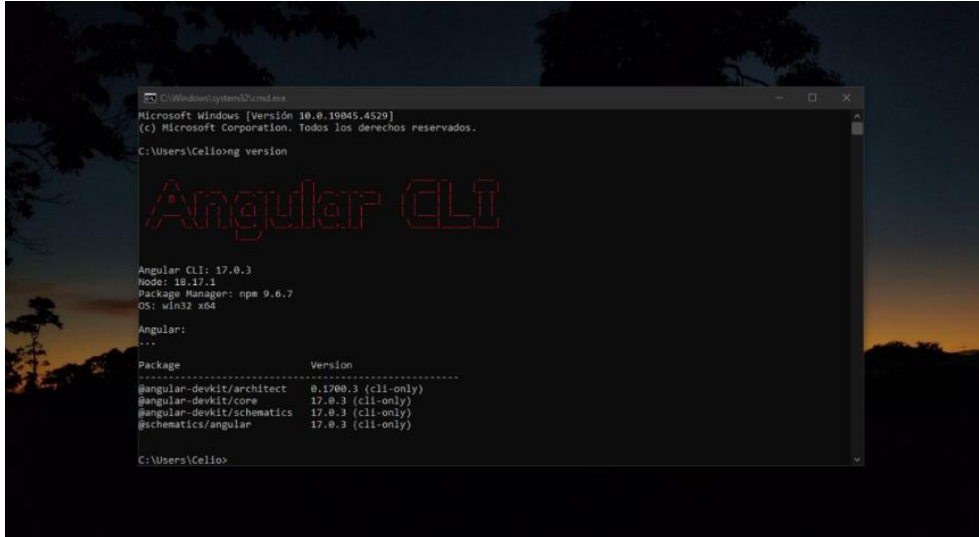
## Base de Datos: SQL Server

Para la creación y gestión de la base de datos del sistema de matrícula, se optó por SQL Server debido a su capacidad robusta y escalable para manejar grandes volúmenes de datos. La Figura 7 detalla el proceso de instalación y configuración de SQL Server, fundamental para asegurar la integridad y eficiencia de los datos del sistema.



## Frontend: Angular

El desarrollo del frontend del sistema de matrícula se llevó a cabo utilizando Angular, un framework de JavaScript/TypeScript ampliamente reconocido por su capacidad de construir interfaces de usuario dinámicas y eficientes. La Figura 8 muestra la versión de Angular instalado para la creación del sistema.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19045.4529]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Celio>ng version

Angular CLI
Angular CLI: 17.0.3
Node: 18.17.1
Package Manager: npm 9.6.7
OS: win32 x64

Angular:
+
-----
Package                                  Version
-----
@angular-devkit/architect                0.1700.3 (cli-only)
@angular-devkit/core                     17.0.3 (cli-only)
@angular-devkit/schematics               17.0.3 (cli-only)
@schematics/angular                     17.0.3 (cli-only)

C:\Users\Celio>
```

## **ANEXO 2. ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE**



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

N.1



**ESPECIFICACIONES DE  
REQUISITOS DEL SOFTWARE**  
<Módulo de Matrícula Unidad de Titulación>

Versión 1.0



## Control del Documento

TÍTULO: MÓDULO DE MATRÍCULA UNIDAD DE TITULACIÓN

VERSIÓN: 1.0

CÓDIGO DEL FORMATO:

DEPENDENCIA: CODESI

## Firmas y Aprobaciones

ELABORADO POR: [Cepeda Pataron Ligia Adriana]

[Estudiante]

FECHA: [2024-07-12]

Firma: \_\_\_\_\_

[Paguay Quiroz Celio Gustavo]

[Estudiante]

FECHA: [2024-07-12]

Firma: \_\_\_\_\_

REVISADO POR: [Henry Javier Paca Quinaluiza]

[Encargado de sistemas informáticos]

FECHA: [2024-07-12]

Firma: \_\_\_\_\_

APROBADO POR: [José Rafael Salguero Rosero]

[Director de dirección Académica]

FECHA: [2024-07-12]

Firma: \_\_\_\_\_

## Lista de Cambios

VERSIÓN	FECHA	AUTOR	DESCRIPCIÓN
1.0	2024-07-12	Cepeda Pataron Ligia Adriana, Paguay Quiroz Celio Gustavo	Emisión Inicial



### ESPECIFICACIONES DE REQUISITOS DE SOFTWARE

<b>Sistema de Información:</b>	<b>Titulación</b>
<b>Módulo /Funcionalidad:</b>	<b>Matriculación</b>
<b>Unidad Requirente:</b>	<b>CODESI</b>
<b>Titular de la unidad:</b>	<b>Henry Javier Paca Quinaluiza</b>

Nombre de requisito: <ESTUDIANTE>					
No	Prioridad	Como	Necesito	Para	Criterios de aceptación
1	ALTA	Estudiante	Registrarse en el sistema	Ingresar con una cuenta válida al sistema	Los estudiantes deben registrarse para poder ingresar al sistema.
2	ALTA	Estudiante	Ingresar al sistema	Postular a una opción de titulación	El estudiante puede postular a una opción de titulación, aceptando los términos requeridos.
3	ALTA	Estudiante	Seleccionar opción de titulación	Seleccionar una opción de titulación disponible para su carrera	El estudiante puede seleccionar una opción de titulación disponible para su carrera.
4	ALTA	Estudiante	Ingresar al apartado de matriculación	Verificar la situación actual del estudiante en su proceso de titulación	El sistema verifica la situación actual del estudiante en su proceso de titulación.
5	ALTA	Estudiante	Ver matrículas en titulación	Ver sus matrículas actuales y futuras en el sistema	El estudiante puede ver sus matrículas actuales y futuras en el sistema.
6	ALTA	Estudiante	Aceptar nueva matrícula	Aceptar la matrícula generada para el periodo en curso	El estudiante puede aceptar la matrícula generada para el periodo en curso.

Nombre de requisito: <SECRETARIA>					
No	Prioridad	Como	Necesito	Para	Criterios de aceptación
1	ALTA	Secretaria de carrera	Realizar matrícula de estudiantes	Procesar matrícula para estudiantes del último nivel de titulación y generar reporte.	Realizar la matrícula de estudiantes del último nivel de la carrera como primera ocasión y obtener un reporte/listado.





Nombre de requisito: <RESPONSABLE DE CARRERA>					
No	Prioridad	Como	Necesito	Para	Criterios de aceptación
1	ALTA	Responsable de carrera	Ver configuraciones de carrera	Visualizar las configuraciones de carrera realizadas	Mostrar todas las configuraciones de carrera realizadas.
2	ALTA	Responsable de carrera	Guardar configuración de carrera	Guardar nueva configuración de carrera incluyendo detalles específicos como semestres.	Guardar nueva configuración de carrera incluyendo detalles específicos como semestres.
3	ALTA	Responsable de carrera	Ver opciones de titulación de carrera	Mostrar todas las opciones de titulación disponibles para la carrera en cierto periodo.	Mostrar todas las opciones de titulación disponibles para la carrera en cierto periodo.
4	ALTA	Responsable de carrera	Guardar opciones de titulación	Registrar nuevas opciones de titulación asociadas al periodo y configuración de carrera.	Registrar nuevas opciones de titulación asociadas al periodo y configuración de carrera.
5	ALTA	Responsable de carrera	Ver secuencia de periodos de la carrera	Visualizar histórico de periodos de la carrera	Visualizar histórico de periodos de la carrera.
6	ALTA	Responsable de carrera	Guardar secuencia de periodos	Almacenar nueva secuencia de periodos incluyendo periodo actual y anterior.	Almacenar nueva secuencia de periodos incluyendo periodo actual y anterior.



Nombre de requisito: <ADMINISTRADOR>					
No	Prioridad	Como	Necesito	Para	Criterios de aceptación
1	ALTA	Administrador	Ver opciones de titulación	Mostrar todas las opciones de titulación disponibles en la universidad.	Mostrar todas las opciones de titulación disponibles en la universidad.
2	ALTA	Administrador	Guardar opciones de titulación	Registrar nuevas opciones de titulación en la universidad.	Registrar nuevas opciones de titulación en la universidad.
3	ALTA	Administrador	Ver tipos de matrícula de titulación	Mostrar los diferentes tipos de matrícula disponibles y sus aranceles.	Mostrar los diferentes tipos de matrícula disponibles y sus aranceles.
4	ALTA	Administrador	Guardar tipos de matrícula de titulación	Agregar nuevos tipos de matrícula con sus respectivos aranceles.	Agregar nuevos tipos de matrícula con sus respectivos aranceles.
5	ALTA	Administrador	Ver periodos de matrícula de titulación	Visualizar historial de periodos de matrícula de titulación en la universidad.	Visualizar historial de periodos de matrícula de titulación en la universidad.
6	ALTA	Administrador	Guardar periodos de matrícula de titulación	Almacenar nuevos periodos de matrícula con fechas de inicio y fin especificadas.	Almacenar nuevos periodos de matrícula con fechas de inicio y fin especificadas.
7	ALTA	Administrador	Ver responsables de titulación	Mostrar lista de responsables actuales y pasados de titulación en la universidad.	Mostrar lista de responsables actuales y pasados de titulación en la universidad.
8	ALTA	Administrador	Guardar responsables de titulación	Registrar nuevos responsables de titulación con detalles de periodo y cargo específicos.	Registrar nuevos responsables de titulación con detalles de periodo y cargo específicos.



Nombre de requisito: <LISTA DE ESTUDIANTES>					
No	Prioridad	Como	Necesito	Para	Criterios de aceptación
1	ALTA	Todos los usuarios excepto estudiantes	Ver listado de estudiantes	Acceder a una lista completa de estudiantes que se encuentran en proceso de titulación, junto con sus matrículas y opciones de titulación seleccionadas.	Se muestra un listado ordenado por opciones de titulación, tipo de matrícula, etc., con información precisa sobre cada estudiante.

## **ANEXO 3. CONFIGURACIÓN BACKEND**



```

1 using Microsoft.EntityFrameworkCore;
2 using Unach.Codexi.Titulacion.Api.Persistencia.Api.Core.AcademicoApi;
3 using Unach.Codexi.Titulacion.Api.Persistencia.Api.Core.Sicoa.InformacionAcademica;
4 using Unach.Codexi.Titulacion.Api.Persistencia.Api.Core.Models;
5 using Unach.Codexi.Titulacion.Mappings;
6
7 var builder = WebApplication.CreateBuilder(args);
8
9 // Add services to the container.
10
11 builder.Services.AddControllers();
12 // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
13 builder.Services.AddEndpointsApiExplorer();
14 builder.Services.AddSwaggerGen();
15
16 //mapper-automapping
17
18 #region Automapper
19
20 var mappingConfig = new MapperConfiguration(mc =>
21 {
22     mc.AddProfile(new PerfilMappings());
23 });
24
25 IMapper mapper = mappingConfig.CreateMapper();
26

```

Program.cs

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Unach.Codexi.Titulacion.Api.Aplicacion.DTOs
8 {
9     public class EntidadLoginAcceso
10     {
11         int IdUsuario { get; set; }
12         string Nombre { get; set; }
13         string ApellidoPaterno { get; set; }
14         string ApellidoMaterno { get; set; }
15         string DocumentoIdentidad { get; set; }
16         string CorreoInstitucional { get; set; }
17         string FotoUrl { get; set; }
18         public List<EntidadMolUsuario> IdMolUsuario { get; set; }
19     }
20 }

```

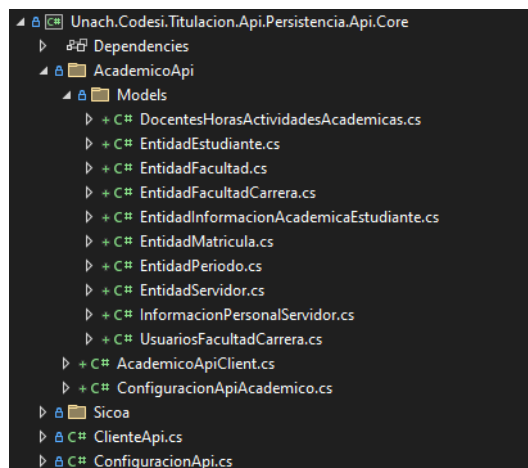
DTO's

```

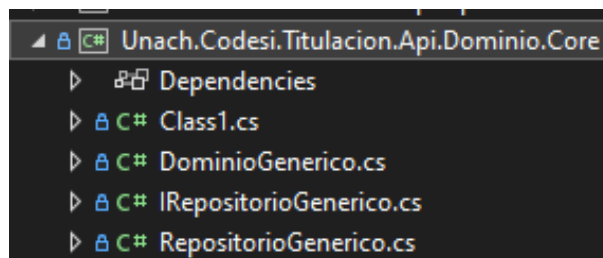
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Unach.Codexi.Titulacion.Api.Entidades
8 {
9     public enum CatalogoFacultades
10     {
11         CIENCIAS_DE_LA_EDUCACION=1,
12         INGENIERIA=2,
13         CIENCIAS_DE_LA_SALUD=3,
14         CIENCIAS_POLITICAS=4
15     }
16 }

```

Creación de catálogos



Persistencia.Api.Core



Dominio.Core

```

[HttpPost("ListaResponsablesVigentes")]
[EnableCors]
public async Task<IActionResult> ListaResponsablesVigentes(EntidadListarResponsables listado)
{
    // Obtener facultades
    var facultades = _academicoApiClient.GetFacultades()
        .Where(f => f.UnidadAcademicaId == (int)CatalogoUnidadAcademica.FACULTADES).ToList();

    // Obtener carreras filtradas por facultades especificas
    var carreras = _academicoApiClient.GetEntidadFacultadCarrera()
        .Where(x => x.IdTipoCarrera == (int)CatalogoTipoCarrera.VIGENTE &&
            (x.IdFacultad == (int)CatalogoFacultades.CIENCIAS_DE_LA_SALUD ||
            x.IdFacultad == (int)CatalogoFacultades.CIENCIAS_DE_LA_EDUCACION ||
            x.IdFacultad == (int)CatalogoFacultades.CIENCIAS_POLITICAS ||
            x.IdFacultad == (int)CatalogoFacultades.INGENIERIA)).ToList();

    // Obtener todos los usuarios
    var usuarios = _dominioGenerico.GetRepositorio<Usuario>().ObtenerTodos();

    // Obtener roles de usuario filtrados por idPeriodo e idCargo
    var rolesUsuario = _dominioGenerico.GetRepositorio<RolUsuario>()
        .ObtenerTodos()
        .Where(ru => ru.IdRol == listado.IdTipoRol
            && ru.IdPeriodo == listado.IdPeriodo
            && ru.Estado == "VIGENTE"
        ).ToList(); // Assuming idPeriodo is not relevant in the role filter
}

```

Creación de endpoints

## **ANEXO 4. CONFIGURACIÓN FRONTEND**



```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <title>UnachCodesTribuconApiWeb</title>
5 <base href="/" />
6 <meta name="viewport" content="width=device-width, initial-scale=1" />
7 <link rel="icon" type="image/x-icon" href="favicon.ico" />
8
9 <!-- Fonts [ OPTIONAL ] -->
10 <link rel="preconnect" href="https://fonts.googleapis.com">
11 <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
12 <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&family=Roboto:wght@400;500" rel="stylesheet">
13 <link rel="stylesheet" href="/assets/css/icons/premium-line-icons.css">
14 <link rel="stylesheet" href="/assets/css/icons/premium-solid-icons.css">
15
16 <!-- Bootstrap CSS [ REQUIRED ] -->
17 <link rel="stylesheet" href="/assets/css/bootstrap.min.css">
18
19 <!-- Nifty CSS [ REQUIRED ] -->
20 <link rel="stylesheet" href="/assets/css/nifty.min.css">
21
22 <!-- Nifty Demo Icons [ OPTIONAL ] -->
23 <link rel="stylesheet" href="/assets/css/demo-purpose/demo-icons.min.css">
24
25 <!-- Demo purpose CSS [ DEMO ] -->
26 <link rel="stylesheet" href="/assets/css/demo-purpose/demo-settings.min.css">
27
28 <!-- Materialpicken [ OPTIONAL ] -->
29 <link rel="stylesheet" href="/assets/vendors/mc-datapicker/mc-calendar.min.css">
30
31

```

## Configuración Index Global

```

1 // You can add global styles to this file, and also import other style files
2 @import "ng-select/ng-select/themes/default.theme.css";
3 @import "@angular/material/prebuilt-themes/indigo-pink.css";
4
5
6 .mat-mdc-form-field {
7   font-size: 15px;
8   width: 100%;
9 }
10
11 .add {
12   font-size: 14px;
13 }
14
15 .mat-mdc-option-mdc-list-item {
16   font-size: 15px;
17 }
18
19 .mdc-text-field--filled:not(.mdc-text-field--disabled) {
20   background-color: #ffff;
21   height: 50px;
22 }
23
24 html,
25 body {
26   height: 100%;
27 }
28

```

## Configuración CSS Global

```

1 {
2   "name": "unach.codes.tribucon.api.web",
3   "version": "0.0.0",
4   "scripts": {
5     "ng": "ng",
6     "start": "ng serve",
7     "build": "ng build",
8     "watch": "ng build --watch --configuration development",
9     "test": "ng test"
10  },
11  "private": true,
12  "dependencies": {
13    "@angular/animations": "~10.2.0",
14    "@angular/cdk": "~11.3.0",
15    "@angular/common": "~10.2.0",
16    "@angular/compiler": "~10.2.0",
17    "@angular/core": "~10.2.0",
18    "@angular/forms": "~10.2.0",
19    "@angular/material": "~10.2.0",
20    "@angular/platform-browser": "~10.2.0",
21    "@angular/platform-browser-dynamic": "~10.2.0",
22    "@angular/router": "~10.2.0",
23    "@ng-select/ng-select": "~10.0.0",
24    "@popperjs/core": "~2.11.0",
25    "@sweetalert2/sweetalert2": "~11.3.0",
26    "rxjs": "~7.2.0",
27    "tslib": "~2.0.0",
28    "zone.js": "~0.11.4"
29  }
30 }

```

## package.json

```

src > app > administrador > administrador.modules.ts
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { FooterModule } from 'src/app/estructura/footer/footer.module';
4 import { AdministradorComponent } from './administrador.component';
5
6 @NgModule({
7   declarations: [
8     AdministradorComponent
9   ],
10  imports: [
11    CommonModule,
12    AdministradorRoutingModule,
13    FooterModule
14  ],
15  exports: [
16    AdministradorComponent
17  ]
18 })
19 export class AdministradorModule {}

```

Creación de componentes

```

src > app > servicio > administrador > administrador.servicios.ts
1 import { Injectable } from '@angular/core';
2 import { PeticionesService } from 'src/app/comun/peticiones';
3 import { HttpClient } from '@angular/common/http';
4 import { EntidadResposta } from 'src/app/comun/entidadResposta';
5 import { EntidadInformacionServidor } from './entidadInformacionServidor';
6 import { EntidadListaResponsables } from './entidadListaResponsables';
7
8 @Injectable({
9   providedIn: 'root'
10 })
11 export class AdministradorService {
12   constructor(private peticones: PeticionesService, private http: HttpClient) {}
13
14   getDecanosApi() {
15     return this.peticiones.ejecutarQueryGet<EntidadResposta>('Administrador/ObtenerDecanos');
16   }
17
18   getDirectoresApi() {
19     return this.peticiones.ejecutarQueryGet<EntidadResposta>('Administrador/ObtenerDirectoresCarrera');
20   }
21
22   getResponsableCarreraApi() {
23     return this.peticiones.ejecutarQueryGet<EntidadResposta>('Administrador/ObtenerResponsablesTitulacion');
24   }
25
26   listaResponsablesVigentes(componentes: EntidadListaResponsables) {
27     let params = { '3506': Stringify(componentes) };
28   }
29 }

```

Creación de servicios

```

src > app > app.routing.modules.ts
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { AppComponent } from './app.component';
4 import { LoginComponent } from './login/login.component';
5 import { EstudianteGuard } from './servicio/guards/estudiante/estudiante.guard';
6
7 const routes: Routes = [
8   {
9     path: '',
10    component: LoginComponent, //primera pantalla que se ve
11  },
12  {
13    path: 'login',
14    loadChildren: () => import('./login/login.module').then(m => m.LoginModule)
15  },
16  {
17    path: 'registro',
18    loadChildren: () => import('./registro/registro.module').then(m => m.RegistroModule)
19  },
20  {
21    path: 'estructura',
22    loadChildren: () => import('./estructura/estructura/estructura.module').then(m => m.EstructuraModule),
23  },
24  {
25    path: 'administrador',
26    loadChildren: () => import('./administrador/administrador/administrador.module').then(m => m.AdministradorModule)
27  },
28 ];

```

Ruteo de componentes

## **ANEXO 5. ACTA DE ENTREGA**



## ACTA DE ENTREGA, SATISFACCIÓN Y RECEPCIÓN DEFINITIVA

El día 15 de julio de 2024 comparece: por parte de Dirección Académica de la Universidad Nacional de Chimborazo el Lcdo. José Rafael Salguero Rosero, como director de la Unidad de Dirección Académica de la Universidad Nacional de Chimborazo, el Ing. Henry Javier Paca Quinaluiza, como encargado de la gestión de desarrollo de sistemas informáticos, el Ing. Stteffano Israel Aguayo Cáceres, como encargado de la revisión del sistema informático y por parte de la carrera de Tecnologías de la Información la Ing. Ana Elizabeth Congacha Aushay DOCENTE TUTOR DEL TEMA DE TESIS “Módulo de matrícula para el sistema de titulación de la Universidad Nacional de Chimborazo utilizando Linq y expresiones Lambda”, con la finalidad de realizar el Acta de entrega, satisfacción y recepción definitiva del módulo de matrícula realizado por los estudiantes: Ligia Adriana Cepeda Pataron y Celio Gustavo Paguay Quiroz. Según los detalles:

### **PRIMERA: ANTECEDENTES**

Mediante la solicitud de Dirección Académica y la carrera de Tecnologías de la Información, las partes acuerdan desarrollar el módulo de matrícula para el sistema de titulación de la Universidad Nacional de Chimborazo utilizando Linq y expresiones Lambda. Para el desarrollo del proyecto participan: Lcdo. José Rafael Salguero Rosero director de Dirección Académica, Ing. Henry Javier Paca Quinaluiza encargado de la gestión de desarrollo de sistemas informáticos, el Ing. Stteffano Israel Aguayo Cáceres, como encargado de la revisión del sistema informático y en calidad de docente tutor el Ing. Ana Elizabeth Congacha Aushay junto a los estudiantes: Ligia Adriana Cepeda Pataron y Celio Gustavo Paguay Quiroz.

Mediante la resolución No 260-CITI-2023 de parte de la Universidad Nacional de Chimborazo se aprueba el perfil de tesis “Módulo de matrícula para el sistema de titulación de la Universidad Nacional de Chimborazo utilizando Linq y expresiones Lambda”

### **SEGUNDA: PRODUCTOS ENTREGADOS**

La carrera de Tecnologías de la Información con relación al proyecto de tesis “Módulo de matrícula para el sistema de titulación de la Universidad Nacional de Chimborazo utilizando Linq y expresiones Lambda” entrega lo siguiente:

ITEMS	CANTIDAD	DETALLE	CONTENIDO
1	1	Módulo de matrícula para el sistema de titulación	Contiene el sistema del módulo de matrícula para el sistema de titulación de la Universidad Nacional de Chimborazo

1	1	Manual de usuario	Contiene una guía para el manejo del Sistema del Módulo de matrícula.
---	---	-------------------	---

#### TERCERA: RECEPCIÓN Y CONFORMIDAD

Previa a la suscripción de la presente Acta, el representante de Dirección Académica de la Universidad Nacional de Chimborazo recibe a entera satisfacción los productos detallados en la segunda cláusula, sobre todo teniendo en cuenta la conformidad con los contenidos entregados dado el funcionamiento y originalidad de cada uno de ellos y también siguiendo los parámetros de evaluación del producto principal (Módulo de matrícula para el sistema de titulación de la Universidad Nacional de Chimborazo utilizando Linq y expresiones Lambda).

#### CUARTA: CAPACITACION

La capacitación esta desarrollada en base a los temas, periodos de tiempo y participantes detallados a continuación:

Nº	TEMA	DETALLE	PARTICIPANTES	HORARIO
1	Socialización del módulo de matrícula para el sistema de titulación	Se trata aspectos como: <ul style="list-style-type: none"> <li>• Funcionamiento</li> <li>• Roles</li> </ul>	Participan: <ul style="list-style-type: none"> <li>• Director de la unidad de Dirección Académica</li> <li>• Encargado de la gestión de desarrollo del sistema</li> <li>• Encargado de la revisión del sistema</li> <li>• Estudiantes</li> </ul>	Fecha: 15/07/2024 Hora: 12:00

En constancia a lo anterior mencionado firman los presentes:

\_\_\_\_\_  
Lcdo. José Rafael Salguero Rosero  
Director de Dirección Académica

\_\_\_\_\_  
Ing. Henry Javier Paca Quinaluiza  
Gestión de sistemas informáticos

\_\_\_\_\_  
Ing. Stteffano Israel Aguayo Cáceres  
Revisión de sistemas informáticos

\_\_\_\_\_  
Ing. Ana Elizabeth Congacha Aushay  
Docente Tutor

\_\_\_\_\_  
Ligia Adriana Cepeda Pataron  
Estudiante

\_\_\_\_\_  
Celio Gustavo Paguay Quiroz  
Estudiante

## **ANEXO 6. MANUAL DE USUARIO**



DIRECCIÓN DE TECNOLOGÍAS DE LA  
INFORMACIÓN Y COMUNICACIÓN

COORDINACIÓN DE GESTIÓN DE N.1  
DESARROLLO DE SISTEMAS INFORMÁTICOS

## **MANUAL DE USUARIO**

*<Módulo de Matrícula Unidad de Titulación>*

Versión 1.0



## Control del Documento

TÍTULO: MANUAL DE USUARIO

VERSIÓN: 1.0

CÓDIGO DEL FORMATO:

DEPENDENCIA: CODESI

## Firmas y Aprobaciones

ELABORADO POR: [Cepeda Pataron Ligia Adriana]  
[Estudiante]  
FECHA: [2024-07-12] Firma: \_\_\_\_\_

[Paguay Quiroz Celio Gustavo]  
[Estudiante]  
FECHA: [2024-07-12] Firma: \_\_\_\_\_

REVISADO POR: [Henry Javier Paca Quinaluiza]  
[Encargado de sistemas informáticos]  
FECHA: [2024-07-12] Firma: \_\_\_\_\_

APROBADO POR: [José Rafael Salguero Rosero]  
[Director de dirección Académica]  
FECHA: [2024-07-12] Firma: \_\_\_\_\_

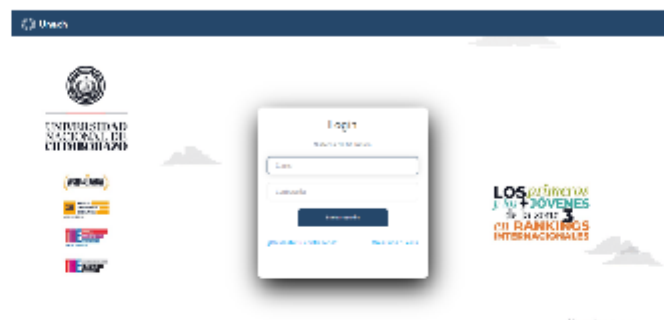




## ROL /ADMINISTRADOR

### INICIO DE SESIÓN

- Pantalla de inicio del sistema de módulo de matrícula de titulación.



- Ingresar las credenciales y presionar en iniciar sesión.



- Deberá aparecer "Credenciales encontradas con éxito".





- Aparecerá la página principal del Administrador\*



### CONFIGURACIÓN RESPONSABLES

- Ingresar al menú de Responsables, se puede hacer desde el panel lateral o del menú de la página principal



- Para ver las personas que están a cargo en el periodo actual se debe seleccionar el cargo y automáticamente aparecerá la lista en la tabla

Administración de Responsables de la Unidad de la Unidad de Titulación

Vigentes
  Histórico de Responsables
 Disponibilidad de Cargos
Periodo 2024-10

ID	Nombre	Correo Institucional	Disponibilidad de Cargos	Cargos
172	RODRIGO ESCOBARDO	rodrigo.escobar@unach.edu.pe	<div style="border: 1px solid black; padding: 2px;">                     Disponibilidad de Cargos                      No tiene asignados cargos                      No tiene asignados cargos                      No tiene asignados cargos  <b>Reservado para Cargos</b> </div>	1. COORDINADOR DE TITULACIÓN 2. ASISTENTE ADMINISTRATIVO
274	RODRIGO ESCOBARDO	rodrigo.escobar@unach.edu.pe	ENCUENTRO DE INVESTIGACIÓN DE LA SALUD	1. COORDINADOR
284	MARIA CRISTINA	maria.cristina@unach.edu.pe	ENCUENTRO DE INVESTIGACIÓN DE LA SALUD	COORDINADORA
125	JULIANA ALVARADO	juliana.alvarado@unach.edu.pe	ENCUENTRO DE INVESTIGACIÓN DE LA SALUD	COORDINADORA



- Para ver los responsables anteriores debe seleccionar el cargo y el periodo y automáticamente aparecerá la lista en la tabla.

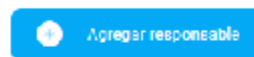
Para más información consulte el manual de usuario de la aplicación en la dirección:

1 2 3

11 MyArea  Director de Carrera

ID	Nombre	Cargo Institucional	Facultad	Carrera
101	LUISA PATRICIA	COORDINADORA DE S.I.	INGENIERÍA DE SISTEMAS	INGENIERÍA

- Para agregar un nuevo responsable ir al botón "Agregar responsable"



- Seleccionamos el cargo, aparecerá una lista y al presionar en agregar se guarda a lista que muestra en la tabla.

Agregar responsables de titulación

Periodo: TT Feb 2024-25 Cargo:

ID	Nombre	Cargo Institucional	Facultad	Carrera
101	LUISA PATRICIA	COORDINADORA DE S.I.	INGENIERÍA DE SISTEMAS	INGENIERÍA
201	RODRIGO	COORDINADOR	INGENIERÍA DE SISTEMAS	INGENIERÍA
301	ANDREA	COORDINADORA	INGENIERÍA DE SISTEMAS	INGENIERÍA
401	ANDREA	COORDINADORA	INGENIERÍA DE SISTEMAS	INGENIERÍA

- **Para agregar secretarías de carrera:** Seleccionamos el cargo, aparecerá un cuadro de texto donde se debe poner el correo institucional y ponemos buscar, en la tabla se muestra el ID y nombre del usuario, seleccionamos la facultad y la carrera a la que pertenece y presionamos agregar.

Agregar responsables de titulación

Periodo: TT Feb 2024-25 Cargo:

Usuario:

ID	Nombre	Cargo Institucional	Facultad	Carrera
101	LUISA PATRICIA	COORDINADORA DE S.I.	<input type="button" value="INGENIERÍA DE SISTEMAS"/>	<input type="button" value="INGENIERÍA DE SISTEMAS"/>



- Aparecerá mensaje de "Usuarios/responsables agregados correctamente!"

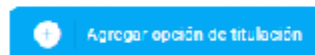


### CONFIGURACIÓN OPCIONES DE TITULACIÓN

- Seleccionamos Opciones de Titulación y aparecerá la lista de las opciones de titulación vigentes en la unidad de titulación.



- Para agregar una opción de titulación, presionamos el botón:



- Escribimos el nombre de la nueva opción de titulación y presionamos Agregar





- Se visualiza mensaje de que se agregó correctamente.



- Se visualiza la nueva opción en la tabla

ID	Opcion de titulación
1	Proyecto de Investigación
2	Examen de grado de caracter complejo
3	Artículo Científico
4	Emprendimiento

## CONFIGURACION TIPOS DE MATRÍCULA

- Seleccionamos el menú Tipos de matrícula

ID	Nombre	Anualidad
1	Tituación	53
2	De Grado	1000
3	De Especialidad	1000
4	De Grado - No concurrida	1000
5	De Grado - No concurrida	1000

- Para agregar un tipo de matrícula, presionamos:





- Ingresamos el nombre y el arancel del tipo de matrícula y presionamos Agregar

Agregar Tipo de Matrícula

Nombre:  
Tipo de Matrícula  
Arancel

Matrícula ordinaria  
500

Cancelar Agregar

- Se visualiza mensaje de que se agregó correctamente.

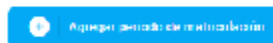


## CONFIGURACION PERIODOS DE MATRÍCULA

- Seleccionamos el menú Periodo de matrículas



- Para agregar un nuevo Periodo de matriculación, presionamos:



- Seleccionamos/llenamos todos los campos, y presionamos Agregar.

Agregar periodo de matriculación configuración

Periodo de Matrícula

Periodo de Matrícula	Inicio	Fin
...	...	...

Periodo de Matrícula

Periodo de Matrícula	Inicio	Fin
...	...	...

Cancelar Agregar



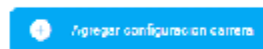
## ROL / RESPONSABLE CARRERA

### CONFIGURACION CARRERA

- Seleccionamos el menú Carrera



- Para agregar una nueva configuración, presionamos:



- Ingresamos un nombre, y seleccionamos/llenamos todos los campos y presionamos en Agregar.

### CONFIGURACIÓN OPCIONES DE TITULACIÓN

- Seleccionamos el menú Opciones de Titulación





- Para agregar una nueva opción de titulación a la carrera, presionamos:

Agregar Opción de titulación

- Seleccionamos el periodo, configuración y opción de titulación y presionamos en Agregar.

Formulario de configuración de opciones de titulación. Incluye campos para seleccionar el periodo, configuración y opción de titulación, con botones 'Cancelar' y 'Agregar'.

### CONFIGURACION SECUENCIA DE PERIODOS

- Seleccionamos el menú Secuencia de periodos

Menú de navegación con 'Secuencia de periodos' resaltado. Tabla de configuración de secuencias de periodos:

ID Secuencia	Periodo	Fecha Inicio
11	11 Periodo 2024-25	11/01/2024 00:00:00
10	10 Periodo 2023-24	01/01/2023 00:00:00
9	9 Periodo 2022-23	11/01/2022 00:00:00
8	8 Periodo 2021-22	01/01/2021 00:00:00

- Para agregar una nueva secuencia de periodos, presionamos:

Agregar secuencia de periodo de titulación

- Seleccionamos el periodo y en la parte derecha seleccionamos el periodo anterior al mismo y presionamos en Agregar.

Formulario de configuración de secuencia de periodo de titulación. Incluye campos para seleccionar el periodo y el periodo anterior, con botones 'Cancelar' y 'Agregar'.







- Seleccionar la opción de titulación deseada

- Se visualiza mensaje de que se seleccionó una opción de titulación correctamente.



- Se muestra en resumen la opción que tiene el estudiante.

## HISTORIAL: TRAYECTORIA ACADÉMICA

- Seleccionamos el menú Trayectoria Académica

ID	NOMBRE	CARRERA	CARRERA	CARRERA	CARRERA	CARRERA	CARRERA	CARRERA	CARRERA
1	INGENIERIA DE SISTEMAS	INGENIERIA DE SISTEMAS	INGENIERIA DE SISTEMAS	INGENIERIA DE SISTEMAS	INGENIERIA DE SISTEMAS	INGENIERIA DE SISTEMAS	INGENIERIA DE SISTEMAS	INGENIERIA DE SISTEMAS	INGENIERIA DE SISTEMAS
2	INGENIERIA DE SISTEMAS	INGENIERIA DE SISTEMAS	INGENIERIA DE SISTEMAS	INGENIERIA DE SISTEMAS	INGENIERIA DE SISTEMAS	INGENIERIA DE SISTEMAS	INGENIERIA DE SISTEMAS	INGENIERIA DE SISTEMAS	INGENIERIA DE SISTEMAS





- Para aceptar y generar la nueva matrícula presionamos en:



- Nos aparece un mensaje de confirmación y presionamos Confirmar



- Se visualiza mensaje de que se generó la nueva matrícula correctamente.



- Aparece una vista previa del modelo de orden de pago, se puede presionar en descargar para tener el pdf.



- PDF del modelo de orden de pago (posterior implementación por CODESI)





## SECRETARIA

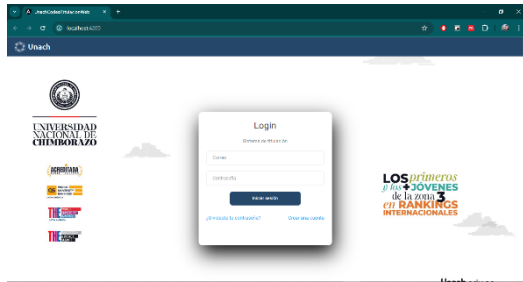
### LISTA DE ESTUDIANTES

- Seleccionamos el menú Lista de Estudiantes y se puede visualizar todas las matrículas de los estudiantes en la unidad de titulación.

Matrícula	Código	Apellidos	DNI	Nombre	Fecha de Nacimiento
101	101	ALVARADO	71000000	ALVARADO	10/10/1990
102	102	ALVARADO	71000000	ALVARADO	10/10/1990

## **ANEXO 7. IMPLEMENTACIÓN FRONTEND**

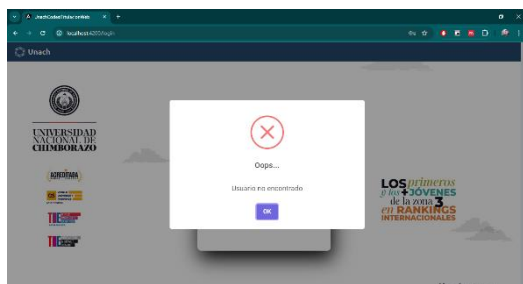
## Pantalla del Inicio de sesión



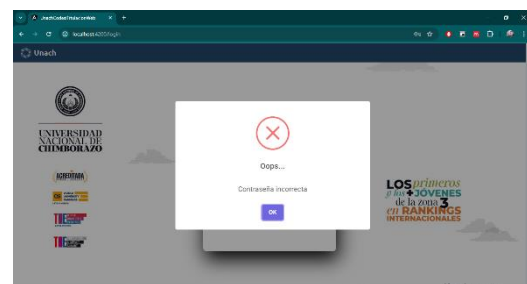
Pantalla principal login



Pantalla ingreso exitoso

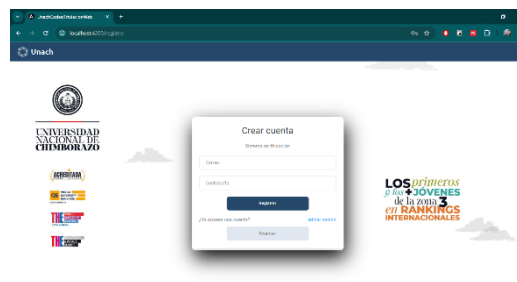


Pantalla usuario no encontrado

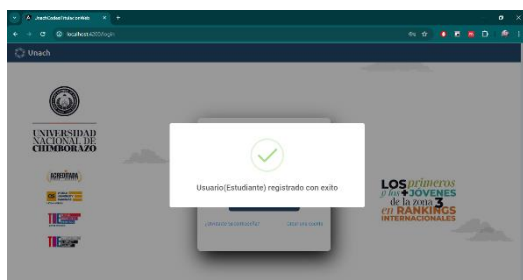


Pantalla contraseña incorrecta

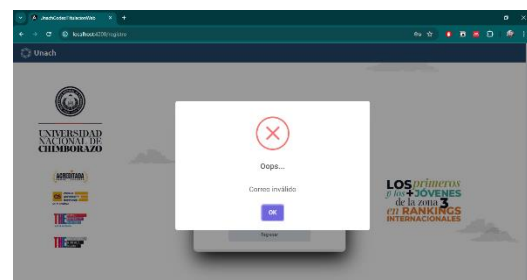
## Pantalla del Registro



Pantalla principal crear cuenta



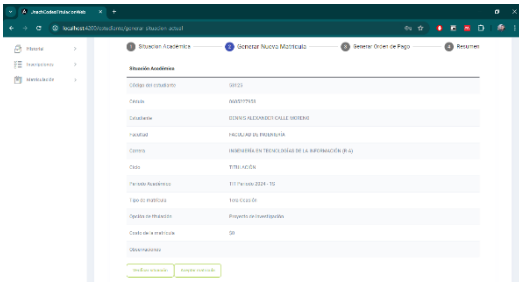
Pantalla registro exitoso



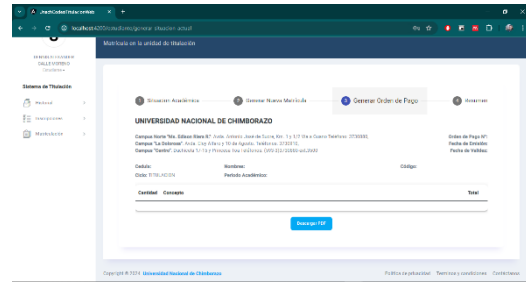
Pantalla correo inválido





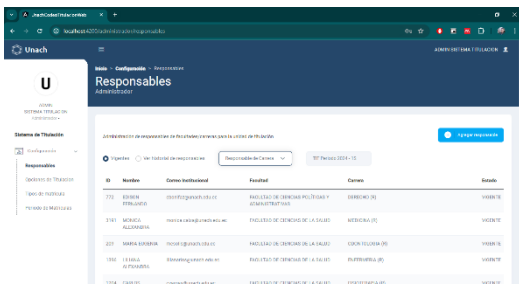


Pantalla opción de titulación estudiante paso 2

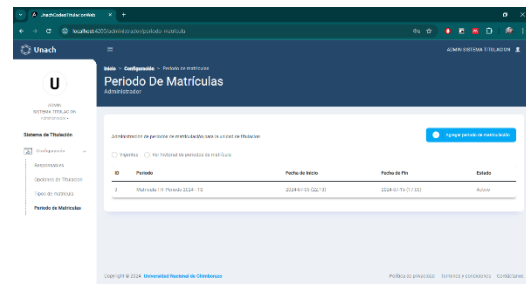


Pantalla opción de titulación estudiante paso 3

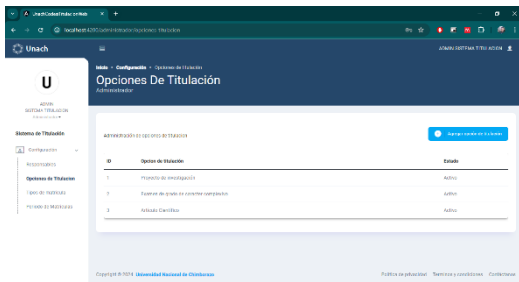
## Pantallas del administrador



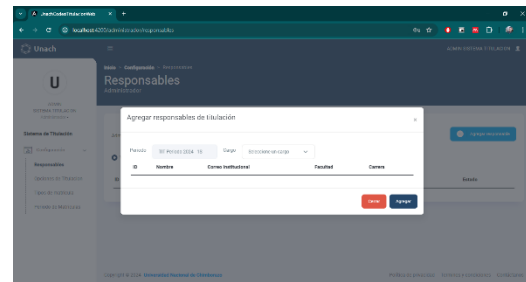
Pantalla responsables - administrador



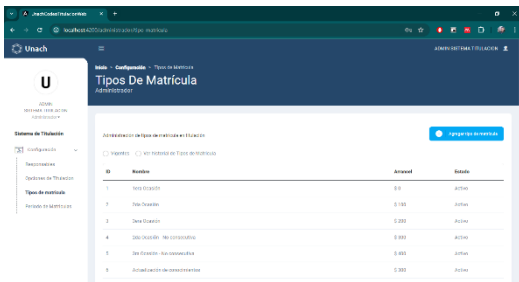
Pantalla periodos de matrículas/administrador



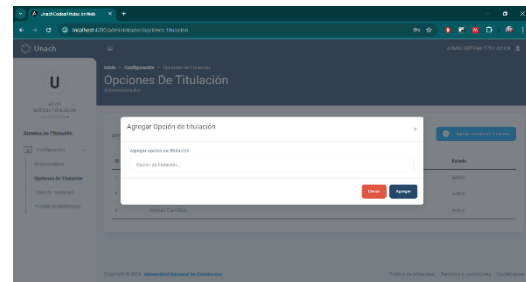
Pantalla opciones de titulación/administrador



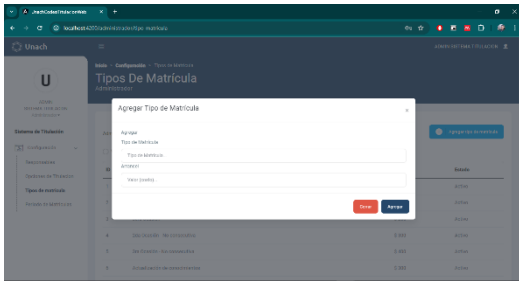
Pantalla agregar responsables - administrador



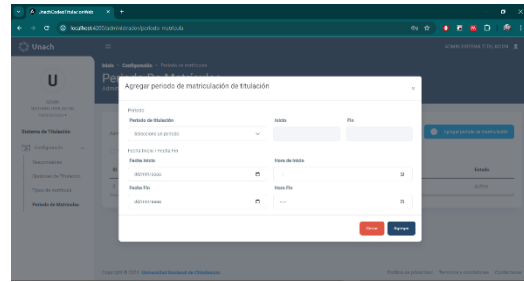
Pantalla tipos de matrícula/administrador



Pantalla agregar opción titulación/administrador



Pantalla agregar tipo de matrícula/administrador

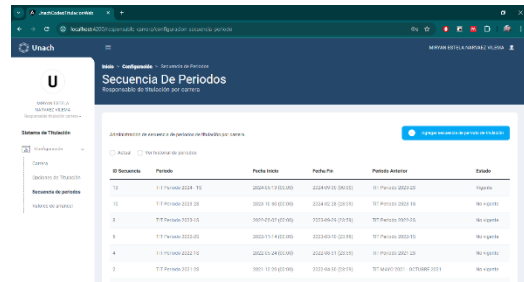


Pantalla agregar periodo de matrícula/administrador

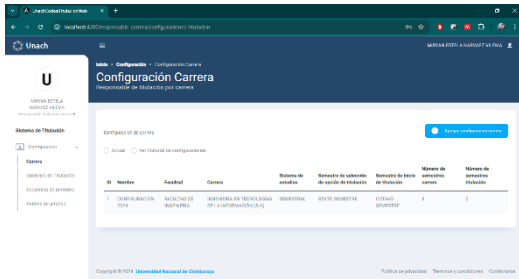
## Pantallas Responsable titulación carrera



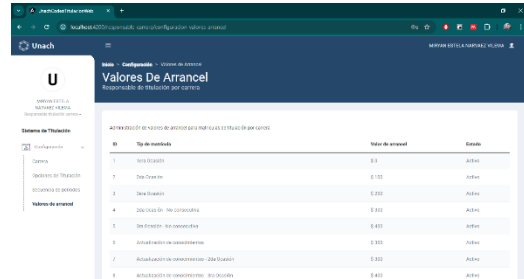
Pantalla principal/responsable



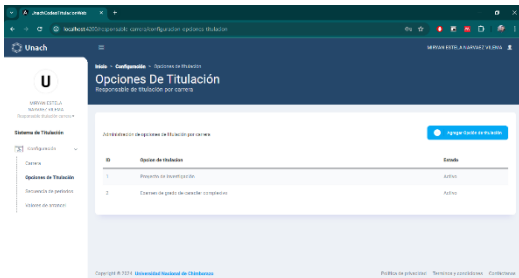
Pantalla secuencia periodos /responsable



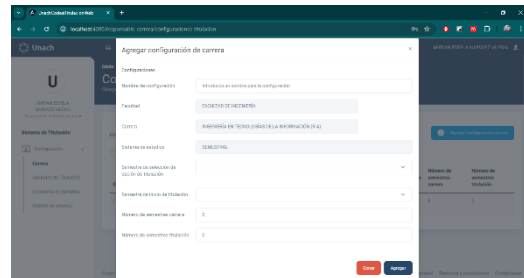
Pantalla configuración carrera/responsable



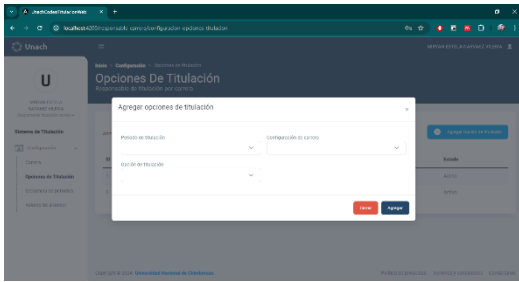
Pantalla valores de arancel/responsable



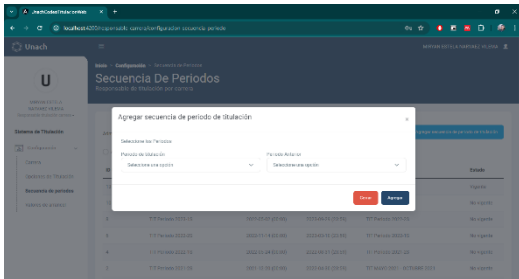
Pantalla opciones de titulación/responsable



Pantalla agregar configuración carrera/responsable



Pantalla agregar opción titulación/responsable



Pantalla agregar secuencia periodos /responsable

## **ANEXO 8. RESULTADOS EVALUACIÓN CON JMETER**

## Tiempo de respuesta

Thread Group

Name:

Comments:

Action to be taken after a Sampler error

Continue  Start Next Thread Loop  Stop Thread  Stop Test  Stop Test Now

Thread Properties

Number of Threads (users):

Ramp-up period (seconds):

Loop Count:  Infinite

Same user on each iteration

Delay Thread creation until needed

Specify Thread lifetime

Duration (seconds):

Startup delay (seconds):

HTTP Request

Name:

Comments:

Basic Advanced

Web Server

Protocol (http):  Server Name or IP:  Port Number:

HTTP Request

POST Path:  Content encoding:

Redirect Automatically  Follow Redirects  Use KeepAlive  Use multipart/form-data  Browser-compatible headers

Parameters Body Data Files Upload

```
1 {
2   "Nombre" : "Nombre Tipo de Matricula",
3   "Cantidad" : 500.0
4 }
```

HTTP Request

Name:

Comments:

Basic Advanced

Web Server

Protocol (http):  Server Name or IP:  Port Number:

HTTP Request

POST Path:  Content encoding:

Redirect Automatically  Follow Redirects  Use KeepAlive  Use multipart/form-data  Browser-compatible headers

Parameters Body Data Files Upload

```
1 {
2   "Nombre" : "Nombre Tipo de Matricula",
3   "Cantidad" : 500.0
4 }
```

tr1.jmx (C:\Users\Celio\Documents\Tesis\Pruebas\tr1.jmx) - Apache JMeter (5.6.3)

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename:  Browse... Log/Display Only:  Errors  Successes  Configur

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes 1
Linq/Lambda	1200	1	0	76	4.41	0.00%	38.3/sec	6.76	10.50	
Sistema Tradicional	1200	5	3	165	6.99	0.00%	38.3/sec	40.68	11.36	
TOTAL	2400	3	0	165	6.16	0.00%	76.5/sec	47.43	21.86	

Thread Group

Name: Comparativa Post Generar Matricula

Comments:

Action to be taken after a Sampler error

Continue  Start Next Thread Loop  Stop Thread  Stop Test  Stop Test Now

Thread Properties

Number of Threads (users): 100

Ramp-up period (seconds): 10

Loop Count:  Infinite 4

Same user on each iteration

Delay Thread creation until needed

Specify Thread lifetime

Duration (seconds):

Startup delay (seconds):

HTTP Request

Name: Linq/Lambda

Comments:

Basic Advanced

Web Server

Protocol [http]: http Server Name or IP: localhost Port Number: 5247

HTTP Request

POST Path: http://localhost:5247/api/TipoMatricula/GenerarMatriculaTitulacion Content encoding:

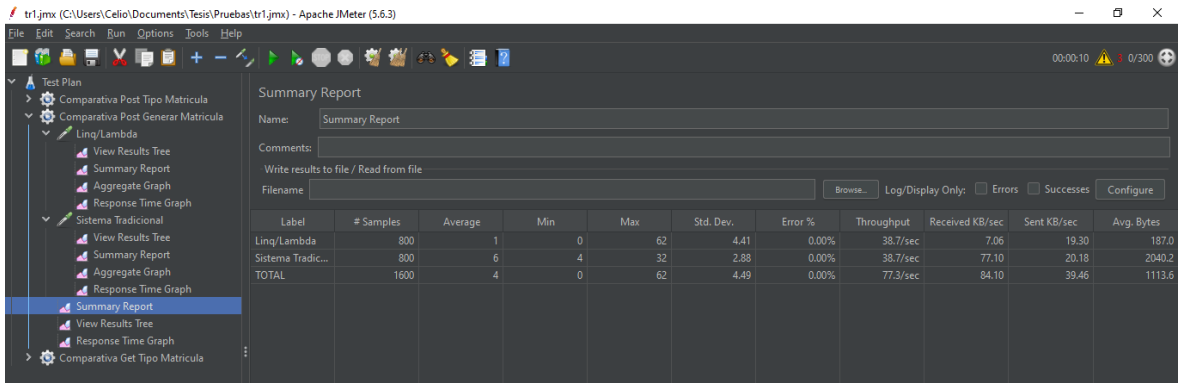
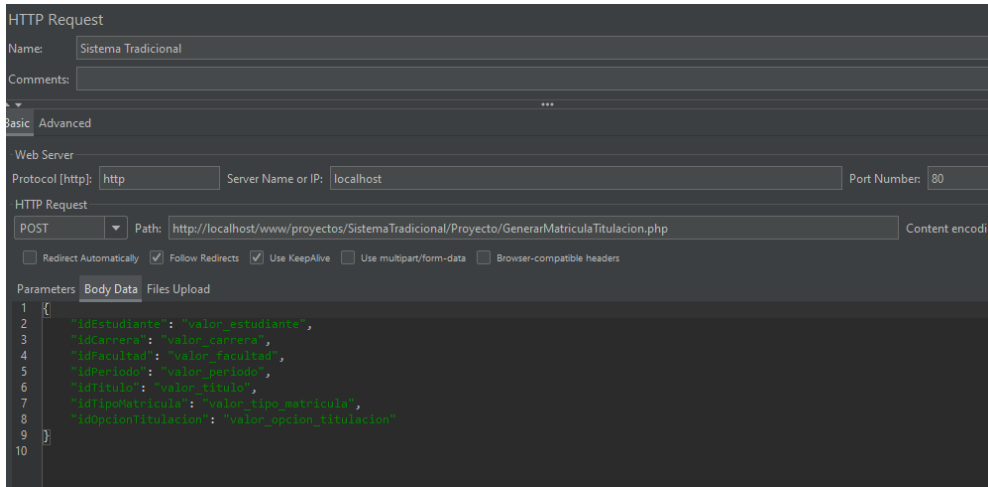
Redirect Automatically  Follow Redirects  Use KeepAlive  Use multipart/form-data  Browser-compatible headers

Parameters Body Data Files Upload

```

1 {
2   "idEstudiante": "valor_estudiante",
3   "idCarrera": "valor_carrera",
4   "idFacultad": "valor_facultad",
5   "idPeriodo": "valor_periodo",
6   "matricula": "valor_matricula",
7   "idTipoMatricula": "valor_tipo_matricula",
8   "idopcionTitulacion": "valor_opcion_titulacion"
9 }
10

```



## Tiempo de procesamiento

