ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

"IMPLEMENTACIÓN DE UN MODELO ALTERNATIVO DE ARQUITECTURA DE SOFTWARE, PARA PROYECTAR Y CONSTRUIR SISTEMAS LÓGICOS DE COMPUTADORAS."

MIGUEL ANGEL AVALOS PÉREZ

Tesis presentada ante la Escuela de Postgrado y Educación Continua de la ESPOCH, como requisito parcial para la obtención del grado de Magister en Informática Aplicada.

RIOBAMBA – ECUADOR

– 2012 –

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

CERTIFICACIÓN:

EL TRIBUNAL DE TESIS CERTIFICA QUE:

El trabajo de investigación titulado: ***"Implementación de un modelo alternativo de arquitectura de software, para proyectar y construir sistemas lógicos de computadoras"***, de responsabilidad del señor **MIGUEL ÁNGEL ÁVALOS PÉREZ**, ha sido prolijamente revisado y se autoriza su presentación:

Tribunal de Tesis

Msc. Narcisa Salazar		
PRESIDENTE		**FIRMA**
Msc. Washington Luna		
DIRECTOR		**FIRMA**
Msc. Diego Ávila		
MIEMBRO		**FIRMA**
Msc. Lorena Aguirre		
MIEMBRO		**FIRMA**

Riobamba, julio de 2012

DERECHOS INTELECTUALES

Yo, Ávalos Pérez Miguel Ángel, declaro que soy responsable de las ideas, doctrinas y resultados expuestos en la presente tesis, y que el patrimonio intelectual generada por la misma, pertenece exclusivamente a la Escuela Superior Politécnica de Chimborazo.

Miguel Ángel Ávalos Pérez
060277639-5

ÍNDICE GENERAL

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE ANEXOS

La presente tesis dedico con todo mi corazón a mis padres, a mi hermana Marthy, su esposo Angelito, y mis queridos sobrinos Diego y George. De manera especial a mi esposa Belén por su apoyo incondicional.

A mis abuelitos y a mi mejor amigo Víctor Hugo, que desde el cielo me miran y me cuidan. A todos mis amigos, quienes son lo más importante en mi vida.

Miguel Ángel.

Un sincero agradecimiento al Ing. Alfredo Colcha, por su aporte desinteresado en la elaboración de este trabajo, al Msc. Washington Luna, Director de tesis por su colaboración, a los miembros del tribunal Msc. Narcisa Salazar, Msc. Lorena Aguirre y Msc. Diego Ávila por su generosa contribución.

RESUMEN

El objetivo de este trabajo fue la de implementar un modelo alternativo de arquitectura de software para proyectar y construir sistemas lógicos de computadores, la misma que fue realizada en el departamento de desarrollo de sistemas informáticos de la fundación “San Luis” de la empresa Pronaca del cantón Bucay provincia del Guayas.

La investigación se desarrolló en torno a 4 partes fundamentales: recopilación de información de parte de los stakeholders y del equipo desarrollador de aplicaciones, análisis de arquitecturas de referencia, determinación e implementación del modelo de arquitectura a través de la modularización de las herramientas de programación .NET y finalmente la evaluación de la arquitectura propuesta con el desarrollo de un software para la automatización de procesos de carácter administrativo.

En base a los resultados que arrojó la investigación en cuanto a la funcionalidad y mantenibilidad, se pudo determinar que mejora en un 88% y 75% respectivamente, lo que permite evidenciar que la eficiencia de implementar la arquitectura de software mejora en un 82%.

En conclusión, una arquitectura de software bien estructurada permite orientar el desarrollo de aplicaciones hacia la generación de ventajas competitivas, provee soluciones bajo un manejo estratégico de recursos tanto humanos como empresariales e incrementa la calidad de los sistemas informáticos dentro de un esquema de mantenibilidad, reusabilidad y escalabilidad.

Se recomienda mejorar la estructura arquitectónica del software para desarrollar productos de calidad que cubran las necesidades empresariales a pesar de los cambios constantes que se puedan generar en el transcurso del tiempo.

SUMMARY

The purpose of this investigation was to create alternating architecture model software for projecting and constructing computer logic system. This work was conducted at the department of computer science systems in “San Luis Endowment” Pronaca district of Bucay, Guayas province.

The research was focused on four main parts as follows: Stakeholders and application developing team´s information compilation; referencing architecture analysis; architecture module by programing tool modularization based on NET devising and specification; and proposed architecture evaluation by means of software development for administrating process automation. The investigation shown that the function and maintainability results improved 88% and 75% respectively; which allowed the researcher to verify that the software architecture implementation had an 82% improvement.

As a way of conclusions it can be said that well-structured software architecture allows development application determination for generating the following prerogatives: competitive advantages; endowment solutions considering human as well as entrepreneuring resources for strategic management; computer science quality increase taking into account maintainability, reusability and scalability scheme.

In order to develop high quality products for entrepreneurial needs fulfillment, no matter how long it takes, it is recommended a software architectonic improvement.

INTRODUCCIÓN

El diseño de aplicaciones es un elemento fundamental en la organización de un sinnúmero de empresas, instituciones o centros de servicio público. Constituyen parte importante dela estructura de sus actividades internas y normalmente brindan soporte para la toma de decisiones así como para la optimización de tiempo y recursos. Hoy en día, con el auge de la tecnología, se han ido descubriendo e innovando también la construcción de los diversos sistemas informáticos, en base a las cuales se puedan resolver los problemas. A estas, se les ha denominado Arquitectura de Software.

Es así que, el objeto principal de esta investigación, es definir una arquitectura flexible y consistente, basándose en la utilización de patrones de diseño, para la construcción de aplicaciones empleando tecnología C#. Se pretende alcanzar un modelo de arquitectura que permita refinar y evolucionar la forma de programar dentro del departamento encargado del desarrollo de sistemas de la fundación “San Luis” de la empresa Pronaca.

El documento se encuentra dividido en cuatro (*4*) apartados. A continuación se señalan algunas características más importantes de cada una de ellas:

El primer apartado muestra un análisis inicial de la forma en que se llevan los procesos de creación de aplicaciones antes de inyectar un nuevo modelo de arquitectura. Aquí también se incluye los resultados más relevantes del estudio de campo realizado en los dos frentes, tanto del lado del cliente como del lado del diseñador de software.

El segundo apartado destaca la importancia del uso de las nuevas tecnologías en la construcción del software, en donde se engloban temas sobre arquitecturas de software, programación orientada a objetos, la herramienta de programación .NET y SQL Server.

En el tercer apartado se describe una introducción sobre arquitectura de software. Se abarcan temas como definiciones, historia, roles que debe cumplir un arquitecto de

SW, etc. Posteriormente se analizan las diferentes características de algunas arquitecturas que sirvieron de referencia para luego determinar la arquitectura final a implementar en base a las necesidades empresariales, del análisis del grupo desarrollador y el uso de dichas referencias.

En el cuarto y último apartado se muestra la aplicabilidad de esta investigación en un proyecto implementado bajo esta arquitectura, la misma que se encuentra actualmente en funcionamiento dentro de la fundación “San Luis”, específicamente en el instituto técnico superior “San Juan” del sector de Bucay. Se dilucida la función que realiza cada uno de los módulos y sub-módulos de la arquitectura propuesta, detallando ciertas recomendaciones y pautas de diseño. Además se recogen y analizan las observaciones del grupo desarrollador y se señalan las principales conclusiones y resultados obtenidos del trabajo efectuado.

En vista que los modelados de arquitectura hoy por hoy ocupan un lugar importante en la tecnología de software, se propone contribuir con un modelo que permita a los diseñadores del departamento de sistemas de la fundación, mejorar y perfeccionar la manera en que actualmente se lleva a cabo la planificación, diseño y puesta a punto de las aplicaciones implementadas y a implementarse. Esto significaría una mejora considerable en este proceso en cuanto a eficiencia, tiempo y organización, contribuyendo favorablemente en el desarrollo de aplicaciones que cubran las necesidades crecientes de la fundación y del equipo de trabajo en el desarrollo de SW.

CAPÍTULO 1

PROBLEMATIZACIÓN

1.1. Planteamiento del problema.

El crecimiento acelerado de empresas, instituciones o centros de atención al público en los últimos años, ha generado la necesidad de automatizar procesos y resguardar información relevante para la consecución de tareas internas o externas a ellas. Esto genera cada vez más la exigencia de innovación tecnológica hardware/software que se presentan en los establecimientos.

El llevar un control adecuado de su información, el seguimiento de las actividades del personal, como también el tener los documentos en orden y perfectamente resguardados, requieren de la implementación de sistemas tecnológicos que logren garantizar su permanencia y cuidado, con aplicaciones cada vez más eficientes, donde su programación no tenga un fuerte impacto al momento de modificar o actualizar sus procedimientos.

La fundación “San Luis” promovida por la empresa Pronaca, aporta significativamente en amplios campos de índole social; uno de ellos es el campo educativo. Varios centros educativos a nivel nacional, brindan sus instalaciones para la educación de niños, jóvenes y adultos, además de capacitar a su personal operativo y administrativo de la empresa Pronaca.

Por este motivo, los últimos años ha gozado un crecimiento gradual, lo que ha generado también un crecimiento en la información que internamente maneja. Con la finalidad de cubrir esta necesidad, se ha volcado la mirada al uso de aplicaciones que permitan la automatización de sus actividades administrativas.Se requiere cada vez de múltiples aplicaciones que cubran con las expectativas de su organización, en lo posible con menos tiempo y costo tanto en su diseño como en su mantenimiento.

El departamento de desarrollo de aplicaciones de la fundación, es el grupo encargado de la creación y mantenimiento de SW, pero se carece de los fundamentos necesarios para que los programadores trabajen en una línea común, limitados muchas veces por problemas con el escalamiento de la aplicación, falta de flexibilidad, reusabilidad, interfiriendo grandemente en su evolución y mantenimiento en caso de errores.

1.2. Justificación.

El uso de modelos de arquitectura, es una evolución en el desarrollo de aplicaciones. Esta herramienta metodológica permite la creación de aplicaciones bajo un esquema de modelamiento y la abstracción de los objetos que representan un problema, logrando así estandarizar y organizar procesos, acortando la brecha entre el diseño de aplicaciones y la calidad que se quiere alcanzar.

El enfoque particular que se pretende conseguir al momento de aplicar una arquitectura de SW es la reutilización de componentes, lo cual logra que los desarrollos a posteriori utilicen recursos existentes sin incurrir en gastos de desarrollos adicionales.[1] Esto permite reducir el tiempo de diseño del proyecto, el esfuerzo que requiere implementar una aplicación y los costos en general. Si se logran reducir estos tres aspectos se lograría incrementar el nivel de productividad en los equipos de desarrollo y minimizar el riesgo global del proyecto, así como los costos de este. De esta manera las pequeñas o grandes empresas pueden tener una mayor confiabilidad si quieren realizar una inversión tecnológica.

Otro gran beneficio de trabajar bajo una estructura arquitectónica eficiente de SW, es el poder integrar lo mejor de varias tecnologías (*desarrollos multiplataforma*), para desarrollar aplicaciones de manera personalizada que se adapten a las necesidades organizacionales evitando, incurrir en gastos de licenciamiento, actualización o soporte de dichas soluciones.

Hoy en día, el uso de aplicaciones personalizadas, se ha convertido en una herramienta importante que puede ser un factor decisivo respecto a su competitividad en el mercado. El software es el elemento tecnológico de más alcance y de mayor

[1] **BERTRAN., M.,** Significado de componentes., Canverra-Australia., CMP Books., 2000., Pp. 212-221.

aceptación por parte de los sectores de servicios, micro establecimientos e industria según la Asociación Ecuatoriana de Software (*Aesoft*).

Esto demuestra que estas herramientas son necesarias y pueden ofrecer una importante ventaja competitiva. De acuerdo a esta investigación, el 54.4% en el sector público, el 41.1% en micro establecimientos y 46.6% en la industria utilizan software diseñado dentro de su propia organización.

A pesar de que las cifras permiten concluir de que el uso de aplicaciones propias es una alternativa estratégica para mejorar la eficiencia en las actividades internas de los establecimientos, aún existen varias barreras que impiden emprender su diseño en torno a factores de calidad y enfocados en desarrollarlos en ambientes adecuados de organización, flexibilidad y crecimiento.

Esta falta visionaria de utilizar nuevas herramientas informáticas, técnicas atractivas y viables para el desarrollo de aplicaciones en un contexto cada vez más dinámico, exigente e integrado, impiden la modernización y competitividad.

Es por esta razón que se pretende con el desarrollo de la investigación, definir un modelo de arquitectura estándar, flexible y consistente para el desarrollo de aplicaciones, basándose en la utilización de modelos referenciales de diseño que facilite su implementación, interoperabilidad de módulos, mejore su escalabilidad y flexibilidad, minimizando el trabajo y esfuerzo del personal del área de sistemas, describiendo ciertas recomendaciones y pautas de diseño, haciendo hincapié en los puntos más débiles que habitualmente aparecen en este tipo de proyectos, como la gestión de usuario, escalabilidad, reusabilidad, etc.

Paralelo a estas ventajas, se encuentra el mejoramiento en la calidad de atención al cliente o usuario final y reducir costos de implementación, desarrollo y actualización de aplicaciones. Lo importante es poder aportar con un proyecto de investigación que permita construir oportunidades reales de progreso informático hacia una perspectiva de organización, producción y crecimiento.

1.3. Objetivos.

1.3.1. General.

Implementar un modelo alternativo de arquitectura de software, para proyectar y construir sistemas lógicos de computadoras.

1.3.2. Específicos.

- Determinar el estado actual de los procesos de desarrollo de software para establecer las necesidades informáticas en el departamento de sistemas de la Fundación San Luis.

- Analizar las diferentes arquitecturas de software de referencia que representen ventajas en el proceso de definición de un modelo de arquitectura.

- Diseñar y emplear un modelo alternativo de arquitectura de software en el desarrollo de una aplicación.

- Promover el uso del modelo de arquitectura propuesto con la participación de los programadores de aplicaciones de la Fundación.

- Realizar un seguimiento de la efectividad de la arquitectura propuesta.

1.4. Marco teórico referencial.

Antecedentes.

En los últimos años, los grupos desarrolladores de software se encuentran con varios paradigmas de infraestructura al momento de crear e implementar aplicaciones. En general, cuando hablamos de estrategias para la construcción de sistemas, existe un criterio ampliamente difundido, pero tal vez no en la misma medida aplicado: “la construcción de sistemas **útiles** y de **calidad.**”

- **Útiles:**
  - Que el software cumpla con los requerimientos del usuario.
  - Que el software tenga éxito, o sea que la organización logre con él, algo que no alcanzaría sin ellos.

- **De calidad:**
  - Que el software cumpla con un conjunto de “áreas clave” del desarrollo de sistemas.

El equipo encargado del desarrollo de proyectos informáticos en la fundación San Luis de la empresa Pronaca, realiza su trabajo con la óptica tendiente a desarrollar aplicaciones que satisfagan los requerimientos delos usuarios directos e indirectos.

Lamentablemente, a medida que se han ido incorporando nuevas funcionalidades, el equipo desarrollador se ha visto inmerso en grandes “cuellos de botella” surgidos en el análisis y diseño del mismo, lo que ha conllevado a importantes desvíos temporales en la ejecución y puesta a punto de la aplicación.

El riesgo de no poder cumplir con los requerimientos del proyecto en tiempo y forma, hace que empiece a aparecer y crezca rápidamente, problemas de calidad en el producto final y en su proceso de diseño, con la posible consecuencia del estancamiento del proyecto. Este es un escenario común que se mantiene, no solo en este grupo desarrollador, sino en muchos equipos de proyectos informáticos.

La crisis de software que se ha presentado en el tratamiento de los problemas de diseño y por ende en la calidad del producto final, se atribuye principalmente a la falta de productividad. Es aquí en donde se hace necesaria la participación activa de las personas involucradas directa o indirectamente en el uso de las herramientas del sistema, con la finalidad de lograr identificar en forma global los objetivos de calidad que debe alcanzarla aplicación. Sin objetivos de calidad que guíen a los desarrolladores, las oportunidades de mejorar el producto son arbitrarias y la medición del sistema durante las revisiones de diseño seguirá siendo insostenible.

Esta problemática, dio paso a la aplicación de un estudio de campo inicial, con la finalidad de obtener una visión clara de las necesidades de los stakeholders y de los

diseñadores de software. Los resultados obtenidos servirán de parámetro para la toma de decisiones en la aplicación de nuevas estrategias de programación.

Dentro de la fundación, se aplicó encuestas a 32 usuarios (*ver anexo 1*) y a 9 desarrolladores de la aplicación (*ver anexo 2*). Los resultados se pueden visualizar en el anexo 3.

En razón de lo analizado, se puede observar que del lado del cliente se tiene una tendencia directa al uso del sistema operativo Windows (100% de usuarios). Esto es entendible, debido a que prácticamente para la mayoría de personas, Windows es el SO más difundido y los conocimientos adquiridos en cuanto al manejo de equipos de cómputo como el ambiente tecnológico que le rodea, se basa en esta plataforma operativa.

Otro punto a observar es la predisposición que tienen las personas al uso de aplicaciones para sus actividades diarias de trabajo, más aún si estas se ajustan directamente a cumplir con las necesidades de su entorno. De las personas entrevistadas, el 81% afirman que el uso de una aplicación personalizada, tiene una gran importancia como herramienta para una mejor producción laboral, sin dejar de lado el otro 19% del sector entrevistado que piensa que tiene una importancia significativa aunque sus labores no dependan de ella.

Cabe indicar, que la aplicación que se ha diseñado e implementado dentro de la Fundación, ha brindado los beneficios para la cual fue construida. El 100% de personal entrevistado, apoya la inclusión de este tipo de material informático. De estos, el 78% indican que han obtenido suficientes beneficios con las herramientas implementadas y el 22% restante revelan que, a pesar de ser importante para efectuar sus labores cotidianas, aún faltan por tratar ciertos detalles para ser un producto de mejor calidad. Esto indica que aún existe cierto grado de inconformidad (*31% neutrales y 9% inconformes*) no exclusivamente por el funcionamiento de la aplicación sino más bien por detalles externos a ella, como por ejemplo:

- → *falta de interpretación de detalles (35%),*
- → *problemas de fiabilidad (6%),*
- → *inconvenientes en su comportamiento temporal (6%),*

- → *complicaciones en la seguridad de acceso (3%),*
- → *ciertos conflictos en la usabilidad (12%),*
- → *problemas de funcionalidad (13%),*
- → *molestias en la disponibilidad (10%),*
- → *poca evolución en la aplicación (59%),*
- → *mucho tiempo en la Modificabilidad (64%).*

Por otra parte, realizado el análisis en el lado del diseñador de aplicaciones, se puede percibir que al igual que en el caso anterior (*lado del operador*), uno de los sistemas operativos de mayor uso es el Windows (*100 %*). Cabe indicar que de este porcentaje se pueden derivar diseñadores con conocimientos en otras plataformas operativas. Además, con los datos obtenidos, se puede deducir que el lenguaje dominado por los programadores, se basa en los lenguajes C# y Visual Basic. Este es un factor imprescindible para definir el ambiente de trabajo futuro para el diseño de las aplicaciones, ya que se debe establecer un ambiente de trabajo común a todos los diseñadores, sin sufrir fuertes impactos en el cambio.

Los avances en los estándares de la tecnología y la industria del software, están también orientados al cambio de paradigma en la arquitectura. A pesar de esta realidad, varios de los diseñadores de SW no se sujetan directamente a una planificación previa de una arquitectura para estandarizar los procesos evolutivos de programación de aplicaciones. Luego de la encuesta efectuada internamente, se pudo determinar que se tiene un ambiguo conocimiento en lo que infraestructura arquitectónica se refiere. La tendencia es programar bajo una arquitectura tradicional estructurada o cliente – servidor (*100 %*). Algunos de los diseñadores afirman conocer ciertos detalles sobre la arquitectura multicapa y distribuida, aunque no han programado bajo estos tipos de estructura.

Todos estos parámetros han dado lugar a que los procesos de diseño de aplicaciones tengan más de un inconveniente. Tan solo el 22% de diseñadores afirman no haber tenido ningún problema al momento de realizar su trabajo. Esto puede justificarse debido a que no están directamente relacionados con la parte procedimental sino a que se dedican al diseño del front-end de la aplicación. El 78% restante han encontrado serios problemas al momento de llevar a cabo su labor de programación, entre ellos, mucho tiempo en el diseño (*86%*), problemas en la creación de nuevos

requerimientos (*71,43%*), inconvenientes en la localización y reparación de errores (*57,14%*) entre los problemas de mayor relevancia.

Se asumen que estos problemas tienen que ver preferentemente por no manejarse bajo una estructura estándar que organice y facilite las tareas de programación. El 71,43% de entrevistados coinciden en que la falta de implementación de una arquitectura organizada, puede ser un factor determinante en el surgimiento de problemas, lo que genera además un crecimiento desordenado de la aplicación (*57,14%*). Se ha propuesto alternativas nuevas de adquirir, aprender y utilizar nuevas herramientas tecnológicas que podrían ayudar a mejorar el trabajo de programación. La mayoría de entrevistados (*75%*) están seguros y conscientes de tomar otras alternativas de solución. Pero de ellos, tan solo el 33% tienen sugerencias sobre estrategias destinadas a mejorar el trabajo y que podrían servir de base para experimentar sus efectos en la programación.

Es importante especificar los requisitos de calidad del producto a diseñar, para lo cual se deben seleccionar los aspectos inherentes a la calidad deseada del producto. Uno de los puntos a analizar para la determinación de la arquitectura, es la prioridad de los atributos de calidad según las características particulares del producto que se está diseñando. Analizando los resultados de la encuesta efectuada, se obtuvo el siguiente orden de prioridad:

1. Mantenibilidad,
2. Fiabilidad,
3. Funcionalidad,
4. Eficiencia.
5. Usabilidad,

Es importante analizar otros factores de calidad y su prioridad debido a que el ciclo de vida del software es largo. Según el análisis de la encuesta, el orden de prioridad estaría determinado de la siguiente manera:

1. Modificabilidad,
2. Reusabilidad,
3. Escalabilidad,
4. Testeabilidad,

5. Portabilidad,
6. Integrabilidad,
7. Interoperabilidad.

En resumen, entre los motivos de la crisis interna al momento de diseñar el software se pueden mencionar:

- Baja comunicación o falta de entendimiento entre usuarios y diseñadores.
- No existe una definición consensuada de las características o atributos de calidad que hay que medir.
- Los requerimientos no funcionales no son especificados en el tiempo o no son estudiados con la atención explícita de los riesgos que involucran.
- No existen construcciones de sistemas modulares, que permitan bajar el esfuerzo de ajuste y extensión.
- La mantenibilidad no es considerada inicialmente en el diseño del sistema. Esto quiere decir que nunca se ha pensado al sistema en función del cambio. El esfuerzo de mantenimiento es alto.
- Crecimiento desorganizado de la aplicación.

Entonces, la solución es obviamente incrementar la productividad, que significa resolver todos los problemas que dan lugar a la crisis. Para esto es prudente establecer “áreas clave”. Según la administración, no todas las áreas de una gestión brindan el mismo rendimiento ante idénticos esfuerzos, es decir, que en algunas las mejoras “por unidad de esfuerzo” serán más significativas que en otras. Esas son las llamadas “áreas clave”, y en principio son las áreas sobre las que se ha desarrollar nuestra gestión.[2] Todos coinciden que una opción interesante sería mejorar la arquitectura actualmente implementada y proyectarse a alcanzar un bajo acoplamiento entre los módulos de la aplicación.

[2] **BASPINEIRO., A.,** Planificación, Control y Calidad del SW., Salta-Argentina., Ed. Kapeluz., 2006., Pp. 2.

CAPÍTULO 2

MARCO TEÓRICO

2.1. Tendencias y tecnologías actuales.

A lo largo de los años hemos visto como los lenguajes de programación han ido evolucionando de una manera vertiginosa. Si a inicios de la programación veíamos como las aplicaciones se desarrollaban con códigos binarios, hoy podemos abstraer la realidad de nuestro entorno con ayuda de herramientas muy completas como la programación orientada a objetos.[3]

En la actualidad existen diversas herramientas y metodologías de desarrollo de aplicaciones. Podemos ver que esta nueva generación de lenguajes busca que la programación sea más sencilla, que las instrucciones se acerquen cada vez más al lenguaje natural y que los tiempos de desarrollo de aplicaciones se reduzcan de forma drástica.

En el entorno de programación, existen los lenguajes del tipo "interpretados", es decir, que para ejecutarlas instrucciones existe un programa o intérprete que se encarga de procesar cada una de las órdenes y producir los resultados deseados. Algunos ejemplos de este tipo de lenguajes son: JavaScript, Lisp, PHP, Python, Ruby, etc. En el otro caso, en los lenguajes que no son interpretados, existe un compilador que toma esas instrucciones y genera un archivo ejecutable. Tenemos en esta categoría a C, C++, Java, Microsoft Visual Basic, etc.

En las nuevas tecnologías de programación, los lenguajes tienen grandes ventajas frente a la forma de programar clásica. Algunas de las características más sobresalientes son:

---

[3] **ROBLES**, Vicente– "Lenguajes de Scripting: ¿u **ROBLES., V.,** Lenguajes de Scripting: ¿una nueva forma de programar?., Universidad Politécnica Salesiana., PUBLICACIÓN., Cuenca-Ecuador., 2008., Pp. 28-30

- No se requiere ciclo de compilación: en este tipo de tecnología de software, los lenguajes no necesariamente compilan el código fuente, sólo se crea el programa y se lo ejecuta.

- Programación más simple: no se requieren declarar los tipos de las variables, ni usar paréntesis para llamar a los métodos. Los arreglos se declaran de forma directa (*sin necesidad de instancias*).

- Orientación a objetos: se pueden crear módulos, clases e interfaces.

- Control de excepciones: se tiene un robusto sistema para control y recuperación de errores.

A continuación se muestra un índice de los mejores lenguajes de programación según la empresa TIOBE Software:

Tabla 1. Índice TIOBE de los mejores lenguajes de programación.
Fuente: TIOBE Software

Position Apr 2011	Position Apr 2010	Delta in Position	Programming Language	Ratings Apr 2011	Delta Apr 2010	Status
1	2	↑	Java	19.043%	+0.99%	A
2	1	↓	C	16.162%	-1.90%	A
3	3	=	C++	9.225%	-0.48%	A
4	6	↑↑	C#	7.185%	+2.75%	A
5	4	↓	PHP	6.584%	-3.08%	A
6	7	↑	Python	4.931%	+0.73%	A
7	5	↓↓	(Visual) Basic	4.682%	-1.71%	A
8	11	↑↑↑	Objective-C	4.386%	+2.10%	A
9	8	↓	Perl	1.991%	-1.56%	A
10	10	=	JavaScript	1.513%	-0.96%	A
11	12	↑	Ruby	1.482%	-0.74%	A
12	20	↑↑↑↑↑↑↑↑	Lua	1.035%	+0.51%	A

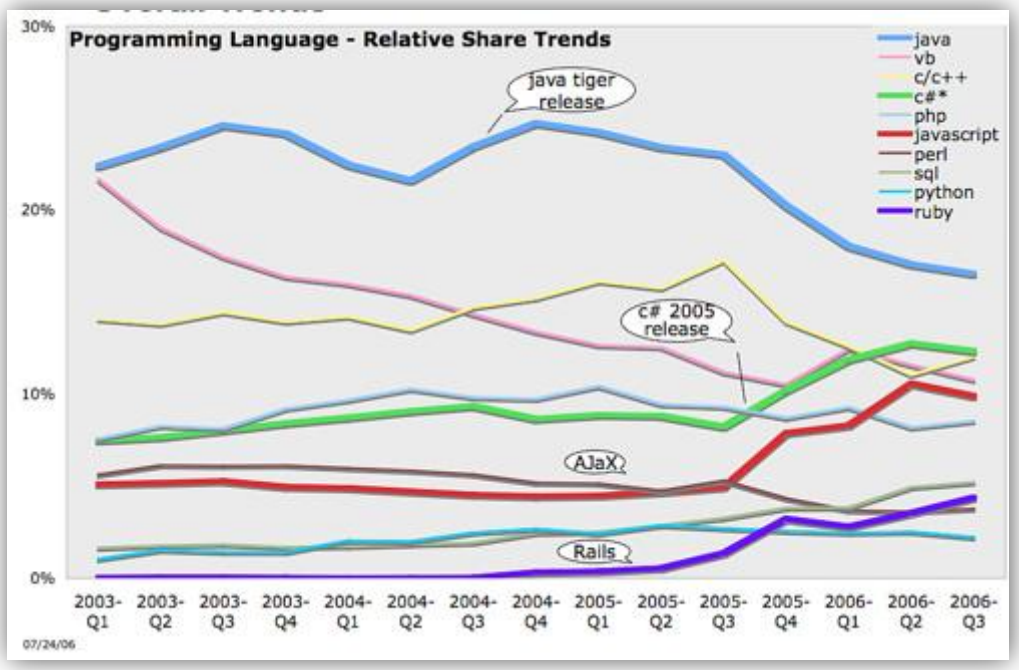

Figura 1. Tendencias de los lenguajes de programación.
Fuente: Tim O'Relly.

2.1.1. Tecnología orientada a objetos.

Hoy en día la tecnología orientada a objetos ya no se aplica solamente a los lenguajes de programación, además se viene aplicando en el análisis y diseño con mucho éxito, al igual que en las bases de datos. Es que para hacer una buena programación orientada a objetos hay que desarrollar todo el sistema aplicando esta tecnología, de ahí la importancia del análisis y el diseño orientado a objetos.

La programación orientada a objetos es una de las formas más populares de programar y viene teniendo gran acogida en el desarrollo de proyectos de software desde los últimos años. Esta acogida se debe a sus grandes capacidades y ventajas frente a las antiguas formas de programar.[4]

Tradicionalmente, la programación fue hecha en una manera secuencial o lineal, es decir una serie de pasos consecutivos con estructuras consecutivas y bifurcaciones.

[4] http://generacion-vc.blogspot.com/

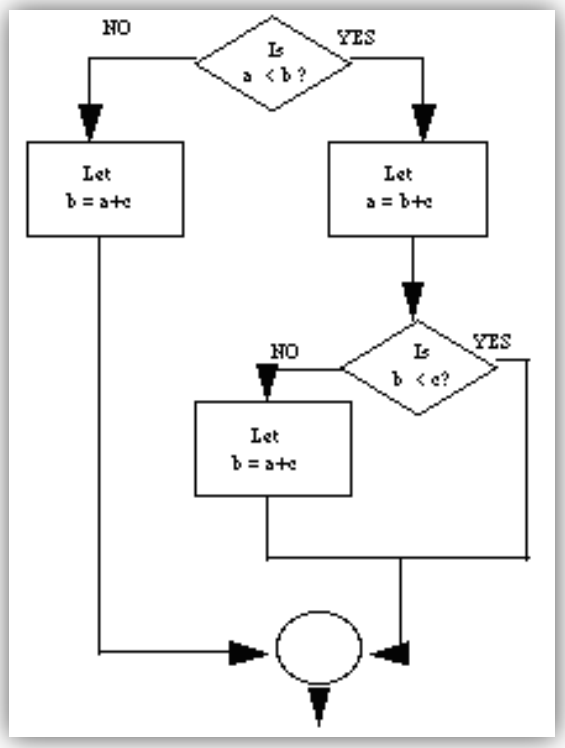

Figura 2. Diagrama de flujo con bifurcaciones.
Fuente: blog "Tecnología Orientada a Objetos". 2008 - http://generacion-vc.blogspot.com/

Los lenguajes basados en esta forma de programación ofrecían ventajas al principio, pero el problema ocurre cuando los sistemas se vuelven complejos. Estos programas escritos al estilo "espaguetti" no ofrecen flexibilidad y el mantener una gran cantidad de líneas de código en sólo bloque se vuelve una tarea complicada.

Frente a esta dificultad aparecieron los lenguajes basados en la programación estructurada. La idea principal de esta forma de programación es separar las partes complejas del programa en módulos o segmentos que sean ejecutados conforme se requieran. De esta manera tenemos un diseño modular, compuesto por módulos independientes que puedan comunicarse entre sí. Poco a poco este estilo de programación fue remplazando al estilo "espaguetti" impuesto por la programación lineal. La evolución que se fue dando en la programación se orientaba siempre a ir descomponiendo más el programa. Este tipo de descomposición conduce directamente a la programación orientada a objetos.

Pues la creciente tendencia de crear programas cada vez más grandes y complejos llevó a los desarrolladores a crear una nueva forma de programar que les permita crear sistemas de niveles empresariales y con reglas de negocios muy complejas. Para estas necesidades ya no bastaba la programación estructurada ni mucho menos la programación lineal. Es así como aparece la programación orientada a objetos (*POO*).

La POO viene de la evolución de la programación estructurada; básicamente la POO simplifica la programación con la nueva filosofía y nuevos conceptos que tiene. La POO se basa en la dividir el programa en pequeñas unidades lógicas de código. A estas pequeñas unidades lógicas de código se les llama objetos. Los objetos son unidades independientes que se comunican entre ellos mediante mensajes.

Entre las ventajas que presenta un lenguaje orientado a objetos se pueden definir:[5]

- Fomenta la reutilización y extensión del código.
- Permite crear sistemas más complejos.
- Relacionar el sistema al mundo real.
- Facilita la creación de programas visuales.
- Construcción de prototipos.
- Agiliza el desarrollo de software.
- Facilita el trabajo en equipo.
- Facilita el mantenimiento del software.

Lo interesante de la POO es que proporciona conceptos y herramientas con las cuales se modela y representa el mundo real tan fielmente como sea posible. Las nuevas tendencias en el software conllevan a que la industria del software entre en un periodo de cambios importantes entre los que destacan:[6]

a) Reutilización de componentes
b) Conectividad Abierta: Independencia de las Bases de Datos.

[5] http://generacion-vc.blogspot.com/
[6] www.geocities.com/eztigma

a) Reutilización de componentes

La reutilización de componentes implica la construcción de soluciones con equipo lógico que ya existía o que construyen terceros. La ventaja principal que aporta es que esta reutilización permite generar aplicaciones eficientes y de gran fiabilidad. El principal problema radica en el hecho de que no existe ningún marco de normalización o acuerdo para que los componentes creados por distintos fabricantes puedan trabajar conjuntamente.

Los **componentes** son bloques de construcción de aplicaciones. Los "constructores de soluciones" utilizan muchos componentes software para la realización de sus sistemas .El concepto de reutilización de componentes abarca el equipo lógico existente para tareas básicas y genéricas como impresión, procesadores de textos, hojas de cálculo, gráficos, diagramas de barras y dibujos. Todas estas piezas deberían estar disponibles como componentes reutilizables para todas las soluciones que los necesiten.

El **modelo de objetos** formaliza la estructura y el comportamiento de los componentes para que puedan trabajar conjuntamente. El modelo ve a los componentes como objetos y utiliza los conceptos de orientación a objetos para definir el marco de desarrollo de los mismos. El problema es que no existe uniformidad en el desarrollo de los componentes para que puedan comunicarse y trabajar conjuntamente. Las librerías de funciones son válidas para algunas aplicaciones, pero tampoco permiten construir un auténtico componente reutilizable, ya que no hay ninguna forma estándar de definir e implementar funciones y las librerías de suministradores distintos no trabajan conjuntamente. El éxito de esta aproximación depende de que sea aceptada y se imponga como estándar de desarrollo de componentes.

Si se acepta de forma universal el modelo de objetos para la construcción de componentes, significa que aparecerá una nueva industria de creación de componentes genéricos que estarán disponibles. Las aplicaciones más habituales (*procesadores, gráficos, etc*.) se encontrarán disponibles en forma de componentes que se podrán integrar para conseguir nuevas aplicaciones de gran flexibilidad y

potencia. Lo único necesario es un lenguaje de programación común para ensamblar los distintos componentes y construir la aplicación.[7]

Para construir una aplicación compleja, se dispondrá por tanto de componentes genéricos fabricados por terceros y que se integrarán junto con los componentes específicos desarrollados para la aplicación concreta. Esto permite concentrar el esfuerzo del desarrollador en las partes de la aplicación que son competencia suya y poder así desarrollar soluciones potentes de una forma muy rápida.

Un **lenguaje de programación común** es un elemento crítico que es necesario para hacer que la reutilización de componentes sea una realidad. Este lenguaje común debe ser:

- Común entre plataformas, aplicaciones y componentes en general.
- Técnicamente completo (*estructuras de control, tipos de datos, control de parámetros, soporte a la depuración, etc.*).
- Fácil de utilizar, para que sea aceptado por los desarrolladores.
- Disponible y poco costoso.

Este lenguaje común permite a los desarrolladores de soluciones rentabilizar los componentes disponibles en el mercado y preservar sus inversiones en el lenguaje de desarrollo para todo tipo de plataformas. Desde otro punto de vista, también se hace necesaria la utilización de herramientas para la creación de esos componentes. La idea central detrás de la reutilización de componentes es la de ocultar a los desarrolladores de soluciones la complejidad que existe detrás del componente genérico. Para el desarrollador, el componente se muestra con una interfaz muy sencilla, aunque realice cálculos complejos y manipulaciones de datos de forma interna.

Estas herramientas deben:

- Ser lo suficientemente potentes como para manejar la compleja infraestructura de información, permitiendo aplicaciones eficientes de bases de datos, aplicaciones de intercambio de información en tiempo real, soporte de arquitecturas distribuidas, etc.

[7] www.geocities.com/eztigma

- Permitir un desarrollo orientado a objetos, para que la complejidad de la implantación quede oculta al desarrollador de soluciones que utiliza el componente.
- Crear componentes robustos y que ofrezcan altos niveles de rendimiento.

b) Conectividad abierta: Independencia de las Bases de Datos.[8]

En las grandes organizaciones cada vez se hace más importante la existencia de conectividad abierta a los datos existentes con independencia del formato o la plataforma.

Para la toma de decisiones, se hace necesario acceder desde un ordenador personal a datos corporativos que se encuentran en bases de datos remotas. Muchas de estas bases de datos tienen una interfaz dedicada, que no tiene mucha relación con las aplicaciones estándar de ordenador personal. Al mismo tiempo, muchas aplicaciones corporativas están siendo transportadas a ordenadores personales. Estas dos tendencias convergen en la arquitectura cliente/servidor.

Desde esta arquitectura se hace necesario proporcionar características de conectividad abierta, para permitir que los PCs puedan acceder a bases de datos heterogéneas.

Algunos suministradores proporcionan ya herramientas para conseguir esta conectividad abierta de bases de datos a través de una Arquitectura de Servicios Abiertos:

- Sin la existencia de un método formal de conexión entre aplicaciones front-end y servicios, los desarrolladores de aplicaciones están forzados a incorporar APIs específicas de los suministradores en sus aplicaciones. Esta es una labor intensa y costosa.
- Una arquitectura de servicios abiertos soluciona este problema presentando a los desarrolladores de aplicaciones distribuidas una interfaz común para todos los servicios que soporte la arquitectura. En lugar de tener que aprender APIs diferentes, sólo existe el API de la arquitectura de servicios abiertos.

[8-9] www.geocities.com/eztigma

En fin, la conectividad abierta a BD:[9]

- Es un componente de la Arquitectura de Servicios Abiertos.
- Es la interfaz para acceder a los datos en entornos heterogéneos de sistemas de gestión de bases de datos relacionales y no-relacionales.
- Define una API común para acceder a todas las bases de datos que se ajusten a esta conectividad (*que tengan el **driver** correspondiente*).
- Permite desarrollar aplicaciones que, de forma concurrente, acceden, modifican y consultan datos de bases de datos múltiples.
- Cada SGDB tiene su propia API.
- Destacan como estándares de facto ODBC (*Open DataBase Connectivity*) e IDAPI (*Independent Database Application Programming Interface*).

Existen en el mercado herramientas de construcción de aplicaciones que utilizan tanto las facilidades de componentes reutilizables como la conectividad abierta a bases de datos.

Otra de las alas en las que se deriva el uso de nuevas tecnologías se lo conoce con el nombre de “Arquitectura de Software”, (*tema motivo de esta investigación*), que se analizará más adelante.

2.1.2. Herramientas de programación.NET.

.NET es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones. Basado en ella, la empresa intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el sistema operativo hasta las herramientas de mercado.[10]

.NET podría considerarse una respuesta de Microsoft al creciente mercado de los negocios en entornos Web, como competencia a la plataforma Java de Sun Microsystems y a los diversos framework de desarrollo web basados en PHP. Su propuesta es ofrecer una manera rápida y económica, a la vez segura y robusta, de

[10]http://es.wikipedia.org/wiki/Microsoft_.NET

desarrollar aplicaciones permitiendo una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

Microsoft .NET es más que un lenguaje de programación. Es un conjunto de tecnologías de software, compuesto de varios lenguajes de programación que se ejecutan bajo el .NET Framework. Es además un entorno completamente orientado a objetos y que es capaz de ejecutarse bajo cualquier plataforma.

a) Consideraciones de .NET.

La plataforma .NET de Microsoft es un componente de software que puede ser añadido al sistema operativo Windows. Provee un extenso conjunto de soluciones predefinidas para necesidades generales de la programación de aplicaciones, y administra la ejecución de los programas escritos específicamente con la plataforma. Esta solución es el producto principal en la oferta de Microsoft, y pretende ser utilizada por la mayoría de las aplicaciones creadas para la plataforma Windows.

Microsoft.NET es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando durante los últimos años con el objetivo de obtener una plataforma sencilla y potente para distribuir el software en forma de servicios que puedan ser suministrados remotamente y que puedan comunicarse y combinarse unos con otros de manera totalmente independiente de la plataforma, lenguaje de programación y modelo de componentes con los que hayan sido desarrollados. Ésta es la llamada **plataforma .NET**, y a los servicios antes comentados se les denomina **servicios Web**.[11]

Para crear aplicaciones para la plataforma .NET, tanto servicios Web como aplicaciones tradicionales (*aplicaciones de consola, aplicaciones de ventanas, servicios de Windows NT, etc.*), Microsoft ha publicado el denominado kit de desarrollo de software conocido como **.NET Framework SDK**, que incluye las herramientas necesarias tanto para su desarrollo como para su distribución y ejecución y **Visual Studio.NET**, que permite hacer todo la anterior desde una

[11]http://www.devjoker.com/contenidos/Conceptos-generales-NET/88/Introducción-a-NET.aspx

interfaz visual basada en ventanas. Ambas herramientas pueden descargarse gratuitamente desde **http://www.msdn.microsoft.com/net**, aunque la última sólo está disponible para subscriptores MSDN Universal (*los no subscriptores pueden pedirlo desde dicha dirección y se les enviará gratis por correo ordinario*)

El concepto de Microsoft.NET también incluye al conjunto de nuevas aplicaciones que Microsoft y terceros han (*o están*) desarrollando para ser utilizadas en la plataforma .NET. Entre ellas podemos destacar aplicaciones desarrolladas por Microsoft tales como Windows.NET, Hailstorm, Visual Studio.NET, MSN.NET, Office.NET, y los nuevos servidores para empresas de Microsoft (*SQL Server.NET, Exchange.NET, etc.*)

b) .NET Framework.

.NET FrameWork es el corazón de la tecnología .NET. Es el marco de trabajo y ejecución común a toda la tecnología .NET. Es por lo tanto un elemento indispensable dentro de la tecnología .NET. Es un componente de software que se instala en el sistema operativo[12]. Podemos descargar el .NET Framework desde el sitio web de Microsoft:

- **.NET Framework 1.1**: http://www.microsoft.com/downloads/details.aspx?displaylang=es&FamilyID=262d25e3-f589-4842-8157-034d1e7cf3a3

- **.NET Framework 2.0**: http://www.microsoft.com/downloads/details.aspx?familyid=0856EACB-4362-4B0D-8EDD-AAB15C5E04F5&displaylang=es

Los componentes del .NET Framework proveen los “ladrillos” necesarios para la construcción de aplicaciones dentro de Visual Studio .NET. A continuación se tiene una visión general del .NET Framework:

- **Lenguajes .NET.** Destacan C#, VB.NET, J#, Delphi (*Object Pascal*), C++, Perl, Python, Fortran, Cobol y PowerBuilder.

[12] http://www.devjoker.com/contenidos/Conceptos-generales-NET/88/-NET-FrameWork.aspx

- **CLR, Common Language Runtime.** Es el motor de ejecución común a todos los lenguajes .NET. Provee lo que se llama código administrado, es decir, un entorno que provee servicios automáticos al código que se ejecuta.[13]
- **MSIL, Microsoft Intermedial language.** Es el lenguaje intermedio al que compilan las aplicaciones (*Asemmblies*) .NET. Transforma código intermedio de alto nivel independiente del hardware que lo ejecuta a código de máquina propio del dispositivo que lo ejecuta. Este lenguaje intermedio es interpretado por el CLR en tiempo de ejecución.
- **CLS, Common Language Specification.** Que engloban las pautas que deben cumplir los lenguajes .NET. Es esta característica la que va a permitir a otras compañías producir lenguajes compatibles con .NET.
- **ADO.NET.** Es la nueva interfaz de bases de datos. No se trata de una evolución de ADO, sino que se trata de una interfaz completamente nueva.
- **ASP.NET.** Es la nueva tecnología para páginas web dinámicas completamente integrada dentro del entorno .NET. Representa una auténtica revolución en el desarrollo Web (*Internet e Intranet*).
- **Biblioteca de clases .NET.** Es el conjunto de clases que componen el .NET framework.

c) Arquitectura básica de la plataforma .NET.

Descripción del Framework y sus principales componentes: Lenguajes, biblioteca de clases y CLR. La nueva tecnología de Microsoft ofrece soluciones a los problemas de programación actuales, como son la administración de código o la programación para Internet. [14]Para aprovechar al máximo las características de .NET es necesario entender la arquitectura básica en la que esta implementada esta tecnología y así beneficiarse de todas las características que ofrece esta nueva plataforma. El Framework de .NET es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican enormemente el desarrollo de aplicaciones.

Mediante esta herramienta se ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables.

---

[13]**GALLEGOS., M.,** Desarrollador 5 estrellas – Microsoft Visual Studio .NET., Quito-Ecuador., Microsoft Corporation., 2006., Pp.16-20

[14]http://www.desarrolloweb.com/articulos/1328.php.

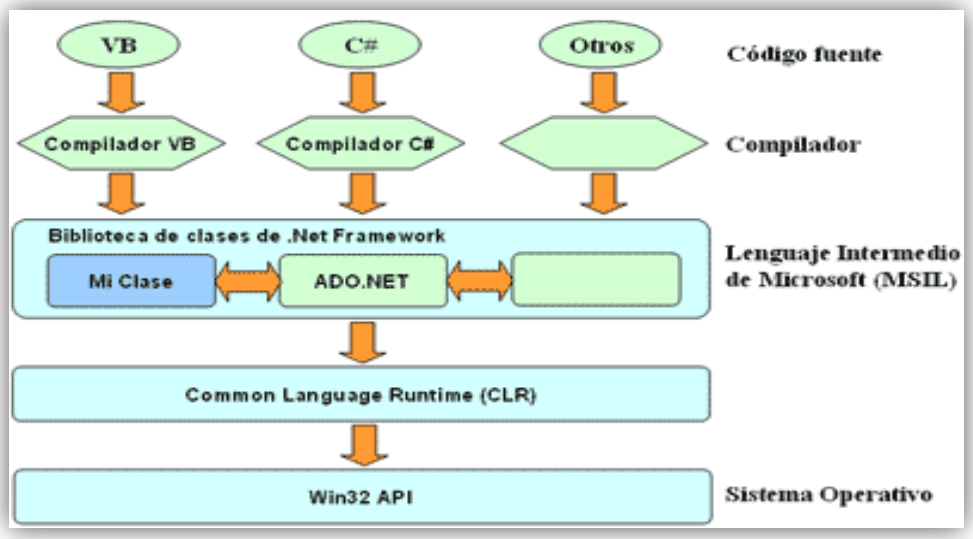

Figura 3. Arquitectura de .NET Framework.

Fuente: Página oficial de .Net Framework – http://msdn.microsoft.com/netframework/

Los principales componentes de este entorno son: Lenguajes de compilación, Biblioteca de clases de .NET, CLR (*Common Language Runtime*).

La **biblioteca de clases** de .NET Framework incluye, entre otros, tres componentes clave:

- ASP.NET para construir aplicaciones y servicios Web.
- Windows Forms para desarrollar interfaces de usuario.
- ADO.NET para conectar las aplicaciones a bases de datos.

La forma de organizar la biblioteca de clases de .NET dentro del código es a través de los espacios de nombres (*namespaces*), donde cada clase está organizada en espacios de nombres según su funcionalidad. Por ejemplo, para manejar ficheros se utiliza el espacio de nombres System.IO y si lo que se quiere es obtener información de una fuente de datos se utilizará el espacio de nombres System.Data. La principal ventaja de los espacios de nombres de .NET es que de esta forma se tiene toda la bliblioteca de clases de .NET centralizada bajo el mismo espacio de nombres (*System*). Además, desde cualquier lenguaje se usa la misma sintaxis de invocación, ya que a todos los lenguajes se aplica la misma biblioteca de clases.

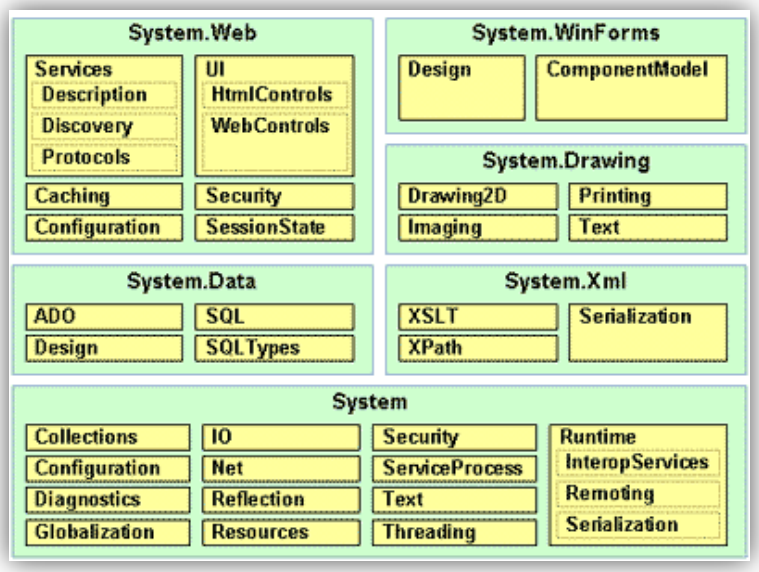

Figura 4. Biblioteca de clases de .NET Framework.
Fuente: Página oficial de .Net Framework – http://msdn.microsoft.com/netframework/

El **CLR** es el verdadero núcleo del Framework de .NET, ya que es el entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios que ofrece el sistema operativo estándar Win32.
Las principales características y servicios que ofrece el CLR son:[15]

- **Modelo de programación consistente:** A todos los servicios y facilidades ofrecidos por el CLR se accede a través de un modelo de programación orientado a objetos.
- **Modelo de programación sencillo:** Con el CLR desaparecen muchos elementos complejos incluidos en los sistemas operativos actuales (*registro de Windows, GUIDs, HRESULTS, IUnknown, etc.*)
- **Eliminación del "infierno de las DLLs":** En la plataforma .NET desaparece el problema conocido como "infierno de las DLLs" que se da en los sistemas operativos actuales de la familia Windows. En la plataforma .NET las versiones nuevas de las DLLs pueden coexistir con las viejas, de modo que las aplicaciones diseñadas para ejecutarse usando las viejas podrán seguir

[15]http://www.devjoker.com/contenidos/Conceptos-generales-NET/89/Common-Language-Runtime-CLR.aspx

usándolas tras instalación de las nuevas. Esto, obviamente, simplifica mucho la instalación y desinstalación de software.

- **Ejecución multiplataforma:** El CLR actúa como una máquina virtual, encargándose de ejecutar las aplicaciones diseñadas para la plataforma .NET. Es decir, cualquier plataforma para la que exista una versión del CLR podrá ejecutar cualquier aplicación .NET. Asimismo, dado que la arquitectura del CLR está totalmente abierta, es posible que en el futuro se diseñen versiones del mismo para otros sistemas operativos.
- **Integración de lenguajes:** La integración de lenguajes es tal que es posible escribir una clase en C# que herede de otra escrita en Visual Basic.NET que, a su vez, herede de otra escrita en C++ con extensiones gestionadas.
- **Gestión de memoria:** El CLR incluye un **recolector de basura** que evita que el programador tenga que tener en cuenta cuándo ha de destruir los objetos que dejen de serle útiles. Gracias a este recolector se evitan errores de programación muy comunes como intentos de borrado de objetos ya borrados, agotamiento de memoria por olvido de eliminación de objetos inútiles o solicitud de acceso a miembros de objetos ya destruidos.
- **Seguridad de tipos:** El CLR facilita la detección de errores de programación difíciles de localizar comprobando que toda conversión de tipos que se realice durante la ejecución de una aplicación .NET se haga de modo que los tipos origen y destino sean compatibles.
- **Aislamiento de procesos:** El CLR asegura que desde código perteneciente a un determinado proceso, no se pueda acceder a código o datos pertenecientes a otro, lo que evita errores de programación muy frecuentes e impide que unos procesos puedan atacar a otros.
- **Tratamiento de excepciones:** En el CLR todos los errores que se puedan producir durante la ejecución de una aplicación se propagan de igual manera: mediante excepciones. Esto es muy diferente a como se venía haciendo en los sistemas Windows hasta la aparición de la plataforma .NET, donde ciertos errores se transmitían mediante códigos de error en formato Win32, otros mediante HRESULTs y otros mediante excepciones.

El CLR[16] da a las aplicaciones la sensación de que se están ejecutando sobre una máquina virtual, y precisamente MSIL (*Microsoft Intermediate Lenguage*) es el

[16]http://www.scribd.com/doc/27358983/Microsoft-Intermediate-Language-Assemblies

código máquina de esa máquina virtual. Es decir, MSIL es el único código que es capaz de interpretar el CLR, y por tanto cuando se dice que un compilador genera código para la plataforma .NET lo que se está diciendo es que genera MSIL.

La principal ventaja del MSIL es que facilita la ejecución multiplataforma y la integración entre lenguajes al ser independiente de la CPU y proporcionar un formato común para el código máquina generado por todos los compiladores que generen código para .NET.

Sin embargo, dado que las CPUs no pueden ejecutar directamente MSIL, antes de ejecutarlo habrá que convertirlo al código nativo de la CPU sobre la que se vaya a ejecutar. De esto se encarga un componente del CLR conocido como compilador JIT (*Just-In-Time*) o jitter que va convirtiendo dinámicamente el código MSIL a ejecutar en código nativo según sea necesario.

d) Ventajas de .NET.[17]

A continuación se resumen las ventajas más importantes que proporciona .NET Framework:

- **Código administrado**: El CLR realiza un control automático del código para que este sea seguro, es decir, controla los recursos del sistema para que la aplicación se ejecute correctamente.
- **Interoperabilidad multilenguaje**: El código puede ser escrito en cualquier lenguaje compatible con .NET ya que siempre se compila en código intermedio (*MSIL*).
- **Compilación just-in-time**: El compilador JIT incluido en el Framework compila el código intermedio (*MSIL*) generando el código máquina propio de la plataforma. Se aumenta así el rendimiento de la aplicación al ser específico para cada plataforma.
- **Garbage collector**: El CLR proporciona un sistema automático de administración de memoria denominado recolector de basura (*garbage collector*). El CLR detecta cuándo el programa deja de utilizar la memoria y la libera

[17] http://www.desarrolloweb.com/articulos/1329.php

automáticamente. De esta forma el programador no tiene por qué liberar la memoria de forma explícita aunque también sea posible hacerlo manualmente (*mediante el método disponse) liberamos el objeto para que el recolector de basura lo elimine de memoria*).

- **Seguridad de acceso al código**: Se puede especificar que una pieza de código tenga permisos de lectura de archivos pero no de escritura. Es posible aplicar distintos niveles de seguridad al código, de forma que se puede ejecutar código procedente del Web sin tener que preocuparse si esto va a estropear el sistema.
- **Despliegue**: Por medio de los ensamblados resulta mucho más fácil el desarrollo de aplicaciones distribuidas y el mantenimiento de las mismas. El Framework realiza esta tarea de forma automática mejorando el rendimiento y asegurando el funcionamiento correcto de todas las aplicaciones.

e) Como funciona .NET.

Cuando se crea una aplicación Windows en algún lenguaje compatible con la plataforma .NET, puede utilizar cualquiera de los servicios que la biblioteca de clases de .NET provee. Por ejemplo: puede usar clases para hacer ventanas que tengan distintos tipos de controles.

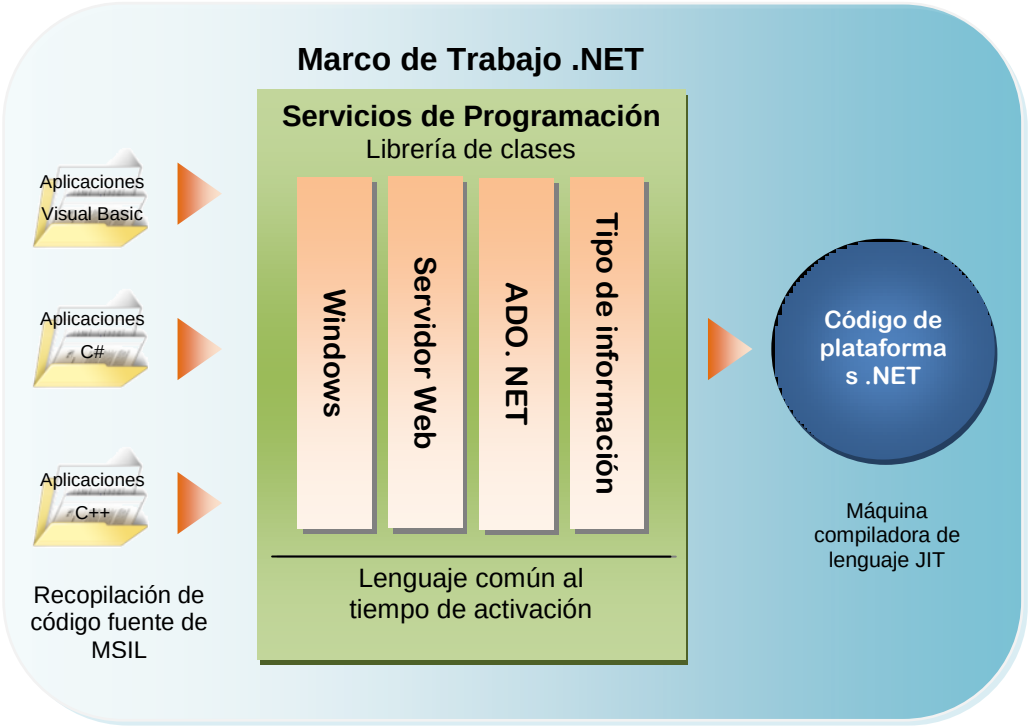

Figura 5. Cómo funciona el marco de trabajo .NET.
Fuente: Folleto "Desarrollador 5 estrellas – Visual Studio .NET"

Cuando compila la aplicación, se crea un código intermedio llamado MSIL. Este código es independiente de la plataforma de hardware. Una vez compilado, el ejecutor de lenguaje común (*CLR*) administra la ejecución de la aplicación.[18]

Uno de los subsistemas del CLR se llama compilación JIT, que transforma el código intermedio MSIL al código de máquina en el sistema donde la aplicación se va a ejecutar. Esta compilación a lenguaje de máquina lo hace en el momento de ejecución del código.

f) El futuro de .NET.[19]

A largo plazo Microsoft pretende remplazar el API Win32 o Windows API con la plataforma .NET. Esto debido a que el API Win32 o Windows API fue desarrollada sobre la marcha, careciendo de documentación detallada, uniformidad y cohesión entre sus distintos componentes, provocando múltiples problemas en el desarrollo de aplicaciones para el sistema operativo Windows. La plataforma .NET pretende solventar la mayoría de estos problemas proveyendo un conjunto único y expandible con facilidad, de bloques interconectados, diseñados de forma uniforme y bien documentados, que permitan a los desarrolladores tener a mano todo lo que necesitan para producir aplicaciones sólidas.

Debido a las ventajas que la disponibilidad de una plataforma de este tipo puede darle a las empresas de tecnología y al público en general, muchas otras empresas e instituciones se han unido a Microsoft en el desarrollo y fortalecimiento de la plataforma .NET, ya sea por medio de la implementación de la plataforma para otros sistemas operativos aparte de Windows (*Proyecto Mono de Ximian/Novell para Linux/MacOS X/BSD/Solaris*), el desarrollo de lenguajes de programación adicionales para la plataforma (*Lexico para hispanoparlantes, ANSI C de la Universidad de Princeton, NetCOBOL de Fujitsu, Delphi de Borland, PowerBuilder de Sybase entre otros*) o la creación de bloques adicionales para la plataforma (*como controles, componentes y bibliotecas de clases adicionales*); siendo algunas de ellas software libre, distribuibles bajo la licencia GPL.

---

[18] **GALLEGOS., M.,** Desarrollador 5 estrellas – Microsoft Visual Studio .NET., Quito-Ecuador., Microsoft Corporation., 2006., Pp.17

[19] http://www.solodisenio.com/net-de-microsoft/

Con esta plataforma Microsoft incursiona de lleno en el campo de los Servicios Web y establece el XML como norma en el transporte de información en sus productos y lo promociona como tal en los sistemas desarrollados utilizando sus herramientas.

2.1.3. SQL Server.

Hoy en día, las organizaciones enfrentan numerosos desafíos de datos, tales como la necesidad de tomar decisiones más rápidas y más orientadas a datos, la necesidad de aumentar la productividad y flexibilidad del personal de desarrollo y presionan para reducir los presupuestos generales de informática (*IT*) a la vez que escalan la infraestructura para satisfacer las exigencias cada vez mayores.[20]

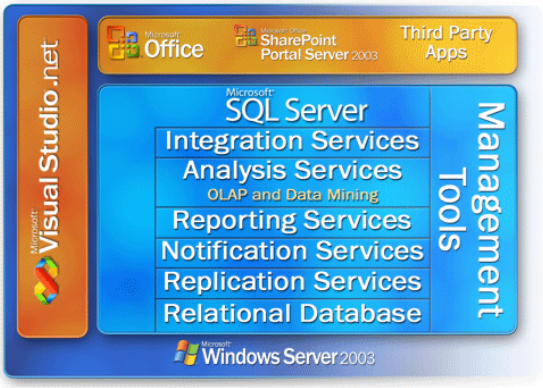

Figura 6. Diseño de la plataforma de datos SQL Server 2005.
Fuente: Microsoft SQL Server

SQL Server es una herramienta que está diseñado para ayudar a las empresas a enfrentar estos desafíos. Esta solución de administración y análisis de datos de próxima generación ofrece seguridad, escalabilidad y disponibilidad mayores a las aplicaciones de datos empresariales y analíticas, a la vez que las hace más fáciles de crear, desplegar y administrar.

Con la ampliación de las ventajas de SQL de sus versiones anteriores, actualmente SQL Server ofrece una solución integrada de administración y análisis de datos que ayuda a las organizaciones de cualquier magnitud a realizar lo siguiente:

- Crear, desplegar y administrar aplicaciones empresariales más seguras, escalables y confiables.

[20] http://www.microsoft.com/spain/sql/productinfo/overview/default.mspx

- Maximizar la productividad de IT mediante la reducción de la complejidad y el soporte de aplicaciones de bases de datos.
- Compartir datos en múltiples plataformas, aplicaciones y dispositivos para facilitar la conexión de sistemas internos y externos.
- Controlar los costes sin sacrificar el rendimiento, la disponibilidad, la escalabilidad o la seguridad.

Hoy en día, SQL Server potencia su infraestructura de datos en tres áreas clave:

1. administración de datos empresariales,
2. productividad del encargado del desarrollo e
3. inteligencia empresarial (*BI*).

También abre nuevos caminos en precios y licencias accesibles, rutas de actualización a SQL Server y el sistema Microsoft Windows Server.

SQL Server es un potente motor de bases de datos de alto rendimiento capaz de soportar millones de registros por tabla con un interface intuitivo y con herramientas de desarrollo integradas como Visual Studio 6.0 o .NET, además incorpora un modelo de objetos totalmente programable (*SQL-DMO*) con el que podemos desarrollar cualquier aplicación que manipule componentes de SQL Server, es decir, hacer aplicación para crear bases de datos, tablas, DTS, backups, etc., todo lo que se puede hacer desde el administrador del SQL Server y podemos hacerlo no solo en Visual C++ sino también en Visual Basic, ASP y por supuesto en .NET.[21]

- **Transact-SQL**[22]es el lenguaje que utiliza **SQL Server** para poder enviar peticiones tanto de consultas, inserciones, modificaciones, y de borrado a las tablas, así como otras peticiones que el usuario necesite sobre los datos. En definitiva, es un lenguaje que utiliza SQL Server para poder gestionar los datos que contienen las tablas. El lenguaje estándar **SQL** (*Structured Query Language*) se emplea para los sistemas de bases de datos relacionales **RDBMS** (*Relational Database Management System),* es el estándar ANSI (*American National Standards Institute*). También es utilizado por otros sistemas como: Oracle, Access, Sybase, etc.

[21]http://www.atp.com.ar/imppost.asp?ID=1797
[22]**FLORES., A.,** Introducción a SQL Server., Madrid- España., Forma Select Grupo Empresarial., 2006., Pp. 1-20

- **Net-Library**: Es el componente que controla las conexiones de diferentes protocolos y redes. Habilita **SQL Server** para escuchar a múltiples protocolos al mismo tiempo. Se puede configurar el servidor fácilmente para escuchar múltiples protocolos, empleando utilidades de red del servidor bajo **SQL Server**.

  *Nota: Cuando intentamos conectar a **SQL Server** y nos devuelve un error de comunicación, lo primero que hay que comprobar es el componente **Net-Library**.*

- **Open Data Services (*ODS*):** Es el componente que está escuchando para nuevas conexiones y respuestas. ODS controla las conexiones a SQL Server. ODS también controla las desconexiones inesperadas y deja libres los recursos del sistema.

- **Tabular Data Stream (*TDS*):** Es un protocolo privado que SQL Server emplea para cifrar los datos y comunicarse con las estaciones clientes.

- **Motores de SQL Server:** Existen dos motores muy importantes en SQL Server:
  - El motor relacional: Incluye los componentes necesarios para la consulta de datos.
  - El motor de almacenaje: Gestiona el almacenaje físico de los datos y la actualización de los datos en disco.

a) Plataforma de datos de SQL Server.[23]

La plataforma de datos SQL Server incluye las siguientes herramientas:

- **Base de datos relacional.** Un motor de base de datos relacional más segura, confiable, escalable y altamente disponible con mejor rendimiento y compatible para datos estructurados y sin estructura (*XML*).
- **Servicios de réplica.** Réplica de datos para aplicaciones de procesamiento de datos distribuidos o móviles, alta disponibilidad de los sistemas, concurrencia escalable con almacenes de datos secundarios para soluciones de información empresarial e integración con sistemas heterogéneos, incluidas las bases de datos Oracle existentes.

[23] **EXO., S.A.,** Microsoft SQL Server 2005., Buenos Aires- Argentina., EXO S.A., 2010., Pp. 1-25

- **Notification Services.** Capacidades avanzadas de notificación para el desarrollo y el despliegue de aplicaciones escalables que pueden entregar actualizaciones de información personalizadas y oportunas a una diversidad de dispositivos conectados y móviles.
- **Integration Services.** Capacidades de extracción, transformación y carga (*ELT*) de datos para almacenamiento e integración de datos en toda la empresa.
- **Analysis Services.** Capacidades de procesamiento analítico en línea (*OLAP*) para el análisis rápido y sofisticado de conjuntos de datos grandes y complejos, utilizando almacenamiento multidimensional.
- **Reporting Services.** Una solución global para crear, administrar y proporcionar tanto informes tradicionales orientados al papel como informes interactivos basados en la Web.
- **Herramientas de administración.** SQL Server incluye herramientas integradas de administración para administración y optimización avanzadas de bases de datos, así como también integración directa con herramientas tales como Microsoft Operations Manager (*MOM*) y Microsoft Systems Management Server (*SMS*). Los protocolos de acceso de datos estándar reducen drásticamente el tiempo que demanda integrar los datos en SQL Server con los sistemas existentes. Asimismo, el soporte del servicio Web nativo está incorporado en SQL Server para garantizar la interoperabilidad con otras aplicaciones y plataformas.
- **Herramientas de desarrollo.** SQL Server ofrece herramientas integradas de desarrollo para el motor de base de datos, extracción, transformación y carga de datos, minería de datos, OLAP e informes que están directamente integrados con Microsoft Visual Studio para ofrecer capacidades de desarrollo de aplicación de extremo a extremo. Cada subsistema principal en SQL Server se entrega con su propio modelo de objeto y conjunto de interfaces del programa de aplicación (*API*) para ampliar el sistema de datos en cualquier dirección que sea específica de su negocio. La plataforma de datos SQL Server ofrece los siguientes beneficios a las organizaciones de todas las magnitudes:
- **Aprovechamiento de activos de datos.** Además de brindar una base de datos segura y confiable para aplicaciones analíticas y del rubro, SQL Server permite que los clientes obtengan más valor de sus datos al incluir una funcionalidad incorporada tal como informe, análisis y minería de datos. Puede aprovechar

esta potencia y flexibilidad para entregar datos a cada rincón de su organización a una fracción del coste de algunos otros sistemas.

- **Aumento de la productividad.** A través de las capacidades globales de BI y la integración con herramientas conocidas como Microsoft Office System, SQL Server brinda a los trabajadores de la información en toda su organización información empresarial crítica y oportuna adaptada a sus necesidades específicas. El objetivo es ampliar la BI a todos los usuarios en una organización y, en última instancia, ayudar a los usuarios en todos los niveles de la organización a tomar mejores decisiones empresariales según uno de sus activos más valiosos: sus datos.
- **Reducción de la complejidad de IT.** SQL Server simplifica el desarrollo, el despliegue y la administración de aplicaciones de unidad de negocios y analíticas al ofrecer un entorno de desarrollo flexible para los encargados del desarrollo y herramientas integradas y automatizadas de administración para los administradores de bases de datos.
- **Menor coste total de propiedad (*TCO*).** El enfoque y la atención integrados sobre la facilidad de uso y despliegue en SQL Server, ofrece los costes directos, de implementación y mantenimiento más bajos de la industria para obtener un rápido rendimiento de su inversión en la base de datos.

SQL Server brinda la tecnología y las capacidades con las que puede contar su organización, con avances significativos en las áreas clave de administración de datos empresariales, productividad del encargado del desarrollo y BI.

b) Funcionamiento de SQL Server.

Luego de conocer algunos fundamentos sobre SQL Server, vamos a ver gráficamente su funcionamiento:[24]

Paso 1: Un cliente realiza una consulta a la base de datos.

Figura 7. Funcionamiento de SQL Server - Paso 1
Fuente: Forma Select – Grupo Empresarial

[24]**FLORES., A.,** Introducción a SQL Server., Madrid- España., Forma Select Grupo Empresarial., 2006., Pp. 1-20

Paso 2: El analizador de consultas recibe la respuesta del ODS.

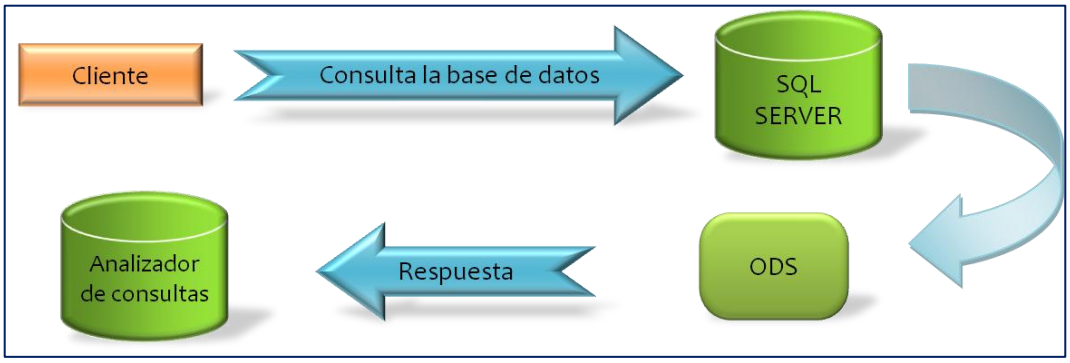

Figura 8. Funcionamiento de SQL Server - Paso 2
Fuente: Forma Select – Grupo Empresarial

Paso 3: El analizador de consultas comprueba la sintaxis.

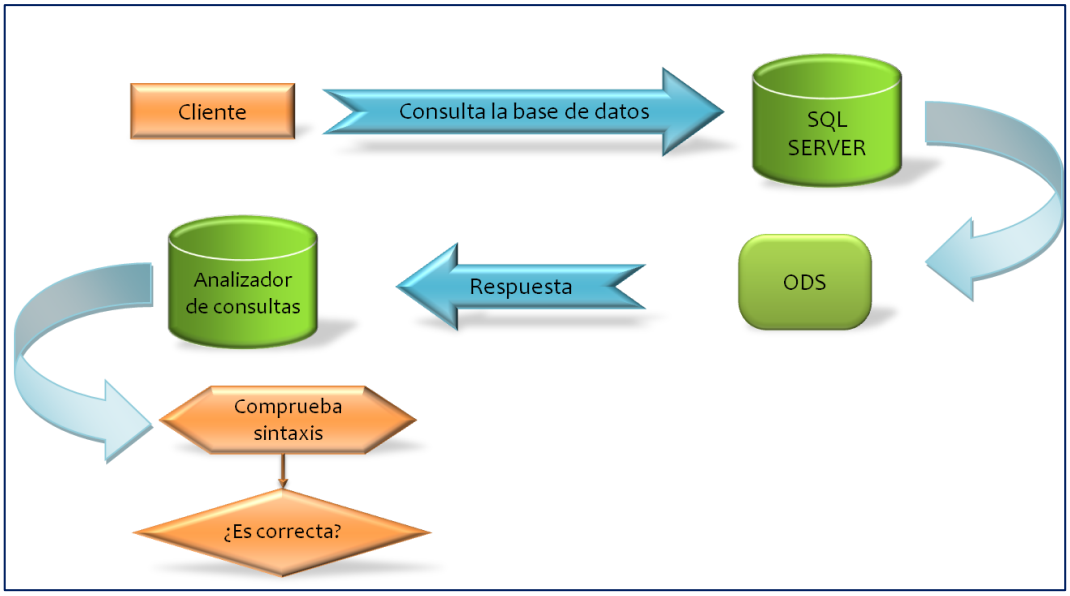

Figura 9. Funcionamiento de SQL Server - Paso 3
Fuente: Forma Select – Grupo Empresarial

Paso 4: Si la sintaxis es incorrecta, el analizador de consultas devuelve un error.

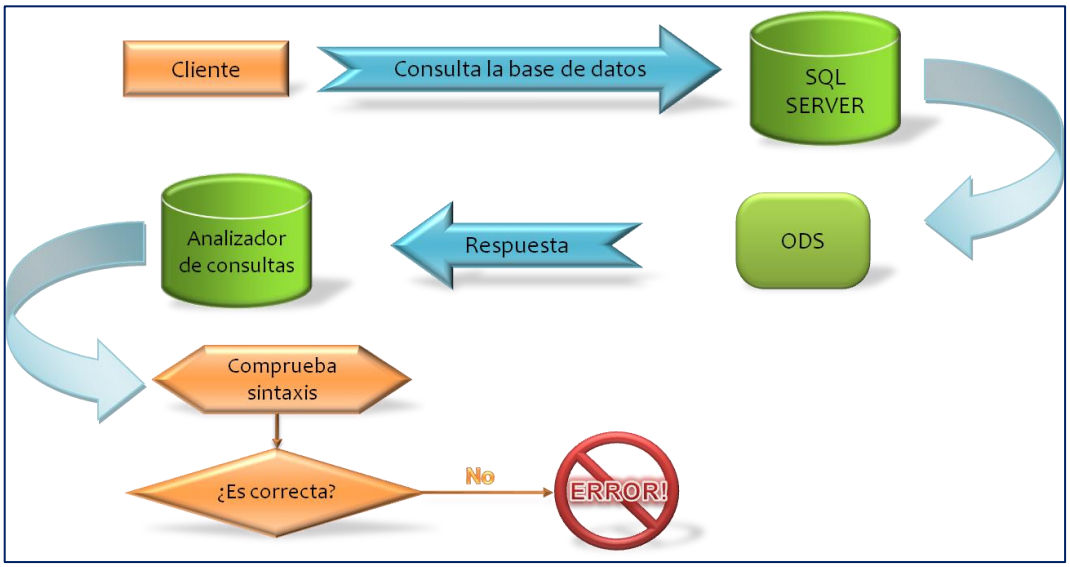

Figura 10. Funcionamiento de SQL Server - Paso 4
Fuente: Forma Select – Grupo Empresarial

Paso 5: Si la sintaxis es correcta, la respuesta se pasa al optimizador de consultas.

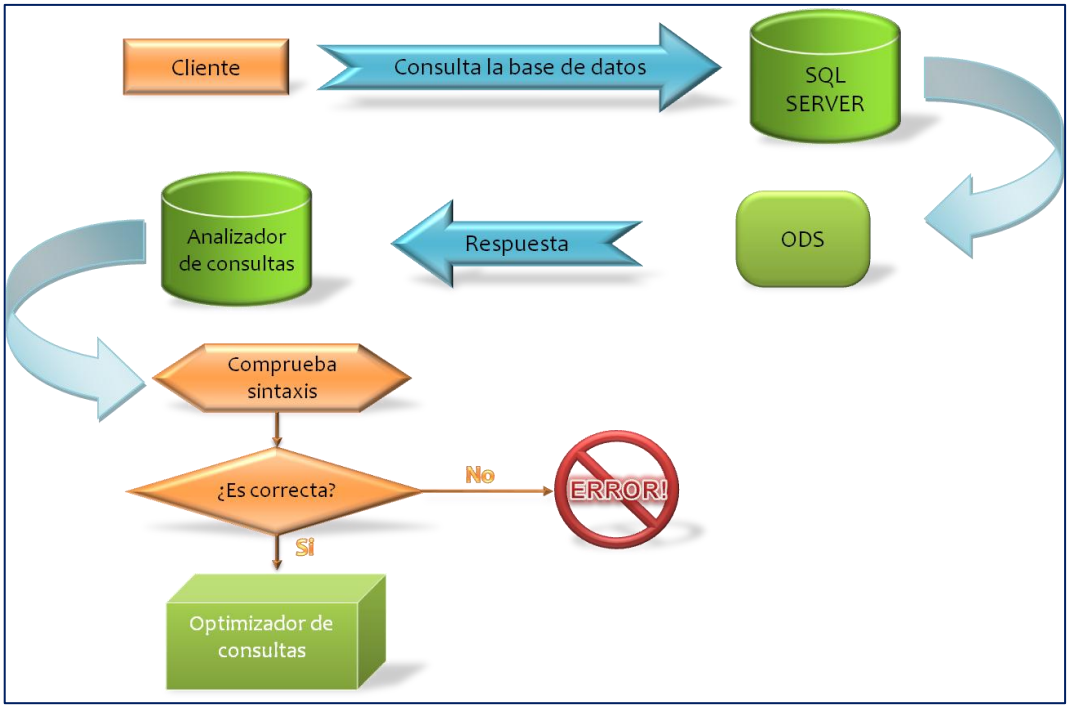

Figura 11. Funcionamiento de SQL Server - Paso 5
Fuente: Forma Select – Grupo Empresarial

Paso 6: Se devuelve la respuesta al cliente.

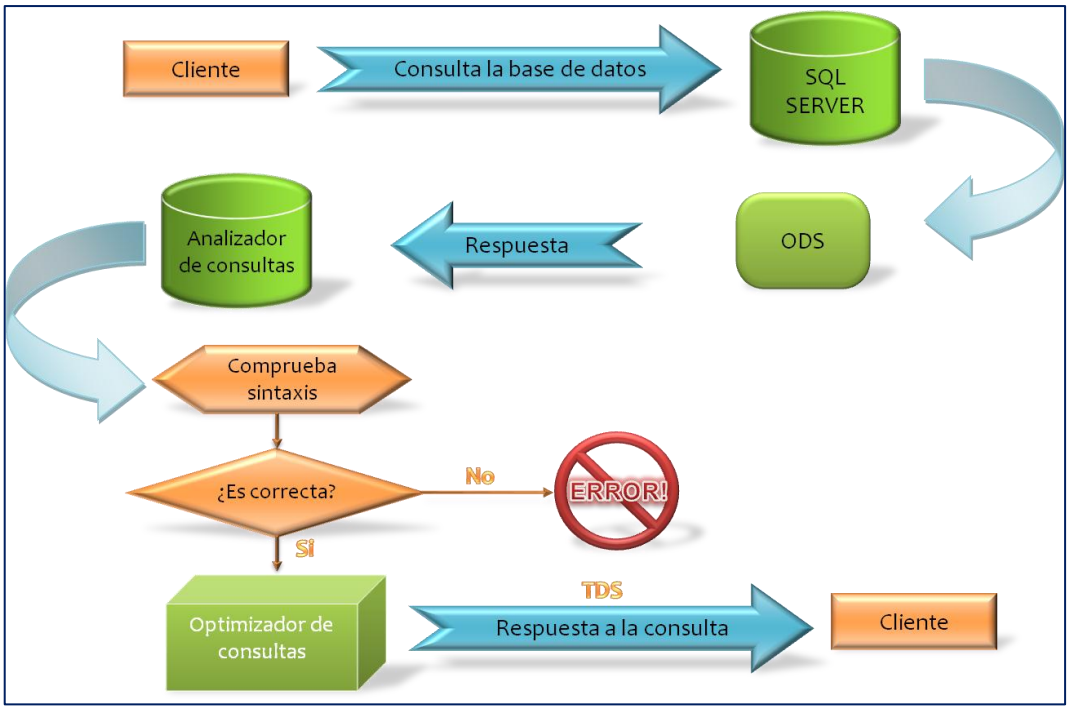

Figura 12. Funcionamiento de SQL Server - Paso 6
Fuente: Forma Select – Grupo Empresarial

c) Utilidades y servicios de SQL Server.[25]

SQL Server puede ser administrado empleando diferentes utilidades:

- SQL-DMO.
- Aplicaciones COM.
- Herramientas de administración gráfica que incluye SQL Server.
- OSQL.
- BCP.

- **SQL-DMO** se emplea para realizar tareas de administración con SQL Server. Todas las herramientas que incluye SQL Server están escritas empleando la colección de objetos SQLDMO. Se puede utilizar Visual Basic como lenguaje que emplee esta colección de objetos. Si generamos un ejecutable del

[25] **FLORES., A.,** Introducción a SQL Server., Madrid- España., Forma Select Grupo Empresarial., 2006., Pp. 5-7

programa, tendremos la ventaja que el código estará oculto, y no se detallarán las sentencias empleadas.

- Las **aplicaciones COM** se pueden emplear para poder acceder a las herramientas de administración de SQL Server, por ejemplo con el lenguaje Visual Basic y con la colección de objetos SQL-DMO.
- **OSQL** (*Object Structured Query Language*) es una utilidad que permite el uso de sentencias **Transact-SQL**, así como de procedimientos del sistema, y también el manejo de ficheros script. Esta utilidad utiliza **ODBC** para conectarse con el servidor. **OSQL** se ejecuta directamente desde el sistema operativo, una vez ejecutado permite sentencias **Transact**-**SQL**, e interactúa directamente con SQL Server.
- **BCP** es una utilidad que permite copiar datos de SQL Server a un fichero de datos definido por el usuario.

SQL Server incluye cuatro servicios:

- MSSQLServer.
- SQLServerAgent.
- Microsoft Distributed Transaction Coordinator (***MSDTC***).
- Microsoft Search.

Figura 13. Administrador de servicios de SQL Server
Fuente: Forma Select – Grupo Empresarial

- **MSSQLServer**se encarga del procesamiento de transacciones y consultas, así como del control de la base de datos y la integridad de los datos.

- **SQL Server Agent** se encarga de la gestión de operadores, alertas y trabajos de la base de datos.
- **MSDTC** se encarga del control de transacciones distribuidas.
- **Microsoft Search** se encarga de la administración de índices y catálogos, para la búsqueda indexada de texto.

d) Ventajas de SQL Server.

Entre las ventajas más significativas que se puede destacar de Microsoft SQL Server son las siguientes:[26]

- **Crear aplicaciones flexibles y fiables basadas en datos.**
  Ahora más que nunca los fabricantes están aprovechando las bases de datos relacionales para proporcionar una experiencia rica al usuario final. La protección y la administración de la información en estas aplicaciones son aspectos fundamentales. SQL Server Express™ ayuda a los programadores a crear aplicaciones sólidas y fiables al ofrecer un sistema de base de datos robusto, gratuito y fácil de usar. Con demasiada frecuencia, los sistemas de bases de datos son excesivamente complejos para crear aplicaciones sencillas. Visual Studio® 2005 y SQL Server Express reducen esta complejidad proporcionando un entorno sencillo pero eficaz para crear aplicaciones basadas en datos. Los programadores pueden diseñar esquemas, agregar datos y realizar consultas en bases de datos locales, todo ello desde el entorno de Visual Studio 2005.Si necesitan características de bases de datos más avanzadas, SQL Server Express puede actualizarse sin problemas a versiones más sofisticadas de SQL Server.

- **Habilitar sitios Web dinámicos**
  En el mundo actual, donde los cambios son vertiginosos, incluso los sitios Web más básicos necesitan ser interactivos. SQL Server Express y Visual Web Developer constituyen una plataforma integrada y sencilla para la creación de sitios Web dinámicos y fiables. Estos dos productos combinados proporcionan las herramientas necesarias para crear, implementar y administrar sitios Web.

[26] http://msdn.microsoft.com/sql/express

SQL Server Express incluye en la capa de base de datos varias características que lo hacen atractivo para su uso en entornos Web. Aprovecha el avanzado motor de optimización de consultas de SQL Server 2005 para ofrecer un gran rendimiento. Además, las bases de datos se pueden cargar fácilmente en un entorno de hospedaje a través de la característica XCopy. La compatibilidad de la base de datos con XML nativo garantiza que su entorno Web puede interactuar fácilmente con otras aplicaciones mediante servicios Web.

- **Sencillez de implementación y servicio**

  Implementar y realizar el mantenimiento de software supone un gran gasto para los fabricantes de aplicaciones de software. Una base de datos embebida debe permitir reducir este coste al ofrecerla posibilidad de realizar tareas de ajuste y administración de forma automática, aplicar y efectuar automáticamente revisiones y tareas de servicio, y permite usar el soporte de configuración e instalación embebido. SQL Server Express está diseñado especialmente para cumplir estos objetivos.

  Usando el motor principal de base de datos de SQL Server 2005, SQL Server Express puede configurarse para ajustar dinámicamente los parámetros de la base de datos para adecuarlo a las diferentes características de utilización. Además, los fabricantes de software no tienen que preocuparse por implementar directamente las actualizaciones de la base de datos, ya que SQL Server Express utiliza el sistema Microsoft Update para proporcionar directamente las actualizaciones a los usuarios finales. Para los usuarios empresariales, SQL Server Express se conecta fácilmente con la mayoría de los entornos de administración empresarial con compatibilidad tanto con Active Directory® como con Windows®Management Instrumentation.

  SQL Server Express también proporciona compatibilidad con programas de instalación y configuración sin intervención del usuario o basados en interfaz gráfica. Esto aporta al fabricante de software el máximo nivel de flexibilidad y control sobre la instalación y la configuración de la base de datos embebida.

- **Más rapidez de creación de informes**
  Crear aplicaciones de elaboración de informes (*reporting*) suele ser un proceso pesado. SQL Server Express simplifica este proceso al integrarse directamente con los controles de SQL Server 2005Reporting Services incluidos en Visual Studio 2005. Usar estos controles con SQL Server Express permite a los fabricantes crear con facilidad sofisticados informes que incluyen tablas, diagramas y gráficos.

2.2. Arquitectura del Software.

Desde hace varios años, lo que el mercado ha demandado con respecto al tema de software, es en su mayoría aplicaciones consistentes y que cumplan las expectativas de cada una de las empresas que la utilizan. Partiendo de estos requerimientos, además de la tecnología empleada para dar solución a los distintos problemas, se han ido creando y perfeccionando distintas arquitecturas más o menos independientes de la tecnología aplicada.[27]

La evolución tecnológica que el sector ha sufrido durante los últimos años ha permitido otra evolución paralela, la de la Arquitectura del Software. A medida que aparecían nuevos recursos técnicos, los patrones de diseño se amoldaban para aprovechar las nuevas característica que estás novedades ofrecían. De esta forma, el modelo arquitectónico de las aplicaciones ha sufrido grandes saltos en su uso, implementación y funcionamiento.

Las técnicas metodológicas desarrolladas con el fin de facilitar la programación se engloban dentro de la Arquitectura de Software o Arquitectura lógica. La AS se refiere a un grupo de abstracciones y patrones que nos brindan un esquema de referencia útil para guiarnos en el desarrollo de software dentro de un sistema informático.

Así, los programadores, diseñadores, ingenieros y analistas pueden trabajar bajo una línea común que les posibilite la compatibilidad necesaria para lograr el objetivo deseado.

---

[27] **FERNÁNDEZ., D.,** Definición de una arquitectura software para el diseño de aplicaciones web basadas en tecnología Java-J2EE., Universidad de Oviedo., TESIS., Oviedo-España., 2003., Pp. 3-10

Algunos objetivos trascendentales dentro de un esquema de Arquitectura de Software pueden ser:

- Mantenibilidad, esto es, fácilmente analizable, modificable, corregible;
- El nivel de interacción con otros sistemas informáticos, o su escalabilidad.

Estas arquitecturas están definidas muchas veces por el tipo de tecnología a la cual se enfrenta un programador o grupo de programadores, por lo cual algunos tipos de arquitectura son más recomendables que otras para ciertas tecnologías.

Cada tarea de computación es asignada a una computadora, por lo cual una arquitectura determinada debe ser implementada físicamente y definir de forma abstracta los componentes que tomarán parte en las tareas y sus interfaces comunicativas.
Todo esto se desarrolla a "alto nivel", ensamblando elementos para lograr la mayor funcionalidad posible siendo a la vez portable, logrando disponibilidad, escalabilidad y confiabilidad.

2.2.1. Breve reseña histórica.

Cada vez que se narra la historia de la arquitectura de software (*o de la ingeniería de software, según el caso*), se reconoce que en un principio, hacia 1968, Edsger Dijkstra, de la Universidad Tecnológica de Eindhoven en Holanda y Premio Turing 1972, propuso que se establezca una estructuración correcta de los sistemas de software antes de lanzarse a programar, escribiendo código de cualquier manera[28].

Dijkstra, quien sostenía que las ciencias de la computación eran una rama aplicada de las matemáticas y sugería seguir pasos formales para descomponer problemas mayores, fue uno de los introductores de la noción de sistemas operativos organizados en capas que se comunican sólo con las capas adyacentes y que se superponen “como capas de cebolla”. Inventó o ayudó a precisar además docenas de conceptos: el algoritmo del camino más corto, los stacks, los vectores, los semáforos, los abrazos

[28] **DIJKSTRA., E.,** The Structure of the Multiprogramming system., 2a.ed., Texas-E.U.A., Springer., 2001., 49-52

mortales. De sus ensayos arranca la tradición de hacer referencia a “niveles de abstracción” que ha sido tan común en la arquitectura subsiguiente.

Aunque Dijkstra no utiliza el término arquitectura para describir el diseño conceptual del software, sus conceptos sientan las bases para lo que luego expresarían Niklaus Wirth[29] como stepwise refinement y DeRemer y Kron[30] como programming-in-the large (*o programación en grande*), ideas que poco a poco irían decantando entre los ingenieros primero y los arquitectos después.

1969 - Fred Brooks Jr y Ken Iverson llamaban arquitectura a la estructura conceptual de un sistema en la perspectiva del programador.

Década de 1970 - Una novedad importante. Fue el advenimiento del diseño estructurado y delos primeros modelos explícitos de desarrollo de software.

1971 - C. R. Spooner tituló uno de sus ensayos “Una arquitectura de software para los 70s”, sin que la mayor parte de la historiografía de la AS registrara ese antecedente.

1972 - Parnas publicó un ensayo en el que discutía la forma en que la modularidad en el diseño de sistemas podía mejorar la flexibilidad y el control conceptual del sistema, acortando los tiempos de desarrollo.

1975 - Brooks, diseñador del sistema operativo OS/360 y Premio Turing 2000, utilizaba el concepto de arquitectura del sistema para designar “la especificación completa y detallada de la interfaz de usuario” y consideraba que el arquitecto es un agente del usuario.

Década de 1980 - Los métodos de desarrollo estructurado demostraron no escalar suficientemente y fueron dejando el lugar a un nuevo paradigma, el de la programación orientada a objetos.

---

[29] **WIRTH., N.,** Program development by stepwise refinement., New York-E.U.A., ACM., 2003., Pp. 221-227

[30] **DE REMER., F., Y OTROS.,** Programming in the large versus programming in the small., 2a. ed., Paris-Francia., Ed. Jacky Estublier., 1995., Pp.80-86

Fines de la década de 1980 y comienzos de la siguiente - La expresión arquitectura de software comienza a aparecer en la literatura para hacer referencia a la configuración morfológica de una aplicación.

El primer estudio en que aparece la expresión “arquitectura de software” en el sentido en que hoy lo conocemos es sin duda el de Perry y Wolf; ocurrió tan tarde como en 1992, aunque el trabajo se fue gestando desde 1989.

1992 - Aparece en el mercado herramientas de ingeniería asistida por computadoras, CASE.

Década de 1990 - Se cree es la década de la arquitectura de software. Se usa el término “arquitectura” en contraste con “diseño”, para evocar nociones de codificación, de abstracción de estándares, de entrenamiento formal (*de los arquitectos de software*) y de estilo.

Un segundo gran tema de la época fue el surgimiento de los patrones, cristalizada en dos textos fundamentales, el de la Banda de los Cuatro en 1995[31] y la serie POSA desde 1996[32]. El primero de ellos promueve una expansión de la programación orientada a objetos, mientras que el segundo desenvuelve un marco ligeramente más ligado a la AS.

2000 - Uno de los acontecimientos arquitectónicos más importantes del año fue la hoy célebre tesis de Roy Fielding que presentó el modelo REST, el cual establece definitivamente el tema de las tecnologías de Internet y los modelos orientados a servicios y recursos en el centro de las preocupaciones de la disciplina[33].

Siglo XXI - La AS aparece dominada por estrategias orientadas a líneas de productos y por establecer modalidades de análisis, diseño, verificación, refinamiento, recuperación, diseño basado en escenarios, estudios de casos y hasta justificación económica, redefiniendo todas las metodologías ligadas al ciclo de vida en términos

[31]**GAMMA., E., Y OTROS.,** Design Patterns: Elements of reusable object oriented software., Massachusetts-E.U.A., Addison Wesley Publishing Company Inc.., 1995., Pp. 22-29

[32]**BUSCHMANN., F., Y OTROS.,** Pattern-oriented software architecture., Munich-Germany., Sigs Datacom., 2008., Pp. 20-33

[33]**THOMAS., F.,** Architectural styles and the design of network based software architectures., Universidad de California., TESIS DOCTORAL., Irvine-California., 2000., Pp. 69-83

arquitectónicos. Todo lo que se ha hecho en ingeniería debe formularse de nuevo, integrando la AS en el conjunto.

La producción de estas nuevas metodologías ha sido masiva, y una vez más tiene como epicentro el trabajo del Software Engineering Institute en Carnegie Mellon. La aparición de las metodologías basadas en arquitectura, junto con la popularización de los métodos ágiles en general y Extreme Programming en particular, han causado un reordenamiento del campo de los métodos, hasta entonces dominados por las estrategias de diseño “de peso pesado”. Después de la AS y de las tácticas radicales, las metodologías nunca volverán a ser las mismas.

La semblanza que se ha trazado no es más que una visión selectiva de las etapas recorridas por la AS. Los lineamientos de ese proceso podrían dibujarse de maneras distintas, ya sea enfatizando los hallazgos formales, las intuiciones dominantes de cada período o las diferencias que median entre la abstracción cualitativa de la arquitectura y las cuantificaciones que han sido la norma en ingeniería de software.

2.2.2. Definiciones.

Existen muchas definiciones de Arquitectura del Software y no parece que ninguna de ellas haya sido totalmente aceptada. En un sentido amplio se puede resumir y estar de acuerdo en que la Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación.

Según Kruchten-Philippe, la arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad.

Una definición reconocida es la de Clements. Según Clements, la AS es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del

sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones. [34]

Pero existe una definición oficial de Arquitectura del Software es la IEEE Std 1471-2000 que dice lo siguiente: "*La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución*".

Ante el gran número y variedad de definiciones, Mary Shaw y David Garlan[35] proporcionaron una sistematización iluminadora, explicando las diferencias entre definiciones en función de distintas clases de modelos. Destilando las definiciones y los puntos de vista implícitos o explícitos, los autores clasifican los modelos de esta forma:[36]

- **Modelos estructurales:** sostienen que la AS está compuesta por componentes, conexiones entre ellos y (*usualmente*) otros aspectos tales como configuración, estilo, restricciones, semántica, análisis, propiedades, racionalizaciones, requerimientos, necesidades de los participantes. El trabajo en esta área está caracterizada por el desarrollo de lenguajes de descripción arquitectónica (*ADLs*).
- **Modelos de framework:** son similares a la vista estructural, pero su énfasis primario radica en la (*usualmente una sola*) estructura coherente del sistema completo, en vez de concentrarse en su composición. Los modelos de framework a menudo se refieren a dominios o clases de problemas específicos. El trabajo que ejemplifica esta variante incluye arquitecturas de software específicas de dominios, como CORBA, o modelos basados en CORBA, o repositorios de componentes específicos, como PRISM.
- **Modelos dinámicos:** enfatizan la cualidad conductual de los sistemas. "Dinámico" puede referirse a los cambios en la configuración del sistema, o a la dinámica

[34]**CLEMENTS., P.,** A Survey of Architecture Description Languages – Proceedings of the International Workshop on Software Specification and Design., Berlin-Alemania., Ed. Springer., 1996., Pp. 97-144
[35]**SHAW., M.,** Software Architecture: Perspectives on an emerging discipline., Universidad Carnegie Mellon., PUBLICACIÓN., Pittsburgh-Pennsylvania., Ed. Prentice Hall, 1996., Pp. 115-132
[36]**BILLY., R.,** Introducción a la arquitectura de Software., Universidad de Buenos Aires., PUBLICACIÓN., Buenos Aires-Argentina., 2007., Pp. 11-25

involucrada en el progreso de la computación, tales como valores cambiantes de datos.

- **Modelos de proceso:** se concentran en la construcción de la arquitectura, y en los pasos o procesos involucrados en esa construcción. En esta perspectiva, la arquitectura es el resultado de seguir un argumento (*script*) de proceso. Esta vista se ejemplifica con el actual trabajo sobre programación de procesos para derivar arquitecturas.
- **Modelos funcionales:** una minoría considera la arquitectura como un conjunto de componentes funcionales, organizados en capas que proporcionan servicios hacia arriba. Es tal vez útil pensar en esta visión como un framework particular.

Ninguna de estas vistas excluye a las otras, ni representa un conflicto fundamental sobre lo que es o debe ser la AS. Por el contrario, representan un espectro en la comunidad de investigación sobre distintos énfasis que pueden aplicarse a la arquitectura: sobre sus partes constituyentes, su totalidad, la forma en que se comporta una vez construida, o el proceso de su construcción.

2.2.3. Utilidad de la Arquitectura de Software.

Cuando se planea un sistema, la mayoría de programadores y expertos tienen presente que hay que definir la "Arquitectura", pero ¿realmente sabemos el por qué o nos dejamos llevar por la tendencia del momento en el mercado (*3-capas, n-capas, SOA, etc*.)?[37]

Como ya se mencionó anteriormente, una arquitectura de software es un modelo abstracto reusable que define la estructura para construir un sistema. Su definición envuelve decisiones de diseño que se traducen en atributos de calidad como:

- Rendimiento,
- Seguridad,
- Costo de hacer un cambio,
- Confiabilidad y
- Usabilidad

[37] http://armanhal.blogspot.com/2004/05/para-qu-sirve-una-arquitectura-de.html

La arquitectura representa la clave para comprender, organizar y comunicar un sistema, además, permite implantar conceptos como el reuso y facilita la evolución de la solución.

Debido a la complejidad de los sistemas, la forma de documentar una arquitectura es mediante “Vistas”, es decir, mediante descripciones simplificadas del sistema desde una perspectiva particular. El modelo más conocido de vistas es el que define RUP (4+1) vistas que comprende:

- Vista del Diseño
- Vista del Proceso
- Vista de la Implementación
- Vista del Deployment y
- La vista común que es: Vista de Casos de Uso.

Durante el desarrollo de una Arquitectura de Software es recomendable considerar los siguientes factores:

- Casos de Uso
- Experiencia del Arquitecto
- Uso de Patrones
- Middleware/Frameworks existentes
- Sistemas (*Legacy y otros*)
- Estándares y Políticas
- Requerimientos No-Funcionales
- Distribución

Por último hay que tener presente que "una arquitectura correcta lleva al éxito a un sistema. Una arquitectura incorrecta generalmente indica una receta segura al desastre".No existe arquitectura que sea buena si no es en función de las necesidades de sus interesados (*stakeholders*).

2.2.4. Rol del Arquitecto de Software.

El arquitecto de software es visto hoy en día como un integrante fundamental de los equipos de desarrollo de software empresarial.

A diferencia de un programador, el arquitecto de software debe dominar la mayor cantidad de tecnologías de software y prácticas de diseño, para así poder tomar decisiones adecuadas para garantizar el mejor desempeño, reuso, robustez, portabilidad, flexibilidad, escalabilidad y mantenibilidad de las aplicaciones. Estas decisiones sobre la estructura y dinámica de la aplicación son plasmadas en una notación formal estandarizada como lo es UML; sobre todo si se utilizan las nuevas tecnologías, en especial con los lenguajes orientados a objetos.[38]

El arquitecto de software es el líder técnico del equipo, el rol natural al que debe aspirar un programador experimentado que desea tomar decisiones técnicas relevantes en el desarrollo de un sistema. Es el principal tomador de decisiones respecto a la manera en que será construida la aplicación por los programadores del equipo. El líder de proyecto se apoya totalmente en este rol para alcanzar el éxito del proyecto optimizando el uso de la tecnología para desarrollar la solución correcta que proporcionará valor real a sus usuarios y al negocio al que le dará soporte.

Hay dos formas de convertirse en arquitecto: aprendiendo a definir las soluciones con base en la propia experiencia (*el camino largo*), o reutilizando el conocimiento de los expertos a nivel mundial plasmado en patrones de arquitectura y diseño (*el camino corto*).

En un contexto general, el arquitecto de software es quien obtiene información sobre el problema y diseña una solución que satisfaga los requisitos funcionales y los no funcionales manteniendo la flexibilidad cuando los requisitos cambien. Para el diseño de aplicaciones se apoya en patrones, modelos y frameworks. Participa activamente en todas las fases del desarrollo y establece lineamientos generales que deben ser tenidos en cuenta en el desarrollo de nuevos proyectos.

[38] http://armanhal.blogspot.com/2004/05/para-qu-sirve-una-arquitectura-de.html

Los arquitectos de sistemas son las personas responsables de:

- Servir de interfaz con los usuarios y patrocinadores, así como cualquier otro que está involucrado en determinar sus necesidades.
- Generar los niveles más altos de requisitos del sistema, basados en las necesidades del usuario, así como, algunas otras limitantes tales como costos y tiempos.
- Asegurarse de que este alto conjunto de requisitos sea consistente, completo, correcto, y operacional.
- Llevar a cabo análisis costo-beneficio para determinar que costos se cumplen mejor ya sea manualmente, por software o con hardware; maximizando así los componentes ya desarrollados o los comerciales.
- Desarrollar algoritmos de particionado (*y otros procesos*) para economizar los requisitos en particiones discretas de manera que se necesite un mínimo de comunicación entre las particiones y los usuarios con el sistema.
- Particionar grandes sistemas en (*capas sucesivas de*) subsistemas y componentes los cuales puedan ser manipulados por un solo ingeniero, equipo de ingenieros o arquitectos subordinados.
- Asegurar que se alcance el máximo de robustez de arquitectura.
- Generar una serie de requisitos de pruebas de aceptación, las cuales junto con los diseñadores, ingenieros de pruebas, y el usuario determinen si se han cumplido los requisitos, especialmente para las interfaces de usuario.
- Generar bosquejos, modelos, guías de usuario y prototipos que mantengan en acuerdo y al corriente a los ingenieros con los usuarios.

2.2.5. Patrones y Estilos.

Los **'Lenguajes de Patrones'** se pueden definir del siguiente modo: "La especificación de una serie de elementos y sus relaciones de modo que nos permiten describir buenas soluciones a los diferentes problemas que aparecen en un contexto específico".[39]

[39] **DÍAZ., D.,** Meta-Patrones, una nueva aproximación a los patrones de diseño., Universidad de Navarra., PUBLICACIÓN., Madrid España., 2002., Pp. 2-6

El objetivo de los patrones de diseño es el de capturar buenas prácticas que nos permitan mejorar la calidad del diseño de un sistema, determinando elementos que soporten roles útiles en dicho contexto, encapsulando complejidad, y haciéndolo más flexible. Por otro lado, con frecuencia se dice que la ***función define a la forma***, es decir, que la estructura o la arquitectura de cualquier sistema está muy relacionada con lo que dicho sistema tiene que hacer.

Esta es la razón por la que los sistemas con objetivos similares comparten también una arquitectura común, unos procesos bien definidos, y un conjunto de elementos similares (*patrones de diseño*). Similar funcionalidad y servicio, similar estructura.

Cuando desarrollamos un sistema que se encuadra dentro de cierto tipo, es muy útil consultar lenguajes de patrones que traten el dominio en el que estamos. Un lenguaje de patrones nos sirve como referencia conceptual del dominio del problema, ya que éstos parten como solución a un conjunto de casos de uso, e interacciones con actores específicos. Además constituyen también un marco conceptual en el diseño de la arquitectura de nuestros sistemas, ya que como la función define a la forma, sintetizan por lo general soluciones arquitectónicas y estructurales bien probadas y muy útiles dentro del tipo de problemas que modelan.

De alguna forma, los patrones nos permiten identificar y completar los casos de uso básicos expuestos por el cliente, comprender la arquitectura del sistema a construir así como su problemática, y buscar componentes ya desarrollados que cumplan con los requisitos del tipo de sistema a construir (*es decir nos permiten obtener de una forma sencilla la arquitectura base que buscamos durante la fase de diseño arquitectónico*).

Desafortunadamente los lenguajes de patrones tampoco son la panacea, y presentan muchas lagunas. Sobre todo, hay que recordar que todo este movimiento de documentación de diseño se origina a mediados de los noventa y que aun siendo mucho el trabajo realizado, no existe todavía ninguna estandarización sobre cómo abordar el desarrollo de estos lenguajes, ni ninguna clasificación que los relacione.

Un **'Estilo'**[40] es un concepto descriptivo que define una forma de articulación u organización arquitectónica. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software, mientras que las formas complejas se articulan mediante composición de los estilos fundamentales.

Sucintamente descriptos, los estilos conjugan elementos (*o "componentes", como se los llama aquí*), conectores, configuraciones y restricciones. Al estipular los conectores como elemento de juicio de primera clase, el concepto de estilo, incidentalmente, se sitúa en un orden de discurso y de método que el modelado orientado a objetos en general y UML en particular no cubren satisfactoriamente.

La descripción de un estilo se puede formular en lenguaje natural o en diagramas, pero lo mejor es hacerlo en un lenguaje de descripción arquitectónica o en lenguajes formales de especificación.

A diferencia de los patrones de diseños, que son centenares, los estilos se ordenan en seis o siete clases fundamentales y unos veinte ejemplares, como máximo. Es digno de señalarse el empeño por subsumir todas las formas existentes de aplicaciones en un conjunto de dimensiones tan modestas.

Las arquitecturas complejas o compuestas resultan del agregado o la composición de estilos más básicos. Algunos estilos típicos son las arquitecturas basadas en flujo de datos, las centradas en datos, las de invocación implícita, las derivadas, las de código móvil, heterogéneas o las peer-to-peer.

Billy Reynoso, integrante del equipo Microsoft, en su exposición "Introducción a la arquitectura", define los siguientes estilos arquitectónicos:

- Estilo de flujo de datos:
  - Tuberías y filtros.

- Estilos Centrados en Datos:
  - Arquitecturas de Pizarra o Repositorio.

[40] **BILLY., R.,** Introducción a la arquitectura de Software., Universidad de Buenos Aires., PUBLICACIÓN., Buenos Aires-Argentina., 2007., Pp. 11-25

- Estilos de Llamada y Retorno:
  - Model-View-Controller (MVC)
  - Arquitecturas en Capas.
  - Arquitecturas Orientadas a Objetos.
  - Arquitecturas Basadas en Componentes.

- Estilos derivados:
  - C2.
  - GenVoca.
  - REST.

- Estilos de Código Móvil:
  - Arquitecturas de Máquinas Virtuales.

- Estilos heterogéneos:
  - Sistemas de control de procesos.
  - Arquitecturas Basadas en Atributos.

- Estilos Peer-to-Peer:
  - Arquitecturas Basadas en Eventos.
  - Arquitecturas Orientadas a Servicios. (*SOA*)
  - Arquitecturas Basadas en Recursos.

A continuación se realizará un breve análisis en cuanto a las características, funcionamiento, ventajas y desventajas de las arquitecturas de software más representativas.

a) Estilo de flujo de datos.[41]

Este tipo de estilo es ideal para realizar transformaciones de datos dividiéndolos en pasos sucesivos. En este tipo de arquitectura nos encontramos con dos tipos de componentes. El primer tipo de componente son los elementos que realizan las transformaciones de los datos cuando pasan por ellos, los datos pasan de forma

[41]**LÓPEZ., J., Y OTROS.,** Modelos avanzados de comunicación de recursos., Universidad de Oviedo., PUBLICACIÓN., Oviedo-España., UPM., 2008., Pp. 57-82

secuencial de uno a otro componente, realizando el procesamiento de los datos de forma secuencial.

Estos elementos son independientes unos de otros. Todos tienen una o varias puertas de entrada-salida que sirven para mandar los datos modificados de un módulo a otro. El otro tipo de elemento es el que se encarga de enviar el flujo de los datos desde la salida de un módulo transformador a la entrada de otro, es decir, realiza el rol de comunicador. En este estilo arquitectónico se va a estudiar el patrón tubería/filtro (*piping/filtering*)

Tubería (*Pipe*): Solamente es una forma de conectar la salida estándar de un programa con la entrada estándar de otro programa automáticamente. Es un concepto muy importante, ya que permite que varios programas puedan encadenarse y hacer que los resultados de un comando puedan tratarse como datos a procesar del siguiente.

Filtro: Es un programa que recibe unos datos por la entrada y devuelve, tras ejecutar una condición, parte de ellos por su salida. Por parte de ellos se entiende tanto su totalidad, tanto nada, como tanto una porción de ellos.

Esta arquitectura es muy simple pero a su vez es muy potente y robusta. Consiste en una serie de componentes, filtros, que están conectados a través de tuberías. Normalmente la estructura es una secuencia Filtro/Tubería/Filtro/Tubería/etc pero puede presentarse en estructuras más complejas. Cada filtro trabaja de manera independiente de los componentes que se encuentran situados antes o después de ella. Se diseñan de tal modo que esperan un conjunto de datos en un determinado formato y obtiene como resultado otros datos de salida en un formato específico. Si el flujo degenera en una única línea de transformación, se denomina secuencia batch.

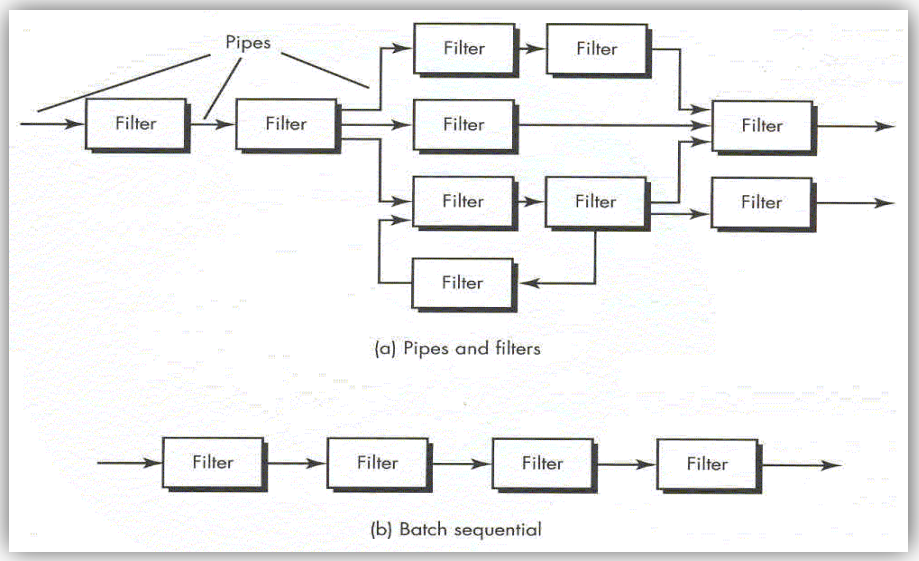

Figura 14. Arquitectura - Tubería y Filtros
Fuente: Ingeniería del Software I, curso 2001-2002 (EPSA UCLM)

Antes de explicar su funcionamiento, se debe detallar cuando es conveniente utilizarla. La popularidad de la arquitectura se debe principalmente al sistema operativo tipo Unix. Y debe de utilizarse siempre y cuando haya una gran cantidad de transformaciones a realizar, es decir un gran número de filtros.

El funcionamiento es el siguiente: todos los filtros son procesos que corren al mismo tiempo, es decir forman diferentes hilos. Cada tubería que llega al filtro tiene un papel diferente, así que en la tubería debe estar expresado este concepto. Los filtros deberán de leer los datos de entrada, aplicarle la condición y devolverlos por la salida. Si no hay datos en la entrada, el filtro no hace nada, espera.

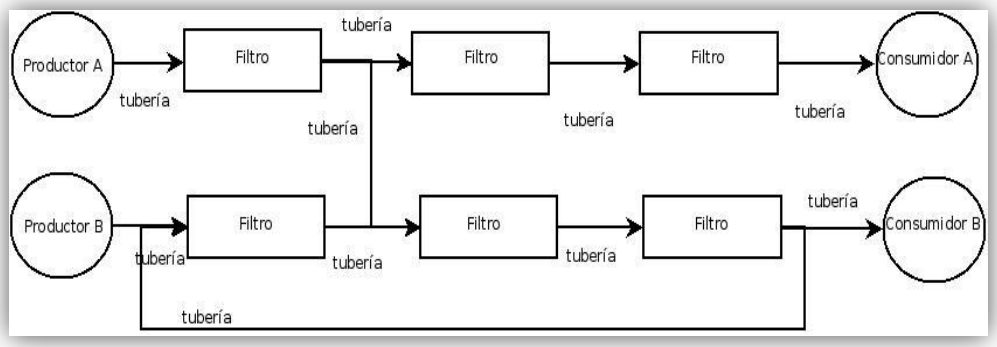

Figura 15. Arquitectura - Tubería y Filtros compleja
Fuente: Modelos avanzados de comunicación de recursos. Versión 1.0. Yaco, 2007.

Esta estructura puede llegar a ser todavía más compleja y llegar a ser incluso recursiva.

Las ventajas más destacables de esta arquitectura son las siguientes:

- Conseguimos ir separando conceptos, lo que nos permite ir entendiendo el sistema poco a poco, lo que sin duda mucho más sencillo.
- La reutilización es muy alta en estos sistemas. Ya que el sistema está dividido.
- De igual manera, el mantenimiento es verdaderamente sencillo. Ya que encontrar los errores es una tarea muy ligera.
- Soporta ejecución concurrente.

Los inconvenientes principales son:

- El tiempo de espera. Si un filtro tiene tres tuberías de entradas, tiene que esperar hasta que las tres tuberías no le aporten los datos para poder ejecutarse totalmente.
- Es normal que las tuberías solo suelen aceptar un tipo de datos, lo que las restringe enormemente.

b) Estilo centrado en datos.[42]

Arquitecturas de Pizarra o Repositorio.

En la arquitectura de pizarra, podemos encontrar dos componentes principales. Una estructura de datos que representa el estado actual, y un número independiente de componentes, los cuales realizan sus operaciones sobre él. Esta arquitectura se puede subdividir en dos:

- Si las transacciones en su flujo de entrada, definen los procesos que van a ejecutarse, entonces el repositorio puede ser algo como una BD tradicional.
- Sin embargo, si el estado de dicha estructura es la que va a disparar los procesos que se ejecuten, el repositorio se llamará pizarra pura.

[42] **LÓPEZ., J., Y OTROS.,** Modelos avanzados de comunicación de recursos., Universidad de Oviedo., PUBLICACIÓN., Oviedo-España., UPM., 2008., Pp. 57-82

Esta arquitectura, se ha usado y se usa, en aplicaciones que necesitan interpretación de proceso de señales bastante complejo. También es usado en implementaciones de estilos de procesos en lotes de BD. Y por último en programación organizada como colecciones de servicios alrededor de un repositorio común.

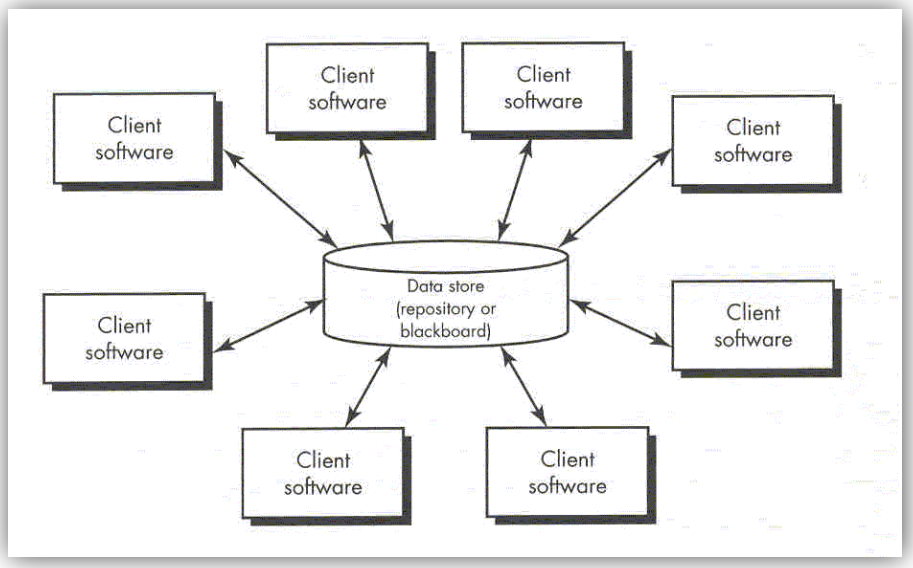

Figura 16. Arquitectura – Pizarra
Fuente: Modelos avanzados de comunicación de recursos. Versión 1.0. Yaco, 2007.

En general, se tiende a pensar que no hay sistemas organizados como repositorios, con la arquitectura de pizarra. Pero en realidad hay muchos más sistemas de lo que se cree. Esta arquitectura es muy conocida en sistemas de Inteligencia artificial (*IA*), como hemos dicho no es una curiosidad histórica, y todavía se usa en gran medida en IA distribuida y cooperativa, robótica, multi-agentes, programación evolutiva, gramáticas complejas, etc.

Algunos compiladores están arquitecturados con el estilo tubería-filtro, pero se podrían representar mejor como estilo de pizarra. Ya que un gran número de compiladores contemporáneos operan con información compartida: tablas de símbolos, arboles abstractos sintácticos, etc. Del mismo modo que acabamos de decir que los estilos de tubería-filtro evolucionan hacia pizarras. Los estilos de pizarra suelen evolucionar hacia el de máquinas virtuales.

Las principales ventajas son las siguientes:

- Hace posible la interacción de agentes contra el sistema.
- Funciona muy bien con los problemas no deterministas (*especial para IA*)
- Sabemos el conocimiento que llevamos en cada momento del proceso.

Pero hay que resaltar los inconvenientes que se encuentran:

- Problemas de seguridad ya que la pizarra es accesible por todos.
- Problemas de sincronización al chequear/vigilar la pizarra.

c) Estilo de llamada y retorno.[43]

Este estilo es interesante en los sistemas que se centran en la interacción de un usuario con el propio sistema. Podemos dividir la arquitectura en dos partes, la primera representa la interfaz del usuario con el que este realiza la llamada al sistema, la segunda contiene la lógica de negocio que se realiza tras la correspondiente llamada del usuario. Esta familia de estilos enfatiza la modificabilidad, la escalabilidad, la reusabilidad y la usabilidad del sistema. Al diseñar una arquitectura de este tipo es importante saber que datos son los que servirán para interactuar con el usuario y cuales servirán solo como lógica de aplicación. En esta familia de arquitecturas se destacan los patrones modelo-vista-controlador y arquitectura en tres capas.

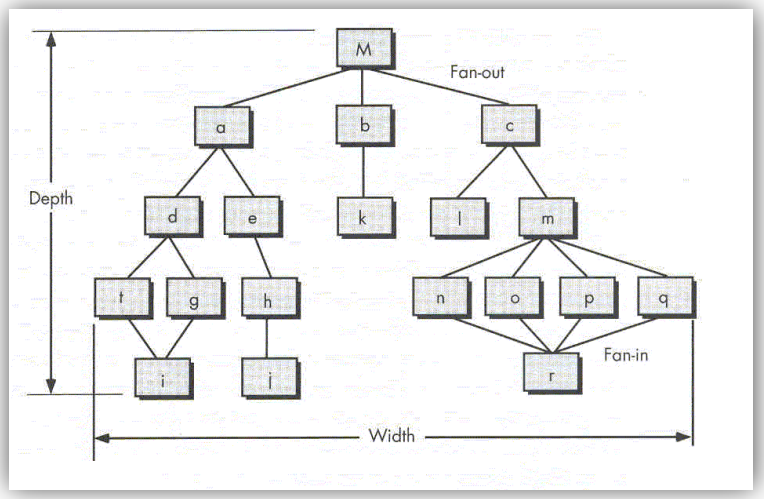

Figura 17. Arquitectura - Llamada y retorno
Fuente: Modelos avanzados de comunicación de recursos. Versión 1.0. Yaco, 2007.

[43] **LÓPEZ., J., Y OTROS.,** Modelos avanzados de comunicación de recursos., Universidad de Oviedo., PUBLICACIÓN., Oviedo-España., UPM., 2008., Pp. 57-82

d) Arquitectura en 3 capas.

Cuando se desarrolla una aplicación cliente, la tendencia habitual es mezclar lógica con presentación. Es normal (*sino tenemos una técnica programando*) que en los formularios implementemos el acceso a la lógica de negocio de nuestra aplicación y la navegación a nuevos formularios. Pero esto trae consigo algunos inconvenientes que veremos en el epígrafe de inconvenientes. Para solucionarlos se utiliza la arquitectura en tres capas.

Una capa[44] es una separación lógica del software, una separación básica de preocupaciones en el nivel del desarrollador, de modo que se pueden dividir más fácilmente las responsabilidades con respecto al sistema. El patrón de capas establece que el uso de capas ayuda a estructurar aplicaciones que pueden descomponerse en grupos de subtareas, cada uno de los cuales se encuentra en un nivel de abstracción determinado. Dicho de otro modo, es una clásica separación de asuntos: dividir las diversas tareas de un sistema empresarial (*la recuperación de datos, el almacenamiento de datos, la ejecución de reglas empresariales con esos datos, la visualización de datos, la recopilación de entradas, etc*.) en componentes o subsecciones, de modo que podamos realizar con mayor facilidad un seguimiento de lo que ocurre, dónde y cuándo. Naturalmente, la división del trabajo más frecuente corresponde a los niveles de presentación, lógica y acceso a datos.

- **Capa de presentación**[45]: esta es la que el usuario puede ver en su ordenador, es donde se tratan los datos que se van a mostrar. Se intenta que en esta capa haya el mínimo de procesamiento. Esta capa se comunicará solamente con la capa de negocio.

- **Capa de negocios**: en esta capa esta la lógica, se recibe las peticiones del usuario, y tras ejecutar una acción se le envía las respuestas del proceso. Esta capa se comunica como se ha dicho con la de presentación, la cual le envía peticiones y esta le responde con los resultados. Y también se comunica con la capa de datos, para pedirle la información requerida.

[44] http://devsoftx.wordpress.com/2008/04/24/arquitectura-en-n-capas/
[45] **LÓPEZ., J., Y OTROS.,** Modelos avanzados de comunicación de recursos., Universidad de Oviedo., PUBLICACIÓN., Oviedo-España., UPM., 2008., Pp. 57-82

- **Capa de datos**: es donde se accede a los datos. Se hace referencia a uno o más gestores de BD que realizan el almacenamiento, modificación y consulta de los datos. Recibe peticiones desde la capa de negocios.

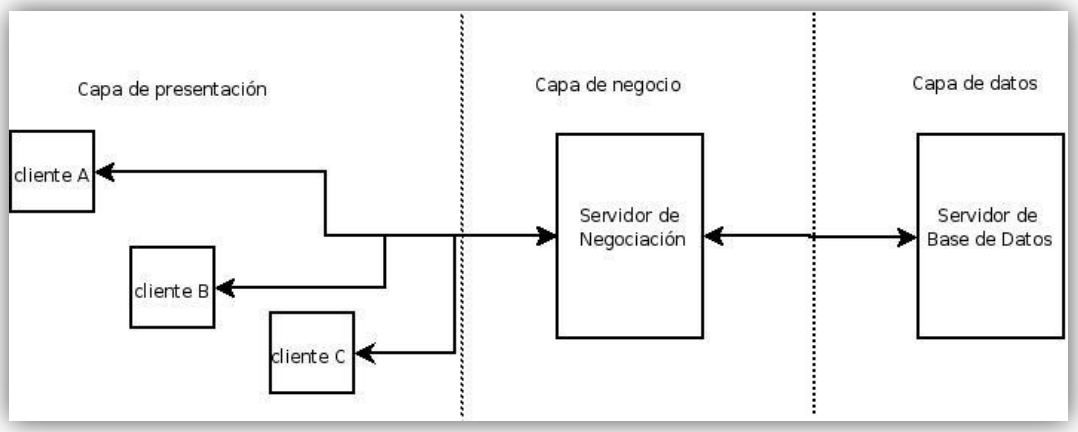

Figura 18. Arquitectura - 3 capas
Fuente: Modelos avanzados de comunicación de recursos. Versión 1.0. Yaco, 2007.

Es destacable hablar del concepto de nivel. Estas capas pueden estar en uno o varios ordenadores. Si todas se encuentran en el mismo ordenador, se dice que es arquitectura en tres capas y un nivel. Si por el contrario se encuentran en dos, se dice que es arquitectura en tres capas y dos niveles. Y si se encuentran en tres, pues tres capas y tres niveles.

Es importante indicar, que en este caso son muchas más las ventajas que los inconvenientes, pero como en anteriores patrones comentaremos los puntos a favor como en contra, pero habiendo resaltado antes que son más los beneficios que los perjuicios. Las ventajas son las siguientes:

- Facilita que se pueda descomponer la aplicación en varios niveles de abstracción.
- Facilita la evolución del sistema, ya que los cambios solo deben de afectar a la capa donde se encuentre la modificación.
- Si la interfaz accede a la misma función (*p.ej. varios lugares para identificarse*), no se repetirá código. Lo que conlleva la ventaja de una mayor facilidad de mantenimiento de la aplicación, entre otros.

- Si se añade un nuevo formulario no ocasionará verdaderos quebraderos de cabeza, ya que los formularios ya no dependen unos de otros, con lo que el cambiar el workflow es algo trivial.
- Los formularios ya no acceden de forma independiente a los datos, ya que no se accede a través de ellos, al modificar los datos. Esto implica que no se nos mostrará diferencia alguna.

Y como inconvenientes se encuentran:

- No todo sistema podrá ser estructura en capas.
- Y aun pudiendo ser estructurado en capas, la separación entre una y otra no es trivial. Ya no solo porque para un desarrollador no lo es, sino también porque muchos lenguajes y/o frameworks no están preparados para ello.

Tabla 2. Frameworks que implementan la arquitectura en 3 capas.
Fuente: Modelos avanzados de comunicación de recursos. Versión 1.0. Yaco, 2007.

LENGUAJE	LICENCIA	NOMBRE
Ruby	MIT	Ruby on Rails
Java / J2ee	Apache	JSP Struts
Java / J2ee	Apache	Spring Framework
Java / J2ee	Apache	Tapestry
Java / J2ee	Apache	Aurora
Java / J2ee	Apache	JavaServerFaces
Perl	GPL	Catalyst
PHP	BSD	Zend Framework
PHP	MIT	CakePHP
PHP	GNU/GPL	Kumbia
PHP	MIT	Symfony
PHP	MIT	QCodo
PHP	GNU/GPL	CodeIgniter
Python	ZPL	Zope3
Python	Varias	Turbogears
Python	BSD	Django
.NET	Castle Project	MonoRail
.NET	Apache	Spring .NET
.NET	Apache	Maverick .NET
.NET	Microsoft Patterns & Practices	User Interface Process (UIP) Application Block

e) Arquitectura Model-View-Controller (*MVC*).[46]

La arquitectura Modelo-Vista-Controlador, es la implementación de la arquitectura en tres capas más extendida. Todo lo que se diga en este epígrafe es extensible a la arquitectura en tres capas, con algunas pequeñas diferencias.

- **Modelo (*Model*)**: Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada. El modelo debe de preservar la integridad de los datos.

- **Vista (*View*)**: Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador. Interactúa con la interfaz de usuario.

- **Controlador (*Controller*)**: Reciben las entradas, usualmente como eventos, e interpreta las operaciones del usuario; codificando los movimientos, pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio (*"service requests"*) para el modelo o la vista. Es el que debe de controlar los eventos.

Después de definir los conceptos de Modelo, Vista y Controlador, se definirán las responsabilidades de cada uno de ellos.

El modelo deberá:
- Acceder a los datos persistentes, a la capa de almacenamiento de datos. Algo ideal es que este sea independiente de la Base de datos utilizada.
- Llevar las estadísticas de sistema (*en caso de haberlas*).
- Definir las reglas de negocio.
- Notificar los cambios que se hayan hecho.

El controlador deberá:
- Recibir los eventos de entrada.
- Realizar la acción correspondiente al evento que se ha disparado.

[46] **LÓPEZ., J., Y OTROS.,** Modelos avanzados de comunicación de recursos., Universidad de Oviedo., PUBLICACIÓN., Oviedo-España., UPM., 2008., Pp. 57-82

Las vistas deberán:

- Conseguir los datos que aporta el modelo y mostrárselos al usuario.
- Saber cuál es su controlador asociado, el cual puede pedirle algunos servicios típicos como el de actualizar.

Flujo de trabajo (*WorkFlow*)

- El usuario realiza una acción que generará un evento.
- El Controlador recibe el evento y lo traduce en una petición al Modelo (*aunque también puede llamar directamente a la vista*).
- El modelo (*si es necesario*) llama a la vista para su actualización.
- Para cumplir con la actualización la Vista puede solicitar datos al Modelo.
- El Controlador recibe el control.

Ya hemos ido comentando las ventajas de este patrón, pero de todas maneras las vamos a recopilar en este punto:

- Conseguimos múltiples vistas del modelo.
- Todas las vistas están sincronizadas.
- No acoplamiento, y facilidad de evolución, para cambiar las vistas y los controladores.
- La aplicación puede soportar un tipo de interfaz para cada usuario (*rol*).

Pero también posee algunos inconvenientes:

- La complejidad va creciendo más rápido que el tamaño de la aplicación.
- Problemas al conectar las distintas capas.
- Acceso ineficiente, pues es necesario varias llamadas al modelo para actualizar los datos.
- La vista y el controlador son específicos para una plataforma.

f) Arquitectura Orientada a Servicio (*SOA*)[47]

La Arquitectura Orientada a Servicios (*en inglés Service Oriented Architecture*), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio. Permite la creación de sistemas altamente escalables que reflejan el negocio de la organización, a su vez brinda una forma

[47] http://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios

estándar de exposición e invocación de servicios (*comúnmente pero no exclusivamente servicios web*), lo cual facilita la interacción entre diferentes sistemas propios o de terceros.

SOA define las siguientes capas de software:

- **Aplicaciones básicas** - Sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad;
- **De exposición de funcionalidades** - Donde las funcionalidades de la capa aplicativas son expuestas en forma de servicios (*servicios web*);
- **De integración de servicios** - Facilitan el intercambio de datos entre elementos de la capa aplicativa orientada a procesos empresariales internos o en colaboración;
- **De composición de procesos** - Que define el proceso en términos del negocio y sus necesidades, y que varía en función del negocio;
- **De entrega** - donde los servicios son desplegados a los usuarios finales.

SOA proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación.

Un servicio es el punto final de una conexión. Además, un servicio tiene algún tipo de sistema de computación subyacente que soporta la conexión ofrecida. La combinación de servicios –internos y externos a la organización- configura una *arquitectura orientada a servicios*. Al ser más compleja la coordinación de todos estos servicios para obtener como resultado una aplicación, también es necesario apelar a conceptos –metáforas- utilizados en actividades de coordinación de conjuntos: *coreografía* y *orquestación*. Y también es cierto que aparecieron es ese orden.

Se considera a la *coreografía* un equivalente de los conceptos de orquestación, de colaboración, coordinación, etc. Por otro lado, la *orquestación* es considerada como la coordinación de eventos de un proceso, también superponiéndose con el concepto de *coreografía*. La *orquestación* dirige y gestiona el ensamblado sobre demanda de múltiples servicios componentes para crear una aplicación o proceso

de negocio compuesto. La orquestación tiende a ser considerada como una única fuerza coordinadora, mientras que la coreografía también se aplica a una coordinación compartida a través de múltiples sistemas autónomos.

En el nuevo modelo de empresa ágil y eficiente, el software deberá implementarse como componentes para una fácil reutilización y adaptación a las arquitecturas orientadas a servicios. La orquestación es una lógica de negocio que secuencia, coordina y gestiona conversaciones entre servicios Web.

Tabla 3. Definiciones SOA.
Fuente: http://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios

TÉRMINO	DEFINICIÓN / COMENTARIO
Servicio	Una función sin estado (*Existen servicios asíncronos en los que una solicitud a un servicio crea, por ejemplo, un archivo, y en una segunda solicitud se obtiene ese archivo*), auto-contenida, que acepta una(s) llamada(s) y devuelve una(s) respuesta(s) mediante una interfaz bien definida. Los servicios pueden también ejecutar unidades discretas de trabajo como serían editar y procesar una transacción. Los servicios no dependen del estado de otras funciones o procesos. La tecnología concreta utilizada para prestar el servicio no es parte de esta definición.
Orquestación	Secuenciar los servicios y proveer la lógica adicional para procesar datos. No incluye la presentación de los datos. Coordinación.
Sin estado	No mantiene ni depende de condición pre-existente alguna. En una SOA los servicios no son dependientes de la condición de ningún otro servicio. Reciben en la llamada toda la información que necesitan para dar una respuesta. Debido a que los servicios son "sin estado", pueden ser secuenciados (*orquestados*) en numerosas secuencias (*algunas veces llamadas tuberías o pipelines*) para realizar la lógica del negocio.
Proveedor	La función que brinda un servicio en respuesta a una llamada o petición desde un consumidor.
Consumidor	La función que consume el resultado del servicio provisto por un proveedor.

En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado.

Los beneficios que puede obtener una organización que adopte SOA son:

- Mejora en los tiempos de realización de cambios en procesos.
- Facilidad para evolucionar a modelos de negocios basados en tercerización.
- Facilidad para abordar modelos de negocios basados en colaboración con otros entes (*socios, proveedores*).
- Poder para remplazar elementos de la capa aplicativa SOA sin disrupción en el proceso de negocio.
- Facilidad para la integración de tecnologías disímiles.

En el contexto empresarial.[48]

La Arquitectura SOA establece un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicios. La forma más habitual de implementarla es mediante Servicios Web, una tecnología basada en estándares e independiente de la plataforma, con la que SOA puede descomponer aplicaciones monolíticas en un conjunto de servicios e implementar esta funcionalidad en forma modular.

Servicio.

Un servicio es una funcionalidad concreta que puede ser descubierta en la red y que describe tanto lo que puede hacer como el modo de interactuar con ella. Desde la perspectiva de la empresa, un servicio realiza una tarea concreta: puede corresponder a un proceso de negocio tan sencillo como introducir o extraer un dato como “Código del Cliente”.

Pero también los servicios pueden acoplarse dentro de una aplicación completa que proporcione servicios de alto nivel, con un grado de complejidad muy superior por ejemplo, “introducir datos de un pedido”, un proceso que, desde que comienza hasta que termina, puede involucrar varias aplicaciones de negocio.

Estrategia de orientación a servicios.

La estrategia de orientación a servicios permite la creación de servicios y aplicaciones compuestas que pueden existir con independencia de las tecnologías subyacentes. En lugar de exigir que todos los datos y lógica de negocio residan en un mismo ordenador, el modelo de servicios facilita el acceso y consumo de los recursos de IT a través de la red. Puesto que los servicios están diseñados para ser independientes, autónomos y para interconectarse adecuadamente, pueden combinarse y recombinarse con suma facilidad en aplicaciones complejas que respondan a las necesidades de cada momento en el seno de una organización.

[48] **HERGUEDA., M.,** La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real., PUBLICACIÓN., Valencia-España, Microsoft Corporation., 2006., Pp. 2-4

Aplicaciones compuestas.

Las aplicaciones compuestas (*también llamadas "dinámicas"*) son lo que permite a las empresas mejorar y automatizar sus procesos manuales, disponer de una visión consistente de sus clientes y socios comerciales y orquestar sus procesos de negocio para que cumplan con las regulaciones legales y políticas internas. El resultado final es que las organizaciones que adoptan la orientación a servicios pueden crear y reutilizar servicios y aplicaciones y adaptarlos ante los cambios evolutivos que se producen dentro y fuera de ellas, y con ello adquirir la agilidad necesaria para ganar ventaja competitiva.

Luego de este análisis, se puede determinar la siguiente tabla que resume las características de los principales estilos arquitectónicos, los atributos de calidad que propician y los atributos que se ven afectados negativamente (*atributos en conflicto*):

Tabla 4. Estilos arquitectónicos y Atributos de calidad
Fuente: Estudio de la influencia de mecanismos arquitectónicos en la calidad del software

ESTILO	DESCRIPCIÓN	ATRIBUTOS ASOCIADOS	ATRIBUTOS EN CONFLICTO
Flujo de datos (*Tubería y filtros*)	El sistema es visto como una serie de transformaciones sobre piezas sucesivas de datos de entrada. El dato ingresa en el sistema, y fluye entre los componentes, de uno en uno, hasta que se le asigne un destino final (*salida o repositorio*).	Reusabilidad Modificabilidad Mantenibilidad	Desempeño
Centrado en datos (*pizarra o repositorio*)	Sistemas en los cuales cierto número de clientes accede y actualiza datos compartidos de un repositorio de manera frecuente.	Integrabilidad Escalabilidad Modificabilidad Mantenibilidad	Seguridad Desempeño Facilidad de prueba
Llamada y retorno (*multicapas*)	El sistema se constituye de un programa principal que tiene el control del sistema y varios subprogramas que se comunican con éste mediante el uso de llamadas. Consiste en estructurar aplicaciones que pueden ser descompuestas en grupos de subtareas.	Modificabilidad Escalabilidad Reusabilidad Facilidad de prueba	Mantenibilidad Desempeño
Máquinas virtuales	Simulan alguna funcionalidad que no es nativa al hardware o software sobre el que está implementado.	Portabilidad	Desempeño
Componentes independientes	Consiste en un número de procesos u objetos independientes que se comunican a través de mensajes.	Modificabilidad Escalabilidad Mantenibilidad	Desempeño Integrabilidad

Tabla 5. Estilos arquitectónicos - Ventajas y desventajas
Fuente: Estudio de la influencia de mecanismos arquitectónicos en la calidad del software

ESTILO	VENTAJAS	DESVENTAJAS
Flujo de datos (*Tubería y filtros*)	Permite entender el sistema global en términos de la combinación de componentes. Permite ir entendiendo el sistema poco a poco. Soporta de buena manera la reutilización ya que el sistema está dividido. Los filtros son independientes de sus vecinos. Facilidad de mantenimiento y facilidad de diagnóstico (*rendimiento*). Se pueden agregar nuevos filtros al sistema y los viejos pueden ser remplazados por versiones mejoradas. Encontrar los errores es una tarea muy ligera. Soportan la ejecución concurrente.	Tiempo de espera. Si un filtro tiene tres tuberías de entradas, tiene que esperar hasta que las tres tuberías no le aporten los datos para poder ejecutarse totalmente. Es normal que las tuberías solo suelen aceptar un tipo de datos, lo que las restringe enormemente. Los datos deben ser transformados para enviar por las tuberías (*hay costo de transformación de datos*). No son adecuados para aplicaciones interactivas (*orientados al procesamiento secuencial, no interactivo*). Puede ser un obstáculo el tener que mantener las correspondientes diferencias entre dos flujos separados pero relacionados. El costo del paralelismo puede ser muy grande.
Centrado en datos (*pizarra o repositorio*)	Los clientes son independientes entre sí; un cliente puede ser modificado sin afectar a los otros. Posibilita la integración de agentes; nuevos clientes pueden ser agregados con facilidad. Adecuado para la resolución de problemas no deterministas. Son especiales para inteligencia artificial.	Estructura de datos común a todos los agentes, lo que genera problemas de seguridad. Problemas de sincronización a la hora de chequear y vigilar el estado de la pizarra.
Llamada y retorno (*multicapas*)	La descomposición en módulos disminuye la complejidad. Como cada nivel implementa unas interfaces claras y lógicas, pueden intercambiarse permitiendo la reutilización de componentes. Facilita la evolución del sistema, ya que los cambios en una capa, apenas afectan a la superior e inferior, mejorando notablemente su mantenibilidad.	No todos los sistemas se pueden estructurar fácilmente como capas. La comunicación a través de las diferentes capas puede hacer ineficiente el sistema.

Máquinas virtuales	Permite trabajar en varios niveles de abstracción. Para implementar los niveles superiores no se requiere conocer el entorno subyacente, basta con las interfaces que proporcionen los niveles inferiores.	
	Facilitan la portabilidad. Permiten la simulación de sistemas no existentes. Proporcionan un alto nivel de abstracción. Mejoran la flexibilidad e interacción.	Problemas de rendimiento e integrabilidad.
Componentes independientes	Fuerte soporte para la reutilización. Cualquier componente puede ser introducido en un sistema simplemente registrándolo para los eventos en el sistema. Se pueden remplazar componentes por otros sin afectar las interfaces de los otros componentes del sistema mejorando el mantenimiento.	Es susceptible a pérdidas de control: se desconoce que componentes responderán a un evento, no se puede confiar en el orden de invocación de los componentes y no se sabe cuando terminan de ejecutarse. Algunas veces los datos pueden ser pasados en un evento, pero en otras situaciones deben confiar en un depósito compartido para interacción. Asegurar el correcto funcionamiento del sistema es difícil porque depende del contexto en que sea invocado.

2.2.6. Atributos de calidad.

El instante en el cual una característica puede ser observada o medida, permite establecer varias clasificaciones de las características de un producto.[49] Para la concepción de los atributos de calidad, se debe considerar dos posibles puntos de vista: desde el punto de vista del diseñador, la calidad del producto final es el grado de concordancia entre los requerimientos del producto y el producto final; desde el punto de vista de la percepción del usuario, la calidad del producto se define por las expectativas y objetivos o necesidades. Partiendo de estas consideraciones, se tienen 2 campos amplios sobre la comprobación en el cumplimiento de los atributos de calidad: atributos observables mediante la ejecución y atributos no observables en la ejecución.

Tabla 6. Atributos de calidad
Fuente: Atributos de calidad para Componentes COTS

CARACTERÍSTICAS	Atributos observables mediante la ejecución	Atributos no visibles durante la ejecución (*durante el ciclo de vida del producto*)
Funcionalidad (*Functionality*)	Seguridad (*Safety & Security*)	Interoperabilidad (*Interoperability*)
		Escalabilidad (*Scalability*)
Fiabilidad (*Reliability*)	Disponibilidad (*Availability*)	
Facilidad de uso		Usabilidad
Eficiencia	Desempeño (*Performance*)	
	Comportamiento temporal	
	Utilización de recursos	
Mantenibilidad (*Maintainability*)		Modificabilidad (*Modifiability*)
		Portabilidad (*Portability*)
		Reusabilidad (*Reusability*)
		Integrabilidad (*Integrity*)
		Capacidad de prueba (*Testability*)

[49] **BERTOA., M.,** Atributos de calidad para Componentes COTS. Universidad de Málaga., Departamento de Lenguajes y Ciencias de la Computación., TESIS., Málaga-España., 2006., Pp. 1-32

La *"**funcionalidad**"* se podría expresar como la capacidad del componente software para proporcionar un conjunto apropiado de funciones que satisfagan las necesidades establecidas o implícitas cuando se usa bajo las condiciones especificadas, proporcionando los resultados acordados con el grado necesario de precisión (*Kazman et al., 2001*).Esto se consigue haciendo una arquitectura que permita que las partes del sistema puedan coordinarse logrando el objetivo común, asignando responsabilidades a cada componente de la arquitectura.

La *"**fiabilidad**"* es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo (*Barbacci et al., 1995*). Es aplicable directamente a los componentes y fundamental para su reutilización.

La *"**facilidad de uso**"* cambia de sentido, dado que un componente no será utilizado por un usuario final directamente sino por los diseñadores y desarrolladores de aplicaciones de software. La facilidad de uso real de un componente debe interpretarse como la capacidad del componente para ser utilizado en la construcción de un producto o sistema de software. En este sentido se busca un atributo que mida la facilidad de uso del componente durante el diseño de las aplicaciones, añadiendo la *Complejidad* como una nueva sub-característica cuyo sentido es dar una medida de la complejidad del uso e integración de un determinado componente en un producto o sistema software.

La *"**eficiencia**"* capacidad del producto software para proporcionar tiempos de respuesta, tiempos de proceso y potencia apropiados, bajo condiciones determinadas y la cantidad de recursos adecuados cuando el software lleva a cabo su función, aunque la mayoría de las propuestas de otros autores prefieren hablar de rendimiento (*performance*).

La *"**mantenibilidad**"* es la capacidad del producto software de poder diagnosticar deficiencias o causas de los fallos en el software, o para identificar las partes que han de ser modificadas, entendiéndose por modificación a cualquier reparación, evolución o adaptación del software (*Barbacci et al., 1995*). En este sentido y aunque las modificaciones internas no serán realizadas por el usuario del componente, sí necesitará probar el componente antes de incluirlo en su aplicación o cambiar alguno de los parámetros que se pueden particularizar.

a) Atributos observables mediante la ejecución.[50]

- **Seguridad**: *Safety*, es la medida de ausencia de errores que generan pérdidas de información (*Barbacci et al., 1995*). *Security*, es la capacidad del producto software para proteger información y datos de manera que las personas o sistemas no autorizados no puedan leerlos o modificarlos, al tiempo que no se deniega el acceso a las personas o sistemas autorizados (*Kazman et al., 2001*). Toda estrategia que se utilice para el control de seguridad, implican identificar componentes especializados y coordinar su funcionamiento con las demás componentes.

- **Disponibilidad**: Capacidad del producto de estar siempre disponible a pesar de presentar fallas en un intervalo temporal determinado. Esto se consigue diseñando los componentes de arquitectura, tolerante a fallas, robustos, fáciles de reparar y modificar, procurando que la arquitectura permita un bajo acoplamiento. La disponibilidad se mide como el tiempo entre fallas o la rapidez en que el sistema puede reiniciar la operación cuando ocurre una falla.

$$\alpha = \frac{\text{tpef}}{\text{tpef} + \text{tmr}}$$

  En donde:
  α→ grado de disponibilidad.
  tpef→ tiempo promedio entre fallas
  tmr→ tiempo medio de reparación.

- **Performance** (*rendimiento*): Capacidad del producto software para responder a un evento o estímulo, o bien el número de eventos procesados en un intervalo de tiempo (*Smith, 1993*). Dependerá de la comunicación existente entre los componentes de la arquitectura y la asignación de funcionalidades a las componentes (*Bass et al., 1998*). Puede analizarse mediante análisis de tasas de llegada, distribución de los requerimientos de servicios, tiempo de procesamiento, tamaño de las colas, latencia (*tasa de servicio*) (*IEEE 610.12*).

[50] **BERTOA., M.,** Atributos de calidad para Componentes COTS. Universidad de Málaga., Departamento de Lenguajes y Ciencias de la Computación., TESIS., Málaga-España., 2006., Pp. 1-32

- **Comportamiento temporal**: Se distingue entre los atributos que son apropiados para valores discretos y los que se pueden asociar a un flujo continuo de datos (*data streaming*). En el primer caso se considera el tiempo de respuesta (*response time*) el mismo que evalúa el tiempo transcurrido desde que se recibe la petición hasta que se emite la respuesta. En el segundo caso, se analiza la capacidad de emisión (*throughput*) en un intervalo de observación dado y la capacidad de recepción (*capacity*) en un intervalo de observación dado. Ambos se miden mediante un /emphEntero en kilobytes de información por unidad de tiempo.

- **Utilización de recursos**: Este atributo mide el consumo de recursos respecto a las necesidades de memoria (*número de kilobytes*) que necesita el componente para su ejecución. Además este atributo mide el espacio en disco (*número de kilobytes*) que necesita el componente tanto para almacenarse y como para ser utilizadas sus operaciones o servicios.

b) Atributos no visibles durante la ejecución.[51]

- **Interoperabilidad**: Capacidad que tiene el sistema puede interactuar con uno o más aplicaciones especificadas, posiblemente en un entorno heterogéneo, desarrolladas por organizaciones independientes. Es un tipo especial de integrabilidad (*Bass et al., 1998*).

- **Escalabilidad**: Grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental (*Pressman, 2002*).

- **Usabilidad**: Capacidad del producto software para ser atractivo al usuario, al mismo tiempo que le permita fácilmente entender, aprender y operar. Gran parte de los mecanismos para lograr la usabilidad no tienen relación con la arquitectura, por ejemplo, la forma de interfaz usuaria, la distribución de los elementos dentro de ella, los colores en la pantalla, etc. Otros elementos si tienen relación con la arquitectura, por ejemplo, el proceso mediante el cual los componentes entregan la información relevante para el usuario en una determinada interfaz.

[51] **BERTOA., M.,** Atributos de calidad para Componentes COTS. Universidad de Málaga., Departamento de Lenguajes y Ciencias de la Computación., TESIS., Málaga-España., 2006., Pp. 1-32

- **Modificabilidad o extensibilidad**: Capacidad del producto software que permite que una determinada modificación (*cambio futuro*) sea implementada, bien para mejorar las propiedades del sistema ("*reparability*") o para introducir nuevas funcionalidades ("*evolvability*") de una forma rápida y poco costosa (*Bosch et al., 1999*). Es el atributo de calidad más íntimamente relacionado con la arquitectura. Hacer modificaciones en un sistema consiste en dos etapas: localizar el (*los*) lugar(*es*) dónde debe aplicarse el cambio y aplicar el cambio propiamente dicho. La arquitectura debe permitir definir las componentes y sus responsabilidades.

- **Portabilidad**: Capacidad del producto software para ser adaptado, instalado y ejecutado en diferentes entornos heterogéneos especificados (*hardware, software o una combinación de ambos*), sin aplicar acciones o mecanismos distintos de aquellos proporcionados para este propósito por el propio software considerado (*Kazman et al., 2001*).

- **Reusabilidad**: Grado en el que la estructura o parte de los componentes del sistema pueden ser reutilizados para construir nuevos productos (*Bass et al., 1998*). Es una forma particular de modificabilidad. La reusabilidad se relaciona con la arquitectura en que las componentes son las principales unidades de reutilización. La reusabilidad dependerá del acoplamiento del componente. La reusabilidad y la modificabilidad son dos caras de la misma moneda; el beneficio de la reutilización es hacer que un sistema sea modificable.

- **Integrabilidad**: Habilidad del producto software para hacer que piezas de software desarrolladas separadamente, trabajen correctamente al ser integrados. Dependerá de la complejidad de los componentes, de los mecanismos y protocolos de comunicación, de la claridad en la asignación de responsabilidades. La interoperabilidad es una forma de integrabilidad donde las partes del sistema deben trabajar con otro sistema. (*Bass et al. 1998*)

- **Testeabilidad**: Es el grado de facilidad que tiene un sistema para mostrar sus fallas y para ser probada en su completitud, típicamente a través de pruebas de ejecución (*Bass et al. 1998*).

En este proceso, la arquitectura se convierte en un instrumento cuya función principal es la de intervenir en favor del hombre y se la considera como una actividad conciliatoria entre los requerimientos del problema, en términos de una función, y la factibilidad de una solución en términos de un sistema de software. La idea básica es conseguir que el producto final sea un software de calidad.Cabe indicar que no todas las características son aplicables a un producto software.

2.2.7. Dando forma a la Arquitectura.

El objetivo de la fase de requerimientos es para decidir que se va a construir y como documentar los resultados. Seguido a esto está la arquitectura que es el primer artefacto en el proceso de desarrollo que direcciona estos requerimientos, para esto se tiene que considerar toda clase de requerimientos, sean funcionales y no funcionales desde un principio.

Los cambios en los requerimientos ***funcionales*** son fáciles de manejar porque tienden a ser contenidos en un conjunto pequeño de componentes y no afectan como tal a todo el sistema. Por el otro lado, cambios en los requerimientos ***no funcionales*** pueden ser muy perjudiciales y pueden causar problemas serios a la arquitectura, ya que estos están ligados a todo el sistema.

El Arquitecto de Software mide el problema que se va a trabajar, observando el contexto en el cual se desenvuelve. Luego trata entender el tamaño del problema y generar una solución a ese problema. La solución puede encontrarse rápidamente, pues se pueden reutilizar módulos que el arquitecto haya usado antes. Si está trabajando en un dominio que no conoce, el arquitecto puede usar la metáfora de construir una arquitectura fácil de entender basada en un problema similar. Dentro de un proyecto de desarrollo, e independientemente de la metodología que se utilice, se puede hablar de “desarrollo de la arquitectura de software”. Este desarrollo, que precede a la construcción del sistema, está dividido en las siguientes etapas: requerimientos, diseño, documentación y evaluación. Cabe señalar que las actividades relacionadas con el desarrollo de la arquitectura de software generalmente forman parte de las actividades definidas dentro de las metodologías de desarrollo.[52]

[52]**BASS., L., Y OTROS.,** Software Architecture in Practice., 2a.ed., Massachusetts-E.U.A., Addison Wesley Publishing Company Inc.., 2003., Pp. 10-28

A continuación se describen dichas etapas:

- **Requerimientos**. La etapa de requerimientos se enfoca en la captura, documentación y priorización de requerimientos que influencian la arquitectura. Como se mencionó anteriormente, los atributos de calidad juegan un papel preponderante dentro de estos requerimientos, así que esta etapa hace énfasis en ellos. Otros requerimientos, sin embargo, son también relevantes para la arquitectura, estos son los requerimientos funcionales primarios y las restricciones.

- **Diseño**. La etapa de diseño es la etapa central en relación con la arquitectura y probablemente la más compleja. Durante esta etapa se definen las estructuras que componen la arquitectura. La creación de estas estructuras se hace en base a patrones de diseño, tácticas de diseño y elecciones tecnológicas. El diseño que se realiza debe buscar ante todo satisfacer los requerimientos que influencian a la arquitectura, y no simplemente incorporar diversas tecnologías por que están “de moda”.

- **Documentación.** Una vez creado el diseño de la arquitectura, es necesario poder comunicarlo a otros involucrados dentro del desarrollo. La comunicación exitosa del diseño muchas veces depende de que dicho diseño sea documentado de forma apropiada. La documentación de una arquitectura involucra la representación de varias de sus estructuras que son representadas a través de distintas vistas. Una vista generalmente contiene un diagrama, además de información adicional, que apoya en la comprensión de dicho diagrama.

- **Evaluación.** Dado que la arquitectura de software juega un papel crítico en el desarrollo, es conveniente evaluar el diseño una vez que este ha sido documentado con el fin de identificar posibles problemas y riesgos. La ventaja de evaluar el diseño es que es una actividad que se puede realizar de manera temprana (*aún antes de codificar),* y que el costo de corrección de los defectos identificados a través de la evaluación es mucho menor al costo que tendría el corregir estos defectos una vez que el sistema ha sido construido. Evaluar la arquitectura antes de la implementación puede reducir costos y detectar desde temprano errores y problemas. Además ayuda a incrementar el entendimiento del sistema y a clarificar y priorizar los requerimientos.

CAPÍTULO 3

SISTEMA HIPOTÉTICO

3.1. Planteamiento de la hipótesis.

Ho: “La implementación del modelo alternativo de arquitectura de software, no permite mejorar la calidad en la proyección y construcción de sistemas lógicos de computadoras.”

Hi: “La implementación de un modelo alternativo de arquitectura de software, permite mejorar la calidad en la proyección y construcción de sistemas lógicos de computadoras.”

3.2. Determinación de variables.

a) Variable Independiente.

Modelo alternativo de arquitectura de software.

b) Variable Dependiente.

Proyección y construcción de sistemas lógicos de computadoras.

3.3. Operacionalización conceptual.

Tabla 7. Tabla de operacionalización conceptual.
Fuente: investigador.

VARIABLES	CONCEPTO
Modelo de arquitectura de software.	Arquitectura que divide y organiza la aplicación en varios módulos que se comunican e interrelacionan entre sí.
Sistema lógico de computadoras.	Aplicaciones de software que cumplen con funcionalidades encaminadas a cubrir necesidades informáticas de todo nivel.

3.4. Operacionalización metodológica de las variables.

Tabla 8. Tabla de Operacionalización metodológica de las variables.
Fuente: investigador.

VARIABLES	DIMENSIONES	INDICADORES	INDICES
Modelo alternativo de arquitectura de software.	Funcionalidad	Escalabilidad	¿En qué nivel cree Ud. que la arquitectura implementada, permita que la aplicación sea escalable? (Bajo – Medio – Alto – Muy alto)
		Disponibilidad	¿Qué valor de relación ha obtenido entre el número de veces que debe parar el sistema, con el número de veces que debe realizar correcciones en el sistema? (Entre 0% y 25% – Entre 25% y 50% – Entre 50% y 75% – Entre 75% y 100%)
		Uso de recursos	¿La división de la aplicación conseguida con la arquitectura propuesta, ha ayudado en la optimización de los recursos tanto del lado del servidor como del lado de los clientes? (Si – No)
	Mantenibilidad	Modificabilidad	¿Qué nivel de esfuerzo y coste conlleva el cambio o diseño de nuevos procesos con el modelo de arquitectura propuesto? (Bajo – Medio – Alto)
		Reusabilidad	La arquitectura propuesta le ha permitido reutilizar componentes diseñados en la aplicación para el diseño de otras funcionalidades. (Siempre – Frecuentemente – A veces – Nunca)
			¿Qué valor de relación ha obtenido entre el número de veces que ha reutilizado un componente, con el número de veces que ha tenido que diseñar procesos en el sistema? (Entre 0% y 25% – Entre 25% y 50% – Entre 50% y 75% – Entre 75% y 100%)
		Acoplamiento	En los procesos de producción de software, ¿Qué nivel de acoplamiento exige la estructura arquitectónica implementada? (Bajo – Medio – Alto)
		Testeabilidad	¿Qué nivel de facilidad presenta el modelo de arquitectura propuesto al momento de la detección de errores en relación a procesos de diseño anteriores? (Mayor – Igual – Menor)
			Al aplicar nuevas funcionalidades o corregir errores en el sistema, ¿se han presentado problemas al momento de probar la aplicación? (Siempre – Frecuentemente – A veces – Nunca)
		Tiempo de desarrollo	El tiempo que emplea actualmente al momento de diseñar y/o mejorar los procesos en la aplicación en relación a la anterior forma de producir aplicaciones es: (Mayor – Igual – Menor)

Proyección y construcción de sistemas lógicos de computadoras.	Calidad	Escalabilidad de módulos	Proyección a la escalabilidad: (*Entre Medio y Alto*)
		Disponibilidad	Relación de caída del sistema: (*Entre 0% y 50%*)
		Uso de recursos	Optimización de recursos: (*Si*)
		Modificabilidad	Nivel de esfuerzo y coste al modificar: (*Entre Bajo y Medio*)
		Reusabilidad de recursos en el proyecto	Reusabilidad de componentes: (*Entre Siempre y Frecuentemente*)
			Relación de reutilización de componentes: (*Entre 50% y 100%*)
		Acoplamiento de módulos	Nivel de acoplamiento: (*Entre Bajo y Medio*)
		Testeabilidad	Facilidad en la detección de errores: (*Mayor*)
			Problemas en pruebas del sistema: (*Entre A veces y Nunca*)
		Tiempo de desarrollo	Tiempo requerido en nuevos diseños: (*Menor*)

CAPÍTULO 4

MARCO METODOLÓGICO

4.1. Localización y temporalización.

El presente trabajo, tuvo lugar en la Fundación San Luis de empresa Pronaca de Bucay, donde las encuestas fueron aplicadas a los profesionales encargados del desarrollo de aplicaciones del departamento de sistemas.

4.2. Tipo y diseño de estudio.

El estudio realizado es de campo – descriptivo – transversal y experimental pues se realizaron estudios específicos en un tiempo determinado y la aplicación de las encuestas, fueron efectuadas a los profesionales del área de sistemas. Además se analizaron y explicaron las características más importantes del modelo de arquitectura de software, sus aplicaciones, el seguimiento y la evaluación del mismo.

4.3. Población, muestra o grupo de estudio.

El modelo alternativo de arquitectura comprometerá a la siguiente población: se considera a todo el personal técnico en vista de que ellos son las personas que van a aplicar el nuevo modelo de arquitectura directamente.

- Técnicos colaboradores del área de sistemas de la empresa Pronaca – Bucay *(3 integrantes)*
- Profesionales del área de sistemas de la Fundación San Luis - Bucay *(3 integrantes*)
- Personal del área de sistemas de la Institución San Juan de Bucay. *(3 integrantes*)

Tabla 9. Tabla de total de muestras.
Fuente: investigador.

DETALLE	MUESTRA
Personal técnico del área de sistemas Pronaca-Bucay	3
Personal técnico del área de sistemas Fundación San Luis	3
Personal técnico del área de sistemas Institución San Juan	3
Total Muestra	**9**

4.4. Métodos, técnicas e instrumentos de recolección de datos.

Se utilizará el método científico y como técnicas se usó la observación y la encuesta.

En cuanto a la prueba de hipótesis, por ser correlacional se utilizó es estadístico “chi-cuadrado”, que es una prueba no paramétrica que mide la discrepancia entre una distribución observada y otra teórica.

CAPÍTULO 5

ANÁLISIS, INTERPRETACIÓN Y PRESENTACIÓN DE RESULTADOS

5.1. Análisis de los profesionales que utilizan modelos de arquitectura para el desarrollo de aplicaciones.

Para el desarrollo de esta investigación, se efectuó una encuesta (*ver anexo 3*) al personal del área de sistemas de la empresa Pronaca, personal encargado del desarrollo de aplicaciones de la fundación San Luis y personal técnico del área de sistemas de la Institución San Juan, las mismas que se encuentran ubicadas en el cantón Bucay de la provincia del Guayas, dando un total de 9 personas.

Las personas encuestadas son profesionales que podrán dar una versión técnica sobre el modelo de arquitectura.

5.2. Resultados de la encuesta.

5.2.1. Pregunta 1 – Escalabilidad.

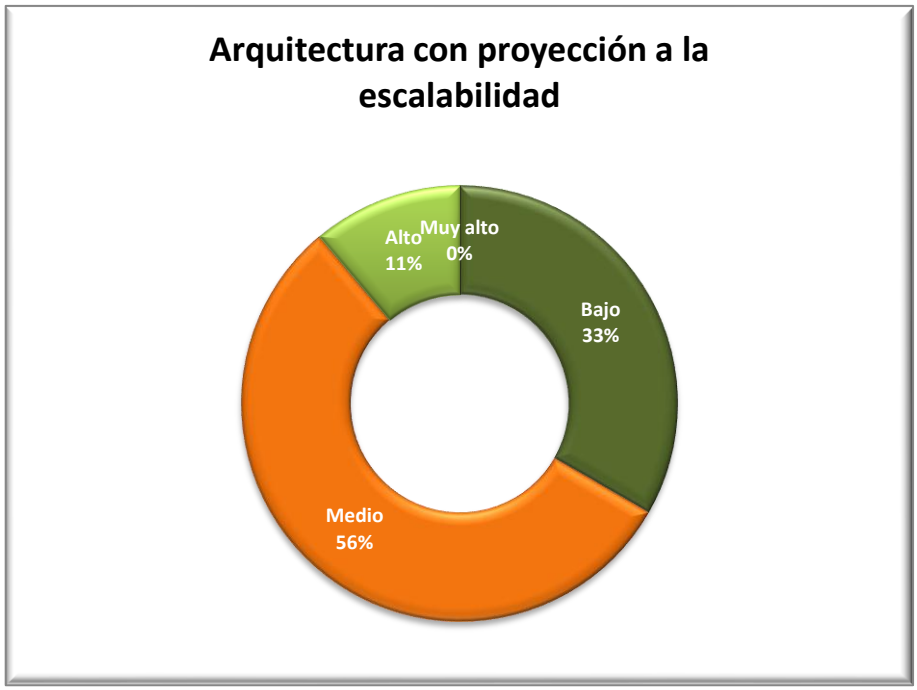

Gráfico 1. Arquitectura con proyección a la escalabilidad
Fuente: investigador

5.2.2. Pregunta 2 – Disponibilidad.

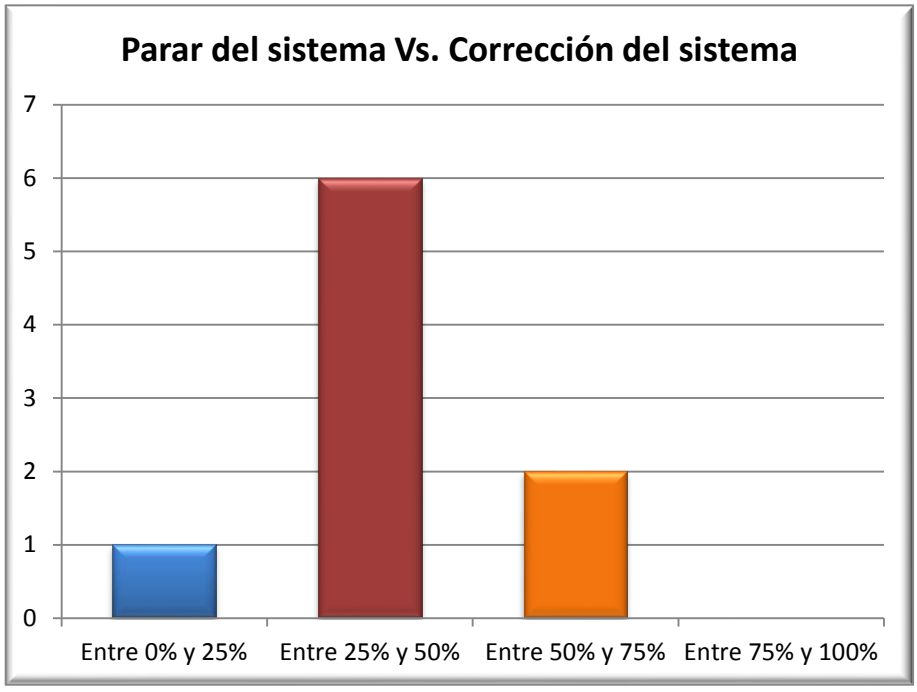

Gráfico 2. Relación entre paradas del sistema Vs. Corrección del sistema
Fuente: investigador

5.2.3. Pregunta 3 – Uso de recursos.

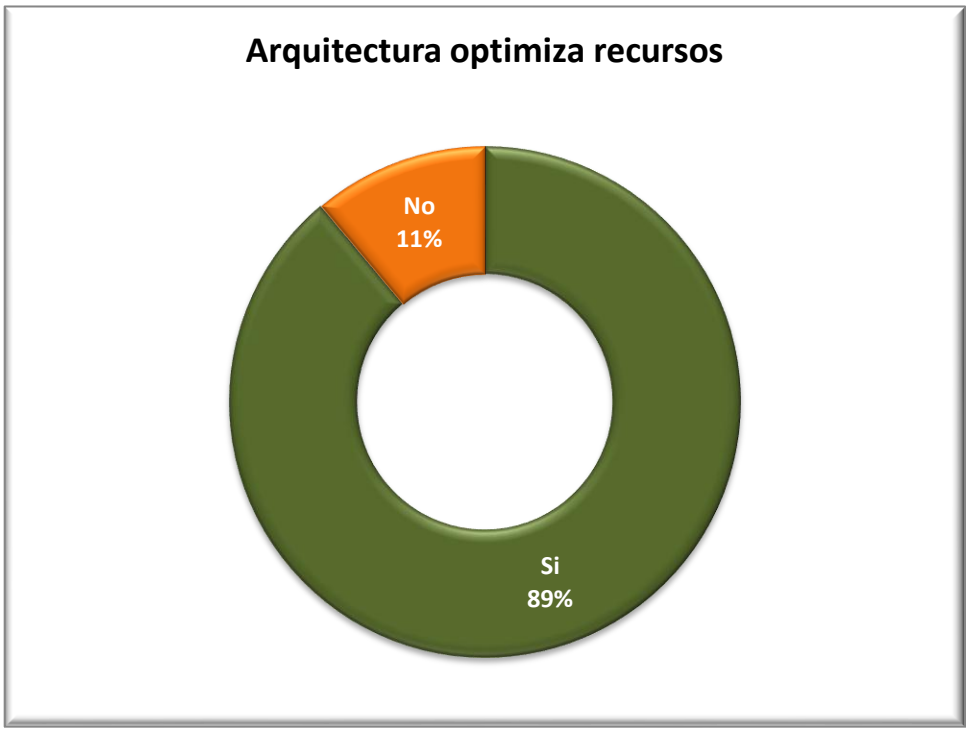

Gráfico 3. Optimización de recursos
Fuente: investigador

5.2.4. Pregunta 4 – Modificabilidad.

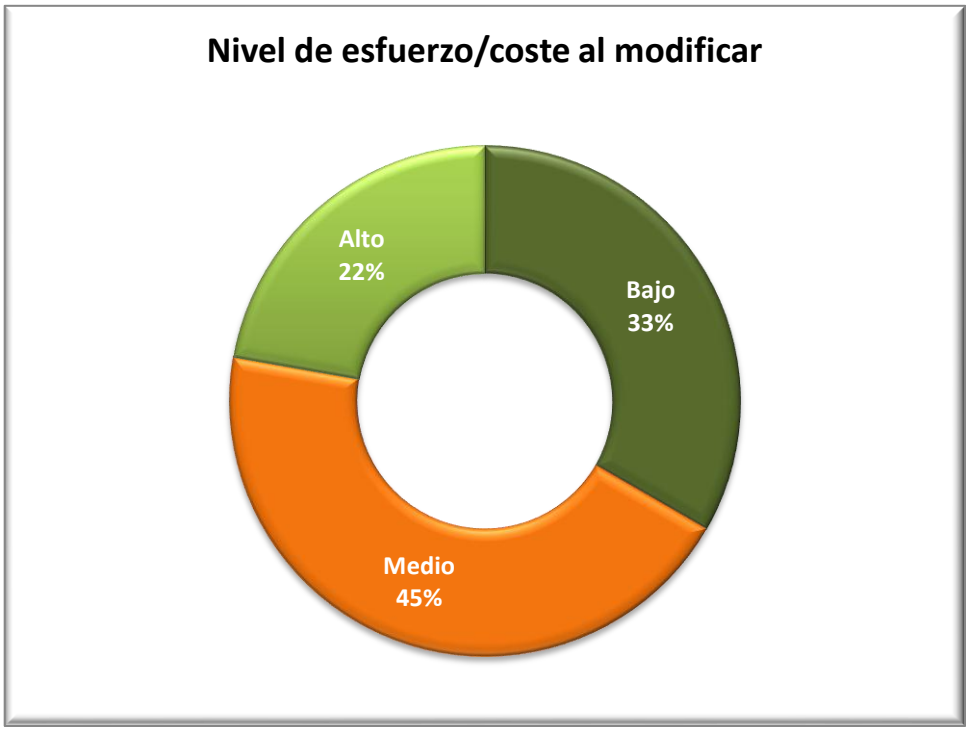

Gráfico 4. Nivel de esfuerzo/coste al modificar
Fuente: investigador

5.2.5. Pregunta 5 - Reusabilidad.

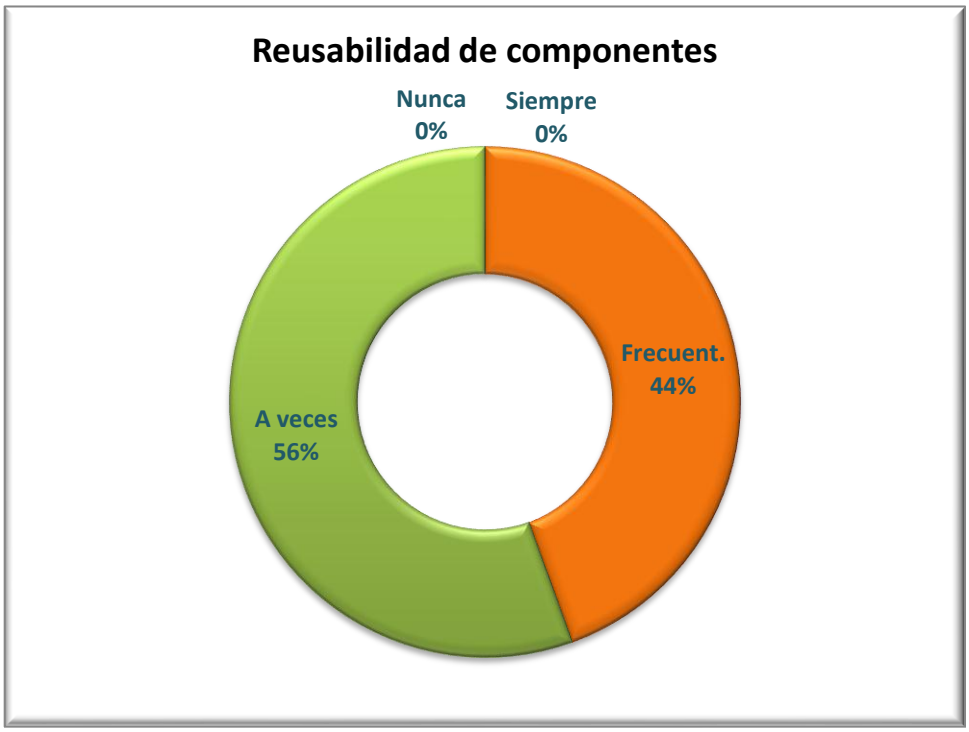

Gráfico 5. Reusabilidad de componentes
Fuente: investigador

5.2.6. Pregunta 6 – Reusabilidad.

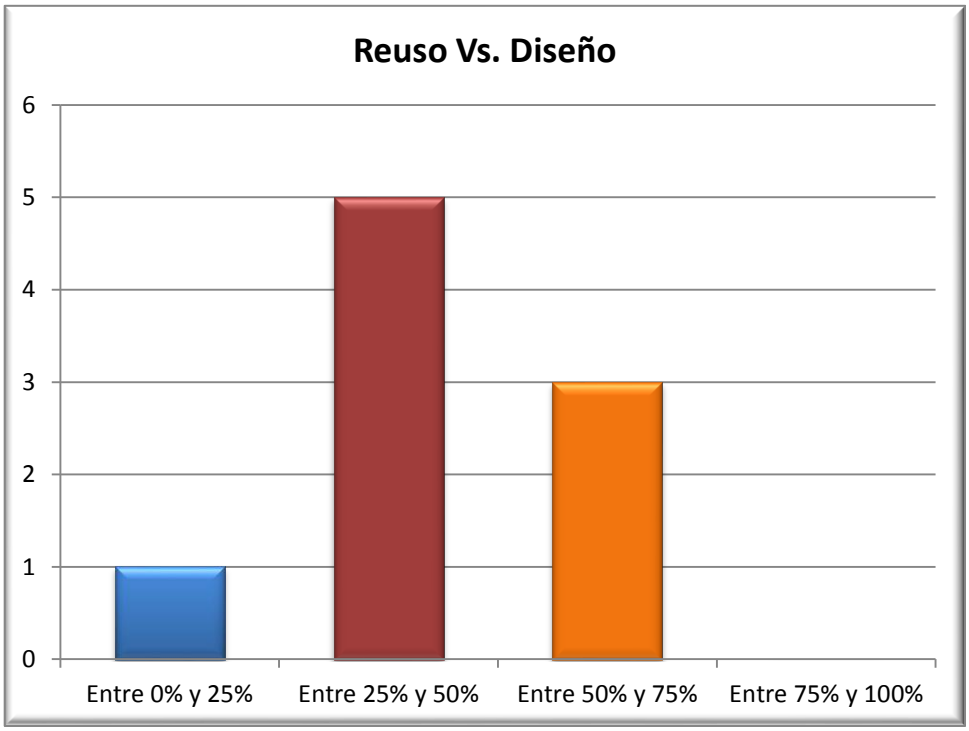

Gráfico 6. Relación Reusabilidad Vs. Diseño
Fuente: investigador

5.2.7. Pregunta 7 – Acoplamiento.

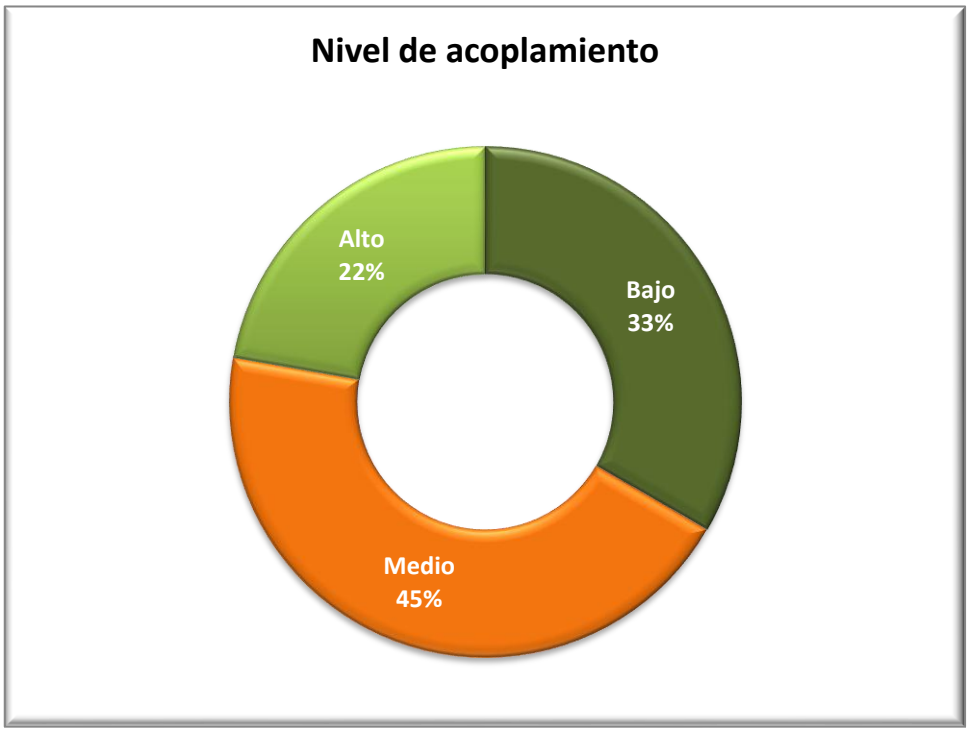

Gráfico 7. Nivel de acoplamiento
Fuente: investigador

5.2.8. Pregunta 8 – Testeabilidad.

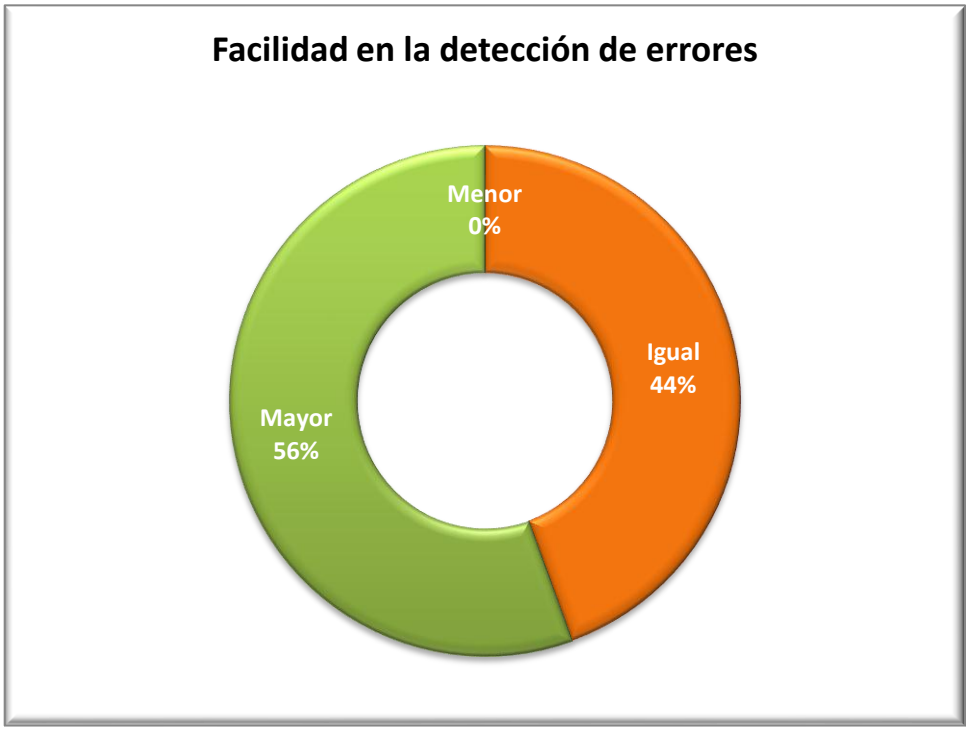

Gráfico 8. Facilidad en la detección de errores
Fuente: investigador

5.2.9. Pregunta 9 – Tiempo de desarrollo.

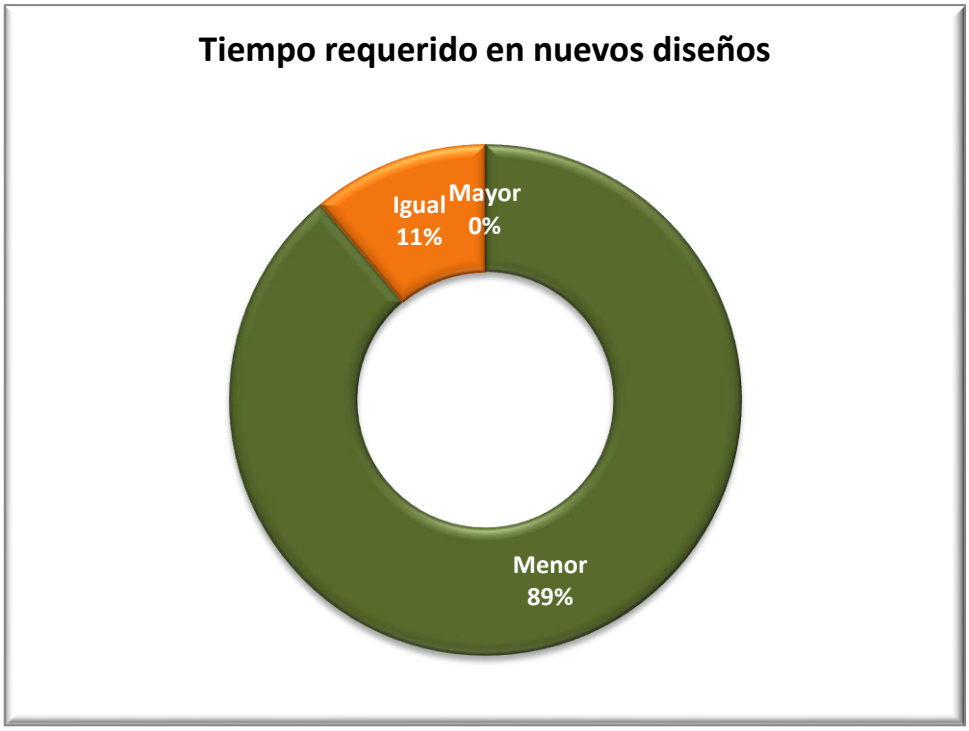

Gráfico 9. Tiempo requerido en nuevos diseños
Fuente: investigador

5.3. Prueba de hipótesis.

5.3.1. Planteamiento de las hipótesis.

Ho: “La implementación del modelo alternativo de arquitectura de software, no permite mejorar la calidad en la proyección y construcción de sistemas lógicos de computadoras.”

Hi: “La implementación de un modelo alternativo de arquitectura de software, permite mejorar la calidad en la proyección y construcción de sistemas lógicos de computadoras.”

5.3.2. Nivel de significancia.

Una vez establecida la hipótesis nula y alternativa, se debe determinar el nivel de significación que para el caso de estudio se utilizará un nivel de significación estadística de **α = 0,10**.

5.3.3. Criterio.

De acuerdo al análisis desarrollado en la presente investigación, se ha seleccionado como estadístico de prueba de hipótesis la técnica “chi-cuadrado”. La fórmula que da el estadístico es la siguiente:

$$x^2 = \sum_i \frac{(observada_i - esperada_i)^2}{esperada_i}$$

Para conocer las frecuencias teóricas o esperadas, se calculan a través del producto de los totales marginales (*total del renglón x total de columna*), dividido por el número total de casos (*gran total*):

$$fe = \frac{(total\ del\ renglón)x(total\ de\ la\ columna)}{gran\ total}$$

En la ***tabla 13*** se pueden observar los resultados de los cálculos, tanto de la frecuencia esperada, como la del valor de "$x^2_{calculado}$", luego de haber aplicado las fórmulas anteriores.

Ahora es necesario determinar el **criterio de decisión**. Entonces se acepta H_0 cuando: $x^2_{calculado} < x^2_{tabla}$, en caso contrario se rechaza Ho. Donde el valor de x^2_{tabla} representa el valor proporcionado por la tabla de "distribución x^2", según el nivel de significación elegido y los grados de libertad.

Como se mencionó anteriormente, el nivel de significancia adoptado para esta investigación es de **α = 0,10**.

Para la determinación de los grados de libertad (**gl**)se debe aplicar la siguiente fórmula:

$$gl = (r-1)x(k-1)$$

Donde $r$es el número de filas o renglones y $k$el de columnas. La investigación generó una matriz de 2rx 2k. (*Ver detalle en tabla 12 del ítem "Cálculos"*).

Entonces:

gl = (2 – 1) * (2 – 1)
gl = (1) * (1)
gl = 1 grado de libertad.

De acuerdo a la tabla estadística de distribución de chi-cuadrado (*Figura 19*), con un nivel de significancia 0,1 a 1 grado de libertad, genera un valor de **x^2_{tabla} =2,706**.

Degrees of Freedom	Possibility of Chance Occurrence in Percentage (5% or Less Considered Significant)								
	90%	80%	70%	50%	30%	20%	10%	5% (sig.)	1%
1	0.016	0.064	0.148	0.455	1.074	1.642	2.706	3.841	6.635
2	0.211	0.446	0.713	1.386	2.408	3.219	4.605	5.991	9.210
3	0.584	1.005	1.424	2.366	3.665	4.642	6.251	7.815	11.341
4	1.064	1.649	2.195	3.357	4.878	5.989	7.779	9.488	13.277
5	1.610	2.343	3.000	4.351	6.064	7.289	9.236	11.070	15.086
6	2.204	3.070	3.828	5.348	7.231	8.558	10.645	12.592	16.812
7	2.833	3.822	4.671	6.346	8.383	9.083	12.017	14.067	18.475
8	3.490	4.594	5.527	7.344	9.524	11.030	13.362	15.507	20.090
9	4.168	5.380	6.393	8.343	10.656	12.242	14.684	16.919	21.666

Figura 19. Tabla de Distribución de X^2
Fuente: Estadística A – Distribución Chi-Cuadrado. Ing. José Manuel García.

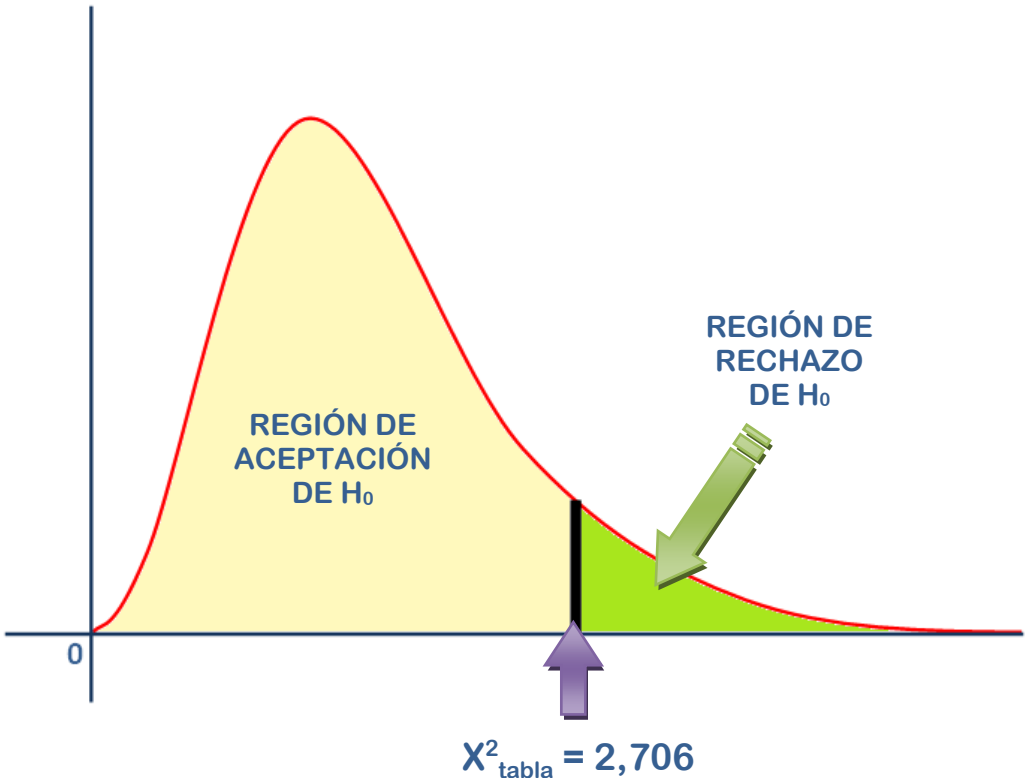

Figura 20. Región de aceptación y rechazo de Ho
Fuente: investigador.

La regla de decisión es entonces: No rechazar H_0 si el valor que se encuentre para $\mathbf{x^2_{calculado}}$ es menor que 2,706. Si el valor calculado es igual o mayor al valor crítico, se rechaza H_0 y se acepta H_1.

5.3.4. Cálculos.

Los resultados que arrojó la investigación realizada se resume en las tablas mostradas en la página siguiente:

Tabla 10. Resultados de encuesta sobre "Funcionalidad"
Fuente: investigador.

FUNCIONALIDAD		
PREGUNTAS	Mejora	No mejora
Pregunta 1 (Escalabilidad)	6	3
Pregunta 2 (Disponibilidad)	7	2
Pregunta 3 (Uso de recursos)	8	1
TOTAL	**21**	**6**

Tabla 11. Resultados de encuesta sobre "Mantenibilidad"
Fuente: investigador.

MANTENIBILIDAD		
PREGUNTAS	Mejora	No mejora
Pregunta 4 (Modificabilidad)	7	2
Pregunta 5 (Reusabilidad)	4	5
Pregunta 6 (Reusabilidad)	3	6
Pregunta 7 (Acoplamiento)	7	2
Pregunta 8 (Testeabilidad)	5	4
Pregunta 9 (Tiempo de desarrollo)	5	4
TOTAL	**31**	**23**

La matriz de resultados se conformaría de la siguiente manera:

Tabla 12. Matriz (*2r x2k*) sobre resultados totales de encuesta.
Fuente: investigador.

Proyección y construcción de sistemas lógicos de computadoras / Modelo alternativo de arquitectura de software	Mejora la calidad	No mejora la calidad	TOTAL
FUNCIONALIDAD	21	6	27
MANTENIBILIDAD	31	23	54
TOTAL	**52**	**29**	**81**

Se diseña la tabla para aplicar la fórmula del chi-cuadrado:

Tabla 13. Tabla de frecuencias observada / frecuencia teórica.
Fuente: investigador.

	fo	fe	$(fo-fe)^2 /fe$
El modelo mejora la calidad en funcionalidad.	21	17	0,776
El modelo mejora la calidad en la mantenibilidad.	31	35	0,388
El modelo no mejora la calidad en funcionalidad.	6	10	1,391
El modelo no mejora la calidad en la mantenibilidad.	23	19	0,695
TOTAL	**81**	**81**	**3,250**

5.3.5. Decisión.

Como:

$\mathbf{x^2_{calculado}} = 3{,}250$ y

$\mathbf{x^2_{tabla}} = 2{,}706$

Entonces:

$\mathbf{x^2_{calculado}} > \mathbf{x^2_{tabla}}$,

Lo que significa que $\mathbf{x^2_{calculado}}$, está en la zona de rechazo de la H_0, entonces se concluye quese rechaza la hipótesis nula y se acepta la de investigación, esto es "*La implementación de un modelo alternativo de arquitectura de software, permite mejorar la calidad en la proyección y construcción de sistemas lógicos de computadoras*."

CAPÍTULO 6

PROPUESTA

6.1. Antecedentes.

Hemos podido observar como la tecnología se ha ido innovando durante los últimos años tanto a nivel de hardware como de software, lo que ha provocado otra evolución paralela: la de la arquitectura de las aplicaciones. A medida que aparecen nuevos recursos técnicos, los patrones de diseño se amoldan para aprovechar las nuevas características que estás novedades ofrecen.

Una arquitectura de software constituye el cimiento para representar la estructura de una aplicación de una manera abstracta y facilita la forma de interpretar el modelo que encapsula. Explica claramente los elementos que la forman y la manera en que se comunican entre ellos. Es importante como disciplina debido a que los sistemas de software crecen de forma tal que resulta muy complicado que sean diseñados, especificados y entendidos por un solo individuo. Uno de los aspectos que motivan el estudio en este campo es el factor humano, en términos de aspectos como inspecciones de diseño, comunicación a alto nivel entre los miembros del equipo de desarrollo, reutilización de componentes y comparación a alto nivel de diseños alternativos.

No se podría hablar de calidad de software si no se tiene un proceso sistemático que garantice cumplir con esta meta. Implementar aplicaciones de software sin una estructura arquitectónica, con llevará a la creación de aplicaciones que no cumplan con los mínimos requerimientos de calidad y su producción será algo más que ineficiente.

6.2. Justificación.

En la actualidad, nos encontramos en una era en donde la necesidad de desarrollar una aplicación simple o compleja en cortos periodos de tiempo, con menores esfuerzos tanto humanos como económicos y una máxima reutilización del software, se acrecienta día a día. Conforme cobra auge el desarrollo de aplicaciones, aumenta la necesidad de disponer de herramientas que permitan tales objetivos. Construir una aplicación se convierte por tanto en la búsqueda de estrategias que permitan alcanzar productos de calidad con el menor coste posible.

El departamento de diseño de sistemas de la Fundación “San Luis” Pronaca del sector de Bucay, se ha enfrentado a problemáticas generales que han mermado el desarrollo de aplicaciones, entre otras están:

- El crecimiento acelerado de las necesidades informáticas internas de la Fundación en relación a la evolución de la aplicación implementada,
- La complejidad para poder desarrollar procesos que ayuden a cubrir estas necesidades y además presentarlos en los plazos previstos,
- Costes que no se ajustan al presupuesto inicial,
- Incumpliendo de los niveles deseables de los requisitos especificados por el usuario,
- Código inmantenible que dificulta la evolución de versiones, retardando la mejora en las prestaciones con respecto a las anteriores,
- Procesos artesanales de producción con escasez de herramientas,
- Baja calidad del software, etc.

Todo esto conlleva a incrementos constantes de los costos de desarrollo, la mantenibilidad y la modificabilidad, debido entre otros factores, a niveles de productividad bajos.

Con la intención de ser productivos dentro del departamento de sistemas de la Fundación, nace esta investigación que consiste en la implementación de un modelo de arquitectura que permita innovar las actividades de producción de software y que permita compatibilizar procesos, utilizar mejores prácticas y acelerar los procesos de

desarrollo de aplicaciones eficientes, de una forma planificada y en periodos de tiempo notablemente cortos.

En este contexto, se pretende que las personas involucradas en el diseño de procesos, utilicen metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan uniformar la filosofía de trabajo, en alas de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software. Se pretende que las personas involucradas en el diseño de procesos, sigan un estándar arquitectónico permitiendo que estos diseños se puedan implementar por separado y que su integración, mejora o corrección se realicen de forma transparente unas de otras con el menor coste posible.

6.3. Modelo de arquitectura para el desarrollo de aplicaciones.

6.3.1. Objetivo general.

Proporcionar a los diseñadores de software una arquitectura para el desarrollo de aplicaciones dentro de la Fundación “San Luis” Pronaca, del sector de Bucay.

6.3.2. Objetivos específicos.

- Establecer el conjunto de módulos que propone el modelo, los mismos que conformarán la solución general a diseñarse.
- Asignar a los módulos, tareas específicas de comportamiento dentro de la solución.
- Fijar la forma en que los módulos han de comunicarse, para lograr la integración de servicios.
- Proporcionar los lineamientos de aplicación del modelo de arquitectura a los diseñadores y programadores de software.

6.4. Pre-requisitos para la construcción de la arquitectura.

Lo que se pretende con el desarrollo de la propuesta, es implementar un modelo de arquitectura de software para que los sistemas construidos dentro de la fundación,

tiendan a cumplir con las características de una aplicación de calidad y reducir el coste de implementación. Se empezará por analizar los pre-requisitos que se consideran influencia para la creación de la arquitectura. (*Ver página siguiente*)

Tabla 14. Análisis de pre-requisitos.
Fuente: investigador.

PARÁMETROS DE CALIDAD	PROGRAMACIÓN TRADICIONAL ESTRUCTURADA	LO QUE SE PRETENDE ALCANZAR CON LA NUEVA ARQUITECTURA
Mantenibilidad	En la forma que se ha realizado la construcción de los sistemas, el mantenimiento de cada uno de ellos se ha tornado muy complejo, como lo mencioné anteriormente. Los sistemas constituían interfaces y procesos como un solo componente y para poder dar mantenimiento, detectar los problemas o pretender modificar, se debía abrir todo el sistema, revisar la parte de código, el procedimiento, la función o la interfaz. Muchas veces el intentar arreglar o mejorar era tan desfavorable que se terminaba realizando todo un módulo completo o remplazando varias partes del sistema. Sumado a estos inconvenientes, el identificar cual iba a ser el impacto con respecto a la funcionalidad del sistema en general, era un asunto preocupante que se comprobaría luego de poner en funcionamiento nuevamente el sistema. A más de estos inconvenientes, siempre ha causado conflictos la distribución del sistema, es decir, al contar con una cantidad de alrededor de 60 equipos en donde se ejecutan las aplicaciones, cuando se ha efectuado un cambio, sea este simple o complejo, se debía realizar un mínimo de 60 actualizaciones.	La arquitectura pretende dar lineamientos de construcción de un sistema, es decir, determinar las partes necesarias para la elaboración de un software y simplemente establecer la vía de comunicación o de interacción entre ellas. Si hacemos una analogía con la arquitectura tradicional, vemos que se tienen planos para cada uno de los componentes que influyen en una construcción como por ejemplo, planos eléctricos, planos de tuberías de agua, planos telefónicos, planos de áreas de almacenamiento y planos de la estructura en general. Si la construcción debería cambiar o agregar más tomas de corriente o más instalaciones telefónicas, lo único que se debería hacer es acudir al plano correspondiente e identificar la mejora o el incremento sin que esto influya en los demás subsistemas. Con la construcción de la arquitectura se pretende definir estructuralmente un sistema, para que sus partes puedan actuar independientemente unas de otras y poder lograr que los cambios o incrementos que se requieran hacer en ellas, no influyan en la estructura ni en el rendimiento de los demás componentes del sistema. Permitir que la modificación del software, ya sea para mejorar las

		propiedades del sistema o para introducir nuevas funcionalidades, se lo realice con mayor simplicidad.
Reusabilidad	Al realizar el análisis de requisitos de un sistema, ya se puede identificar todas aquellas partes que pueden ser comunes, es decir, que se repetirán varias veces en el desarrollo de la aplicación. A pesar de esto, en el modelo que se ha venido implementando la aplicación (*estructurado*), lo único que se ha podido hacer es tratar de agruparlos en procedimientos o funciones. Estos componentes se los ha podido utilizar de forma eficiente pero únicamente dentro de la misma interfaz y se ha convertido en una tarea compleja el tratar de utilizarlos en otras interfaces o en varias instancias de la aplicación; esto refiriéndonos dentro del mismo sistema o aplicación. Se puede analizar además que es factible utilizar al mismo tiempo, muchas de las partes de código, funciones o procedimientos creados para esta aplicación en la implementación de otras aplicaciones, pero por el estilo de programación, el poder transportar dichos elementos se ha convertido en un proceso muy complejo ya que parecen estar hechos específicamente para un solo sistema. Esto ha generado que el tratar de que el sistema o parte de él sea reusable, resulte una tarea difícil.	De lo analizado anteriormente, uno de los atributos que considero importante para el diseño del nuevo modelo de arquitectura, es la reusabilidad, ya que posibilitará la utilización de elementos de la aplicación, varias veces en los diferentes módulos o capas que se desean realizar y además construir estos elementos de tal manera que dependan en una forma mínima de otros componentes para así poder transportarlos o migrarlos a otros sistemas. Para lograr este objetivo, pretendo dividir en módulos las funcionalidades macros de todo sistema, es decir: - Identificar la interfaz que va a usar (*windows form, aplicación web, interfaz pda, web service o cualquier forma de presentación del sistema*), - Identificar un módulo en la que se resuelvan las funcionalidades del sistema; dicho de otra manera, donde que se establezca la lógica del negocio, - Considerar también en la actualidad siempre el acceso a datos ya sea archivos, XML, o DBA. La propuesta es realizar clases, métodos o elementos independientes con parámetros generales para que no dependan uno del otro y esto

		nos facilite el poder utilizar un elemento muchas veces dentro del mismo sistema o poderlo transportar a cualquier otra aplicación que necesite un elemento parecido.
Acoplamiento	A pesar de que todo sistema antes de ser diseñado cumple con los pasos básicos del análisis, diseño, implementación y pruebas, cuando ya se encuentra en fase de producción, generalmente podemos detectar varios ajustes que se deben hacer al mismo. Muchas de las veces estos ajustes derivan en la modificación o el remplazo de toda una parte del sistema. Por la forma en la que se estructuraba la aplicación, cada parte del sistema se la realizaba con el objetivo de interrelacionarse con otras, con lo que podemos concluir que existía un nivel de acoplamiento muy alto, dificultando el remplazo o la modificación de las partes afectadas del sistema. De lo analizado anteriormente, también se puede decir que si existe un acoplamiento demasiado fuerte, también será complicado hablar de reusabilidad.	De las experiencias en programación de aplicaciones que no contemplan una arquitectura, puedo mencionar que se ha venido construyendo interfaces como formularios, los mismos que en su interior guardan los procedimientos de la lógica de negocio, los procedimientos de acceso a datos y procedimientos de control. Esta forma de programación exige un alto nivel de acoplamiento, ya que si se desea remplazar un formulario, se lo hace con todos los componentes internos y esto definitivamente influye en la estabilidad de la aplicación. La propuesta es minimizar el nivel de acoplamiento identificando las partes o módulos críticos (*la interfaz, lógica y acceso*) y que los métodos de programación se distribuyan en ellos.
Disponibilidad	Técnicamente, la disponibilidad es el tiempo en el que el sistema es funcional y trabaja sin ningún problema. Pero viéndolo desde otro punto de vista, también se podía tomar como parámetro de	De lo que se ha venido exponiendo, la investigación pretende concentrar las tareas críticas en módulos claramente definidos, los componentes que pertenezcan a estos módulos deben ser

	disponibilidad, la cantidad de tiempo que un sistema permanece caído diferenciando también el tiempo en el que los desarrolladores o encargados del sistema pueden restaurar la funcionalidad de la aplicación. Al tener aplicaciones que estructuralmente no se rigen a una arquitectura, se torna muy complejo el detectar las causas por las cuales la aplicación ha dejado de funcionar y por ende corregirlo. Inclusive al detectar los errores y efectuar los cambios que se deben efectuar para corregir, necesariamente se debe suspender el funcionamiento de todo el sistema, ya que como lo hemos mencionado, al no tener una arquitectura, los componentes o elementos se encuentran dispersos dentro de toda la aplicación.	identificados de tal manera que sea fácil su localización y posteriormente su corrección o actualización. Pretendemos con eso tener un sistema que bordee el 100% de disponibilidad ya que con la nueva arquitectura el objetivo que se quiere alcanzar es que no pare el sistema a pesar de que se tenga que hacer correcciones. La disponibilidad, aparte de ser un factor con mucha influencia externa (*HW, cableado, sistemas eléctricos, sistemas de telecomunicaciones, etc.*) sigue siendo importante para el trabajo diario de toda empresa o institución.
Escalabilidad	En la manera en que se ha venido programando, ha sido muy complejo el hablar de escalabilidad ya que como sabemos, es la capacidad de que un sistema pueda crecer sin desmejorar su rendimiento y además sin afectar a los componentes o subsistemas ya existentes. Los atributos funcionales de un sistema pueden variar en el tiempo de acuerdo a las necesidades o a nuevos requerimientos que se vayan presentando. Para incrementar las nuevas características, se ha tenido que hacer una revisión a casi el 70% del sistema para establecer todo lo que afectaría el incrementar un nuevo módulo para satisfacer dichas necesidades.	Tomando en cuenta los parámetros que definen un sistema escalable, la presente investigación determina que para que el sistema amplíe sus servicios sin aminorar su rendimiento, la mejor opción es tener sus componentes distribuidos en varias partes de hardware y con la posibilidad de que se incremente nuevas funcionalidades no como un subsistema completo, sino más bien como un conjunto de partes que pueden resolver un problema. Lo que se pretende es dividir interfaz, lógica de negocio, acceso a datos, como partes globales del sistema tomando en cuenta también el

	Todo se lo ha realizado en forma secuencial, es decir, los formularios, procesos o funciones, están estrechamente relacionados y además enfocados a resolver un problema específico, lo que implica que en muchas ocasiones, estos componentes no se los puede relacionar con otros de similares características. En este ambiente, para que el sistema pueda ser escalable, se debe programar completamente el módulo a agregar.	modelo de capas. Esto nos posibilita el ir incrementando partes completamente independientes sin que afecten ni dependan de los módulos diseñados anteriormente. Al decir que cada componente tiene sus propios recursos, podemos establecer que al agregar nuevas partes, no influirán en lo más mínimo al rendimiento y por lo tanto nos posibilita escalar de una manera ordenada dentro del sistema.
Rendimiento	Todos los procesos que se han ido construyendo dentro del sistema para cubrir las necesidades informáticas de la Fundación, tienen integrados todos los componentes en un solo archivo ejecutable. Esto hace que el sistema deba correr completamente en el mismo computador. La complejidad que presenta el sistema sin base arquitectónica, determina el rendimiento del mismo ya sea por la estructura de datos que utiliza o la forma de realizar los diferentes cálculos para resolver sus funciones específicas. Esto mejoró en un inicio cuando se separó los procesos de la interfaz conocido como el Front-End y el Back-End, pero lo único que se hizo con esta estrategia fue separar la parte gráfica de la interfaz, pero los procesos siguen siendo integrados como un solo componente, lo que igual influye muchas veces en el buen rendimiento del sistema.	Con la investigación se pretende proponer un modelo de arquitectura que permita mejorar el rendimiento de la aplicación. Del análisis de las experiencias previas, un sistema mejoraría notablemente si distribuimos y organizamos sus componentes, para ello se toma como base el modelo estructural de capas. Creemos también que no basta con separar un sistema en su Front-End y Back-End sino más bien, dividir en partes que puedan funcionar en diferentes computadoras o servidores, logrando así que el rendimiento mejore notablemente ya que vamos a tener más infraestructura de hardware al servicio del nuevo sistema creado.

6.5. Modelo de arquitectura propuesto.

Para poder construir una arquitectura es primordial analizar algunas de las características de arquitecturas existentes (*Ver resumen en tablas 4 y 5 – ítem 2.2.5*), y de acuerdo a esto, construir una arquitectura que abarque las fortalezas de cada una de ellas. Se priorizan, aquellas fortalezas que puedan cumplir con las necesidades presentadas en la producción de aplicaciones. Estas necesidades fueron analizadas a través de un estudio de campo efectuado tanto a usuarios como a los desarrolladores (*ver resultados en anexo 3 – literales a y b*).

El modelo de arquitectura que se plantea en esta investigación, amplía las funcionalidades de la arquitectura en capas, con el objetivo primordial de poder concebir una plataforma cómoda para el diseñador de aplicaciones de la fundación. La arquitectura propuesta está constituida de la siguiente estructura:

- Módulo de Interfaz de Usuario,
- Módulo de Procesos del Sistema,
- Módulo de Acceso a Datos,
- Módulo de Comunicaciones,
- Módulo de Seguridad.

En la ***figura21***, se muestra de forma gráfica la estructura del modelo de arquitectura propuesto, definido anteriormente.

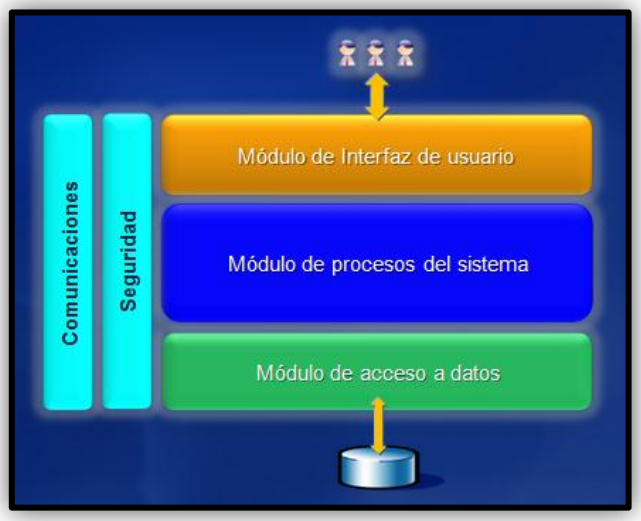

Figura 21. Estructura del modelo de arquitectura propuesto.
Fuente: investigador.

Esta arquitectura está construida con diversos tipos de componentes lógicos, cada uno de los cuales cumplen con una función diferente dentro de la aplicación. Cada uno de los módulos propuestos, se encargan de realizar una separación lógica del software; esto quiere decir, una separación básica de preocupaciones en el nivel del desarrollador, de modo que se pueden dividir más fácilmente las responsabilidades con respecto al sistema. Esto se detalla con mayor extensión en la ***figura22***, en donde el uso de módulos ayuda a estructurar aplicaciones que pueden descomponerse en grupos de sub-tareas, cada uno de los cuales se encuentra en un nivel de abstracción determinado.

Dicho de otro modo, es una separación de responsabilidades; se dividen las diversas tareas del sistema (*la recuperación de datos, el almacenamiento de datos, la ejecución de reglas con esos datos, la visualización de datos, la recopilación de entradas, etc*.) en componentes o sub-secciones, de modo que podamos realizar con mayor facilidad un seguimiento de lo que ocurre, dónde y cuándo.

En la ***figura22***, se detalla la división lógica del modelo de arquitectura propuesto, en donde se puede visualizar las sub-tareas que cumplen cada uno de los módulos.

Figura 22. División lógica del modelo de arquitectura propuesto.
Fuente: investigador.

Como se puede observar, cada componente lógico agrupa un número de tipos de componentes discretos (*sub-módulos*), las mismas que cumplen una sub-tarea específica. La estructura de esta división lógica se implementó en su totalidad en la plataforma Microsoft .NET (*ver figura 23*).

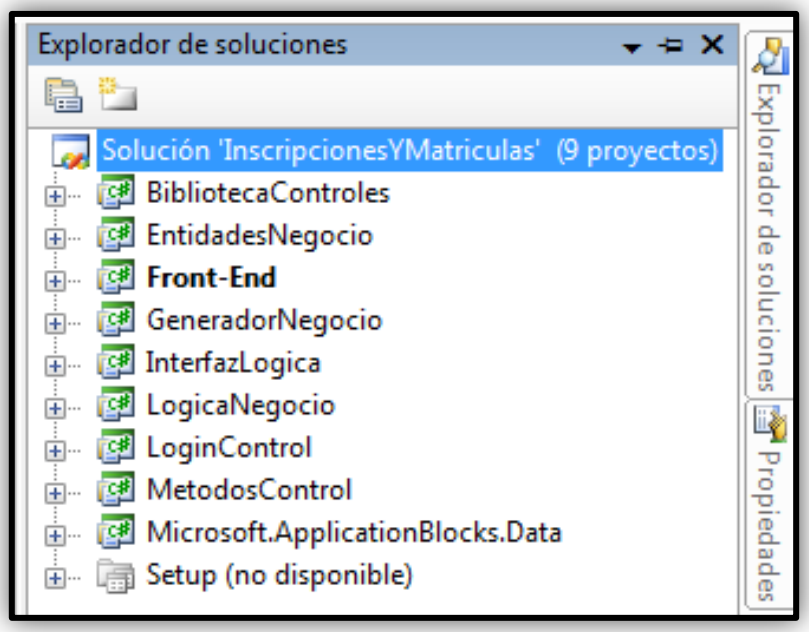

Figura 23. División lógica del modelo de arquitectura propuesto, diseñado en .NET.
Fuente: investigador

En este entorno, se puede observar que cada módulo es una agrupación lógica de los componentes de software que conforman la aplicación. Esta estructura muestra un diseño coherente y proporciona para el soporte de aplicaciones, las siguientes ventajas:

- Divide la aplicación en componentes que ofrecen servicios de presentación, lógica de negocios, datos, comunicación y seguridad, facilitando de esta manera la migración de las mismas y el mantenimiento de código.
- Está diseñado generalmente para comunicarse entre sí con el mínimo grado de acoplamiento.
- Ayuda a diferenciar entre los distintos tipos de tareas que realizan los componentes, facilitando el diseño de la reutilización en la solución.
- Brinda la posibilidad de trabajar en una forma de programación multilenguaje.

- Permite el diseño de aplicaciones, cuya funcionalidad no solo trabaje bajo interfaces Windows, sino también para aplicaciones web, sin realizar modificaciones en la lógica de negocios y de acceso a datos.

Los detalles de organización, programación y funcionamiento de cada módulo y su división lógica, se detallan a continuación:

6.5.1. Módulo de Interfaz de usuario.

El módulo de interfaz de usuario, es el lugar en donde se ubicarán los elementos encargados de permitir interactuar al sistema con diferentes usuarios, entendiéndose como usuarios a personas, sistemas o procesos. Estos elementos son los encargados de entender las solicitudes activadas por parte del usuario. Aquí se procesa y da formato a los datos de los usuarios, así como se adquiere y valida los datos entrantes procedentes de éstos.

La arquitectura propone que este módulo este compuesto por 3 elementos principales. Con esto se pretende que los elementos aquí constituidos, se los estandarice para que se pueda construir no solo interfaces tradicionales como WindowsForms o páginas ASP, sino interfaces que interactúen con cualquier tipo de usuario que desee utilizar el sistema.

Al tener separado el módulo de interfaz de usuario, aseguramos la independencia con el siguiente módulo que es el de procesos, dicha independencia facilita que todo lo que se cree para la funcionalidad del sistema no dependa de la interfaz que en su momento se decida utilizar.

En la arquitectura propuesta, este módulo va a estar constituido de tres partes:

Figura 24. Partes del módulo de interfaz de usuario.
Fuente: investigador.

a) Front–end

De forma general, el front-end hace referencia al estado inicial de un proceso. Contrasta con back-end, que se refiere al estado final del proceso. La idea general es que el front-end va a ser la parte del sistema de software que interactúa directamente con el usuario. Aquí se ubicarán los diferentes formularios diseñados para recoger la entrada de datos, para que luego toda la información recogida pueda ser procesada de tal manera que cumplan las especificaciones para que el back-end pueda usarlas y visualizar toda la información requerida.

Además, el front-end es el que decide en su momento el tipo de interfaz que va a interactuar directamente con el usuario que puede ser una interfaz de tipo Windows Forms, formularios web de Microsoft ASP.Net, PDA, controles y otro tipo de tecnologías para el envío o recepción de mensajes como se los realiza en los web service. Estas interfaces permitirán:

- Adquirir y validar los datos entrantes procedentes de los usuarios,
- Dar formato a los datos a exponer a los usuarios,
- Interpretar los eventos generados por el usuario para actuar con los datos.

Cuando un usuario interactúa con un elemento de la interfaz, se genera un evento que llama al código de una función de control. Ésta, a su vez a componentes lógicos de acceso a datos o componentes de proceso de usuario para implementar la acción deseada y recuperar los datos que se han de mostrar. A continuación, la función de control actualiza los elementos de la interfaz. Los controles insertados en cada formulario, contendrán el código necesario para capturar la información ingresada por el usuario y que servirá de base para la realización de las tareas dentro del sistema.

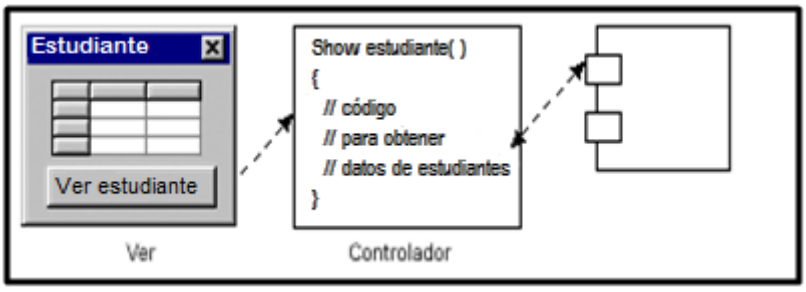

Figura 25. Funcionamiento del Front-End.
Fuente: investigador.

Es posible que la aplicación requiera usar el mismo proceso de usuario para visualizar información desde una interfaz de usuario basada en basada en Windows Forms así como de una interfaz basada en Web. Entonces, la separación entre la interactuación del usuario y la funcionalidad, en componentes de interfaz y proceso de usuario, permite que varias interfaces de usuario puedan utilizar el mismo proceso.

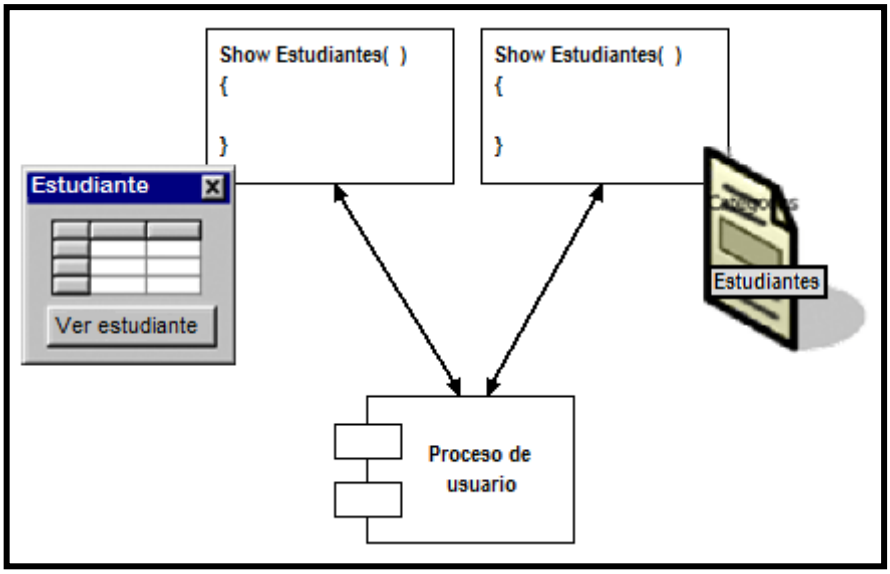

Figura 26. Resultado de procesos mostrado en diferentes interfaces.
Fuente: investigador.

A la hora de tomar la decisión de diseñar en una interfaz, se debe tener en cuenta las ventajas e inconvenientes derivados de insertar un grupo de controles en el formulario. Entre estas ventajas e inconvenientes se encuentran:

- **Tiempo de carga**: cuando se carga un proyecto, se crea una instancia de todos los formularios del proyecto. Cuando se tiene muchos formularios y estos varios controles, tardarán más en cargarse que las que tengan una menor cantidad de elementos.
- **Tiempo de exploración**: es más rápido desplazarse de un control a otro del mismo formulario que cargar un formulario nuevo. Esto ocurre aunque el formulario nuevo contenga menos controles.

En la aplicación implementada para la demostración de la arquitectura, las interfaces que se diseñaron en este elemento son del tipo WindowsForm y CrystalReports.

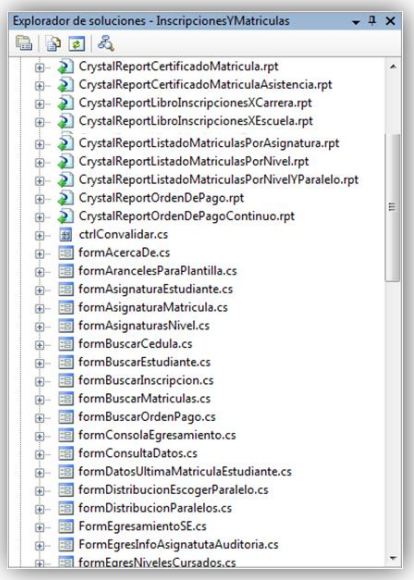

Figura 27. Lista de elementos diseñados en el front-end.
Fuente: investigador

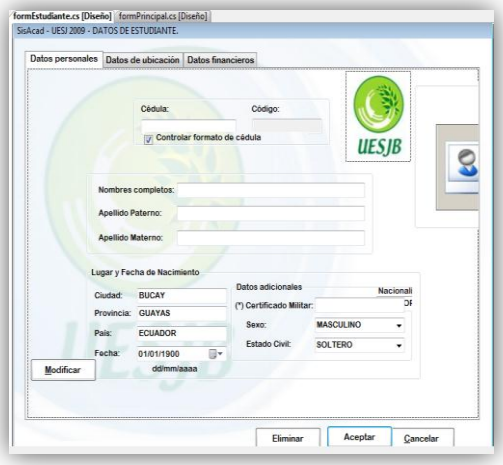

Figura 28. Ejemplo de WindowsForms.
Fuente: investigador.

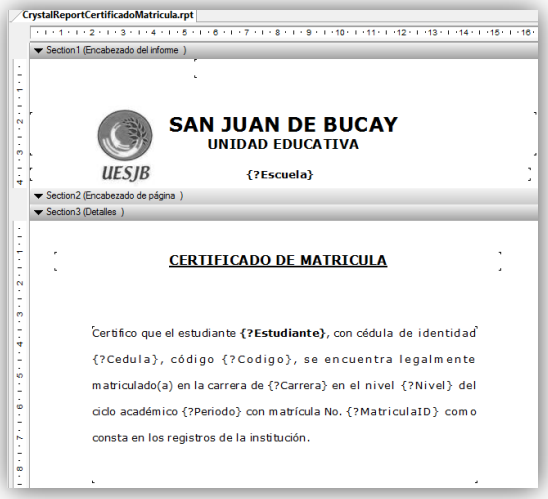

Figura 29. Ejemplo de CrystalReports.
Fuente: investigador.

b) Biblioteca de control

Dependiendo de la interfaz que se decida utilizar en el front-end, se irán creando diferentes controles apropiados para el trabajo que desempeña el usuario. La tendencia de realizar controles personalizados es con la finalidad de que estos sean independientes del sistema en el que se encuentran. Al lograr esta independencia dichos controles personalizados podrán ser usados en cualquier otro formulario o aplicación de las mismas características. Es por este motivo que a este módulo se lo ha denominado “biblioteca de controles”.

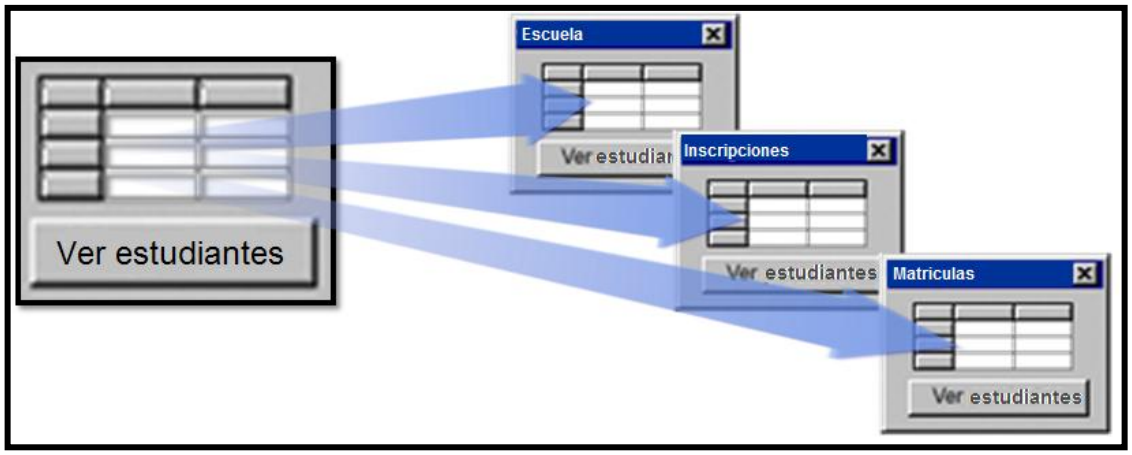

Figura 30. Biblioteca de controles.
Fuente: investigador.

Esta biblioteca tiene la posibilidad de seguirse incrementando para que los controles aquí diseñados, se los pueda utilizar en la construcción de otra(*s*) aplicación(*es*) y de esta manera disminuir el esfuerzo en la creación de controles que realizan actividades completamente similares. Cada nuevo control, genérico y único, podrá servir de contenedor de varios controles adicionales.

Por ejemplo, en la aplicación implementada para la demostración de la arquitectura, uno de los controles diseñados en la biblioteca es el “loginControl.cs” encargado de recoger la información referente la login y password del usuario. Este control es independiente del sistema diseñado y se lo puede utilizar en cualquier formulario que requiera de esta funcionalidad que es muy común.

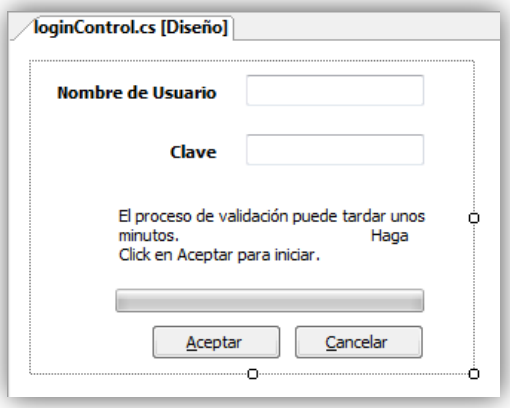

Figura 31. Ejemplo de control almacenado en la biblioteca de controles.
Fuente: investigador.

c) Generador de negocio.

Una de las características de esta arquitectura es la de desacoplar el código procedimental de la interfaz de usuario. Es decir que en esta arquitectura, el flujo del proceso y la lógica de negocios no se incluyen en el código de los elementos de la interfaz de usuario.

Al realizarse esta división de tareas, el módulo de interfaz se independiza de los demás módulos y se requiere de un factor que facilite la sincronización y organización de las interactuaciones del usuario con la funcionalidad del sistema.

Es entonces donde el generador de negocio cumple con la función de comunicar el módulo de interfaz de usuario con la lógica de negocios contenida en el módulo de procesos. Esto permite que varias interfaces puedan utilizar el mismo “motor” de interactuación básica.

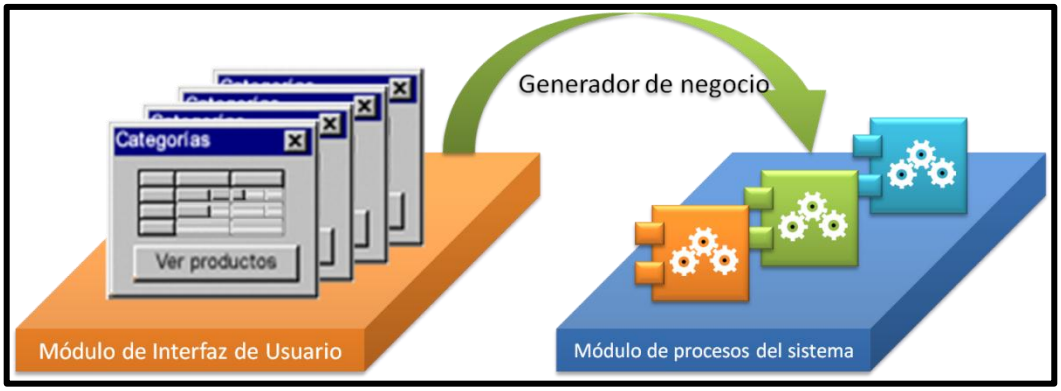

Figura 32. Generador de negocio.
Fuente: investigador.

Los elementos diseñados en este sub-módulo se implementan normalmente como clases .NET que “exponen” los métodos a los cuales pueden llamar las interfaces de usuario.

Este elemento contiene el código necesario para efectuar el proceso de comunicación entre ambas partes y de esta manera poder tener acceso a todas las interfaces lógicas que se implementen en el módulo de procesos del sistema. En la ***figura 33*** se puede observar parte del contenido del Generador de Negocio, en donde se crea una instancia del componente de la lógica de negocios y la utiliza para efectuar la transición a través de los pasos del proceso.

```
publicstaticIEstudiante Operaciones()
{
      returnnewLEstudiante();

}// Fin de Operaciones


publicstaticIUnidades DatosUnidades()
{
      returnnewLUnidades();

}//Fin de DatosUNidades
```

```
publicstaticIInscripciones OperacionesInscripcion()
{
      returnnewLInscripciones();

}//Fin de OperacionesInscripcion()


publicstaticIMatriculas OperacionesMatricula()
{
      returnnewLMatriculas();

}//Fin de OperacionesInscripcion()
```

Figura 33. Parte de código implementado en el Generador de negocio.
Fuente: investigador.

Lo que se hace es crear un método generador con un nombre que la identifique y que sea de tipo “interfaz”. Este método a través de codificación deberá acceder a la interfaz lógica y a la lógica de negocios del módulo de procesos del sistema, constituyéndose así en el nexo entre estos dos módulos. En resumen, este proceso permitirá utilizar los métodos y funciones diseñados en la lógica de negocios, creándose así un nuevo objeto que herede las características de la clase.

6.5.2. Módulo de procesos del sistema.

La finalidad de crear un proyecto de software es que cumpla con las funcionalidades específicas para las cuales se va a diseñar la nueva aplicación. Una aplicación realiza un proceso de negocios que consta de una o varias tareas. En los casos más simples, cada tarea se puede encapsular en un método de un componente .Net. En casos más complejos, se puede diseñar la lógica en sub-módulos de negocios, en donde parte de los métodos están encapsulados en componentes como servicio y son llamados a través de una interfaz lógica, que coordina la conversación con los llamadores del servicio.

Es por eso que luego de efectuar el análisis de las diferentes arquitecturas, se ha considerado que todos los procesos que permitan realizar y resolver las funciones específicas de la aplicación, deben estar separadas de cualquier otro módulo del sistema y eso es lo que representa esta arquitectura. Entonces, el módulo de procesos

del sistema, es el módulo independiente que constituye la base fundamental de la programación dentro del sistema.

El alcance de trabajar independientemente con diversos procesos, se logrará también hacer que la implementación del proyecto se convierta en multipersonal y multilenguaje, aprovechando los conocimientos de las diferentes personas que pueden conformar el equipo de programación.

En este módulo se ha considerado que debe estar formado por dos elementos: la interfaz lógica y la lógica de negocios.

Figura 34. Partes del módulo de procesos del sistema.
Fuente: investigador.

a) Interfaz lógica

Dentro del módulo de procesos, para poder resolver las funcionalidades de la aplicación, se deberán crear varios métodos. Estos métodos deben ser visibles de alguna manera para el módulo de interfaz de usuario y además deben ser cargados en memoria de las pc's de cada usuario para que la interfaz las pueda utilizar.

Es verdad que en la actualidad las computadoras tienen mucho poder de hardware y además los pc's tienen un mínimo de 2 Gb de memoria RAM, pero a pesar de eso, mientras se pueda optimizar la utilización de dicha memoria, se seguirán buscando alternativas para que el funcionamiento de las aplicaciones sean lo más livianas posibles para la memoria RAM.

La arquitectura propuesta, trata de que esta cualidad se cumpla, evitando subir todo el código de los métodos de funcionalidad. Entonces, para tener un canal o mecanismo de comunicación y poder utilizar la lógica empresarial o lógica de negocios, es necesario crear interfaces de esta lógica que admitan los contratos de comunicación. Esta interfaz ayuda a describir la funcionalidad que ofrece cada uno

de los procesos programados así como la semántica de comunicación requerida para llamar al mismo.

Esto se conseguirá haciendo una “publicación” del listado de los métodos existentes con los atributos necesarios para su funcionamiento, es decir, que en vez de tener el código del método, solo se va a tener el nombre del mismo pero cumpliendo la funcionalidad para la que fue creada. Además, esto permitirá que los procesos sean cargados en la memoria de las pc’s de cada uno de los usuarios, para que la interfaz las pueda utilizar.

Figura 35. Interfaz lógica.
Fuente: investigador.

De cada elemento programado en lógica de negocio, se debe diseñar una interfaz lógica donde se publican los métodos.Esto es lo que se logra con la Interfaz Lógica de Negocio. A continuación se muestra una parte de la codificación contenida en la interfaz lógica llamada “IEstudiante”:

```
publicinterfaceIEstudiante
{
        int DevuelveIdEstudiante(string cedula);
        int DevuelveNumeroEstudiantes(string cedula);
        int DevuelveNumeroEstudiantesXEstudianteID(int estudianteID);
        void BuscarEstudiante(string cedula);
        void BuscarEstudianteSolicitud(string cedula);
        void BuscarEstudianteSolicitud(string cedula,string fecha);
        void ActualizarEstudiante();
        void ActualizarEstudianteTitulos();
        void ActualizarEstudianteSolicitud();
        void InsertarEstudiante();
        void InsertarFilaEstudianteDataset(paramsobject[] parametros);
        void InsertarTitulo();
        void InsertarFilaTitulosDataset(paramsobject[] parametros);
        void InsertarSolicitud();
```

```
        void InsertarFilaSolicitudDataset(paramsobject[] parametros);
        void LlenarIntitucionesEducativas();

//*************Controles para estudiante
        int DevuelveNumeroEstudiantesEnOrden(int estudianteID);
        int DevuelveNumeroTitulosInscripciones(int tituloID);
        int DevuelveNumeroTitulosMatriculas(int tituloID);

//************Para foto*****************
        void ActualizarEstudianteFoto(int estudianteID,string fotoRuta,int
usuarioID);
        void LlenarDatosEstudianteXCedula(string cedula);

//************Para Buscar estudiante*****************
        void LlenarBuscarEstudianteXEstudianteID(int estudianteID);
        void LlenarBuscarEstudianteXCedula(string cedula);
        void LlenarBuscarEstudianteXCaracteres(string caracteres);

//******************Genera Claves para estudiantes
        string ObtenerClaveEstudiante(int estudianteID);

}
```

Figura 36. Parte de código implementado en la Interfaz lógica.
Fuente: investigador.

b) Lógica de negocios

Los componentes empresariales constituyen la raíz de la solución. Aquí se dará respuesta a todas las necesidades requeridas por los stakeholders y es donde se organizará toda la lógica de programación del sistema o aplicación, independientemente de si el proceso consta de un único paso o de un flujo de trabajo organizado.

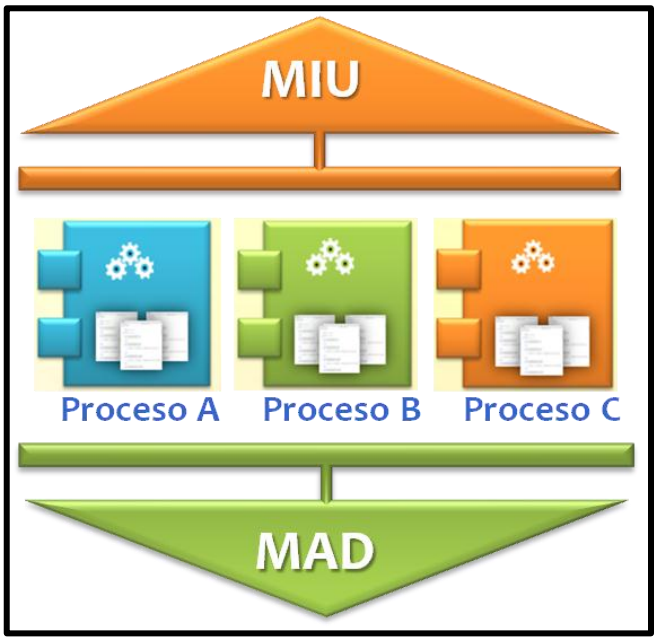

Figura 37. Lógica de Negocios.
Fuente: investigador.

Prácticamente es el espacio donde toda la parte procedimental será implementada, en donde las diferentes clases estarán agrupadas de acuerdo al proceso que van a resolver para cubrir todas las necesidades requeridas por los stakeholders.

Por la facilidad del lenguaje estos métodos podrían ser diseñados en diferente lenguaje ya que la lógica de negocio no es unitaria, sino más bien se podrían crear varias clases de lógica de negocio. Al final lo que determina la arquitectura es que cualquier método o función que se diseñe, sea publicado en el elemento que llamado “Interfaz Lógica de Negocio”.

Aquí se debe exponer su funcionalidad de modo que sea independiente de los almacenes de datos y de los elementos de interfaz necesarios para realizar las diferentes tareas. Para la construcción de la lógica empresarial, se debe prestar especial atención a la evolución de las transacciones

Para seguir con los ejemplos señalados anteriormente, a continuación (*ver figura 38*) se muestra una porción de código llamado a cumplir con tareas específicas de un estudiante. Esto está diseñado en la lógica de negocios “LEstudiante”:

```
# region Buscar Estudiante Nombres Solicitud, Cedula , Fecha Inicio, Fecha Fin

publicvoid BuscarEstudianteSolicitud(string cedula,string fechaI,string fechaF)
{
        EntidadesEstudiante.ConjuntoEstudianteSolicitud = newDatasetEstudianteSolicitud();

//************* Llena la propiedad dataset ConjuntostudianteSolicitud con los datos del nombre estudiante

        SqlHelper.FillDataset(ConnectionString,"spConsultaEstudianteNombre",EntidadesEstudiante.ConjuntoEstudia
nteSolicitud ,newstring[]{"VISTA_ESTUDIANTE_NOMBRE"},cedula);

//*****************************************************************************
// Para Buscar ID del estudiante

        int estudianteId=DevuelveIdEstudianteSolicitud(cedula);

//*******************************

// Para LLenar la propiedad datased ConjuntostudianteSolicitud con los datos de las solicitudes

        SqlHelper.FillDataset(ConnectionString,"spConsultaSolicitudEstudiantePorPeriodo",EntidadesEstudiante.Conju
ntoEstudianteSolicitud ,newstring[]{"ESTUDIANTE_SOLICITUD"},estudianteId,fechaI,fechaF);

// ************************************************************


}// Fin de BuscarEstudianteSolicitud
#endregion

#region  InsertarEstudiante

publicvoid InsertarEstudiante()
{
```

```
//Para crea los parámetros para llamar a Actualizar dataset
        string nombreTabla ="ESTUDIANTE";
        string procedimiento="spInsertarEstudiante";
        string [] columnasTabla =newstring[36]{   "EstudianteID","Cedula", "Nombre", "ApellidoPaterno",
                                "ApellidoMaterno","CiudadNacimiento","ProvinciaNacimiento","PaisNacimiento",
                                "FechaNacimiento","Nacionalidad","CertificadoMilitar","Sexo","EstadoCivil",
                                "Direccion","Telefono1","Telefono2","LugarTrabajo","TelefonoTrabajo","Email",
                                "FinanciamientoEstudios","Observaciones","FotoRuta","UsuarioID","FechaTransaccion",
                                "DireccionProvincia","DireccionCanton","DireccionOrigen","TrabajoLugar","TrabajoCargo
                                ","OcupacionPadre",
                                "OcupacionMadre","IngresosFamiliares","DomicilioEstado","NumeroIntegrantesHogar","
                                ActividadDeportiva","ActividadCultural"};


        SqlHelper.ActualizarDataset(ConnectionString,EntidadesEstudiante.ConjuntoEstudianteTitulos,nombreTabla,c
olumnasTabla,procedimiento);
        EntidadesEstudiante.ConjuntoEstudianteTitulos.ESTUDIANTE.AcceptChanges();
}// Fin de InsertarEstudianteDataset
#endregion
```

Figura 38. Parte de código implementado en la Lógica de Negocios.
Fuente: investigador.

6.5.3. Módulo de acceso a datos.

En vista de que en la actualidad, la tendencia es el interactuar con la información, ya sean bases de datos, archivos, mensajes, etc., en la arquitectura se propone necesariamente un módulo de acceso a datos. Todas las aplicaciones comerciales o de servicios, necesitan almacenar y obtener acceso a un determinado tipo de datos. La aplicación o el proceso en sí, puede disponer de uno o varios orígenes de datos, los cuales pueden ser heterogéneos. La lógica utilizada para obtener acceso a los datos de un origen determinado, se encapsulará en componentes lógicos de acceso a datos que proporcionan los métodos necesarios para la consulta y actualización de datos.

Se debe considerar que la mayoría de las aplicaciones utilizan una base de datos relacional como almacén principal de los datos de la aplicación. Las bases de datos relacionales, como las bases de datos SQL Server, proporcionan funcionalidad de administración de un gran volumen de datos transaccionales de alto rendimiento con capacidades de seguridad, operaciones y transformación de datos.

Anteriormente, cada instante que se requería que la aplicación acceda a la información almacenada en el repositorio de datos, se debía ejecutar los comandos específicos para la conexión y posteriormente para la consulta, inserción, modificación o eliminación de alguna parte de la información.

La arquitectura propone que no se tenga código de consultas o acceso a datos regado por todos los módulos de la aplicación, sino más bien estandarizar el acceso a la información y todo lo que se refiera a manejo de datos, agruparlos en este módulo.

Figura 39. Módulo de acceso a datos.
Fuente: investigador.

Estas funcionalidades pueden alojar instrucciones y funciones complejas de lógica de datos en forma de procedimientos almacenados *(códigos de programación de las bases de datos)*.

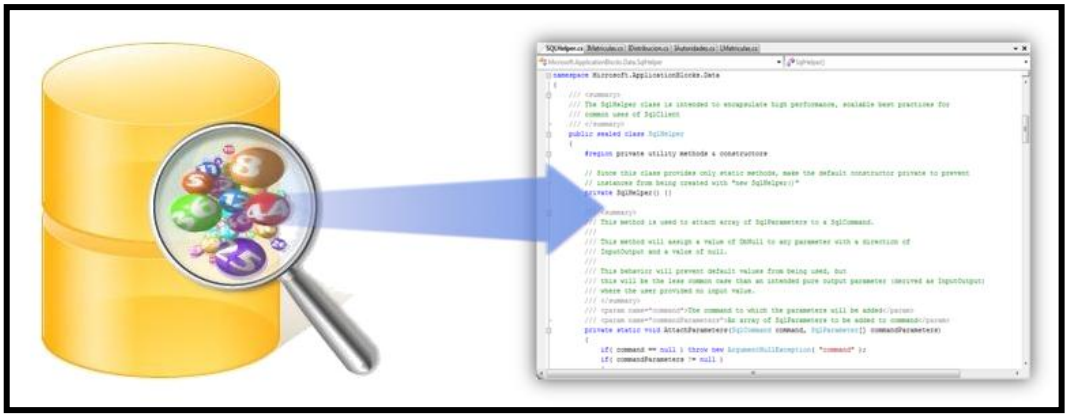

Figura 40. Funcionamiento del módulo de acceso a datos.
Fuente: investigador.

Este módulo de acceso a datos, a más de constituir una porción de código que se encarga de la interacción con la base de datos, permitirá que en el momento que sea necesario cambiar el motor de base de datos, solamente se tenga que corregir este módulo.

El módulo de acceso a datos, no se encuentra íntegramente en el nivel de base de datos, dado que los comandos necesarios para el acceso y la manipulación de los datos (*es decir, SQL*) deben generarse y enviarse desde el exterior del nivel de base de datos.

La implementación de una lógica propia de acceso a datos permite encapsular toda la lógica de acceso a datos de la aplicación completa en una única ubicación central, lo que facilita el mantenimiento y la extensibilidad de la aplicación.
Es decir, que un método creado para acceder al repositorio de datos será estándar y podrá ser utilizado por el módulo que lo necesite.

```
#region FillDataset
///<summary>
/// Execute a SqlCommand (that returns a resultset and takes no parameters) against the database specified
in
/// the connection string.
///</summary>
///<remarks>
/// e.g.:
///  FillDataset(connString, CommandType.StoredProcedure, "GetOrders", ds, new string[] {"orders"});
///</remarks>
///<param name="connectionString">A valid connection string for a SqlConnection</param>
///<param name="commandType">The CommandType (stored procedure, text, etc.)</param>
///<param name="commandText">The stored procedure name or T-SQL command</param>
///<param name="dataSet">A dataset wich will contain the resultset generated by the command</param>
///<param name="tableNames">This array will be used to create table mappings allowing the DataTables to
be
Referenced
/// by a user defined name (probably the actual table name)</param>

publicstaticvoid FillDataset(string connectionString, CommandType commandType, string commandText,
DataSet dataSet, string[] tableNames)
{
        if( connectionString == null || connectionString.Length == 0 ) thrownewArgumentNullException(
"connectionString" );

        if( dataSet == null ) thrownewArgumentNullException( "dataSet" );

// Create & open a SqlConnection, and dispose of it after we are done

using (SqlConnection connection = newSqlConnection(connectionString))
{
    connection.Open();

// Call the overload that takes a connection in place of the connection string
    FillDataset(connection, commandType, commandText, dataSet, tableNames);
}
 }
```

Figura 41. Parte de código implementado en el módulo de acceso a datos.
Fuente: investigador.

El ejemplo anterior (***Figura 41***), es una porción de código que se encuentra a nivel de módulo de acceso a datos. Este ejemplo, ayuda a llenar cualquier dataset. La estandarización se identifica claramente ya que nosotros solo debemos llenar los parámetros tales como a que base de datos se conecta, que procedimiento ejecuta y que dataset llena. En este caso, por las características del sistema, siempre se deben llenar datasets de diferentes nombres y diferentes estructuras. La utilización del módulo de acceso a datos significa mucho ahorro de código en la arquitectura que se propone.

Hasta el momento se ha explicado la estructura básica de la arquitectura y de acuerdo a la estructuración de los módulos se puede decir que esta nueva propuesta se basa en la arquitectura de capas. Dicha estructura tiene como característica el usar a las capas con métodos de llamada y de retorno, es decir que la comunicación se lo hace en forma vertical la única con la otra sin poder saltar ninguna de las capas intermedias.

Debido a que la aplicación requiere realizar la administración de excepciones, autorizar a los usuarios a que realicen tareas determinadas y comunicarse con otros procesos, se recurre al uso un conjunto de directivas de organización dispuestos en forma vertical, las mismas que definen las reglas que determinan la forma en que se protege una aplicación, el modo en que se administra, así como la forma en que los distintos componentes de una aplicación se comunican entre sí.

Por este motivo se ha implementado un bloque de organización que se encuentra dividido en 2 partes dispuestas en forma vertical: directiva de comunicación y directiva de seguridad.

6.5.4. Comunicaciones.

Una de las características que se pudo evidenciar en la construcción de aplicaciones es que existen muchos datos que se los carga y descarga de forma constante y siempre se ha mencionado el costo de recursos que significa cada vez que la aplicación accede a un repositorio de datos; recursos tales como tráfico de red, memoria y procesador del servidor de base de datos, lo que en horas pico significaría un gran problema de acceso.

Una vez identificado este inconveniente, la arquitectura propone tener un módulo de comunicaciones denominado “Entidades de negocio”, en el que se permita almacenar los datos o conjunto de datos que se utilizan con frecuencia o que no sean necesarios pasar de capa en capa sino más bien a un módulo específico. Este módulo permite levantar la información desde la base de datos, una sola vez a un dataset en el momento que arranca el sistema (*ver* ***figura 42***).

En efecto, una de las características particulares y que resalta la ventaja de la aplicación de este módulo es que, cualquiera de los módulos anteriormente analizados, pueden acceder a los datos en forma directa.

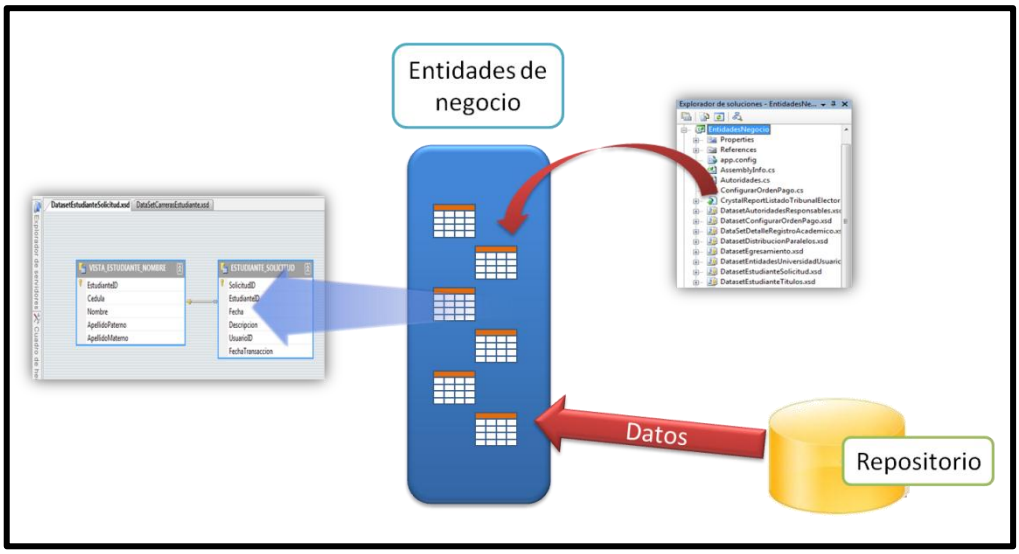

Figura 42. Entidades de negocio.
Fuente: investigador.

Para ver la funcionalidad de este módulo, se analiza el siguiente entorno: dados los requerimientos de los stakeholders, siempre o casi siempre se estará llamando a las unidades de estudio de la Fundación y esta acción se lo llevará a cabo muchas veces a lo largo del trabajo y en forma repetitiva. Con la implementación del módulo de comunicaciones, lo que se hace es levantar la información una sola vez al dataset correspondiente en el momento que arranca el sistema y luego simplemente se accede al dataset para la consulta de los datos. A esto se lo conoce también como el “trabajo sin conexión”.

En la ***figura 43*** se muestra uno de los dataset's creado en el módulo de comunicaciones. Este dataset se encargará de levantar la información requerida a cualquier módulo que lo solicite, sin necesidad de recurrir al repositorio de datos, descongestionando de esta manera el tráfico en la red.

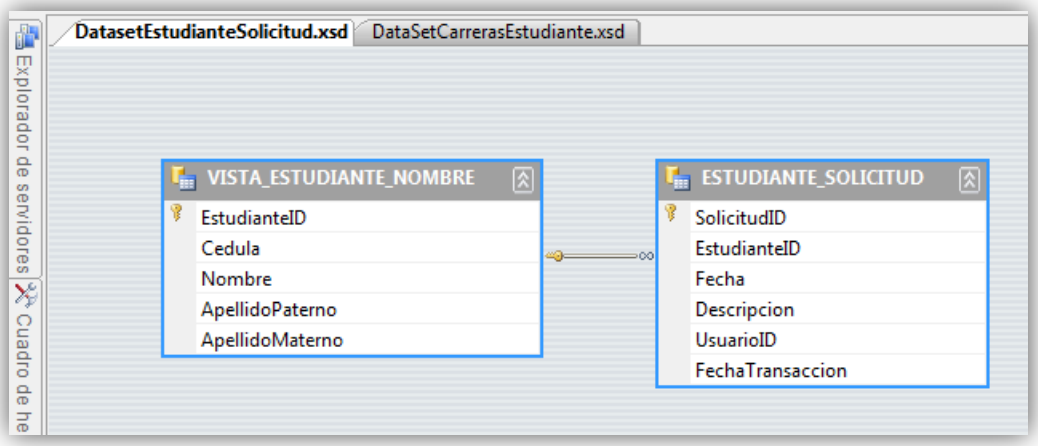

Figura 43. Ejemplo de dataset creado en el módulo de comunicaciones.
Fuente: investigador.

6.5.5. Seguridad.

Toda aplicación va a tener diversos niveles de seguridad, tales como: seguridades de red, bases de datos y aplicación. La arquitectura propone estandarizar todo lo que se refiere a la seguridad de aplicación.

Al hablar de seguridad de aplicación, se está refiriendo a niveles de acceso, a datos ingresados y a datos recibidos. La arquitectura implementa un módulo llamado de "Métodos de control", en el cual se incluirán todos los métodos que se encargarán del control global de la información que se manejará en cada uno de los módulos. Este módulo permite dar seguridad a los niveles de aplicación, por ejemplo, verificar que un valor numérico sea de tipo entero. Este método se lo puede requerir tanto en la interfaz, como en el módulo de procesos e inclusive en el módulo de acceso a datos. Al estandarizar y colocar este método de verificación en un módulo, se estará centralizando el uso del mismo para evitar la repetición de código.

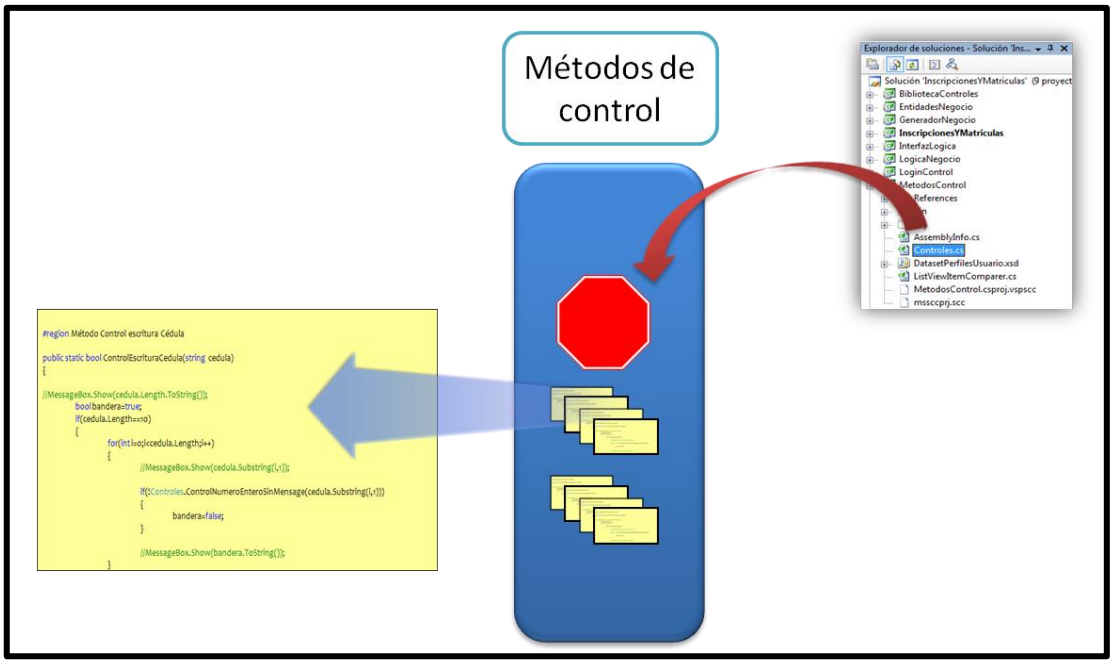

Figura 44. Métodos de control.
Fuente: investigador.

Si este módulo sigue creciendo con los diferentes métodos de seguridad y además como se lo trata de manera independiente, se lo podría utilizar en el diseño de cualquier otro proceso o sistema, ya que adquiere la facultad de poder encargarse del control global de la información que se manejará en cada uno de los módulos. A continuación se puede observar una porción de código que controla si el ingreso del número de cédula se lo efectuó de acuerdo a la sintaxis requerida por la aplicación. Este método puede ser requerido por cualquier módulo para el control de la información.

```
#region Método Control escritura Cédula

publicstaticbool ControlEscrituraCedula(string cedula)
{

//MessageBox.Show(cedula.Length.ToString());
        bool bandera=true;
        if(cedula.Length==10)
        {
                for(int i=0;i<cedula.Length;i++)
                {
                        //MessageBox.Show(cedula.Substring(i,1));

                        if(!Controles.ControlNumeroEnteroSinMensage(cedula.Substring(i,1)))
                        {
                                bandera=false;
                        }
```

```
                        //MessageBox.Show(bandera.ToString());
                }

        }

        else

        {
                bandera=false;
        }

        if(bandera==true)
        {
                returntrue;
        }
        else
        {
        MessageBox.Show("El formato de la cédula ingresada es incorrecto", "SisAcad - UESJ 2009",
        MessageBoxButtons.OK, MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);
                        returnfalse;
        }
}

#endregion
```

Figura 45. Porción de código implementado en el método de control.
Fuente: investigador.

CONCLUSIONES

- De la investigación realizada, se determina que las aplicaciones diseñadas en la Fundación San Luis, fueron elaboradas bajo una programación tradicional estructurada durante mucho tiempo, la misma que no permitía seguir una disposición efectiva, organizada, estandarizada y planificada, sino más bien que los procesos de diseño se han sustentado intuitivamente en base a la experiencia y conocimientos de sus hacedores, sin considerar los cambios que se pueden presentar en la vida del producto, los nuevos requerimientos de los stakeholders, el coste de la generación de los cambios evolutivos, la escalabilidad del producto en el transcurso del tiempo, la reusabilidad.

- Del análisis de las arquitecturas referenciales, se pudo establecer que muchas de ellas tienen una serie de componentes comunes e inevitables en la composición de su estructura. El uso de un modelo de arquitectura responde a las necesidades presentadas a nivel tanto de desarrolladores como de usuarios. Para su diseño e implementación es necesario verificar las características y potencialidades que brindan cada una de ellas y es importante acotar que a pesar de sus diferencias notables, todas tienen un objetivo en común, alcanzar productos de calidad. De este modo, son consideradas como una estrategia tecnológica esencial para el diseño de aplicaciones.

- El diseño y empleo de un modelo alternativo de arquitectura de software constituye el elemento básico en el desarrollo de soluciones informáticas, cuya efectividad se verá reflejada en la optimización de recursos, pues la estructura del modelo contiene módulos y componentes discretos que simplifican el trabajo de los profesionales encargados de la implementación de aplicaciones, permitiendo de esta manera un trabajo organizado, estandarizado, planificado y efectivo. Otro alcance importante que ofrece esta organización arquitectónica es la detección oportuna de errores y la flexibilidad que brinda al momento de trabajar en línea común.

- Para el modelado de una arquitectura existen varias perspectivas y dentro de éstas pueden mostrarse varios puntos de vista dependiendo de los involucrados en la toma de decisiones y de la prioridad de los requisitos de calidad que se desea alcanzar, tanto en el producto final como en los proceso de programación. La elección de una arquitectura adecuada, resulta fundamental para el éxito de un proyecto y por ende la adquisición general de una aplicación. Una arquitectura adecuada proporcionará el correcto equilibrio entre la experiencia del usuario, la facilidad de desarrollo por parte de los programadores, realización de pruebas y los requisitos operativos de la aplicación. Los componentes que vayan a formar parte de la arquitectura, deben ser los suficientes para garantizar la viabilidad del proyecto y, a su vez, los mínimos que permitan dar tales garantías con el mínimo gasto de tiempo, esfuerzo y recursos en general.

- El diseño de aplicaciones bajo una estructura arquitectónica estable, ha reportado significativos beneficios en los procesos de construcción del software. Ha permitido mejorar la interacción entre los métodos diseñados sin considerar la plataforma en los cuales fueron implementados, aumentando así la productividad del equipo desarrollador debido a que se puede trabajar de forma independiente sin necesidad de tener que dominar un determinado lenguaje de programación. Además, la reutilización de componentes ha significado grandes avances en los procesos de diseño en vista de que no es necesario conocer la forma en la que fueron diseñados determinados procesos ni el lenguaje utilizado; tan solo es necesario conocer su funcionalidad y la sintaxis requerida para su aplicación. Cabe indicar que cualquier cambio generado dentro de los módulos de la arquitectura es completamente transparente tanto para los integrantes del equipo desarrollador de aplicaciones como para los usuarios finales.

- Como resultado del uso de un modelo de arquitectura que se ajusta a las necesidades internas del equipo desarrollador de la Fundación, se ha podido comprobar que el trabajo diario es más organizado y ha permitido el crecimiento constante de la aplicación, la evaluación frecuente mediante demostraciones periódicas y la resolución de los riesgos más peligrosos en etapas tempranas de diseño. Sin embargo, siempre será importante profundizar su estudio y no perder de vista la evolución tecnológica que se vaya generando, ya que puede proveer beneficios tangibles como la reducción de esfuerzo requerido en la implementación de aplicaciones y la reducción del coste de la misma.

- El modelo de arquitectura y los módulos implementados dentro de ésta, no son de uso exclusivo; no ha sido estructurado únicamente para cubrir las necesidades empresariales de la Fundación, sino que se ha propuesto para ser extendido en el diseño de nuevas aplicaciones internas e incluso para el diseño de aplicaciones externas a la Fundación, que tengan las mismas características y necesidades que la implementada dentro de la institución.

RECOMENDACIONES

- Proveer de un modelo de arquitectura de software al equipo de desarrolladores de aplicaciones y motivar al uso de nuevas tecnologías. Esto permitirá que el trabajo diario se vea beneficiado en la parte estructural y organizativa. El acogerse a un modelo de arquitectura no sólo mejorará la interacción entre aplicaciones existentes, y el trabajo que conlleva su desarrollo e implementación sino que disminuirá notablemente en su coste: tiempo de trabajo, esfuerzo del equipo técnico y recursos.

- Se recomienda la utilización de una arquitectura cuyo diseño arquitectónico puede servir para varias aplicaciones distintas. Es necesario recalcar que un diseño arquitectónico facilita el desarrollo de nuevas aplicaciones y reduce el tiempo que se invierte en dicho proceso.

- Antes de dar inicio a la producción de software en cualquier área que se desee implementar su funcionalidad, será importante partir analizando los requerimientos no funcionales, como parte del proceso de educción de requerimientos de un producto informático. Una buena comunicación entre los *stakeholder*s, incluyendo clientes y desarrolladores permitirá facilitar el trabajo de priorizar el tratamiento de las diferencias de calidad que puedan surgir en el sistema a crear. Los resultados de este trabajo y la utilización del modelo de arquitectura propuesto, han sido de gran utilidad para los desarrolladores de sistemas de la Fundación, considerando que contribuye a mejorar el funcionamiento y la organización de los procesos de programación. Se considera además que su uso puede ser extendido para el desarrollo de nuevas aplicaciones.

- Es prioritario crear conciencia de aprovechar las ventajas de implementar aplicaciones bajo un esquema arquitectónico que se ajuste a sus necesidades empresariales. Esto brindará una diferenciación competitiva en el cumplimiento de los objetivos a largo plazo del negocio.

- Proseguir con la implementación de la propuesta será una alternativa válida para perfeccionar e incrementar las funcionalidades del modelo de arquitectura. Solo así se conseguirá que las aplicaciones diseñadas y las nuevas a implementar, tiendan a alcanzar la calidad esperada y se los genere optimizando la mayor cantidad de recursos humanos, tecnológicos, de tiempo y económicos.

GLOSARIO DE TÉRMINOS

A

Abstracción.
Consiste en aislar un elemento de su contexto del resto de los elementos que lo acompañan. En programación, el término se refiere al énfasis en el "¿qué hace?" más que en el "¿cómo lo hace?" (*Característica de caja negra*)

Aplicación.
Arquitectura de SW. Grupo de abstracciones y patrones que nos brindan un esquema de referencia útil para guiarnos en el desarrollo de software dentro de un sistema informático.

Algoritmo.
Es la secuencia finita, ordenada y no ambigua de instrucciones que resuelve determinado problema.

Atributo de calidad.
Propiedades con las que debe cumplir un producto software para calificarse como un producto de calidad.

Automatizar.
Aplicar procesos automáticos a un aparato, proceso o sistema.

B

Base de datos.
Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Biblioteca de clases.

Conjunto de subprogramas utilizados para desarrollar software. Contienen código y datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de éstos. Esto permite que el código y los datos se compartan y puedan modificarse de forma modular.

Bifurcación.

Acción de crear un proceso o proyecto en una dirección diferente a un proceso o proyecto ya existente; deja de hacer un proceso para comenzar otro hasta que este culmine para luego seguir en el anterior.

Blueprints.

Framework CSS (*Cascading Style Sheets*) diseñado para reducir los tiempos de desarrollo y mejorar la compatibilidad entre los distintos navegadores web cuando se trabaja con los estilos en cascada (*CSS*). También sirve como base para muchas herramientas que permiten el desarrollar CSS fácilmente y de una manera más accesible para principiantes.

Booleano.

Elemento cuyo valor puede representarse como verdadero o falso.

C

Cliente.

Elemento que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

Compilador.

Programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar.

E

Encapsular.

Proceso por el cual los datos que se deben enviar a través de una red se deben colocar en paquetes que se puedan administrar y rastrear.

F

Firewall.

Sistema que impone una política de seguridad entre la organización de red privada y el Internet. Determina cual de los servicios de red pueden ser accesados, es decir, determina quién puede entrar para utilizar los recursos de red.

Framework.

Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar. Es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado.

G

Granularidad.

Es una expresión de riqueza funcional del servicio. Por ejemplo, entre mayor la granulación, mas riqueza funcional ofrecida. Los servicios típicamente son funciones de negocio de grano grueso, como abrir la cuenta, puesto que la operación puede resultar en la ejecución de múltiples operaciones de grano fino, verificar identificación y crear cuenta.

M

Metasistema.

Todo aquello que se encuentra fuera de la frontera del sistema bajo estudio. Se denomina también entorno o medio ambiente.

Middleware.
Software de computadora que conecta componentes de software o aplicaciones para que puedan intercambiar datos entre éstas. Es utilizado a menudo para soportar aplicaciones distribuidas.

Motor de base de datos.
El Database Engine, es el servicio principal para almacenar, procesar y proteger datos. Proporciona acceso controlado y procesamiento de transacciones rápido para cumplir con los requisitos de las aplicaciones consumidoras de datos más exigentes de su empresa.

Multicast.
Envío de información en una red a múltiples destinos simultáneamente, usando la estrategia más eficiente para el envío de los mensajes sobre cada enlace de la red sólo una vez y creando copias cuando los enlaces en los destinos se dividen.

Multiplexación.
En informática es una forma que se mantengan varias copias idénticas de este archivo, esto para respaldar información en caso de que ocurra un fallo en el archivo principal.

Multiproceso.
Uso de múltiples procesos concurrentes en un sistema en lugar de un único proceso en un instante determinado.

N

Nodo.
Punto de intersección o unión de varios elementos que confluyen en un mismo lugar.

O

Objetos.
Unidades independientes que se comunican entre ellos mediante mensajes.

P

Plataforma.

Es el principio en el cual se constituye un hardware, sobre el cual un software puede ejecutarse/desarrollarse. Define un estándar alrededor del cual, un sistema puede ser desarrollado.

S

Servidor.

Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LANs o WANs, para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, etc.

Sincronización.

Se refiere a que dos o más elementos eventos u operaciones sean programadas para que ocurran en un momento predefinido de tiempo o lugar. Hace referencia a la coordinación de procesos que se ejecutan simultáneamente para completar una tarea, con el fin de obtener un orden de ejecución correcto y evitar así estados inesperados.

Sintaxis.

Conjunto de reglas que definen las secuencias correctas de los elementos de un lenguaje de programación.

Stakeholders.

Término inglés utilizado por primera vez por Freeman en su obra: "*Strategic Management: A Stakeholder Approach*", para referirse a "quienes pueden afectar o son afectados por las actividades de una empresa".

SIGLAS

AS	Arquitectura de software.
ADL´s	Lenguajes de Descripción Arquitectónica.
ADM	Método de Descripción Arquitectónica.
ADO	ActiveX Data Objects
API	Application prigramming interface.
ASP	Active Server Pages.
BCL	Biblioteca de clases.
BD	Base de datos.
BPEL4WS	Business Process Execution Language for Web Services.
CLI	Infraestructura de Lenguaje Común.
CLS	Common Language Specification.
CLR	Common Language Runtime.
CTS	Common Type System.
DLL	Biblioteca de enlace dinámico.
ECMA	European Computer Manufacturers Association.
FTP	File Transfer Protocol.
GPL	General Public License.
GUI	Interfaz gráfica de usuario.
HTML	Hypertext Markup Language.
HTTP	Hypertext Transfer Protocol.
HP	Hewlett-Packard.
HW	Hardware.
IDAPI	Independent DataBase Application Programming Interface.
IEC	International Electrotechnical Commission.
ISO	International Organization for Standardization.
IT	Tecnologías de la información.

JIT	Just-In-Time.
LAN	Local Area Network.
MSDN	Microsoft Developer Network.
MSIL	Microsoft Intermediate Language.
MFC	Microsoft Foundation Class Library
MVC	Model-View-Controler
ODBC	Open DataBase Connectivity.
OLE	Object Linking and Embedding.
PC	Computador personal.
POO	Programación Orientada a Objetos.
RM-ODP	Modelo de Referencia para Procesamiento Distribuido Abierto.
RUP	Rational Unified Process.
SGDB	Sistema de gestión de bases de datos.
SO	Sistema Operativo.
SOA	Arquitectura Orientada a Servicios.
SOAP	Simple Object Access Protocol.
SQL	Structured Query Language.
SW	Software.
UDDI	Universal Description, Discovery an Integration.
UML	Lenguaje de modelado unificado.
VB	Visual Basic.
W3C	World Wide Web Cosortium.
WAN	Wide Area Network.
WCF	Windows Communication Foundation.
WF	Workflow Foundation.
WSDL	Web Services Description Language.
WWW	World Wide Web.
XML	Extensible Markup Language.

BIBLIOGRAFIA

1. **BASS., L., Y OTROS.,** Software Architecture in Practice., 2a.ed., Massachusetts-E.U.A., Addison Wesley Publishing Company Inc.., 2003., Pp. 10-28

2. **BERTRAN., M.,** Significado de componentes., Canverra-Australia., CMP Books., 2000., Pp. 212-221

3. **BUSCHMANN., F., Y OTROS.,** Pattern-oriented software architecture., Munich-Germany., Sigs Datacom., 2008., Pp. 20-33

4. **CLEMENTS., P.,** A Survey of Architecture Description Languages – Proceedings of the International Workshop on Software Specification and Design., Berlin-Alemania., Ed. Springer., 1996., Pp. 97-144

5. **DE REMER., F., Y OTROS.,** Programming in the large versus programming in the small., 2a. ed., Paris-Francia., Ed. Jacky Estublier., 1995., Pp.80-86

6. **DIJKSTRA., E.,** The Structure of the Multiprogramming system., 2a.ed., Texas-E.U.A., Springer., 2001., 49-52

7. **FLORES., A.,** Introducción a SQL Server., Madrid- España., Forma Select Grupo Empresarial., 2006., Pp. 1-20

8. **GAMMA., E., Y OTROS.,** Design Patterns: Elements of reusable object oriented software., Massachusetts-E.U.A., Addison Wesley Publishing Company Inc.., 1995., Pp. 22-29

9. **IRIBARNE., L., Y OTROS.,** A non-functional approach for COTS-components trading., Buenos Aires-Argentina., In Proc. of WER., 2001., Pp. 76-81

10. **STEPHEN., T.,** The Art of Software Architecture: Design Methods and Techniques., Jakarta-Indonesia., Ed. Wiley, 2003., Pp. 172-189

11. **WIRTH., N.,** Program development by stepwise refinement., New York-E.U.A., ACM., 2003., Pp. 221-227

12. **BASPINEIRO., A.,** Planificación, Control y Calidad del SW., Salta-Argentina., Ed. Kapeluz., 2006., Pp. 2

13. **BERTOA., M.,** Atributos de calidad para Componentes COTS. Universidad de Málaga., Departamento de Lenguajes y Ciencias de la Computación., TESIS., Málaga-España., 2006., Pp. 1-32

14. **BILLY., R.,** Introducción a la arquitectura de Software., Universidad de Buenos Aires., PUBLICACIÓN., Buenos Aires-Argentina., 2007., Pp. 11-25

15. **CRISTIÁ., M.,** Introducción a la arquitectura de Software., Universidad de Navarra, Facultad de Ciencias Exactas, Ingeniería y Agrimensura., PUBLICACIÓN., Madrid-España., 2008., Pp.42-46

16. **DÍAZ., D.,** Meta-Patrones, una nueva aproximación a los patrones de diseño., Universidad de Navarra., PUBLICACIÓN., Madrid España., 2002., Pp. 2-6.

17. **EXO., S.A.,** Microsoft SQL Server 2005., Buenos Aires- Argentina., EXO S.A., 2010., Pp. 1-25

18. **FERNÁNDEZ., D.,** Definición de una arquitectura software para el diseño de aplicaciones web basadas en tecnología Java-J2EE., Universidad de Oviedo., TESIS., Oviedo-España., 2003., Pp. 3-10.

19. **GALLEGOS., M.,** Desarrollador 5 estrellas – Microsoft Visual Studio .NET., Quito-Ecuador., Microsoft Corporation., 2006., Pp.16-20

20. **HERGUEDA., M.,** La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real., PUBLICACIÓN., Valencia-España, Microsoft Corporation., 2006., Pp. 2-4

21. **LÓPEZ., J., Y OTROS.,** Modelos avanzados de comunicación de recursos., Universidad de Oviedo., PUBLICACIÓN., Oviedo-España., UPM., 2008., Pp. 57-82

22. **ROBLES., V.,** Lenguajes de Scripting: ¿una nueva forma de programar?., Universidad Politécnica Salesiana., PUBLICACIÓN., Cuenca-Ecuador., 2008., Pp. 28-30

23. **SHAW., M.,** Software Architecture: Perspectives on an emerging discipline., Universidad Carnegie Mellon., PUBLICACIÓN., Pittsburgh-Pennsylvania., Ed. Prentice Hall, 1996., Pp. 115-132

24. **THOMAS., F.,** Architectural styles and the design of network based software architectures., Universidad de California., TESIS DOCTORAL., Irvine-California., 2000., Pp. 69-83

25. **AJAX (PROGRAMMING)**
http://en.wikipedia.org/wiki/Ajax_%28programming%29
2010/08/09

26. AJAX: A NEW APPROACH TO WEB APPLICATIONS

http://adaptivepath.com/ideas/essays/archives/000385.php

2010/08/12

27. ALGUNAS DE LAS VENTAJAS DE LA PLATAFORMA .NET

http://www.desarrolloweb.com/articulos/1329.php

2010/03/12

28. ARQUITECTURA BÁSICA DE LA PLATAFORMA .NET

http://www.desarrolloweb.com/articulos/1328.php

2010/07/15

29. ARQUITECTURA EN N CAPAS

http://devsoftx.wordpress.com/2008/04/24/arquitectura-en-n-capas/

2010/05/21

30. ARQUITECTURA ORIENTADA A SERVICIOS

http://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios

2010/05/12

31. ASP.NET

http://es.wikipedia.org/wiki/ASP.NET

2010/07/22

32. INFORMACIÓN GENERAL DEL PRODUCTO SQL SERVER 2005

http://www.microsoft.com/spain/sql/productinfo/overview/default.mspx

2010/09/20

33. INTRODUCCIÓN A LA ARQUITECTURA DE SOFTWARE

http://www.willydev.NET/descargas/prev/IntroArq.pdf

2010/05/12

34. INTRODUCCIÓN A .NET

http://www.devjoker.com/contenidos/Conceptos-generales-NET/88/Introducción-a-NET.aspx
http://www.devjoker.com/contenidos/Conceptos-generales-NET/89/Common-Language-Runtime-CLR.aspx
2010/03/10

35. MICROSOFT INTERMEDIATE LANGUAGE ASSEMBLIES

http://www.scribd.com/doc/27358983/Microsoft-Intermediate-Language-Assemblies
2010/01/25

36. MICROSOFT .NET

http://es.wikipedia.org/wiki/Microsoft_.NET
2010/03/14

37. MICROSOFT SQL SERVER 2008

http://www.exo.com.ar/exos/paginas/plantillas_contenido/Page.asp?seccion=258&pagina=47&plantilla=Page.asp
2010/09/22

38. MODEL-VIEW-CONTROLLER

http://msdn.microsoft.com/en-us/library/ff649643.aspx
2010/08/15

39. MS SQL SERVER

http://msdn.microsoft.com/sql/express
2010/09/24

40. .NET DE MICROSOFT

http://www.solodisenio.com/net-de-microsoft/
2010/05/17

41. .NET FRAMEWORK

http://www.microsoft.com/net/
2010/07/12

42. .NET FRAMEWORK DEVELOPER CENTER.

http://msdn.microsoft.com/netframework/

2010/05/17

43. NUEVAS TENDENCIAS EN EL SOFTWARE

www.geocities.com/eztigma

2011/01/20

44. ¿PARA QUÉ SIRVE UNA ARQUITECTURA DE SOFTWARE?

http://armanhal.blogspot.com/2004/05/para-qu-sirve-una-arquitectura-de.html

2010/03/03

45. PROGRAMMING C#: BUILDING .NET APPLICATIONS WITH C#

http://www.amazon.com/exec/obidos/ASIN/0596006993/techbookrepor-20#reader_0596006993

2010/07/20

46. PROTOTYPE JAVASCRIPT FRAMEWORK

http://www.prototypejs.org/

2010/08/13

47. PROTOTYPE JAVASCRIPT FRAMEWORK

http://en.wikipedia.org/wiki/Prototype_JavaScript_Framework

2010/08/15

48. TECNOLOGÍA ORIENTADA A OBJETOS

http://generacion-vc.blogspot.com/

2010/02/15

ANEXOS

Anexo 1. Formato de encuesta a usuarios.

ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO

ENCUESTA APLICADA A USUARIOS DEL SISTEMA

Estimado compañero de la fundación San Luis (PRONACA). Un cordial saludo. La presente encuesta es de vital importancia en esta investigación, ya que servirá de parámetro para mejorar la labor en el diseño de la aplicación que Ud. actualmente utiliza. Por su colaboración le anticipo mi agradecimiento.

DATOS DE ENCUESTADO

Nombre: _____

Cargo: _____

INSTRUCCIONES

- Marque con una "X", la respuesta que crea conveniente.

PREGUNTAS

1. ¿Qué tipo de sistema operativo conoce y/o utiliza con frecuencia?

Linux	
SCO Unix	
Windows	
Solaris	
OS2	
Otro ( )	

2. ¿Qué grado de importancia tiene para su trabajo, la implementación, uso y futuras actualizaciones de un sistema acorde a sus necesidades?

Muy importante	
Importante	
Poco importante	
Nada importante	

3. ¿Ha encontrado beneficios en su entorno de trabajo desde la implementación del programa desarrollado dentro de la fundación?

Suficiente beneficio	
Lo justo	
Ningún beneficio	

4. ¿Cuál es su grado de conformidad o inconformidad con el programa que actualmente utiliza para satisfacer las necesidades específicas de su labor?

Completamente conforme	
Conforme	
Neutral	
Inconforme	
Completamente inconforme	

5. ¿Cuál es el nivel de comunicación y entendimiento entre Ud. como usuario y el grupo desarrollador en caso de sugerir o solicitar cambios y mejoras en el sistema?

Buena comunicación	
Comunicación adecuada	
Poca comunicación	

6. ¿El programa lleva a cabo las operaciones especificadas y con la precisión requerida?

Siempre es fiable	
Casi siempre es fiable	
Aún tiene problemas	

7. ¿Qué tasa de información envía y recibe comúnmente al momento de utilizar la aplicación?

Gran cantidad de información	
Mucha información	
Poca información	
Nada de información	

8. ¿Qué tiempo le lleva al sistema, el procesar y responder la tasa de información que maneja comúnmente al utilizar la aplicación?

Mucho tiempo	
Tiempo prudencial	
Poco tiempo	

9. ¿Qué cantidad de recursos hardware y software que necesita la aplicación para la realización de las operaciones con los tiempos de respuesta adecuados?

Usa muchos recursos	
Usa lo justo	
Pocos recursos	

10. ¿En qué grado puede controlarse el acceso al software o a los datos a personal no autorizado?

Controla eficientemente	
Controla bien	
Aún tiene problemas	

11. ¿Cuál es el grado de esfuerzo que Ud. emplea para aprender el manejo de la aplicación, trabajar con ella, ingresar datos y conseguir resultados?

Mucho esfuerzo	
Esfuerzo normal	
Ningún problema	

12. ¿Con qué frecuencia se presentan problemas en el funcionamiento de la aplicación?

Siempre existen problemas	
Existen pocos problemas	
No hay problemas	

13. ¿Qué periodo de tiempo le lleva al sistema reiniciar la operación ejecutada en caso de presentarse una falla en un intervalo temporal determinado?

Mucho tiempo	
Tiempo prudencial	
Poco tiempo	

14. ¿Han existido cambios evolutivos en la aplicación que utiliza desde su puesta en funcionamiento?

Evolución	
Evolución normal	
Poca evolución	

15. ¿Qué tiempo ha tenido que esperar para que los cambios, mejoras o nuevas implementaciones solicitadas, se vean reflejadas en la aplicación?

Mucho tiempo	
Tiempo adecuado	
Poco tiempo	

Gracias por su colaboración.

Anexo 2. Formato de encuesta a desarrolladores.

ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO

ENCUESTA APLICADA A DESARROLLADORES DEL SISTEMA

Estimado compañero del departamento de desarrollo de sistemas. Un cordial saludo. La presente encuesta es de vital importancia en esta investigación, ya que servirá de parámetro para mejorar la labor en el diseño y mantenimiento de la aplicación que se está desarrollando. Por su colaboración le anticipo mi agradecimiento.

DATOS DE ENCUESTADO

Nombre: _____

Cargo: _____

INSTRUCCIONES

- Desde la pregunta 1 a 9, marque con una "X", la respuesta que crea conveniente.
- La pregunta 10 y 11, ordene numéricamente según la importancia cada uno de los criterios.

PREGUNTAS

1. ¿Qué tipo de sistema operativo conoce y/o utiliza con frecuencia?

Linux	
Unix	
Windows	
Solaris	
OS2	
Apple	
Ubuntu	
Otro ( )	

2. ¿Qué lenguaje de programación conoce o domina?

Visual Basic	
C	
C++	
C#	
PHP	
Java	
Delphi	
Otro ( )	

3. ¿Qué plataforma conoce o maneja para el diseño de base de datos?

Oracle	
PL/SQL	
SQL Server	
Informix	
SyBase	
Access	
DB2	
Otro ( )	

4. ¿Qué tipos de arquitectura conoce y/o ha utilizado en su experiencia profesional?

Cliente – Servidor	
Multicapa	
Embebidos	
Centralizados	
Par a par	
Distribuidos	
Orientado a servicio	
Otra ( )	

5. ¿Ha tenido inconvenientes al momento de la implementación, diseño y puesta a punto de la aplicación?

Demasiados	
Algunos	
Pocos	
Ningún	

(*Si la respuesta en "Ningún", pase a la pregunta 8*)

6. ¿En qué contextos se le ha presentado los inconvenientes?

Creación de nuevos requerimientos	
Localización y reparación de errores	
Modificación de la aplicación en funcionamiento	
Prueba de la aplicación	
Tiempo de diseño	
Entendimiento en la estructura del programa	

7. ¿Cuál de los siguientes aspectos considera que ha sido el factor por el cual presenta inconvenientes en la programación de aplicaciones?

Hardware	
Sistema operativo	
Lenguaje de programación en el que se está trabajando	
Falta de organización en la programación	
Falta de comunicación entre diseñadores y usuarios	
Tipo de arquitectura	
Falta de nuevas estrategias y herramientas tecnológicas	

8. ¿Ha considerado dentro de sus planes a corto plazo adquirir, aprender y utilizar nuevas herramientas tecnológicas?

Si	
No	
Indeciso	

9. ¿Si se desarrollaran nuevas estrategias de programación de componentes de software que le permitiera agilitar su trabajo, las utilizaría?

Si	
No	
Indeciso	

10. Ordene, según su criterio, los criterios de calidad que considera importantes en su entorno de trabajo.

CRITERIO	DEFINICIÓN	ORDEN
Funcionalidad	Capacidad del componente software para proporcionar un conjunto apropiado de funciones que satisfagan las necesidades establecidas o implícitas.	
Fiabilidad	Capacidad del producto de estar siempre disponible a pesar de presentar fallas en un intervalo temporal determinado.	
Usabilidad	Atributo que mide la facilidad de uso del componente durante el diseño de las aplicaciones.	
Eficiencia	Capacidad del producto software para proporcionar tiempos de respuesta, tiempos de proceso y potencia apropiados.	
Mantenibilidad	Capacidad del producto software de poder diagnosticar deficiencias o causas de los fallos en el software, o para identificar las partes que han de ser modificadas.	

11. En la siguiente lista, ordene según la importancia, los atributos durante el ciclo de vida del producto debe tener el producto de software que se encuentra implementando actualmente.

CRITERIO	DEFINICIÓN	ORDEN
Interoperabilidad	Capacidad que tiene el sistema puede interactuar con uno o más aplicaciones especificadas.	
Escalabilidad	Capacidad de extenderse a un número mayor o más poderoso de servidores al incrementarse la demanda o la carga.	
Modificabilidad	Atributo que permite que una determinada modificación sea implementada, bien para mejorar las propiedades del sistema o para introducir nuevas funcionalidades de una forma rápida y poco costosa.	
Portabilidad	Atributo que permite al software, ser adaptado e instalado a diferentes entornos heterogéneos especificados.	
Reusabilidad	Grado en el que elementos del sistema pueden ser utilizados para construir nuevos productos. Es una forma particular de modificabilidad.	
Integrabilidad	Habilidad del producto software para hacer que piezas de software desarrolladas separadamente, trabajen correctamente juntas.	
Testeabilidad	Es el grado de facilidad que tiene un sistema para mostrar sus defectos y para ser probada en su completitud	

16. ¿La aplicación actualmente interactúa con una o más aplicaciones desarrolladas por organizaciones independientes?

Si	
No	

Gracias por su colaboración.

Anexo 3. Resultado de encuestas previas al diseño de la arquitectura, dirigida a usuarios y desarrolladores.

a) Encuesta a usuarios.

Pregunta 1.

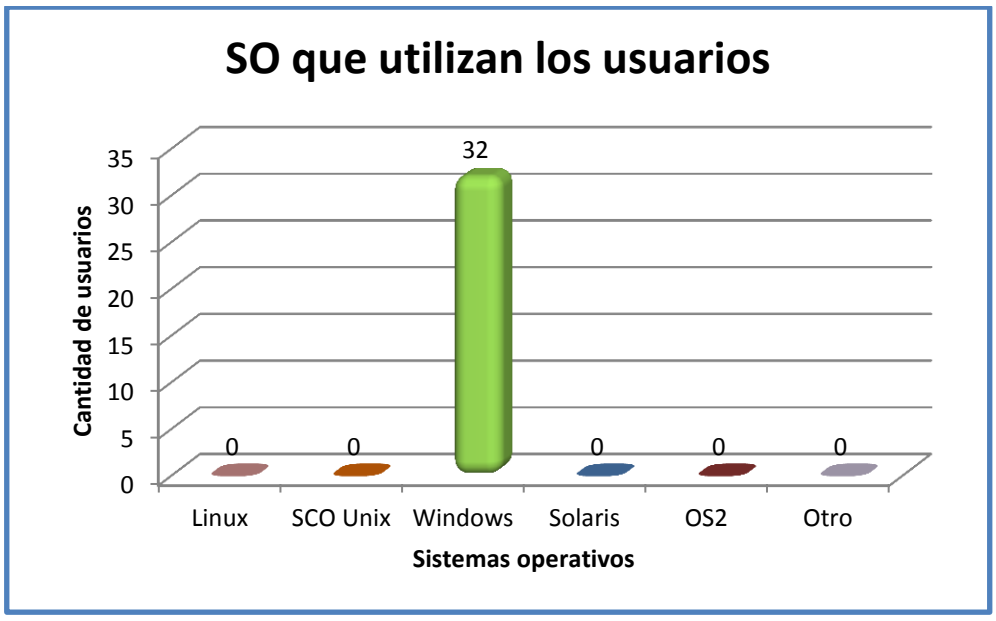

Gráfico 10. Sistema operativo utilizado por los usuarios
Fuente: investigador

Pregunta 2

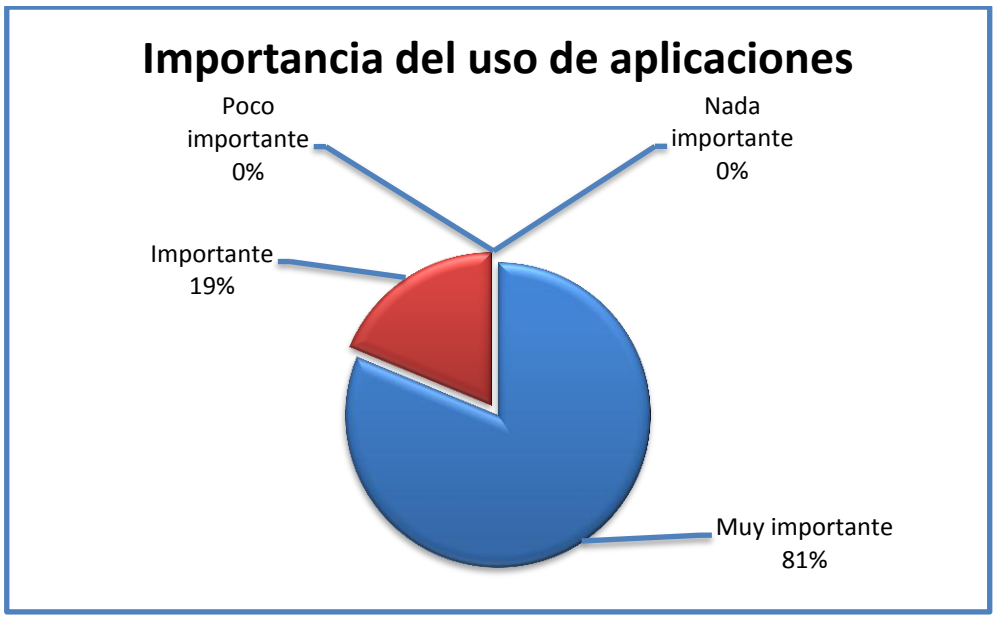

Gráfico 11. Importancia del uso de aplicaciones
Fuente: investigador

Pregunta 3.

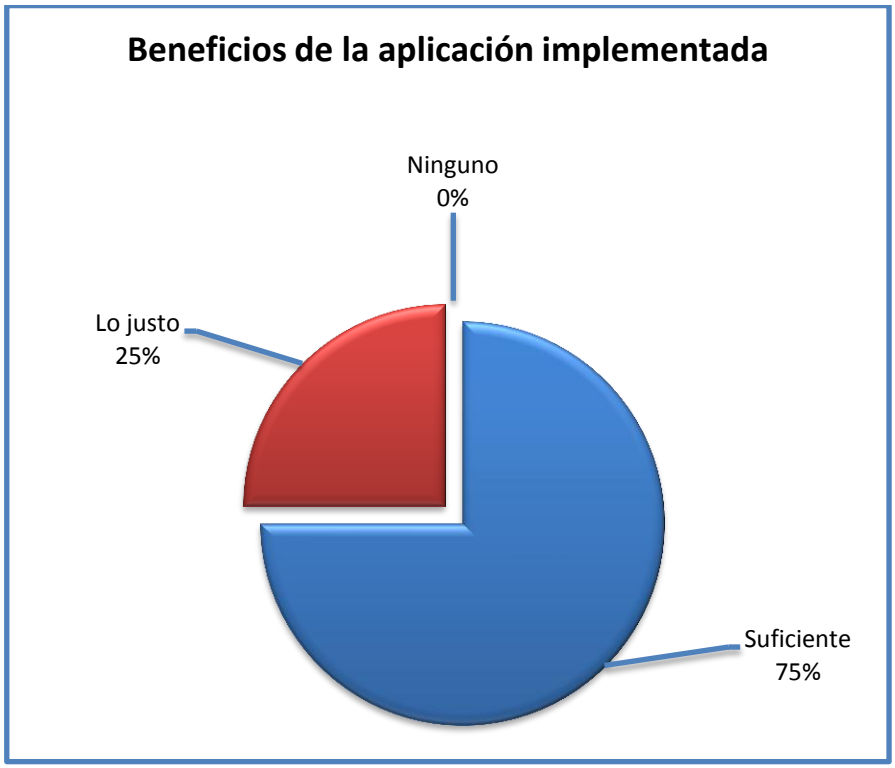

Gráfico 12. Beneficios alcanzados con la implementación de la aplicación
Fuente: investigador

Pregunta 4.

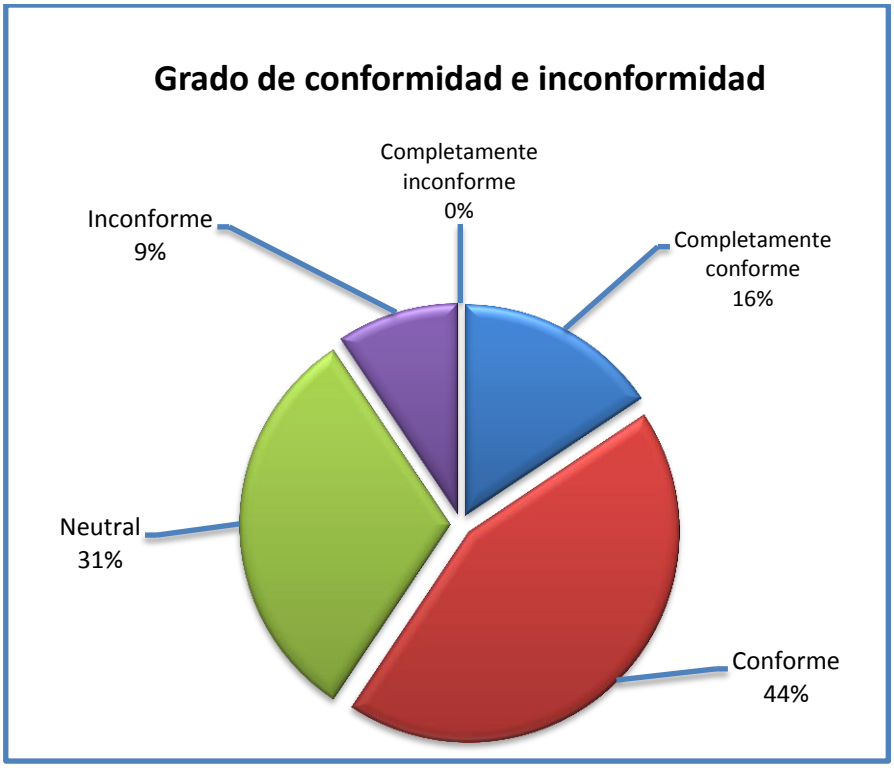

Gráfico 13. Grado de conformidad con la aplicación implementada
Fuente: investigador

Pregunta 5.

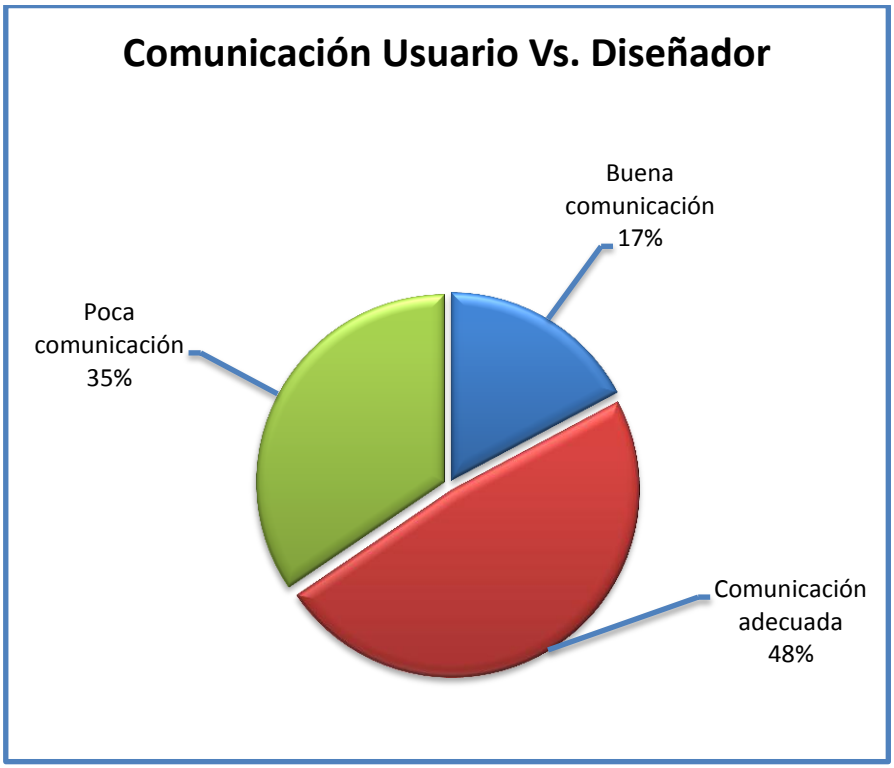

Gráfico 14. Comunicación - Usuario Vs. Diseñador
Fuente: investigador

Pregunta 6.

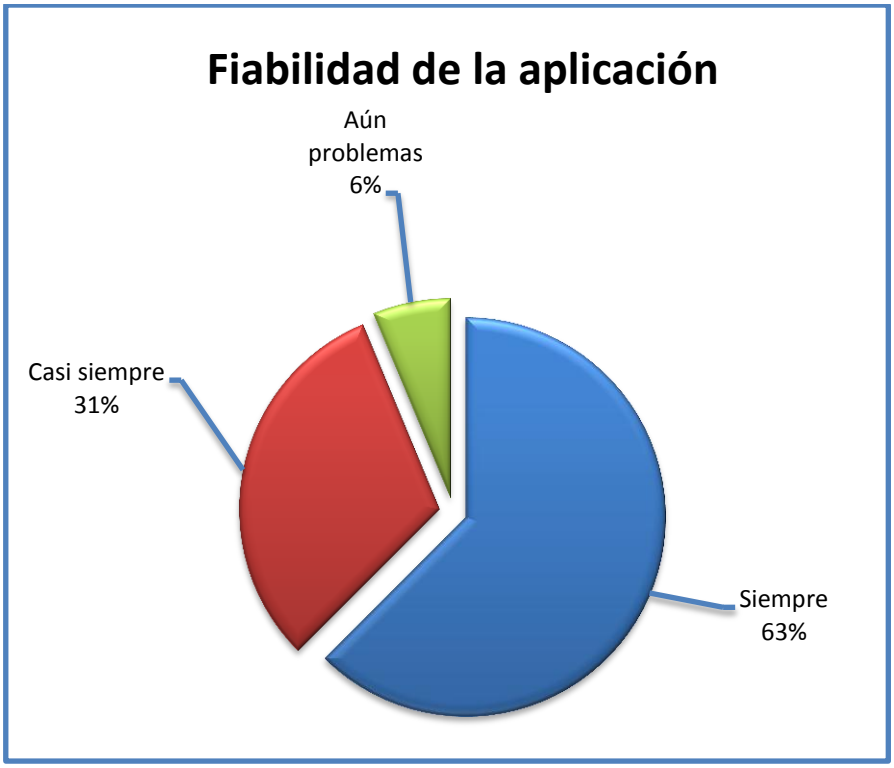

Gráfico 15. Fiabilidad de la aplicación
Fuente: investigador

Pregunta 7.

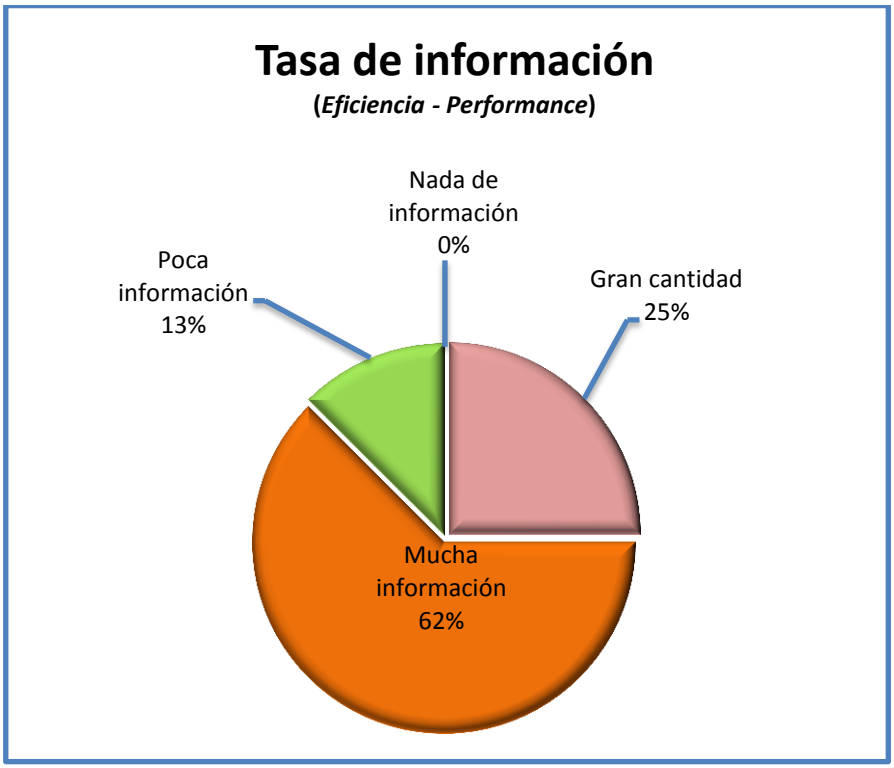

Gráfico 16. Eficiencia – Performance de la aplicación
Fuente: investigador

Pregunta 8.

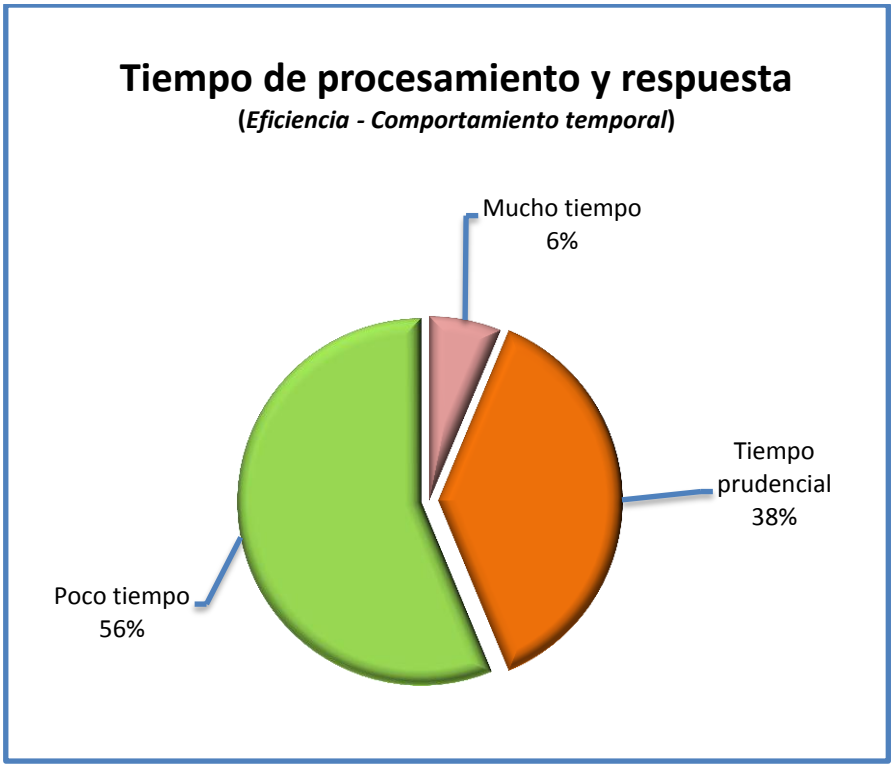

Gráfico 17. Eficiencia – Comportamiento temporal de la aplicación
Fuente: investigador

Pregunta 9.

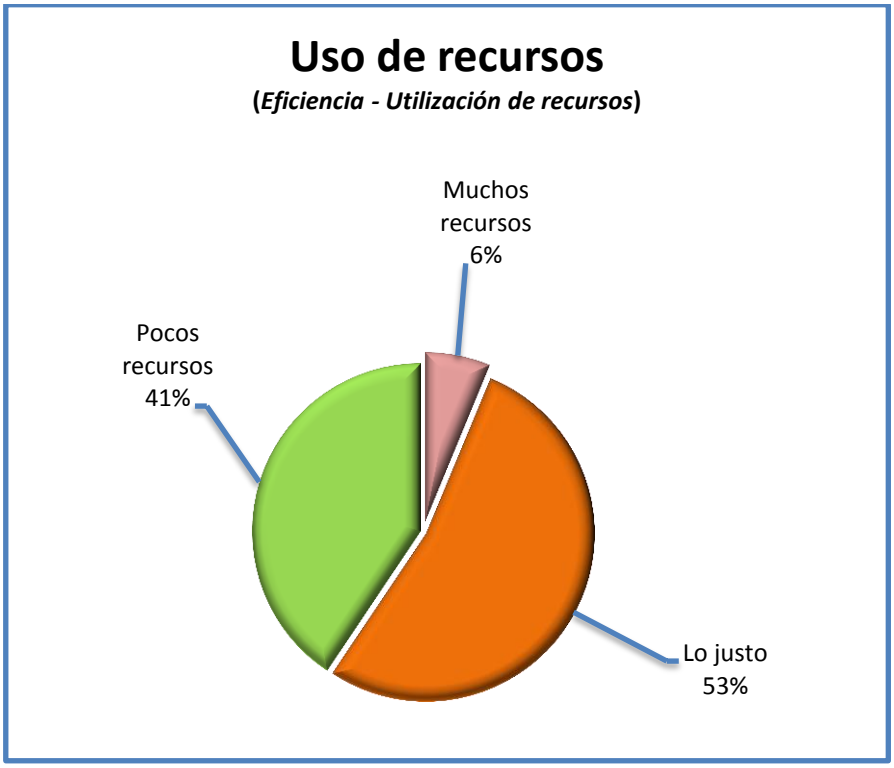

Gráfico 18. Eficiencia – Utilización de recursos de la aplicación
Fuente: investigador

Pregunta 10.

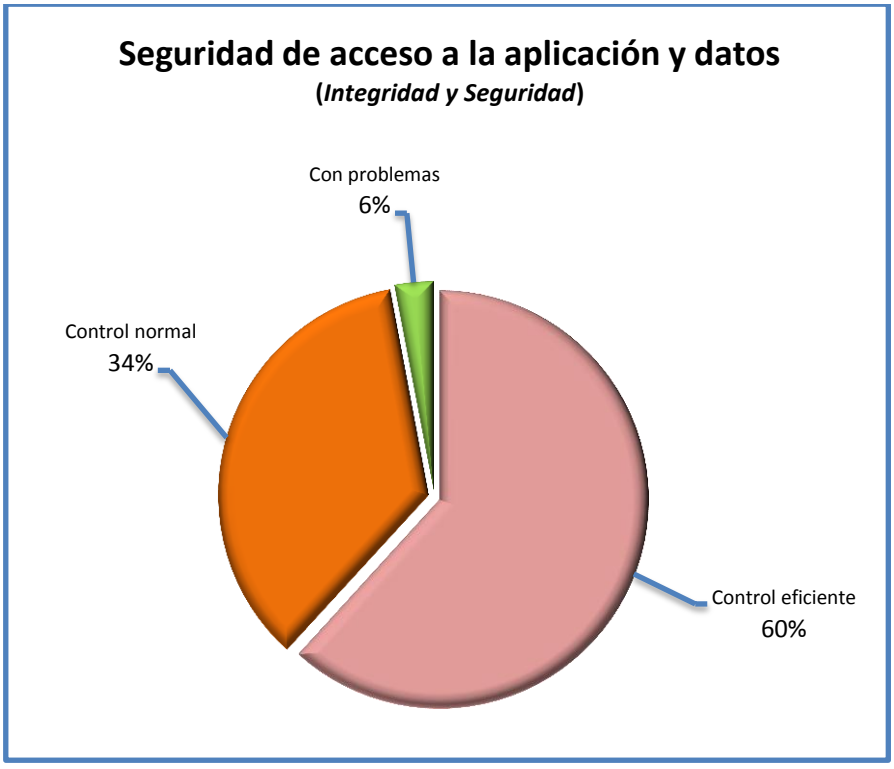

Gráfico 19. Integridad y seguridad de acceso a la aplicación y datos
Fuente: investigador

Pregunta 11.

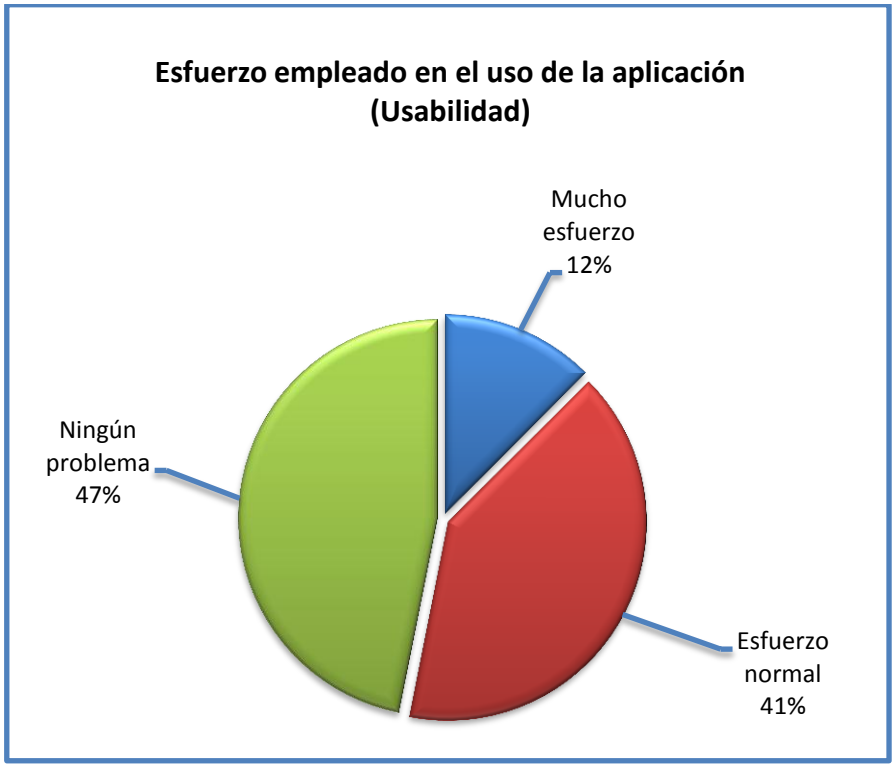

Gráfico 20. Usabilidad de la aplicación
Fuente: investigador

Pregunta 12.

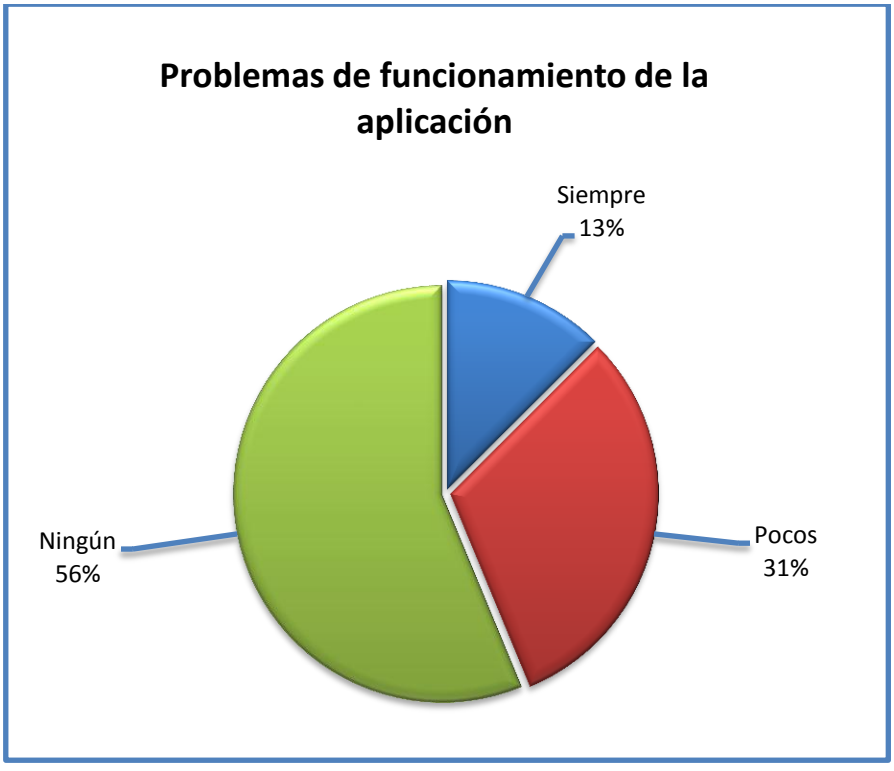

Gráfico 21. Funcionalidad de la aplicación
Fuente: investigador

Pregunta 13.

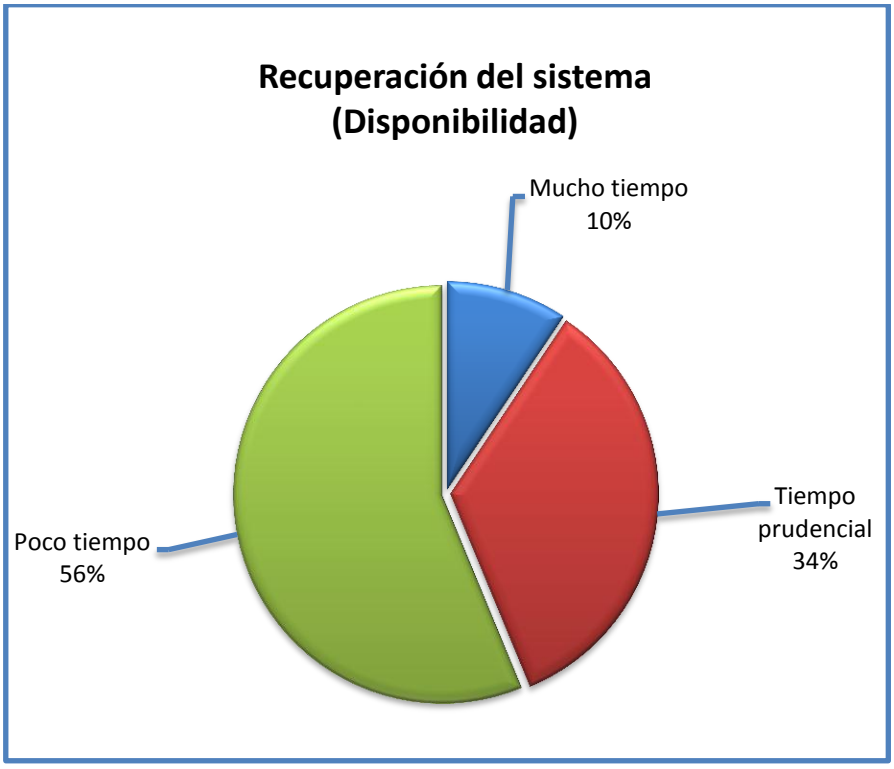

Gráfico 22. Disponibilidad de la aplicación
Fuente: investigador

Pregunta 14.

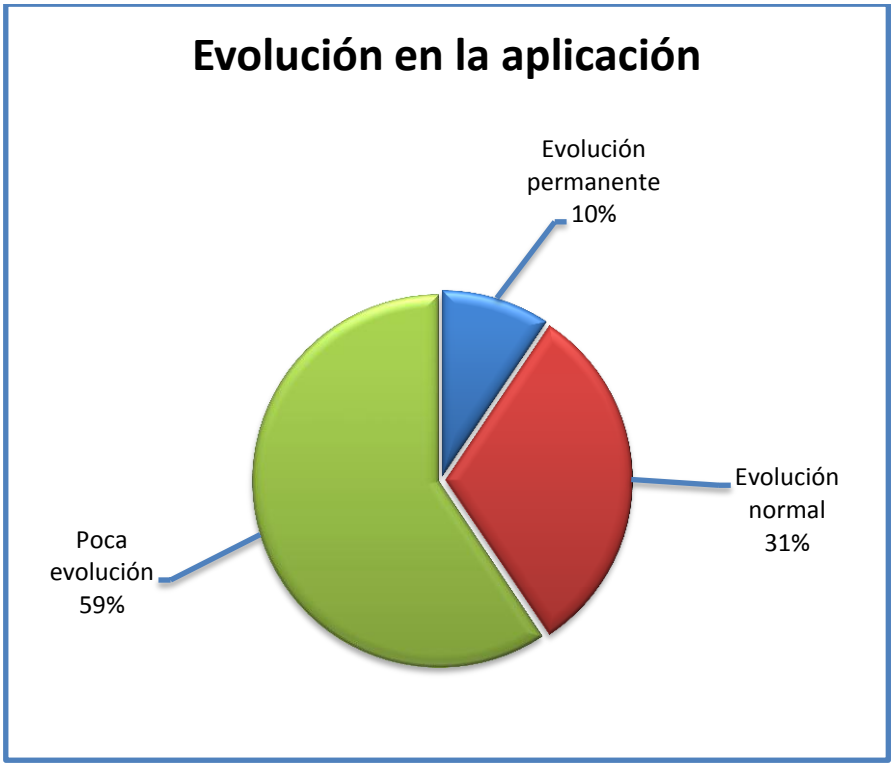

Gráfico 23. Evolución de la aplicación
Fuente: investigador

Pregunta 15.

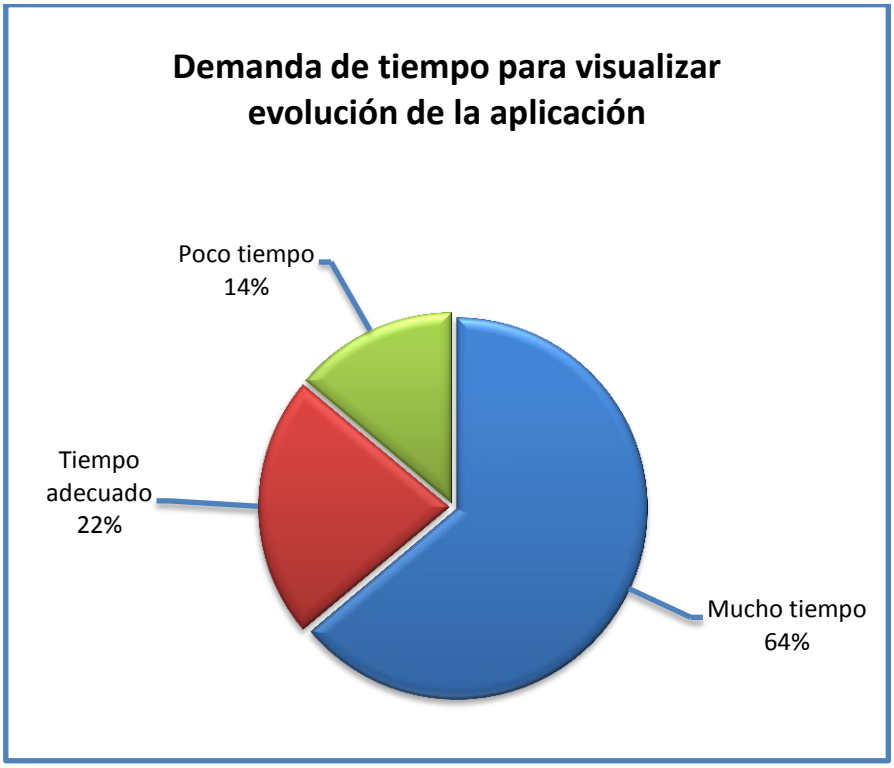

Gráfico 24. Tiempo de espera en la evolución de la aplicación
Fuente: investigador

b) Encuesta a desarrolladores.

Pregunta 1.

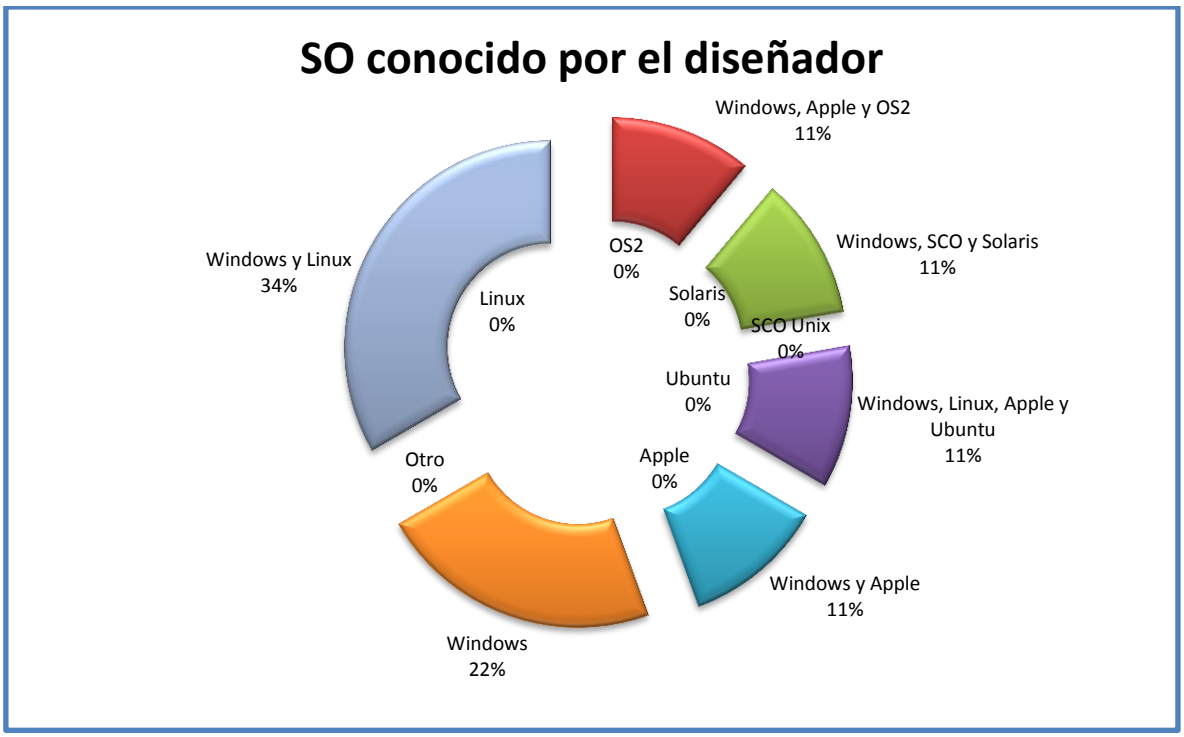

Gráfico 25. Sistema operativo conocido por el grupo desarrollador
Fuente: investigador

Pregunta 2.

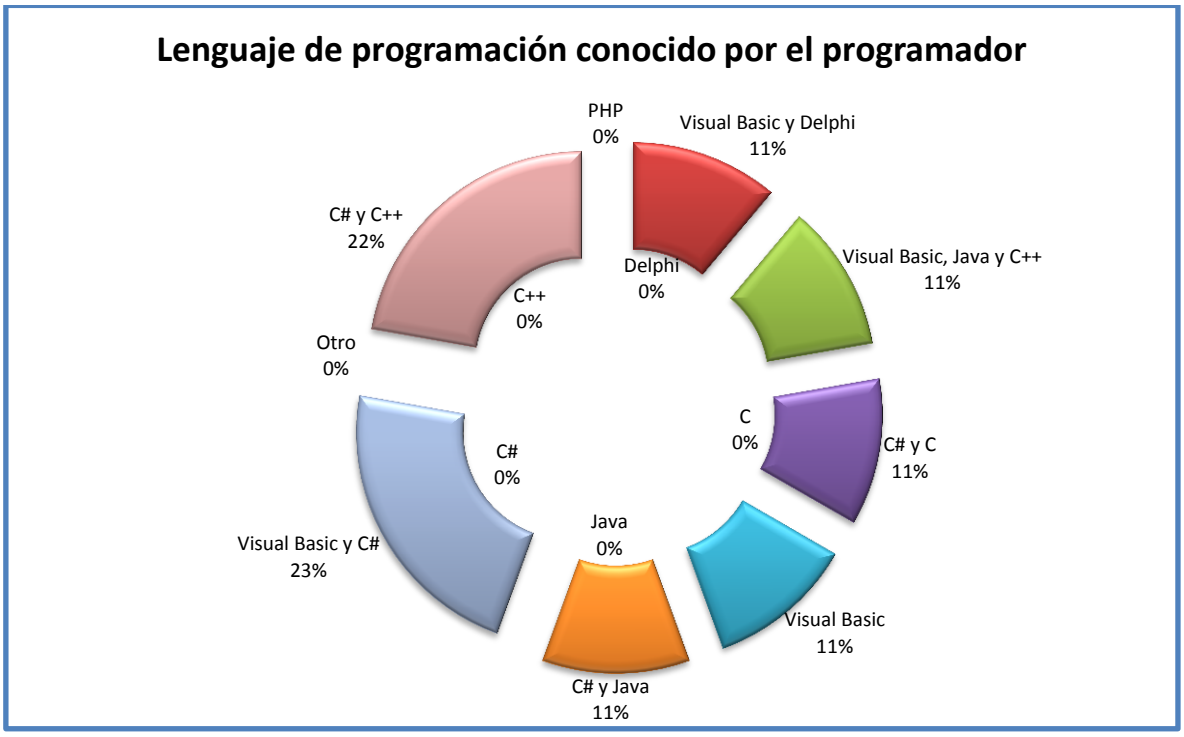

Gráfico 26. Lenguaje de programación conocido por el diseñador
Fuente: investigador

Pregunta 3.

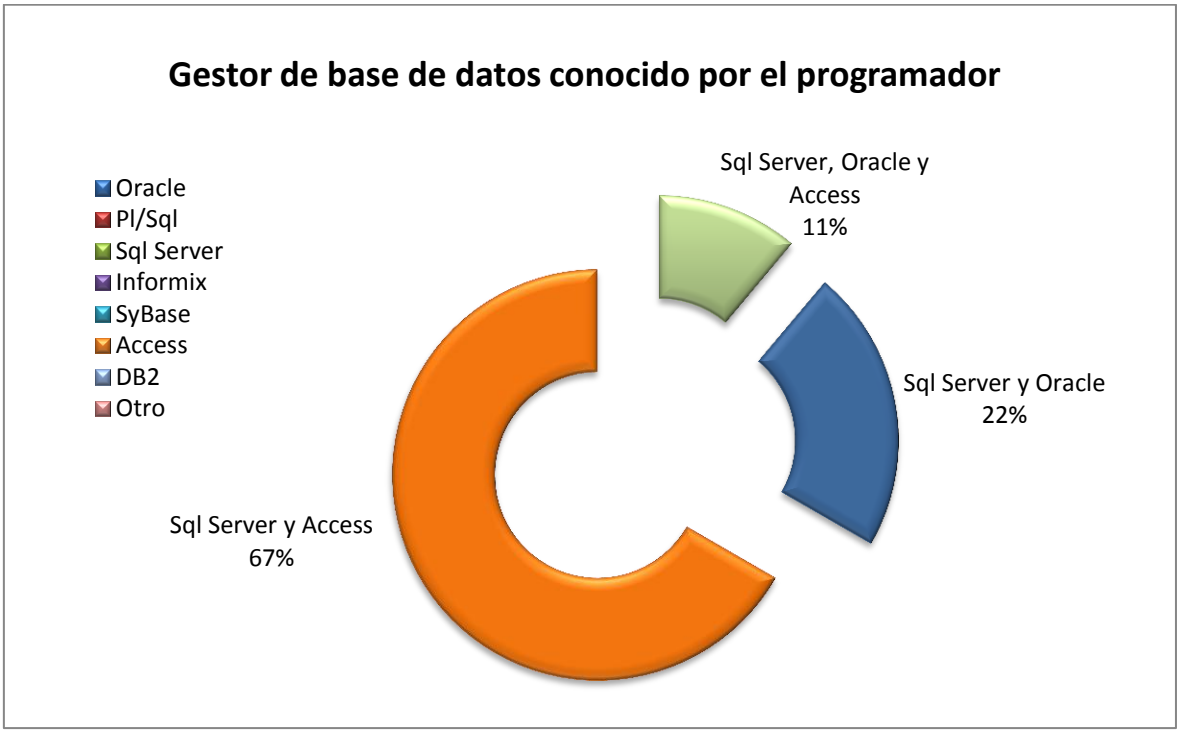

Gráfico 27. Gestor de BD conocido por el diseñador
Fuente: investigador

Pregunta 4.

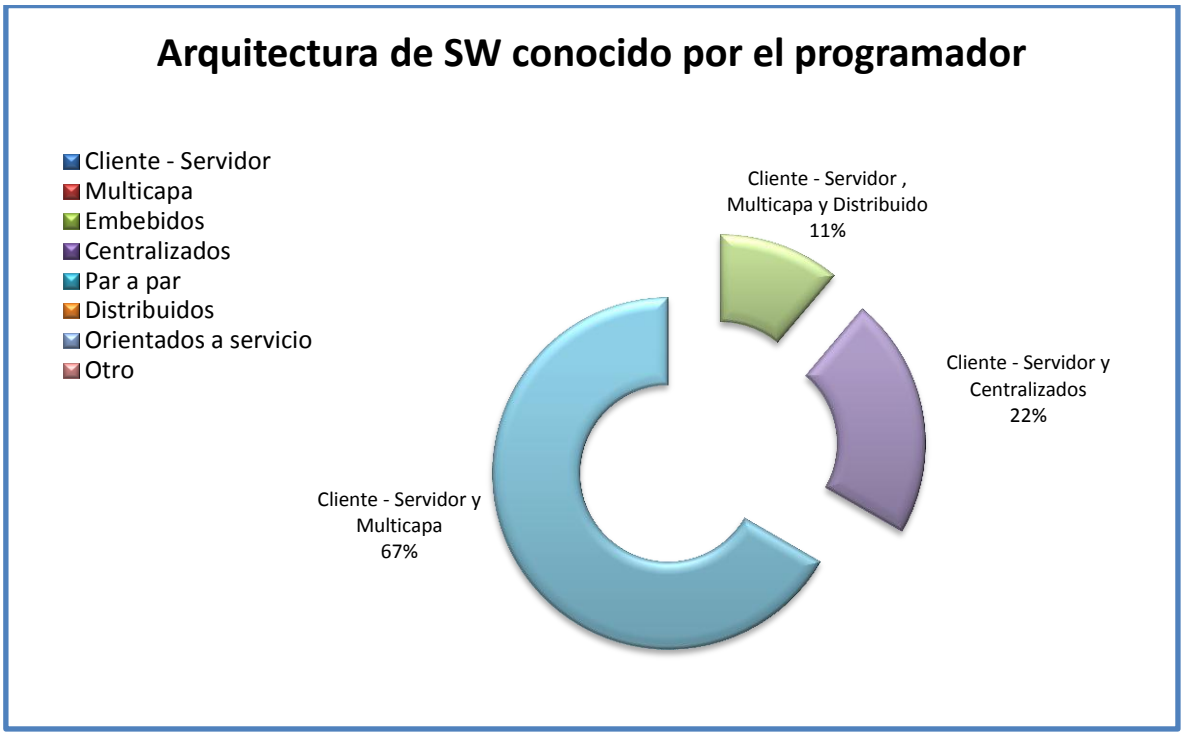

Gráfico 28. Arquitectura de SW conocido por el diseñador
Fuente: investigador

Pregunta 5.

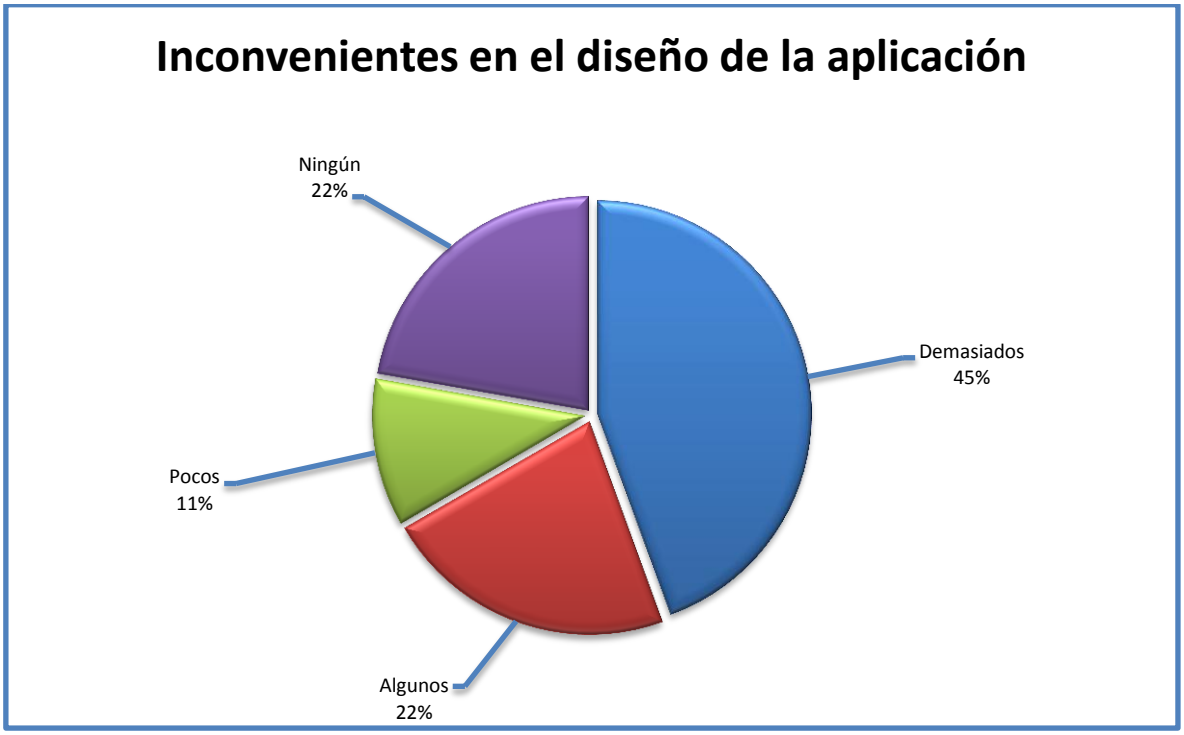

Gráfico 29. Diseñadores que han encontrado inconvenientes en el diseño de la aplicación
Fuente: investigador

Pregunta 6.

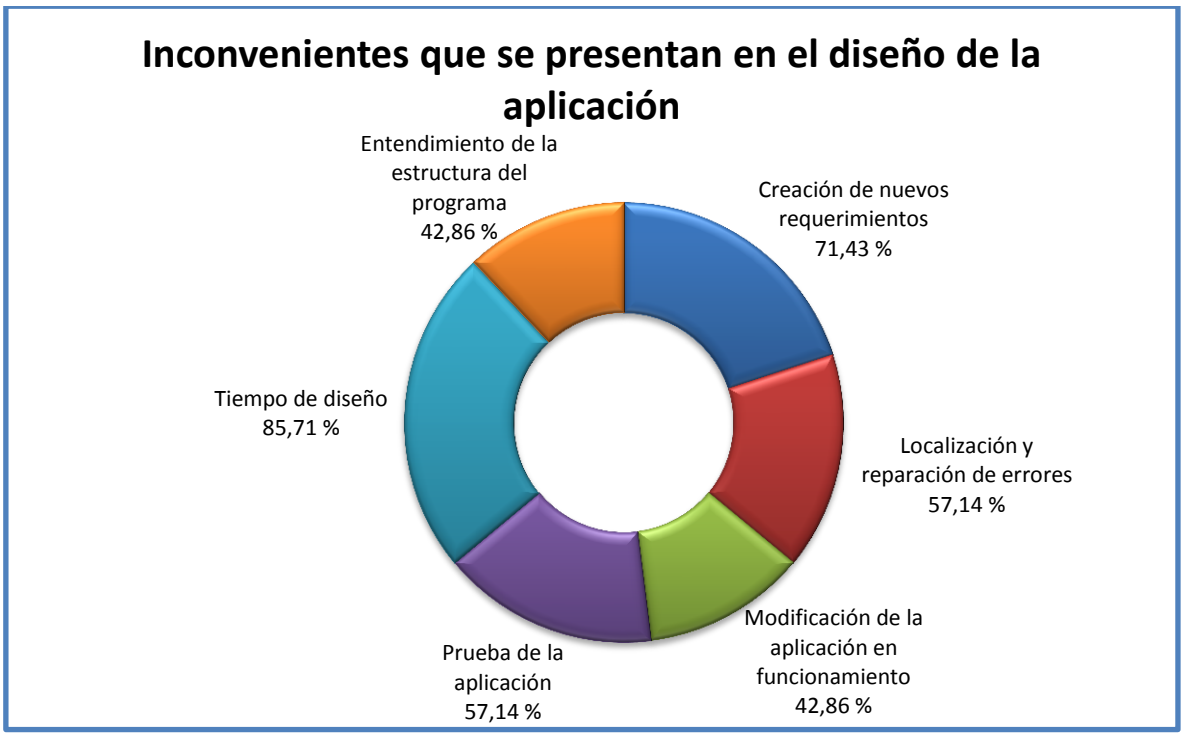

Gráfico 30. Inconvenientes presentados en el diseño de la aplicación
Fuente: investigador

Pregunta 7.

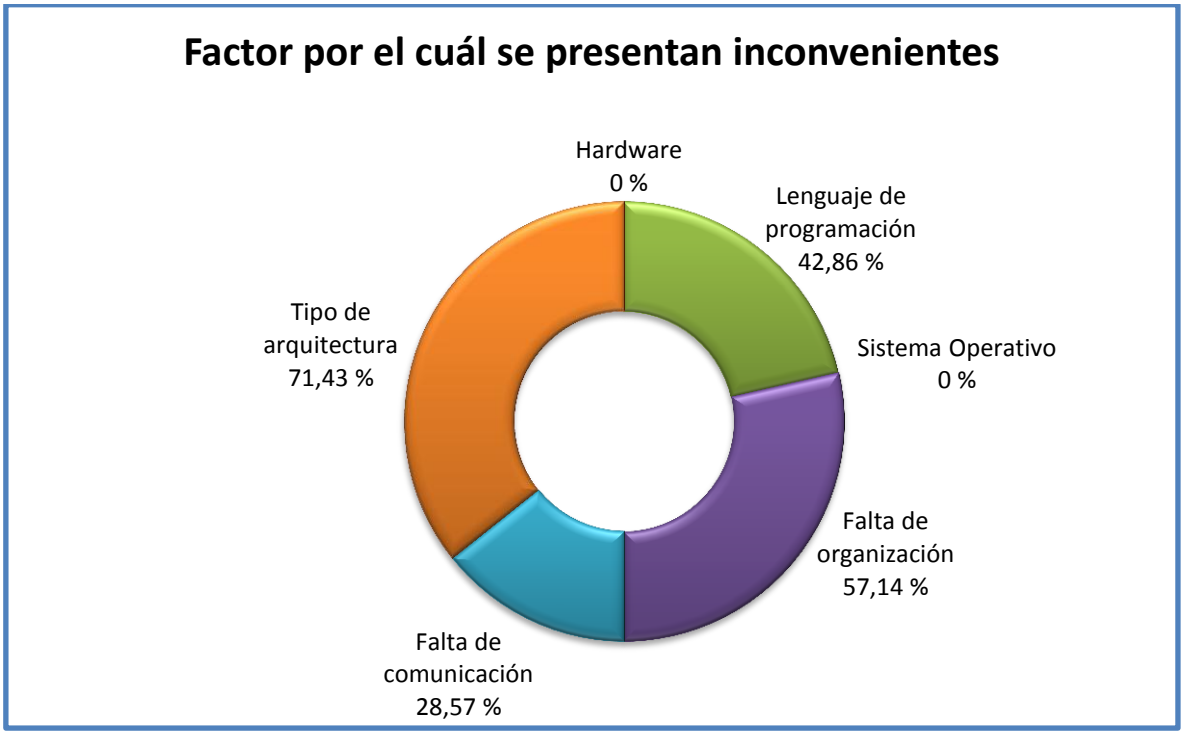

Gráfico 31. Factores que determinan los inconvenientes en el diseño de la aplicación
Fuente: investigador

Pregunta 8.

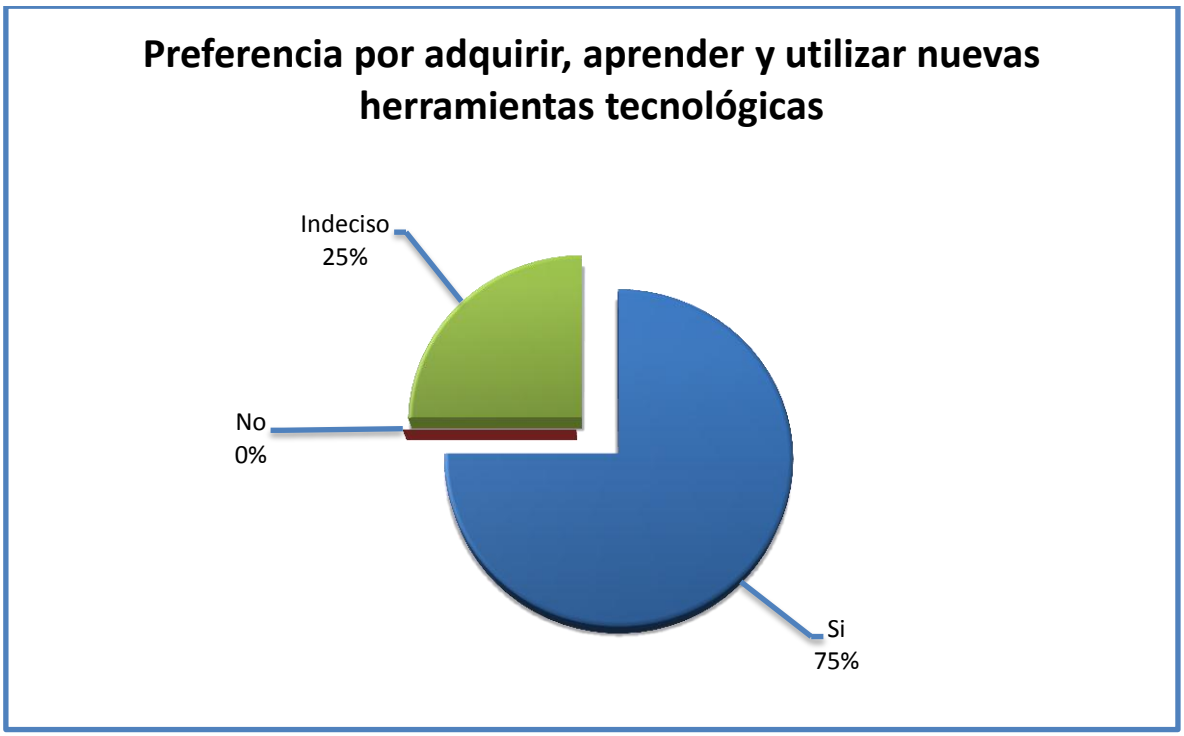

Gráfico 32. Preferencia por el uso de nuevas herramientas tecnológicas
Fuente: investigador

Pregunta 9.

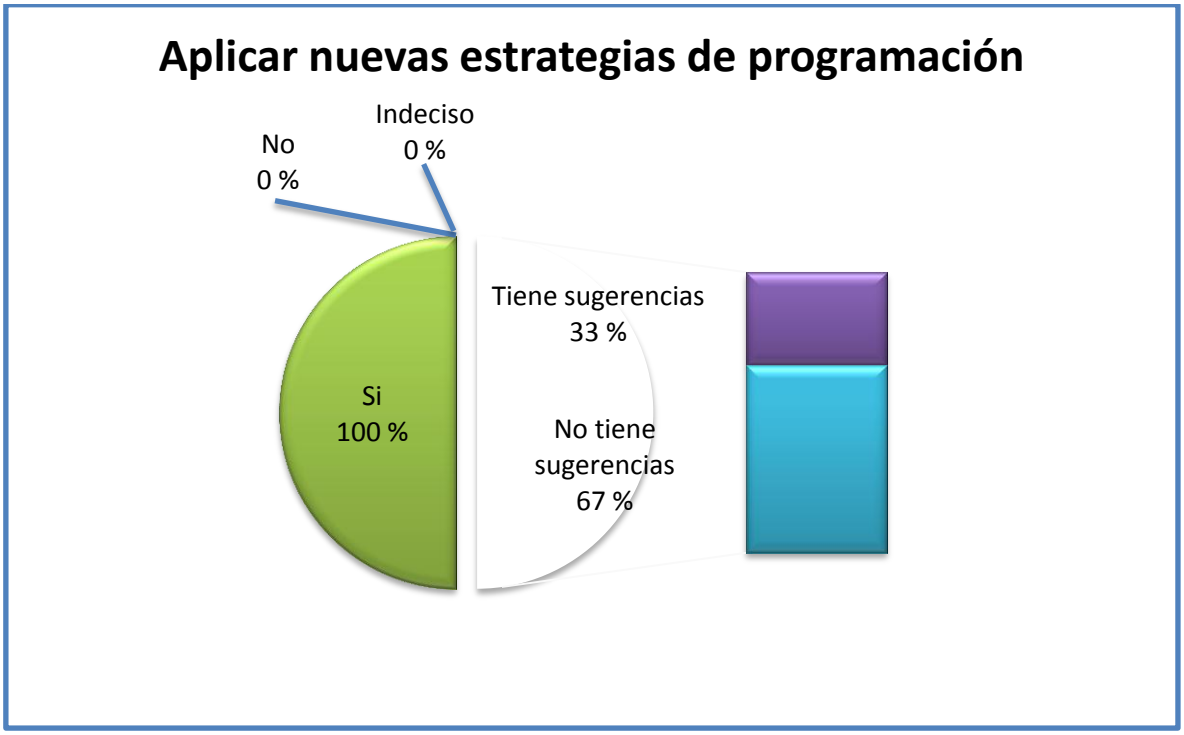

Gráfico 33. Decisión de aplicar nuevas estrategias de programación

Pregunta 10.

Gráfico 34. Prioridad de los criterios generales de calidad
Fuente: investigador

Pregunta 11.

Gráfico 35. Prioridad de los atributos de calidad durante el ciclo de vida del producto
Fuente: investigador

Anexo 4. Formato de encuesta final a desarrolladores.

ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO

ENCUESTA APLICADA A DESARROLLADORES DEL SISTEMA

Estimado compañero del departamento de desarrollo de sistemas. Un cordial saludo. La presente encuesta es de vital importancia en esta investigación, ya que servirá de parámetro para determinar el estado actual de su labor en el diseño y mantenimiento de la aplicación que se está desarrollando. Por su colaboración le anticipo mi agradecimiento.

DATOS DE ENCUESTADO

Nombre: _____

Cargo: _____

PREGUNTAS

FUNCIONALIDAD

ESCALABILIDAD

1. ¿En qué nivel cree Ud. que la arquitectura implementada, permita que la aplicación sea escalable?

☐ Bajo ☐ Medio ☐ Alto ☐ Muy alto

DISPONIBILIDAD

2. ¿Qué valor de relación ha obtenido entre el número de veces que debe parar el sistema, con el número de veces que debe realizar correcciones en el sistema?

☐ Entre 0 y 25% ☐ Entre 25% y 50% ☐ Entre 50% y 75% ☐ Entre 75% y 100%

UTILIZACIÓN DE RECURSOS

3. ¿La división de la aplicación conseguida con la arquitectura propuesta, ha ayudado en la optimización de los recursos tanto del lado del servidor como del lado de los clientes?

☐ Si. ☐ No

MANTENIBILIDAD

MODIFICABILIDAD

4. ¿Qué nivel de esfuerzo y coste conlleva el cambio o diseño de nuevos procesos con el modelo de arquitectura propuesto?

☐ Bajo ☐ Medio ☐ Alto

REUSABILIDAD

5. La arquitectura propuesta le ha permitido reutilizar componentes diseñados en la aplicación para el diseño de otras funcionalidades.

☐ Siempre ☐ Frecuentemente ☐ A veces ☐ Nunca

6. ¿Qué valor de relación ha obtenido entre el número de veces que ha reutilizado un componente, con el número de veces que ha tenido que diseñar procesos en el sistema?

☐ Entre 0 y 25% ☐ Entre 25% y 50% ☐ Entre 50% y 75% ☐ Entre 75% y 100%

ACOPLAMIENTO

7. En los procesos de producción de software, ¿Qué nivel de acoplamiento exige la estructura arquitectónica implementada?

☐ Bajo ☐ Medio ☐ Alto

TESTABILITY

8. ¿Qué nivel de facilidad presenta el modelo de arquitectura propuesto al momento de la detección de errores en relación a procesos de diseño anteriores?

☐ Mayor ☐ Igual ☐ Menor

TIEMPO DE DESARROLLO

9. El tiempo que emplea actualmente al momento de diseñar y/o mejorar los procesos en la aplicación en relación a la anterior forma de producir aplicaciones es:

☐ Menor ☐ Igual ☐ Mayor

Gracias por su colaboración.

Anexo 5. X^2 - Valores y fórmulas.

En la investigación efectuada, la prueba de hipótesis por ser correlacional se utilizó la prueba chi-cuadrado (X^2), que es una prueba no paramétrica que mide la discrepancia entre una distribución observada y otra teórica.

La prueba del χ^2 se usa para variables de distintos niveles de medición, incluyendo las de menor nivel, que son las nominales. Sirve para determinar si los datos obtenidos de una sola muestra presentan variaciones estadísticamente significativas respecto de la hipótesis nula.

Cuando formulamos una hipótesis de trabajo, simultáneamente definimos la hipótesis nula, que niega nuestra hipótesis de trabajo. De acuerdo a la hipótesis nula (H_0) las variaciones en la variable independiente no tienen correspondencia con las variaciones que pudiere haber de la variable dependiente. Es decir que existe "independencia estadística". Las variaciones que pudiese encontrarse se deberían a factores aleatorios, ajenos a la variable independiente.

Para comprobar si esto es así o no, podemos someter los resultados obtenidos de nuestra muestra a una prueba de χ^2, que se postula con la siguiente ecuación:

$$\chi^2 = \frac{\Sigma\,(f_e - f_o)^2}{f_e}$$

Figura 46. Fórmula para el cálculo del chi-cuadrado.

Pero la aplicación del chi cuadrado no se puede hacer directamente. Es necesario, antes de ello, realizar dos pasos. Por una parte, establecer el nivel de significación (α) con el cual vamos a trabajar, y determinar los grados de libertad (*gl*) de nuestra muestra.

El nivel de significación es arbitrario y se fija de antemano (*usualmente entre 0.01 y 0.10*). Los grados de libertad se establecen en función de la cantidad de celdas que tenemos, producto de las categorías de una variable o bien de la cantidad resultante del cruce de dos variables.

Cuanto mayor sea el valor de χ^2, menos verosímil es que la hipótesis sea correcta. De la misma forma, cuanto más se aproxima a cero el valor de chi-cuadrado, más ajustadas están ambas distribuciones.

Cálculo de los grados de libertad (*gl*)

Los grados de libertad **gl** vienen dados por:

gl= (*r*-1)(*k*-1). Donde *r* es el número de filas y *k* el de columnas.

Cálculo de las frecuencias esperadas (f_e)

Las frecuencias esperadas (f_e) vienen dadas por la hipótesis nula (H_0), pero no siempre se puede establecer de manera inmediata. Esto solo es posible cuando trabajamos con una variable, pero cuando tenemos cuadros de doble entrada la forma de establecer el valor de la frecuencia esperada de cada celda es el siguiente:

Tabla 15. Matriz de frecuencias y resultados.

	Categoría 1	Categoría 2	Categoría 3	Marginal 1/2/3
Categoría A	a	b	c	$(a+b+c)$
Categoría B	d	e	f	$(d+e+f)$
Categoría C	g	h	i	$(g+h+i)$
Marginal A/B/C	$(a+d+g)$	$(b+e+h)$	$(c+f+i)$	**N**

Cálculo de la frecuencia esperada (f_e) para la celda *a*

$$\frac{(a+d+g)(a+b+c)}{N}$$

Cálculo de la frecuencia esperada (f_e) para la celda *b*

$$\frac{(b+e+h)(a+b+c)}{N}$$

Cálculo de la frecuencia esperada (f_e) para la celda *c*

$$\frac{(c+f+i)(a+b+c)}{N}$$

Cálculo de la frecuencia esperada (f_e) para la celda *d*

$$\frac{(a+d+g)(d+e+f)}{N}$$

Cálculo de la frecuencia esperada (f_e) para la celda *e*

$$\frac{(b+e+h)(d+e+f)}{N}$$

Cálculo de la frecuencia esperada (f_e) para la celda *f*

$$\frac{(c+f+i)(d+e+f)}{N}$$

Cálculo de la frecuencia esperada (f_e) para la celda *g*

$$\frac{(a+d+g)(g+h+i)}{N}$$

Cálculo de la frecuencia esperada (f_e) para la celda *h*

$$\frac{(b+e+h)(g+h+i)}{N}$$

Cálculo de la frecuencia esperada (f_e) para la celda *i*

$$\frac{(c+f+i)(g+h+i)}{N}$$

Como puede observarse, el procedimiento es bien sencillo. Se trata de la razón entre el producto de los marginales de la celda considerada y el total (*N*).

Comparación del valor obtenido y lectura del χ^2

Una vez que se obtiene el resultado de la ecuación, el número arrojado no tienen significación por sí mismo. En realidad lo obtenido es un parámetro para establecer la validez o no de mi hipótesis de trabajo. Si se observa la fórmula de χ^2

$$\chi^2 = \frac{\Sigma\,(f_e - f_o)^2}{f_e}$$

Puede notarse que cuanto mayor es la diferencia entre las frecuencias observadas y las esperadas (f_o *y* f_e *respectivamente*), mayor será el numerador $\sum(f_e - f_o)^2$ y, consecuentemente, también será mayor número que se obtenga. Una mayor diferencia indica, por otra parte, que es menos probable que las mismas se deban puramente al azar (*que es lo que indicaría la* H_o). Por esta razón, cuanto mayor sea el número obtenido, más probable es que podamos rechazar la hipótesis nula.

Se decía que el número obtenido es simplemente un parámetro, es decir, un punto para comparar. ¿Y contra qué lo que se debe comparar? Contra la ***tabla 15***, que es la distribución del χ^2. Para ello debemos considerar los grados de libertad (*gl*) y el nivel de significación (α) que hemos elegido. En los cabezales de las columnas de la ***tabla 15***, encontramos los niveles de significación, y en las filas, los grados de libertad. Cruzando ambos (*columna y fila*) llegamos a una celda con un número determinado.

Tabla 16. Tabla de distribución del Chi-Cuadrado.

Degrees of Freedom	Possibility of Chance Occurrence in Percentage (5% or Less Considered Significant)								
	90%	80%	70%	50%	30%	20%	10%	5% (sig.)	1%
1	0.016	0.064	0.148	0.455	1.074	1.642	2.706	3.841	6.635
2	0.211	0.446	0.713	1.386	2.408	3.219	4.605	5.991	9.210
3	0.584	1.005	1.424	2.366	3.665	4.642	6.251	7.815	11.341
4	1.064	1.649	2.195	3.357	4.878	5.989	7.779	9.488	13.277
5	1.610	2.343	3.000	4.351	6.064	7.289	9.236	11.070	15.086
6	2.204	3.070	3.828	5.348	7.231	8.558	10.645	12.592	16.812
7	2.833	3.822	4.671	6.346	8.383	9.083	12.017	14.067	18.475
8	3.490	4.594	5.527	7.344	9.524	11.030	13.362	15.507	20.090
9	4.168	5.380	6.393	8.343	10.656	12.242	14.684	16.919	21.666

Si el número que nosotros obtenemos mediante el cálculo de χ^2 es igual o mayor (= ó >) al que figura en la tabla, rechazamos la hipótesis nula (H_o) y validamos, en con-secuencia, nuestra hipótesis de trabajo (H_l). Si, por el contrario, es inferior, debemos aceptar la hipótesis nula (H_o), quedando inválida nuestra hipótesis de trabajo (H_l).

Tabla 17. Tabla de resultados - Hipótesis

			Lectura
Número obtenido de χ^2	>	Número de la tabla	Rechazo H_o. Acepto H_l
Número obtenido de χ^2	=	Número de la tabla	Rechazo H_o. Acepto H_l
Número obtenido de χ^2	<	Número de la tabla	Acepto H_o. Rechazo H_l

Anexo 6. Módulos de la arquitectura propuesta, programada en .Net.

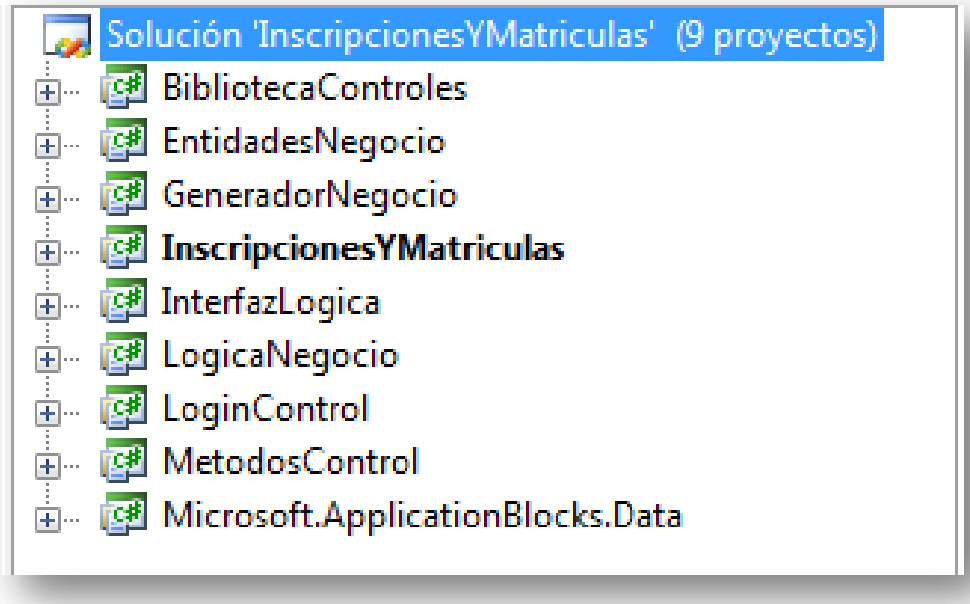

Figura 47. Modelo de arquitectura propuesto, implementada en .NET.
Fuente: investigador.

Anexo 7. Parte de los elementos implementados en el MIU – Front end.

Figura 48. Formulario de acceso al sistema.
Fuente: investigador.

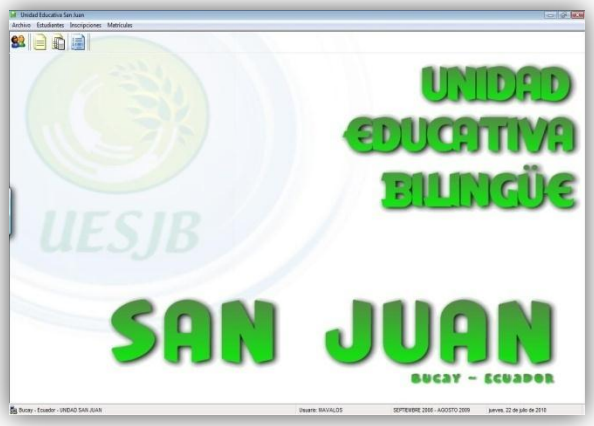

Figura 49. Formulario principal.
Fuente: investigador.

Figura 50. Formulario consola para la creación de un nuevo registro de estudiante.
Fuente: investigador.

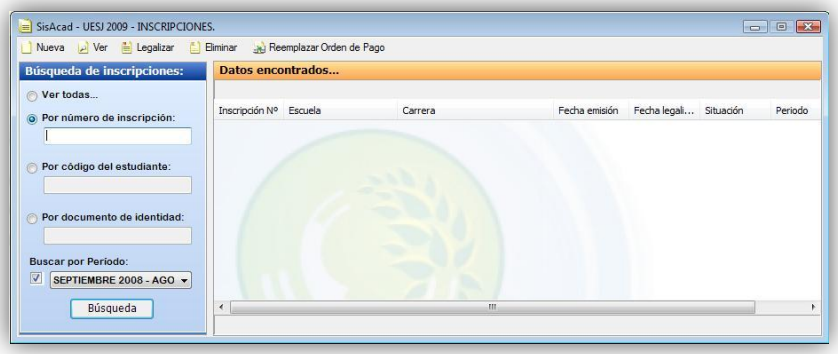

Figura 51. Formulario consola para la inscripción de un estudiante. Fuente: investigador.

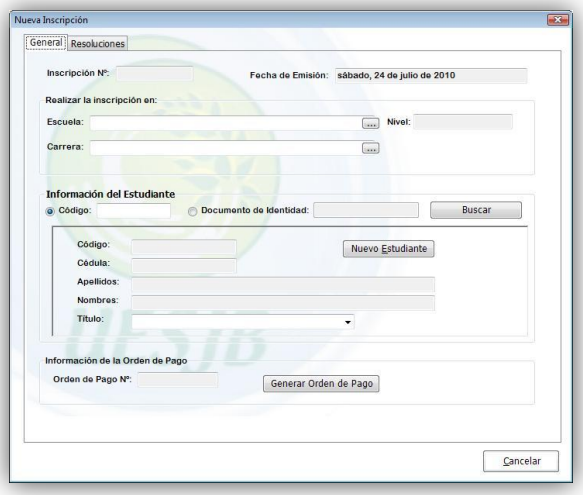

Figura 52. Formulario para la creación de una nueva inscripción. Fuente: investigador.

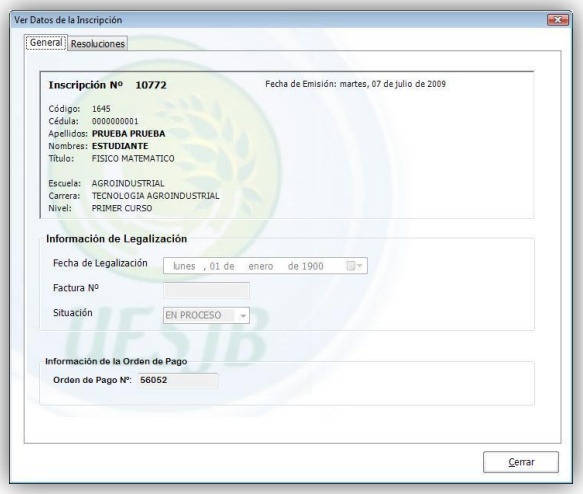

Figura 53. Formulario para visualizar datos de inscripción. Fuente: investigador.

Figura 54. Formulario consola para generar reportes de inscripciones. Fuente: investigador.

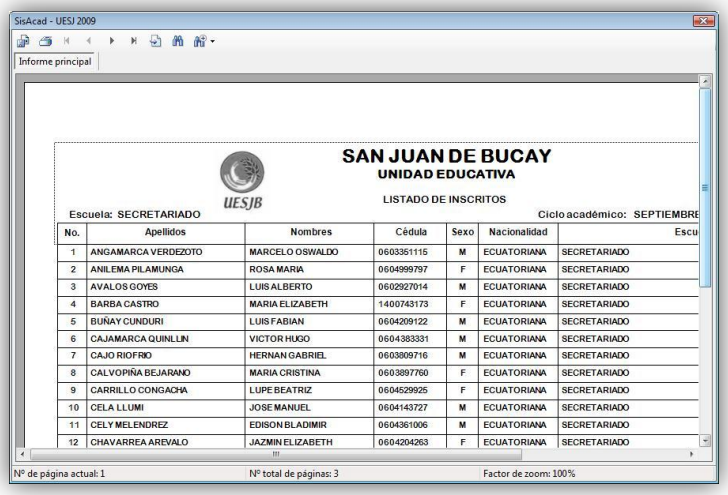

Figura 55. Crystal Report para imprimir listado de inscritos (1). Fuente: investigador.

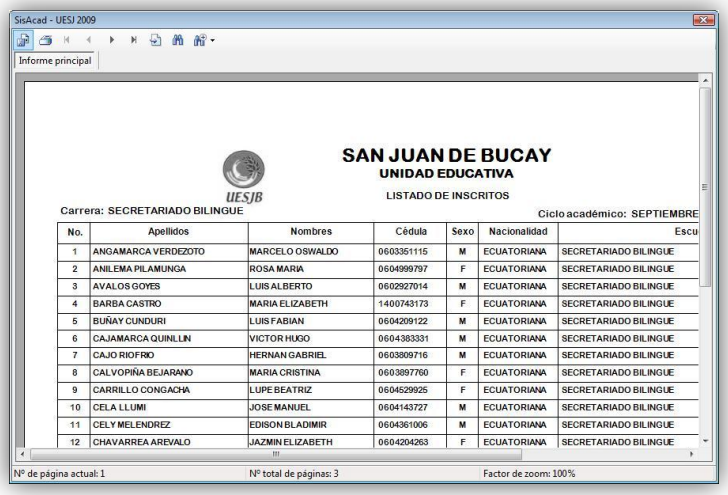

Figura 56. Crystal Report para imprimir listado de inscritos (2). Fuente: investigador.

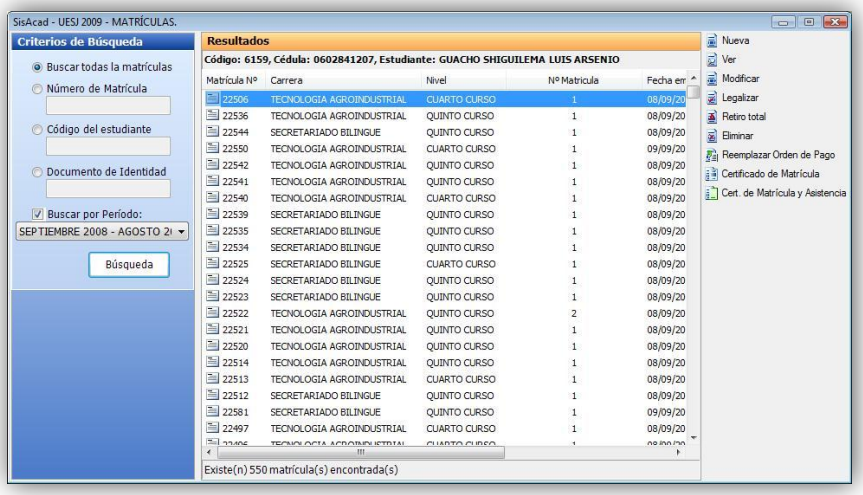

Figura 57. Formulario consola para matriculación de estudiantes.
Fuente: investigador.

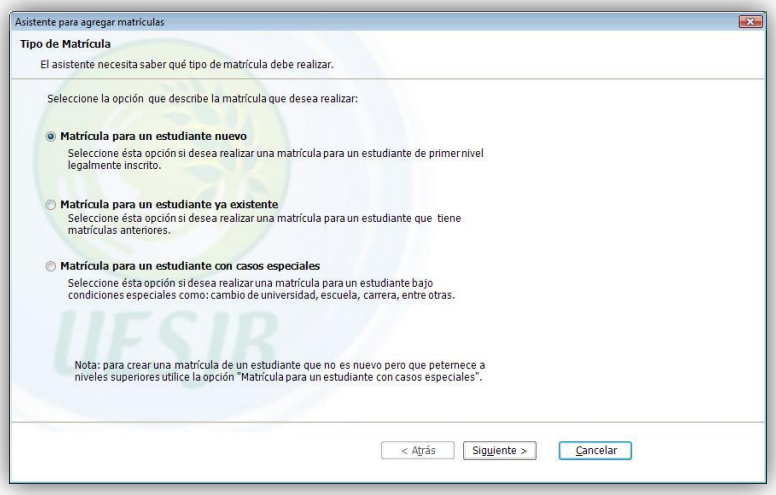

Figura 58. Formulario asistente para matriculación de estudiantes – Paso 1.
Fuente: investigador.

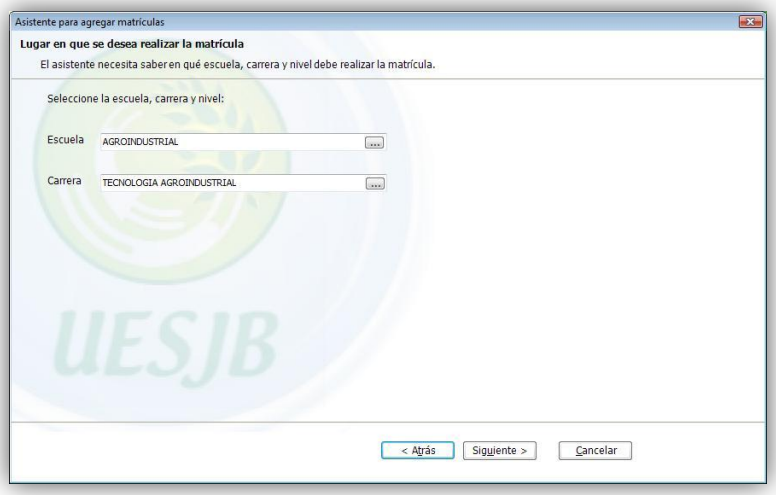

Figura 59. Formulario asistente para matriculación de estudiantes - Paso 2.
Fuente: investigador.

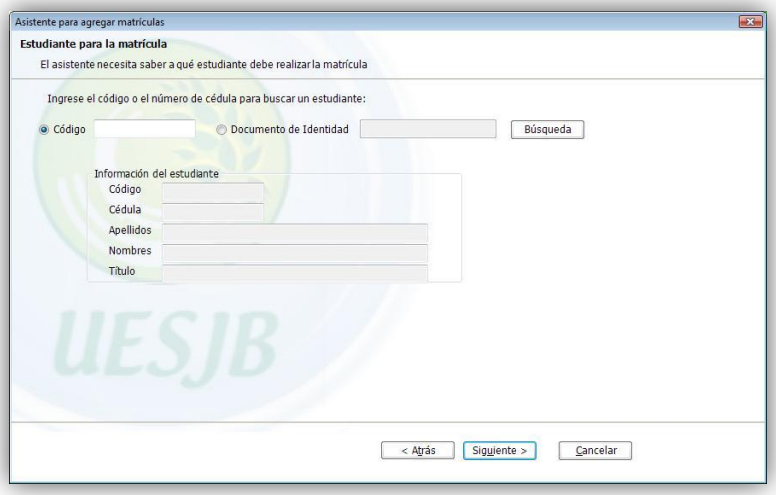

Figura 60. Formulario asistente para matriculación de estudiantes - Paso 3. Fuente: investigador.

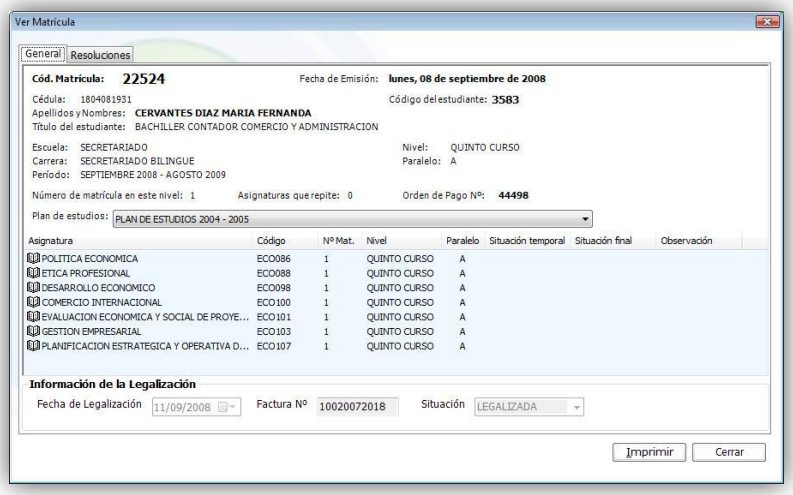

Figura 61. Formulario para la visualización de los datos de matrículas. Fuente: investigador.

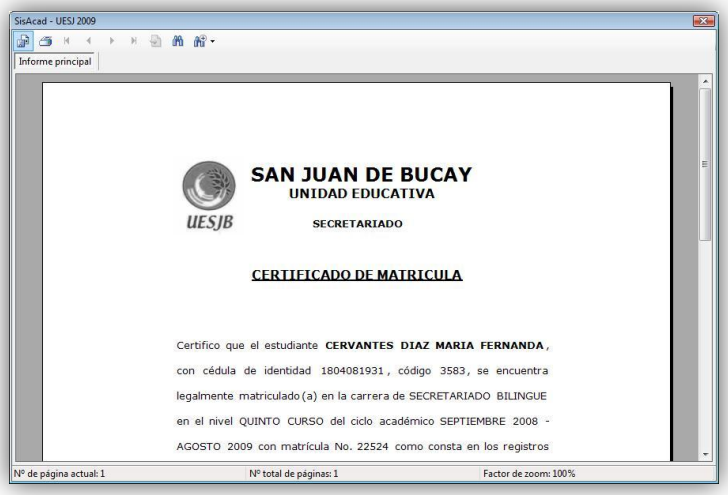

Figura 62. Crystal Report para imprimir reporte de certificados de matrícula. Fuente: investigador.

Figura 63. Crystal Report para imprimir reporte de certificados de matrícula y asistencia. Fuente: investigador.

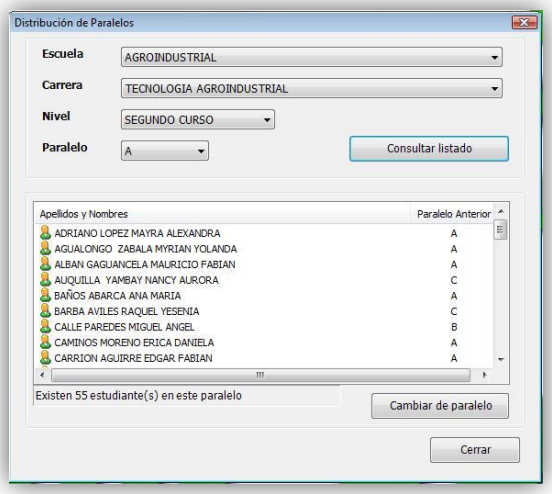

Figura 64. Formulario para consultar listado de estudiantes. Fuente: investigador.

Anexo 8. Parte de los elementos implementados en el MIU – Biblioteca de control.

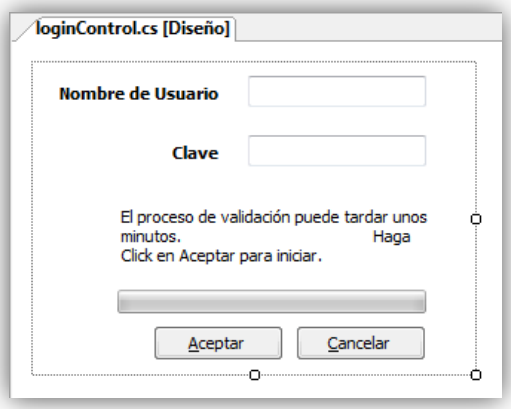

Figura 65. Biblioteca de controles para el control de acceso a usuarios.
Fuente: investigador.

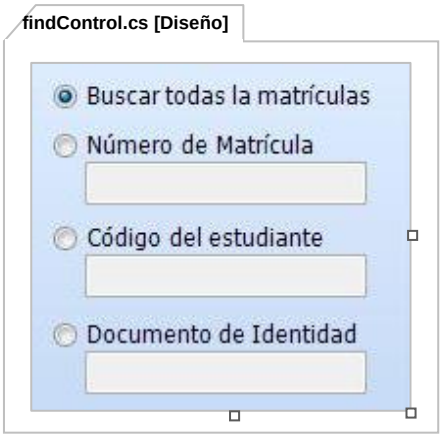

Figura 66. Biblioteca de controles para la búsqueda de registros.
Fuente: investigador.

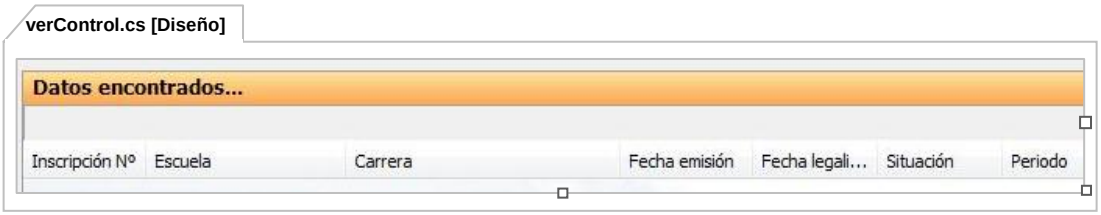

Figura 67. Biblioteca de controles para la visualización de registros.
Fuente: investigador.

Anexo 9. Parte de código implementado en el MIU – Generador de negocios.

```
namespace GeneradorNegocio
{
        ///<summary>
        /// Descripción breve de Class1.
        ///</summary>
        publicclassGeneradorEstudiante
        {
                publicstaticIEstudiante Operaciones()
                {
                        returnnewLEstudiante();

                }// Fin de Operaciones

                publicstaticIUnidades DatosUnidades()
                {
                        returnnewLUnidades();

                }//Fin de DatosUNidades

                publicstaticIInscripciones OperacionesInscripcion()
                {
                        returnnewLInscripciones();

                }//Fin de OperacionesInscripcion()

                publicstaticIMatriculas OperacionesMatricula()
                {
                        returnnewLMatriculas();

                }//Fin de Operaciones Matrículas()

        }
}
```

Anexo 10. Parte de código implementado en el MPS – Interfaz lógica.

```
namespace InterfazLogica
{
///<summary>
/// Clase para la interfaz de la logica que servira para
/// el trabajo de la entidad Estudiante, con sus tablas relacionadas.
///</summary>

publicinterfaceIEstudiante
{
        int DevuelveIdEstudiante(string cedula);
        int DevuelveNumeroEstudiantes(string cedula);
        int DevuelveNumeroEstudiantesXEstudianteID(int estudianteID);
        void BuscarEstudiante(string cedula);
        void BuscarEstudianteSolicitud(string cedula);
        void BuscarEstudianteSolicitud(string cedula,string fecha);
        void BuscarEstudianteSolicitud(string cedula,string fechaI,string fechaF);
        void ActualizarEstudiante();
        void ActualizarEstudianteTitulos();
        void ActualizarEstudianteSolicitud();
        void InsertarEstudiante();
        void InsertarFilaEstudianteDataset(paramsobject[] parametros);
        void InsertarSolicitud();
        void InsertarFilaSolicitudDataset(paramsobject[] parametros);
        void LlenarIntitucionesEducativas();

        //*************Controles para estudiante
        int DevuelveNumeroEstudiantesEnOrden(int estudianteID);

        //*************Controles para títulos
        int DevuelveNumeroTitulosInscripciones(int tituloID);
        int DevuelveNumeroTitulosMatriculas(int tituloID);

        //************Para Buscar estudiante*****************
        void LlenarBuscarEstudianteXEstudianteID(int estudianteID);
        void LlenarBuscarEstudianteXCedula(string cedula);
        void LlenarBuscarEstudianteXCaracteres(string caracteres);

        //****************Para generar record
        void ObtenerDetalleMatriculasEstudiante(int estudianteID, int carreraID);
        void ObtenerDetalleMatriculasEstudiante(int estudianteID, int carreraID, int periodoID);
        void ObtenerDatosEstudiante(int estudianteID, int carreraID);
        decimal calcularPromedioGeneralAprob(int estudianteID, int carreraID);

        //******************Genera Claves para estudiantes
        string ObtenerClaveEstudiante(int estudianteID);

        }
}
```

Anexo 11. Parte de código implementado en el MPS – Lógica de negocios.

```
namespace LogicaNegocio
{
	///<summary>
	/// Descripción breve de LEstudiante.
	///</summary>
	publicclassLEstudiante: IEstudiante
	{
		# region Campos
		// Declaracion de campos
		// Declaración de un string para la conección

		string ConnectionString = Controles.CadenaConexion;

		//*********************************************
		# endregion

		# region Método para recuperar el ID del estudiante
		publicint DevuelveIdEstudiante(string cedula)
		{
			object identificador;
			identificador=SqlHelper.ExecuteScalar(ConnectionString,"spDevuelveIdPrueba",cedula);
			return (int)identificador;

		}//Fin de DevuelveIdEstudiante
		# endregion

		# region Método para recuperar Numero de estudiantes por cédula
		publicint DevuelveNumeroEstudiantes(string cedula)
		{
			object numero;
	numero=SqlHelper.ExecuteScalar(ConnectionString,"spDevuelveNumeroEstudiantes",cedula);
			return (int)numero;

		}//Fin de DevuelveIdEstudiante
		# endregion

		# region Método para recuperar Numero de estudiantes por EstudianteID
		publicint DevuelveNumeroEstudiantesXEstudianteID(int estudianteID)
		{
			object numero;
	numero=SqlHelper.ExecuteScalar(ConnectionString,"spDevuelveNumeroEstudiantesXEstudianteID",estudiant
eID);
			return (int)numero;

		}//Fin de DevuelveIdEstudiante
		# endregion

		# region Método para recuperar el ID del estudiante para solicitud
		publicint DevuelveIdEstudianteSolicitud(string cedula)
		{
			object identificador;
	identificador=SqlHelper.ExecuteScalar(ConnectionString,"spDevuelveIdEstudianteSolicitud",cedula);
			return (int)identificador;

		}//Fin de DevuelveIdEstudianteSolicitud
		# endregion

		# region Buscar estudiante
		// Busca el estudiante con todos sus datos incluso sus títulos
```

```
                    publicvoid BuscarEstudiante(string cedula)
                    {


                    //************* Llena la propiedad dataset ConjuntostudianteTitulos con los datos del Estudiante

            SqlHelper.FillDataset(ConnectionString,"spConsultaEstudiante",EntidadesEstudiante.ConjuntoEstudianteTitulo
s,newstring[]{"ESTUDIANTE"},cedula);

                    //*****************************************************************************

                    // Busco ID del estudiante

                    int estudianteId=DevuelveIdEstudiante(cedula);

                    //*********************************

                    }// Fin de BuscarEstudiante
                    #endregion

                    #region Buscar Estudiante Solicitud Sobrecargado
                    // Busca el nombre del estudiante y sus solicitudes
                    # region Buscar Estudiante Solicitud, cedula
                    publicvoid BuscarEstudianteSolicitud(string cedula)
                    {
                              EntidadesEstudiante.ConjuntoEstudianteSolicitud = newDatasetEstudianteSolicitud();

                    //************* Llena la propiedad dataset ConjuntostudianteSolicitud con los datos del nombre
estudiante

            SqlHelper.FillDataset(ConnectionString,"spConsultaEstudianteNombre",EntidadesEstudiante.ConjuntoEstudia
nteSolicitud ,newstring[]{"VISTA_ESTUDIANTE_NOMBRE"},cedula);

                    //*****************************************************************************

                    // Busco ID del estudiante

                    int estudianteId=DevuelveIdEstudianteSolicitud(cedula);

                    //*********************************

                    // LLeno la propiedad datased ConjuntostudianteSolicitud con los datos de las solicitudes

            SqlHelper.FillDataset(ConnectionString,"spConsultaSolicitudEstudiante",EntidadesEstudiante.ConjuntoEstudia
nteSolicitud ,newstring[]{"ESTUDIANTE_SOLICITUD"},estudianteId);

                    // ***************************************************************

                    }// Fin de BuscarEstudianteSolicitud
                    #endregion

                    # region Buscar Estudiante Nombres Solicitud, Cedula , Fecha
                    publicvoid BuscarEstudianteSolicitud(string cedula,string fecha)
                    {
                              EntidadesEstudiante.ConjuntoEstudianteSolicitud  = newDatasetEstudianteSolicitud();

                    //************* Llena la propiedad dataset ConjuntostudianteSolicitud con los datos del nombre
estudiante

            SqlHelper.FillDataset(ConnectionString,"spConsultaEstudianteNombre",EntidadesEstudiante.ConjuntoEstudia
nteSolicitud ,newstring[]{"VISTA_ESTUDIANTE_NOMBRE"},cedula);
```

```
                    //*****************************************************************************

                    // Busco ID del estudiante

                    int estudianteId=DevuelveIdEstudianteSolicitud(cedula);

                    //*********************************

                    // LLeno la propiedad datased ConjuntostudianteSolicitud con los datos de las solicitudes

            SqlHelper.FillDataset(ConnectionString,"spConsultaSolicitudEstudiantePorFecha",EntidadesEstudiante.Conjun
toEstudianteSolicitud ,newstring[]{"ESTUDIANTE_SOLICITUD"},estudianteId,fecha);

                    // ***************************************************************

                    }// Fin de BuscarEstudianteSolicitud
                    #endregion

                    # region Buscar Estudiante Nombres Solicitud, Cedula , Fecha Inicio, Fecha Fin
                    publicvoid BuscarEstudianteSolicitud(string cedula,string fechaI,string fechaF)
                    {
                            EntidadesEstudiante.ConjuntoEstudianteSolicitud = newDatasetEstudianteSolicitud();

                    //************* Llena la propiedad dataset ConjuntostudianteSolicitud con los datos del nombre
estudiante

            SqlHelper.FillDataset(ConnectionString,"spConsultaEstudianteNombre",EntidadesEstudiante.ConjuntoEstudia
nteSolicitud ,newstring[]{"VISTA_ESTUDIANTE_NOMBRE"},cedula);

                    //*****************************************************************************

                    // Busco ID del estudiante

                    int estudianteId=DevuelveIdEstudianteSolicitud(cedula);

                    //*********************************

                    // LLeno la propiedad datased ConjuntostudianteSolicitud con los datos de las solicitudes

            SqlHelper.FillDataset(ConnectionString,"spConsultaSolicitudEstudiantePorPeriodo",EntidadesEstudiante.Conju
ntoEstudianteSolicitud ,newstring[]{"ESTUDIANTE_SOLICITUD"},estudianteId,fechaI,fechaF);

                    // ***************************************************************


                    }// Fin de BuscarEstudianteSolicitud
                    #endregion

                    #endregion

                    #region  InsertarEstudiante

                    publicvoid InsertarEstudiante()
                    {

                            //Crea los parámetros para llamar a Actualizar dataset
                            string nombreTabla ="ESTUDIANTE";
                            string procedimiento="spInsertarEstudiante";
                            string [] columnasTabla =newstring[36]{   "EstudianteID","Cedula", "Nombre",
"ApellidoPaterno",
```

```
            "ApellidoMaterno","CiudadNacimiento","ProvinciaNacimiento","PaisNacimiento",
            "FechaNacimiento","Nacionalidad","CertificadoMilitar","Sexo","EstadoCivil",
            "Direccion","Telefono1","Telefono2","LugarTrabajo","TelefonoTrabajo","Email",
            "FinanciamientoEstudios","Observaciones","FotoRuta","UsuarioID","FechaTransaccion",
            "DireccionProvincia","DireccionCanton","DireccionOrigen","TrabajoLugar","TrabajoCargo","OcupacionPadre",
"OcupacionMadre","IngresosFamiliares","DomicilioEstado","NumeroIntegrantesHogar","ActividadDeportiva","ActividadCu
ltural"};


            SqlHelper.ActualizarDataset(ConnectionString,EntidadesEstudiante.ConjuntoEstudianteTitulos,

            nombreTabla,columnasTabla,procedimiento);
                        EntidadesEstudiante.ConjuntoEstudianteTitulos.ESTUDIANTE.AcceptChanges();
                }// Fin de InsertarEstudianteDataset
                #endregion


                #region  InsertarFilaEstudianteDataset

                publicvoid InsertarFilaEstudianteDataset(paramsobject[] parametros)
                {


                        // Crea la Fila
                        DataRow
nuevaFilaEstudiante=EntidadesEstudiante.ConjuntoEstudianteTitulos.ESTUDIANTE.NewRow();
                        //**************************************************************************

                        // Asigna valosres a la nueva fila
                        nuevaFilaEstudiante[1]=parametros[0];
                        nuevaFilaEstudiante[2]=parametros[1];
                        nuevaFilaEstudiante[3]=parametros[2];
                        nuevaFilaEstudiante[4]=parametros[3];
                        nuevaFilaEstudiante[5]=parametros[4];
                        nuevaFilaEstudiante[6]=parametros[5];
                        nuevaFilaEstudiante[7]=parametros[6];
                        nuevaFilaEstudiante[8]=parametros[7];
                        nuevaFilaEstudiante[9]=parametros[8];
                        nuevaFilaEstudiante[10]=parametros[9];
                        nuevaFilaEstudiante[11]=parametros[10];
                        nuevaFilaEstudiante[12]=parametros[11];
                        nuevaFilaEstudiante[13]=parametros[12];
                        nuevaFilaEstudiante[14]=parametros[13];
                        nuevaFilaEstudiante[15]=parametros[14];
                        nuevaFilaEstudiante[16]=parametros[15];
                        nuevaFilaEstudiante[17]=parametros[16];
                        nuevaFilaEstudiante[18]=parametros[17];
                        nuevaFilaEstudiante[19]=parametros[18];
                        nuevaFilaEstudiante[20]=parametros[19];
                        nuevaFilaEstudiante[22]=parametros[20];
                        nuevaFilaEstudiante[23]=parametros[21];
                        nuevaFilaEstudiante["FotoRuta"]=parametros[22];

                        nuevaFilaEstudiante["DireccionProvincia"]=parametros[23];
                        nuevaFilaEstudiante["DireccionCanton"]=parametros[24];
                        nuevaFilaEstudiante["DireccionOrigen"]=parametros[25];
                        nuevaFilaEstudiante["TrabajoLugar"]=parametros[26];
                        nuevaFilaEstudiante["TrabajoCargo"]=parametros[27];
                        nuevaFilaEstudiante["OcupacionPadre"]=parametros[28];
                        nuevaFilaEstudiante["OcupacionMadre"]=parametros[29];
                        nuevaFilaEstudiante["IngresosFamiliares"]=parametros[30];
```

```
                                nuevaFilaEstudiante["DomicilioEstado"]=parametros[31];
                                nuevaFilaEstudiante["NumeroIntegrantesHogar"]=parametros[32];
                                nuevaFilaEstudiante["ActividadDeportiva"]=parametros[33];
                                nuevaFilaEstudiante["ActividadCultural"]=parametros[34];

                                //*********************************************************
                                // Asignamos la nueva fila al dataset

        EntidadesEstudiante.ConjuntoEstudianteTitulos.ESTUDIANTE.Rows.Add(nuevaFilaEstudiante);
                                // ************************************************************************

                        }// Fin de InsertarFilaEstudianteDataset
                        #endregion

                        #region  InsertarTitulo Base
                        publicvoid InsertarTitulo()
                        {
                                //Crea los parámetros para llamar a Actualizar dataset
                                string nombreTabla ="ESTUDIANTE_TITULOS";
                                string procedimiento="spInsertarEstudianteTitulo";
                                string [] columnasTabla =newstring[11]{   "TituloID", "EstudianteID",
"InstitucionID","NivelTitulo", "Especialidad",
        "FechaGrado", "Calificacion","Pension",
        "GradoAcademico","Situacion","UsuarioID"};
                                //*********************************************************************
                                //Llama ActualizarDataset

        SqlHelper.ActualizarDataset(ConnectionString,EntidadesEstudiante.ConjuntoEstudianteTitulos,nombreTabla,c
olumnasTabla,procedimiento);
                                //*********************************************************************


        EntidadesEstudiante.ConjuntoEstudianteTitulos.ESTUDIANTE_TITULOS.AcceptChanges();
                        }// Fin de InsertarTitulo
                        #endregion

                        #region  InsertarFilaTitulosDataset

                        publicvoid InsertarFilaTitulosDataset(paramsobject[] parametros)
                        {
                                // Ccrea una fila en el Dataset
                                DataRow nuevaFilaTitulo =
EntidadesEstudiante.ConjuntoEstudianteTitulos.ESTUDIANTE_TITULOS.NewRow();
                                //*************************************************************************

                                // Asigna  valosres a la nueva fila
                                nuevaFilaTitulo["EstudianteID"]=parametros[0];
                                nuevaFilaTitulo["InstitucionID"]=parametros[1];
                                nuevaFilaTitulo["NivelTitulo"]=parametros[2];
                                nuevaFilaTitulo["Especialidad"]=parametros[3];
                                nuevaFilaTitulo["FechaGrado"]=parametros[4];
                                nuevaFilaTitulo["Calificacion"]=parametros[5];
                                nuevaFilaTitulo["Pension"]=float.Parse(parametros[6].ToString());
                                nuevaFilaTitulo["GradoAcademico"]=parametros[7];
                                nuevaFilaTitulo["Situacion"]=parametros[8];
                                nuevaFilaTitulo["UsuarioID"]=parametros[9];
                                nuevaFilaTitulo["FechaTransaccion"]=parametros[10];
                                //*********************************************************

                                // Asignamos la nueva fila al dataset
```

```
            EntidadesEstudiante.ConjuntoEstudianteTitulos.ESTUDIANTE_TITULOS.Rows.Add(nuevaFilaTitulo);
                        // ********************************************************************************

                    }// Fin de InsertarFilaTitulosDataset
                    #endregion

                    #region  InsertarSolicitud Base
                    publicvoid InsertarSolicitud()
                    {
                        //Crea los parámetros para llamar a Actualizar dataset
                        string nombreTabla ="ESTUDIANTE_SOLICITUD";
                        string procedimiento="spInsertarEstudianteSolicitud";
                        string [] columnasTabla =newstring[5]{"SolicitudID",
"EstudianteID","Fecha","Descripcion","UsuarioID"};
                        //*****************************************************************
                        //Llama ActualizarDataset

            SqlHelper.ActualizarDataset(ConnectionString,EntidadesEstudiante.ConjuntoEstudianteSolicitud,nombreTabla,
columnasTabla,procedimiento);
                        //*****************************************************************
                    }// Fin de InsertarSolicitud
                    #endregion

                    #region  InsertarFilaSolicitudDataset

                    publicvoid InsertarFilaSolicitudDataset(paramsobject[] parametros)
                    {
                        // Ccrea una fila en el Dataset
                        DataRow nuevaFilaSolicitud =
EntidadesEstudiante.ConjuntoEstudianteSolicitud.ESTUDIANTE_SOLICITUD.NewRow();
                        //********************************************************************************

                        // Asigna  valosres a la nueva fila
                        nuevaFilaSolicitud["EstudianteID"]=parametros[0];
                        nuevaFilaSolicitud["Fecha"]=parametros[1];
                        nuevaFilaSolicitud["Descripcion"]=parametros[2];
                        nuevaFilaSolicitud["UsuarioID"]=parametros[3];
                        nuevaFilaSolicitud["FechaTransaccion"]=parametros[4];
                        //*************************************************************

                        // Asignamos la nueva fila al dataset

            EntidadesEstudiante.ConjuntoEstudianteSolicitud.ESTUDIANTE_SOLICITUD.Rows.Add(nuevaFilaSolicitud);
                        // ********************************************************************************

                    }// Fin de InsertarFilaSolicitudDataset
                    #endregion


                    #region  Actualizar Estudiante
                    publicvoid ActualizarEstudiante()
                    {
                        //Crear los parámetros para llamar Actualizar Dataset
                        string nombreTabla ="ESTUDIANTE";
                        string procedimientoA="spActualizarEstudiante";
                        string procedimientoI="spInsertarEstudiante";
                        string procedimientoE="spBorrarEstudiante";

                        string [] columnasTablaA =newstring[36]{  "EstudianteID","Cedula", "Nombre",
"ApellidoPaterno",
```

```
"ApellidoMaterno","CiudadNacimiento","ProvinciaNacimiento","PaisNacimiento",
"FechaNacimiento","Nacionalidad","CertificadoMilitar","Sexo","EstadoCivil",
"Direccion","Telefono1","Telefono2","LugarTrabajo","TelefonoTrabajo","Email",
                "FinanciamientoEstudios","Observaciones","FotoRuta","UsuarioID","FechaTransaccion",
"DireccionProvincia","DireccionCanton","DireccionOrigen","TrabajoLugar","TrabajoCargo","OcupacionPadre",
"OcupacionMadre","IngresosFamiliares","DomicilioEstado","NumeroIntegrantesHogar","ActividadDeportiva","A
ctividadCultural"};

                        string [] columnasTablaE =newstring[1]{"EstudianteID"};


                        //Llamar a Actualizar dataset
                        SqlHelper.ActualizarDataset(
        ConnectionString,EntidadesEstudiante.ConjuntoEstudianteTitulos,

        nombreTabla,columnasTablaA,columnasTablaA,columnasTablaE,

        procedimientoA,procedimientoI,procedimientoE);

                        EntidadesEstudiante.ConjuntoEstudianteTitulos.ESTUDIANTE.AcceptChanges();
                }
                #endregion

                #region  Actualizar Estudiante Solicitud
                publicvoid ActualizarEstudianteSolicitud()
                {
                        //Crear los parámetros para llamar Actualizar Dataset
                        string nombreTabla ="ESTUDIANTE_SOLICITUD";
                        string procedimientoA="spActualizarEstudianteSolicitud";
                        string procedimientoI="spInsertarEstudianteSolicitud";
                        string procedimientoE="spBorrarEstudianteSolicitud";

                        string [] columnasTablaA =newstring[4]{"SolicitudID","Fecha","Descripcion","UsuarioID"};

                        string [] columnasTablaI =newstring[5]{"SolicitudID",
"EstudianteID","Fecha","Descripcion","UsuarioID"};

                        string [] columnasTablaE =newstring[1]{"SolicitudID"};

                        //Llamar a Actualizar dataset
                        SqlHelper.ActualizarDataset(
        ConnectionString,EntidadesEstudiante.ConjuntoEstudianteSolicitud,nombreTabla,

        columnasTablaA,columnasTablaI,columnasTablaE,

        procedimientoA,procedimientoI,procedimientoE);

        EntidadesEstudiante.ConjuntoEstudianteSolicitud.ESTUDIANTE_SOLICITUD.AcceptChanges();
                }// Fin de ActualizarEstudianteSolicitud
                #endregion


                # region Llenar Institucion educativa: extrae todas las intituciones educativas de la base
                publicvoid LlenarIntitucionesEducativas()
                {
                        //Instancio la Propiedad ConjuntoEstudiantes Titulos
                        //EntidadesEstudiante.ConjuntoEstudianteTitulos=new DatasetEstudianteTitulos();

                        //************ Llena la propiedad dataset ConjuntostudianteTitulos con los datos Intitucion
```

```
            SqlHelper.FillDataset(ConnectionString,"spConsultaInstitucionEducativa",EntidadesEstudiante.ConjuntoEstudi
anteTitulos,newstring[]{"INSTITUCION_EDUCATIVA"});
                                    //*****************************************************************************

                        }// Fin de BuscarEstudiante
                        #endregion

                        //*************Control para estudiantes ****************
                        # region Método para recuperar Numero de Estudiantes en Orden de Pago
                        publicint DevuelveNumeroEstudiantesEnOrden(int estudianteID)
                        {
                                    object numero;

            numero=SqlHelper.ExecuteScalar(ConnectionString,"spDevuelveNumeroEstudiantesEnOrden_AC",estudianteI
D);
                                    return (int)numero;

                        }//Fin de DevuelveNumeroEstudiantesEnOrden
                        # endregion

                        //*************Controles para titulos*********************

                        # region Método para recuperar Numero de Títulos en Inscripciones
                        publicint DevuelveNumeroTitulosInscripciones(int tituloID)
                        {
                                    object numero;

            numero=SqlHelper.ExecuteScalar(ConnectionString,"spDevuelveNumeroTitulosEnInscripciones_AC",tituloID);
                                    return (int)numero;

                        }//Fin de DevuelveNumeroTitulosInscripciones
                        # endregion

                        # region Método para recuperar Numero de Títulos en Matrículas
                        publicint DevuelveNumeroTitulosMatriculas(int tituloID)
                        {
                                    object numero;

            numero=SqlHelper.ExecuteScalar(ConnectionString,"spDevuelveNumeroTitulosEnMatricula_AC",tituloID);
                                    return (int)numero;

                        }//Fin de DevuelveNumeroTitulosInscripciones
                        # endregion


                        #region DataReader Datos Estudiante X Cedula
                        publicvoid LlenarDatosEstudianteXCedula(string cedula)
                        {

            EntidadesEstudiante.DatosEstudianteDR=SqlHelper.ExecuteReader(ConnectionString,"spConsultarDatosEstu
dianteDR",cedula);
                        }
                        #endregion

                        //*****************ParaBuscar*****************************

                        #region DataReader Buscar Estudiante X EstudianteID
                        publicvoid LlenarBuscarEstudianteXEstudianteID(int estudianteID)
                        {
```

```
            EntidadesEstudiante.BuscarEstudianteDR=SqlHelper.ExecuteReader(ConnectionString,"spConsultaBuscarEst
udianteXEstudianteIDDR",estudianteID);
                        }
                        #endregion

                        #region DataReader Buscar Estudiante X Cedula
                        publicvoid LlenarBuscarEstudianteXCedula(string cedula)
                        {

            EntidadesEstudiante.BuscarEstudianteDR=SqlHelper.ExecuteReader(ConnectionString,"spConsultaBuscarEst
udianteXCedulaDR",cedula);
                        }
                        #endregion

                        #region DataReader Buscar Estudiante X Caracteres
                        publicvoid LlenarBuscarEstudianteXCaracteres(string caracteres)
                        {

            EntidadesEstudiante.BuscarEstudianteDR=SqlHelper.ExecuteReader(ConnectionString,"spConsultaBuscarEst
udianteXCaracteresDR",caracteres);
                        }
                        #endregion


                        #region void ObtenerDetalleMatriculasEstudiante(string cedula, int carrera)
                        ///<summary>
                        /// Obtiene el record académico de un estudiante en una carrera
                        ///</summary>
                        ///<param name="estudianteID"></param>
                        ///<param name="carreraID"></param>
                        publicvoid ObtenerDetalleMatriculasEstudiante(int estudianteID, int carreraID)
                        {
                                    EntidadesEstudiante.DSDetalleRegistroAcademico =
newDataSetDetalleRegistroAcademico();


            SqlHelper.FillDataset(Controles.CadenaConexion,"F2_spConsultarDetalleMatriculasEstudiante_PO",
                                                EntidadesEstudiante.DSDetalleRegistroAcademico,
newstring[]{"VISTA_DETALLE_REGISTRO_ACADEMICO_ESTUDIANTE_PO"},estudianteID,carreraID);
                        }

                        #endregion

                        #region void ObtenerMasterLibretaCalificaciones(string EstudianteID, int carreraID, periodoID)
                        ///<summary>
                        /// Obtiene el record académico de un estudiante en una carrera en un período dado.
                        ///
                        ///
                        ///</summary>
                        ///<param name="estudianteID"></param>
                        ///<param name="carreraID"></param>
                        publicvoid ObtenerMasterLibretaCalificaciones(int estudianteID, int carreraID, int periodoID)
                        {
                                    EntidadesEstudiante.DatosEstudianteDR =
SqlHelper.ExecuteReader(Controles.CadenaConexion,"F3_spCosultarDatosEstudianteCalificaciones_PO",estudianteID,
carreraID,periodoID);
                        }

                        #endregion
```

```
//*****************************Claves para estudiantes*****************

                        #region void ObtenerClaveEstudiante(int EstudianteID)
                                ///<summary>
                                /// Obtiene la clave para acceso al consultas desde internet
                                ///</summary>
                                ///<param name="estudianteID"></param>
                                publicstring ObtenerClaveEstudiante(int estudianteID)
                                {
                                        object identificador;

            identificador=SqlHelper.ExecuteScalar(ConnectionString,"F2_spConsultarClaveEstudiante_PO",estudianteID);
                                        return identificador.ToString();


                                }

                        #endregion
                        }
}
```

Anexo 12. Parte de código del MAD.

```
namespace Microsoft.ApplicationBlocks.Data
{
	///<summary>
	/// The SqlHelper class is intended to encapsulate high performance, scalable best practices for
	/// common uses of SqlClient
	///</summary>
publicsealedclassSqlHelper
   {
      #region private utility methods & constructors

// Since this class provides only static methods, make the default constructor private to prevent
// instances from being created with "new SqlHelper()"
private SqlHelper() {}

///<summary>
/// This method is used to attach array of SqlParameters to a SqlCommand.
///
/// This method will assign a value of DbNull to any parameter with a direction of
/// InputOutput and a value of null.
///
/// This behavior will prevent default values from being used, but
/// this will be the less common case than an intended pure output parameter (derived as InputOutput)
/// where the user provided no input value.
///</summary>
///<param name="command">The command to which the parameters will be added</param>
///<param name="commandParameters">An array of SqlParameters to be added to command</param>
privatestaticvoid AttachParameters(SqlCommand command, SqlParameter[] commandParameters)
      {
                              if( command == null ) thrownewArgumentNullException( "command" );
                              if( commandParameters != null )
                              {
                                    foreach (SqlParameter p in commandParameters)
                                    {
                                          if( p != null )
                                          {
                                                // Check for derived output value with no value assigned
                                                if ( ( p.Direction == ParameterDirection.InputOutput ||
                                                      p.Direction == ParameterDirection.Input ) &&
                                                      (p.Value == null))
                                                {
                                                      p.Value = DBNull.Value;
                                                }
                                                command.Parameters.Add(p);
                                          }
                                    }
                              }
      }

///<summary>
/// This method assigns dataRow column values to an array of SqlParameters
///</summary>
///<param name="commandParameters">Array of SqlParameters to be assigned values</param>
///<param name="dataRow">The dataRow used to hold the stored procedure's parameter values</param>
privatestaticvoid AssignParameterValues(SqlParameter[] commandParameters, DataRow dataRow)
      {
if ((commandParameters == null) || (dataRow == null))
         {
// Do nothing if we get no data
return;
```

```
        }

                                int i = 0;
// Set the parameters values
foreach(SqlParameter commandParameter in commandParameters)
        {
                                        // Check the parameter name
                                        if( commandParameter.ParameterName == null ||
                                                commandParameter.ParameterName.Length <= 1 )
                                                thrownewException(
                                                        string.Format(
                                                                "Please provide a valid parameter name on the
parameter #{0}, the ParameterName property has the following value: '{1}'.",
                                                                i, commandParameter.ParameterName ) );
if (dataRow.Table.Columns.IndexOf(commandParameter.ParameterName.Substring(1)) != -1)
                commandParameter.Value = dataRow[commandParameter.ParameterName.Substring(1)];
                                        i++;
        }
      }

///<summary>
/// This method assigns an array of values to an array of SqlParameters
///</summary>
///<param name="commandParameters">Array of SqlParameters to be assigned values</param>
///<param name="parameterValues">Array of objects holding the values to be assigned</param>
privatestaticvoid AssignParameterValues(SqlParameter[] commandParameters, object[] parameterValues)
      {
if ((commandParameters == null) || (parameterValues == null))
        {
// Do nothing if we get no data
return;
        }

// We must have the same number of values as we pave parameters to put them in
if (commandParameters.Length != parameterValues.Length)
        {
thrownewArgumentException("Parameter count does not match Parameter Value count.");
        }

// Iterate through the SqlParameters, assigning the values from the corresponding position in the
// value array
for (int i = 0, j = commandParameters.Length; i < j; i++)
        {
// If the current array value derives from IDbDataParameter, then assign its Value property
                                        if (parameterValues[i] isIDbDataParameter)
                                        {
                                                IDbDataParameter paramInstance =
(IDbDataParameter)parameterValues[i];
                                                if( paramInstance.Value == null )
                                                {
                                                        commandParameters[i].Value = DBNull.Value;
                                                }
                                                else
                                                {
                                                        commandParameters[i].Value = paramInstance.Value;
                                                }
                                        }
                                        elseif (parameterValues[i] == null)
                                        {
                                                commandParameters[i].Value = DBNull.Value;
                                        }
```

```
                                        else
                                        {
                                                commandParameters[i].Value = parameterValues[i];
                                        }
            }
        }

///<summary>
/// This method opens (if necessary) and assigns a connection, transaction, command type and parameters
/// to the provided command
///</summary>
///<param name="command">The SqlCommand to be prepared</param>
///<param name="connection">A valid SqlConnection, on which to execute this command</param>
///<param name="transaction">A valid SqlTransaction, or 'null'</param>
///<param name="commandType">The CommandType (stored procedure, text, etc.)</param>
///<param name="commandText">The stored procedure name or T-SQL command</param>
///<param name="commandParameters">An array of SqlParameters to be associated with the command or 'null' if no
parameters are required</param>
///<param name="mustCloseConnection"><c>true</c> if the connection was opened by the method, otherwose is
false.</param>
privatestaticvoid PrepareCommand(SqlCommand command, SqlConnection connection, SqlTransaction transaction,
CommandType commandType, string commandText, SqlParameter[] commandParameters, outbool
mustCloseConnection )
        {
                                if( command == null ) thrownewArgumentNullException( "command" );
                                if( commandText == null || commandText.Length == 0 ) thrownewArgumentNullException(
"commandText" );

// If the provided connection is not open, we will open it
                                if (connection.State != ConnectionState.Open)
                                {
                                        mustCloseConnection = true;
                                        connection.Open();
                                }
                                else
                                {
                                        mustCloseConnection = false;
                                }

// Associate the connection with the command
            command.Connection = connection;

// Set the command text (stored procedure name or SQL statement)
            command.CommandText = commandText;

// If we were provided a transaction, assign it
if (transaction != null)
            {
                                        if( transaction.Connection == null ) thrownewArgumentException( "The
transaction was rollbacked or commited, please provide an open transaction.", "transaction" );
                command.Transaction = transaction;
            }

// Set the command type
            command.CommandType = commandType;

// Attach the command parameters if they are provided
if (commandParameters != null)
            {
                AttachParameters(command, commandParameters);
            }
```

```
return;
        }

        #endregion private utility methods & constructors

        #region ExecuteNonQuery

///<summary>
/// Execute a SqlCommand (that returns no resultset and takes no parameters) against the database specified in
/// the connection string
///</summary>
///<remarks>
/// e.g.:
///  int result = ExecuteNonQuery(connString, CommandType.StoredProcedure, "PublishOrders");
///</remarks>
///<param name="connectionString">A valid connection string for a SqlConnection</param>
///<param name="commandType">The CommandType (stored procedure, text, etc.)</param>
///<param name="commandText">The stored procedure name or T-SQL command</param>
///<returns>An int representing the number of rows affected by the command</returns>
publicstaticint ExecuteNonQuery(string connectionString, CommandType commandType, string commandText)
        {
// Pass through the call providing null for the set of SqlParameters
return ExecuteNonQuery(connectionString, commandType, commandText, (SqlParameter[])null);
        }

///<summary>
/// Execute a SqlCommand (that returns no resultset) against the database specified in the connection string
/// using the provided parameters
///</summary>
///<remarks>
/// e.g.:
///  int result = ExecuteNonQuery(connString, CommandType.StoredProcedure, "PublishOrders", new
SqlParameter("@prodid", 24));
///</remarks>
///<param name="connectionString">A valid connection string for a SqlConnection</param>
///<param name="commandType">The CommandType (stored procedure, text, etc.)</param>
///<param name="commandText">The stored procedure name or T-SQL command</param>
///<param name="commandParameters">An array of SqlParamters used to execute the command</param>
///<returns>An int representing the number of rows affected by the command</returns>
publicstaticint ExecuteNonQuery(string connectionString, CommandType commandType, string commandText,
paramsSqlParameter[] commandParameters)
        {
                                        if( connectionString == null || connectionString.Length == 0 )
thrownewArgumentNullException( "connectionString" );

// Create & open a SqlConnection, and dispose of it after we are done
using (SqlConnection connection = newSqlConnection(connectionString))
            {
                connection.Open();

// Call the overload that takes a connection in place of the connection string
return ExecuteNonQuery(connection, commandType, commandText, commandParameters);
}
        }
```

Anexo 13. Parte de los elementos implementados en el MDC – Entidades de negocios.

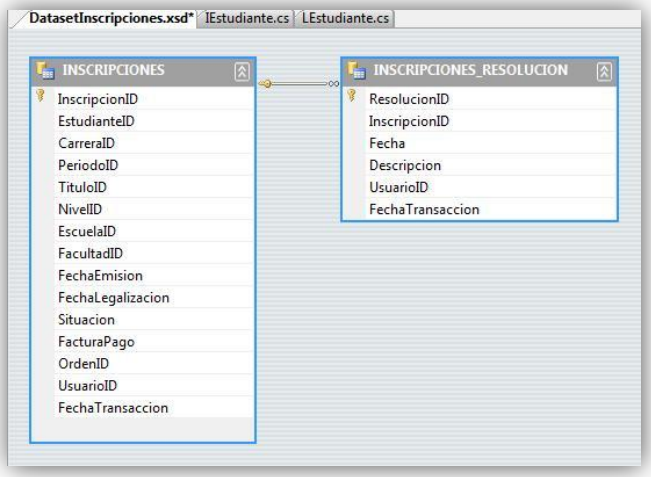

Figura 68. Dataset Inscripciones.
Fuente: investigador.

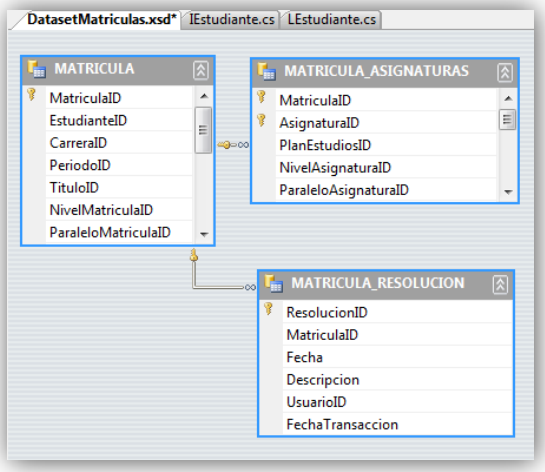

Figura 69. Dataset Matrículas.
Fuente: investigador.

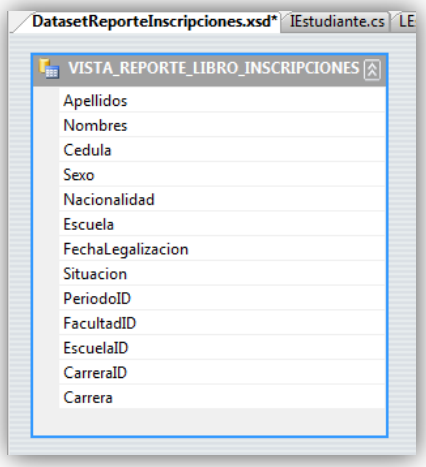

Figura 70. Dataset Reporte de Inscripciones.
Fuente: investigador.

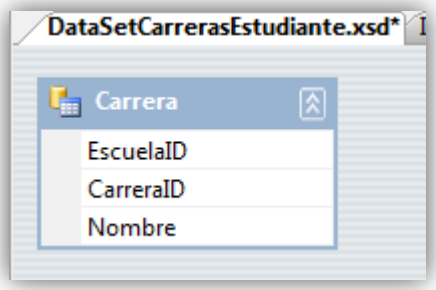

Figura 71. Dataset Carrera de Estudiantes.
Fuente: investigador.

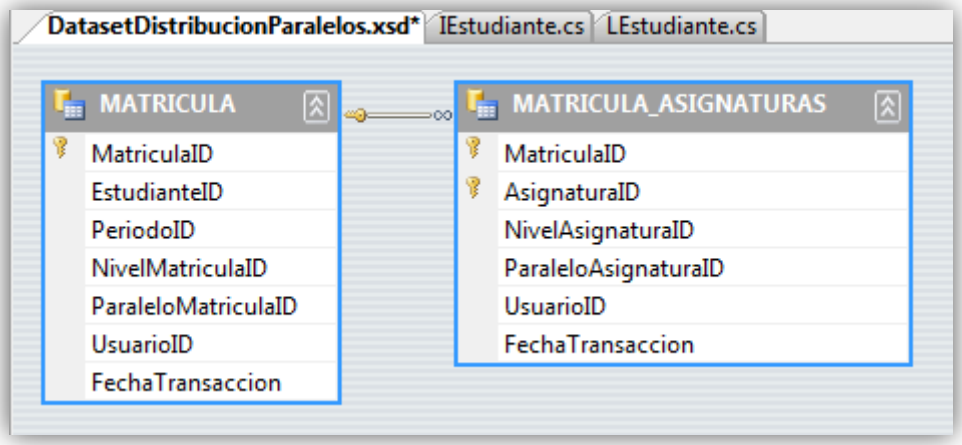

Figura 72. Dataset Distribución de paralelos.
Fuente: investigador.

Anexo 14. Parte de código implementado en el MDS – Métodos de Control.

```
namespace MetodosControl
{
	///<summary>
	/// Descripción breve de Class1.
	///</summary>
	publicclassControles
	{
		#region campos
		//fecha que debe establecerse en el dataset cuando no se introduce una fecha
		privatestaticstring fechaPredeterminada = "01/01/1900";
		privatestaticbool prerrequisitosAsignaturaNueva;

		//Columna de ordenación para ordenar ListView
		privatestaticint sortColumn = -1;
		privatestaticDatasetPerfilesUsuario dsPerfilUsuario;
		#endregion


		#region propiedad FechaPredeterminada
		publicstaticstring FechaPredeterminda
		{
			get
			{
				return fechaPredeterminada;
			}
		}
		#endregion


		#region propiedad PrerrequisitosAsignaturaNueva
		publicstaticbool PrerrequisitosAsignaturaNueva
		{
			get
			{
				return prerrequisitosAsignaturaNueva;
			}

			set
			{
				prerrequisitosAsignaturaNueva = value;
			}

		}

		#endregion

		#region propiedad CadenaConexion

	privatestaticstring cadenaConexion = "SERVER = (local); DATABASE = UnidadDB; INTEGRATED SECURITY
= SSPI";

		publicstaticstring CadenaConexion
		{
			get
			{
				return cadenaConexion;
			}
		}
```

```
#endregion

#region propiedad Perfiles de Usuario
publicstaticDatasetPerfilesUsuario DSPerfilUsuario
{
	get
	{
		return dsPerfilUsuario;
	}

	set
	{
		dsPerfilUsuario = value;
	}

}

#endregion


#region método ControlFechas
publicstaticbool ControlFechas(string cadenaFecha)
{
	string mensajeError;

	//string a;

	try
	{
		System.IFormatProvider formatoRegional =
			new System.Globalization.CultureInfo("es-EC", true);

		string[] formatosEsperados = {"G", "g", "f" ,"F", "D", "d"};

		DateTime fechaEcuatoriana =
			DateTime.ParseExact(cadenaFecha, formatosEsperados,
			formatoRegional, DateTimeStyles.AllowWhiteSpaces);

		returntrue;

	}//cierra try

	catch(System.Exception captura)
	{
		mensajeError = captura.Message + "\n El formato esperado es:  dd/mm/aaaa";
		MessageBox.Show(mensajeError, "SisAcad - UESJ 2010",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
		returnfalse;
	}//cierra catch

}//fin ControlFechas

#endregion

#region método ControlCamposRequeridos
publicstaticbool ControlRequeridos(string contenidoCampo)
{
	string mensajeError;

	try
```

```
                {
                        if(contenidoCampo.Length == 0)
                        {
                                thrownewArgumentNullException();
                        }
                        else
                                returntrue;

                }//cierra try

                catch(System.Exception error)
                {
                        mensajeError = error.Message + "\nSe ha omitido el ingreso de un dato
requerido";
MessageBox.Show(mensajeError, "SisAcad - UESJ 2010", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                        returnfalse;
                }//cierra catch

        }//fin ControlCamposRequeridos

        #endregion

        #region método ControlCamposNumeroentero

        publicstaticbool ControlNumeroEntero(string contenidoCampo)
        {
                string mensajeError;
                int conversion;

                try
                {
                        conversion = System.Int32.Parse(contenidoCampo);
                        returntrue;
                }//cierra try

                catch(System.FormatException error)
                {
                        mensajeError = error.Message + "\nDebe ingresar un número entero";
MessageBox.Show(mensajeError, "SisAcad - UESJ 2010", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                        returnfalse;
                }//cierra catch

        }//fin ControlCamposRequeridos

        #endregion

        #region método ControlCamposNumeroentero Sin Mensage

        publicstaticbool ControlNumeroEnteroSinMensage(string contenidoCampo)
        {

                int conversion;

                try
                {
                        conversion = System.Int32.Parse(contenidoCampo);
                        returntrue;
                }//cierra try

                catch
                {
```

```
                    returnfalse;
                }//cierra catch

            }//fin ControlCamposRequeridos

            #endregion


            #region método ControlDecimal

            publicstaticbool ControlDecimal(string cadenaDecimal)
            {
                string mensajeError;
                try
                {

                    //Declara una variable de tipo formato de número
                    System.Globalization.NumberFormatInfo formatoNumero=
newNumberFormatInfo();

                    //Personaliza el formato del número

                    formatoNumero.NumberDecimalDigits=2;
                    formatoNumero.NumberDecimalSeparator=",";
                    formatoNumero.NumberGroupSeparator=".";
                    int[] grupo =newint[1] {3};

                    formatoNumero.NumberGroupSizes=grupo;

                    //Combierte a la cadena en un tipo decimal
                    Double
numeroDecimal=Double.Parse(cadenaDecimal,System.Globalization.NumberStyles.Float,formatoNumero);

                    returntrue;

                }//cierra try

                catch(System.Exception captura)
                {
                    mensajeError = captura.Message + "\n El formato esperado debe ser como el
ejemplo:  222,20";
MessageBox.Show(mensajeError, "SisAcad - UESJ 2010", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    returnfalse;
                }//cierra catch

            }//fin ControlDecimal

            #endregion

            #region método ControlDecimalSinMensaje

            publicstaticbool ControlDecimalSinMensaje(string cadenaDecimal)
            {


                try
                {

                    //Declara una variable de tipo formato de número
                    System.Globalization.NumberFormatInfo formatoNumero=
newNumberFormatInfo();
```

```
                                        //Personaliza el formato del número

                                        formatoNumero.NumberDecimalDigits=2;
                                        formatoNumero.NumberDecimalSeparator=",";
                                        formatoNumero.NumberGroupSeparator=".";
                                        int[] grupo =newint[1] {3};

                                        formatoNumero.NumberGroupSizes=grupo;

                                        //Combierte a la cadena en un tipo decimal
                                        Double
numeroDecimal=Double.Parse(cadenaDecimal,System.Globalization.NumberStyles.Float,formatoNumero);

                                        returntrue;

                                }//cierra try

                                catch
                                {
                                        returnfalse;
                                }//cierra catch

                        }//fin ControlDecimal

                        #endregion

                        #region método Convierte Decimal

                        publicstaticfloat ConvierteDecimal(string cadenaDecimal)
                        {
                                string mensajeError;
                                try
                                {

                                        //Declara una variable de tipo formato de número
                                        System.Globalization.NumberFormatInfo formatoNumero=
newNumberFormatInfo();

                                        //Personaliza el formato del número

                                        formatoNumero.NumberDecimalDigits=2;
                                        formatoNumero.NumberDecimalSeparator=",";
                                        formatoNumero.NumberGroupSeparator=".";
                                        int[] grupo =newint[1] {3};

                                        formatoNumero.NumberGroupSizes=grupo;

                                        //Combierte a la cadena en un tipo decimal
                                        Double
numeroDecimal=Double.Parse(cadenaDecimal,System.Globalization.NumberStyles.Float,formatoNumero);

                                        return (float)numeroDecimal;

                                }//cierra try

                                catch(System.Exception captura)
                                {
                                        mensajeError = captura.Message + "\n El formato esperado debe ser como el
ejemplo:  2,222.20";
MessageBox.Show(mensajeError, "SisAcad - UESJ 2010", MessageBoxButtons.OK, MessageBoxIcon.Warning);
```

```
                    return 0;
            }//cierra catch

}//fin ConvierteDecimal

#endregion

#region Propiedad: Bandera Nombre Institucion
privatestaticstring banderaNombreInstitucion;
publicstaticstring BanderaNombreInstitucion
{
        get { return banderaNombreInstitucion;}
        set {banderaNombreInstitucion= value;}
}
#endregion


#region Formulario Solicitud Estudiante

#region Propiedad: Bandera Indice Solicitud
privatestaticint banderaIndiceSolicitud;
publicstaticint BanderaIndiceSolicitud
{
        get { return banderaIndiceSolicitud;}
        set {banderaIndiceSolicitud= value;}
}
#endregion

#region Propiedad: Bandera ModificaSolicitud
privatestaticbool  banderaModificaSolicitud;
publicstaticbool BanderaModificaSolicitud
{
        get { return banderaModificaSolicitud;}
        set {banderaModificaSolicitud= value;}
}
#endregion


#region Controles Formulario Principal

#region Propiedad:  Tipo Usuario
privatestaticint tipoUsuario;
publicstaticint TipoUsuario
{
        get { return tipoUsuario;}
        set {tipoUsuario= value;}
}
#endregion

#region Propiedad: Login Usuario
privatestaticstring  loginUsuario;
publicstaticstring LoginUsuario
{
        get { return loginUsuario;}
        set {loginUsuario= value;}
}
#endregion

#region Propiedad: UsuarioID
privatestaticint usuarioID;
//************************************************
```

```
publicstaticint UsuarioID
{
        get {return usuarioID;}
        set { usuarioID=value;}
}// Fin de UsuarioID
#endregion

#region Propiedad: PeriodoID
privatestaticint periodoID;
//*************************************************
publicstaticint PeriodoID
{
        get {return periodoID;}
        set { periodoID=value;}
}// Fin de periodoID
#endregion



#region Propiedad: Período Descripcion
privatestaticstring  periodoVigenteDescripcion;
publicstaticstring PeriodoVigenteDescripcion
{
        get { return periodoVigenteDescripcion;}
        set {periodoVigenteDescripcion= value;}
}
#endregion

#region Propiedad: Faculatad Nombre
privatestaticstring  facultadNombre;
publicstaticstring FacultadNombre
{
        get { return facultadNombre;}
        set {facultadNombre= value;}
}
#endregion

#region MessageBox: Metodos para Confirmación de Eliminar y Modificar

#region ConfirmarEliminacionItem: Mensaje de Confirmacion para eliminar un item
publicstaticDialogResult ConfirmarEliminacionItem(Form formulario,string nombreElemento)
{
        string mensaje = "Confirma que desea eliminar el elemento: "+nombreElemento+
                " ?\n"+"Le recordamos que todo cambio esta bajo su responsabilidad";
string titulo = "SisAcad - UESJ 2010";
        MessageBoxButtons buttons = MessageBoxButtons.YesNo;
        DialogResult resultado;

        // Displays the MessageBox.

        resultado = MessageBox.Show(formulario,mensaje, titulo, buttons,
                MessageBoxIcon.Question, MessageBoxDefaultButton.Button1);

        return resultado;

}// Fin de DialogResult ConfirmarEliminacionItem

#endregion

#region ConfirmarActualizarItem: Mensaje de Confirmacion para actualizar un item
```

```
publicstaticDialogResult ConfirmarActualizarItem(Form formulario,string nombreElemento)
{
	string mensaje = "Confirma que desea actualizar el elemento: "+nombreElemento+
		" ?\n"+"Le recordamos que todo cambio esta bajo su responsabilidad";
string titulo = "SisAcad - UESJ 2010";
	MessageBoxButtons buttons = MessageBoxButtons.YesNo;
	DialogResult resultado;

	// Displays the MessageBox.

	resultado = MessageBox.Show(mensaje, titulo, buttons,
		MessageBoxIcon.Question, MessageBoxDefaultButton.Button1);

	return resultado;

}// Fin de DialogResult ConfirmarActualizarItem

#endregion

#region Error: Mensaje de error
publicstaticDialogResult MensajeError(Form formulario,string mensajeParaError)
{
	string mensaje = mensajeParaError+"\n\nComúniquese con Soporte Técnico.";
string titulo = "SisAcad - UESJ 2010";
	MessageBoxButtons buttons = MessageBoxButtons.OK;
	DialogResult resultado;

	// Displays the MessageBox.

	resultado = MessageBox.Show(mensaje, titulo, buttons,
		MessageBoxIcon.Error, MessageBoxDefaultButton.Button1);

	return resultado;

}// Fin de DialogResult ConfirmarActualizarItem

#endregion

#region Aviso: Mensaje de aviso
publicstaticDialogResult MensajeAviso(Form formulario,string mensajeParaAviso)
{
	string mensaje = mensajeParaAviso;
	string titulo = "SisAcad - UESJ 2010";
	MessageBoxButtons buttons = MessageBoxButtons.OK;
	DialogResult resultado;

	// Displays the MessageBox.

	resultado = MessageBox.Show(mensaje, titulo, buttons,
		MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);

	return resultado;

}// Fin de DialogResult ConfirmarActualizarItem

#endregion

#region Aviso: Mensaje de Informacion
publicstaticDialogResult MensajeAvisoInformacion(Form formulario,string mensajeParaAviso)
{
	string mensaje = mensajeParaAviso;
```

```
            string titulo = "SisAcad - UESJ 2010";
            MessageBoxButtons buttons = MessageBoxButtons.OK;
            DialogResult resultado;

            // Displays the MessageBox.

            resultado = MessageBox.Show(mensaje, titulo, buttons,
                        MessageBoxIcon.Information, MessageBoxDefaultButton.Button1);

            return resultado;

}// Fin de DialogResult ConfirmarActualizarItem

#endregion

#region Aviso: Mensaje de aviso  Si No
publicstaticDialogResult MensajeAvisoSiNo(Form formulario,string mensajeParaAviso)
{
            string mensaje = mensajeParaAviso;
            string titulo = "SisAcad - UESJ 2010";
            MessageBoxButtons buttons = MessageBoxButtons.YesNo;
            DialogResult resultado;

            // Displays the MessageBox.

            resultado = MessageBox.Show(mensaje, titulo, buttons,
                        MessageBoxIcon.Question, MessageBoxDefaultButton.Button1);

            return resultado;

}// Fin de DialogResult ConfirmarActualizarItem

#endregion

#endregion


#region método ControlCamposNumeroentero positivo

publicstaticbool ControlNumeroEnteroPositivo(string contenidoCampo)
{
            //                              string mensajeError;
            int conversion;

            try
            {
                        conversion = System.Int32.Parse(contenidoCampo);


                        if(conversion>0)
                        {
                                    returntrue;
                        }
                        else
                        {

                                    returnfalse;
                        }
            }//cierra try

            catch(System.FormatException error)
```

```
                {
                        string tem = error.Message;
                        returnfalse;
                }//cierra catch

        }//fin ControlCamposRequeridos

        #endregion

        #region método ControlDecimal positivo

        publicstaticbool ControlDecimalPositivo(string cadenaDecimal)
        {
                //string mensajeError;
                try
                {

                        //Declara una variable de tipo formato de número
                        System.Globalization.NumberFormatInfo formatoNumero=
newNumberFormatInfo();

                        //Personaliza el formato del número

                        formatoNumero.NumberDecimalDigits=2;
                        formatoNumero.NumberDecimalSeparator=",";
                        formatoNumero.NumberGroupSeparator=".";
                        int[] grupo =newint[1] {3};

                        formatoNumero.NumberGroupSizes=grupo;

                        //Convierte a la cadena en un tipo decimal
                        Double
numeroDecimal=Double.Parse(cadenaDecimal,System.Globalization.NumberStyles.Float,formatoNumero);

                        if(numeroDecimal>0)
                        {
                                returntrue;
                        }
                        else
                        {
                                returnfalse;
                        }

                }//cierra try

                catch(System.Exception captura)
                {
                        string tem= captura.Message;
                        returnfalse;
                }//cierra catch

        }//fin ControlDecimal

        #endregion

        #region Método Control escritura Cédula
        publicstaticbool ControlEscrituraCedula(string cedula)
        {

                //MessageBox.Show(cedula.Length.ToString());
                bool bandera=true;
```

```
                    if(cedula.Length==10)
                    {
                        for(int i=0;i<cedula.Length;i++)
                        {
                            if(!Controles.ControlNumeroEnteroSinMensage(cedula.Substring(i,1)))
                            {
                                bandera=false;
                            }
                        }
                    }
                    else
                    {
                        bandera=false;
                    }

                    if(bandera==true)
                    {
                        returntrue;
                    }
                    else
                    {

MessageBox.Show("El formato de la cédula ingresada es incorrecto", "SisAcad - UESJ 2010", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);
                        returnfalse;
                    }
                }

                #endregion


                //******ACTUALIZADO CONTROL POR MA 02SEP2009
                publicstaticvoid FechaACurrencyString(object sender, ConvertEventArgs cevent)
                {
                    // The method converts only to string type. Test this using the DesiredType.
                    if(cevent.DesiredType != typeof(string)) return;

                    Thread.CurrentThread.CurrentCulture = newCultureInfo( "es-EC", false );

                    System.IFormatProvider formatoRegional = new System.Globalization.CultureInfo("es-EC",
true);
                    string[] formatosEsperados = {"G","g","f","F","D","d"};

                    DateTime fechaEcuatoriana = DateTime.ParseExact(cevent.Value.ToString(),
formatosEsperados, formatoRegional, DateTimeStyles.AllowWhiteSpaces);

                    // Use the ToString method to format the value as currency ("D").
                    cevent.Value = fechaEcuatoriana.ToString("D");
                }

                publicstaticvoid CurrencyStringAFecha(object sender, ConvertEventArgs cevent)
                {
                    //
                    if(cevent.DesiredType != typeof(DateTime)) return;

                    System.IFormatProvider formatoRegionalActual = new
System.Globalization.CultureInfo(CultureInfo.CurrentCulture.Name, true);

                    string[] formatosEsperados = {"G","g","f","F","D","d"};
                    DateTime fechaRegionalActual = DateTime.ParseExact(cevent.Value.ToString(),
formatosEsperados, formatoRegionalActual, DateTimeStyles.AllowWhiteSpaces);
```

```
                    // Use the ToString method to format the value as currency ("D").
                    cevent.Value = fechaRegionalActual;
                }


                #region Obtener Fecha del Servidor
                privatestaticDateTime fechaSistema;

                publicstaticDateTime FechaSistema
                {
                    get
                    {

        fechaSistema=System.Convert.ToDateTime(SqlHelper.ExecuteScalar(cadenaConexion,"spDevuelveFechaSist
ema"));
                        return fechaSistema;
                    }

                }


                #endregion


                publicstaticvoid ObtenerPerfilUsuario(int usuarioID)
                {
                    dsPerfilUsuario = newDatasetPerfilesUsuario();

                    SqlHelper.FillDataset(Controles.CadenaConexion,"F3_spConsultarPerfilUsuario_PO",
                        dsPerfilUsuario, newstring[]{"VISTA_USUARIO_PERFILES"},usuarioID);

                }
```