



UNIVERSIDAD NACIONAL DEL ALTIPLANO
FACULTAD DE INGENIERÍA MECÁNICA ELÉCTRICA
ELECTRÓNICA Y SISTEMAS
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS



MEJORAMIENTO DEL SISTEMA ACADÉMICO DE LA
UNIVERSIDAD PRIVADA SAN CARLOS – PUNO MEDIANTE
COMPONENTES LINQ DE MICROSOFT .NET

TESIS

PRESENTADA POR:

Bach. EDGAR RAÚL LÓPEZ QUISPE

Bach. FELIX SUERO QUISPE

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO DE SISTEMAS

PUNO - PERÚ

2021



DEDICATORIA

A nuestras familias por el apoyo incondicional.



AGRADECIMIENTOS

A Dios por la vida y la bondad de la existencia.

A la Universidad Nacional del Altiplano, a la Escuela Profesional de Ingeniería de Sistemas.

A nuestros docentes por el conocimiento impartido.

Agradecemos a nuestro asesor Dr. Sc. Adolfo Carlos Jiménez Chura, por su valioso apoyo de asesoría y constante orientación y ayuda incondicional durante el desarrollo del trabajo de investigación.

Y de manera especial al presidente y a los miembros del jurado, Dr. Sc. Elmer, Coylla Idme, Dr. Sc. Fidel Ernesto Ticona Yanqui, Mg. Edwin Fredy Calderón Vilca respectivamente, por sus sugerencias y correcciones para mejorar la tesis, durante todo el proceso de control del trabajo de investigación.



INDICE GENERAL

DEDICATORIA

AGRADECIMIENTOS

INDICE GENERAL

ÍNDICE DE FIGURAS

INDICE DE TABLAS

ÍNDICE DE ACRÓNIMOS

RESUMEN 13

ABSTRACT..... 14

CAPÍTULO I

INTRODUCCIÓN

1.1. PLANTEAMIENTO DEL PROBLEMA 16

1.2. PROBLEMA GENERAL..... 16

1.3. HIPÓTESIS DE LA INVESTIGACIÓN 16

1.3.1. Hipótesis general..... 16

1.4. DELIMITACIÓN DE LA INVESTIGACIÓN 17

1.5. OBJETIVOS..... 17

1.5.1. Objetivo general..... 17

1.5.2. Objetivos específicos 17

1.6. JUSTIFICACIÓN DE LA INVESTIGACIÓN 18

CAPÍTULO II

REVISIÓN DE LITERATURA

2.1. ANTECEDENTES DE ESTUDIO 19

2.2. LOS LENGUAJES DE PROGRAMACIÓN 24

2.2.1. Algo de historia de LINQ 26

2.2.2. Lenguaje Integrado de Consultas (LINQ) 26

2.2.3. Consultas LINQ 28

2.2.3.1. Expresión lambda 28

2.2.3.2. Operadores básicos 29

2.2.3.3. Sintaxis fluida 30



2.2.3.4. Operadores de consultas en cadena	30
2.2.4. Expresiones de consulta.....	32
2.2.4.1. Rango de variable	34
2.2.4.2. Cómo funciona la ejecución diferida.....	35
2.2.4.3. Decoradores encadenados.....	36
2.2.4.4. Cómo se ejecutan las consultas.....	37
2.2.5. Operadores de consulta estándar LINQ.....	38
2.2.6. LINQ para objetos	38
2.2.7. Subconsultas	39
2.2.8. Tipos anónimos.....	41
2.2.9. Microsoft Visual Studio .NET	42
2.2.10. Evolución de Visual Studio	43
2.2.10.1. Datos generales de Visual Studio	44
2.2.10.2. Tipos de Visual Studio.....	44
2.2.11. Integridad referencial	45
2.2.11.1. Tipos de integridad referencial	47
2.2.11.2. Clave foránea	47
2.2.12. Normalización.....	49
2.2.12.1. Primera forma normal.....	49
2.2.12.2. Segunda forma normal.....	50
2.2.12.3. Tercera forma normal	50
2.2.13. iTextSharp.....	51
2.2.13.1. Cosas que se hacen con iTextSharp.....	51
2.2.13.2. Versiones	52
2.2.13.3. Creando un archivo	52
2.2.14. ClosedXML	53
2.2.14.1. Instalación del componente	53
2.2.14.2. Creando y grabando documentos.....	54
2.2.14.3. Integrando closedXML con LINQ.....	54
2.2.15. Lenguaje de programación C#.....	55
2.2.16. Evolución de C#.....	56



2.2.16.1. Orientado a objetos	56
2.2.17. Arquitectura de la plataforma .NET Framework	58
2.2.18. La interoperabilidad entre lenguajes	60
2.2.19. Características de .Net Framework	61
2.2.19.1. Soporte de estándares	61
2.2.19.2. Soporte de varios lenguajes de programación	61
2.2.19.3. Modelo simplificado de desarrollo	61
2.2.19.4. Jerarquía de clases extensible	62
2.2.19.5. Ensamblados	62
2.2.19.6. Espacios de nombres	62
2.2.19.7. Construcción y Ejecución del Código	62
2.3. SISTEMA ACADÉMICO	63
2.3.1. Actividades del sistema académico	63
2.4. GESTIÓN ACADÉMICA	64
2.5. GESTIÓN Y PLANEACIÓN ESTRATÉGICA UNIVERSITARIA	65
2.6. LAS TICS COMO FACTOR CLAVE EN LA GESTIÓN ACADÉMICA. 66	
2.7. PROGRAMACIÓN POR CAPAS	66
2.7.1. Capa de presentación - vista	67
2.7.2. Capa de negocios - controlador	67
2.7.3. Capa de datos - modelo	67
2.7.4. Ventajas	67
2.7.5. Desventajas	68
2.8. MAPEO DE DATOS Y LAS TÉCNICAS	68
2.8.1. Mapeo de datos	68
2.8.2. La importancia del mapeo de datos	70
2.8.2.1. Integración de datos	70
2.8.2.2. Migración de datos	71
2.8.2.3. Almacenamiento de datos	71
2.8.2.4. Transformación de datos	72
2.8.2.5. Intercambio electrónico de datos	72
2.8.3. Técnicas de mapeo de datos	72



2.8.3.1. Mapeo manual de datos	73
2.8.3.2. Mapeo de datos automatizado.....	73
2.8.3.3. Mapeo automatizado de datos.....	74
2.9. MARCO CONCEPTUAL	75
2.9.1. Base de datos	75
2.9.2. iTextSharp.....	75
2.9.3. MySql.Data.dll.....	75
2.9.4. Entidad - relación.....	75
2.9.5. Componentes	76
2.9.6. Aplicación de escritorio	76
2.9.7. Programación	76
2.9.8. Software	76
2.9.9. Administración.....	76
CAPÍTULO III	
MATERIALES Y MÉTODOS	
3.1. TRABAJO EXPERIMENTAL.....	77
3.1.1. Tipo de investigación.....	77
3.1.2. Diseño de la investigación	77
3.1.3. Variables de estudio.....	78
3.1.4. Población	78
3.1.5. Ámbito de estudio.....	78
3.2. TÉCNICAS DE RECOLECCIÓN DE DATOS.....	79
3.3. MATERIAL EXPERIMENTAL.....	80
3.3.1. Herramientas usadas	80
3.4. MÉTODOS EXPERIMENTALES	81
3.4.1. Método de tratamiento de datos.....	81
3.4.2. Prueba de hipótesis para dos muestras apareadas	81
CAPÍTULO IV	
RESULTADOS Y DISCUSIÓN	
4.1. DIAGRAMA DE BASE DE DATOS	82
4.1.1. Errores encontrados	83



4.2. CAPA DE ACCESO A DATOS	87
4.3. CAPA DE NEGOCIO	88
4.4. TIEMPO DE RESPUESTA DE CONSULTAS	89
4.4.1. Tiempo de ejecución de consultas de SQL y LINQ	89
4.5. ANÁLISIS ESTADÍSTICO	94
4.5.1. Prueba de hipótesis	95
4.5.2. Nivel de significancia	95
4.5.3. Test de Shapiro-Wilk	95
V. CONCLUSIONES.....	98
VI. RECOMENDACIONES	99
VII. REFERENCIAS.....	100
ANEXOS.....	103

ÁREA: Ciencias de la ingeniería

LÍNEA: Ingeniería de Software, Bases de Datos e Inteligencia de Negocios

FECHA DE SUSTENTACIÓN: 27 de agosto del 2021



ÍNDICE DE FIGURAS

Figura 1: Lenguaje integrado de consultas	27
Figura 2: Filtros en expresiones LINQ.....	32
Figura 3: Sintaxis de consultas en expresiones LINQ.....	34
Figura 4: Decoradores de consultas.....	35
Figura 5: Las capas y los decoradores	36
Figura 6: Secuencia de ejecución de consultas.....	38
Figura 7: Ejecución de subconsultas	40
Figura 8: Visual Studio .NET	43
Figura 9: Relación entre tablas	49
Figura 10: ClosedXML.....	53
Figura 11: Secuencia de ejecución del constructor	58
Figura 12: El lenguaje C#.....	60
Figura 13: Programación en capas	67
Figura 14: Mapeo de datos	68
Figura 15: Mapeo de datos complejo	70
Figura 16: Relación de base de datos automatizada.....	73
Figura 17: Diagrama de base de datos inicial.....	83
Figura 18: Script de consulta SQL estándar	84
Figura 19: Base de datos final con integridad referencias.....	86
Figura 20: Tipos de datos de objetos	87
Figura 21: Script de consulta entre varias tablas	87
Figura 22: Script de expresión LINQ	88
Figura 23: Porcentaje de tiempos de ejecución para tres tablas	90
Figura 24: Porcentaje de tiempos de ejecución para cuatro tablas.....	91



Figura 25: Porcentaje de tiempos de ejecución para cinco tablas	92
Figura 26: Porcentaje de tiempos de ejecución para seis tablas	93
Figura 27: Porcentaje de tiempos de ejecución para siete tablas.....	94



INDICE DE TABLAS

Tabla 1: Operadores de consultas	39
Tabla 2: Información de Visual Studio.....	44
Tabla 3: Integridad referencial.....	46
Tabla 4: Versiones de iTextSharp.....	52
Tabla 5: Herramientas usadas	80
Tabla 6: Errores iniciales en la base de datos	83
Tabla 7: Errores encontrados en códigos de matrícula.....	85
Tabla 8: Tiempo de ejecución de tres tablas.....	89
Tabla 9: Tiempo de ejecución de cuatro tablas	90
Tabla 10: Tiempo de ejecución de cinco tablas.....	91
Tabla 11: Tiempo de ejecución de seis tablas	92
Tabla 12: Tiempo de ejecución de siete tablas	93



ÍNDICE DE ACRÓNIMOS

- LINQ** : Language Integrated Query, son un conjunto herramientas de Microsoft para realizar todo tipo de consultas a distintas fuentes de datos.
- PLINQ** : Es una forma de ejecutar las consultas LINQ en paralelo en sistemas multi-core / multi-procesador.
- XML** : Extensible Markup Language, es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos.
- SQL** : Structured Query Language, en español lenguaje de consulta estructurada, es un lenguaje de dominio específico utilizado en programación.
- MySQL** : es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales.
- CRUD** : Hace referencia a un acrónimo en el que se reúnen las primeras letras de las cuatro operaciones fundamentales de aplicaciones persistentes en sistemas de bases de datos (**C**reate, **R**ead, **U**ppdate, **D**elete).



RESUMEN

El desarrollo del presente trabajo de tesis titulado “Mejoramiento del Sistema Académico de la Universidad Privada San Carlos – Puno mediante componentes LINQ de Microsoft .NET” permitirá mejorar el Sistema Académico con un software desarrollado en Visual Studio 2010 empleando el componente LINQ con la finalidad de elaborar expresiones lambda y expresiones SQL sobre el motor de base de datos MySQL para obtener el mejor rendimiento en el resultado de la obtención de datos. Para la solución a este problema se emplearon las siguientes técnicas y métodos para el desarrollo: las entrevistas que permitieron determinar los nuevos requerimientos; modelo relacional para el diseño de la base de datos, la metodología orientada a objetos para el desarrollo del software con el lenguaje de programación C#. El diseño elegido de la investigación fue el cuasi-experimental, con un solo grupo de prueba al cual se le aplicó el pretest y postest a las consultas SQL estándar y expresiones LINQ para así evaluar y validar la hipótesis de investigación aplicando la prueba de Wilcoxon para dos muestras relacionadas con el estadístico de prueba Z. La principal conclusión que se desprende de la investigación es que con el uso del componente LINQ las consultas sobre la base de datos tienen un menor tiempo de respuesta usando las expresiones LINQ con el lenguaje de programación C#.

Palabras claves: LINQ, SQL estándar, Sistema académico, Microsoft .NET, software.



ABSTRACT

The development of this thesis work entitled "Improvement of the Academic System of the San Carlos Private University - Puno through LINQ components of Microsoft .NET" will allow to improve the Academic System with software developed in Visual Studio 2010 using the LINQ component in order to build lambda expressions and SQL expressions on the MySQL database engine to obtain the best performance in the result of data retrieval. For the solution to this problem, the following techniques and methods were used for development: the interviews that made it possible to determine the new requirements; relational model for database design, object-oriented methodology for software development with the C # programming language. The chosen design of the research was the quasi-experimental, with a single test group to which the pretest and posttest were applied to the standard SQL queries and LINQ expressions in order to evaluate and validate the research hypothesis by applying the Wilcoxon test to two samples related to the Z test statistic. The main conclusion that emerges from the research is that with the use of the LINQ component, the queries on the database have a shorter response time using the LINQ expressions with the C # programming language.

Keywords: LINQ, standard SQL, Academic system, Microsoft .NET, software.



CAPÍTULO I

INTRODUCCIÓN

El trabajo de investigación tiene como finalidad de mejorar el Sistema Académico de la Universidad Privada San Carlos – Puno mediante componentes LINQ de Microsoft .NET, para lo cual lo hemos dividido en cuatro capítulos, los cuales están divididos de la siguiente manera:

En el Capítulo I, se refiere a la problemática de investigación, es decir al planteamiento y formulación del problema, los antecedentes de la investigación, referentes al tema de investigación, el objetivo general, objetivos específicos de la investigación e hipótesis de la investigación.

En el Capítulo II, se desarrolla el marco teórico referidos al tema de estudio que corresponde a los decoradores de las expresiones LINQ, la sintaxis, la arquitectura de la plataforma .NET, también tenemos la definición de los términos básicos utilizados en la investigación y finalmente se formuló la hipótesis de la investigación.

En el Capítulo III, se presenta la metodología utilizada en el trabajo de investigación; también se especifican el tipo y diseño de investigación y finalmente se tiene la población utilizada en la investigación.

En el Capítulo IV, se exponen los resultados de la investigación, donde se considera el análisis comparativo de las consultas SQL estándar y las expresiones LINQ hasta consultas con siete (7) tablas relacionadas.

Y finalmente tenemos las conclusiones a las cuales se ha llegado con la investigación, las recomendaciones y anexos.



1.1. PLANTEAMIENTO DEL PROBLEMA

El presente proyecto de tesis titulado MEJORAR EL SISTEMA ACADÉMICO DE LA UNIVERSIDAD PRIVADA SAN CARLOS – PUNO MEDIANTE COMPONENTES LINQ DE MICROSOFT .NET, busca aplicar el componente LINQ de Microsoft NET al sistema académico, dado que es un lenguaje similar a las sentencias SQL estándar y, que permite realizar consultas en memoria y mejorar el mismo en rendimiento y legitimidad para el sistema administrado por la oficina de Registro Académico de la Universidad Privada San Carlos. Se detectaron problemas de consultas SQL implementadas en los módulos del sistema actual, igualmente, el gestor de base de datos presenta problemas de diseño e integridad referencial. El objetivo principal es mejorar el Sistema Académico con un software desarrollado en Visual Studio 2010 empleando el componente LINQ con la finalidad de elaborar expresiones lambda y expresiones SQL sobre el motor de base de datos MySQL para obtener el mejor rendimiento en el resultado de la obtención de datos, además, establecer una adecuada integridad referencial en las tablas y así evitar errores al momento de actualizar o eliminar datos en el gestor de base de datos MySQL.

1.2. PROBLEMA GENERAL

Por lo expuesto se plantea la siguiente pregunta:

¿Qué mejora se obtendría sobre el Sistema Académico de la Universidad Privada San Carlos al aplicar el componente LINQ de Microsoft .NET?

1.3. HIPÓTESIS DE LA INVESTIGACIÓN

1.3.1. Hipótesis general

El Sistema Académico desarrollado en Visual Studio 2010, las expresiones lambda y expresiones SQL del componente LINQ logran mejorar el tiempo de respuesta



en la recuperación de datos sobre el motor de base de datos MySQL de la Universidad Privada San Carlos de Puno.

1.4. DELIMITACIÓN DE LA INVESTIGACIÓN

Espacial

Universidad Privada San Carlos, ubicado en Conde de Lemos 128 de la ciudad de Puno.

Temporal

Doce (12) meses.

1.5. OBJETIVOS

1.5.1. Objetivo general

Mejorar el Sistema Académico desarrollado en Visual Studio 2010 empleando las expresiones lambda y expresiones SQL del componente LINQ para obtener el mejor rendimiento en la obtención de datos sobre el motor de base de datos MySQL de la Universidad Privada San Carlos de Puno.

1.5.2. Objetivos específicos

- Realizar un análisis de la integridad referencial de la base de datos MySQL para obtener el mejor rendimiento al momento de usar las operaciones CRUD sobre el motor MySQL.
- Realizar un análisis comparativo del Lenguaje de Consultas Integrado LINQ y las consultas SQL.
- Desarrollar el Sistema Académico con el lenguaje de programación C# y el componente LINQ de Microsoft .NET para mejorar el rendimiento de la obtención de datos.



1.6. JUSTIFICACIÓN DE LA INVESTIGACIÓN

La presente investigación se enfocará en estudiar el componente LINQ y aplicarlo al Sistema Académico de la Universidad Privada San Carlos de Puno, ya que debido a los recientes avances de desarrollo tecnológicos producidos por Microsoft .NET es necesario aprender estas herramientas y aplicarlos a los procesos que se dan en el Sistema Académico.

LINQ define un conjunto de operadores de consulta estándar que permiten seleccionar, filtrar datos, y crear proyecciones a nivel de programación de varios tipos de almacenamiento, entre ellos: arreglos, clases enumerables, XML, bases de datos relacionales y orígenes de datos de terceros; usando la misma sintaxis del lenguaje de acceso a datos SQL.

Lo que significa una enorme simplificación al usar muy pocas líneas de código en el lenguaje de programación natural para realizar operaciones y obtener resultados. Al usar LINQ junto con expresiones Lambda para usar funciones personalizadas obteniendo mayor expresividad en la programación empleada y permitiendo resolver problemas complejos en menos líneas de código.

La plataforma de Microsoft .NET Framework 4.5 y el lenguaje C# proveen un conjunto de mejoras y herramientas para el soporte y explotación de LINQ. Los elementos participantes serán: las personas, los procesos y herramientas que puedan ser favorables para el desarrollo de la mejora, entre ellas tenemos; el Sistema Operativo Windows 10, el motor de base de datos MySQL, la herramienta de programación de Visual Studio.Net 2010 (C#).



CAPÍTULO II

REVISIÓN DE LITERATURA

2.1. ANTECEDENTES DE ESTUDIO

En nuestro medio, no se ha realizado estudios relacionados al tema de investigación, sin embargo, a nivel nacional e internacional se han desarrollado temas concernientes a la investigación las que incluyen las variables de estudio.

A lo largo de la historia de la programación y el advenimiento de los gestores de base de datos, obligaba a que se integren para obtener un producto final, tal como indica en el informe final de grado titulado: “**Estudio, prototipación y análisis comparativo de LINQ**” (Malaquina A. 2008), en el cual menciona que:

El propósito del trabajo de investigación es realizar un estudio del alcance y características de LINQ, incluyendo en la tarea no solo la investigación literaria sino también la experimentación y prototipación con fines de evaluación, tanto para aspectos de alcance funcional, facilidad de uso y performances de esta herramienta.

Para evaluar la pertinencia de la herramienta LINQ, se lo compara con herramientas que realizan tareas similares. La comparación anterior se encuentra basada en:

- Un estudio comparativo de performance, para lo cual se realizaron dos aplicaciones con las mismas características, pero donde una desarrolla sus funcionalidades con LINQ y la otra utilizando herramientas tradicionales. En esta comparación se ha incluido la observación de los tiempos de respuesta obtenidos, así como en el consumo de memoria RAM y del acceso a disco de ambas aplicaciones. Un primer eje de análisis comparativo está orientado a un aspecto de difícil definición: la facilidad de uso del ambiente de trabajo. Para



ello se ha medido cuanto trabajo llevó realizar las respectivas aplicaciones, cantidad de líneas de código, y en cuan sustentable es (sustentable en el sentido de que el código escrito es “claro” y fácil de mantener).

Dentro del estudio comparativo se muestra además de qué forma fueron inculcadas las funcionalidades de LINQ en los proyectos realizados por la empresa Active Software.

Luego se desarrolló una aplicación, la cual fue construida sobre una arquitectura en tres capas, bajo el Framework 3.0 de .Net, escrita en lenguaje C#, y basada en una aplicación similar desarrollada en la empresa Active Software (dicha aplicación se encarga de gestionar el tiempo dedicado por una persona a realizar un determinado trabajo dentro de la empresa).

La realización de la aplicación posee como objetivo primordial poner a prueba a LINQ to ENTITY, una de las implementaciones de LINQ, encargada de manejar base de datos Relacional provenientes de diferentes orígenes. Esta permite manipular a las tablas como si fueran objetos, permitiendo que el acceso y manipulación de la capa de datos, pensando en un sistema en tres capas, sea sencillo y eficiente.

El segundo objetivo es el de realizar un análisis comparativo entre LINQ to ENTITY y el ambiente de desarrollo (Framework) construido por Active Software para el desarrollo de cierto tipo de aplicaciones. El análisis se focaliza en mostrar las ventajas y desventajas que poseen cada uno de esos ambientes para la realización de sus funciones, se dan a conocer y/o a tener en cuenta que características debería de tener un modelador de base de datos a objetos.



En la tesis titulada: “Análisis de rendimiento de las Tecnologías Plinq y LinQ en Sistemas Informáticos” (Guerra et al., 2015) muestran los resultados alentadores, tal como se muestra en el siguiente resumen:

La investigación se basó en analizar el rendimiento de los lenguajes integrados de consulta de datos LINQ y PLINQ, para el desarrollo de sistemas informáticos. La medición del rendimiento contempla cuatro indicadores: Tiempo de respuesta, porcentaje de uso de procesamiento, uso de memoria RAM, y operaciones de entrada y salida a discos duros. Los valores correspondientes a los indicadores son obtenidos a través de un módulo software incrustado en el prototipo desarrollado, mismo que a la vez integra LINQ y PLINQ. Con los datos de cada indicador y aplicando estadística descriptiva e inferencial, se determinó que PLINQ ofrece un mejor rendimiento (81.75%) que el lenguaje integrado de consultas LINQ (61.50%). En la ejecución de consultas complejas se obtuvo una diferencia de 61.25%, mientras que LINQ en consultas de datos simples, resalta su superioridad con 33%.

Como cualquier herramienta de desarrollo, ésta se debe aprender y aplicar según los objetivos que se desea alcanzar, tal es el caso de la siguiente **tesis titulada: “Estudio del Lenguaje Integrado de Consultas (Linq), aplicado al desarrollo del Sistema de Inventario y facturación de la librería politécnica”** (Alfonso, Torres, Paguay, & Riobamba -Ecuador, 2008), para la obtención del título de ingeniero en sistemas informático, se plantea el siguiente objetivo:

Realizar un estudio sobre la nueva tecnología LINQ (Language Integrated Query) que ofrece Microsoft en su Framework 3.0, para el desarrollo del sistema de inventario y facturación de la librería politécnica, es



determinar la productividad de la utilización de la misma. Para su realización se utilizó herramientas: Visual Studio 2008, Visual Basic 9.0, LINQ y SqlServer 2000. Los métodos utilizados fueron el científico, deductivo y comparativo además de técnicas estadísticas aplicadas en todo el estudio basado en parámetros. Se ha llegado a obtener los siguientes resultados: La productividad que se gana al momento de programar utilizando LINQ es de 28,18% en lo referente a las líneas de código, el tiempo utilizando LINQ mejoró en un 20%, comprobándose que la utilización de LINQ mejora en un 48.18 % al momento de desarrollar un sistema informático. Se concluye que LINQ mejoró la productividad al desarrollar el sistema de inventario y facturación para la Librería Politécnica, alcanzándose el objetivo propuesto. Se recomienda que, al momento de desarrollar un sistema informático, se utilice la nueva tecnología LINQ ya que reduce el tiempo y mejora la productividad de un programador.

En la siguiente tesis titulada: “Desarrollo de sistema automatizado para la gestión de los principales procesos de facturación y reservas del auto lavado D’Autos en la ciudad de Managua durante el segundo semestre del año 2020” (Nuñez Amador C., Nicaragua, 2020), para la obtención del título de Ingeniero de Sistemas de Información, plantea el siguiente objetivo: Desarrollar para el negocio D’Autos lavado un sistema de escritorio que automatice el proceso de facturación y una aplicación móvil que permita el acceso al servicio de reservas, en la ciudad de Managua durante el segundo semestre del año 2020, usa una serie de componentes de software entre ellas el lenguaje integrado de consultas (LINQ), que proporciona funcionalidades de consultas integradas en diversos orígenes de datos, Entity Frameworks para trabajar con objetos que le permite un nivel de abstracción alto con los datos y aplicaciones orientadas a datos y menos código. Finalmente, en su



conclusión principal el elemento de mayor importancia en este trabajo es la generación de valor agregado al cliente y por ende al negocio una vez que el sistema desarrollado se implemente, en donde ambas partes (cliente y negocio) convergen en la necesidad existente de las herramientas informáticas desarrolladas, éstas son: el sistemas de facturación computarizado y la aplicación móvil con accesibilidad al servicio de reserva, con los cuales se crea un medio de interacción entre cliente y negocio en donde el enfoque de ganar-ganar y satisfacción tiene que perdurar con el fin de propiciar el desarrollo, mejora e innovación de la sociedad nicaragüense.

Otra de las investigaciones es la tesis titulada: “Desarrollo de un aplicativo informático mediante ENTITY FRAMEWORK para el control y monitoreo de proyectos de investigación con mensajería instantánea” (Gorozabel Bazurto, Melvin, 2019), para la obtención del título de Ingeniero de Sistemas y Computación, plantea el siguiente objetivo:

Analizar Entity Framework, resaltar las ventajas que brinda esta herramienta y comprender su funcionamiento mediante el desarrollo de un aplicativo web para el Control y Seguimiento de Proyectos del departamento de investigación de la PUCE- Esmeraldas. Para identificar los procesos que se llevan a cabo en el Departamento de investigación se usaron técnicas como investigación cualitativa, investigación del tipo exploratoria y observación directa; como instrumento para recopilar información se usó una entrevista dirigida al jefe del departamento de investigación, proporcionando así una visión más clara de los requerimientos del aplicativo web. El resultado fue el desarrollo de un aplicativo web según los requerimientos obtenidos en la investigación, incorporando alertas automáticas a través de Telegram al momento en que se envía una tarea o se le asigne como participante, también



proporciona vistas generales de los docentes, estudiantes y proyectos que se encuentran en vigencia, cuenta con las opciones de registrar, agregar participantes, envío y recepción de tareas con el fin de mejorar la gestión del proyecto. Todo esto permitió determinar el funcionamiento de Entity Framework para desarrollo web, este brinda varias funcionalidades como la generación automática de las vistas, métodos, conexión a la base de datos ahorrando tiempo en la etapa de desarrollo.

2.2. LOS LENGUAJES DE PROGRAMACIÓN

En los últimos 30 años, los lenguajes de programación orientados a objetos han evolucionado para convertirse en las principales herramientas para el desarrollo de aplicaciones empresariales. Estos han estado aumentando en frameworks, API y herramientas rápidas de desarrollo de aplicaciones. Todavía lo que falta es una forma de vincular íntimamente los programas orientados a objetos con bases de datos relacionales (y otros datos que no existen como objetos). El paradigma a objetos es conceptualmente diferente del relacional y esto crea impedancia significativa entre los objetos que usan los programas y las tablas donde los datos residen. ADO.NET proporciona una interfaz conveniente para datos relacionales, pero no orientado a objetos (Albahari J. B., 2012). Por ejemplo, este pseudocódigo sería realmente genial:

```
// una clase que representa la tabla de empleados
Employees e = new Employees();

// asignar al identificador ID el valor de 1
e.ID = 1;

// retorna la fila cuando ID=1
e.Retrieve();

// cambia la columna Name con el valor de Alan
e.Name = "Alan";

// Modifica los datos de la base de datos
```




```
e.Udate();
```

El pseudocódigo muestra un enfoque orientado a objetos para la gestión de datos; No existe consultas o instrucción SQL visible para los desarrolladores.

Los programadores de ADO.NET tienen que almacenar el SQL en un objeto Command, asociar el comando con un objeto de conexión y ejecutarlo en ese Objeto de conexión, luego use un DataReader u otro objeto para recuperar el resultado conjunto. Por ejemplo, el siguiente código es necesario para recuperar la fila individual accedido en el pseudocódigo presentado anteriormente (Albahari J. B., 2012).

```
// especificar la conexión a la DB
SqlConnection c = new SqlConnection(...);

// abre la conexión
c.Open();

// especifica el SQL Command
SqlCommand cmd = new SqlCommand(@"

SELECT
*
FROM
Employees e
WHERE
e.ID = @p0
");

// adiciona un valor a parameter
cmd.Parameters.AddWithValue("@p0", 1);

// ejecuta el comando
DataReader dr = c.Execute(cmd);

// retorna el valor de la columna Name
while (dr.Read()) {
string name = dr.GetString(0);
}

// actualiza el registro usando otro comando
```



```
...  
// cierra la conexión  
c.Close();
```

2.2.1. Algo de historia de LINQ

En la Conferencia de Desarrolladores Profesionales de Microsoft (PDC) 2005, Anders Hejlsberg y su equipo presentaron un nuevo enfoque, Language Integrated Query (LINQ), que unifica la forma en que los datos se pueden recuperar en .NET. LINQ proporciona una forma uniforme de recuperar datos de cualquier objeto que implemente la Interfaz IEnumerable <T>. Con LINQ, se puede recuperar de diferentes fuentes de datos almacenados en matrices, colecciones, datos relacionales, XML y todas las posibles fuentes de datos.

Con LINQ, se puede usar la siguiente sintaxis para recuperar los mismos datos de los ejemplos arriba mencionados:

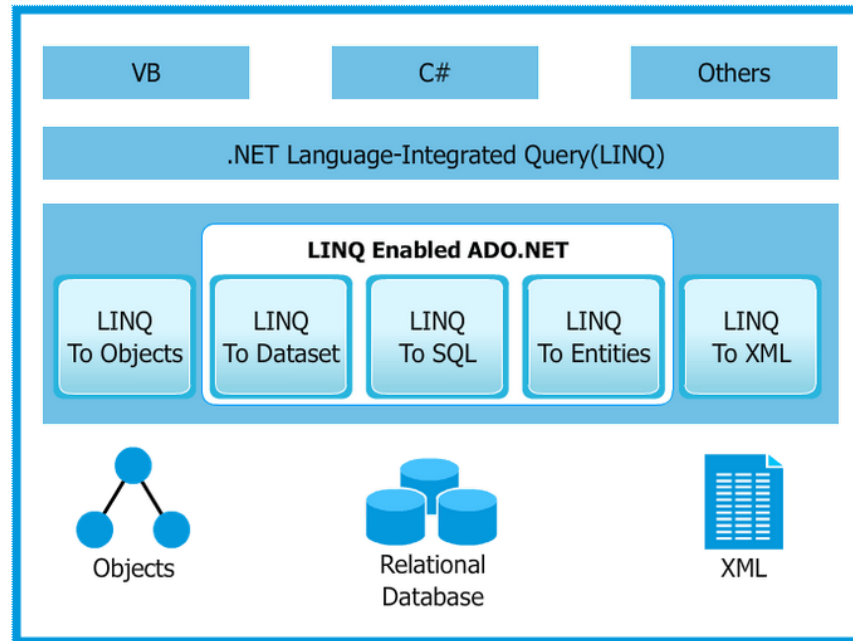
```
var query = from e in employees  
            where e.id == 1  
            select e.name
```

2.2.2. Lenguaje Integrado de Consultas (LINQ)

LINQ, o Language Integrated Query, es un conjunto de características de lenguaje y marcado para escribir consultas estructuradas de tipo seguro sobre colecciones de objetos locales y datos remotos fuentes. LINQ se introdujo en C # 3.0 y Framework 3.5.

LINQ permite consultar cualquier colección que implemente IEnumerable <T>, ya sea una matriz, lista o DOM XML, así como fuentes de datos remotas, como tablas en SQL Servidor. LINQ ofrece los beneficios de la verificación de tipos en tiempo de compilación y ejecución. Todos los tipos principales se definen en System.Linq y System.Linq.Expressions (Ferracchiati, 2008).

Figura 1: Lenguaje integrado de consultas



Fuente: C# Corner, por Deepak Dwij, 2019, <https://www.c-sharpcorner.com/UploadFile/72d20e/concept-of-linq-with-C-Sharp/>. Copyright [2020].

Cuando se extraen datos desde una aplicación, ésta consta en crear la conexión y sentencias SQL para extraer los datos desde un medio de almacenamiento ya sea esta un archivo XML, objetos de datos o algún proveedor de datos, esto implica aprender cada uno de los elementos que intervienen al momento de crear el software para las necesidades del usuario. Se puede extraer los datos y almacenarlos en un dataset's, el cual nos permite mantener los datos en memoria cache y así evitar realizar varias peticiones al servidor, pero por así mismo éste objeto no ofrece facilidades para generar consultas sobre estos datos recuperados (Alfonso et al., 2008).

LINQ to DataSet facilita y acelera las consultas en datos almacenados en caché en un objeto DataSet. Esas consultas se expresan en el lenguaje de programación mismo, en lugar de cadenas incrustadas en el código de la aplicación. Esto significa que los desarrolladores no tienen que aprender un lenguaje de consultas diferente.



2.2.3. Consultas LINQ

LINQ, o Language Integrated Query, es un conjunto de características de lenguaje y marco para escribir consultas estructuradas de tipo seguro sobre colecciones de objetos locales y datos remotos fuentes. LINQ se introdujo en C # 3.0 y Framework 3.5.

LINQ le permite consultar cualquier colección que implemente `IEnumerable <T>`, ya sea una matriz, lista o DOM XML, así como fuentes de datos remotas, como tablas en el servidor SQL. LINQ ofrece los beneficios de la verificación de tipos en tiempo de compilación y dinámica. Todos los tipos principales se definen en los namespace `System.Linq` y `System.Linq.Expressions` ("Alonzo Church, 2019).

2.2.3.1. Expresión lambda

El concepto de expresión lambda proviene del trabajo formulado por el matemático-lógico norteamericano Alonzo Church ("Alonzo Church - Wikipedia, la enciclopedia libre," 2019) en su cálculo lambda (λ -calculus) de 1936 que consiste en "...un sistema formal diseñado para investigar la definición de función, la noción de aplicación de funciones y la recursión..." ("Lambda calculus - Wikipedia," 2019).

Existen varios conceptos sinónimos de las expresiones lambda, entre ellos:

- ✓ Función anónima
- ✓ Función literal
- ✓ Abstracción lambda

sin embargo, a este tipo de funciones se le denomina "expresión lambda", más aún, en el contexto de la ciencia de la computación.

Una expresión lambda es una función anónima que no posee un identificador (o nombre) específico. Algo como las siguientes operaciones aritméticas básicas, tal como se muestra a continuación:



Función convencional:

```
SumaCuadrados(x, y) = x*x + y*y
```

Función lambda:

```
(x, y) → x*x + y*y
```

El cuerpo de una declaración lambda puede constar de cualquier número de declaraciones; sin embargo, en la práctica, generalmente no hay más de dos o tres.

```
int[] numbers = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };  
int impar = numbers.Count(n => n % 2 == 1);  
Console.WriteLine($"Hay { impar } números impares {string.Join(" ", numbers)}");
```

2.2.3.2. Operadores básicos

Las unidades básicas de datos en LINQ son secuencias y elementos. Una secuencia es cualquier objeto que implementa `IEnumerable <T>` y un elemento es cada ítem de la secuencia. En el siguiente ejemplo, "names" es la secuencia y "Tom", "Dick" y "Harry" son elementos:

```
string [] names = {"Tom", "Dick", "Harry"};
```

Llamamos a esto una secuencia local porque representa una colección local de objetos en memoria.

Un operador de consulta es un método que transforma una secuencia. Un operador de consulta típico acepta una secuencia de entrada y emite una secuencia de salida transformada. En el 'enumerable class' en System.Linq, hay alrededor de 40 operadores de consulta, todos implementados como métodos de extensión estática, estos se llaman operadores de consulta estándar (Albahari & Albahari, 2012).

Por ejemplo, se puede aplicar el operador 'Where' en una matriz simple para extraer aquellos datos cuya longitud es de al menos cuatro caracteres, de la siguiente manera:

```
using System;

using System.Collections.Generic;

using System.Linq;

class LinqDemo

{

    static void Main()

    {

        string[] names = { "Tom", "Dick", "Harry" };

        IEnumerable<string> filteredNames = names.Where (n => n.Length >= 4);

        foreach (string name in filteredNames)

            Console.WriteLine (name);

    }

}

Salida:

Dick

Harry
```

2.2.3.3. Sintaxis fluida

La sintaxis fluida es la más flexible y fundamental. Aquí se describe cómo encadenar operadores de consultas para formar consultas más complejas y mostrar por qué los métodos de extensión son importantes para este proceso. También se describe cómo formular expresiones lambda para un operador de consulta e introducir varios operadores de consulta nuevos.

2.2.3.4. Operadores de consultas en cadena

Para crear consultas más complejas, agregue operadores de consulta adicionales a la expresión, creando una cadena. Para ilustrar, la siguiente consulta extrae todas las

cadenas que contienen la letra "a", las ordena por longitud y luego convertir el resultado en mayúsculas:

```
using System;

using System.Collections.Generic;

using System.Linq;

class LinqDemo

{

    static void Main()

    {

        string[] names = { "Tom", "Dick", "Harry", "Mary", "Jay" };

        IEnumerable<string> query = names

            .Where (n => n.Contains ("a"))

            .OrderBy (n => n.Length)

            .Select (n => n.ToUpper());

        foreach (string name in query)

            Console.WriteLine (name);

    }

}

Salida:

JAY

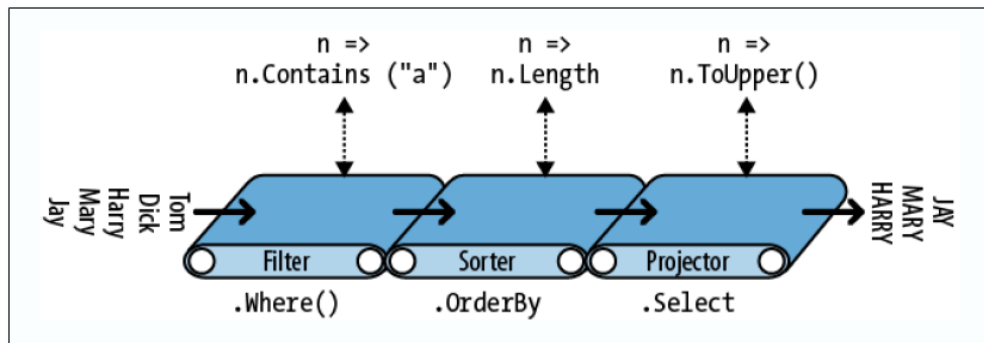
MARY

HARRY
```

La variable 'n', tiene un ámbito privado para cada uno de Las expresiones lambda razón por la cual se reutiliza.

Where, OrderBy y Select son operadores de consulta estándar que se encuentran estos métodos de extensión en la clase Enumerable. El operador Where, emite una versión filtrada de la entrada, el operador OrderBy emite una versión ordenada de su secuencia de entrada; el método de selección emite una secuencia donde cada elemento de entrada se transforma o proyecta con una expresión lambda dada (n.ToUpper (), en este caso) (Albahari & Albahari, 2012).

Figura 2: Filtros en expresiones LINQ



Fuente: C# 5 in a nutshell, (Albahari & Albahari, 2012)

Se puede construir de forma idéntica en forma progresiva de la siguiente manera:

```
IEnumerable<string> filtered = names .Where (n => n.Contains ("a"));  
IEnumerable<string> sorted = filtered.OrderBy (n => n.Length);  
IEnumerable<string> finalQuery = sorted .Select (n => n.ToUpper());
```

2.2.4. Expresiones de consulta

C # proporciona un acceso directo sintáctico para escribir consultas LINQ, llamadas expresiones de consulta. Contrariamente a la creencia popular, una expresión de consulta no es un medio para incrustar SQL en C #. De hecho, el diseño de las expresiones de consulta se inspiró principalmente en la lista comprensión de lenguajes de programación funcionales como LISP y Haskell, aunque SQL tuvo una influencia cosmética (Albahari & Albahari, 2012).



```
using System;

using System.Collections.Generic;

using System.Linq;

class LinqDemo

{

    static void Main()

    {

        string[] names = { "Tom", "Dick", "Harry", "Mary", "Jay" };

        IEnumerable<string> query =

            from n in names

                where n.Contains ("a") // Filter elements

                orderby n.Length // Sort elements

                select n.ToUpper(); // Translate each element (project)

        foreach (string name in query)

            Console.WriteLine (name);

    }

}

SALIDA:

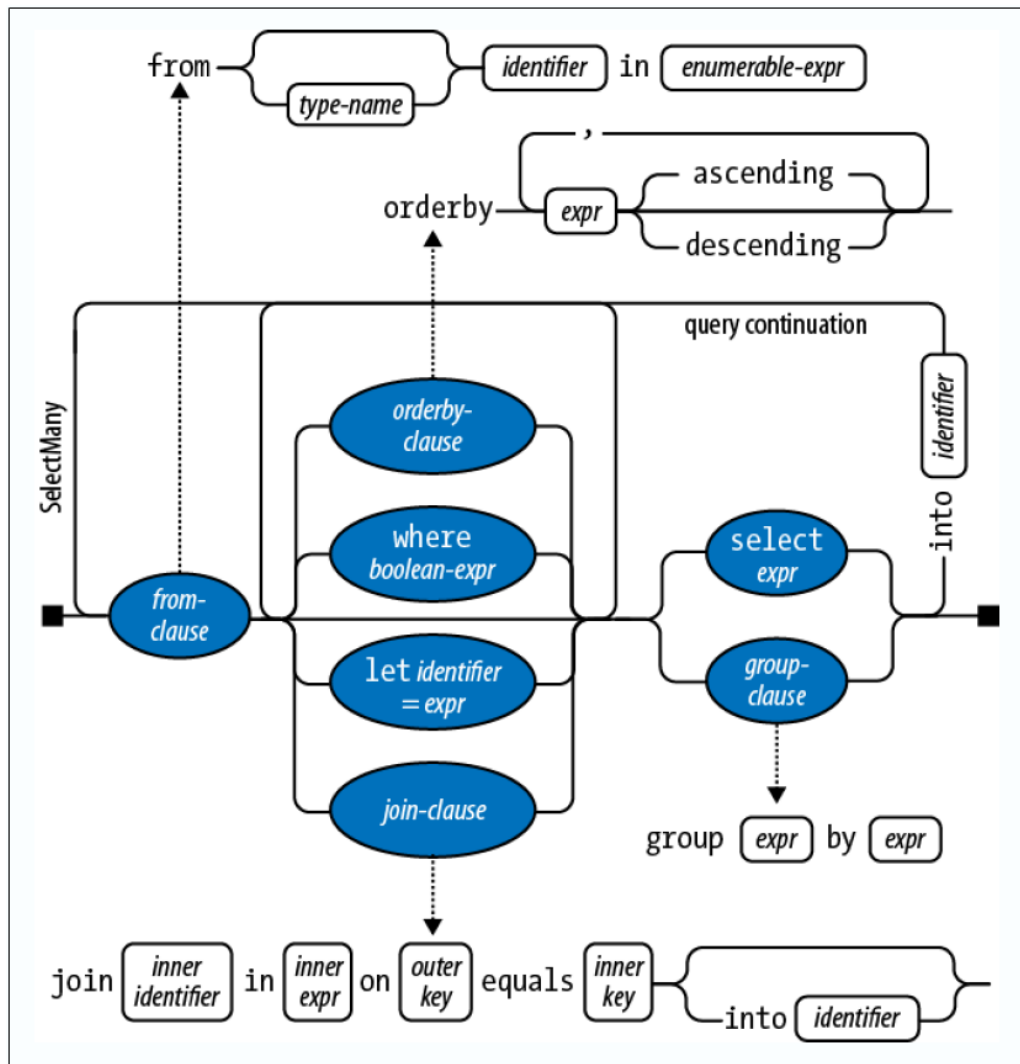
JAY

MARY

HARRY
```

Las expresiones de consulta siempre comienzan con una cláusula 'from' y terminan con 'select' o cláusula de 'group'. La cláusula from declara una variable de rango (en este caso, n), que puede considerarse como atravesar la secuencia de entrada, más bien como foreach. La siguiente figura ilustra la sintaxis completa como diagrama de ferrocarril.

Figura 3: Sintaxis de consultas en expresiones LINQ



Fuente: C# 5 in a nutshell, Sintaxis de consulta, (Albahari & Albahari, 2012)

2.2.4.1. Rango de variable

El identificador que sigue inmediatamente a la sintaxis de la palabra clave 'from' se llama rango variable. Una variable de rango se refiere al elemento actual en la secuencia de operación que se debe realizar. En el siguiente ejemplo, la variable de rango 'n' aparece en cada cláusula de la consulta. Y todavía, la variable realmente enumera sobre una secuencia diferente con cada cláusula:

```
from n in names // n is our range variable
where n.Contains("a") // n = directly from the array
orderby n.Length // n = subsequent to being filtered
select n.ToUpper() // n = subsequent to being sorted
```

Esto queda claro cuando examinamos la traducción mecánica del compilador a fluido sintaxis:

```
names.Where (n => n.Contains ("a")) // Locally scoped n  
  
.OrderBy (n => n.Length) // Locally scoped n  
  
.Select (n => n.ToUpper()) // Locally scoped n
```

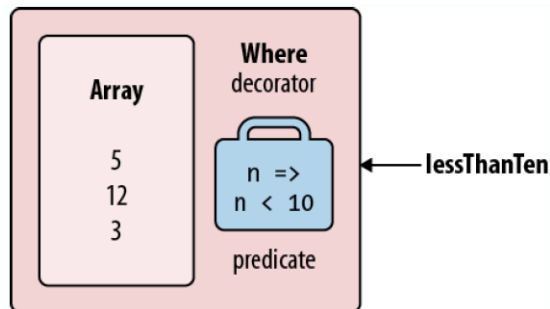
2.2.4.2. Cómo funciona la ejecución diferida

Los operadores de consulta proporcionan ejecución diferida al devolver secuencias *decorator*. A diferencia de una clase de colección tradicional, como una matriz o una lista vinculada, un *decorator* no tiene una estructura de respaldo propia para almacenar elementos. En lugar, envuelve otra secuencia que se proporciona en tiempo de ejecución, a la que mantiene una permanente dependencia. Cada vez que solicita datos de un decorador, a su vez debe solicitar datos de la secuencia de entrada envuelta.

Sea la siguiente consulta:

```
IEnumerable<int> lessThanTen = new int[] { 5, 12, 3 }.Where (n => n < 10);
```

Figura 4: Decoradores de consultas



Fuente: Decoradores de consulta, (Albahari & Albahari, 2012)

El enumerador `lessThanTen`, está consultando la matriz a través del decorador `Where`.

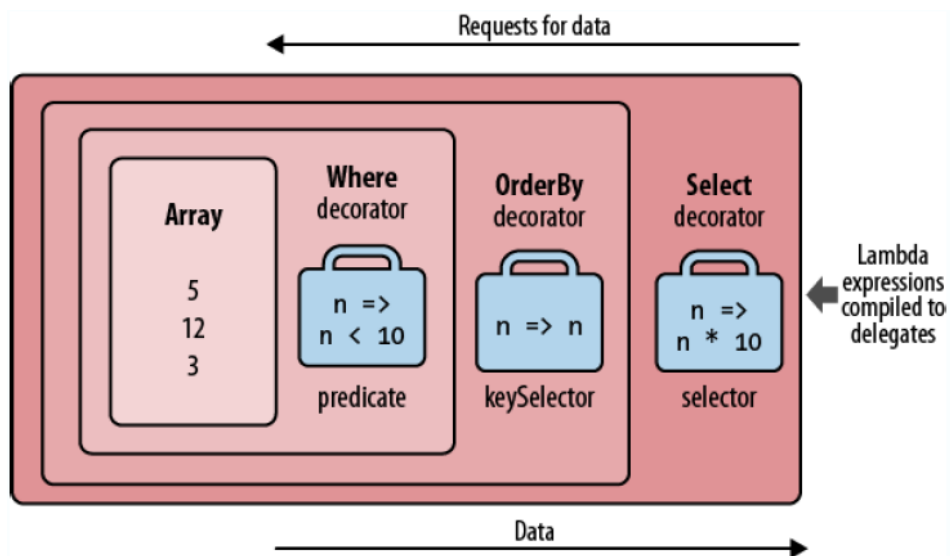
2.2.4.3. Decoradores encadenados

El encadenamiento de operadores de consulta crea una estratificación de decoradores. Considerar la siguiente consulta:

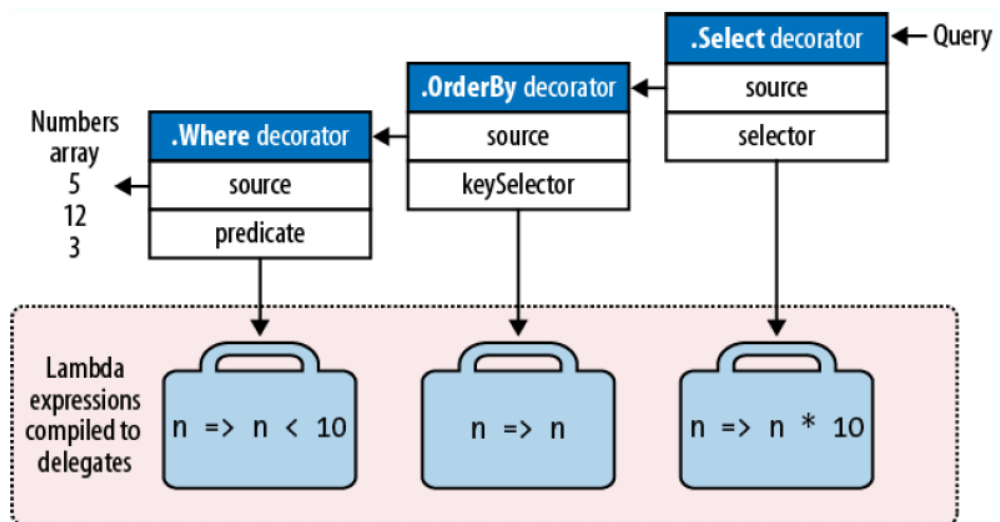
```
IEnumerable<int> query = new int [] { 5, 12, 3 } .Where (n => n < 10)  
                                         .OrderBy (n => n)  
                                         .Select (n => n * 10);
```

Cada operador de consulta crea una instancia de un nuevo decorador que envuelve la secuencia anterior (más bien como una muñeca rusa de anidación). El modelo de objeto de esta consulta se ilustra en Figura siguiente. Se debe tener en cuenta que este modelo de objeto está completamente construido antes de cualquier enumeración.

Figura 5: Las capas y los decoradores



...continuación



Fuente: Secuencia de capas de decoradores, (Albahari & Albahari, 2012)

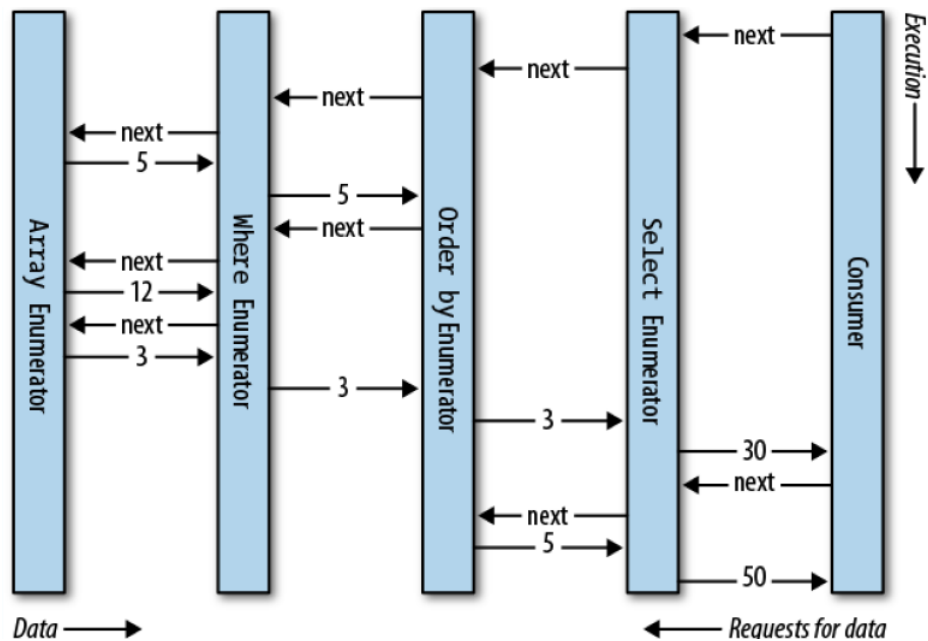
2.2.4.4. Cómo se ejecutan las consultas

Estos son los resultados de enumerar la consulta anterior:

```
foreach (int n in query) Console.WriteLine (n);  
  
Salida:  
  
30  
  
50
```

El resultado es una cadena de enumeradores que refleja estructuralmente la cadena de secuencias decorativas. La siguiente figura ilustra el flujo de ejecución a medida que avanza la enumeración.

Figura 6: Secuencia de ejecución de consultas



Fuente: Secuencia ejecución de los decoradores, (Albahari & Albahari, 2012)

2.2.5. Operadores de consulta estándar LINQ

Aquí se analiza el nivel de consulta de los operadores de consulta. Estos operadores contienen la sintaxis básica tanto de C# como en Visual Basic. A continuación, están los operadores utilizados con mayor frecuencia, tienen un dedicado lenguaje y sintaxis con palabras clave, razón por la cual son usados como parte de expresiones de consultas (Albahari & Albahari, 2012).

2.2.6. LINQ para objetos

Linq para Objeto se extiende para todo tipo de objetos que vengan de IEnumerable (desde una gran colección de clases en .Net, o hasta un simple arreglo de List<T>), para soportar los operadores de consulta similares a los disponibles en SQL. Se puede escribir consultas usando algún built-in Estandar Query Operators, o añadiendo nuestros propios operadores si fueran necesarios. Los operadores estándares cubren una amplia variedad de categorías, en la actualidad existen más de secuencia que forman parte de LINQ. Para

tener una idea de su alcance, aquí esta una lista de los operadores a nuestra disposición (Albahari & Albahari, 2012).

Tabla 1: Operadores de consultas

Lista de operadores de consulta estándar LINQ	
Aggregate	Aplica una función a una secuencia, manteniendo un único valor
All	Aplica una función a una secuencia para ver si todos los elementos satisfacen la función
Any	Aplica una función a una secuencia para ver si solo un elemento satisface la función
Average	Calcula el promedio de una secuencia de números
Cast	Transforma una secuencia de elementos de un tipo a otro
Contains	Busca dentro de una secuencia para ver si esta contiene un elemento dado
ElementAt	Retorna el elemento de la secuencia de la posición especificada
EqualAll	Compara dos secuencias que sean iguales
First	Retorna el primer elemento de la secuencia
FirstOrDefault	Retorna el primer elemento de la secuencia, o el valor predeterminado de la secuencia si esta es vacía
GroupBy	Agrupar los elementos de la secuencia por clave
GroupJoin	Une dos secuencias S1 y S2, y agrupa jerárquicamente los elementos de las mismas
Intersect	Dado dos secuencias S1 y S2, retorna el conjunto de la intersección entre S1 y S2
Last	Retorna el último elemento de una secuencia dada

Elaborado por el equipo de trabajo

2.2.7. Subconsultas

Una subconsulta es una consulta contenida dentro de la expresión lambda de otra consulta. El siguiente ejemplo utiliza una subconsulta para ordenar a los músicos por su apellido:

```
string[] musicos = { "David Gilmour", "Roger Waters", "Rick Wright", "Nick Mason" };
IEnumerable<string> query = musicos.OrderBy (m => m.Split().Last());
```

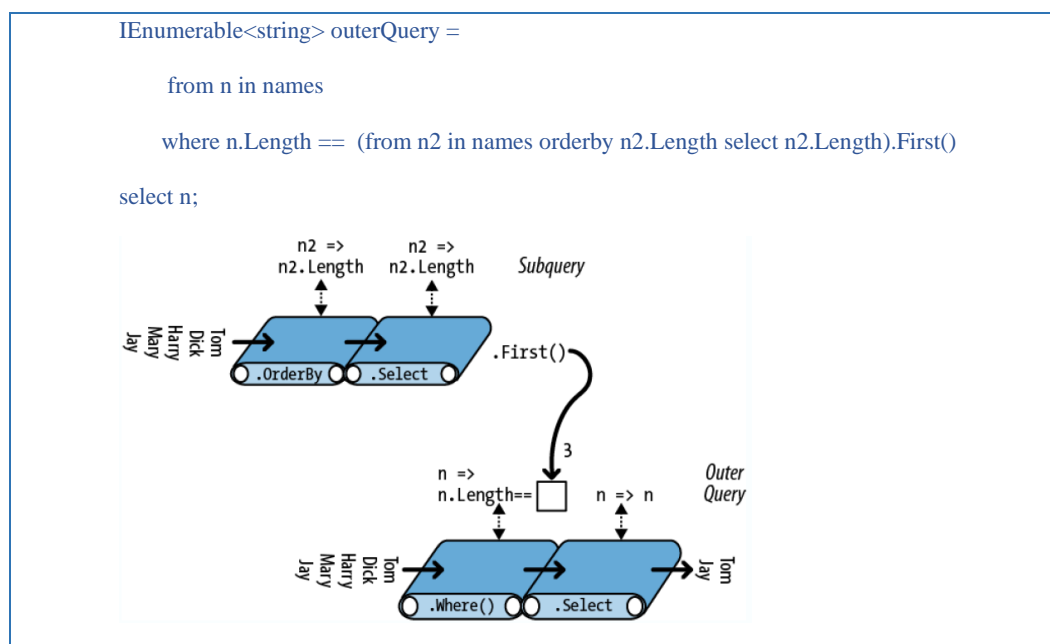
m.Split convierte cada cadena en una colección de palabras, sobre la cual llamamos al operador de consulta Last. 'm.Split ().Last' es la subconsulta, consulta que hace referencia al exterior de la consulta.

Una subconsulta tiene un ámbito privado para la expresión adjunta y puede hacer referencia a los parámetros en la expresión lambda externa. La siguiente consulta recupera todas las cadenas en un matriz cuya longitud coincide con la de la cadena más corta:

```
string[] names = { "Tom", "Dick", "Harry", "Mary", "Jay" };
IEnumerable<string> outerQuery = names
    .Where (n => n.Length == names.OrderBy (n2 => n2.Length)
    .Select (n2 => n2.Length).First());
Salida:
Tom, Jay
```

Ahora, la misma consulta con una expresión de consulta:

Figura 7: Ejecución de subconsultas



Fuente: (Albahari & Albahari, 2012). Secuencia ejecución de las subconsultas.

Con la agregación 'Min' se puede obtener el mismo resultado:



```
IEnumerable<string> query =  
    from n in names  
    where n.Length == names.Min (n2 => n2.Length)  
    select n;
```

2.2.8. Tipos anónimos

Los tipos anónimos le permiten estructurar sus resultados intermedios sin escribir clases especiales podemos eliminar la clase TempProjectionItem en el siguiente ejemplo con tipos anónimos:

```
class TempProjectionItem  
{  
    public string Original; // Original name  
    public string Vowelless; // Vowel-stripped name  
}  
string[] names = { "Tom", "Dick", "Harry", "Mary", "Jay" };  
IEnumerable<TempProjectionItem> temp =  
    from n in names  
    select new TempProjectionItem  
    {  
        Original = n,  
        Vowelless = n.Replace("a", "").Replace("e", "").Replace("i", "")  
            .Replace("o", "").Replace("u", "")  
    };
```

El resultado de este tipo `IEnumerable<TempProjectionItem>`, que posteriormente se puede consultar:

```
IEnumerable<string> query = from item in temp  
    where item.Vowelless.Length > 2  
    select item.Original;  
Salida:  
Dick  
Harry
```

Mary

Trabajando con tipos anónimos se tendría el mismo resultado con la siguiente expresión:

```
var intermediate = from n in names
    select new
    {
        Original = n,
        Vowelless = n.Replace("a", "").Replace("e", "").Replace("i", "")
            .Replace("o", "").Replace("u", "")
    };
IEnumerable<string> query = from item in intermediate
    where item.Vowelless.Length > 2
    select item.Original;
foreach (string n in query)
    Console.WriteLine(n);
```

2.2.9. Microsoft Visual Studio .NET

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para Windows, Linux y macOS. Es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc., a lo cual hay que sumarle las nuevas capacidades en línea bajo Windows Azure en forma del editor Monaco.

Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno compatible con la plataforma .NET (a partir de la versión .NET 2002). Así, se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y

videoconsolas, entre otros (*Microsoft Visual Studio - Wikipedia, La Enciclopedia Libre, n.d.*).

Figura 8: Visual Studio .NET



Fuente: Microsoft Visual Studio, <https://www.msn.com/es-cl/noticias/microsoftstore/%C2%BFqu%C3%A9-es-y-para-qu%C3%A9-sirve-visual-studio-2017/ar-AAAnLZL9>. Copyright [2020].

2.2.10. Evolución de Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, y Visual Basic .NET, al igual que entornos de desarrollo web como ASP.NET. Aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se intercomunican entre estaciones de trabajo, páginas web y dispositivos móviles.

Microsoft Visual Studio es una colección completa de herramientas y servicios para desarrollar aplicaciones para equipos de escritorio, la Web, dispositivos y la nube. Tanto si va a crear su primera aplicación para la Tienda Windows como si va a compilar un sitio web compatible con los últimos exploradores, puede aprovechar los conocimientos que ya tiene con el entorno de desarrollo vanguardista que ofrece Visual



Studio para lenguajes .NET, HTML/JavaScript y C++. Para aquellos equipos que trabajen en varias plataformas, Visual Studio proporciona un entorno de colaboración flexible que permite conectar con otras herramientas de desarrollo, como Eclipse y Xcode.

2.2.10.1. Datos generales de Visual Studio

Tabla 2: Información de Visual Studio

Información general	
Lanzamiento inicial	Visual Studio 97. 01 de mayo de 1997
Género	Entorno de Desarrollo Integrado
Sistema operativo	Windows
Plataforma	x86-64, Itanium
Licencia	Propietario
Estado actual	Con soporte
Idiomas	Alemán, Chino, Coreano, Francés, Inglés, Italiano, Portugués y Ruso
Español	Si

Elaborado por el equipo de trabajo

2.2.10.2. Tipos de Visual Studio

Visual Studio 6.0

Visual Studio .net 2002

Visual Studio 2003

Visual Studio 2005

Visual Studio 2008

Visual Studio 2010

Visual Studio 2012

Visual Studio 2013

Visual Studio 2015

Visual Studio 2017



Visual Studio 2019

A partir de la versión 2005 Microsoft ofrece gratuitamente las Express Editions. Estas son varias ediciones básicas separadas por lenguajes de programación o plataforma enfocadas para novatos y entusiastas. Estas ediciones son iguales al entorno de desarrollo comercial, pero sin características avanzadas. Las ediciones que hay son las siguientes:

Visual Basic Express Edition

Visual C# Express Edition

Visual C++ Express Edition

Visual J# Express Edition (Desapareció en Visual Studio 2008)

Visual Web Developer Express Edition (para programar en ASP.NET)

Visual F# (Apareció en Visual Studio 2010, es parecido al J#)

Adicionalmente, Microsoft ha puesto gratuitamente a disposición de todo el mundo una versión reducida de MS SQL Server llamada SQL Server Express Edition cuyas principales limitaciones son que no soporta bases de datos superiores a 4 GB de tamaño, únicamente utiliza un procesador y un Gb de RAM, y no cuenta con el Agente de SQL Server (“Historia de Visual Studio | Visual Studio – Lenguaje de Programación,” 2015.).

2.2.11. Integridad referencial

La integridad referencial es una propiedad deseable en las bases de datos. Gracias a la integridad referencial se garantiza que una entidad (fila o registro) siempre se relaciona con otras entidades válidas, es decir, que existen en la base de datos. Implica que en todo momento dichos datos sean correctos, sin repeticiones innecesarias, datos perdidos y relaciones mal resueltas.

Todas las bases de datos relacionales gozan de esta propiedad gracias a que el software gestor de base de datos vela por su cumplimiento. En cambio, las bases de datos jerárquicas requieren que los programadores se aseguren de mantener tal propiedad en sus programas (Gonzales, 2011).

Para poder establecer una relación entre dos tablas, es necesario asignar un campo en común a las dos tablas. Para este ejemplo, el campo `id_cliente` existe tanto en la tabla cliente como en la tabla venta. La mayoría de las veces, este campo en común debe ser una clave primaria en alguna de las tablas. Vamos a insertar algunos datos en estas tablas.

Tabla 3: Integridad referencial

Tabla cliente	
<code>id_cliente</code>	nombre
1	Juan penas
2	Pepe el Toro

cliente	venta
<code>id_cliente</code>	<code>id_factura</code>
nombre	<code>id_cliente</code>
	cantidad

Tabla venta		
<code>id_factura</code>	<code>id_cliente</code>	cantidad
1	1	23
2	3	39
3	2	81

Elaborado por el equipo de trabajo

Hay dos registros en la tabla cliente, pero existen 3 `id_cliente` distintos en la tabla venta. Habíamos dicho que las dos tablas se relacionan con el campo `id_cliente`, por lo tanto, podemos decir que Juan Penas tiene una cantidad de 23, y Pepe el Toro 81, sin embargo, no hay un nombre que se corresponda con el `id_cliente` 3.

Las relaciones de claves foráneas se describen como relación padre/hijo (en nuestro ejemplo, cliente es el padre y venta es el hijo), y se dice que un registro es huérfano cuando su padre ya no existe.



Cuando en una base de datos se da una situación como esta, se dice que se tiene una integridad referencial pobre (pueden existir otra clase de problemas de integridad). Generalmente esto va ligado a un mal diseño, y puede generar otro tipo de problemas en la base de datos, por lo tanto, debemos evitar esta situación siempre que sea posible (“Integridad referencial en MySQL. Programación en Castellano.,” 2015).

2.2.11.1. Tipos de integridad referencial

Integridad referencial débil

Si en una dupla de R todos los valores de los atributos de K, tiene un valor que no es nulo, entonces debe existir una tupla en S que tome en los atributos de J los mismos valores en los atributos J.

Integridad referencial parcial

Si en una tupla R algún atributo de K toma el valor nulo, entonces debe existir una tupla en S que tome en los atributos de los mismos valores que los atributos de K con el valor no nulo.

Integridad completa

En una tupla de R todos los atributos de K deben de tener el valor nulo o bien todos tienen un valor que no es nulo y entonces debe existir una tupla en S que tome en los atributos de J los mismos valores que toman los de K.

2.2.11.2. Clave foránea

Una clave foránea es simplemente un campo en una tabla que se corresponde con la clave primaria de otra tabla. Para el anterior ejemplo, el campo `id_cliente` en la tabla `venta` es la clave foránea. Nótese que este campo se corresponde con el campo `id_cliente` en la tabla `cliente`, en dónde este campo es la clave primaria.



Las claves foráneas tienen que ver precisamente con la integridad referencial, lo que significa que, si una clave foránea contiene un valor, ese valor se refiere a un registro existente en la tabla relacionada.

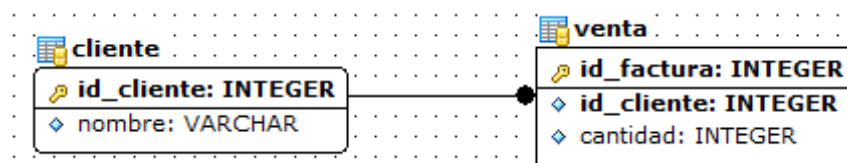
Para trabajar con claves foráneas en MySQL, necesitamos hacer lo siguiente:

- Crear ambas tablas del tipo InnoDB
- Usar la sintaxis `FOREIGN KEY(campo_fk) REFERENCES nombre_tabla (nombre_campo)`
- Crear un índice en el campo que ha sido declarado clave foránea

InnoDB no crea de manera automática índices en las claves foráneas o en las claves referenciadas, así que debemos crearlos de manera explícita. Los índices son necesarios para que la verificación de las claves foráneas sea más rápida. A continuación, se muestra cómo definir las dos tablas de ejemplo con una clave foránea.

```
CREATE TABLE cliente (  
    id_cliente INT NOT NULL,  
    nombre VARCHAR(30),  
    PRIMARY KEY (id_cliente) ) TYPE = INNODB;  
  
CREATE TABLE venta (  
    id_factura INT NOT NULL,  
    id_cliente INT NOT NULL,  
    cantidad INT,  
    PRIMARY KEY(id_factura),  
    INDEX (id_cliente),  
    FOREIGN KEY (id_cliente) REFERENCES cliente(id_cliente)  
    ) TYPE = INNODB;
```


Figura 9: Relación entre tablas



Elaborado por el equipo de trabajo

Las columnas correspondientes en la clave foránea y en la clave referenciada deben tener tipos de datos similares para que puedan ser comparadas sin la necesidad de hacer una conversión de tipos. El tamaño y el signo de los tipos enteros debe ser el mismo. En las columnas de tipo carácter, el tamaño no tiene que ser el mismo necesariamente (“Integridad referencial en MySQL. Programación en Castellano.,” 2015).

2.2.12. Normalización

La normalización es la transformación de las vistas de usuario complejas y del almacén de datos a un juego de estructuras de datos más pequeñas y estables. Además de ser más simples y estables, las estructuras de datos son más fáciles de mantener que otras estructuras de datos. (Kendall, 2005)

Existen varios niveles de normalización de las cuales las más principales son:

2.2.12.1. Primera forma normal

La regla de la Primera Forma Normal establece que las columnas repetidas deben eliminarse y colocarse en tablas separadas.

Poner la base de datos en la Primera Forma Normal resuelve el problema de los encabezados de columna múltiples. La Normalización ayuda a clarificar la base de datos y a organizarla en partes más pequeñas y más fáciles de entender.



2.2.12.2. Segunda forma normal

La regla de la Segunda Forma Normal establece que todas las dependencias parciales se deben eliminar y separar dentro de sus propias tablas. Una dependencia parcial es un término que describe a aquellos datos que no dependen de la llave primaria de la tabla para identificarlos.

Una vez alcanzado el nivel de la Segunda Forma Normal, se controlan la mayoría de los problemas de lógica. Podemos insertar un registro sin un exceso de datos en la mayoría de las tablas.

2.2.12.3. Tercera forma normal

Una tabla está normalizada en esta forma si todas las columnas que no son llave son funcionalmente dependientes por completo de la llave primaria y no hay dependencias transitivas (aquella en la cual existen columnas que no son llave que dependen de otras columnas que tampoco son llave). Cuando las tablas están en la Tercera Forma Normal se previenen errores de lógica cuando se insertan o borran registros. Cada columna en una tabla está identificada de manera única por la llave primaria, y no debe haber datos repetidos. Esto provee un esquema limpio y elegante, que es fácil de trabajar y expandir.

Estas tres formas proveen suficiente nivel de normalización para cumplir con las necesidades de la mayoría de las bases de datos. Normalizar demasiado puede conducir a tener una base de datos ineficiente y hacer a su esquema demasiado complejo para trabajar. Un balance apropiado de sentido común y práctico puede ayudarnos a decidir cuándo normalizar. (Herrera, 2016)



2.2.13. iTextSharp

Normalmente, en los sistemas que se desarrolla, se encuentran con la necesidad de exportar algún tipo de información ya sea un archivo de texto, de excel o PDF. Generar archivos de texto es sumamente fácil ya que .Net provee librerías para hacerlo pero ¿qué pasa cuando necesitamos exportar información a formatos más complejos como excel o PDF?. Una excelente librería para crear y manipular archivos de excel llamada EPPlus, pero para generar archivos PDF en formato estándar se usa la librería iTextSharp.

iTextSharp es una librería, de código abierto (open source) y específicamente para .Net, que nos permite crear y modificar documentos PDF. Si prefiere programar en Java, también se puede beneficiar de iText para Java y Android. Se puede descargar las librerías de los siguientes enlaces:

- .Net: <http://sourceforge.net/projects/itextsharp/>
- Java o Android: <http://sourceforge.net/projects/itext/>

2.2.13.1. Cosas que se hacen con iTextSharp

- Genere documentos e informes basados en datos de un archivo XML o una base de datos
- Crea mapas y libros, explotando numerosas funciones interactivas disponibles en PDF
- Agregue marcadores, números de página, marcas de agua y otras características a los documentos PDF existentes
- Dividir o concatenar páginas de archivos PDF existentes
- Rellena formularios interactivos
- Firmar digitalmente documentos PDF

- Servir documentos PDF generados o manipulados dinámicamente en un navegador web.

2.2.13.2. Versiones

Tabla 4: Versiones de iTextSharp

Versión	Primer lanzamiento	Último lanzamiento	Fin de la vida
0.30 - 0.99	14/02/2000	01/05/2003	31/12/2005
1.00 - 1.4.8	25/06/2003	19/12/2006	31/12/2009
2.00 - 2.1.7	15/02/2003	07/07/2009	31/12/2012
5.0.0 - 5.5.11	07/12/2009	20/03/2017	31/12/2018
7.0.0 - ...	03/05/2016	...	31/12/2025

Elaborado por el equipo de trabajo

2.2.13.3. Creando un archivo

Una vez agregado al proyecto de Visual Studio, se puede generar un archivo pdf con el siguiente código:

```
using iTextSharp.text;

using iTextSharp.text.pdf;

using System.IO;

// Creamos el documento con el tamaño de página tradicional

Document doc = new Document(PageSize.LETTER);

// Indicamos donde vamos a guardar el documento

PdfWriter writer = PdfWriter.GetInstance(doc,

    new FileStream(@"C:\prueba.pdf", FileMode.Create));

// Le colocamos el título y el autor

// Nota: Esto no será visible en el documento

doc.AddTitle("Mi primer PDF");

doc.AddCreator("Roberto Torres");

// Abrimos el archivo

doc.Open();
```

2.2.14. ClosedXML

ClosedXML, es un componente para .NET basado en el estándar OpenXML que permite la generación de archivos Excel, con formato, estilos, fórmulas, rangos, filtros, y casi todo lo que se pueda ocurrir.

ClosedXML, es un proyecto iniciado por Manuel de León y distribuido bajo licencia MIT, se aleja de la verbosidad y amplitud de alcance del Open XML SDK de Microsoft, ofreciendo un API mucho más natural e intuitivo exclusivamente diseñado para crear y manipular documentos Excel. De hecho, el nombre ClosedXML lo eligió después de conocer el SDK oficial y pensar “si es así como se trabaja con Open XML, preferiría utilizar algo que estuviera cerrado”, en referencia a la complejidad que el primero supone. Dado que se basa en Open XML, para abrir los archivos generados se necesita Excel 2007 o una versión posterior.

Figura 10: ClosedXML



Fuente: ClosedXML, <https://www.variablenotfound.com/2013/03/generar-archivos-excel-como-un-senor.html>. licencia CC BY-SA 4.0.

2.2.14.1. Instalación del componente

La instalación de ClosedXML se puede realizar a través de Nuget:

```
PM> Install-Package ClosedXML

Attempting to resolve dependency 'DocumentFormat.OpenXml (≥ 1.0)'.

Successfully installed 'ClosedXML 0.68.1'.

Successfully added 'DocumentFormat.OpenXml 1.0' to ClosedXmlDemo.Model.

Successfully added 'ClosedXML 0.68.1' to ClosedXmlDemo.Model.
```



2.2.14.2. Creando y grabando documentos

La creación y grabado a disco de un documento Excel es absolutamente trivial. Basta con instanciar un objeto de la clase `XLWorkbook`, añadir una nueva hoja a la colección de `Worksheets`, e introducir valores en ella a través de su propiedad `Cell`, como se puede observar en el código genérico mostrado a continuación:

```
var workbook = new XLWorkbook();  
  
var worksheet = workbook.Worksheets.Add("Sheet 1");  
  
worksheet.Cell(1, 1).Value = "Hello, world!";  
  
workbook.SaveAs("c:\\temp\\excel.xlsx");
```

Las líneas anteriores generarán el archivo 'excel.xlsx' con el valor de 'Hello world', en la celda A1 (“Generar archivos Excel como un señor con ClosedXml | Variable not found,” 2019).

2.2.14.3. Integrando closedXML con LINQ

Se puede utilizar LINQ para moverse entre las hojas y celdas, lo cual facilita mucho el trabajo de acceder a zonas concretas del nuestro documento en Excel (“Generación de ficheros ‘Excel’ (xlsx) con ClosedXML,” 2018).

```
using ClosedXML.Excel;  
  
using System.Linq;  
  
namespace PostClosedXML {  
  
    class Program {  
  
        static void Main(string[] args) {  
  
            using (var workbook = new XLWorkbook("HelloWorld.xlsx"))  
            {  
  
                //Buscamos con LinQ la hoja que interesa copiar  
  
                var SampleSheet=workbook.Worksheets.Where(x=> x.Name == "Sample Sheet").First();  
  
                //Añadimos una hoja nuevo  
  
                var worksheet = workbook.Worksheets.Add("FixedBuffer");  
  
                //Copiamos los valores
```



```
worksheet.Cell("A1").Value=SampleSheet.Cell("A1").GetString().ToUpper();  
  
worksheet.Cell("A2").FormulaA1 = SampleSheet.Cell("A2").FormulaA1;  
  
//Guardamos el libro  
  
workbook.Save();  
  
}  
  
}  
  
}  
  
}
```

2.2.15. Lenguaje de programación C#

C# admite los conceptos de encapsulación, herencia y polimorfismo. Todas las variables y métodos, incluido el método Main, el punto de entrada de la aplicación, se encapsulan dentro de las definiciones de clase. Una clase puede heredar directamente de una clase primaria, pero puede implementar cualquier número de interfaces. Los métodos que invalidan los métodos virtuales en una clase primaria requieren la palabra clave `override` como una manera de evitar redefiniciones accidentales. En C#, un `struct` es como una clase sencilla; es un tipo asignado en la pila que puede implementar interfaces pero que no admite herencia.

Además de estos principios básicos orientados a objetos, C# facilita el desarrollo de componentes de software mediante varias construcciones de lenguaje innovadoras, incluidas las siguientes:

- Signaturas de método encapsulado llamadas delegados, que permiten notificaciones de eventos con seguridad de tipos.
- Propiedades, que actúan como descriptores de acceso para variables miembro privadas.
- Atributos, que proporcionan metadatos declarativos sobre tipos en tiempo de ejecución.



- Comentarios de documentación XML insertados
- Language Integrated Query (LINQ), que proporciona funcionalidades de consulta integradas en diversos orígenes de datos.

2.2.16. Evolución de C#

C # es un lenguaje de programación de propósito general y orientado a objetos. La meta del lenguaje es productividad del programador. Para este fin, el lenguaje equilibra simplicidad, expresividad y rendimiento. El principal arquitecto del lenguaje desde su primera versión es Anders Hejlsberg (creador de Turbo Pascal y arquitecto de Delphi). El lenguaje C# es neutral en la plataforma, pero fue escrito para funcionar bien con Microsoft .NET Framework (Albahari & Albahari, 2012).

2.2.16.1. Orientado a objetos

C # es una rica implementación del paradigma orientado a objetos, que incluye encapsulación, herencia y polimorfismo. Encapsulación significa crear un límite alrededor de un objeto, para separar su comportamiento externo (público) de su interno (privado), detalles de implementación. Las características distintivas de C# desde una perspectiva orientada a objetos son:

Sistema de tipo unificado

El bloque de construcción fundamental en C# es una unidad de datos encapsulada y funciones llamadas como tipo. C# tiene un sistema de tipos unificado, donde todos los tipos finalmente comparten un tipo base común. Esto significa que todos los tipos, ya sea que representen los objetos comerciales o son tipos primitivos como los números, y comparten el mismo elemento básico del conjunto de funcionalidades. Por ejemplo, una instancia



de cualquier tipo se puede convertir en una cadena llamando a su método ToString.

Clases e interfaces

En un paradigma tradicional orientado a objetos, el único tipo es una clase. En C# hay varios otros tipos, uno de los cuales es una interfaz. Una interfaz es como una clase, excepto que solo describe miembros. Las implementaciones para esos miembros provienen de tipos que implementan la interfaz. Las interfaces son particularmente útiles en escenarios donde se requiere herencia múltiple (a diferencia de lenguajes como C++ y Eiffel, C# no admite la herencia múltiple de clases).

Propiedades, métodos y eventos

En el paradigma puro orientado a objetos, todas las funciones son métodos (este es el caso en Smalltalk). En C#, los métodos son sólo un tipo de miembro de función, que también incluye propiedades y eventos (también hay otros). Las propiedades son funciones miembros que encapsulan una parte del estado de un objeto, como el color de un botón o el texto de una etiqueta. Los eventos son miembros de funciones que simplifican la acción sobre los cambios de estado del objeto.

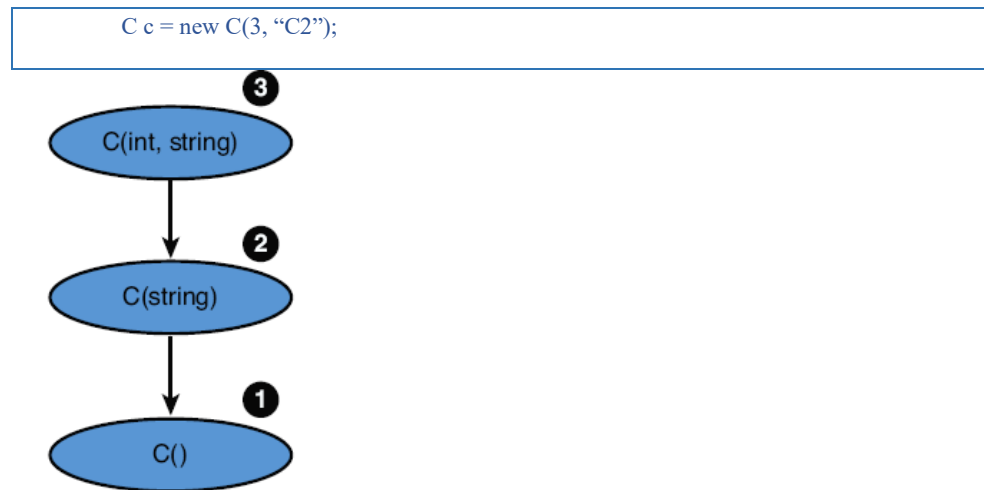
```
public class C {  
    string c1;  
    string c2;  
    int c3;  
    public C() {  
        Console.WriteLine("Default constructor");  
    }  
    public C(int i, string p1) : this(p1) {
```

```
        Console.WriteLine(i);  
    }  
    public C(string p1) : this() {  
        Console.WriteLine(p1);  
    }  
}
```

Orden de ejecución

El orden de ejecución de las anteriores líneas de código con sus respectivos constructores es el siguiente:

Figura 11: Secuencia de ejecución del constructor



Elaborado por el equipo de trabajo

2.2.17. Arquitectura de la plataforma .NET Framework

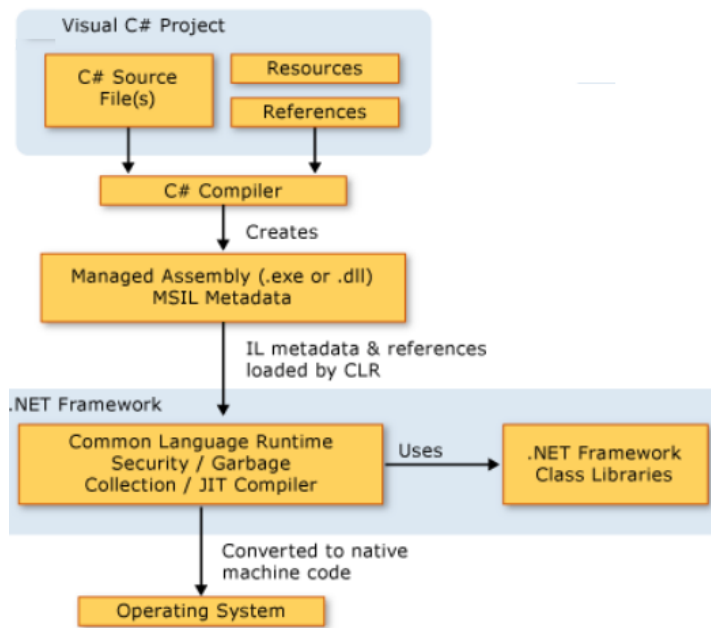
Los programas de C# se ejecutan en .NET Framework, un componente integral de Windows que incluye un sistema de ejecución virtual denominado Common Language Runtime (CLR) y un conjunto unificado de bibliotecas de clases. El CLR es la implementación comercial de Microsoft de Common Language Infrastructure (CLI), un estándar internacional que es la base para la creación de entornos de ejecución y desarrollo en los que los lenguajes y las bibliotecas trabajan juntos sin problemas.



El código fuente escrito en C# se compila en un lenguaje intermedio (IL) que guarda conformidad con la especificación de CLI. El código y los recursos IL, como mapas de bits y cadenas, se almacenan en disco en un archivo ejecutable denominado ensamblado, normalmente con la extensión .exe o .dll. Un ensamblado contiene un manifiesto que proporciona información sobre los tipos, la versión, la referencia cultural y los requisitos de seguridad del ensamblado.

Cuando se ejecuta el programa de C#, el ensamblado se carga en el CLR, el cual podría realizar diversas acciones en función de la información en el manifiesto. Luego, si se cumplen los requisitos de seguridad, el CLR realiza la compilación Just-In-Time (JIT) para convertir el código IL en instrucciones máquina nativas. El CLR también proporciona otros servicios relacionados con la recolección de elementos no utilizados, el control de excepciones y la administración de recursos. El código que se ejecuta en el CLR se conoce a veces como "código administrado", a diferencia del "código no administrado" que se compila en lenguaje de máquina nativo destinado a un sistema específico. En el siguiente diagrama se ilustran las relaciones de tiempo de compilación y tiempo de ejecución de archivos de código fuente de C#, las bibliotecas de clases de .NET Framework, los ensamblados y el CLR.

Figura 12: El lenguaje C#



Fuente: Microsoft.NET. Introducción al Lenguaje C# y .NET Framework, <https://docs.microsoft.com/es-es/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework>. [2015]

2.2.18. La interoperabilidad entre lenguajes

La interoperabilidad entre lenguajes es una característica fundamental de .NET Framework. Debido a que el código IL generado por el compilador de C# cumple la especificación de tipo común (CTS), este código puede interactuar con el código generado a partir de las versiones .NET de Visual Basic, Visual C++ o cualquiera de los más de 20 lenguajes compatibles con CTS. Un solo ensamblado puede contener varios módulos escritos en diferentes lenguajes .NET y los tipos se pueden hacer referencia mutuamente igual que si estuvieran escritos en el mismo lenguaje.

Además de los servicios de tiempo de ejecución, .NET Framework también incluye una amplia biblioteca de más de 4000 clases organizadas en espacios de nombres que proporcionan una gran variedad de funciones útiles para todo, desde la entrada y la salida de archivos, pasando por la manipulación de cadenas para el análisis XML, hasta controles de formularios Windows Forms. La aplicación de C# típica usa mucho la



biblioteca de clases de .NET Framework para controlar tareas comunes de infraestructura (“Introducción al lenguaje C# y .NET Framework | Microsoft Docs,” 2015).

2.2.19. Características de .Net Framework

El .NET Framework tiene varias características:

2.2.19.1. Soporte de estándares

Microsoft diseñó el .NET Framework de forma que estuviera basado en prácticas y estándares de la industria. Existen especificaciones como el CLI (Common Language Infrastructure) o el lenguaje C#, que han sido estandarizadas por organizaciones como la ISO (International Organization for Standardization), la ECMA (European Computer Manufacturers Association) o la IEC (International Electrotechnical Commission), permitiendo ampliar el uso de la plataforma en otros sistemas operativos.

2.2.19.2. Soporte de varios lenguajes de programación

Las aplicaciones .NET pueden ser creadas usando diferentes tipos de lenguajes como C#, Visual Basic .NET, etc. Además, se permite la integración entre lenguajes, lo cual permite la reutilización de código. Los lenguajes pueden ser desarrollados por Microsoft u otros fabricantes que cumplan con los estándares.

2.2.19.3. Modelo simplificado de desarrollo

Las bibliotecas de .NET no se graban en el registro del sistema, para la implementación de aplicaciones se usa información adicional de los componentes necesarios, la cual es incluida en los bloques de construcción de una aplicación.



2.2.19.4. Jerarquía de clases extensible

La plataforma cuenta con una jerarquía de clases que permite el desarrollo de aplicaciones facilitando el acceso y la extensión de las clases de la forma que sea requerida.

2.2.19.5. Ensamblados

Un ensamblado es el bloque de construcción de las aplicaciones de .NET, está formado por código, recursos y un manifiesto, que provee la información para la ejecución. Un ensamblado se puede guardar en un solo archivo ejecutable, en una sola biblioteca o varios archivos que contengan la información separada en bibliotecas, otros archivos y un archivo de manifiesto. La función del manifiesto es proveer información que describa al ensamblado, contiene elementos como la identidad ensamblado, descripción de las clases y tipos que expone el ensamblado, referencias a otros ensamblados y detalles de seguridad para ejecutar el ensamblado.

2.2.19.6. Espacios de nombres

Un espacio de nombres es un elemento que proporciona una estructura jerárquica a las clases dentro de la plataforma, permitiendo su organización y reduciendo la posibilidad de conflictos. Las clases del .NET Framework se encuentran dentro del espacio de nombres System, el cual se subdivide según la funcionalidad de las clases. Para acceder a un espacio de nombres se debe referenciar al ensamblado que lo contiene.

2.2.19.7. Construcción y Ejecución del Código

El código que produce un lenguaje que se ejecuta y dispone de los servicios del entorno de ejecución de .NET se denomina código administrado, lo cual implica que el código es independiente del procesador. En el proceso de compilación, el código fuente es convertido a un lenguaje intermedio, mediante un compilador del lenguaje. El lenguaje



intermedio se guarda en un ensamblado y representa el código administrado. Posteriormente, se carga el ensamblado y se convierte el lenguaje intermedio en código binario a través del compilador JIT (Just-In-Time) para el procesador en el cual se ejecuta la aplicación. El compilador no convierte todo el código del lenguaje intermedio directamente, sino según se necesite y se almacena en memoria para futuras llamadas.

2.3. SISTEMA ACADÉMICO

Un sistema académico es una herramienta que puede ser aplicada en centros de enseñanza como: institutos, escuelas, colegios, academias, universidades, etc. además el sistema de evaluación (periodos, exámenes, evaluaciones, porcentajes, etc.) se establece con los mismos parámetros y formas de evaluación donde será implementado, pero con la ventaja de que el sistema organiza, administra y sirve como fuente de datos para toda la institución, ya que se establecen roles de trabajo para cada usuario que tiene acceso al sistema. Modernizando de esta forma los procesos académicos de los estudiantes y de la institución (“Sistema academicos,” 2016.).

2.3.1. Actividades del sistema académico

- Administrar y controlar mejor la información de docentes y estudiantes
- Crear y asignar niveles académicos asignado un docente como coordinador
- Crear las diferentes asignaturas y asignar el respectivo docente que la impartirá junto con las aulas donde la impartirá
- Crear los diferentes periodos o áreas de evaluación asignado los porcentajes respectivos de cada periodo o área
- Crear las diferentes evaluaciones y porcentajes de cada periodo o área
- Imprimir colectores vacíos por asignatura para el procesamiento de notas



- Ingreso de notas al sistema por parte del docente que imparte la asignatura o por parte de departamento académico
- Escribir comentarios por alumno sobre su conducta o los resultados académicos obtenidos durante el periodo para ser utilizado por el coordinador
- Generar constancia de notas por estudiantes
- Crear e imprimir las boletas de nota por estudiante
- Entre otras tareas

2.4. GESTIÓN ACADÉMICA

El éxito académico, formativo, investigador, en cualquier institución universitaria, tiene una relación intensa con el desarrollo de su gestión en los procesos académicos que se llevan a cabo día a día, enmarcados siempre en la misión y visión que se tenga como organización. Hay que tener claro dentro de cualquier cargo en que se tomen decisiones los principios gerenciales y de planeación estratégica, se debe estar en sintonía con los diferentes niveles de estructura organizacional de la universidad, autoridades, decanos, jefes de departamento o coordinadores académicos, así como los distintos consejos donde se discuten las políticas a seguir.

La universidad como institución de educación superior, tiene que utilizar los sistemas de información como herramienta estratégica para la toma de decisiones, y de igual forma estar en constante revisión de las tendencias que en materia de educación se estén dando a nivel nacional y mundial, para ajustar sus procesos en busca de una verdadera transformación universitaria.

En cuanto a las bases legales que soportan la gestión académica en la Universidad Nacional Privada San Carlos, se encuentra la Ley Universitaria 30220 (2014) en sus



artículos 31, 55, 67, 68, 69, 70; relacionado a las Normas de Funcionamiento de los Decanatos, Departamentos Académicos y Coordinaciones Académicas.

2.5. GESTIÓN Y PLANEACIÓN ESTRATÉGICA UNIVERSITARIA

Según Castillo y Cabrerizo (2006), destacan qué en el ámbito de la gestión académica, las TIC prestan un gran servicio al posibilitar la automatización y descentralización de la gestión en las instituciones universitarias de una manera integrada y coordinada, evitando la centralización y burocratización. Algunas de las aportaciones que pueden hacer estas tecnologías en la gestión son:

- Mejor coordinación entre los diversos servicios.
- Proporcionar completa información sobre todos los aspectos relacionados con la universidad, sus servicios y titulaciones, a través de un buen espacio web institucional
- Realización de múltiples trámites administrativos desde Internet: matrículas, consulta de notas, control de partidas presupuestarias, otros
- Comunicación ágil de la administración con los estudiantes y con el profesorado a través de los oportunos canales telemáticos. Progresiva sustitución de las comunicaciones en papel
- Sistemas para aumentar la participación de los miembros de la comunidad universitaria, profesores, estudiantes y personal de administración y servicios, sin necesidad de abusar de las reuniones, pudiendo utilizar otros canales virtuales alternativos (Contreras Bustamante, 2013).



2.6. LAS TICS COMO FACTOR CLAVE EN LA GESTIÓN ACADÉMICA

En el contexto de las universidades, actualmente hay coincidencias en reconocer, la influencia de las tecnologías, en la gestión universitaria; así como del impacto y desafío que tiene que hacer frente a los nuevos retos de la gestión académica y administrativa. El estudio de investigación concentrará su perspectiva en la dimensión del desarrollo tecnológico, este es un fenómeno de rápida acumulación de información de conocimiento. Las nuevas tecnologías de información y de las telecomunicaciones ofrecen un potencial enorme de transformación de los sistemas educativos y de difusión de sus aplicaciones productivas; que se logra mediante una actividad sistemática de usos de las capacidades de la mente, (conocida como conocimiento e investigación) lo que hace crucial que su implementación y desarrollo académico demande una gestión administrativa con modelos organizativos, que impliquen calidad para que sus procesos y recursos interactúen con el uso intensivo de la tecnología. Las tecnologías podrían permitir el desenvolvimiento de sistemas de aprendizaje paralelos al sistema educativo formal que ahora conocemos, estos poseerían sus propios títulos y certificaciones; con flexibilidad institucional y adaptada a las necesidades específicas de cada educando. El rol de la gestión administrativa, en la implementación y desarrollo de las tecnologías, demanda sustituir sus modelos organizativos asumiendo una gestión de calidad acreditada, mejora continua y que garantice la excelencia y la competitividad (“Las TICS como factor clave en la gestión académica y administrativa de la universidad | Gestión en el Tercer Milenio,” 2017.).

2.7. PROGRAMACIÓN POR CAPAS

La programación por capas es un estilo de programación, su objetivo es la separación del sistema en distintas partes, en todo caso, capas. Existen muchos enfoques posibles para éste estilo, pero el más común es la programación de 3 capas (Gutierrez, 2014).

2.7.1. Capa de presentación - vista

Es todo lo visible al usuario, se muestra y captura toda la información la necesaria con un mínimo de procesos. Ésta capa se comunica únicamente con la capa de negocios, también se conoce como interfaz gráfica y tiene como característica principal, el ser “amigable” con el usuario (Gutierrez, 2014).

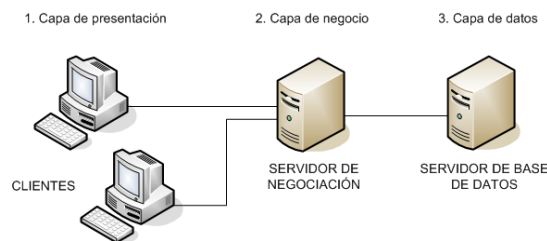
2.7.2. Capa de negocios - controlador

Es donde se reciben las peticiones del usuario (capa de presentación) y se envían las respuestas tras el proceso. Recibe éste nombre o también lógica de negocios, porque aquí se establecen todas las reglas que se deben cumplir. Se comunica con la capa de presentación para recibir solicitudes y presentar resultados y con la capa de datos para almacenar o solicitar datos (Gutierrez, 2014).

2.7.3. Capa de datos - modelo

Es la encargada del almacenado de los datos y la recuperación de los mismos. Está compuesta por uno o más gestores de bases de datos. Se comunica únicamente con la capa de negocios para recibir las peticiones de almacenaje o recuperación de datos (Gutierrez, 2014).

Figura 13: Programación en capas



Fuente: Gutierrez, A. (2014). Programación en capas. <http://pixelg.org/software/item/44-programacion-en-capas.html>

2.7.4. Ventajas

- El desarrollo del sistema puede dividirse en varios niveles
- Facilita la modificación del código fuente

- Proporciona escalabilidad
- Organización

2.7.5. Desventajas

- Complejidad
- Procesos redundantes o innecesarios
- Gasto de espacio en la aplicación

2.8. MAPEO DE DATOS Y LAS TÉCNICAS

Los datos empresariales se están volviendo más dispersos y voluminosos cada día, y al mismo tiempo, se ha vuelto más importante que nunca para que las empresas aprovechen los datos y los transformen en información procesable. Sin embargo, las empresas de hoy recopilan información de una serie de puntos de datos y es posible que no siempre hablen el mismo idioma. Para integrar estos datos y darle sentido, se utiliza la asignación de datos, que es el proceso de establecer relaciones entre modelos de datos separados.

2.8.1. Mapeo de datos

En palabras simples, la asignación de datos es el proceso de asignación de campos de datos de un archivo de origen a sus campos de destino relacionados. Por ejemplo, en la Figura siguiente, los campos 'Nombre', 'Correo electrónico' y 'Teléfono' de una fuente de Excel se asignan a los campos relevantes en un archivo delimitado, que es nuestro destino.

Figura 14: Mapeo de datos



Elaborado por el equipo de trabajo

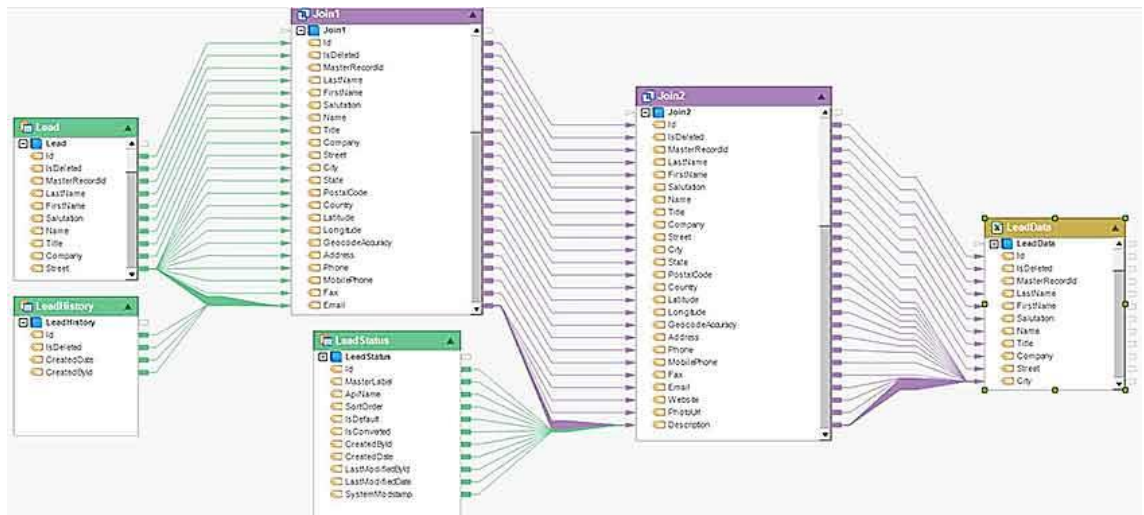


Las tareas de mapeo varían en complejidad, dependiendo de la jerarquía de los datos que se están mapeando, así como de la disparidad entre la estructura de la fuente y el destino. Cada aplicación comercial, ya sea local o en la nube, utiliza metadatos para explicar los campos de datos y los atributos que constituyen los datos, así como las reglas semánticas que rigen cómo se almacenan los datos dentro de esa aplicación o repositorio.

Por ejemplo, Microsoft Dynamics CRM contiene varios conjuntos de datos que forman parte de diferentes objetos, como Leads, Opportunities y Competitors. Cada uno de estos conjuntos de datos tiene varios campos como Nombre, Propietario de la cuenta, Ciudad, País, Título del trabajo y más. La aplicación también tiene un esquema definido junto con atributos, enumeraciones y reglas de mapeo. Por lo tanto, si se va a agregar un nuevo registro al esquema de un objeto de datos, se debe crear un mapa de datos desde el origen de datos a la cuenta de Microsoft Dynamics CRM.

Según el número, el esquema y las claves primarias y externas de las fuentes de datos de las bases de datos relacionales, las asignaciones de bases de datos pueden tener un grado de complejidad variable. Por ejemplo, en el siguiente ejemplo, los datos de tres tablas de bases de datos diferentes se unen y se asignan a un destino de Excel (Fatima, 2018).

Figura 15: Mapeo de datos complejo



Elaborado por el equipo de trabajo

Dependiendo de las necesidades de administración de datos de una empresa y de las capacidades del software de mapeo de datos, el mapeo de datos se utiliza para llevar a cabo una serie de tareas de integración y transformación de datos.

2.8.2. La importancia del mapeo de datos

Para aprovechar los datos y extraer el valor comercial de la misma, la información recopilada de varias fuentes externas e internas se debe unificar y transformar en un formato adecuado para los procesos operativos y analíticos. Esto se logra a través del mapeo de datos, que es un paso integral en varios procesos de administración de datos, que incluyen:

2.8.2.1. Integración de datos

Para una integración de datos exitosa, los repositorios de datos de origen y destino deben tener el mismo modelo de datos. Sin embargo, es raro que dos repositorios de datos tengan el mismo esquema. Herramientas de mapeo de datos ayude a salvar las diferencias en los esquemas de origen de datos y destino, permitiendo a las empresas consolidar información de diferentes puntos de datos fácilmente.



2.8.2.2. Migración de datos

Migración de datos es el proceso de mover datos de una base de datos a otra. Si bien hay varios pasos involucrados en el proceso, la creación de asignaciones entre el origen y el destino es una de las tareas más difíciles y que requieren mucho tiempo, especialmente cuando se realiza manualmente. Las asignaciones inexactas e inválidas en esta etapa no solo afectan la precisión y la integridad de los datos que se migran, sino que incluso pueden conducir al fracaso del proyecto de migración de datos. Por lo tanto, usar una solución de mapeo sin código que pueda automatizar el proceso es importante para migrar los datos al destino con éxito.

2.8.2.3. Almacenamiento de datos

La asignación de datos en un almacén de datos es el proceso de crear una conexión entre las tablas o atributos de origen y destino. Mediante el mapeo de datos, las empresas pueden construir un modelo de datos lógico y definir cómo se estructurarán y almacenarán los datos en el almacén de datos. El proceso comienza con la recopilación de toda la información requerida y la comprensión de los datos de origen. Una vez hecho esto y creado un documento de mapeo de datos, construir las reglas de transformación y crear mapeos es un proceso simple con una solución de mapeo de datos.

2.8.2.4. Transformación de datos

Debido a que los datos empresariales residen en una variedad de ubicaciones y formatos, transformación de datos Es esencial romper los silos de información y dibujar ideas. El mapeo de datos es el primer paso en la transformación de datos. Se realiza para crear un marco de trabajo de los cambios que se realizarán en los datos antes de cargarlos en la base de datos de destino.

2.8.2.5. Intercambio electrónico de datos

El mapeo de datos juega un papel importante en la conversión de archivos EDI al convertir los archivos en varios formatos, como XML, JSON y Excel. Una intuitiva herramienta de mapeo de datos permite al usuario extraer datos de diferentes fuentes y utilizar transformaciones y funciones integradas para asignar datos a formatos EDI sin escribir una sola línea de código. Esto ayuda a realizar un intercambio continuo de datos B2B.

2.8.3. Técnicas de mapeo de datos

Aunque es un paso esencial en cualquier proceso de administración de datos, el mapeo de datos puede ser complejo y requerir mucho tiempo. El proceso de conectar fuentes de datos, crear asignaciones para la transformación e integración de datos, y validar los datos transformados puede ocupar importantes recursos de desarrollador, en particular cuando el proceso completo se realiza manualmente.

Según el nivel de automatización, las técnicas de mapeo de datos se pueden dividir en tres tipos:

2.8.3.1. Mapeo manual de datos

La asignación manual de datos implica la codificación manual de las asignaciones entre la fuente de datos y la base de datos de destino. Aunque el proceso de mapeo manual de datos, codificado a mano, ofrece flexibilidad ilimitada para escenarios de mapeo únicos inicialmente, puede ser un desafío mantener y escalar a medida que las necesidades de mapeo del negocio se vuelven complejas.

2.8.3.2. Mapeo de datos automatizado

El mapeo de esquemas a menudo se clasifica como una técnica de mapeo de datos semiautomatizada. El proceso implica la identificación de dos objetos de datos que están relacionados semánticamente y luego la creación de asignaciones entre ellos. Por ejemplo, para crear asignaciones entre los esquemas de la base de datos 1 y la base de datos 2, las posibles coincidencias que un desarrollador puede usar incluyen Database1: StudentName≈Database2: Name y Database1: ID≈Database2: SSN.

Figura 16: Relación de base de datos automatizada

1 base de datos	2 base de datos
Nombre del estudiante	Nombre y Apellidos
ID	SSN
nivel	Mayor
Mayor	grados
Marcas	

Elaborado por el equipo de trabajo

Una vez que se ha realizado la asignación de esquemas, se genera código Java, C ++ o C # para lograr las tareas de conversión de datos



requeridas. El lenguaje de programación utilizado puede variar según la herramienta de mapeo de datos utilizada.

2.8.3.3. Mapeo automatizado de datos

Las herramientas de mapeo de datos se pueden dividir en tres tipos generales:

- En las instalaciones: dichas herramientas se alojan en el servidor de una empresa y en la infraestructura informática nativa. Muchas herramientas de mapeo de datos en las instalaciones eliminan la necesidad de codificación manual para crear mapeos complejos y automatizan tareas repetitivas en el proceso de mapeo de datos.
- Basado en la nube: estas herramientas aprovechan la tecnología de la nube para ayudar a una empresa a realizar sus proyectos de mapeo de datos.
- Código abierto: las herramientas de mapeo de código abierto proporcionan una alternativa de bajo costo a las soluciones de mapeo de datos en las instalaciones. Estas herramientas funcionan mejor para pequeñas empresas con volúmenes de datos más bajos y casos de uso más simples.



2.9. MARCO CONCEPTUAL

2.9.1. Base de datos

Es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido, una Biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta (*Bases de Datos - EcuRed*, n.d.).

2.9.2. iTextSharp

iTextSharp es una librería, de código abierto (open source) y específicamente para .Net, que nos permite crear y modificar documentos PDF. Si se prefiere programar en Java, también se puede beneficiar de iText para Java y Android (“C#: Creando archivos PDF con iTextSharp,” 2012.).

2.9.3. MySql.Data.dll

Conector que permite interactuar como una interface entre la base de datos MySQL y el lenguaje de programación C#.

2.9.4. Entidad - relación

El modelo entidad-relación ER es un modelo de datos que permite representar cualquier abstracción, percepción y conocimiento en un sistema de información formado por un conjunto de objetos denominados entidades y relaciones, incorporando una representación visual conocida como diagrama entidad-relación (Gonzales, 2011).



2.9.5. Componentes

Los componentes son unos elementos genéricos con una funcionalidad muy concreta, cuya única finalidad es la reutilización. Cada uno de ellos está destinado a realizar una tarea típica en una aplicación (EcuRed, 2014).

2.9.6. Aplicación de escritorio

Una aplicación de escritorio es aquella que se encuentra instalado en el ordenador o sistema de almacenamiento (USB) y podemos ejecutarlo sin internet en nuestro sistema operativo, al contrario que las aplicaciones en la nube que se encuentran en otro ordenador (servidor) al que accedemos a través de la red o internet (Wikipedia, 2020).

2.9.7. Programación

Es el proceso por el cual una persona desarrolla un programa valiéndose de una herramienta que le permita escribir el código el cual puede estar en uno o varios lenguajes, como C++, Java y Python, entre otros.

2.9.8. Software

Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora (“software | Diccionario de la lengua española | RAE - ASALE,” 2015.).

2.9.9. Administración

Es la responsabilidad convencional que tienen los administradores para la empresa o institución. Esto es llevar adelante a la institución a fin de cumplir con las metas programadas, además de buscar y manejar liderazgo. Dentro de la administración existen diferentes niveles de responsabilidades y diferentes tipos de control, además de la parte legal de los actuados.



CAPÍTULO III

MATERIALES Y MÉTODOS

3.1. TRABAJO EXPERIMENTAL

3.1.1. Tipo de investigación

Esta investigación de acuerdo con las características de la hipótesis, los objetivos y la pregunta de investigación, se enmarca dentro del enfoque cuantitativo correlacional, es decir, una investigación que analizó la relación entre las variables mejoramiento del Sistema Académico y el componente LINQ de Microsoft .NET, tal como indica Hernández, R., Fernández, C., & Baptista, (2014): “los estudios correlacionales miden el grado de asociación entre dos o más variables. Es decir, miden cada variable presuntamente relacionada y, después, miden y analizan la correlación”.

3.1.2. Diseño de la investigación

El diseño de la investigación es el cuasi-experimental. Los diseños cuasi-experimentales manipulan al menos, una variable independiente para observar su efecto y relación con una o más variables dependientes.

Según Cardona, (2017); un estudio cuasi-experimental no necesariamente posee dos grupos, el experimental y el de control, esto condujo a elegir un solo grupo experimental al cual se le sometió a una prueba de pretest y postest.

El grupo experimental estuvo conformado por el gestor de base de datos MySQL.

La representación gráfica es la siguiente:

G1: O1 X O2

Donde:

G1 : Grupo experimental



O1 : Prueba(pretest) antes del experimento (consulta SQL)

X : Mejoramiento del Sistema Académico de la UPSC

O2 : Prueba(postest) después del experimento (consulta LINQ)

Este diseño con grupo experimental permitió la comparación de resultados pretest y postest, con la finalidad de medir el tiempo de respuesta de las consultas SQL y expresiones LINQ sobre el motor de base de datos MySQL, y así determinar que el uso del componente LINQ de Microsoft .NET (variable independiente) ha sido el factor determinante en el tiempo de respuesta del Sistema Académico de la UPSC (variable dependiente). Para dichas pruebas se utilizó un grupo de consultas que involucran a una, dos, tres, etc. tablas relacionadas para medir dichos tiempos.

3.1.3. Variables de estudio

Variable independiente: Componente LINQ de Microsoft .NET

Variable dependiente: Sistema Académico de la Universidad Privada San Carlos.

3.1.4. Población

La población de estudio son las consultas tradicionales desarrolladas con SQL y las expresiones LINQ de Microsoft .NET.

3.1.5. Ámbito de estudio

Denominación:

Universidad Privada San Carlos (UPSC)

Fecha de creación:

La Universidad Privada San Carlos de Puno, fue creada mediante Resolución N° 354-2006 del Consejo Nacional para la Autorización de Funcionamiento de



Universidades (CONAFU), de fecha 25 de octubre del 2006, con las carreras profesionales de Contabilidad y Finanzas, Enfermería e Ingeniería Informática.

Ubicación geográfica:

Departamento: Puno

Provincia: Puno

Distrito: Puno

Dirección: Jr. Conde de Lemos N° 128, Puno, Perú

E-mail: informes@upsc.edu.pe

Web: www.upsc.edu.pe

Reglamento (Anexo C)

Ubicación política:

Políticamente se sitúa en el distrito de Puno, provincia de Puno y departamento de Puno.

3.2. TÉCNICAS DE RECOLECCIÓN DE DATOS

Para el presente trabajo de investigación se recurrieron a las siguientes técnicas:

Análisis documental. La técnica mencionada fue aplicada en la Universidad Privada San Carlos de Puno, solicitando a la oficina de Registro Académico el reglamento de matrículas, directivas académicas, manual de usuario del sistema actual, entre otros que son requeridos para confrontar ambas variables de estudio.

Entrevista. Se realizó una entrevista no estructurada al personal encargado de la oficina de registro académico, quienes indicaron verbalmente los reportes que más se usan, las dificultades y necesidades que el sistema actual no contempla (Anexo 2).









Verificación de consultas. Se obtuvieron las consultas más frecuentes usadas en el sistema con la finalidad de establecer tiempos de respuesta en el sistema actual con el componente LINQ.

3.3. MATERIAL EXPERIMENTAL

3.3.1. Herramientas usadas

Para el desarrollo del mejoramiento del Sistema Académico usando el componente LINQ se usaron las siguientes herramientas:

Tabla 5: Herramientas usadas

LinqPad 	MySQL Database versión 5.0.16 
ClosedXML 	Visual Studio 2010 (CSharp) 
Lenguaje de Programación C# 	SQL Manager 2005 for MySQL 
iTextSharp .NET PDF Library 	RStudio 

Elaborado por el equipo de trabajo



3.4. MÉTODOS EXPERIMENTALES

3.4.1. Método de tratamiento de datos

Para el tratamiento de los datos se realizó las siguientes actividades: los datos numéricos se procesaron y se tabularon, luego se construyeron con ellos cuadros estadísticos. Para el procesamiento e interpretación de los datos se utilizó el software estadístico R versión 3.4.

3.4.2. Prueba de hipótesis para dos muestras apareadas

La contrastación de la hipótesis se realizó mediante la prueba de diferencia de medias para datos apareados, el cual permitió aceptar o rechazar la hipótesis. Planteamiento de la hipótesis nula (H_0) y la hipótesis alterna (H_1):

- ✓ H_0 : El Sistema Académico desarrollado en Visual Studio 2010, las expresiones lambda y expresiones SQL del componente LINQ no mejora el tiempo de respuesta en la recuperación de datos sobre el motor de base de datos MySQL de la Universidad Privada San Carlos de Puno.
- ✓ H_1 : El Sistema Académico desarrollado en Visual Studio 2010, las expresiones lambda y expresiones SQL del componente LINQ mejora significativamente el tiempo de respuesta en la recuperación de datos sobre el motor de base de datos MySQL de la Universidad Privada San Carlos de Puno.



CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

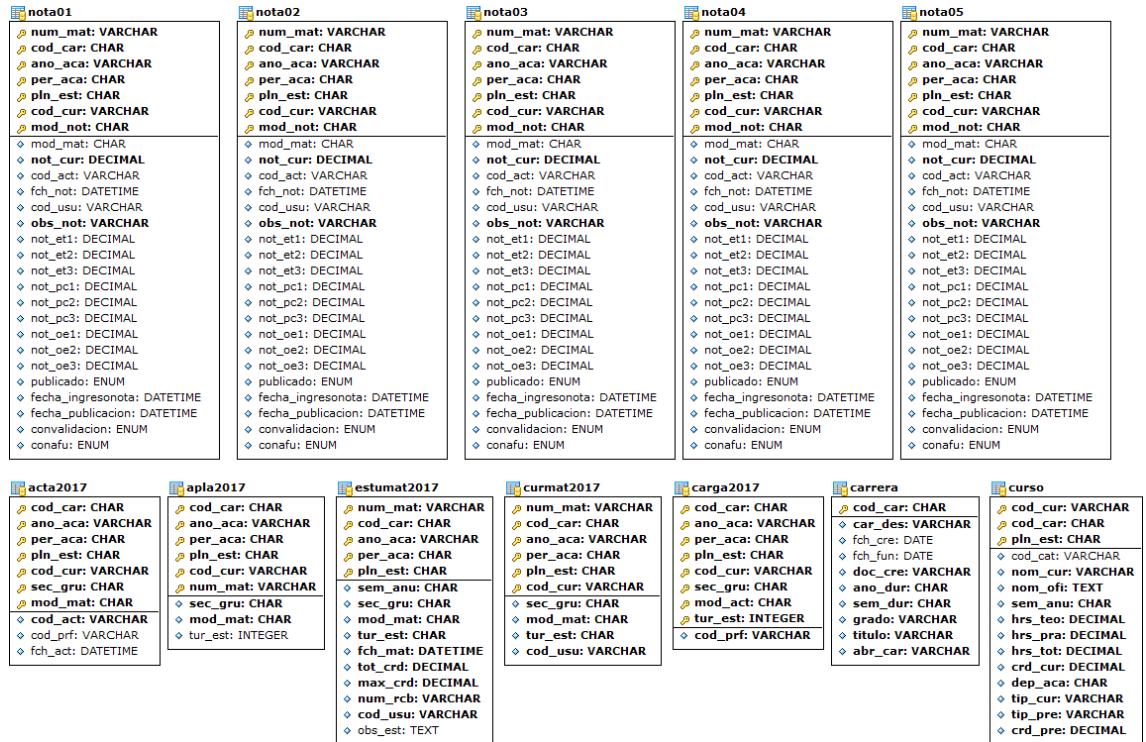
El Sistema Académico para la Universidad Privada San Carlos de Puno, se desarrolló fundamentalmente para mejorar las actividades de la oficina de registro académico, mediante la automatización de los procesos que se dan dentro de la institución; del mismo modo mejorar la gestión administrativa en la oficina mediante el uso del componente LINQ. La creciente demanda por parte de los estudiantes en estos últimos años ha originado nuevas necesidades la cual requiere del uso de nuevas tecnologías de vanguardia a la solución de problemas.

4.1. DIAGRAMA DE BASE DE DATOS

Los diagramas permiten tener una clara idea de la relación que existe entre cada una de las tablas de la base de datos. Los tipos de tablas con los que se trabajaba fue MyISAM y no existía integridad referencia ente ellas, razón por la cual se encontraron datos inconsistentes.

La siguiente figura muestra las tablas más importantes del gestor de base de datos MySql inicial del sistema actual.

Figura 17: Diagrama de base de datos inicial



Elaborado por el equipo de trabajo

4.1.1. Errores encontrados

Se encontró errores de espacios en blanco que no permitían realizar el cruce de datos, ya que este dato fue necesario para enlazar con los códigos de los cursos de la base de datos.

Por ejemplo, en la Escuela Profesional de Ciencias Contables, 2009-1, primer y segundo semestre en los turnos mañana y tarde.

Tabla 6: Errores iniciales en la base de datos

MERY ROSANGELA	INV	0105		18	61 tarde	primero
MERY ROSANGELA	SOC	0106		15	61 tarde	primero
MERY ROSANGELA	ECL	0107		14	61 tarde	primero
RONALD VALERO	MAT	0201		13	13 manana	segundo
RONALD VALERO	INF	0202		14	13 manana	segundo
RONALD VALERO	SOC	0203		13	13 manana	segundo

Elaborado por el equipo

Para detectar los errores se usó la siguiente consulta para convertir las tablas a tipo InnoDB, seguidamente se quiso establecer la relación entre tablas lo que generó un error de integridad referencial.

```
ALTER TABLE tabla ENGINE = INNODB
```

Debido a esta inconsistencia se tuvo que generar un script para determinar qué registros (s) de la tabla estuvo generando tal error. Esto se logró luego de entender la relación entre tablas.

Script de consulta

Figura 18: Script de consulta SQL estándar

```
select P.* FROM (
  select M.*, A.nota AS 'NotaConafu' FROM (
    select distinct n.num_mat, concat(e.paterno, ' ', e.materno) AS 'Apellidos',
    e.nombres AS 'Nombres', c.cod_cat, c.cod_cur, c.sem_anu, n.mod_mat,
    mo.mod_des AS 'Modalidad', c.nom_cur AS 'NombreCurso', n.not_cur as 'NotaSistema'
    from nota01 n

    inner join curmat2007 cm on cm.num_mat=n.num_mat and cm.cod_car=n.cod_car and
    cm.ano_aca=n.ano_aca and cm.per_aca=n.per_aca and cm.pln_est=n.pln_est
    inner join curso c on n.cod_cur = c.cod_cur and n.cod_car = c.cod_car and c.pln_est=n.pln_est
    inner join estudiante e on e.num_mat = cm.num_mat and e.cod_car=cm.cod_car
    INNER JOIN modmat mo ON mo.mod_mat = n.mod_mat
    WHERE n.cod_car='01' and
    n.ano_aca='2007' and
    n.per_aca in ('02')

    and cm.tur_est = '1'
    and cm.sec_gru='02'
    --order by e.paterno, e.materno, e.nombres, c.cod_cat, n.not_cur desc
    ORDER BY e.paterno, e.materno, e.nombres, n.mod_mat desc --(2)
  ) AS M INNER JOIN a2007_2_ciclo2 A ON M.num_mat=A.num_mat and M.cod_cat=A.codcurso
  --WHERE M.NotaSistema <> A.nota
  WHERE A.turno="manana"
  --GROUP BY M.num_mat, Apellidos, M.Nombres, M.cod_cat
  GROUP BY M.num_mat, M.Apellidos, M.Nombres, M.cod_cur, M.sem_anu --(2)
) AS P WHERE P.NotaConafu <> P.NotaSistema
```

Elaborado por el equipo de trabajo

Se encontró números de matrícula con cuatro (04) y cinco (05) dígitos en la base de datos. Lo correcto es que cada número de matrícula contenga seis (06) dígitos.

Número de matrícula con 4 dígitos: 1

Número de matrícula con 5 dígitos: 14





Se encontró cuatro (04) estudiantes con dos códigos de matrícula en la base de datos.

Tabla 7: Errores encontrados en códigos de matrícula

COD. MAT.	NOMBRES	CARRERA PROFESIONAL
93015	APAZA FLORES, NEPTALI LINDON	ING. INFORMÁTICA
91015		CONTABILIDAD Y FINANZAS
124255	SALAS MANZANO, PATRICIA SURAMA	ING. AMBIENTAL
91013		CONTABILIDAD Y FINANZAS
20435	CONDORI ROJAS, MIRIAM MARILIA	ING. AMBIENTAL
91031		CONTABILIDAD Y FINANZAS
71149	TORRES COPA, OSCAR FILIBERTO	ENFERMERÍA
91057		CONTABILIDAD Y FINANZAS
NUM. MAT.	NOMBRES	CARRERA PROFESIONAL
24003	SALAS HINOJOSA, HANS ALBERTO	ING. AMBIENTAL
81065		ING. INFORMÁTICA
94093		ING. AMBIENTAL
20414	ESTRELLA PACA, JAVIER	ING. AMBIENTAL
93002		ING. INFORMÁTICA
94007		ING. AMBIENTAL
20407	RAMIREZ RIOS, MIJAIL	ING. AMBIENTAL
93004		ING. INFORMÁTICA
72224	PEREZ NAHUINCHA, ROXANA AYDE	CIENCIAS CONTABLES
81007		ING. INFORMÁTICA
91087		CIENCIAS CONTABLES
COD. MAT.	NOMBRES	CARRERA PROFESIONAL
124238	COTRADO CCALLI, WALTER	ING. AMBIENTAL
201100		CIENCIAS CONTABLES
120230	AGUILAR HUMPIRI, CLAUDIO LUIS	ING. AMBIENTAL
124300		ING. AMBIENTAL
94028	IBEROS PARI, ADERLY DAVID	ING. AMBIENTAL
124303		ING. AMBIENTAL
20471	ARIAS QUISPE, JULIO CESAR	CIENCIAS CONTABLES
94030		ING. AMBIENTAL
121166	CALCINA SIRENA, LUZ EVA	CIENCIAS CONTABLES
124106		ING. AMBIENTAL
81069	CARI RIVERA, YOSEP ENZOM	ING. INFORMÁTICA
94038		ING. AMBIENTAL

Elaborado por el grupo de trabajo

En la sección de color  de la tabla, se encontró cuatro (04) estudiantes con tres (03) y dos (02) códigos de matrícula (cada uno), en la base de datos del semestre 2009-1.

En la sección de color  de la tabla, se encontró seis (06) estudiantes con dos códigos de matrícula en la base de datos.


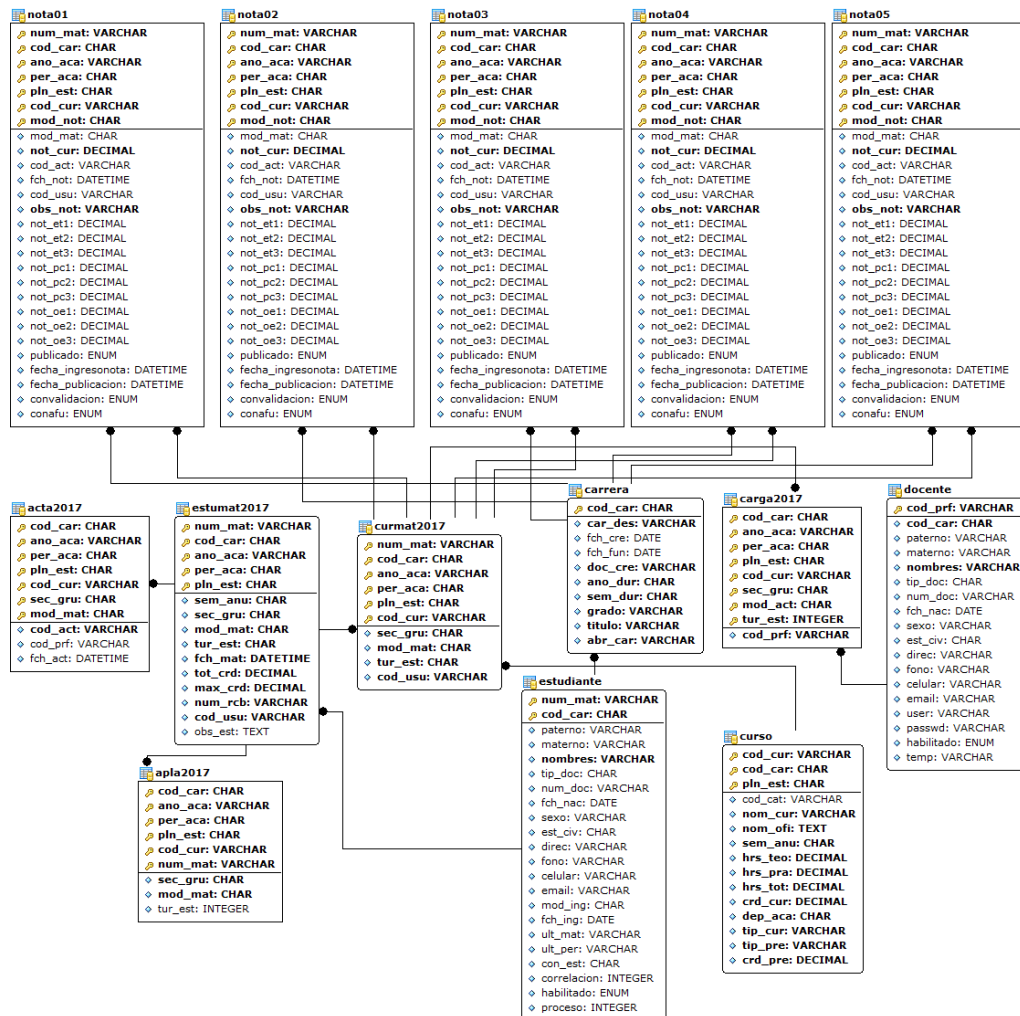
En la sección de color  de la tabla, luego de constatar los errores se procedió a realizar las modificaciones de los registros errados, se procedió a cambiar el tipo de tabla a InnoDB con la finalidad de establecer la integridad referencial entre ellas y así evitar la inconsistencia de datos. El resultado fue la siguiente figura.

Figura 19: Base de datos final con integridad referencias



Elaborado por el equipo de trabajo

4.2. CAPA DE ACCESO A DATOS

En esta capa se hizo uso de una propiedad denominada **persistencia de objetos**, que permitió vincular objetos de la base de datos relacional (MySQL) a objetos de lenguaje de programación C#, para aumentar el nivel de abstracción y facilitar el acceso a los datos desde la capa de negocio.

En la figura se observa parte de la representación del modelo de clases para usarlo con LINQ a través del diseñador en C#.

Figura 20: Tipos de datos de objetos

```
class Carga
{
    public string PerAca { get; set; }
    public string AnioAca { get; set; }
    public string NombreCurso { get; set; }
    public string Abrev { get; set; }
    public string NombreCarrera { get; set; }
    public string Semestre { get; set; }
    public stringCodigoDocente { get; set; }
    public string Plan { get; set; }
    public stringCodigoTurno { get; set; }
    public string TurnoNombre { get; set; }
    public stringCodigoCurso { get; set; }
    public stringCodigoGrupo { get; set; }
    public string GrupoNombre { get; set; }
    public stringCodigoCarrera { get; set; }
    public string ModActa { get; set; }
    public string ModalidadNombre { get; set; }
}
```

Elaborado por el equipo de trabajo

Consulta entre siete (7) tablas para determinar la carga académica durante los semestres enseñados por un determinado docente en las diferentes carreras profesionales de la universidad.

Figura 21: Script de consulta entre varias tablas

```
SELECT cga.per_aca AS 'PerAca', cga.ano_aca AS 'AnioAca',
cga.nom_cur AS 'Curso', cga.abr_car AS 'Abrev', cga.car_des AS
'Carrera', cga.cod_prf AS 'CodDocente', se.sem_des AS 'Semestre',
cga.pln_est AS 'Plan', cga.tur_est AS 'CodTurno', cga.tur_des AS
'TurNomb', cga.cod_cur AS 'CodCurso', cga.sec_gru AS 'CodGrupo',
cga.sec_des AS 'GrupoNomb', cga.cod_car AS 'CodCarrera', cga.mod_act
AS 'ModActa', cga.act_des AS 'ModalidadNombre'
FROM (
select cg.per_aca, cg.ano_aca, cg.pln_est, cg.cod_car,
cg.cod_cur, cu.nom_cur, cu.sem_anu, ca.car_des,
```

...continuación

```
cg.cod_prf, ca.abr_car, cg.tur_est, tu.tur_des,  
cg.sec_gru, gr.sec_des, cg.mod_act, ma.act_des  
FROM carga2016 AS cg  
LEFT JOIN curso cu ON cg.cod_car = cu.cod_car and  
cg.pln_est = cu.pln_est and cg.cod_cur = cu.cod_cur  
LEFT JOIN carrera AS ca ON ca.cod_car = cg.cod_car  
  
LEFT JOIN turno AS tu ON tu.tur_est = cg.tur_est  
LEFT JOIN grupo AS gr ON cg.sec_gru = gr.sec_gru  
LEFT JOIN modact AS ma ON ma.mod_act = cg.mod_act  
ORDER BY per_aca, sem_anu, cod_cur  
) cga LEFT JOIN semestre se ON cga.sem_anu = se.sem_anu  
WHERE cga.cod_prf='08038'
```

Elaborado por el equipo de trabajo

4.3. CAPA DE NEGOCIO

La capa de negocio hace uso de las entidades creadas para realizar las operaciones CRUD (Create, Read, Update, Delete) sobre la base de datos.

En esta capa se definen los distintos métodos de las operaciones que se necesita efectuar para el modelo del negocio. Esta capa hace uso de los distintos objetos que almacenan la información de las tablas provistos en este nivel.

El siguiente fragmento de código muestra la estructura de una consulta basada en una expresión similar a sentencias SQL en LINQ, tales como group, from, select y where sobre un objeto almacenado en una lista. Dicha consulta LINQ retorna la carga académica de los diferentes años de enseñanza de un docente en particular.

Figura 22: Script de expresión LINQ

```
var cargaReal = from m in c  
                group m by new  
                {  
                    m.PerAca, m.AnioAca,  
                } into gcs  
                select new MyLista()  
                {  
                    periodo = gcs.Key.PerAca,  
                    anio = gcs.Key.AnioAca,  
                };
```


...continuación

```
foreach (var product in cargaReal)
{
    var cursosDeUnaCarga = (from valores in c
                            where valores.AnioAca == product.anio &&
                                   valores.PerAca == product.periodo
                            select new
                            {
                                CPerAca = valores.PerAca,
                                CAnioAca = valores.AnioAca,
                                CCurso = valores.NombreCurso,
                                CNombreCarrera = valores.NombreCarrera,
                                CABrev = valores.NombreCarrera,
                                CSemestre = valores.Semestre,
                                CPlan = valores.Plan,
                            }).ToList();
}
```

Elaborado por el equipo de trabajo

4.4. TIEMPO DE RESPUESTA DE CONSULTAS

La comparación de tiempos (milisegundos) de respuesta de las consultas SQL estándar y expresiones LINQ. Usando sentencias SQL se realizó desde tres (3) tablas hasta siete (7) tablas y usando expresiones LINQ ésta se realizó sobre objetos cargados en memoria. Las siguientes tablas presentan los siguientes resultados, donde P representa la primera ejecución, P1, P2, P3, P4 y P5 son las 5 siguientes ejecuciones para cada una de las tablas.

4.4.1. Tiempo de ejecución de consultas de SQL y LINQ

Tabla 8: Tiempo de ejecución de tres tablas

		Consulta SQL	Expresión LINQ
Consulta de tres tablas	P	2.000	1.000
	P1	0.030	0.020
	P2	0.020	0.018
	P3	0.030	0.015
	P4	0.030	0.017
	P5	0.030	0.016
	Promedio	0.357	0.181

Elaborado por el equipo de trabajo

Figura 23: Porcentaje de tiempos de ejecución para tres tablas



Elaborado por el equipo de trabajo

De acuerdo con los resultados mostrados en la tabla 8 se aprecia que en promedio las consultas SQL estándar tienen un tiempo de respuesta de 0.357 (34%) milisegundos mayores que 0.181 (66%) milisegundos que corresponde al tiempo de respuesta de las expresiones LINQ, por lo tanto, las expresiones LINQ llevan una ventaja de 32% en el tiempo de respuesta para tres (3) tablas.

Tabla 09: Tiempo de ejecución de cuatro tablas

		Consulta SQL	Expresión LINQ
P		1.500	1.000
Consulta de cuatro tablas	P1	0.045	0.040
	P2	0.037	0.030
	P3	0.040	0.033
	P4	0.042	0.030
	P5	0.043	0.030
Promedio		0.285	0.194

Elaborado por el equipo de trabajo

Figura 24: Porcentaje de tiempos de ejecución para cuatro tablas



Elaborado por el equipo de trabajo

De acuerdo con los resultados mostrados en la tabla 9 se aprecia que en promedio las consultas SQL estándar tienen un tiempo de respuesta de 0.285 (41%) milisegundos mayores que 0.194 (59%) milisegundos que corresponde al tiempo de respuesta de las expresiones LINQ, por lo tanto, las expresiones LINQ llevan una ventaja de 18% en el tiempo de respuesta para cuatro (4) tablas.

Tabla 10: Tiempo de ejecución de cinco tablas

		Consulta SQL	Expresión LINQ
Consulta de cinco tablas	P	1.200	1.000
	P1	0.035	0.032
	P2	0.037	0.030
	P3	0.390	0.037
	P4	0.041	0.037
	P5	0.042	0.039
	Promedio	0.291	0.196

Elaborado por el equipo de trabajo

Figura 25: Porcentaje de tiempos de ejecución para cinco tablas



Elaborado por el equipo de trabajo

De acuerdo con los resultados mostrados en la tabla 10 se aprecia que en promedio las consultas SQL estándar tienen un tiempo de respuesta de 0.291 (40%) milisegundos mayores que 0.196 (60%) milisegundos que corresponde al tiempo de respuesta de las expresiones LINQ, por lo tanto, las expresiones LINQ llevan una ventaja de 20% en el tiempo de respuesta para cinco (5) tablas.

Tabla 11: Tiempo de ejecución de seis tablas

		Consulta SQL	Expresión LINQ
Consulta de seis tablas	P	1.500	1.100
	P1	0.035	0.032
	P2	0.039	0.030
	P3	0.045	0.039
	P4	0.041	0.036
	P5	0.046	0.037
Promedio		0.284	0.212

Elaborado por el equipo de trabajo

Figura 26: Porcentaje de tiempos de ejecución para seis tablas



Elaborado por el equipo de trabajo

De acuerdo con los resultados mostrados en la tabla 11 se aprecia que en promedio las consultas SQL estándar tienen un tiempo de respuesta de 0.284 (43%) milisegundos mayores que 0.212 (57%) milisegundos que corresponde al tiempo de respuesta de las expresiones LINQ, por lo tanto, las expresiones LINQ llevan una ventaja de 14% en el tiempo de respuesta para seis (6) tablas.

Tabla 12: Tiempo de ejecución de siete tablas

		Consulta SQL	Expresión LINQ
Consulta de siete tablas	P	1.550	1.100
	P1	0.035	0.032
	P2	0.041	0.036
	P3	0.390	0.034
	P4	0.041	0.034
	P5	0.040	0.037
Promedio		0.350	0.212

Elaborado por el equipo de trabajo

Figura 27: Porcentaje de tiempos de ejecución para siete tablas



Elaborado por el equipo de trabajo

De acuerdo con los resultados mostrados en la tabla 12 se aprecia que en promedio las consultas SQL estándar tienen un tiempo de respuesta de 0.350 (42%) milisegundos mayores que 0.212 (58%) milisegundos que corresponde al tiempo de respuesta de las expresiones LINQ, por lo tanto, las expresiones LINQ llevan una ventaja de 18% en el tiempo de respuesta para siete (7) tablas.

4.5. ANÁLISIS ESTADÍSTICO

A la investigación se aplicó un análisis estadístico de prueba de diferencia de medias para datos relacionados para así contrastar la hipótesis planteada. Esta prueba no paramétrica de los rangos con signo de Wilcoxon permitió comparar el rango medio de dos muestras relacionadas y determinar si existen diferencias entre ellas.

El estadístico de prueba que se usó fue el de Wilcoxon con el valor de 'Z' para la diferencia de dos muestras relacionadas, con las siguientes fórmulas:

$$\mu_T = \frac{n(n+1)}{4} \quad \sigma_T^2 = \frac{n(n+1)(2n+1)}{24} \quad Z = \frac{T - \mu_T}{\sigma_T}$$

Donde:

'n': cantidad de muestras

'T': menor de las dos sumas de rangos



4.5.1. Prueba de hipótesis

$$H_0: Me_1 = Me_2 \quad (Me = \text{mediana})$$

H0: El Sistema Académico desarrollado en Visual Studio 2010, las expresiones lambda y expresiones SQL del componente LINQ no mejora en el tiempo de respuesta en la recuperación de datos sobre el motor de base de datos MySQL de la Universidad Privada San Carlos de Puno.

$$H_1: Me_1 \neq Me_2$$

H1: El Sistema Académico desarrollado en Visual Studio 2010, las expresiones lambda y expresiones SQL del componente LINQ si hay mejora en el tiempo de respuesta en la recuperación de datos sobre el motor de base de datos MySQL de la Universidad Privada San Carlos de Puno.

4.5.2. Nivel de significancia

Nivel de significancia de $\alpha = 0,05 = 5 \%$ (que es equivalente a un 95% de nivel de confianza).

4.5.3. Test de Shapiro-Wilk

Para poder testear si los datos se ajustaban a una distribución normal o no, se usó la prueba estadística de Shaphiro-Wilk.

Para la prueba estadística se usó el programa R obteniendo el siguiente resultado.

Para las consultas SQL estándar

```
querySQL <- c(2.000,0.030,0.020,0.030,0.030,0.030,1.500,0.045,0.037,  
0.040,0.042,0.043,1.200,0.035,0.037,0.390,0.041,0.042,  
1.500,0.035,0.039,0.045,0.041,0.046,1.550,0.035,0.041,  
0.390,0.041,0.040)  
shapiro.test(querySQL)
```

Shapiro-wilk normality test

```
data: querySQL  
w = 0.54375, p-value = 1.604e-08
```

Para las consultas con expresiones LINQ

```
queryLINQ <- c(1.000,0.020,0.018,0.015,0.017,0.016,1.000,0.040,0.030,  
0.033,0.030,0.030,1.000,0.032,0.030,0.037,0.037,0.039,  
1.100,0.032,0.030,0.039,0.036,0.037,1.100,0.032,0.036,  
0.034,0.034,0.037)  
shapiro.test(queryLINQ)
```

Shapiro-wilk normality test

```
data: queryLINQ  
w = 0.47873, p-value = 3.216e-09
```

Vemos que en ambos casos el valor de probabilidad (*p-value*) es muy inferior al nivel de significancia elegido (0.05).

Para las **consultas SQL** estándar el $p\text{-value} = 1.604e-08$ y teniendo el valor de significancia de 0.05 y contrastándola se observa que $1.604e-08 < 0.05$.

Para las **expresiones LINQ** el $p\text{-value} = 3.216e-09$ y teniendo el valor de significancia de 0.05 y contrastándola se observa que $3.216e-09 < 0.05$.

Lo que indica que la población no tiene una distribución normal por lo cual se rechaza la hipótesis nula.

Decisión

```
datos <- data.frame(  
  grupo = as.factor(rep(c("SQL", "LINQ"), c(30, 30))),  
  valores = c(querySQL, queryLINQ))  
wilcox_test(valores ~ grupo, data = datos,  
  distribution = "exact", conf.int=0.95)
```




```
Exact wilcoxon-Mann-whitney Test
data: valores by grupo (LINQ, SQL)
Z = -2.8833, p-value = 0.003489
alternative hypothesis: true mu is not equal to 0
95 percent confidence interval:
 -0.013 -0.003
sample estimates:
difference in location
 -0.008
```

Como (p-value) $0.003489 < 0.05$

Entonces se concluye que se rechaza la hipótesis nula (H_0) y aceptamos la hipótesis alterna (H_1), con un error del $0.05 = 5\%$ probando que después de aplicar el componente LINQ en el sistema académico mejoró significativamente el tiempo de respuesta de en la recuperación de información sobre el gestor de base de datos MySQL en la Universidad Privada San Carlos de Puno. Esto significa que si existe diferencia para un 95% de nivel de confianza.



V. CONCLUSIONES

De la investigación realizada y los objetivos planteados sobre el uso del componente LINQ y su mejora en el sistema académico de la Universidad Privada San Carlos de Puno, se llegó a las siguientes conclusiones.

De acuerdo a lo planteado en el objetivo específico N° 1:

Se concluye que para garantizar la integridad referencial de los registros en una base de datos transaccional MySQL, se debe usar el tipo de tabla InnoDB ya que garantiza el almacenamiento transaccional y mantiene las características ACID.

De acuerdo a lo planteado en el objetivo específico N° 2:

La ejecución del estudio comparativo entre las expresiones LINQ frente a las sentencias SQL para el desarrollo del software, permitió determinar en base a los resultados obtenidos que la tecnología LINQ y sus expresiones reduce la complejidad y el tiempo al momento de codificar los scripts de forma comprensible, compacto y mejor aún ya que permite trabajar con varios formatos de datos y orígenes de datos generando una ayuda sustancial.

De acuerdo a lo planteado en el objetivo específico N° 3:

Se implementó el software con el lenguaje de programación C# empleando los operadores de consulta LINQ ya que ofrecen al desarrollador un conjunto de expresiones predefinidas listas para emplearlas ya que ofrecen menos esfuerzo reduciendo significativamente el tiempo de implementación sobre cualquier colección de objetos.

Por lo tanto, la utilización del componente LINQ mejoró notablemente el tiempo de obtención de datos en el Sistema de Académico de la Universidad Privada San Carlos.



VI. RECOMENDACIONES

PRIMERO. Se recomienda elaborar una base de datos conservando la integridad referencial hasta su tercera forma normal como mínimo, ya que su omisión puede acarrear errores y resultados inesperados al momento de realizar peticiones a la base de datos y más aún si se realizan transacciones monetarias.

SEGUNDO. Se recomienda el uso de operadores estándar LINQ ya que por su naturaleza de ser objetos pre establecidos aceleran la ejecución haciendo que el rendimiento en mejore al procesar las consultas sobre la base de datos.

TERCERO. Se recomienda usar el componente LINQ conjuntamente con el lenguaje de programación C# ya que se encuentran integradas y permiten trabajar de forma cómoda y rápida con colecciones de datos.



VII. REFERENCIAS

- Albahari, J., & Albahari, B. (2012). C# 5.0 in a Nutshell. In *Vasa*.
- Alfonso, R., Torres, B., Paguay, P. X., & Riobamba S. (2008). “*Estudio del Lenguaje Integrado de Consultas (Linq), Aplicado al Desarrollo del Sistema de Inventario y Facturación de la Librería Politécnica*”. *Escuela Superior Politécnica de Chimborazo Facultad de Informática y Electrónica, Ecuador*.
- Alonzo Church - *Wikipedia, la enciclopedia libre*. (n.d.). Retrieved January 23, 2020, from https://es.wikipedia.org/wiki/Alonzo_Church
- Aplicación de escritorio. (2021, 1 de febrero). *Wikipedia, La enciclopedia libre*. Fecha de consulta: 00:12, julio 27, 2021 desde https://es.wikipedia.org/w/index.php?title=Aplicaci%C3%B3n_de_escritorio&oldid=132856056.
- Bases de datos - EcuRed*. (n.d.). Retrieved January 21, 2020, from https://www.ecured.cu/Bases_de_datos
- Carlos, Nuñez Amador, (2020). “Desarrollo de sistema automatizado para la gestión de los principales procesos de facturación y reservas del auto lavado D’Autos en la ciudad de Managua durante el segundo semestre del año 2020”. *Tesis de Grado*. Universidad Nacional Autónoma de Nicaragua. <https://repositorio.unan.edu.ni/14838/3/14838.pdf>
- C#: *Creando archivos PDF con iTextSharp*. (n.d.). Retrieved January 21, 2020, from <https://desarrolladores.me/2014/01/c-creando-archivos-pdf-con-itextsharp/>
- Cardona, A. M. S. (2017). Diseños Cuasiexperimentales. *Communication Quarterly*, 65(5), 580–602. <https://doi.org/10.1080/01463373.2017.1321027>
- Contreras Bustamante, J. R. A. (2013). El Sistema de Registro de la Carga Académica mediante Entornos Web. Una propuesta tecnológica para la Gestión en la Universidad Nacional Experimental del Táchira (Venezuela). *Universitas Tarraconensis. Revista de Ciències de l’Educació*, 1(2), 109. <https://doi.org/10.17345/ute.2013.2.578>
- EcuRed. (2014).c++ Builder. https://www.ecured.cu/C%2B%2B_Builder
- Fatima, N. (2018). *¿Qué es el mapeo de datos / Herramientas y técnicas de mapeo de*



- datos*. <https://www.astera.com/es/type/blog/understanding-data-mapping-and-its-techniques/>
- Ferracchiati, F. C. (2008). LINQ for Visual C# 2008. In *LINQ for Visual C# 2008*.
<https://doi.org/10.1007/978-1-4302-1581-3>
- Generación de ficheros “Excel” (xlsx) con ClosedXML*. (n.d.). Retrieved January 21, 2020, from <https://www.fixedbuffer.com/generacion-de-ficheros-excel-con-closedxml/>
- Generar archivos Excel como un señor con ClosedXml | Variable not found*. (2014).
<https://www.variablenotfound.com/2013/03/generar-archivos-excel-como-un-senor.html>
- Gonzales, A. (2011). *Gestión de base de datos* (Primera ed).
- Guerra, J., Santillán, J., & Silva, J. (2015). Análisis de rendimiento de las tecnologías Plinq y Linq en sistemas informáticos. *Revista Científica y Tecnológica UPSE*, 3(1), 143–151. <https://doi.org/10.26423/rctu.v3i1.82>
- Gutierrez, A. (2014). *Programación en capas - PixelG*.
<http://pixelg.org/software/item/44-programacion-en-capas.html>
- Hernández Roberto, Fernández Carlos, B. P. (2014). *Metodología de la Investigación* (E. McGRAW-HILL (Ed.); 6ta Edició).
- Herrera, A. (2016). *Sistema de información para el Instituto de Informática de la Universidad Nacional del Altiplano Puno -2016*. Universidad Nacional del Altiplano.
- Historia de Visual Studio | Visual Studio – Lenguaje de Programación*. (n.d.). Retrieved January 21, 2020, from <https://reyesjoseling.wordpress.com/historia-de-visual-studio/>
- Integridad referencial en MySQL. Programación en Castellano*. (n.d.). Retrieved January 24, 2020, from https://programacion.net/articulo/integridad_referencial_en_mysql_263/3
- Introducción al lenguaje C# y .NET Framework | Microsoft Docs*. (n.d.). Retrieved January 22, 2020, from <https://docs.microsoft.com/es-es/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework>



Lambda calculus - Wikipedia. (n.d.). Retrieved January 23, 2020, from https://en.wikipedia.org/wiki/Lambda_calculus

Las TICS como factor clave en la gestión académica y administrativa de la universidad | Gestión en el Tercer Milenio. (n.d.). Retrieved January 24, 2020, from <https://revistasinvestigacion.unmsm.edu.pe/index.php/administrativas/article/view/14141>

Malaquina Antonio, (2008). "Estudio, prototipación y análisis comparativo de LINQ". Informe de tesis de grado. Universidad de la República Oriental de Uruguay. <https://www.colibri.udelar.edu.uy/jspui/bitstream/20.500.12008/3118/1/tg-malaquina.pdf>

Melvin Bolívar, Gorozabel Bazurto, (2019). "*Desarrollo de un aplicativo informático mediante ENTITY FRAMEWORK para el control y monitoreo de proyectos de investigación con mensajería instantánea*". Tesis de Grado. Pontificia Universidad Católica del Ecuador. <https://repositorio.pucese.edu.ec/handle/123456789/1899>

Microsoft Visual Studio - Wikipedia, la enciclopedia libre. (n.d.). Retrieved January 21, 2020, from https://es.wikipedia.org/wiki/Microsoft_Visual_Studio

Sistema academicos. (n.d.). Retrieved January 24, 2020, from <http://www.sistemasacademicos.com/sistema-academico>

software | Diccionario de la lengua española | RAE - ASALE. (n.d.). Retrieved February 1, 2020, from <https://dle.rae.es/?w=software>



ANEXOS

ANEXO A: Capturas de pantalla del sistema

Pantalla principal

The screenshot shows the 'Sistema Académico' main interface. On the left, there are several filter panels: 'ESCUELAS' (with options like CONTABILIDAD Y FINANZAS, ENFERMERIA, etc.), 'PERIODO' (SEMESTRE I, II, VACACIONAL), 'AÑO ACADÉMICO' (years from 2009 to 2017), and 'SEMESTRE' (from TODOS to DECIMO SEGUNDO). The main area contains a 'Buscar estudiante' form with fields for 'Paterno' and 'Materno' (both empty), and 'Nombres' and 'Num. Mat.' (both empty). There are 'Buscar' and 'Limpiar' buttons. Below the form is a table header with columns: 'Num.Mat.', 'Paterno', 'Materno', and 'Nombres'. The table body is currently empty. At the bottom right, there are 'Listado de Cursos', 'Certificado Est.', and 'Tercio y Quinto' options, along with 'Listar' and 'Imprimir' buttons.

Pantalla de carga académica de docentes

The screenshot shows the 'Sistema Académico' interface for loading academic data for teachers. It features the same filter panels on the left as the main screen. The main area has a 'Buscar Docente' form with 'Paterno' (containing 'MAMA') and 'Materno' (empty) fields, and a 'Buscar Docente' button. Below the form is a table with the following columns: 'Codigo', 'Dni', 'Paterno', 'Materno', 'Nombres', 'Fecha Nac.', 'Direccion', 'Telefono', 'Celular', 'Correo', and 'Estado'. The table contains several rows of data, with the row for '02072' (MAMANI CHOQUE, SABINO EDGAR) highlighted in blue. Below the table, there is a 'Desplegar' button and a list of semesters: '2010 - 02' and '2010 - 01'. At the bottom right, there are 'Exportar Excel' and 'IMPRESIÓN DE ACTAS' buttons.

Codigo	Dni	Paterno	Materno	Nombres	Fecha Nac.	Direccion	Telefono	Celular	Correo	Estado
01289	44473483	MAMANI	GAMARRA	JAVIER ELIAS	01/01/1930					Si
01317	40826226	MAMANI	GOMEZ	JOSE ANTONIO	15/02/1981	JR. AGUSTIN G...		951655253		Si
02072	70963843	MAMANI	CHOQUE	SABINO EDGAR	01/01/1930					Si
02079	64878545	MAMANI	TICONA	JUAN LUIS	01/01/1930					Si
08004	56863539	MAMANI	AGUILAR	SONIA	03/06/1981	URB. CHANU C...				Si
081208	64395254	MAMANI	INOJOSA	JUAN CARLOS	29/01/1988	JR LORETO NA...				Si
01396	44767591	MAMANI	GUEVARA	MARCIAL	01/01/1988	Jr. Luis Banche...		951410498	nguevara16_88...	Si
01600	46213163	MAMANI	PEREZ	WILFRIED ALBER	26/08/1989	JR. BELLAVISTA...		980858722	wilfried.mamani@...	Si



Pantalla de administración de postulantes

Sistema Académico

Docentes Nominas Postulante

Inicio Editor Aplazados

ESCUUELAS

- CONTABILIDAD Y FINANZAS
- ENFERMERIA
- INGENIERIA INFORMATICA
- INGENIERIA AMBIENTAL
- DERECHO

PERIODO

- SEMESTRE I
- SEMESTRE II
- VACACIONAL

AÑO ACADÉMICO

- 2017
- 2016
- 2015
- 2014
- 2013
- 2012
- 2011
- 2010
- 2009
- 2008

SEMESTRE

- TODOS
- PRIMERO
- SEGUNDO
- TERCERO
- CUARTO
- QUINTO
- SEXTO
- SEPTIMO
- OCTAVO
- NOVENO
- DECIMO
- DECIMO PRIMERO
- DECIMO SEGUNDO

Proceso: 2016-01 Modalidad: Examen ordinario Carrera Profesional: Contabilidad y fina Paterno: Materno: Nombres:

Ingreso	Modalidad	Grupo	Horario	Canera	Dni	Modalidad	Paterno	Materno	Nombres	Sex	Telefono	Pas
<input type="checkbox"/>	REGULAR	GRUPO A	MAÑANA	CONTABILIDAD Y FIN...	70614539	EXAMEN ORDINARIO	AGUILAR	MAMANI	VILMA BENILDA	F	970878152	0
<input type="checkbox"/>	REGULAR	GRUPO A	MAÑANA	CONTABILIDAD Y FIN...	70791516	EXAMEN ORDINARIO	ANCCASI	QUISPE	JUAN CLISMAR	M	982004120	0
<input type="checkbox"/>	REGULAR	GRUPO A	MAÑANA	CONTABILIDAD Y FIN...	75196957	EXAMEN ORDINARIO	APAZA	CALSIN	LUZ VALERIA	F	966698432	0
<input type="checkbox"/>	REGULAR	GRUPO A	MAÑANA	CONTABILIDAD Y FIN...	47661999	EXAMEN ORDINARIO	APAZA	CCOSI	HERNANDO	M	958164127	0
<input type="checkbox"/>	REGULAR	GRUPO A	MAÑANA	CONTABILIDAD Y FIN...	01873701	EXAMEN ORDINARIO	APAZA	CHAMBI	ALDO ANDRES	M	957880047	0
<input type="checkbox"/>	REGULAR	GRUPO A	MAÑANA	CONTABILIDAD Y FIN...	42532718	EXAMEN ORDINARIO	APAZA	QUISPE	CARMEN GRISELA	F	969633335	0
<input type="checkbox"/>	REGULAR	GRUPO A	MAÑANA	CONTABILIDAD Y FIN...	46054314	EXAMEN ORDINARIO	ARUHUANCA	MAMANI	HENRY	M	985035344	0
<input type="checkbox"/>	REGULAR	GRUPO A	MAÑANA	CONTABILIDAD Y FIN...	43380248	EXAMEN ORDINARIO	AVILA	MERLO	JONATHAN	M	977330867	0
<input type="checkbox"/>	REGULAR	GRUPO A	MAÑANA	CONTABILIDAD Y FIN...	43007395	EXAMEN ORDINARIO	BELIZARIO	SUCAPUCA	GUSTAVO ADOLFO	M	995099990	0
<input type="checkbox"/>	REGULAR	GRUPO A	MAÑANA	CONTABILIDAD Y FIN...	46492509	EXAMEN ORDINARIO	CAIRA	YUCRA	DORIS KARINA	F	972912840	0

CARGAR INGRESANTES

Datos para modificación

Núm. Mat: Dni:

Modalidad: Grupo: Turno: Actualizar

Activar Postulante Apto Matricular Ingresante Relación Ingresantes

Pantalla de administración de matrícula

Sistema Académico

Docentes Nominas Postulante

Inicio Editor Aplazados

ESCUUELAS

- CONTABILIDAD Y FINANZAS
- ENFERMERIA
- INGENIERIA INFORMATICA
- INGENIERIA AMBIENTAL
- DERECHO

PERIODO

- SEMESTRE I
- SEMESTRE II
- VACACIONAL

AÑO ACADÉMICO

- 2017
- 2016
- 2015
- 2014
- 2013
- 2012
- 2011
- 2010
- 2009
- 2008

SEMESTRE

- TODOS
- PRIMERO
- SEGUNDO
- TERCERO
- CUARTO
- QUINTO
- SEXTO
- SEPTIMO
- OCTAVO
- NOVENO
- DECIMO
- DECIMO PRIMERO
- DECIMO SEGUNDO

Buscar estudiante

Paterno: Materno: Nombres: Num. Mat. Buscar Limpiar

NumMat	Paterno	Materno	Nombres
071263	QUISPE	APAZA	BERNARDO
071264	QUISPE	CONDORI	GROBER
071265	QUISPE	YANARICO	JESUS RENE
072227	QUISPE	ALBERTO	JESICA BETTY
081222	QUISPE	TICONA	KATTY GINA
081240	QUISPE	CHOQUE	URIEL MARCO
081242	QUISPE	VIZCARRA	LEYDY DIANA
081272	QUISPE	BIZARRO	ANDDY FELIX
091007	QUISPE	MAMANI	ROCIO DEL CAR...
091079	QUISPE	GUTIERREZ	MICHAEL HUGO

CONTABILIDAD Y FINANZAS
Quispe Alberto, Jesica Betty

Cart	Año	Periodo	Semestre	NombreCurso	Sem	Cred	Nota	Modalidad	Sem/Nombre	tot_cred
1	2007	02	SEMESTRE II	MATEMATICA I	01	4	12	REGULAR	PRIMERO	22.00
2	2007	02	SEMESTRE II	LENGUAJE Y COMUNICACION	01	3	12	REGULAR	PRIMERO	22.00
3	2007	02	SEMESTRE II	INTRODUCCION A LAS CIENCIAS SOCIALES	01	3	16	REGULAR	PRIMERO	22.00
4	2007	02	SEMESTRE II	METODOLOGIA DEL TRABAJO UNIVERSITARI	01	4	13	REGULAR	PRIMERO	22.00
5	2007	02	SEMESTRE II	FILOSOFIA Y EPISTEMOLOGIA	01	2	15	REGULAR	PRIMERO	22.00
6	2007	02	SEMESTRE II	VISION HISTORICA DEL PERU Y DEL MUNDO	01	3	14	REGULAR	PRIMERO	22.00
7	2007	02	SEMESTRE II	ECOLOGIA Y MEDIO AMBIENTE	01	3	14	REGULAR	PRIMERO	22.00
8	2008	01	SEMESTRE I	MATEMATICA II	02	4	13	REGULAR	SEGUNDO	22.00
9	2008	01	SEMESTRE I	INFORMATICA I	02	4	13	REGULAR	SEGUNDO	22.00
10	2008	01	SEMESTRE I	REALIDAD NACIONAL	02	3	14	REGULAR	SEGUNDO	22.00
11	2008	01	SEMESTRE I	METODOLOGIA DE LA INVESTIGACION CIEN	02	3	15	REGULAR	SEGUNDO	22.00
12	2008	01	SEMESTRE I	INGLES I	02	3	15	REGULAR	SEGUNDO	22.00
13	2008	01	SEMESTRE I	ECONOMIA GENERAL	02	3	14	REGULAR	SEGUNDO	22.00

Listado de Cursos Certificado Est. Tercio y Quinto

Cuadro de méritos:

Nu	NumMat	Apellidos y nombres	Modalidad	PPS	CM	CrA	CD	CM	CA	CD
1	081252	AMANGUI MENDOZA, WILSON	REGULAR	17.00	22.00	5.00	0.00	1	1	0
2	081246	VILLALTA RAMOS, LOURDES	REGULAR	17.00	22.00	5.00	0.00	1	1	0
3	091011	VILCARANA CAHUACHIA, VERONICA GLADYS	REGULAR	16.64	22.00	22.00	0.00	7	7	0
4	091083	CCAMA MAMANI, RICHARD	REGULAR	16.00	22.00	5.00	0.00	1	1	0
5	081268	PAVIO AROCUTIPA, AMELIA	REGULAR	15.57	22.00	21.00	0.00	7	7	0
6	081022	TICONA CCALLA, ANGELICA LIA	REGULAR	15.41	22.00	22.00	0.00	7	7	0
7	091021	CARRION POMA, MARIA JESUSA	REGULAR	15.27	22.00	22.00	0.00	7	7	0
8	081296	BECERRA MOLLO, GORETTI PIONIA	REGULAR	15.10	22.00	21.00	0.00	7	7	0
9	081081	OCHOA MERMBA, LUZ NOELIA	REGULAR	14.67	22.00	21.00	0.00	7	7	0

Listar Imprimir



ANEXO B: Scripts, expresiones y reportes

Script de conexión:

```
class Conexion
{
    private static MySqlConnection conex;
    private string servidor;
    //private string puerto;
    private string pass;
    private string usuario;
    private string database;
    private string cadena;

    public Conexion()
    {
        this.servidor = Parametros.servidor;
        //this.puerto = Parametros.puerto; "3306";
        this.database = Parametros.database;
        this.usuario = Parametros.usuario;
        this.pass = Parametros.pass;
        //Console.Write(usuario);
        cadena = "server=" + this.servidor + ";database=" +
            this.database + ";uid=" + this.usuario + ";password=" + this.pass;
        conex = new MySqlConnection(cadena);
    }
    public MySqlConnection getConexion()
    {
        return conex;
    }

    public bool openConnection()
    {
        try
        {
            conex.Open();
            return true;
        }
        catch (MySqlException ex)
        {
            return false;
        }
    }

    public bool closeConnection()
    {
        try
        {
            conex.Close();
            return true;
        }
        catch (MySqlException ex)
        {
            MessageBox.Show(ex.Message);
            return false;
        }
    }
}
```



Expresiones LINQ

```
var lista = from result in ultSemestre
            where result.AnioAca == anio_periodo[0] && result.PerAca == anio_periodo[1]
            select new
            {
                AnioPeriodo = result.AnioAca + "-" + result.PerAca,
                result.Abrev,
                result.NombreCurso,
                result.Semestre,
                result.GrupoNombre,
                result.TurnoNombre,
                Datos = lblProfe.Text + ";" +
                    result.NombreCarrera + ";" +
                    result.NombreCurso + ";" +
                    result.Semestre + ";" +
                    result.AnioAca + ";" +
                    result.PerAca + ";" +
                    result.ModalidadNombre + ";" +
                    result.TurnoNombre + ";" +
                    result.GrupoNombre + ";" +
                    result.CodigoCurso + ";" +
                    result.CodigoCarrera + ";" +
                    result.CodigoDocente + ";" +
                    result.CodigoGrupo + ";" +
                    result.CodigoTurno + ";" +
                    result.Plan + ";" +
                    result.ModActa,
                result.Plan
            };
```



Expresiones LINQ para general archivos excel

```
var campos = from t in tabla.AsEnumerable()
             select new { mat = t["MAT"], nom = t["NOM"], sem = t["SEM"], gru = t["GRU"], tur = t["TUR"]

var columnasAgrupadas = from table in campos
                        group table by new { table.sem, table.gru, table.tur }
                          into respuestaTabla
                          select new
                          {
                              camp01 = respuestaTabla.Key,
                              total = respuestaTabla.Count()
                          };

int cantHojas = 0;

foreach (var key in columnasAgrupadas)
{
    //MessageBox.Show(key.camp01.sem.ToString() + " - " + key.camp01.gru.ToString() + " - " + key.camp01.
    DataRow[] result = tabla.Select("SEM = '" + key.camp01.sem.ToString() + "' AND GRU = '" + key.camp01.
    foreach (DataRow row in result)
    {
        celda1 = new PdfPCell(new Paragraph((+numFilas).ToString(), FontFactory.GetFont("Arial", 8)));
        celda1.FixedHeight = 15.0f;
        celda1.HorizontalAlignment = Element.ALIGN_CENTER;
        celda1.VerticalAlignment = Element.ALIGN_MIDDLE;
        unaTabla.AddCell(celda1);
        celda1 = new PdfPCell(new Paragraph(row[1].ToString(), FontFactory.GetFont("Arial", 8)));
        celda1.FixedHeight = 15.0f;
        celda1.HorizontalAlignment = Element.ALIGN_CENTER;
        celda1.VerticalAlignment = Element.ALIGN_MIDDLE;
        unaTabla.AddCell(celda1);
    }
}
```



Script de consulta de tres tablas

```
public List<TercioQuinto> ResultadoTercioQuinto(string codCar, string numMat, string anio, string periodo)
{
    List<TercioQuinto> lista = new List<TercioQuinto>();
    //lista = null;
    try
    {
        if (this.openConnection() == true)
        {
            string sql = "select no.num_mat, no.cod_car, round(sum(no.not_cur*cu.crd_cur)/sum(crd_cur),2) as pps,
                "sum(if(no.not_cur>10, cu.crd_cur, 0)) as cra, sum(if(no.not_cur<=10, cu.crd_cur, 0)) as crd, " +
                "sum(if(no.not_cur>10, 1, 0)) as ca, sum(if(no.not_cur<=10, 1, 0)) as cd, es.sem_anu, sem.sem_des
                "from nota" + codCar + " no left join curso cu on no.cod_car = cu.cod_car and no.pln_est = cu.pln_
                "no.cod_cur = cu.cod_cur " +
                "LEFT JOIN estimat" + anio + " AS es ON es.num_mat=no.num_mat and " +
                "es.cod_car = no.cod_car and es.ano_aca = no.ano_aca and " +
                "es.per_aca = no.per_aca and es.pln_est = '01'" +
                "LEFT JOIN semestre AS sem ON es.sem_anu = sem.sem_anu " +
                "where no.ano_aca = '" + anio + "' and no.cod_car = '" + codCar + "' and " +
                "no.per_aca = '" + periodo + "' and no.mod_not in ('01') " +
                "and no.num_mat ='" + numMat + "' ";

            v_cmd = new MySqlCommand(sql, this.getConexion());
```

Script de consulta de datos de cinco tablas

```
public DataTable EstudianteMatriculado(string anio, string periodo, string codigo)
{
    DataTable dt = new DataTable();    DataColumn inc = new DataColumn();
    inc.AutoIncrement = true;
    inc.AutoIncrementSeed = 1;
    inc.AutoIncrementStep = 1;
    inc.ColumnName = "ID";
    inc.DataType = System.Type.GetType("System.Int32");
    inc.Unique = true;
    dt.Columns.Add(inc);
    try
    {
        if (this.openConnection() == true)
        {
            dt.Columns.Add("MAT", typeof(string));
            dt.Columns.Add("NOM", typeof(string));
            dt.Columns.Add("SEM", typeof(string));
            dt.Columns.Add("GRU", typeof(string));
            dt.Columns.Add("TUR", typeof(string));
            v_cmd = new MySqlCommand("SELECT em.num_mat AS 'NumMat', concat(es.paterno, ' ', es.materno, ',
                "se.sem_des AS 'Semestre', mm.mod_des AS 'Modalidad', gr.sec_des AS 'Grupo', tu.tur_des
                "FROM estimat" + anio + " em " +
                "LEFT JOIN estudiante es ON em.num_mat = es.num_mat and em.cod_car = es.cod_car " +
                "LEFT JOIN semestre se ON em.sem_anu = se.sem_anu " +
                "LEFT JOIN modmat mm ON em.mod_mat = mm.mod_mat " +
                "LEFT JOIN grupo gr ON em.sec_gru = gr.sec_gru " +
                "LEFT JOIN turno tu ON em.tur_est = tu.tur_est " +
                "WHERE em.cod_car = '" + codigo + "' AND em.ano_aca = '" + anio + "' AND " +
                "em.per_aca = '" + periodo + "' " +
                "ORDER BY se.sem_anu, gr.sec_des, tu.tur_des, NombreEst ", this.getConexion());
```

Script de consulta de seis tablas

```
public DataTable EstudiantePorSemestreGrupoTurno(string semestre, string grupo, string turno, string codCar,
{
    DataTable dt = new DataTable();
    try
    {
        if (this.openConnection() == true)
        {
            dt.Columns.Add("Codigo", typeof(string));
            dt.Columns.Add("Nombre", typeof(string));
            dt.Columns.Add("Semestre", typeof(string));
            dt.Columns.Add("Modalidad", typeof(string));
            dt.Columns.Add("Grupo", typeof(string));
            dt.Columns.Add("Turno", typeof(string));

            string sql = "select em.num_mat AS 'NumMat', " +
                "concat(es.paterno, ' ', es.materno, ' ', es.nombres) as 'NombreEst', " +
                "se.sem_des as 'Semest', mm.mod_des AS 'Modalid', " +
                "gr.sec_des AS 'Grup', tu.tur_des AS 'Turn' " +
                "from estumat " + anio + " em " +
                "left join estudiante es on em.num_mat = es.num_mat and em.cod_car = es.cod_car " +
                "left join semestre se on em.sem_anu = se.sem_anu " +
                "left join modmat mm on em.mod_mat = mm.mod_mat " +
                "left join grupo gr on em.sec_gru = gr.sec_gru " +
                "left join turno tu on em.tur_est = tu.tur_est " +
                "where em.cod_car = '" + codCar + "' and em.ano_aca = '" + anio + "' and " +
                "em.per_aca = '" + periodo + "' ";
            v_cmd = new MySqlCommand(sql, this.getConexion());
        }
    }
}
```

Script de consulta de siete tablas

```
public string RetornaCargaDocente(string carga, string codigo)
{
    return " SELECT cga.per_aca AS 'PerAca', cga.ano_aca AS 'AnioAca', cga.nom_cur AS 'Curso', cga.abr_car /
        " se.sem_des AS 'Semestre', cga.pln_est AS 'Plan', cga.tur_est AS 'CodTurno', cga.tur_de
        " cga.cod_cur AS 'CodCurso', cga.sec_gru AS 'CodGrupo', cga.sec_des AS 'GrupoNomb', cga.
        " FROM ( " +
        " select cg.per_aca, cg.ano_aca, cg.pln_est, cg.cod_car, cg.cod_cur, cu.nom_cur, cu.sem_
        " cg.cod_prf, ca.abr_car, cg.tur_est, tu.tur_des, cg.sec_gru, gr.sec_des, cg.mod_act, ma
        " FROM " + carga + " AS cg " +
        " LEFT JOIN curso cu ON cg.cod_car = cu.cod_car and cg.pln_est = cu.pln_est and cg.cod_c
        " LEFT JOIN carrera AS ca ON ca.cod_car = cg.cod_car " +
        " LEFT JOIN turno AS tu ON tu.tur_est = cg.tur_est " +
        " LEFT JOIN grupo AS gr ON cg.sec_gru = gr.sec_gru " +
        " LEFT JOIN modact AS ma ON ma.mod_act = cg.mod_act " +
        " ORDER BY per_aca, sem_anu, cod_cur " +
        " ) cga " +
        " LEFT JOIN semestre se ON cga.sem_anu = se.sem_anu " +
        " WHERE cga.cod_prf='" + codigo + "' order by cga.per_aca desc, cga.cod_car, cga.cod_cur
    }
}
```



Certificado de estudios

Sistema Académico

Docentes Nominas Postulante

Inicio Editar Aplazados

ESCUELAS

- CONTABILIDAD Y FINANZAS
- ENFERMERIA
- INGENIERIA INFORMATICA
- INGENIERIA AMBIENTAL
- DERECHO

PERIODO

- SEMESTRE I
- SEMESTRE II
- VACACIONAL

AÑO ACADÉMICO

- 2017
- 2016
- 2015
- 2014
- 2013
- 2012
- 2011
- 2010
- 2009
- 2008

SEMESTRE

- TODOS
- PRIMERO
- SEGUNDO
- TERCERO
- CUARTO
- QUINTO
- SEXTO
- SEPTIMO
- OCTAVO
- NOVENO
- DECIMO
- DECIMO PRIMERO
- DECIMO SEGUNDO

Buscar estudiante

Patemo: MAMAN, Matemo:

Nombres: Num. Mat.: Buscar Limpiar

NumMat	Patemo	Matemo	Nombres
071125	MAMANI	TORRES	ANA YSMENA
071123	MAMANI	MAMANI	NANCY ALEJAN...
081136	MAMANI	CHOQUE	EVA IRENE
071124	MAMANI	MAMANI	NOHELIA KARINA
081152	MAMANI	ORTEGA	JOSE PEPE
081155	MAMANI	CCAMA	GRISSEL ERIKA
081156	MAMANCHURA	MIRANDA	MARISOL
081166	MAMANI	SAGUA	NANCY
092006	MAMANI	QUISPE	ANA MELVA
092025	MAMANI	BALBOA	NURY

ENFERMERIA
Mamani Ortega, Jose Pepe

Cart	Anio	Periodo	Semestre	NombreCurso	Sem	Cred	Nota	Modalidad	SemNombre	tot_cred
1	2008	02	SEMESTRE II	MATEMATICA I	01	4	0	REGULAR	PRIMERO	22.00
2	2008	02	SEMESTRE II	LENGUAJE Y COMUNICACION	01	4	0	REGULAR	PRIMERO	22.00
3	2008	02	SEMESTRE II	INTRODUCCION A LAS CIENCIAS SOCIALES	01	3	0	REGULAR	PRIMERO	22.00
4	2008	02	SEMESTRE II	METODOLOGIA DEL TRABAJO UNIVERSITARI	01	3	0	REGULAR	PRIMERO	22.00
5	2008	02	SEMESTRE II	FILOSOFIA Y EPISTEMOLOGIA	01	2	0	REGULAR	PRIMERO	22.00
6	2008	02	SEMESTRE II	VISION HISTORICA DEL PERU Y DEL MUND	01	3	0	REGULAR	PRIMERO	22.00
7	2008	02	SEMESTRE II	ECOLOGIA Y MEDIO AMBIENTE	01	3	0	REGULAR	PRIMERO	22.00
8	2011	01	SEMESTRE I	MATEMATICA I	01	4	0	REGULAR	PRIMERO	22.00
9	2011	01	SEMESTRE I	LENGUAJE Y COMUNICACION	01	4	0	REGULAR	PRIMERO	22.00
10	2011	01	SEMESTRE I	INTRODUCCION A LAS CIENCIAS SOCIALES	01	3	0	REGULAR	PRIMERO	22.00
11	2011	01	SEMESTRE I	METODOLOGIA DEL TRABAJO UNIVERSITARI	01	3	0	REGULAR	PRIMERO	22.00
12	2011	01	SEMESTRE I	FILOSOFIA Y EPISTEMOLOGIA	01	2	0	REGULAR	PRIMERO	22.00
13	2011	01	SEMESTRE I	VISION HISTORICA DEL PERU Y DEL MUND	01	3	0	REGULAR	PRIMERO	22.00

Listado de Cursos Certificado Est. Tercio y Quinto

Cuadro de méritos:

Listar Imprimir

Estudiante Mamani Ortega José Pepe



: MAMANI ORTEGA, JOSE PEPE
: ENFERMERIA
: 081152

CICLO: PRIMERO					
1	MATEMATICA I	CERO	0	4	2008-II
2	LENGUAJE Y COMUNICACION	CERO	0	4	2008-II
3	INTRODUCCION A LAS CIENCIAS SOCIALES	CERO	0	3	2008-II
4	METODOLOGIA DEL TRABAJO UNIVERSITARI	CERO	0	3	2008-II
5	FILOSOFIA Y EPISTEMOLOGIA	CERO	0	2	2008-II
6	VISION HISTORICA DEL PERU Y DEL MUND	CERO	0	3	2008-II
7	ECOLOGIA Y MEDIO AMBIENTE	CERO	0	3	2008-II
8	MATEMATICA I	DOCE	12	4	2012-II
9	LENGUAJE Y COMUNICACION	DOCE	12	4	2012-II
10	INTRODUCCION A LAS CIENCIAS SOCIALES	TRECE	13	3	2012-II
11	METODOLOGIA DEL TRABAJO UNIVERSITARI	DOCE	12	3	2012-II
12	FILOSOFIA Y EPISTEMOLOGIA	CATORCE	14	2	2012-II
13	VISION HISTORICA DEL PERU Y DEL MUND	CATORCE	14	3	2012-II
14	ECOLOGIA Y MEDIO AMBIENTE	ONCE	11	3	2012-II
CICLO: PRIMERO					
15	MATEMATICA I	CERO	0	4	2011-I
16	LENGUAJE Y COMUNICACION	CERO	0	4	2011-I
17	INTRODUCCION A LAS CIENCIAS SOCIALES	CERO	0	3	2011-I
18	METODOLOGIA DEL TRABAJO UNIVERSITARI	CERO	0	3	2011-I
19	FILOSOFIA Y EPISTEMOLOGIA	CERO	0	2	2011-I
20	VISION HISTORICA DEL PERU Y DEL MUND	CERO	0	3	2011-I
21	ECOLOGIA Y MEDIO AMBIENTE	CERO	0	3	2011-I
CICLO: SEGUNDO					
22	MATEMATICA II	DOCE	12	4	2013-I
23	INFORMATICA I	CERO	0	3	2013-I
24	BIOLOGIA GENERAL	TRECE	13	4	2013-I
25	METODOLOGIA DE LA INVESTIGACION CIEN	CERO	0	3	2013-I
26	INGLES I	CERO	0	3	2013-I
27	ECONOMIA GENERAL	ONCE	11	3	2013-I
28	PSICOLOGIA GENERAL	DOCE	12	2	2013-I
CICLO: TERCERO					
29	BIOESTADISTICA I	CERO	0	3	2013-II
30	BIOQUIMICA	CERO	0	3	2013-II
31	ENFERMERIA Y SALUD COMUNITARIA	CERO	0	4	2013-II
32	REALIDAD NACIONAL	CERO	0	3	2013-II
CICLO: TERCERO					
33	INFORMATICA I	ONCE	11	3	2015-I
34	METODOLOGIA DE LA INVESTIGACION CIEN	DOCE	12	3	2015-I
35	INGLES I	DOCE	12	3	2015-I
36	BIOESTADISTICA I	ONCE	11	3	2015-I
37	BIOQUIMICA	TRECE	13	3	2015-I
38	ENFERMERIA Y SALUD COMUNITARIA	TRECE	13	4	2015-I
39	REALIDAD NACIONAL	TRECE	13	3	2015-I



CICLO: CUARTO					
40	INFORMATICA II	CERO	0	3	2015-II
41	ANATOMIA HUMANA I	ONCE	11	3	2015-II
42	INGLES II	CERO	0	3	2015-II
43	BIOESTADISTICA II	ONCE	11	3	2015-II
44	ENFERMERIA CLINICA	DOCE	12	3	2015-II
45	BIOFISICA	DOCE	12	3	2015-II
46	REALIDAD REGIONAL	ONCE	11	3	2015-II
CICLO: QUINTO					
47	INFORMATICA II	DOCE	12	3	2016-I
48	INGLES II	DOCE	12	3	2016-I
49	ANATOMIA HUMANA II	ONCE	11	3	2016-I
50	FISIOLOGIA HUMANA	TRECE	13	4	2016-I
51	RELACIONES HUMANAS	QUINCE	15	3	2016-I
52	NUTRICION HUMANA	DOCE	12	4	2016-I
53	FARMACOLOGIA	ONCE	11	3	2016-I
54	PSICOLOGIA APLICADA A LA SALUD	DIECISEIS	16	3	2016-I
CICLO: SEXTO					
55	METODOLOGIA DE LA ATENCION EN ENFERM	ONCE	11	3	2016-II
56	MICROBIOLOGIA - PARASITOLOGIA	DOCE	12	3	2016-II
57	BIOLOGIA CELULAR	DOCE	12	3	2016-II
58	ENFERMERIA EN SALUD PUBLICA	TRECE	13	3	2016-II
59	SEMIOLOGIA	TRECE	13	4	2016-II
60	ENFERMERIA EN EPIDEMIOLOGIA	ONCE	11	3	2016-II
61	PLANIFICACION ESTRATEGICA EN SALUD	ONCE	11	3	2016-II
62	EDUCACION SANITARIA	TRECE	13	3	2016-II
CICLO: SEXTO					
63	ENFERMERIA Y SALUD DEL NINO Y ADOLES	DOCE	12	2	2017-Vac
64	PEDIATRIA	TRECE	13	4	2017-Vac



REPORTE CARGA ACADÉMICA DE DOCENTE

Sistema Académico

Docentes Nóminas Postulante

Inicio Editar Aplazados

ESCUELAS

- CONTABILIDAD Y FINANZAS
- ENFERMERIA
- INGENIERIA INFORMATICA
- INGENIERIA AMBIENTAL
- DERECHO

PERIODO

- SEMESTRE I
- SEMESTRE II
- VACACIONAL

AÑO ACADÉMICO

- 2017
- 2016
- 2015
- 2014
- 2013
- 2012
- 2011
- 2010
- 2009
- 2008

SEMESTRE

- TODOS
- PRIMERO
- SEGUNDO
- TERCERO
- CUARTO
- QUINTO
- SEXTO
- SEPTIMO
- OCTAVO
- NOVENO
- DECIMO
- DECIMO PRIMERO
- DECIMO SEGUNDO

Buscar Docente

Paterno: ZANA Materno: Nombres: Buscar Docente

Codigo	Dni	Paterno	Materno	Nombres	Fecha Nac.	Direccion	Telefono	Celular	Correo	Estado
02066	34984592	ZANABRIA	GALVEZ	ALDO HERNAN	01/01/1930					Si
08038	01326563	ZANABRIA	ORTEGA	MILDER	18/01/1976	URB. CHANU C...	354631	951522828	milyzanabria@hot...	Si
01581	01335507	ZANABRIA	RAMOS	ROGER	24/07/1976	Jr. Deustua NA# ...	12345	951074909	rganabria_10@h...	Si

Desplegar ZANABRIA ORTEGA, MILDER Exportar Excel

IMPRESIÓN DE ACTAS

AñoPer	Carrera	Nombre del curso	Semestre	Grupo	Turno	Plan	Regular	Aplazado
2015-02	CONTA	MATEMATICA BASICA	PRIMERO	GRUPO A	MAÑANA	02	Regular	Aplazado
2015-02	CONTA	MATEMATICA BASICA	PRIMERO	GRUPO B	TARDE B	02	Regular	Aplazado
2015-02	INFOR	CALIDAD DE SISTEMAS DE INFORMACION	SEPTIMO	GRUPO B	TARDE A	01	Regular	Aplazado
2015-02	INFOR	TELEMATICA	DECIMO	GRUPO B	TARDE A	01	Regular	Aplazado



40d1f213-a526-4c2f-bd7b-2946d260531a.xlsx - Excel (Error de activa...)

Archivo Inicio Inserta Diseño Fórmu Datos Revisa Vista Desarr Comp LOAD Power Team Indicar... Compartir

Calibri 11 General Formato condicional Dar formato como tabla Estilos de celda Celdas Modificar

Portapapeles Fuente Alineación Número Estilos

A1 Semestre 2016 - 02

	A	B	C	D
1	Semestre 2016 - 02			
2		CALIDAD DE SISTEMAS DE INFORMACION	INGENIERIA INFORMATICA - SEPTIMO ciclo	
3		ADMINISTRACION DE CENTROS DE COMPUTO	INGENIERIA INFORMATICA - OCTAVO ciclo	
4		PROCESADOR DE TEXTOS Y BASE DEDATOS	INGENIERIA AMBIENTAL - SEGUNDO ciclo	
5		PROCESADOR DE TEXTOS Y BASE DEDATOS	INGENIERIA AMBIENTAL - SEGUNDO ciclo	
6	Semestre 2016 - 01			
7		CALIDAD DE SISTEMAS DE INFORMACION	INGENIERIA INFORMATICA - SEPTIMO ciclo	
8		ADMINISTRACION DE CENTROS DE COMPUTO	INGENIERIA INFORMATICA - OCTAVO ciclo	
9		LOGISTICA	INGENIERIA INFORMATICA - DECIMO ciclo	
10		PROCESADOR DE TEXTOS Y BASE DEDATOS	INGENIERIA AMBIENTAL - SEGUNDO ciclo	
11		PROCESADOR DE TEXTOS Y BASE DEDATOS	INGENIERIA AMBIENTAL - SEGUNDO ciclo	
12		INFORMATICA II	INGENIERIA AMBIENTAL - TERCERO ciclo	
13	Semestre 2015 - 02			
14		MATEMATICA BASICA	CONTABILIDAD Y FINANZAS - PRIMERO ciclo	
15		MATEMATICA BASICA	CONTABILIDAD Y FINANZAS - PRIMERO ciclo	
16		CALIDAD DE SISTEMAS DE INFORMACION	INGENIERIA INFORMATICA - SEPTIMO ciclo	
17		TELEMATICA	INGENIERIA INFORMATICA - DECIMO ciclo	
18				
19				

Carga del docente

Listo 100 %



ESTUDIANTES POR SEMESTRE, GRUPO Y TURNO

Sistema Académico

Docentes Nóminas Postulante

Inicio Editar Aplazados

ESCUELAS

- CONTABILIDAD Y FINANZAS
- ENFERMERIA**
- INGENIERIA INFORMATICA
- INGENIERIA AMBIENTAL
- DERECHO

PERIODO

- SEMESTRE I
- SEMESTRE II
- VACACIONAL

AÑO ACADÉMICO

- 2017
- 2016**
- 2015
- 2014
- 2013
- 2012
- 2011
- 2010
- 2009
- 2008

SEMESTRE

- TODOS
- PRIMERO
- SEGUNDO
- TERCERO
- CUARTO
- QUINTO**
- SEXTO
- SEPTIMO
- OCTAVO
- NOVENO
- DECIMO
- DECIMO PRIMERO
- DECIMO SEGUNDO

Buscar Docente

Apellido Materno Nombres

Buscar Docente

Estudiante por Semestre, Grupo y Turno

ENFERMERIA

Seleccione cada una de las opciones

Semestre Grupo Turno

SEGUNDO TODOS TODOS

Ver documento Cerrar

Desplegar

IMPRESIÓN DE ACTAS

Escribe aquí para buscar

21:20 06/07/2021



EstSemGruTur_02_99_9.xlsx - Excel (Error de activación de productos)

Archivo Inicio Inserta Diseño Fórmu Datos Revisar Vista Desarr Compl LOAD Power Team Indicar... Compartir

Calibri 10 General Formato condicional Dar formato como tabla Estilos de celda Celdas Modificar

Portapapeles Fuente Alineación Número Estilos

A1 ESTUDIANTES MATRICULADOS

ESTUDIANTES MATRICULADOS						
Nº	CÓDIGO	APELLIDOS Y NOMBRES	SEMESTE	MODALIDAD	GRUPO	TURNO
1	150685	ALBARRACIN MAMANI, DORIS PATRICIA	SEGUNDO	REGULAR	GRUPO B	TARDE B
2	150686	ARCATA MAMANI, DANIA ESMERALDA	SEGUNDO	REGULAR	GRUPO B	TARDE B
3	150969	BECERRA QUISPE, VILMA	SEGUNDO	REGULAR	GRUPO C	TARDE A
4	150994	CACERES PUMA, GLADYS	SEGUNDO	REGULAR	GRUPO C	TARDE B
5	150842	CARCAUSTO ALVAREZ, TANIA ZAIDA	SEGUNDO	REGULAR	GRUPO B	TARDE B
6	150995	CCOSI APAZA, ANDREA	SEGUNDO	REGULAR	GRUPO C	TARDE B
7	150858	CHAYÑA TAPIA, KATERIN MARIA	SEGUNDO	REGULAR	GRUPO B	TARDE B
8	150843	CHOQUE LUNA, MARCIA EVELIN	SEGUNDO	REGULAR	GRUPO B	TARDE B
9	140053	CHOQUEHUANCA MAMANI, ELIZABETH MYRIAN	SEGUNDO	REGULAR	GRUPO B	TARDE B
10	150970	CHURA CANCAPA, VERONICA	SEGUNDO	REGULAR	GRUPO C	TARDE A
11	150996	CONDORI AÑAMURO, PAULINA	SEGUNDO	REGULAR	GRUPO C	TARDE B
12	150997	CONDORI VERA, HENRY EDGARDO	SEGUNDO	REGULAR	GRUPO C	TARDE B
13	150972	ESPEZUA MAMANI, VANESSA	SEGUNDO	REGULAR	GRUPO C	TARDE A
14	150999	FAIJOO CANAZA, CARMEN ROSA	SEGUNDO	REGULAR	GRUPO C	TARDE B
15	150973	FLORES ARIAS, ANYELA MYRLA	SEGUNDO	REGULAR	GRUPO C	TARDE A
16	150845	FLORES RODRIGUEZ, SONIA ISIDORA	SEGUNDO	REGULAR	GRUPO B	TARDE B
17	151000	GARCIA TACCA, JHANET GRISELDA	SEGUNDO	REGULAR	GRUPO C	TARDE B
18	202036	GUTIERREZ PEREZ, KATHERINE BRIGITTE	SEGUNDO	REGULAR	GRUPO B	TARDE B
19	150974	HUANCA MAMANI, PLAYER KLINTON	SEGUNDO	REGULAR	GRUPO C	TARDE A
20	150975	HUANCA MAMANI, YOBANA	SEGUNDO	REGULAR	GRUPO C	TARDE A
21	151003	HUAYLLAPUMA SANTA CRUZ, CARMEN ROSA	SEGUNDO	REGULAR	GRUPO C	TARDE B

Estudiantes

Listo 100%



ANEXO C: Guía y reglamento de matrícula

**UNIVERSIDAD PRIVADA
SAN CARLOS - PUNO**



**GUIA Y REGLAMENTO DE
MATRÍCULA
2016 - II**



AUTORIDADES DE LA UPSC

LIC. CARMEN ALCIRA DEL MAR AVILA
PRESIDENTA DE PROMOTORIA

DR. JUAN PASTOR HERRERA CARPIO
PRESIDENTE DE LA COMISION ORGANIZADORA UPSC

DR. ANARCO VALENCIA VARGAS
VICEPRESIDENTE ACADEMICO COMISIÓN ORGANIZADORA

DR. HIPOLITO PARI YUFRA
VICEPRESIDENTE ADMINISTRATIVO COMISIÓN ORGANIZADORA

DR. FELICIANO UTURUNCO MAMANI
RESPONSABLE C. P. DE CONTABILIDAD Y FINANZAS.

DRA. CARMEN LUZ RAMIRES RUIZ
RESPONSABLE C.P. DE ENFERMERIA

DR. VICTOR TORRES ESTEVES
RESPONSABLE C.P. DE DERECHO

DR. MARIO SOTO GODOY
RESPONSABLE C.P. DE INGENIERIA AMBIENTAL

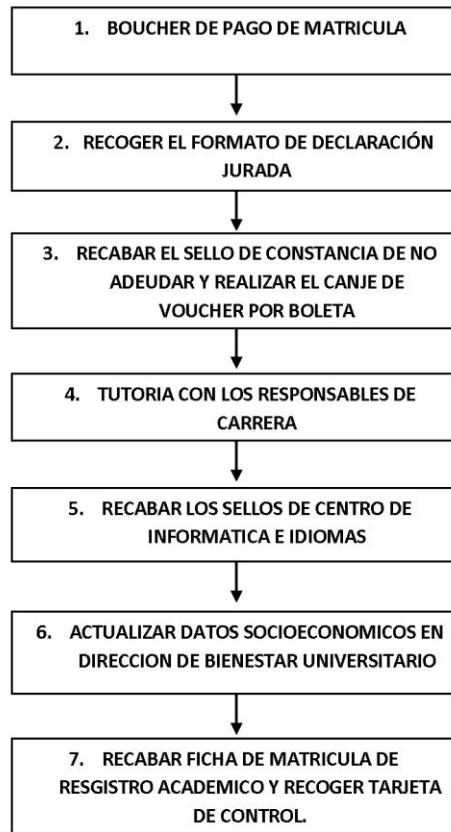
DR. JUDITH SALAZR DEL MAR
SECRETARIA GENERAL Y ASESORÍA LEGAL.

T. S. JUANA MARITZA MELO GONZALES
RESPONSABLE DE BIENESTAR UNIVERSITARIO.

Ms.C. MARIA VALLENAS GAONA
OFICINA DE REGISTRO ACADÉMICO



PROCESO DE MATRICULA





LOS ALUMNOS INGRESANTES:

Oficina de Registro Académico

Matricula:

1. Los ingresantes deberán acompañar:
 - a) Los documentos especificados en el Art. 10 del Reglamento de Matriculas
 - b) Fotografía y pago por Carné Universitario
2. Constancia de no adeudar (Biblioteca, C. Cómputo, C. Idiomas y otros servicios.
3. Constancia de haber llenado Ficha Socio Económica
4. Ficha de Matricula con el VºBº del Profesor Consejero
5. Finaliza matricula con el sello de MATRICULADO de la Oficina de Registro Académico



REGLAMENTO DE MATRICULA DE LA UNIVERSIDAD PRIVADA SAN CARLOS

TITULO I

FINALIDAD Y ALCANCES

Art. 1. Las disposiciones contenidas en el presente Reglamento norman los requisitos y procedimientos académicos del Sistema Curricular Flexible de la Universidad Privada San Carlos, de conformidad con las disposiciones de la Ley Universitaria 23733, Ley N° 26439 de Creación del CONAFU, Estatuto del CONAFU, incisos a) y b) del Art. 4º, y Estatuto de la Universidad.

Art. 2. El presente Reglamento establece las normas para procesos de matrícula procedimiento, y/o cancelación de las mismas, previstos en el Proyecto de Desarrollo Institucional aprobado por el Consejo Nacional para la Autorización de Funcionamiento de Universidades del País.

TITULO II

MATRICULA

Art.3. La matrícula es formal y voluntaria que acredita la condición de estudiante universitario, implicando su compromiso de cumplir la Ley Universitaria, Normas del CONAFU, el Estatuto y Reglamentos de la UPSC; mediante la observancia de un elevado nivel de comportamiento y dedicación a los estudios y actividades propias de la Universidad.

Art.4. La matrícula regular en cada semestre académico se realizará en fecha señalada por la Comisión de Gobierno de la Universidad, previa publicación.

Art.5. Durante la primera quincena, luego de iniciado oficialmente el dictado de clases; el estudiante podrá realizar modificación de matrícula con la autorización del profesor consejero y la Oficina de Registros Académicos.

Art.6. Podrá existir matrícula opcional en caso de programarse ciclo de verano, previa autorización del CONAFU, para dicho efecto se solicitará hasta el 31 de octubre la realización del ciclo en referencia.



Art.7. Los costos por derecho de matrícula serán fijados en el TUPA de la Universidad.

Art.8. El estudiante podrá matricularse en una Carrera Profesional y excepcionalmente en dos Carreras Profesionales, siempre que haya obtenido el derecho de ingreso, cumpla con los requisitos y pague las tasas establecidas.

Art.9. La Vicepresidencia Académica implementará y supervisará el proceso de matrícula. La ejecución es responsabilidad del Director de la Carrera Profesional en coordinación con la Oficina de Registros Académicos y la Dirección de Bienestar Universitario.

CAPITULO I

DE LOS ESTUDIANTES INGRESANTES

Art.10. Los requisitos para la matrícula del ingresante por las modalidades de examen ordinario, CEPRE y examen extraordinario, son:

- a) Partida de nacimiento original
- b) Certificados oficiales de estudios de los cinco años de Educación Secundaria, con todas las asignaturas aprobadas.
- c) Declaración jurada de no tener antecedentes penales y policiales (Formato Especial)
- d) Recibo de pago por derechos de matrícula, primera mensualidad y carné universitario.
- e) Constancia de examen de aptitud médica, para los estudiantes de la Carrera Profesional de Enfermería, otorgado por el Departamento Médico.
- f) Dos fotografías de frente tamaño carné, actualizado a colores y constancia de haberse tomado la foto digital para el Carné Universitario.
- g) Constancia de haber entregado la ficha socioeconómica a la Dirección de Bienestar Universitario.
- h) Constancia de registro en el Seguro Universitario (Opcional)
- i) Formato de compromiso de estudiante universitario.

Art. 11. Para la matrícula de estudiantes por traslado externo nacional e internacional; además de los documentos exigidos en el Art. 10. del presente Reglamento, deberán presentar:

- Certificado de estudios superiores de la Universidad de procedencia que acredite haber aprobado dos periodos lectivos semestrales completos o uno anual, acompañado de los sílabos de las asignaturas a convalidar.
- Constancia de no haber sido separado de la Universidad de procedencia por causas disciplinarias.

Art.12. Los requisitos para la matrícula de estudiantes por traslado interno, además de los documentos exigidos en el Art. 10. del presente Reglamento, deberán presentar:



- Certificado de Estudios Superiores originales, de haber aprobado dos semestres o un año académico en calidad de invicto expedido por la Dirección de Servicios Académicos de la UPSC, acompañado de los sílabos de las asignaturas a convalidar.

Art.13. Los requisitos para matrícula de estudiantes que ingresaron por la modalidad de Profesionales, además de los documentos exigidos en el Art. 10. del presente Reglamento, son:

- Certificado de estudios superiores originales, acompañado de los sílabos de las asignaturas a convalidar.

CAPITULO II

DE LOS ESTUDIANTES REGULARES

Art. 14. Son estudiantes regulares todos aquellos que aprobaron satisfactoriamente los créditos en que se han matriculado, en el semestre anterior y aquellos que repiten asignaturas por primera vez.

Art. 15. El promedio ponderado semestral (PPS) se calcula en base a la sumatoria del producto, de la nota de los cursos por el número de créditos; todo ello dividido entre la sumatoria de créditos matriculados en el semestre

$$PPS = \frac{\sum_{i=1}^K C_i N_i}{\sum_{i=1}^K C_i}$$

K = Número de cursos

C_i = Numero de créditos del curso

N_i = Notas del curso

Art. 16. Los estudiantes pueden matricularse en un máximo de 24 créditos, salvo en los siguientes casos:

Los estudiantes que hayan obtenido un promedio ponderado semestral entre 12 y 13 inclusive, pueden inscribirse hasta un máximo de 27 créditos.

Los estudiantes que hayan obtenido un promedio ponderado mayor de 13 pueden inscribirse en un máximo de 30 créditos.



Para acogerse a esta aplicación de créditos, los estudiantes deben registrar matrícula en el Semestre, con la autorización de su Profesor Consejero.

Art. 17. Los estudiantes regulares deben matricularse en las asignaturas cumpliendo con los prerrequisitos exigidos, caso contrario serán depurados automáticamente de oficio por la Oficina de Registros Académicos.

Art. 18. Los requisitos de matrícula, para los estudiantes regulares son:

Constancia de no adeudar a la Universidad expedida por Biblioteca, Centro de Cómputo, Centro de Idiomas, Laboratorios y Tesorería

Recibo de pago por derechos de matrícula, primera mensualidad y carné universitario.

CAPITULO III

DE LOS ESTUDIANTES CON TERCERA MATRICULA

Art. 19. Los estudiantes con tercera matrícula son aquellos que desaprueban una o mas asignaturas por segunda vez, debiendo matricularse en ellas obligatoriamente observando lo siguiente:

Si las asignaturas desaprobadas suman 12 créditos, se matriculan sólo en éstos.

Si suma menos de doce créditos, deberá matricularse en los cursos desaprobados y además puede completar los créditos restantes con otras asignaturas.

Si las asignaturas desaprobadas suman más de doce créditos, sólo se matriculan hasta en 12 créditos.

Art. 20. Los estudiantes que durante dos semestres consecutivos hubieran obtenido un promedio ponderado menor o igual a seis, serán considerados como estudiantes no regulares, se inscribirán en un máximo de 12 créditos durante el ciclo, se le considerara como OBSERVADO durante el semestre.

Art. 21. El Profesor Consejero registrará la palabra OBSERVADO en el Historial Académico del estudiante y consignará en la ficha de matrícula la información de las asignaturas cursadas en el semestre anterior con sus respectivas notas y créditos y los cursos en que se matricula durante el semestre en observación.



Art. 22. La condición de OBSERVADO será levantada si el estudiante aprueba la(s) asignatura(s), pudiendo continuar sus estudios como alumno regular; y si desaprueba nuevamente la(s) asignatura(s), será suspendido de la Universidad por un Semestre Académico.

CAPITULO IV

DE LOS ESTUDIANTES SUSPENDIDOS

Art. 23. Los estudiantes suspendidos son aquellos que se matriculan en una asignatura(s) por cuarta vez, luego de haber cumplido la sanción de suspensión por un Semestre Académico; en este caso se reincorporará a la Universidad y sólo se matricula en la asignatura (s) desaprobada (s); hasta un máximo de 12 créditos.

Art.24. La condición de CUARTA MATRICULA es levantada, si el estudiante aprueba la asignatura(s) pudiendo continuar sus estudios como alumno regular. En caso de ser reprobado es separado de la Universidad Privada San Carlos en forma definitiva.

CAPITULO V

DE LA MATRÍCULA POR CAMBIO DE PLAN DE ESTUDIOS

Art. 25. La matricula por cambio de plan de estudios, comprende a estudiantes que son alcanzados por un nuevo currículo y se tienen que adecuar al mismo, siempre que haya sido evaluado y autorizado por el CONAFU.

Art. 26. La matricula de los estudiantes por efecto de cambio de plan de estudios, se realizará de acuerdo a la Directiva de Convalidaciones del CONAFU, y por autorización de este Organismo.

CAPÍTULO VI

DEL PROCEDIMIENTO DE MATRÍCULA

Art. 27. El Profesor Consejero designado por el Responsable de la Carrera Profesional, verifica si las asignaturas propuestas por el estudiante, tienen aprobados los prerrequisitos o han sido



desaprobados por única vez, así como las demás exigencias de matrícula establecidas en el presente Reglamento.

Art. 28. La Biblioteca Central, Centro de Cómputo, Centro de Idiomas y otras dependencias de la Universidad harán alcance a los Responsables de las Carreras Profesionales, la relación de estudiantes deudores, a efectos que estos realicen la regularización respectiva.

Art. 29. La matrícula es personal o por carta poder. En el último caso quedará pendiente hasta ser ratificada por el mismo estudiante en un periodo máximo de 15 días.

Art. 30. El Registro de Matrícula será centralizado por la Oficina de Registros Académicos de la Universidad en coordinación con los Responsables de las Carreras Profesionales.

Art. 31. La Oficina de Registros Académicos deberá elaborar los Registros de Notas, Asistencia y Actas por cursos de cada Carrera Profesional, los mismos que serán alcanzados a los Responsables de Carrera; para su entrega a los docentes al inicio del semestre.

CAPÍTULO VII

DE LA ANULACIÓN, RETIRO Y CAMBIO DE ASIGNATURAS

Art. 32. Pierden derecho de matrícula:

Los ingresantes que no han hecho uso de su matrícula durante el año de ingreso, salvo casos justificados, previa solicitud dirigida al Responsable de la Carrera Profesional.

Los estudiantes que habiéndose matriculado y cursado por lo menos un (01) semestre dejaron de estudiar dos (02) semestres consecutivos.

Art. 33. El estudiante puede retirarse dentro de los 15 días después del inicio oficial de clases, en un máximo de tres (03) cursos matriculados, previa autorización de su Profesor Consejero y visto bueno del Responsable de la Carrera profesional, caso contrario para efectos del cálculo del promedio ponderado semestral y acumulado se incluirán los cursos no retirados aún con la nota cero (00), si el estudiante no hubiera realizado ninguna prueba de evaluación.



Art. 34. El Retiro Total del semestre solo procede por causas de fuerza mayor, que impidan al estudiante continuar sus estudios; la solicitud será resuelta por el Responsable de la Carrera Profesional. Así mismo, los alumnos que estén comprendidos en el marco de algún dispositivo legal que lo favorezca.

Art. 35. Si la Oficina de Registros Académicos o la Dirección de la Carrera Profesional detecta que la matrícula del estudiante excede en uno o más créditos de lo establecido, se procederá a la anulación primeramente de los cursos electivos y luego los obligatorios.

Art. 36. La matrícula en una asignatura sin haber cumplido con el Pre-Requisito establecido en el plan de estudios será anulado.

CAPÍTULO VIII

FORMACIÓN COMPLEMENTARIA

Art. 37. La UPSC, con la finalidad de promover la formación integral de sus estudiantes, reconocerá como formación complementaria y como requisito para el Bachillerato, de un máximo de cinco (05) créditos, en las siguientes actividades:

- Conferencias, seminarios, cursos, talleres y congresos en la especialidad, con una duración mínima de 16 Hs.
- Actividades artísticas, culturales y cívicas organizadas por la UPSC.
- Actividades deportivas organizadas por la UPSC.



ANEXO D:

**PREGUNTAS DE ENTREVISTA AL PERSONAL DE LA OFICINA DE
REGISTRO ACADÉMICO DE LA UNIVERSIDAD PRIVADA SAN CARLOS –
PUNO**

1. **¿Los docentes de la UPSC que reportes requieren?**
.....
.....
2. **¿El personal de Registro académico, que necesidades tiene?**
.....
.....
3. **¿Qué problemas tienen actualmente con el sistema anterior?**
.....
.....
4. **¿Qué tecnología se está usando con el sistema actual?**
.....
.....
5. **¿El sistema actual es suficiente para realizar su trabajo?**
.....
.....
6. **¿Qué documentos se tiene para implementar el sistema?**
.....
.....
7. **¿El sistema actual requiere de mantenimiento permanente?**
.....
.....
8. **¿Qué necesidades urgentes de información tienen los estudiantes?**
.....
.....
9. **¿Qué necesidades urgentes de información tienen los docentes?**
.....
.....
10. **¿Qué tipo de reportes requiere la institución y en qué formatos?**
.....
.....