ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

ANÁLISIS COMPARATIVO DE LOS FRAMEWORKS EJB3 Y ADO.NET ENTITY FRAMEWORK PARA EL DESARROLLO DE APLICACIONES EMPRESARIALES.

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

EDISON ALFREDO SUÁREZ SILVA
s_edison@hotmail.com

DIRECTOR: ING. ROSA DEL CARMEN NAVARRETE RUEDA
rosan18@gmail.com

Quito, Octubre 2011

DECLARACIÓN

Yo, Edison Alfredo Suárez Silva declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido en la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad constitucional vigente.

_____

Edison Alfredo Suárez Silva

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Edison Alfredo Suárez Silva bajo mi supervisión.

_____

Ing. Rosa del Carmen Navarrete Rueda

DIRECTOR DEL PROYECTO

AGRADECIMIENTOS

Agradezco a Dios por la fortaleza y la sabiduría que me regala cada día.

A mis padres y hermanas por estar junto a mí incondicionalmente. Un especial agradecimiento a mi madre por su gran amor y apoyo en todo.

A toda mi familia que siempre ha confiado y creído en mí.

A mi directora de tesis Ing. Rosa Navarrete por la guía brindada durante el desarrollo del proyecto.

A mi gran amigo Pablo por todo su empuje y ayuda en este trabajo.

A todos aquellos que directa o indirectamente fueron parte de este proceso brindándome su conocimiento y mejores deseos.

Y un especial agradecimiento al amor de mi vida, Grace, por darme el cariño, comprensión y las fuerzas para culminar esta etapa de mi vida, y por todo el apoyo que he tenido desde el momento que llego a mi vida.

EDISON

DEDICATORIA

A mis padres que con todo su esfuerzo y sacrificio me dieron una profesión y en las buenas y en las malas me han sabido dar su cariño, compresión y apoyo.

A mis hermanas que han sido mis ejemplos de dedicación y coraje en cada cosa que les presenta la vida.

A mi sobrina Amelí por colmar mi vida de alegrías desde el día que nació.

A Grace por su cariño incondicional y por brindarme tantos momentos especiales y únicos.

Y a todas aquellas personas que forman parte de mi vida, que han dado su tiempo y colaboración en todo momento.

CONTENIDO

1. ANÁLISIS DE LOS FRAMEWORKS 3
1.1 Definición de los criterios de Evaluación de los Frameworks 3
1.1.1 Descripción de un Framework de persistencia 3
1.1.2 Definición de metodología de valoración de criterios 8
1.1.3 Definición de criterios 10
1.1.4 Elaboración de la matriz de comparación 21
1.2 Análisis de características de Ado .Net Entity Framework 23
1.2.1 Descripción de Ado .Net Entity Framework 23
1.2.2 Valoración de criterios de evaluación para Ado .Net Entity Framework 32
1.3 Análisis de características de EJB3 (Enterprise Java Beans 3) 36
1.3.1 Descripción de Enterprise Java Beans 3 36
1.3.2 Valoración de criterios de evaluación para EJB3/JPA 45
2. DESARROLLO DEL PROTOTIPO BASE 51
2.1 Metodología de diseño 51
2.2 Definición del alcance del prototipo base 57
2.2.1 Identificación del producto 57
2.2.2 Responsabilidades y Exclusiones del Sistema 58
2.3 Prototipo con base a Frameworks 59
2.3.1 Elaboración de Diagramas de la Metodología 59
3 ANÁLISIS COMPARATIVO 88
3.1 Evaluación de prototipos 88
3.1.1 ADO .Net Entity Framework 88
3.1.2 EJB 3/ JPA 92

3.2 Comparación de Frameworks 96

3.3 Análisis de Resultados 98

4 CONCLUSIONES Y RECOMENDACIONES..... 99

4.1 CONCLUSIONES 99

4.2 RECOMENDACIONES 101

GLOSARIO..... 103

BIBLIOGRAFÍA..... 106

ANEXOS..... 107

LISTA DE FIGURAS

FIGURA 1-1 FRAMEWORK DE PERSISTENCIA..... 6

FIGURA 1-2 ESCALA DE SAATY..... 8

FIGURA 1-3 ESTRUCTURA GENERAL DE UNA MATRIZ DE PRIORIZACIÓN. 9

FIGURA 1-4 REPRESENTACIÓN GRÁFICA DE ENTITY TYPES. 24

FIGURA 1-5 ARQUITECTURA ADO .NET ENTITY FRAMEWORK 25

FIGURA 1-6 ESTRUCTURA DE EDM. 26

FIGURA 1-7 ARQUITECTURA MULTICAPA – ENTITY FRAMEWORK..... 31

FIGURA 1-8 ARQUITECTURA JEE (EJB 3 Y JPA)..... 37

FIGURA 1-9 ARQUITECTURA JPA. 39

FIGURA 1-10 UBICACIÓN DE EJB Y JPA EN UNA ARQUITECTURA JEE. 44

FIGURA 2-1 ENFOQUE AERODINÁMICO DE DESARROLLO – ICONIX 52

FIGURA 2-2 FASES DE DESARROLLO – ICONIX..... 54

FIGURA 2-3 INTERFAZ – PÁGINA PRINCIPAL..... 59

FIGURA 2-4 INTERFAZ – GESTIONAR SOLICITUD (INGRESAR) 60

FIGURA 2-5 INTERFAZ – GESTIONAR SOLICITUD (MODIFICAR O ELIMINAR). ELABORADOR POR: EL AUTOR. 60

FIGURA 2-6 INTERFAZ – BUSCAR SOLICITUD. 61

FIGURA 2-7 MODELO DEL DOMINIO. 62

FIGURA 2-8 DESCRIPCIÓN DE ACTORES. ELABORADO POR: EL AUTOR..... 62

FIGURA 2-9 MODELO DE CASO DE USO. 63

FIGURA 2-10 DIAGRAMA DE ROBUSTEZ – GESTIONAR SOLICITUD DE TARJETA DE CRÉDITO..... 67

FIGURA 2-11 MODELO DE CLASES. 69

FIGURA 2-12 DIAGRAMA DE SECUENCIA – GESTIONAR SOLICITUD DE TARJETA DE CRÉDITO. 70

FIGURA 2-13 PANTALLA PRINCIPAL – SISTEMA DE GESTIÓN DE SOLICITUD DE TARJETA DE CRÉDITO..... 71
FIGURA 2-14 PANTALLA PRINCIPAL – SISTEMA DE GESTIÓN DE SOLICITUD DE TARJETA DE CRÉDITO..... 71
FIGURA 2-15 PANTALLA DE INGRESO DE SOLICITUD..... 72
FIGURA 2-16 PANTALLA DE SOLICITUD INGRESADA CON ÉXITO. ELABORADO POR: EL AUTOR. 72
FIGURA 2-17 PANTALLA DE EDICIÓN Y ELIMINACIÓN DE SOLICITUDES. 73
FIGURA 2-18 PANTALLA DE ACTUALIZACIÓN DE DATOS DE SOLICITUD..... 73
FIGURA 2-19 PANTALLA DE SOLICITUD ACTUALIZADA CON ÉXITO. 74
FIGURA 2-20 PANTALLA DE EDICIÓN Y ELIMINACIÓN DE SOLICITUDES. 74
FIGURA 2-21 PANTALLA – DESEA ELIMINAR EL REGISTRO SELECCIONADO. 75
FIGURA 2-22 PANTALLA SOLICITUD ELIMINADA CON ÉXITO. 75
FIGURA 2-23 PANTALLA PRINCIPAL – SISTEMA DE GESTIÓN DE SOLICITUD DE TARJETA DE CRÉDITO..... 76
FIGURA 2-24 PANTALLA DE BÚSQUEDA SOLICITUD. 76
FIGURA 2-25 PANTALLA DE VISUALIZACIÓN DE LOS DATOS DE SOLICITUD. 77
FIGURA 2-26 PANTALLA PRINCIPAL – SISTEMA DE GESTIÓN DE SOLICITUD DE TARJETA DE CRÉDITO..... 77
FIGURA 2-27 PANTALLA DE OPCIONES PARA GESTIONAR SOLICITUD. 78
FIGURA 2-28 PANTALLA DE INGRESO DE SOLICITUD..... 78
FIGURA 2-29 PANTALLA DE SOLICITUD INGRESADA CON ÉXITO. 79
FIGURA 2-30 PANTALLA DE EDICIÓN Y ELIMINACIÓN DE SOLICITUDES. 79
FIGURA 2-31 PANTALLA DE ACTUALIZACIÓN DE DATOS DE SOLICITUD..... 80
FIGURA 2-32 PANTALLA DE SOLICITUD ACTUALIZADA CON ÉXITO. 80
FIGURA 2-33 PANTALLA DE EDICIÓN Y ELIMINACIÓN DE SOLICITUDES. 81
FIGURA 2-34 PANTALLA – DESEA ELIMINAR EL REGISTRO SELECCIONADO. 81
FIGURA 2-35 PANTALLA SOLICITUD ELIMINADA CON ÉXITO. 82
FIGURA 2-36 PANTALLA PRINCIPAL – SISTEMA DE GESTIÓN DE SOLICITUD DE TARJETA DE CRÉDITO..... 82
FIGURA 2-37 PANTALLA DE BÚSQUEDA SOLICITUD. 83
FIGURA 2-38 PANTALLA DE VISUALIZACIÓN DE LOS DATOS DE SOLICITUD. 83
FIGURA 2-39 ARQUITECTURA DE DESARROLLO UTILIZANDO EJB3 85
FIGURA 2-40 ARQUITECTURA DE DESARROLLO UTILIZANDO ENTITY FRAMEWORK 85
FIGURA 2-41 DIAGRAMA DE RED. ELABORADO POR: EL AUTOR. 86

LISTA DE TABLAS

TABLA 1 RESUMEN DE CRITERIOS DE EVALUACIÓN. 11
TABLA 2 VALORACIÓN DE MADUREZ DEL FRAMEWOK. 12
TABLA 3 VALORACIÓN DE DOCUMENTACIÓN Y SOPORTE DEL FRAMEWORK. 13
TABLA 4 VALORACIÓN DE TIPO DE LICENCIAMIENTO. 13
TABLA 5 VALORACIÓN DE DEFINICIÓN DE METADATOS. 14
TABLA 6 VALORACIÓN DE GENERACIÓN DE CONSULTAS. 15
TABLA 7 VALORACIÓN DE CONTROL DE TRANSACCIONES Y CONCURRENCIA. 15
TABLA 8 VALORACIÓN DE REQUERIMIENTOS DE HARDWARE Y SOFTWARE. 18
TABLA 9 VALORACIÓN DE COSTO DE DESARROLLO. 18
TABLA 10 VALORACIÓN DE CURVA DE APRENDIZAJE. 19
TABLA 11 VALORACIÓN DE HERRAMIENTAS DE DESARROLLO. 20
TABLA 12 VALORACIÓN DE INTERACCIÓN CON OTRAS TECNOLOGÍAS. 21
TABLA 13 ESCALA DE VALORES DE PREFERENCIA. 21
TABLA 14 RESUMEN DE LA MATRIZ DE COMPARACIÓN. 23
TABLA 15 MATRIZ DE PRIORIZACIÓN PRELIMINAR. 50
TABLA 16 MATRIZ DE PRIORIZACIÓN PRELIMINAR NORMALIZADA. 50
TABLA 17 COMPARATIVA DE METODOLOGÍAS ÁGILES DE DESARROLLO DE SOFTWARE. 51
TABLA 18 CUADRO DE ACTIVIDADES PARA LAS FASES DE ICONIX. 57
TABLA 19 CARACTERÍSTICAS DE LOS RECURSOS UTILIZADOS EN EL DESARROLLO DE PROTOTIPOS. 84
TABLA 20 CARACTERÍSTICAS DE LOS RECURSOS UTILIZADOS EN EL DESARROLLO DE PROTOTIPOS. 87
TABLA 21 CARACTERÍSTICAS DE LOS RECURSOS UTILIZADOS EN EL DESARROLLO DE PROTOTIPOS. 87
TABLA 22 MATRIZ DE PRIORIZACIÓN FINAL SIN NORMALIZAR. 97
TABLA 23 MATRIZ DE PRIORIZACIÓN FINAL NORMALIZADA. 97

RESUMEN

En el presente Proyecto de Titulación se establece un conjunto de criterios que permiten evaluar frameworks de desarrollo de la capa de persistencia de aplicaciones empresariales en ambientes Java y .Net, en base a estos criterios se valoró los frameworks Entity Framework y EJB 3 y se identificó fortalezas y debilidades de cada uno de ellos tanto a nivel teórico como práctico a través del desarrollo de prototipos. Los resultados obtenidos permitirán seleccionar el framework más idóneo considerando las necesidades específicas que una organización tenga para desarrollar aplicaciones empresariales.

En el Capítulo Uno se especifican los criterios referenciales y su forma de evaluación así como también se definen como mecanismos para determinar el framework más apto el uso de la matriz de priorización y la matriz de comparación. Además se realiza una descripción de las generalidades de un framework de persistencia y de cada uno de los frameworks objetos de estudio de este trabajo.

En el Capitulo Dos se define la metodología de desarrollo con el que se elaborarán los prototipos, se define además el caso de aplicación que permitirá un análisis de rendimiento de los componentes de cada framework y posteriormente se realiza la implementación del caso de aplicación con cada uno de los frameworks.

El Capítulo Tres permite realizar la evaluación, basada en los criterios definidos en el capítulo uno y la experiencia obtenida durante el proceso de implementación del capítulo dos, para cada criterio se obtienen valoraciones finales que permiten determinar la conveniencia de los frameworks para el caso de aplicación escogido.

Finalmente, en el Capítulo Cuatro se presentan las conclusiones y recomendaciones, resultado del trabajo desarrollado.

PRESENTACIÓN

La orientación a objetos se ha convertido en el paradigma dominante para el desarrollo de aplicaciones empresariales. En este contexto, el mecanismo utilizado para resolver la persistencia de datos es de vital importancia, no solo por su impacto en el desempeño final del sistema sino también por los efectos de asegurar aplicaciones de calidad. Al existir numerosas alternativas a la hora de seleccionar un mecanismo para acceder a los datos y posteriormente manipularlos, resulta imprescindible contar con una visión general, objetiva y bien estructurada que permita discriminar la aplicabilidad de cada uno de los frameworks objetos de estudio de este trabajo.

1. ANÁLISIS DE LOS FRAMEWORKS

1.1 Definición de los criterios de Evaluación de los Frameworks

1.1.1 Descripción de un Framework de persistencia

Actualmente, todas las empresas almacenan la información que manejan (clientes, empleados, productos, etc.) en sistemas persistentes, los cuales permiten preservar los datos para su posterior uso. El sistema más común y extendido es el uso de una base de datos relacional.

Alrededor de estos sistemas y para facilitar la extracción de la información se construyen aplicaciones que se alimentan de los datos almacenados en la base de datos; los procesan, realizan cálculos necesarios, se los presenta al usuario permitiendo así mismo realizar operaciones, modificarlos e inclusive eliminarlos de una forma sencilla y flexible.

El presente proyecto centrará su análisis en las aplicaciones desarrolladas con los lenguajes de programación Java y C# que siguen un modelo de programación orientado a objetos.

Cuando se crean sistemas orientados a objetos que usan bases de datos relacionales, los desarrolladores deben crear objetos persistentes, es decir, convertir los objetos del lenguaje de programación a registros de la base de datos. De la misma manera, hay que implementar la operación inversa, es decir, convertir los registros en objetos. Esto debido a la diferencia lógica entre los dos paradigmas, los objetos tienen atributos y operaciones así como herencia, mientras que las bases de datos relacionales contienen tablas, atributos y relaciones entre tablas.

Para persistir un objeto en una base de datos relacional, el uso más común es que el desarrollador utilice llamadas manuales a comandos de consulta y persista el objeto o construya el objeto a partir de los resultados. Esta opción hace que el

desarrollo sea costoso. Además, el desarrollador debe tratar con aspectos complejos tales como abrir conexiones con la base de datos, cerrar conexiones, comprobación de bloqueos de escritura y lectura, conversión de tipos entre los campos de la tabla y los atributos del objeto, etc. Tareas repetitivas que pueden dar lugar a error.

Con el fin de solventar estas tareas recurrentes y evitar fallos, empresas como Sun y Microsoft desarrollaron soluciones que pasan por la utilización de un framework[1] que se encargue de automatizar el proceso de conversión entre el paradigma relacional de las bases de datos a un modelo de objetos de manera transparente al desarrollador, de tal forma que éste no tenga qué utilizar llamadas manuales a comandos de consulta, y que la gestión del acceso a la base de datos esté centralizado en un componente único, permitiendo su reutilización. Para ello, el uso de un framework de persistencia es la opción más óptima y que mayores ventajas aporta.

Los frameworks de persistencia nacieron a mediados de la década del 90, y se hicieron populares a partir del crecimiento de Java. Por esa razón, los frameworks más populares hoy en día en .Net son adaptaciones del modelo pensado para Java.

Los frameworks de persistencia actúan como capas intermedias (middleware) entre la capa de negocios y la capa de datos facilitando el almacenamiento de la información proveniente de la aplicación en bases de datos, especialmente relacionales.

Para facilitar el trabajo de los objetos con una base de datos relacional utilizan una técnica de programación denominada Mapeo Objeto Relacional (Object Relational Mapping con sus siglas en inglés ORM).

[1] Framework → Glosario de términos 1.1

El ORM permite enlazar un lenguaje de programación orientado a objetos con una base de datos relacional (mapeo de atributos de un objeto, con filas y columnas de una tabla de una base de datos relacional), permitiendo al desarrollador mantener una perspectiva orientada a objetos y cuidar que se consiga aportar una solución a la lógica de negocio, eliminando el obstáculo que existía al tratar de conseguir la compatibilidad entre objetos con modelos relacionales (Object Relational Impedance Mismatch[2]).

Los ORM cumplen con las siguientes condiciones (Bauer & King, 2007):

- Permite operaciones CRUD[3] para la persistencia de objetos.
- Trabaja con un lenguaje para especificar las consultas (lenguaje de consulta) que permita trabajar con clases o atributos de las clases.
- Permite diferentes formas de especificación de mapeo de clases (metadatos) con las correspondientes tablas.
- Permite conocer el estado de los objetos para determinar si han sido modificados.
- Admite el uso de cargas retardadas de objetos (lazy loading) como una forma de optimizar las consultas.

Un ORM entonces debe encapsular la interacción entre la aplicación y la base de datos relacional, dejando a los desarrolladores la libertad de concentrarse en temas de negocios.

Los frameworks de persistencia no son únicamente herramientas utilizadas para el mapeo de objetos de negocio con bases relacionales, sino que además de contener un ORM ofrecen servicios que transparentan hacia el desarrollador la gestión de un objeto a una base de datos y viceversa. En la figura 1-1 se puede observar la estructura de un framework de persistencia.

[2] Object Relational Impedance Mismatch → Glosario de términos 1.2

[3] CRUD → Glosario de términos 1.3

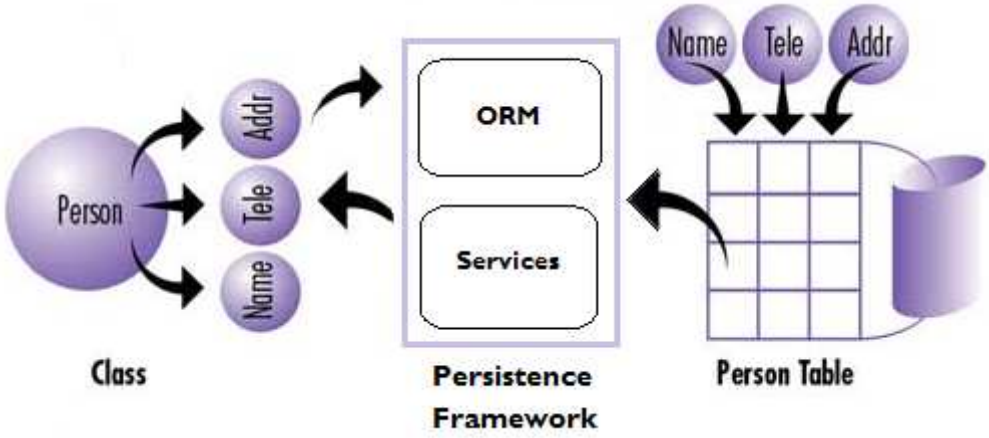

Figura 1-1 Framework de Persistencia

Fuente: Gráfico mejorado por el Autor de: http://www2.sys-con.com/itsg/virtualcd/Java/archives/0511/Jasnowski/index.html

Entre los servicios que un framework de persistencia ofrece se puede nombrar los siguientes (IBM Corporation, 2004):

- Persistencia.- Permite que un usuario almacene objetos en una sesión y pueda acceder a ellos en una sesión posterior.
- Consultas.- Permite a las aplicaciones consultar y recuperar objetos basados en una variedad de criterios.
- Transaccionalidad.- El soporte transaccional permite al desarrollador de la aplicación definir una unidad atómica de trabajo. En terminología de base de datos, significa que el sistema debe ser capaz de aplicar un conjunto de cambios en la base de datos, o que debe asegurarse de que ninguno de los cambios se apliquen. Un framework de persistencia al menos debería proveer operaciones transaccionales como commit y rollback.
- Concurrencia.- Los sistemas multiusuario orientados a objetos deben controlar el acceso concurrente a los objetos. Cuando un objeto es accedido simultáneamente por muchos usuarios, el sistema debe proporcionar un mecanismo para asegurar que las modificaciones del objeto en el almacenamiento persistente se producen de una manera predecible y controlada. Los frameworks de persistencia pueden aplicar controles de tipo pesimista u optimista.

- Relaciones.- Permite manejar las relaciones que se utilice entre los objetos. En los sistemas relacionales, las relaciones entre las entidades se implementan utilizando claves externas ó claves primarias. En los sistemas orientados a objetos, las relaciones suelen ser explícitamente implementadas a través de los atributos.

Este aislamiento de la capa de base de datos con la capa de negocio, significa que la aplicación en general será más útil y las capas individuales podrán ser reemplazadas fácilmente. Consecuentemente, el aspecto de administrar el almacenamiento, consultas y retornos de datos, transaccionalidad y el acceso concurrente a los datos es completamente removido del código del negocio y se lo está haciendo sin intervención del desarrollador.

Los frameworks de persistencia ofrecen los siguientes beneficios:

- Modelo de programación natural, el programador accede a los datos en forma de objetos en el mismo lenguaje con el que se encuentra trabajado, abstrayendo totalmente el lugar donde se encuentren almacenados o la manera de accederlos.
- Minimiza la cantidad de líneas de código, evita la repetición o duplicación de código para acceso de datos.
- Oportunidades para el manejo de caches, se puede implementar lógica de manejo de caches en la capa de datos.
- Mapeo estructural más robusto cuando el modelo objeto-tabla cambia, existe la facilidad de que los cambios se aíslen en un solo punto.
- Permite delegar a la BD las actividades que realiza mejor y obtener todos los beneficios del trabajo con objetos para el desarrollador.

Finalmente desde el punto de vista de un desarrollador de aplicaciones, la principal ventaja de los frameworks de persistencia es la posibilidad que tienen para centrar su atención en la lógica de negocio en lugar del manejo de la funcionalidad de persistencia, que a menudo es una tarea tediosa y propensa a errores.

1.1.2 Definición de metodología de valoración de criterios

Para dar una ponderación y estimar que criterios tendrán mayor relevancia en la evaluación, se utilizará una parte del proceso analítico jerárquico o proceso analítico de jerarquías; y la matriz de priorización, como herramienta para determinar el framework más adecuado para el caso de aplicación.

1.1.2.1 Proceso Analítico Jerárquico

El Proceso Analítico Jerárquico (Analytic Hierarchy Process con sus siglas en inglés AHP), desarrollado por Thomas Saaty, es una técnica estructurada utilizada para la toma de decisiones, que permite, determinar de forma cuantitativa la importancia de cada uno de los criterios que participan en una comparación (Universidad Politécnica de Valencia, 2010).

En este proceso los criterios son evaluados por pares mediante una matriz de comparación y una escala de valores numéricos establecida en base a expresiones verbales que se puede observar en la figura 1-2.

ESCALA NUMERICA	ESCALA VERBAL	EXPLICACION
1.0	Ambos elementos son de igual importancia.	Ambos elementos contribuyen con la propiedad en igual forma.
3.0	Moderada importancia de un elemento sobre otro.	La experiencia y el juicio favorece a un elemento por sobre el otro.
5.0	Fuerte importancia de un elemento sobre otro.	Un elemento es fuertemente favorecido.
7.0	Muy fuerte importancia de un elemento sobre otro.	Un elemento es muy fuertemente dominante.
9.0	Extrema importancia de un elemento sobre otro.	Un elemento es favorecido, por lo menos con un orden de magnitud de diferencia.
2.0,4.0,6.0,8.0	Valores intermedios entre dos juicios adyacentes.	Usados como valores de consenso entre dos juicios.
Incrementos de 0.1	Valores intermedios en la graduación más fina de 0.1 (Por ejemplo 5.2 es una entrada válida).	Usados para graduaciones más finas de los juicios.

Figura 1-2 Escala de Saaty.

Fuente: Ávila R. (2000). El AHP (Proceso Analítico Jerárquico). P19.

En la matriz de comparación se colocan los criterios en la primera fila y en la primera columna de la matriz, en cada una de las celdas se coloca el valor de la escala obtenido de la comparación entre un elemento de la fila con el correspondiente elemento de la columna, para elementos ya comparados se coloca el valor del inverso multiplicativo del valor previamente asignado.

Posteriormente se realiza la suma de los valores por columna, y se realiza una nueva matriz normalizada obtenida de dividir el valor de cada celda con el correspondiente valor de la suma de su columna.

Finalmente se obtiene el promedio de los valores de la fila y este es el factor que determinará la ponderación de los criterios.

En el presente trabajo únicamente se utilizará la matriz de comparación y la escala para establecer pesos a los diferentes criterios.

1.1.2.2 Matriz de Priorización

Una matriz de priorización permite clasificar opciones, problemas o asuntos basados en un conjunto de criterios que permiten fijar la prioridad de los mismos, estos criterios son particulares dependiendo de la situación considerada.
La forma general de una matriz de priorización es la que se describe en la figura 1-3

Opciones	Criterios				Total
	C#1	C#2	C#3	C#4	
Opción 1					
Opción 2					
Opción 3					

Figura 1-3 Estructura general de una matriz de priorización.

Fuente: http://www.ongconcalidad.org/priorizacion.pdf

La primera columna se compone de las opciones a ser priorizadas, en el caso particular del presente trabajo, esta columna estará compuesta por los frameworks a ser evaluados.
El resto de columnas se componen por los criterios descritos en la sección 1.1.3
La columna final corresponde a los totales de cada opción, luego de ordenar las opciones de mayor a menor se obtendrá una matriz que refleje el framework más adecuado de acuerdo a los criterios.

1.1.3 Definición de criterios

De los criterios de evaluación definidos en los trabajos realizados por (Pothu, 2008) y (Bauer & King, 2007) para la evaluación de frameworks de persistencia, se establecen a continuación los criterios que se utilizarán para valorar los frameworks objetos de estudio de este trabajo.

Para los criterios cuya valoración se puede obtener basada en la descripción de las secciones 1.2.1 y 1.3.1, se dará una valoración final mientras que aquellos criterios que requieren de comprobación mediante la ejecución de pruebas sobre los prototipos obtendrán una valoración preliminar.

Los criterios considerados para la evaluación de los frameworks se resumen en la Tabla 1, el detalle de cada criterio se realiza de manera inmediata a la tabla de resumen.

Categoría	Criterio	Valoración
I. General	Madurez del framework.	1-4
	Soporte y documentación del framework.	1-4
	Tipo de licenciamiento.	1-2

II. Funcionalidad	Definición de metadatos.	1-4
	Facilidades de generación de consultas (Queries).	1-4
	Control de transacciones y concurrencia.	1-3
III. Rendimiento	Tiempo de ejecución de operaciones CRUD - Create.	
	Tiempo de ejecución de operaciones CRUD - Retrieve.	
	Tiempo de ejecución de operaciones CRUD - Update.	
	Tiempo de ejecución de operaciones CRUD - Delete.	
	Tiempo en la Generación del Modelo de Entidades.	
IV. Recursos	Requerimientos de Hardware y Software para el desarrollo.	1-4
	Costo de Desarrollo.	1-4
V. Facilidades para el desarrollo	Curva de aprendizaje.	1-4
	Herramientas de desarrollo.	1-4
	Facilidad de interacción con otras tecnologías.	1-4

Tabla 1 Resumen de criterios de evaluación.

Elaborado por: El Autor.

I. General

a.- Madurez del framework (G1)

Este criterio se refiere al crecimiento del framework a través del tiempo, desde su creación hasta la actualidad, conlleva la adopción de nuevas tecnologías, procesos y modelos con el objetivo de fortalecer el producto, cubrir de mejor manera los requerimientos y adaptarse a nuevos entornos sin perder rendimiento y compatibilidad, logrando con esto mantener o mejorar su presencia en el mercado.

VALORACIÓN

Valor	Significado
1	El framework se ha mantenido estático en funcionalidades, metodología y emplea tecnologías que hacen prever su obsolescencia en un tiempo corto.
2	El framework soportó cambios desde su creación, pero mantiene limitaciones que le impiden competir con nuevas tecnologías.
3	El framework ha soportado cambios desde su creación y a pesar de cubrir limitaciones y adaptarse a nuevas tecnologías no prevalece de una forma notable en su entorno.
4	El framework ha adoptado las mejores prácticas desde su creación para cubrir el mayor número de demandas y limitaciones, mejorando su posición en el mercado y motivando a mejorar o crear especificaciones basadas en su desarrollo.

Tabla 2 Valoración de Madurez del Framewok.

Elaborado por: El Autor.

b.- Documentación y soporte del framework (G2)

Criterio que considera las responsabilidades que cada organización a cargo del desarrollo del framework tiene en la solución de problemas o bugs propios de sus productos; a la vez se refiere al acceso a documentación, capacitación y posibilidades de certificación.

VALORACIÓN

Valor	Significado
1	No existe soporte alguno para el framework.
2	El soporte lo brinda únicamente la comunidad de desarrolladores.
3	El soporte es exclusivo de la empresa desarrolladora, y no existe apoyo formal para la comunidad de desarrolladores.
4	Existe soporte formal para los errores propios del framework, además de una comunidad que respalda el uso, esta comunidad recibe apoyo de la empresa desarrolladora.

Tabla 3 Valoración de Documentación y soporte del framework.

Elaborado por: El Autor.

c.- Tipo de Licenciamiento (G3)

Considera las diferencias en licenciamiento de los productos de software desarrollados a partir del uso de los frameworks, las limitaciones y las perspectivas para comercializar dichos productos. Permite hacer una diferenciación entre la orientación de los frameworks con respecto a software libre y software privativo.

VALORACIÓN

Valor	Significado
1	Software libre GPL
2	Software libre sin obligación de generar software libre (LGPL)
1	Software privativo
2	Código propietario abierto

Tabla 4 Valoración de Tipo de licenciamiento.

Fuente: Proyecto de Titulación, CORREA, Pablo. Análisis comparativo de los Frameworks Adobe Flex, Java Rich Faces y Extjs para el desarrollo de Aplicaciones Enriquecidas en Internet (RIA). 2010

II. Funcionalidad

a.- Definición de metadatos (F1)

Determina el tipo de metadata que utiliza el framework por medio de la cual se puede definir configuraciones o comportamiento en el modelo de objetos para interactuar con sistemas de base de datos o con capas superiores.

VALORACIÓN

Valor	Significado
1	No utiliza metadata para la interacción del framework con el resto de capas.
2	Limitado uso de metadata, especifico para configuraciones y no para comportamientos del modelo de objetos.
3	Utiliza un solo tipo de mecanismo para expresar metadata tanto para configuraciones como para el modelo de objetos.
4	Utiliza diferentes mecanismos para expresar metadata tanto para configuraciones como para el modelo de objetos.

Tabla 5 Valoración de Definición de Metadatos.

Elaborado por: El Autor.

b.- Facilidades de generación de consultas (F2).

Considera el grado de interacción que tiene el framework con lenguajes nativos de consulta o lenguajes propios del framework para realizar operaciones y consultas en el modelo de objetos o en el sistema de almacenamiento de datos.

VALORACIÓN

Valor	Significado
1	Soporta únicamente el lenguaje nativo de consulta SQL[4] para el manejo de datos.
2	Soporta el lenguaje nativo de consulta SQL y lenguajes de consulta de terceros.

[4] SQL → Glosario de Términos 1.4

3	Soporta el lenguaje nativo de consulta SQL y lenguajes de consulta propios del framework con funcionalidades limitadas
4	Soporta lenguajes nativos de consulta y lenguajes de consulta robustos propios del framework transparentando de la mejor manera la interacción del modelo de objetos con el sistema de almacenamiento de datos.

Tabla 6 Valoración de Generación de consultas.

Elaborado por: El Autor.

c.- Control de transacciones y concurrencia (F3).

Toma en cuenta las variadas funcionalidades o servicios que el framework provee para el control de transacciones y concurrencia que se maneja al pasar información entre el modelo de objetos y el sistema de almacenamiento de datos.

VALORACIÓN

Valor	Significado
1	No provee servicios para el control de transacciones y concurrencia.
2	Provee servicios para el control de transacciones y concurrencia a través de otros frameworks.
3	Provee servicios para el control de transacciones y concurrencia a través de librerías propias del framework.

Tabla 7 Valoración de Control de transacciones y concurrencia.

Elaborado por: El Autor.

III. Rendimiento

a.- Tiempo de ejecución de operación CRUD - CREATE (R1).

Se refiere al tiempo que el framework emplea en la ejecución de la operación CRUD - CREATE desde el modelo de objetos a un sistema de base de datos.

VALORACIÓN

La valoración de este criterio se dará en función del menor tiempo de ejecución de la operación CRUD – CREATE obtenida en la evaluación de prototipos en el capítulo 3, asignándole el valor más alto al que tome menor tiempo en la ejecución de la operación CRUD - CREATE.

b.- Tiempo de ejecución de operación CRUD - RETRIEVE (R2).

Se refiere al tiempo que el framework emplea en la ejecución de la operación CRUD - RETRIEVE desde el modelo de objetos a un sistema de base de datos.

VALORACIÓN

La valoración de este criterio se dará en función del menor tiempo de ejecución de la operación CRUD - RETRIEVE obtenida en la evaluación de prototipos en el capítulo 3, asignándole el valor más alto al que tome menor tiempo en la ejecución de la operación CRUD - RETRIEVE.

c.- Tiempo de ejecución de operación CRUD - UPDATE (R3).

Se refiere al tiempo que el framework emplea en la ejecución de la operación CRUD - UPDATE desde el modelo de objetos a un sistema de base de datos

VALORACIÓN

La valoración de este criterio se dará en función del menor tiempo de ejecución de la operación CRUD - UPDATE obtenida en la evaluación de prototipos en el capítulo 3, asignándole el valor más alto al que tome menor tiempo en la ejecución de la operación CRUD - UPDATE.

d.- Tiempo de ejecución de operación CRUD - DELETE (R4).

Se refiere al tiempo que el framework emplea en la ejecución de la operación CRUD - DELETE desde el modelo de objetos a un sistema de base de datos

VALORACIÓN

La valoración de este criterio se dará en función del menor tiempo de ejecución de la operación CRUD - DELETE obtenida en la evaluación de prototipos en el capítulo 3, asignándole el valor más alto al que tome menor tiempo en la ejecución de la operación CRUD - DELETE.

e.- Tiempo en la Generación del Modelo de Entidades (R5).

Se refiere al tiempo que el framework emplea en generar el modelo de entidades partiendo de un esquema de base de datos.

VALORACIÓN

La valoración de este criterio se dará en función del menor tiempo que tome la generación del modelo de entidades obtenida en la evaluación de prototipos en el capítulo 3, el valor más alto se le asignará al que tome menor tiempo en la generación del modelo de entidades.

IV. Recursos

a.- Requerimientos de Hardware y Software para el desarrollo (RS1).

Criterio que evalúa las características requeridas para el desarrollo, esto debido a que en cuanto a requerimientos de cliente no existen mayores especificaciones. En el tema de software considera principalmente la integración con diferentes sistemas operativos como plataforma de soporte para la ejecución del framework, considera además independencia de software o hardware adicional requerido para el funcionamiento del framework.

VALORACIÓN

Valor	Significado
1	El framework requiere de hardware especializado para el desarrollo de aplicaciones y además es dependiente del sistema operativo.

2	El framework es limitado en cuanto al soporte de sistemas operativos, sin embargo no requiere de hardware adicional.
3	El framework soporta múltiples sistemas operativos, sin embargo requiere del soporte de software adicional para su correcto funcionamiento.
4	Soporta múltiples sistemas operativos y no se requiere hardware adicional al requerido por el sistema operativo sobre el cual se encuentra instalado. De la misma manera el framework no requiere de software adicional para su funcionamiento.

Tabla 8 Valoración de Requerimientos de Hardware y Software.

Fuente: Proyecto de Titulación, CORREA, Pablo. Análisis comparativo de los Frameworks Adobe Flex, Java Rich Faces y Extjs para el desarrollo de Aplicaciones Enriquecidas en Internet (RIA). 2010

b.- Costo de Desarrollo (RS2).

Criterio que se refiere primordialmente a toda la inversión necesaria para desarrollar una aplicación empresarial con cualquiera de los frameworks, toma en cuenta costo de licencias así como costo del soporte del producto. El costo a evaluar será el costo individual de una licencia de desarrollo y el costo de soporte promedio para un periodo de un año.

VALORACIÓN

Valor	Significado
1	Costo total mayor a 5000 dólares.
2	Costo total entre 3000 y 5000 dólares.
3	Costo total entre 1000 y 3000 dólares.
4	Costo total menor a 1000 dólares.

Tabla 9 Valoración de Costo de Desarrollo.

Elaborado por: El Autor.

V. Facilidades para el desarrollo

a.- Curva de aprendizaje (FD1).

Determina la facilidad e inversión de tiempo necesarios para obtener un nivel de conocimiento aceptable del uso de los componentes de cada uno de los frameworks.

VALORACIÓN

Valor	Significado
1	El framework utiliza componentes cuyo manejo requiere un profundo conocimiento y años de estudio.
2	El lenguaje que utiliza el framework es una implementación propietaria que requiere del aprendizaje de la misma, tomando un tiempo de aprendizaje de varios meses.
3	Si se conoce lenguajes como C#, Java y SQL, la utilización del framework a un nivel intermedio requiere de pocas semanas.
4	Sin experiencia previa en lenguajes de programación, únicamente conociendo los fundamentos, el desarrollador es capaz de utilizar y conocer los componentes de una manera adecuada en pocas semanas.

Tabla 10 Valoración de Curva de aprendizaje.

Fuente: El Autor.

b.- Herramientas de desarrollo (FD2).

Criterio que busca determinar la productividad en el desarrollo de aplicaciones empresariales, fijándose en la existencia de IDE's, herramientas para depuración de código y herramientas para realización de pruebas.

Es necesario anotar que buenas herramientas facilitan al desarrollador la ejecución de tareas recurrentes.

VALORACIÓN

Valor	Significado
1	Los componentes del framework se encuentran aislados y no existe una herramienta que mejore la productividad.
2	El framework se integra con IDE's de terceros mediante complementos permitiendo la edición de aplicaciones.
3	Existe un IDE perteneciente a la misma empresa que desarrolló el framework o de terceros que permite el desarrollo de aplicaciones, así como la ejecución de pruebas, al menos al nivel de editor de código.
4	Existe un IDE que permite el desarrollo de aplicaciones a nivel gráfico además de la edición de código. El IDE considera la integración para la ejecución de pruebas y despliegue de la aplicación.

Tabla 11 Valoración de Herramientas de Desarrollo.

Fuente: Proyecto de Titulación, CORREA, Pablo. Análisis comparativo de los Frameworks Adobe Flex, Java Rich Faces y Extjs para el desarrollo de Aplicaciones Enriquecidas en Internet (RIA). 2010

c.- Facilidad de interacción con otras tecnologías (FD3).

Criterio que se refiere a la capacidad que brindan los frameworks para interactuar con bases de datos y acoplamiento con aplicaciones desarrolladas con lenguajes distintos a los nativos de cada framework, con esto comprendemos la facilidad de acoplamiento del framework de persistencia con la capa de negocio, la cual puede estar desarrollada utilizando diferentes tecnologías.

VALORACIÓN

Valor	Significado
1	No existen API's de integración con tecnologías de terceros
2	Las API's de integración abarcan gran cantidad de tecnologías, mas requieren de un conocimiento profundo del framework.

3	Las API's de integración son limitadas y se suscriben a tecnologías especificas, sin embargo no requieren de un gran esfuerzo para conseguir esta integración.
4	Existe un gran número de API's que soporta un conjunto amplio de tecnologías estándar.

Tabla 12 Valoración de Interacción con otras tecnologías.

Elaborado por: El Autor.

1.1.4 Elaboración de la matriz de comparación

Como se mencionó en la sección 1.1.2.1 (Proceso Analítico Jerárquico) para la elaboración de la matriz de comparación es necesario seleccionar los criterios de evaluación y la escala de valores con los que se estimará cada uno de los criterios.

Los criterios fueron seleccionados y definidos en la sección 1.1.3 y a continuación se establece la escala de valores para elaborar la matriz de comparación del presente trabajo.

Escala de valores de preferencia:

Preferencia	Valor
Igual	1
Media	3
Fuerte	5
Muy Fuerte	7

Tabla 13 Escala de valores de preferencia.

Elaborado por: El Autor.

Con la escala y criterios establecidos se procede a dar valores a cada celda de la matriz, mediante una comparación de filas vs columnas, tomando en cuenta las siguientes consideraciones (Universidad Politécnica de Valencia, 2010):

- Si el criterio de la fila es considerado más importante que el criterio de la columna, se asigna a la celda el valor neto.
- Si el criterio de la fila es considerado menos importante que el criterio de la columna, se asigna a la celda el recíproco del valor.

Una vez determinados los valores en cada criterio se realiza una sumatoria por columnas de la matriz para obtener los pesos relativos de cada criterio que posteriormente permitirán elaborar la matriz de comparación normalizada. En el Anexo1 se puede observar la matriz de comparación sin normalizar del presente trabajo con los criterios y valores asignados.

Luego se elabora la matriz de comparación normalizada que se obtiene al dividir el valor de cada celda con el peso relativo de su correspondiente columna obtenido en la matriz anterior. Se realiza un promedio por filas de la matriz y los valores resultantes son los pesos finales que cada criterio obtuvo luego de la comparación, estos valores se utilizarán como factores de ponderación en la sección 1.2 y en el capítulo 3 cuando se elaboren las matrices de Priorización Preliminar y Final respectivamente. La matriz de comparación normalizada se puede ver en el Anexo 2.

A continuación se muestra el resumen de la matriz de comparación normalizada, indicando el orden de preferencia de los criterios. Cada valor se ha multiplicado por 100 con el fin de apreciar las diferencias en las preferencias.

Criterio	Valor (normalizado x100)
Control de transacciones y concurrencia (F3)	17.183
Curva de Aprendizaje (FD1)	15.294
Tiempo de ejecución de operaciones CRUD - Create (R1)	9.619
Tiempo de ejecución de operaciones CRUD - Retrieve (R2)	9.619
Tiempo de ejecución de operaciones CRUD - Update (R3)	9.619
Tiempo de ejecución de operaciones CRUD - Delete (R4)	9.619
Tiempo Generación Modelo de Entidades (R5)	5.145
Definición de metadatos (F1)	5.046
Generación de consultas (F2)	4.283

Facilidad de interacción con otras tecnologías (FD3)	3.607
Madurez del Framework (G1)	2.805
Costo de desarrollo (RS2)	2.253
Herramientas de desarrollo (FD2)	1.873
Requerimientos de HW y SW (RS1)	1.739
Tipo de Licenciamiento (G3)	1.397
Soporte y Documentación (G2)	0.898

Tabla 14 Resumen de la Matriz de comparación.

Elaborado por: El Autor.

1.2 Análisis de características de Ado .Net Entity Framework

1.2.1 Descripción de Ado .Net Entity Framework

ADO.NET Entity Framework es una mejora a ADO.NET[5] creada por Microsoft, que dota a los desarrolladores de un mecanismo adicional para acceder a los datos y trabajar con los resultados, además de los Data Readers[6] y Data Sets[7] utilizados en versiones anteriores.

Utilizando esta tecnología se pueden desarrollar aplicaciones de escritorio, aplicaciones cliente – servidor, aplicaciones web, utilizando herramientas complementarias que nos ayuden a lograr estos propósitos como ASP.NET, Silverlight o Windows Communication Foundation (WCF).

ADO.Net Entity Framework se da a conocer en Julio del 2008 como parte de Visual Studio 2008 Service Pack 1 y .NET 3.5 Service Pack 1, dos años después de que Microsoft lo anunciara en el evento Tech – Ed en el 2006 (Wikipedia, 2010).

La segunda y actual versión de Entity Framework fue publicada como parte de .NET 4.0 el 12 de abril de 2010 respondiendo muchas de las críticas de la primera versión. Una tercera versión todavía sin nombre, tiene su liberación planeada para

[5] ADO .NET→ Glosario de términos 1.5

[6] Data Reader → Glosario de términos 1.6

[7] Data Set → Glosario de términos 1.7

el primer cuarto del 2011. Unos meses después de la salida de la segunda versión han surgido varios betas de una tercera versión y el último de estos se publicó en Diciembre del 2010.

Entity Framework (EF) abstrae la estructura de la base de datos y todo el acceso y almacenamiento de datos se realiza frente a un modelo conceptual de datos que refleja los objetos de negocio. Los ítems usados para representar el modelo conceptual se conocen como Entidades.

Una Entidad para Entity Framework es una instancia de un EntityType que gráficamente es una caja formada por varias propiedades que se pueden observar en la Figura 1-4.

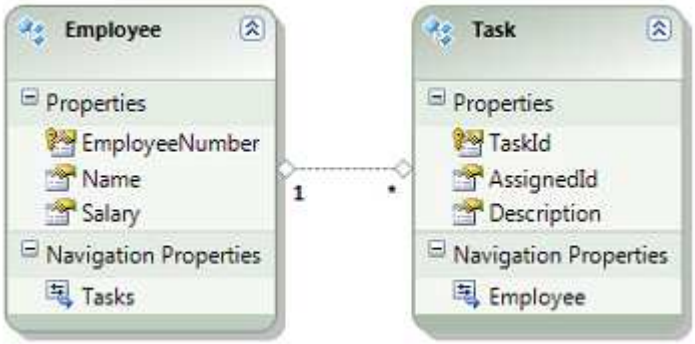

Figura 1-4 Representación gráfica de Entity Types.

Fuente: TENNY, Larry; HIRANI Zeeshan – Entity Framework 4.0 Recipes. Apress. 2010

De forma general un EntityType es una plantilla que representa un tipo de dato del negocio en el dominio de la aplicación. Está formado por varios componentes como: nombre del entity type, propiedades del entity type (pueden utilizar tipos de datos simples, compuestos (otros entity types o colecciones), propiedades de navegación (hacen referencia a otra entidad dentro de la asociación), entity key (propiedad utilizada como identificador del EntityType) y asociaciones. (Tenny & Hirani, 2010).

Cabe recalcar que un EntityType no representa exclusivamente a una tabla de la base datos ya que un EntityType puede estar formado por propiedades que representan diferentes tablas de una base de datos.

1.2.1.1 Arquitectura Entity Framework

Los principales componentes que forman la arquitectura de EF que se pueden observar en la Figura 1-5 serán analizados a continuación:

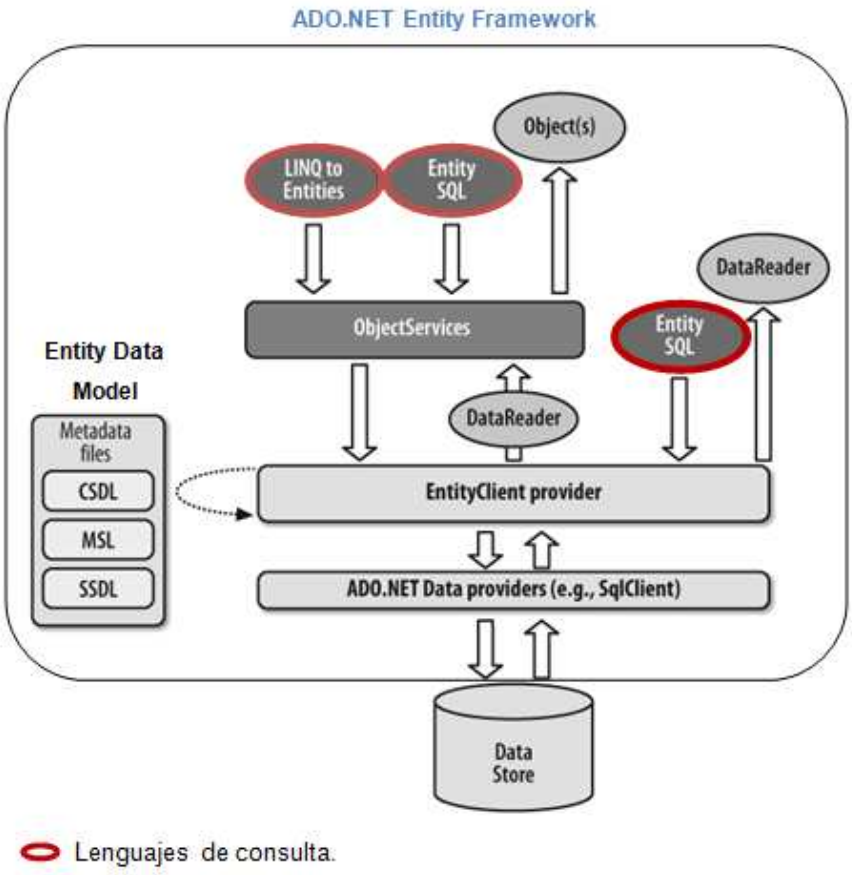

Figura 1-5 Arquitectura ADO .Net Entity Framework

Fuente: LERMAN, Julia. Programming Entity Framework, 1st Edition. O'Reilly Media. 2010

1.2.1.1.1 Componentes – Entity Framework

- Data Providers. - Esta capa le permite a Entity Framework comunicarse con una base de datos específica a través de un proveedor de datos con soporte ADO .NET. Hace unos años sólo el proveedor de SQL Server era soportado como proveedor ADO.NET pero ahora existen proveedores de datos de terceros como Oracle, MySQL, Postgres SQL (Npgsql), Sybase SQL Anywhere, Firebird, IBM que ofrecen esta característica.

- Entity Data Model (EDM). – Entity Data Model permite describir la estructura de los datos de una aplicación en forma de entidades y relaciones creando una vista conceptual de la información que se encuentra en una base de datos, xml, archivos de texto, informes, etc.

  Esto permite a los desarrolladores centrarse en el modelo conceptual sin tener que preocuparse por el esquema de almacenamiento. Entity Data Model materializa esta vista conceptual en un archivo de extensión *.EDX que consiste de tres partes que se pueden observar en la Figura 1-6 y que serán analizadas a continuación:

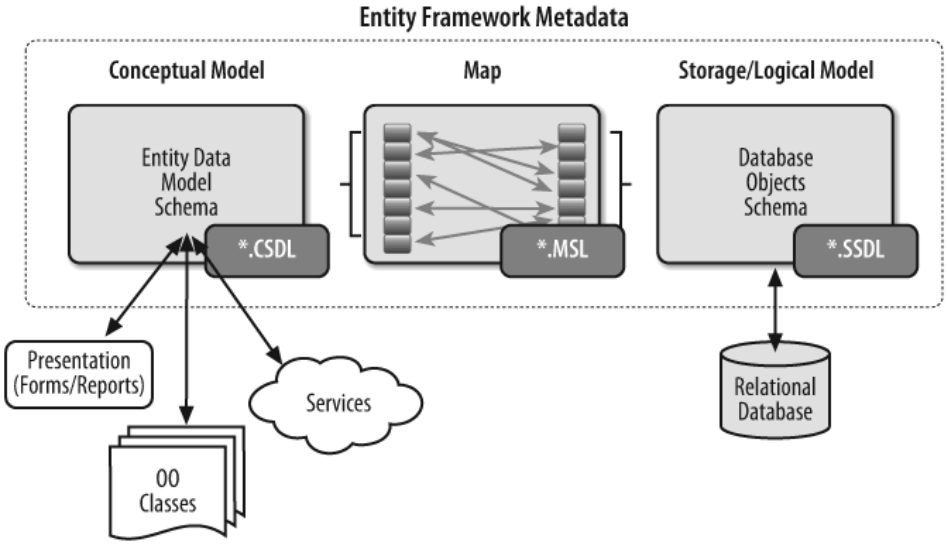

Figura 1-6 Estructura de EDM.

Fuente: LERMAN, Julia. Programming Entity Framework, 1st Edition. O'Reilly Media. 2010

1. Modelo Conceptual.- En esta sección se declaran y definen entidades, asociaciones y herencia, se utiliza el Lenguaje de definición del esquema conceptual (Conceptual Schema Definition Language con sus siglas en inglés CSDL).

2. Modelo Lógico / Almacenamiento.- En esta sección se describen las tablas, relaciones, etc. del contenedor de almacenamiento (base de datos) en el cual se va a persistir los datos, se utiliza el Lenguaje de

definición del esquema de almacenamiento (Store Schema Definition Language con sus siglas en inglés SSDL).

3. Mapeo.- En esta sección se define el mapeo de las entidades descritas en la sección CSDL con las tablas descritas en la sección SSDL, se utiliza el Lenguaje de especificación de Mapeo (Mapping Specification Language con sus siglas en inglés MSL).

- EntityClient.- El proveedor EntityClient sigue el modelo de otros proveedores de datos como SQLClient, OledbClient, y OracleClient y al igual que estos primero se crea una conexión, pero en lugar de apuntar a una fuente de datos física se realiza una conexión hacia un modelo conceptual representado en la capa EDM.
  Al obtener la conexión EntityClient permite crear comandos de consulta usando una sintaxis ideada para consultar entidades que es muy similar al lenguaje SQL. Esta sintaxis de consulta utilizada por esta capa es conocida como Entity SQL.

- Object Services. – La mayor parte de trabajo que se realiza con Entity Framework, tiene que ver con objetos que se basan en las entidades definidas en EDM. El API Object Services es la parte del framework que crea y administra estos objetos. Al igual que EntityClient esta capa también permite realizar consultas, la diferencia es que cuando se trabaja directamente con EntityClient los resultados que se obtienen son DataReaders (resultados tabulares solo de lectura) pero al trabajar con Object Services se puede transformar los datos tabulares creados por EntityClient en objetos, con la capacidad de realizar cambios en los valores de sus propiedades y retornarlos a la base de datos. Como se puede observar en la Figura 1-5, el API Object Services permite crear, procesar consultas expresadas en LINQ[8] to Entities o utilizando Entity SQL, y transformar los resultados de estas consultas en objetos.

[8] LINQ → Glosario de términos 1.8

1.2.1.1.2 Lenguajes de Consulta – Entity Framework

Entity Framework permite consultar el modelo conceptual de datos descrito a través Entity Data Model de varias formas, ya sea utilizando lenguajes de nativos de EF como Entity SQL o a través de lenguajes provistos por el framework .Net como es el caso de LINQ to Entities.

- Entity SQL (eSQL)
  Es un lenguaje derivado de SQL con modificaciones para consultar y manipular entidades definidas en el Entity Data Model (EDM). Entity SQL permite realizar consultas entre entidades que manejen herencia o relaciones. Tanto el componente Object Services como el componente Entity Client pueden ejecutar sentencias expresadas con eSQL.

- LINQ to Entities
  Es una modificación de LINQ (Language-Integrated Query) para realizar consultas en las entidades definidas en el Entity Data Model mediante Visual Basic o Visual C#.
  LINQ es un lenguaje que fue añadido a VisualBasic y C# en .Net 3.5 y fue creado para proveer un lenguaje de consulta independiente que puede ser utilizado por todos los objetos que conforman .Net. Actualmente existen varias implementaciones de este lenguaje y una de estas es LINQ to Entities.

1.2.1.2 Herramientas asociadas con Entity Framework

Microsoft a través de Visual Sudio ofrece un conjunto de herramientas que permiten desarrollar aplicaciones utilizando EF:

- Diseñador del Modelo de Entidades.- Permite crear y modificar de forma visual las entidades, asociaciones, mapeos, herencias y relaciones. En resumen nos permite crear el modelo conceptual, el modelo lógico y el mapeo en lugar de escribir manualmente cada sección del archivo *edmx. Esta herramienta también permite generar código C# o Visual Basic de los objetos que forman parte del modelo creado.

- Asistente del Modelo de Entidades.- Permite generar un modelo conceptual desde una base de datos existente.

- Asistente para creación de Base de Datos.- Permite crear primero un modelo conceptual para luego crear la base de datos que se derive de este modelo. Esta característica sólo es aplicable con SQL Server.

- Asistente de actualización del Modelo.- Permite actualizar el modelo conceptual, el modelo de datos y el mapeo cuando se han realizado cambios en la base de datos.

1.2.1.3 Características Especiales

Con la segunda versión de Entity Framework publicada hasta el momento, se destacan las siguientes características:

- Model – First (Enfoque Top - Down).- Esta característica fue agregada en la segunda versión de EF junto con .Net 4.0 y Visual Studio 2010 y permite crear un modelo conceptual partiendo desde un modelo vacío para luego generar la base de datos correspondiente a este modelo. Esta característica se aplica cuando se trabaja con SQL Server.

- Database – First (Enfoque Bottom - Up).- Esta característica se encuentra desde la primera versión de EF con Visual Studio 2008 SP1 y permite crear un modelo conceptual desde una base de datos existente.

- Soporte POCO (Plain Old CLR Objects). - EF en su segunda versión tiene la capacidad de manejar objetos simples de .Net (objetos POCO) que no extienden de ninguna clase o interfaz especial. Con esta nueva característica se puede desarrollar aplicaciones multicapas (N - tiers) reutilizando estos objetos en cualquiera de las capas ya que son objetos planos sin referencias que los ate a alguna capa específica.

- Lazy Loading.- Esta característica se encuentra en la segunda versión de EF y permite posponer la inicialización de un objeto hasta el punto en que sea necesario. Esta tarea en la primera versión de EF se la realizaba con

inicializaciones manuales de los objetos desde código ahora con esta característica la carga de objetos dependientes de otros se lo realiza de forma automática. La inicialización diferida de objetos si es apropiadamente utilizada puede contribuir a la eficiencia en el funcionamiento de un programa.

1.2.1.4 Comunicación con otras capas

Entity Framework es una herramienta que puede ser incluida en ambientes donde la lógica de negocio, datos y presentación se encuentran en el lado del cliente o en ambientes multicapa (N -Tier) donde se requiere un mayor desglose de componentes.

EF como se ha podido observar en puntos anteriores de este trabajo, provee APIs que permiten crear aplicaciones cliente sin mucha ayuda de otras herramientas, pero al tratarse de aplicaciones donde la arquitectura es multicapa Entity Framework tiene la capacidad de acoplarse a otras tecnologías. Es por esta razón que Microsoft ofrece una gamma de herramientas que faciliten el mantenimiento y escalabilidad de cada una de las capas para al final obtener una aplicación robusta, con componentes reusables.

En la Figura 1-7 se puede observar las herramientas que se pueden utilizar en las diferentes capas de una aplicación N – Tier y la ubicación de Entity Framework en este tipo de arquitecturas.

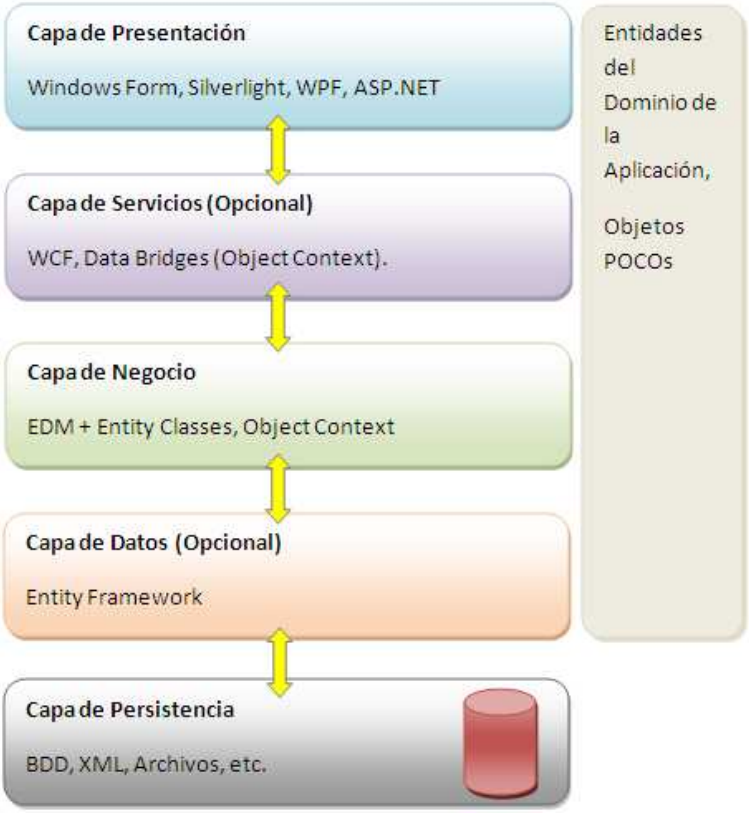

Figura 1-7 Arquitectura Multicapa – Entity Framework

Elaborado por: El Autor.

1.2.1.5 Licenciamiento y Requerimientos de Software y Hardware

A pesar de que Entity Framework por ser parte del Framework de .Net de Microsoft no se encuentra regido a pagos de licencia ya que existe una versión que se distribuye de forma gratuita y libre (.NET Framework Redistributable Package), si se debería tomar en cuenta para el licenciamiento, la herramienta que nos va a facilitar el desarrollo de aplicaciones utilizando EF:

- Visual Studio 2008 o 2010 (únicas versiones hasta el momento que dan soporte a Entity Framework) para la parte gráfica, modelamiento, etc., se dispone de una versión Express sin costo, con algunas limitaciones funcionales pero suficiente para ejemplificar y crear aplicaciones de nivel intermedio, y versiones con pago cuyas licencias varían entre los $500 (Professional Edition) y $11000 (Ultimate Edition) que ofrecen un IDE más completo para aplicaciones más complejas. Para el presente proyecto se

utilizará la versión Professional de Visual 2010 que cuenta con la funcionalidad necesaria para cumplir con el objetivo de este trabajo. (Microsoft, 2011).

En cuanto a los requerimientos de hardware y software ya que EF sólo es soportado desde .Net Framework 3.5 SP1 con Visual Studio 2008, las características mínimas de Hardware y Software son (Microsoft, 2011):

- Hardware
  - Procesador: 1,6 Ghz o superior.
  - RAM: 384 MB o más.
  - Disco: 2 GB de espacio libre.
  - Pantalla con una resolución mínima de 1024 x 768.

- Software
  - Windows Installer 3.1 o posterior.
  - Internet Explorer 5.01 o posterior.

1.2.2 Valoración de criterios de evaluación para Ado .Net Entity Framework

1.2.2.1 Madurez del Framework

Entity Framework desde su creación en el 2008 ha soportado cambios para ofrecer al desarrollador una herramienta capaz de competir en el campo de los frameworks de persistencia, cubriendo necesidades del mercado mediante la adopción de modelos y tecnologías actuales, transparentando y facilitando el desarrollo de aplicaciones empresariales en la capa de persistencia. Por otro lado el corto tiempo que tiene el framework en producción limita de alguna forma su retroalimentación y notoriedad ya que recientemente ha sido introducido en el mercado y se lo está dando a conocer.

Valoración final: 3

1.2.2.2 Soporte y documentación del framework

Microsoft es la empresa a cargo del soporte de Entity Framework. El soporte para productos Microsoft incluyendo Entity Framework puede realizarse utilizando el centro de soporte Microsoft cuyo sitio web es: http://support.microsoft.com/?ln=es-es , la red de desarrolladores de Microsoft (MSDN) en la cual se puede encontrar información, noticias, ejemplos y descargas de productos Microsoft. Su sitio web es: http://www.msdn.com. A través de los blogs que ofrece la comunidad de desarrolladores Microsoft. Su sitio web es: http://blogs.msdn.com o a través de contacto con Profesionales Microsoft para asistencia web o telefónica con cargos económicos.

En cuanto a documentación la comunidad de desarrolladores Microsoft también es una fuente bastante extensa de documentación que tiene Entity Framework y a pesar de no tener mucho tiempo en el mercado alrededor de la web se puede encontrar un gran número de foros, blogs, presentaciones, libros y videos con información y ejemplos que permiten un mayor conocimiento del framework.

Un punto a tomar en cuenta es que Microsoft no ofrece una certificación específica para Entity Framework ya que al ser parte de .Net Framework la certificación se obtiene a través de esta última.

Valoración final: 4

1.2.2.3 Tipo de licenciamiento

Entity Framework al ser parte de .Net Framework no tiene costo alguno para desarrollo a pesar de que al momento de la instalación es necesario aprobar un acuerdo de tipo EULA el cual da ciertas restricciones de uso y distribución. Por otra parte ya que EntityFramework se implementa a través de Visual Studio se va a tomar en cuenta el tipo de licenciamiento que tiene este último.

Visual Studio es una herramienta que utiliza una licencia propietaria comercial en la versión Professional que se va a utilizar para el desarrollo del prototipo de este trabajo.

Valoración final: 1

1.2.2.4 Definición de metadatos

Entity Framework permite realizar el mapeo ORM, representación de la base datos, manejo de entidades y configuraciones de los proveedores de base de datos a través de archivos XML. Existen indicios que en futuras versiones se permita el uso de anotaciones para realizar las mismas funciones que los XML utilizados hasta el momento, como lo hacen otros frameworks de persistencia existentes.

Valoración preliminar: 3

1.2.2.5 Facilidades de generación de consultas (Queries).

Entity Framework además de dar soporte al lenguaje nativo de consultas (SQL), ofrece lenguajes de consulta propios del framework que facilitan la interacción entre los objetos y la base de datos. Los lenguajes que trabajan con este framework son eSQL y Linq to Entities, cada uno de estos lenguajes dota al framework de grandes recursos de consulta para objetos como lo hace SQL con las tablas en una base de datos relacional.

Valoración preliminar: 4.

1.2.2.6 Control de transacciones y concurrencia.

Entity Framework ofrece librerías propias del framework para el manejo de transacciones y concurrencia que son complementadas con librerías del framework de .Net desde la versión 3.5 SP1.

Valoración preliminar: 3.

1.2.2.7 Requerimientos de Hardware y Software para el desarrollo.

Entity Framework no puede ser utilizando en un sistema operativo que no sea Windows en las versiones descritas en el punto 1.2.1.5, además por sí solo no puede activar sus características de framework de persistencia para esto se necesita software adicional como Visual Studio 2010 que trabaja en conjunto con Entity Framework para poner en marcha las características del framework. En cuanto a Hardware no se necesita de componentes especiales.

Valoración preliminar: 3.

1.2.2.8 Costo de Desarrollo.

Entity Framework no tiene costo por ser parte de .Net Framework que se lo distribuye de forma gratuita, pero al necesitar de Visual Studio para su implementación, el costo de desarrollo tomando en cuenta una licencia de tipo Profesional para Visual 2010 que ofrece características suficientes para un desarrollo de aplicaciones empresariales asciende a un valor de $549.

Valoración final: 4.

1.2.2.9 Curva de aprendizaje

Entity Framework considera que al ser un framework orientado a la interacción de objetos con base datos en primer lugar se debe tener experiencia con lenguajes orientados a objetos como C# con lo cual requiere de un esfuerzo moderado para conseguir un manejo adecuado de sus componentes. Además se requiere experiencia en base de datos que con lleva al manejo de SQL, lenguaje de consulta base para los utilizados por este framework. En complemento con la experiencia en el campo de objetos y base de datos se suma el conocimiento a nivel medio en el manejo de Entity Framework el cual puede tomar un tiempo un considerable a pesar de que existe un buen número de foros, blogs, libros y ejemplos que pueden ayudar de gran manera en el aprendizaje del framework.

Valoración preliminar: 2.

1.2.2.10 Herramientas de desarrollo

Entity Framework al ser parte de las librerías de Visual Studio permite la creación de aplicaciones empresariales bajo la misma herramienta desde la capa de presentación hasta la capa de persistencia, esta característica evita al desarrollador el uso de herramientas de terceros que pueden implicar costos extras de desarrollo o curvas de aprendizaje más extensas.

Valoración preliminar: 4.

1.2.2.11 Facilidad de interacción con otras tecnologías

Entity Framework para la persistencia de datos en sus inicios únicamente podía trabajar con SQL, limitando mucho su trabajo con otras tecnologías.

Actualmente existe una gran cantidad de proveedores de base de datos que le permiten a Entity Framework interactuar con un mayor número de tecnologías en la capa de datos.

Para las otras capas, Microsoft ofrece sus propios APIs de integración que integran Entity Framework con herramientas de la misma línea de negocio. No existe documentación o proyectos en los que se pueda observar una fácil interacción de Entity Framework con tecnologías que no sean Microsoft.

Valoración preliminar: 2.

1.3 Análisis de características de EJB3 (Enterprise Java Beans 3)

1.3.1 Descripción de Enterprise Java Beans 3

Enterprise Java Beans (EJB) es un framework que forma parte de la especificación JEE (Java Enterprise Edition)[9] y permite construir aplicaciones de negocio portables, reusables y escalables usando el lenguaje de programación Java.

EJB fue originalmente desarrollada por IBM en 1997 y luego adoptada un año más tarde por Sun Microsystems (Oracle en la actualidad). En Mayo del 2006 tras los constantes reclamos por las limitaciones que presentaron las versiones anteriores y el surgimiento de nuevas herramientas open source que ofrecían alternativas a EJB, aparece EJB 3 con un esquema totalmente diferente a sus predecesoras versiones, simplificando el desarrollo de aplicaciones y estandarizando el API de persistencia para la plataforma Java. En la Figura 1-8 se puede observar los componentes que forman la arquitectura JEE con el nuevo contenedor EJB v3 y el nuevo API de persistencia:

[9] JEE → Glosario de términos 1.9

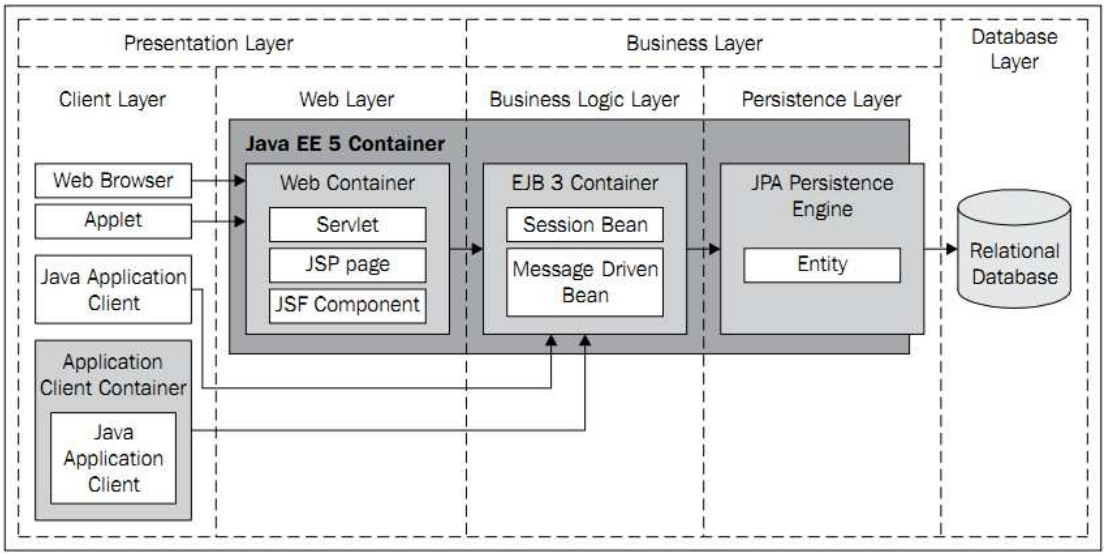

Figura 1-8 Arquitectura JEE (EJB 3 y JPA)

Fuente: SIKORA, Michael – EJB3 Developer Guide.

El objetivo de EJB es dotar al programador de un modelo que le permita abstraerse de los problemas generales de una aplicación empresarial (concurrencia, transacciones, persistencia, seguridad, etc.) para centrarse en el desarrollo de la lógica de negocio en sí. El hecho de estar basado en componentes (beans) permite que éstos sean flexibles y sobre todo reutilizables.

Los beans utilizados a partir de EJB3 son considerados objetos POJO[10] con características especiales (transaccionabilidad, persistencia, seguridad, etc.) que se ejecutan en un servidor de aplicaciones, destinados al desarrollo y despliegue de aplicaciones empresariales Java, por esta razón este tipo de componentes reciben el nombre de componentes de negocio Java (Enterprise Java Beans con sus siglas en inglés EJBs). Estas características especiales son configurables mediante metadata asociada al código de la clase POJO utilizando anotaciones (alternativa a los descriptores XML de despliegue usado en versiones anteriores) o descriptores XML, teniendo preferencia los últimos en caso de sobreponerse la metadata.

[10] POJO → Glosario de términos 1.10

Los beans de negocio (EJBs) de forma general desde la versión EJB3 se clasifican en (Sikora, 2008):

- Beans de Sesión.- Representan una conversación temporal entre los componentes disponibles en el servidor y el cliente.
- Beans dirigidos por Mensajes.- Son usados para invocar métodos de forma asincrónica. Encapsulan el popular servicio de mensajería de Java (Java Message Service con sus siglas en inglés JMS).

Esta nueva forma de clasificar los beans del modelo EJB, es debido a la estandarización del API de persistencia que se dio a partir de EJB 3 para el mapeo de objetos Java a una base de datos relacional, lo cual permitió reemplazar el manejo de datos que se realizaba en versiones anteriores a través de los Beans de Entidad a un nuevo modelo robusto y confiable de persistencia conocido como Java Persistence API (JPA). Esta nueva especificación ha tenido tal acogida que los frameworks ORM del mercado se han adaptado a esta especificación y, hoy en día, cualquier framework de persistencia ORM Java soporta JPA.

En términos de JPA, los datos persistentes son conocidos como entidades. La entidad hace referencia a una colección lógica de datos que se pueden almacenar o recuperar en conjunto. Una entidad para JPA puede traducirse a una fila de una tabla en una base datos, mientras que sus propiedades serían cada una de las columnas que la conforman.

Además esta nueva especificación puede trabajar de forma nativa en el interior de un contenedor JEE o independientemente del resto de los componentes de EJB 3 y hasta puede ser usado en una aplicación desktop común y corriente, una aplicación Java SE (Standard Edition).

Finalmente la nueva versión de JPA con versión 2.0 fue aprobada en Diciembre del 2009 con el objetivo de hacer frente a las características que estaban presentes en algunos de los vendedores ORM populares, que no fueron posibles

con JPA 1.0. JPA 2 incluye actualizaciones para el mapeo objeto – relacional y soporte a consultas dinámicas.

Para el análisis de los frameworks que se va a realizar en este trabajo, de todos los componentes que ofrece el entorno EJB 3, se va a tomar en cuenta el API especializado en la persistencia de datos que ofrece este framework, conocido con el nombre de Java Persitence API (JPA).

1.3.1.1 Arquitectura JPA

Los principales componentes que forman la arquitectura JPA que se pueden observar en la Figura 1-9 serán analizados a continuación:

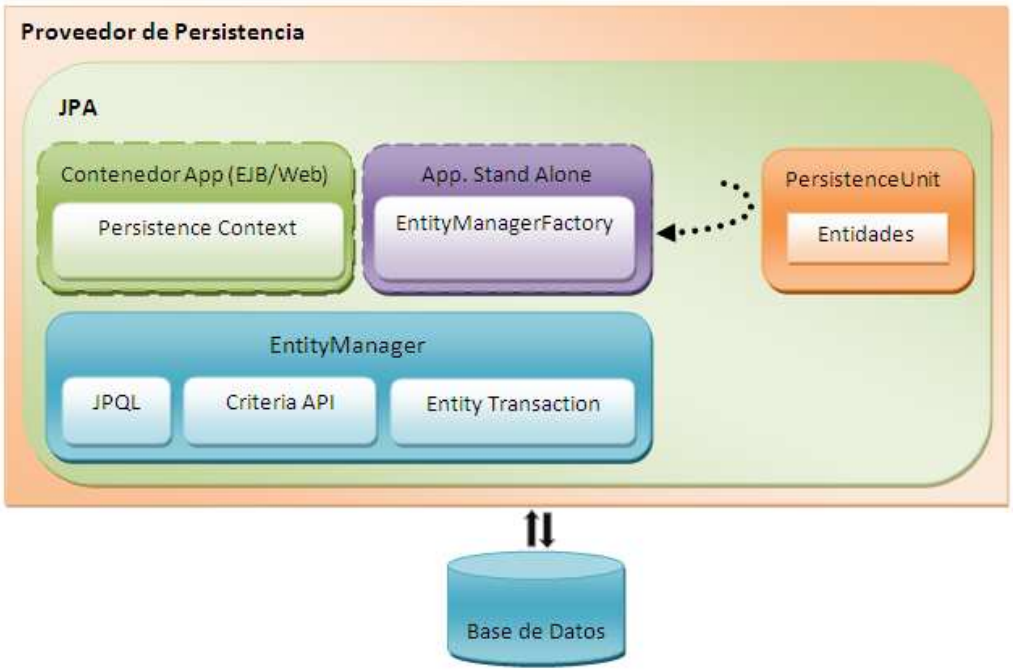

Figura 1-9 Arquitectura JPA.

Elaborado por: El Autor.

1.3.1.1.1 Componentes – Entity Framework

- Persistence Unit / Unidad de Persistencia.- Agrupa el conjunto de entidades que serán utilizadas en el proceso de persistencia. La definición de las unidades de persistencia con sus entidades se lo realiza en el archivo

persistence.xml. Cada unidad de persistencia tiene asociado un nombre único para evitar conflictos al momento de ser referenciada.

- EntityManager.- Provee servicios para interactuar con las entidad definidas en las unidades de persistencia. EntityManager es utilizado para crear, remover, buscar, actualizar y consultar (JPQL, Criteria API) sobre las entidades existentes en un contexto de persistencia.

- EntityManagerFactory.- Este componente basa su función en un patrón de diseño denominado Factory, que permite crear objetos de forma dinámica. Una aplicación JSE (Stand Alone) utiliza esta interfaz para asociar una unidad de persistencia a un EntityManager. Por lo tanto EntityManagerFactory proveerá un repositorio de instancias de tipo EntityManager construidas para una unidad de persistencia y el conjunto de instancias que proporcione serán para esa unidad de persistencia.
  Cuando se utiliza un EntityManagerFactory para asociar una unidad de persistencia a un EntityManager se dice que la persistencia es administrada por la aplicación. En este caso la gestión que realice EntityManager al ciclo de vida de las entidades se lo deberá programar manualmente.

- Persistence Context.- En una aplicación JEE, el Contenedor de Aplicaciones inyecta una referencia directa a EntityManager para asociarle una unidad de persistencia a través de la anotación “@PersistenceContext” haciendo el código más simple. El contexto de persistencia lleva un registro del estado y/o los cambios que puedan ocurrirle a una entidad.

  Al utilizar un PersistenceContext para asociar una unidad de persistencia a un EntityManager se dice que la persistencia es administrada por el contenedor. En este caso la gestión que realice EntityManager al ciclo de vida de las entidades es totalmente transparente al desarrollador ya que lo realiza el contenedor de aplicaciones EJB (Jboss, GlassFish, etc) o Web (Tomcat, Jetty, etc).

- EntityTransaction.- Provee funcionalidades para el manejo de transacciones en una base datos (begin, commit, rollback, etc).

- Proveedor de Persistencia.- Debido a que JPA es una especificación, para solucionar la persistencia de un sistema se debe decidir que implementación de dicha especificación ha de usarse. Las diferentes implementaciones son denominadas proveedores de persistencia que evitan al desarrollador el manejo manual de las tareas relacionadas con persistencia usando esta especificación. Algunos de los proveedores de persistencia que implementan JPA son: Hibernate, Oracle TopLink / Eclipse Link y Open JPA.

1.3.1.1.2 Lenguajes de Consultas – JPA

JPA permite realizar consultas sobre las entidades, ya sea utilizando el lenguaje de consulta propio de JPA conocido como JPQL, o a través de un API especializado en consultas integrado desde la segunda versión JPA, conocido con el nombre de Criteria API.

- Java Persistence Query Language (JPQL).-

  JPQL es el lenguaje de consulta estándar de JPA a partir de la versión EJB 3, que sustituye y mejora el lenguaje de consulta EJBQL, introducido por primera vez en EJB 2.0. JPQL es un lenguaje de consulta portable diseñado para combinar la sintaxis y la semántica simple de SQL con la orientación a objetos. Las consultas creadas utilizando este lenguaje pueden ser compiladas a SQL en los servidores de bases de datos más conocidos. Este lenguaje permite realizar consultas sobre un modelo de objetos formado por entidades en lugar de utilizar un modelo físico de datos formado por tablas.

Adoptar el lenguaje JPQL no significa perder las bondades que estamos acostumbrados a usar y observar en SQL. Las características de SQL son directamente soportadas incluyendo sub consultas, funciones de agregación, operaciones para actualizar/eliminar/insertar, numerosas funciones SQL y muchas otras más.

- Criteria API.-

JPA 2.0 introduce un nuevo API para definir consultas que trabajen sobre el modelo de entidades. Criteria API utiliza un lenguaje estructurado orientado a objetos en lugar de representar consultas concatenando cadenas simples de texto al estilo de JPQL. Este nuevo lenguaje de consultas orientado a objetos al estar asociado con el compilador de JAVA no necesita de ninguna herramienta especial para obtener validación de sintaxis, auto llenado de sentencias y refactorización.

1.3.1.2 Herramientas asociadas con JPA

JPA al ser un estándar para el entorno Java cuenta con un gran número de opciones de herramientas que facilitan la creación de aplicaciones utilizando este framework. A continuación se mencionan las herramientas que a criterio subjetivo brindan más productividad al momento de crear una aplicación con JPA:

- Asistente del modelo de Entidades.- Permite la creación de entidades desde las tablas de una base de datos.
- Asistente para Anotaciones.- Provee una interfaz gráfica para trabajar con las anotaciones JPA y definir el mapeo de las entidades con las tablas y columnas de una base de datos.
- Asistente de configuración de archivos.- Provee una interfaz gráfica, que permite la configuración de los archivos persistence.xml y orm.xml (utilizado para definir el mapeo objeto relacional en lugar de utilizar anotaciones).

- Asistente modelo Entidades – POJO.- Provee un asistente para la creación de entidades partiendo desde clases simples de Java (POJO) y el mapeo de objetos POJO con campos de la base de datos.

1.3.1.3 Características Especiales

Hasta la versión de JPA publicada al momento, se puede recalcar las siguientes características especiales:

- Soporte POJO.- Tal vez el aspecto más importante de JPA es el hecho de que los objetos son POJO, lo que significa que no existe nada especial en los objetos que se van a persistir. Lo único que necesitan estos objetos para su interacción con la base de datos lo provee la metadata que es expresada con anotaciones.

- Integración y facilidad de pruebas.- Las aplicaciones multicapa alojadas en un servidor de aplicaciones se han convertido en un estándar en las arquitecturas de las aplicaciones. Las pruebas unitarias en un servidor de aplicaciones toman mucho tiempo y por lo general existe mucha dificultad. Es por esa razón que JPA fue pensado para trabajar tanto fuera (aplicaciones JSE) como dentro (aplicaciones JEE) del servidor de aplicaciones, permitiendo la ejecución de pruebas de forma cómoda y sencilla.

- Validación de Entidades.- A partir de JPA 2.0 se introduce una nueva característica conocida con el nombre de Bean Validation, esta característica permite validar fácilmente ingresos de datos de los usuarios a través de anotaciones asociadas a las entidades. Una vez asociada la anotación de validación con la entidad, EntityManager no permitirá persistir datos que no pasen la validación.

1.3.1.4 Comunicación con otras capas

JPA al ser parte de EJB 3 cuenta con beans (Beans de Sesión, Beans dirigidos por Mensajes) que de forma nativa le facilitan la comunicación con las demás capas de una aplicación dentro de una arquitectura JEE. Además al ser parte de Java, tiene la facilidad de interactuar con herramientas y servicios que le ofrece esta tecnología para crear aplicaciones empresariales.

En la Figura 1-10 se puede observar las herramientas que se pueden utilizar en las diferentes capas de una aplicación JEE y la ubicación de los Beans de EJB y JPA en este tipo de arquitecturas.

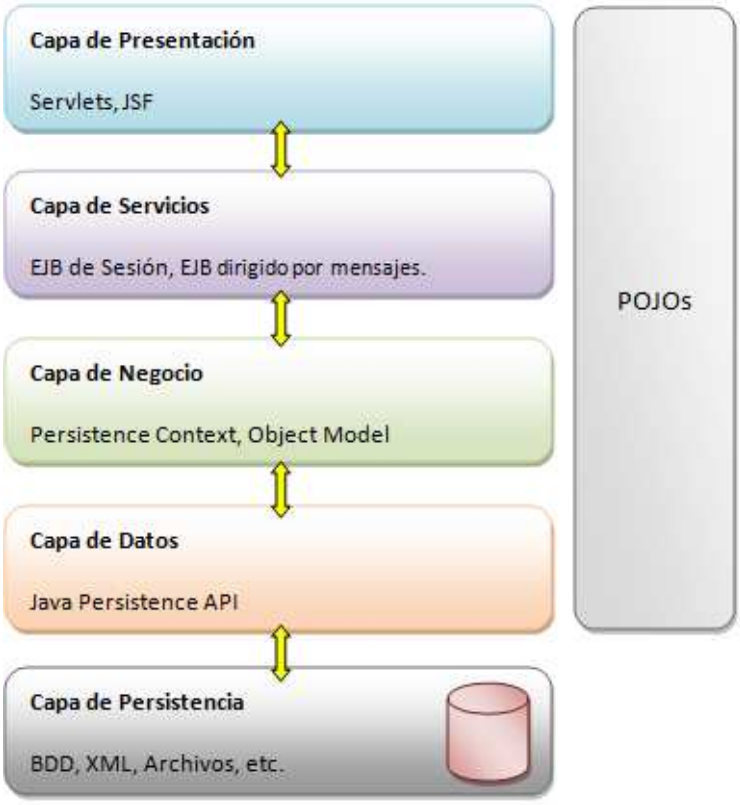

Figura 1-10 Ubicación de EJB y JPA en una Arquitectura JEE.

Elaborado por: El Autor.

1.3.1.5 Licenciamiento y Requerimientos de Software y Hardware

JPA y EJB como especificaciones tienen restricciones de derechos de autor y sólo podrá ser descargado, usado y reproducido para propósitos de evaluación. Como framework mantiene un licenciamiento GPL para implementación y distribución de aplicaciones utilizando JPA.

Por otra parte hay que tomar en cuenta el tipo de licenciamiento de la herramienta que permite implementar JPA que para el presente trabajo será OpenJPA, esta herramienta maneja un licenciamiento de tipo Apache License 2.0 que permite el uso del código fuente para desarrollos de software propietario o software open source sin costo para el desarrollador pero con posibilidad de distribuir los trabajos realizados utilizando esta licencia de forma comercial o manteniendo políticas de un producto de software libre.

En cuanto a requerimientos de software es necesario:

- Un proveedor de persistencia que soporte la especificación JPA.
- Un contenedor de Aplicaciones que de soporte a proveedores de persistencia con características del punto anterior para aplicaciones JEE.
- Un IDE de programación con soporte JAVA para el desarrollo de la capa de negocios y presentación.

En lo que tiene que ver con requerimientos de hardware no existen especificaciones concretas ya que varían dependiendo del proveedor de persistencia, IDE y/o contenedor de aplicaciones seleccionado.

1.3.2 Valoración de criterios de evaluación para EJB3/JPA

1.3.2.1 Madurez del Framework

JPA como parte de EJB3 tiene la gran ventaja de estar formado por componentes de persistencia que han sido utilizados, probados y certificados por más de diez años, desde las primeras versiones de EJB por medio de los Entity Beans hasta la estandarización del framework de persistencia con la adopción de las mejores prácticas de otros frameworks de persistencia como Hibernate, Toplink, JDO.

Este constante cambio y adopción de nuevas estrategias, le permite a EJB ofrecer un framework de persistencia estándar, confiable y robusto para el entorno Java, logrando tal acogida que hoy por hoy la mayoría de tecnologías ORM dan soporte a JPA.

Valoración final: 4

1.3.2.2 Soporte y documentación del framework

JPA al ser una especificación no cuenta con el servicio de soporte que tienen el resto de aplicaciones comunes. Actualmente Oracle es la empresa a cargo de todo lo concerniente a información y mantenimiento de JPA. Esta empresa ofrece una red de tecnologías Oracle (Oracle Technology Network) en la cual se puede encontrar documentación, artículos, noticias, foros, ejemplos y descargas de productos Oracle incluyendo JPA. Su sitio web es: http://www.oracle.com/technetwork. Además se puede obtener mucha información de funcionalidad y arquitectura a través de los blogs, videos, ejemplos y documentación que ofrecen las comunidades de desarrolladores que implementan o dan soporte a JPA como IBM, Hibernate, OpenJPA, etc.

Un punto a tomar en cuenta es que Oracle ofrece una certificación específica para JPA (Oracle Certified Expert, Java Platform, Enterprise Edition 6 Java Persistence API Developer) la cual permite el crecimiento profesional para los desarrolladores que utilizan esta tecnología.

Valoración final: 4

1.3.2.3 Tipo de licenciamiento

EJB y JPA al igual que los componentes que permiten su implementación manejan licenciamientos con soporte a GPL pero con variaciones que permiten desarrollar aplicaciones tanto comerciales como open source si costos para el programador ni restricciones en su uso y al momento de distribuir el producto realizado utilizando estas herramientas se pueda decidir en mantenerlo comercial o libre.

Valoración final: 2

1.3.2.4 Definición de metadatos

JPA permite realizar el mapeo ORM, representación de la base datos, manejo de entidades y configuraciones de los proveedores de base de datos a través de metadata que puede se expresada de dos formas diferentes, ya sea utilizando Anotaciones o a través de archivos XML.

Además JPA da la facilidad de utilizar los dos tipos de metadata teniendo preferencia los archivos XML en caso de que se sobrepongan.

Valoración preliminar: 4

1.3.2.5 Facilidades de generación de consultas (Queries).

JPA ofrece lenguajes de consulta propios del Framework que facilitan la persistencia de la información en un entorno orientado a objetos. Desde la versión 2.0 se incorporó un nuevo mecanismo (Criteria API) que evoluciona las consultas que esta ese momento se las realizaba con el lenguje estándar de JPA para consultas JPQL, el cual facilita la creación de consultas dinámicas y permite que el desarrollador estructure consultas utilizando objetos en lugar de concatenar cadenas de texto.

Valoración preliminar: 4.

1.3.2.6 Control de transacciones y concurrencia.

JPA al ser parte de EJB la transaccionalidad y la concurrencia pueden ser gestionadas desde aplicaciones JEE o JSE. Con las primeras el manejo de estas operaciones se puede realizar a través de los beans de sesión o delegando este trabajo al contenedor o servidor de aplicaciones. Para las últimas JPA ofrece de forma nativa un componente especializado en el manejo de transacciones y concurrencia formado por varios métodos que facilitan y transparentan el manejo de este tipo de operaciones.

Desde la versión JPA 2.0 se agregó nuevas características al framework que le permiten ejecutar transacciones concurrentes con detección y prevención de colisiones.

Valoración preliminar: 3.

1.3.2.7 Requerimientos de Hardware y Software para el desarrollo.

EJB y JPA al ser frameworks para plataformas Java pueden ser utilizados en entornos Microsoft o Unix. JPA por su parte necesita de software adicional para su implementación a través de los proveedores de persistencia.

Requerimientos especiales de hardware no son necesarios para el desarrollo con esta herramienta.

Valoración preliminar: 3.

1.3.2.8 Costo de Desarrollo.

EJB y JPA no representan costos en el desarrollo así como las implementaciones de esta última si se sigue la línea de productos que deriven de GPL. En cuanto a utilitarios que faciliten el desarrollo de aplicaciones utilizando este framework, Java ofrece IDEs como Eclipse y NetBeans que de igual manera evitan costos de desarrollo, contrario a IBM que ofrece utilitarios que dan soporte a JPA pero su adquisición ya representarían gastos de desarrollo importantes.

Valoración final: 4.

1.3.2.9 Curva de aprendizaje

La curva de aprendizaje de JPA se basa en lograr un conocimiento medio de su especificación y manejo adecuado de una implementación, lo cual puede tomar un tiempo considerable si se toma en cuenta conceptos JEE, JSE, contenedores y servidores de aplicaciones a pesar de que existe suficiente información para este fin. Además se necesita contar con experiencia en un lenguaje orientado a objetos, siendo para este caso JAVA y en el lenguaje estándar para consultas SQL.

Valoración preliminar: 2.

1.3.2.10 Herramientas de desarrollo

Al ser JPA un estándar para la persistencia de datos para el entorno JAVA existe una considerable variedad de herramientas que implementan el framework, ya sea de la misma empresa Oracle o de terceros como IBM, Red Hat, Apache, etc. Así mismo Java ofrece IDE’s, plugins, etc que facilitan las tareas del desarrollador haciéndolas más transparentes y ágiles.

Valoración preliminar: 3.

1.3.2.11 Facilidad de interacción con otras tecnologías

Considerando que JPA forma parte de EJB la interacción con otras tecnologías se torna más dinámica ya que esta última de forma natural tiene integración con componentes empresariales facilitando la creación de aplicaciones empresariales completas.

Además existe suficiente información que ejemplifica la interacción de JPA con otras tecnologías en las diferentes capas de una aplicación, no solo desde el punto de vista de la empresa propietaria del framework Oracle sino también de todas las empresas que lo implementan.

Valoración preliminar: 4.

A continuación se detalla la Matriz de Priorización Preliminar para la valoración de los criterios definidos hasta este punto:

Matriz de Priorización Preliminar

La matriz descrita a continuación está formada por los valores establecidos en cada uno de los criterios analizados en las secciones 1.2.2 (Valoración de criterios de evaluación para Ado .Net Entity Framework) y 1.3.2 (Valoración de criterios de evaluación para EJB3/JPA).

	Madurez (G1)	Soporte y Documentación (G2)	Tipo de Licencia-miento (G3)	Definición de metadatos (F1)	Generación de consultas (F2)	Control de transacciones y concurrencia (F3)	Requerimiento de HW y SW (R1)	Costo de desarrollo (R2)	Curva de Aprendizaje (FD1)	Herramientas de desarrollo (FD2)	Facilidad de interacción con otras tecnologías (FD3)	Total
Entity Framework	3	4	1	3	4	3	3	4	2	4	2	**33**
EJB3/JPA	4	4	2	4	4	3	3	4	2	3	4	**37**

Tabla 15 Matriz de Priorización Preliminar.
Elaborado por: El Autor.

Matriz de Priorización Preliminar Normalizada

La Matriz de Priorización Preliminar Normalizada añade a su valoración los pesos de los criterios obtenidos en la matriz de comparación descrita en la sección 1.1.4 (Elaboración de la matriz de comparación).

	Madurez (G1)	Soporte y Documentación (G2)	Tipo de Licencia-miento (G3)	Definición de metadatos (F1)	Generación de consultas (F2)	Control de transacciones y concurrencia (F3)	Requerimiento de HW y SW (R1)	Costo de desarrollo (R2)	Curva de Aprendizaje (FD1)	Herramientas de desarrollo (FD2)	Facilidad de interacción con otras tecnologías (FD3)	Total
Entity Framework	0.08	0.04	0.01	0.15	0.17	0.52	0.05	0.09	0.31	0.08	0.07	**1.57**
EJB3/JPA	0.11	0.04	0.03	0.20	0.17	0.52	0.05	0.09	0.31	0.06	0.14	**1.71**

Tabla 16 Matriz de Priorización Preliminar Normalizada.
Elaborado por: El Autor.

2. DESARROLLO DEL PROTOTIPO BASE

2.1 Metodología de diseño

ICONIX es la metodología escogida para la fase de diseño debido a la flexibilidad y características que esta brinda entre las metodologías ágiles de desarrollo de software (véase Tabla 17).

	Metodología			
Características	**RUP**	**XP**	**ICONIX**	**Observaciones**
Dirigido a pequeños grupos de trabajo.	1	2	3	RUP no es apto para equipos pequeños o proyectos cortos ya que se convierte en un gasto de tiempo y dinero innecesario.
Dirigido a proyectos de corta duración.	1	2	3	
Utilizado para equipos jóvenes.	1	2	3	ICONIX es más fácil de implementar y aprender, por lo que los equipos de trabajo jóvenes lo pueden incorporar de una manera más natural.
Iteraciones entre fases.	3	3	3	
Disminución de jerarquías en el interior del equipo de trabajo.	1	2	3	
Menos documentación.	1	2	3	RUP tiene una rigurosa documentación, ICONIX y XP confían más en las personas.
	8	**13**	**18**	

Tabla 17 Comparativa de Metodologías Ágiles de desarrollo de Software.

Fuente: Proyecto de Titulación, BUSTAMANTE, Eduardo; MIÑO Alexandra. Desarrollo de un Sistema para la Gestión de Versiones de la Infraestructura TI. Aplicado a un caso de estudio. 2009

La puntuación más alta representa a la metodología que mejores características ofrece frente a desarrollos ágiles de software.

A continuación se describe rápidamente las bases de la metodología de diseño seleccionada:

El enfoque de ICONIX maneja casos de uso como RUP [11], implementa relativamente el análisis y diseño que hace XP[12], realiza un uso aerodinámico de UMLs y guarda un enfoque sutil al seguimiento de requisitos. Esta metodología no pierde de vista la perspectiva original de Jacobson (Universidad San Martin de Porres, 2009) para el manejo de casos de uso, generando de esta forma resultados concretos y específicos, obteniendo casos de uso fácilmente comprensibles, los mismos que pueden ser utilizados por un equipo para el desarrollo de un proyecto.

- **Enfoque Aerodinámico de Desarrollo.-**

  La Figura 2-1 muestra la naturaleza del enfoque aerodinámico de desarrollo, que incluye los diagramas UML necesarios para la implementación de un producto de software. Además permite que los involucrados en el desarrollo, puedan adecuarse fácilmente a los modelos utilizados en las diferentes etapas de la metodología ICONIX.

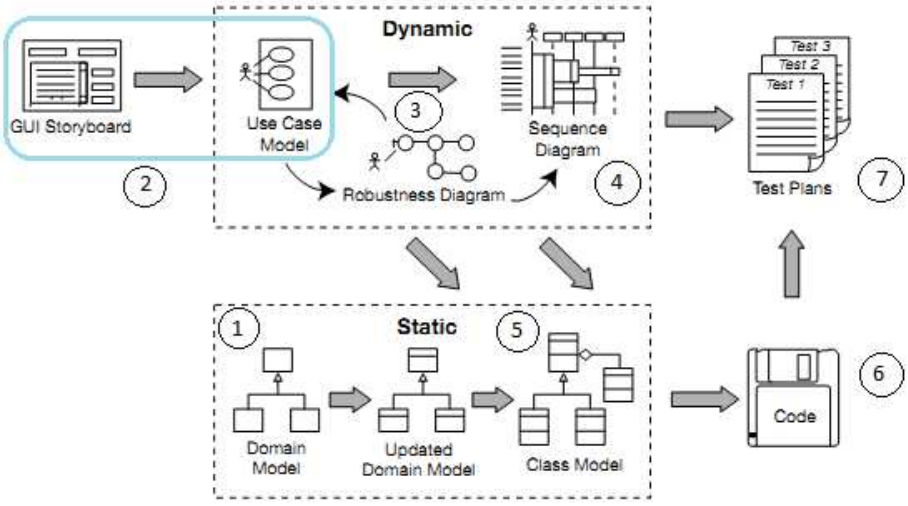

Figura 2-1 Enfoque aerodinámico de desarrollo – ICONIX

Fuente: ROSENBERG, Doug. Agile Development with ICONIX Process. Apress. 2005

[11] RUP → Glosario de términos 1.11

[12] XP → Glosario de términos 1.12

A continuación se describen los elementos que conforman el enfoque aerodinámico que utiliza ICONIX:

1 **Modelo del Dominio.-** Permite identificar los objetos (clases que representan cosas y conceptos) del dominio y la relación que tienen entre ellos. Con esto se puede tener un vocabulario común para las siguientes etapas del desarrollo.

2 **Prototipos de Interfaces y Casos de Uso.-** Definen el comportamiento de los requerimientos. Los prototipos de interfaces permiten ver la interacción del usuario con el sistema para identificar las posibles acciones que puede realizar, además sirve de base para la descripción detalla de los requerimientos. Los casos de uso se usan para capturar los requisitos del usuario detallando todos los guiones que realizarán los usuarios.

3 **Diagrama de Robustez.-** Este diagrama es similar a un diagrama de colaboración UML, es este se muestra los objetos que participan en el escenario y como esos objetos actúan recíprocamente entre sí. El análisis de robustez refina y verifica los casos de uso. Así también descubre objetos que se necesitan y que no se tomaron en cuenta en el modelo del dominio.

4 **Diagrama de Secuencia.-** Representa la interacción entre el usuario y el sistema mediante mensajes, basándose en UML, lo cual permite asignar comportamiento a los objetos.

5 **Modelo de Clases.-** Permite ver las relaciones de las clases del modelo del dominio con las clases del sistema. Con este diagrama se puede agregar información detallada y verificar si todos los requerimientos están satisfechos.

6 **Escribir y Generar Código.-** Se realiza el proceso de codificación de las respectivas clases y del proyecto en sí.

7 **Pruebas.-** Se describe un conjunto de pruebas utilizando como fuente los casos de uso para las pruebas de funcionalidad y se acuerda con el cliente la aceptación del sistema desarrollado.

En relación a la metodología, ICONIX se subdivide en cuatro fases que se pueden observar en la Figura 2-2, las mismas que contienen subprocesos que representan la secuencia de desarrollo de software bajo este concepto (Rosenberg & Matt, 2007).

Las fases son las siguientes: Análisis de Requerimientos, Análisis y Diseño Preliminar, Diseño Detallado e Implementación que serán brevemente descritas a continuación:

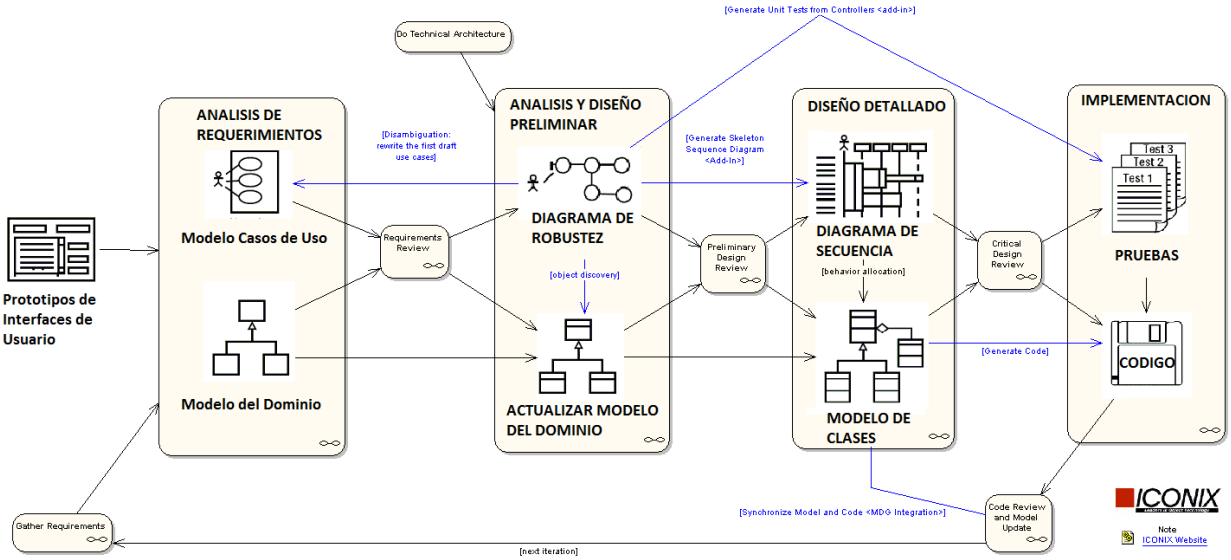

Figura 2-2 Fases de Desarrollo – ICONIX

Traducido por: El Autor.

Fuente: http://iconixprocess.com/iconix-process/roadmap

Análisis de Requerimientos.- Esta fase inicial tiene como objetivo principal, realizar la revisión de requerimientos del problema que serán representados a través de los diagramas de casos de uso y del modelo del dominio. La revisión de requerimientos permite asegurar que los casos de uso y el modelo del dominio trabajen conjuntamente y en la misma dirección de los requerimientos funcionales de los usuarios. Esto también permite que los usuarios tengan una idea clara de lo que el equipo se encuentra desarrollando en base al diseño de los requerimientos.

Para el presente proyecto se tomará como base de requerimientos la definición del alcance del prototipo que será descrito en el punto 2.2 de este trabajo.

Las actividades que le corresponden a esta fase son:

- Reunir información del sistema.
- Realizar prototipos UI del nuevo sistema propuesto.
- Identificar los objetos del dominio del mundo real. Artefacto: Modelo del Dominio.
- Identificar los Casos de Uso. Artefacto: Diagrama de Casos de Uso.
- Asignar los requerimientos funcionales a los casos de uso y objetos del dominio.
- Escribir el primer boceto de los Casos de Uso.

Análisis y Diseño Preliminar.- La fase de Análisis y Diseño Preliminar contiene los diagramas de Análisis de Robustez y la Actualización del Modelo de Dominio. Los objetivos de esta etapa son: proveer al desarrollador de un preliminar (diseño de exploración) para el cliente en el cual se pueda verificar que los resultados del análisis de requerimientos son correctos y llevar a cabo la revisión del Diseño Preliminar que involucra: la revisión del diagrama de robustez y la documentación de los casos de uso para cada escenario que se planea construir y asegurar que los diagramas realizados y el caso de uso refiriéndonos a la descripción de estos, se encuentran completos y representen correctamente el diseño del sistema y el comportamiento deseado.

Las actividades que le corresponden a esta fase son:

- Realizar el Análisis de Robustez. Artefacto: Diagrama de Robustez.
- Para cada caso de uso: quitar la ambigüedad del primer boceto textual de casos de uso, identificar un primer corte de objetos para lograr cada escenario y actualizar el modelo del dominio con el descubrimiento de nuevos objetos y atributos.
- Finalizar la actualización del nivel de análisis del diagrama de clases.

Diseño Detallado.- Esta fase tiene como objetivo principal, realizar la revisión del diseño detallado modelado a través de los Diagramas de Secuencia y .del diseño final del Modelo de Clases.
La revisión del diseño detallado implica validar que las fases antes detalladas cubran todas las necesidades del usuario y sean un preámbulo de calidad para facilitar la etapa de codificación, lo cual permita disminuir costos en el desarrollo del sistema.

Las actividades que le corresponden a esta fase son:

- Dividir el modelo del dominio en los Diagramas de Clase necesarios. Artefacto: Modelo de Clases.
- Por cada Caso de Uso:
    - Asignar comportamiento a través de los Diagramas de Secuencia. Artefacto: Diagrama de Secuencia.
- Actualizar los Diagramas de Clase con los nuevos atributos y operaciones encontradas.
- Finalizar con el modelo estático.
- Verificar el cumplimiento de los requerimientos.

Implementación y Pruebas.- Esta fase comprende los procesos respectivos a la Codificación y Pruebas, entre las actividades que contiene esta fase se encuentran las siguientes:

- Producir los Diagramas de Componentes y Despliegue necesarios.
- Escribir el código.
- Pruebas de Unidad e Integración.
- Pruebas de Sistema y Aceptación.

Para las Pruebas de Sistema a las que serán expuestos los prototipos del presente proyecto, se utilizarán los criterios de evaluación descritos en el punto 1.1.3 de este trabajo.

Para el desarrollo de los prototipos del presente proyecto de todas las actividades que define la metodología ICONIX para el desarrollo ágil de aplicaciones completas, se tomarán en cuenta los siguientes puntos:

Fase	Actividad
Análisis de Requerimientos	Requerimientos del Sistema. Prototipos de Interfaces de Usuario Modelo del Dominio Modelo de Casos de Uso
Análisis y Diseño Preliminar	Diagrama de Robustez
Diseño Detallado	Modelo de Clases Diagrama de Secuencia
Implementación y Pruebas	Codificación Diagramas de Componentes y Despliegue Pruebas de Sistema

Tabla 18 Cuadro de actividades para las fases de ICONIX.

Elaborado por: El Autor.

Para los diagramas, modelos y prototipos UI que se van a realizar como parte del proceso del desarrollo de los prototipos base, se utilizarán las herramientas StarUml, Microsfoft Office Visio y Power Designer.

2.2 Definición del alcance del prototipo base

2.2.1 Identificación del producto

El caso de aplicación consiste en crear un sistema base de gestión de clientes orientado al manejo de información de clientes de un banco que participan como solicitantes de una tarjeta crédito, la misma que ofrecerá una solución mediante la automatización del proceso de ingreso de datos de solicitudes de aplicación a tarjetas de crédito, con el fin de facilitar el acceso de ejecutivos de negocio a los datos de los solicitantes.

La información que se tomará en cuenta en el caso de aplicación para el ingreso de datos de solicitudes de aplicación a tarjetas de crédito, tomará como base una solicitud para tarjeta - habientes del Banco Pichincha que se puede ver en el Anexo 8.

Para la selección del caso de aplicación a utilizarse en los prototipos, se consideró la cantidad de conceptos persistentes y transaccionales que se pueden utilizar en un sistema base medianamente transaccional; en lugar de realizar una aplicación con mayor riqueza informativa o gráfica que utilice los mismos servicios pero que implique mayor complejidad en su desarrollo.

2.2.2 Responsabilidades y Exclusiones del Sistema

Responsabilidades

- Agilizar el proceso de ingreso de información requerida en la solicitud de aplicación a una tarjeta de crédito.
- Proveer procesos de consulta, ingreso, actualización y eliminación de la información de clientes solicitantes de tarjetas de crédito.
- Proporcionar un sistema base, que permita visualizar la información referente a los clientes solicitantes de una tarjeta de crédito.

Exclusiones

- El sistema no se encargará de la actualización de información en la plantilla de solicitud para tarjeta – habientes.
- El sistema no se encargará del registro de usuarios autorizados y definición de roles de usuarios.
- El sistema no actuará como intermediario para la aceptación o rechazo de solicitudes de tarjetas de crédito.

2.3 Prototipo con base a Frameworks

2.3.1 Elaboración de Diagramas de la Metodología

2.3.1.1 Análisis de Requerimientos

2.3.1.1.1 Prototipos de Interfaces de Usuario

Las interfaces principales del Sistema de Gestión de Solicitudes de Aplicación de Tarjetas de Crédito son las siguientes:

Interfaz – Página Principal: Interfaz principal del sistema, permite visualizar las opciones de menú que tiene disponible el usuario.

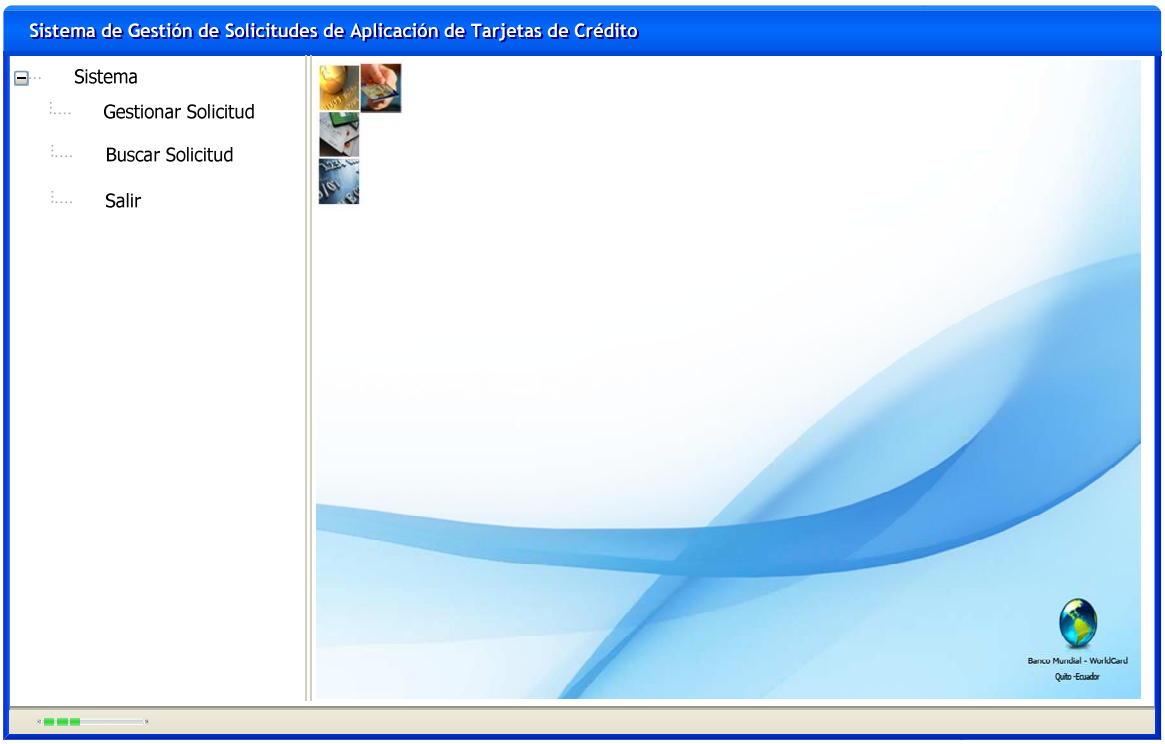

Figura 2-3 Interfaz – Página Principal.

Elaborador por: El Autor.

Interfaz – Gestionar Solicitud (Ingresar): Interfaz que permite ingresar los datos de una solicitud de aplicación de tarjeta de crédito.

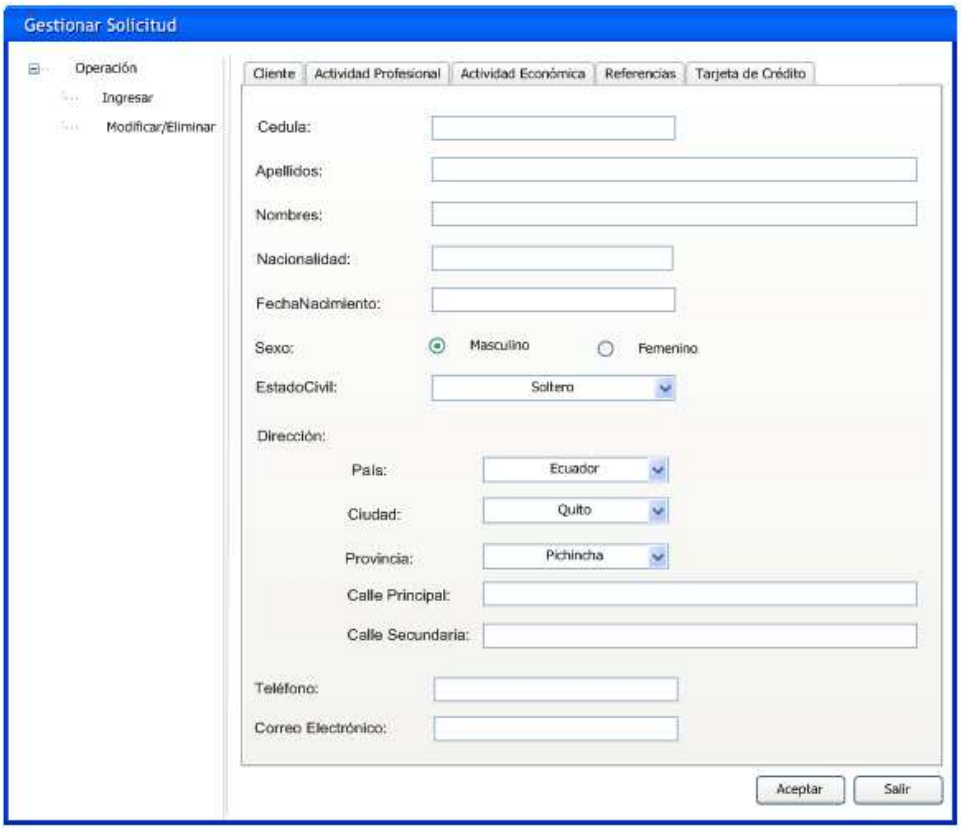

Figura 2-4 Interfaz – Gestionar Solicitud (Ingresar)

Elaborador por: El Autor.

Interfaz – Gestionar Solicitud (Modificar / Eliminar): Interfaz que permite modificar o eliminar los datos de una solicitud de aplicación de tarjeta de crédito previamente ingresada.

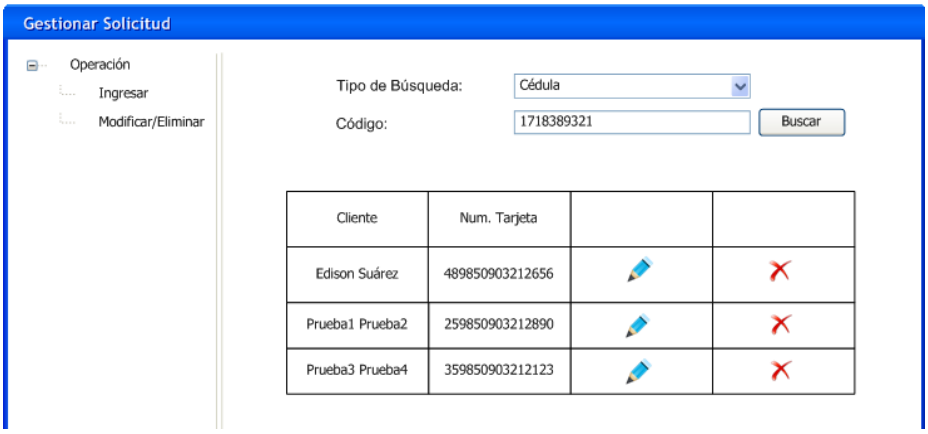

Figura 2-5 Interfaz – Gestionar Solicitud (Modificar o Eliminar). Elaborador por: **El Autor.**

Interfaz – Buscar Solicitud: Interfaz que permite buscar una solicitud de aplicación de tarjeta de crédito mediante la cédula de identidad de un cliente solicitante previamente registrado.

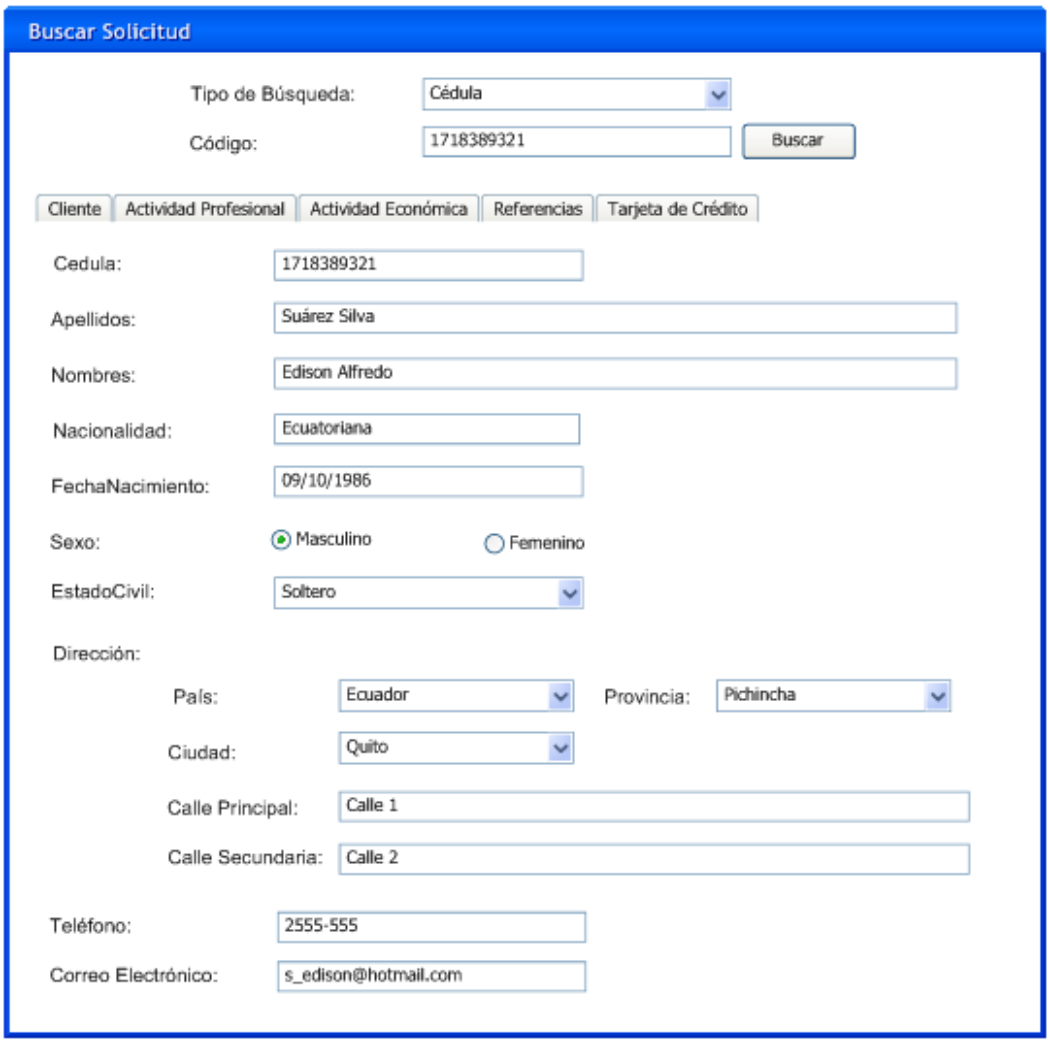

Figura 2-6 Interfaz – Buscar Solicitud.

Elaborador por: El Autor.

2.3.1.1.2 Modelo del Dominio

Como parte de la fase de Análisis de Requerimientos se tiene el primer diagrama que corresponde al modelo del dominio, el mismo que se basa en la definición del alcance del sistema y en los prototipos de las interfaces.

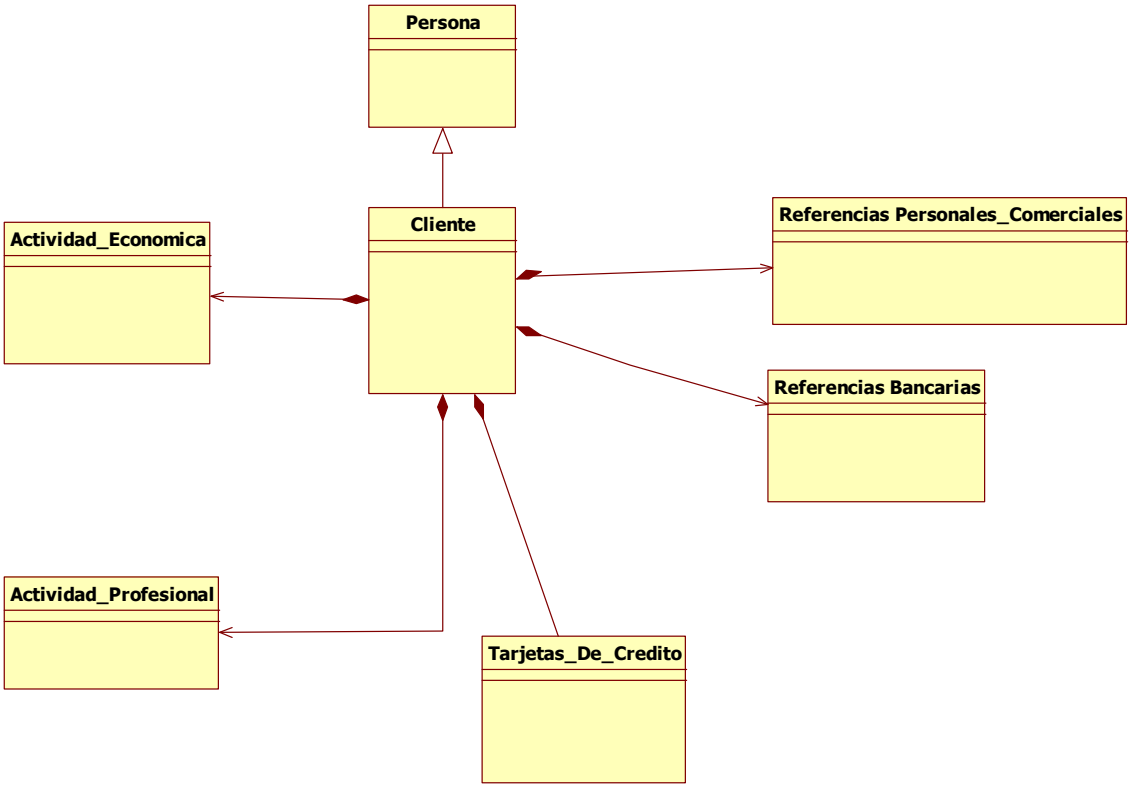

Figura 2-7 Modelo del Dominio.

Elaborado por: El Autor.

2.3.1.1.3 Modelo de Caso de Uso

A continuación se detalla el caso de uso correspondiente al actor Ejecutivo de Negocios, encargado de la gestión clientes solicitantes de una tarjeta de crédito.

Descripción de Actores

Actor	Descripción
1 Ejecutivo de Negocios	Es la persona autorizada de gestionar la solicitud de la tarjeta de crédito, se encarga de recibir e ingresar la información.

Figura 2-8 Descripción de Actores. Elaborado por: **El Autor.**

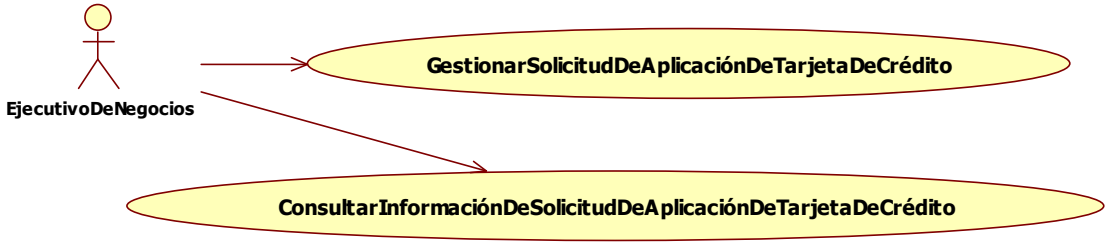

Figura 2-9 Modelo de Caso de Uso**.**

Elaborado por: El Autor.

Descripción de Casos de Uso

Caso de Uso: **Gestionar Solicitud de Aplicación de Tarjeta de Crédito**	*ID:* CU1
Actor: Ejecutivo de Negocios	
Objetivo: Permitirá al ejecutivo de negocios registrar, modificar o eliminar la solicitud de aplicación a una tarjeta de crédito.	
Pre Condiciones: • El Ejecutivo de Negocios debe haber iniciado sesión en el sistema.	
Post Condiciones: • El cliente queda registrado. • La solicitud de aplicación a una tarjeta de crédito queda registrada.	
Escenarios: **Flujo Normal:** 1. Ingresar a la pantalla para Gestionar Solicitud de Aplicación de Tarjeta de Crédito. 2. Seleccionar la actividad a realizar dentro de la pantalla.	

2.1 Ingresar información del Cliente.

2.1.1 SubFlujo 1.

2.2 Modificar información del Cliente.

2.2.1 SubFlujo 2.

2.3 Eliminar información del Cliente.

2.3.1 SubFlujo 3.

3. Fin del Caso de Uso

SubFlujo 1: Ingresar información del Cliente.

1. El Actor ingresa los datos personales del cliente que solicita la tarjeta de crédito.

1.1 Si *camposCliente* = Empty

1.1.1 Ir a Excepción: *ValidaciónCampos.*

2. El actor ingresa la información correspondiente a la actividad profesional del cliente.

2.1 Si *camposActividadProfesional* = Empty

2.1.1 Ir a Excepción: *ValidaciónCampos.*

3. El actor ingresa la información correspondiente a la actividad económica del cliente.

3.1 Si *camposActividadEconómica* = Empty

3.1.1 Ir a Excepción: *ValidaciónCampos*

4. El actor ingresa la información correspondiente a las referencias personales_comerciales que tiene el cliente.

4.1 Si *camposReferenciasPersonales_Comerciales* = Empty

4.1.1 Ir a Excepción: *ValidaciónCampos*

5. El actor ingresa la información correspondiente a las referencias bancarias que tiene el cliente.

5.1 Si *camposReferenciasBancarias* = Empty

5.1.1 Ir a Excepción: *ValidaciónCampos*

6. El actor ingresa la información de la tarjeta de crédito a la que está aplicando el cliente.

6.1 Si *camposTarjetaDeCredito* = Empty

6.1.1 Ir a Excepción: *ValidaciónCampos.*

7. El sistema despliega un resumen del ingreso de datos de la solicitud de aplicación a una tarjeta de crédito.

8. El sistema informa que la solicitud de aplicación a una tarjeta de crédito ha sido registrada.

9. Fin del SubFlujo 1.

SubFlujo 2: Modificar información del Cliente.

1. Buscar registro del cliente.
2. Modificar información del cliente.
3. Fin del Subflujo 2.

SubFlujo 3: Eliminar información del Cliente.

1. Buscar registro del cliente.
2. Eliminar registro.
3. El sistema despliega confirmación de eliminación del registro.

3.1Si *confirmación* = Aceptar

3.1.1 El registro del cliente seleccionado es eliminado del sistema.

4. Fin del SubFlujo 3.

Excepción: *ValidaciónCampos*

1.1. El sistema despliega un error de campos vacíos o incorrectos.

1.2. Regresa al paso de invocación.

Caso de Uso: **Consultar información de Solicitud de Aplicación de Tarjeta de Crédito**	*ID:* 2
Actor: Ejecutivo de Negocios	
Objetivo: Permitirá al ejecutivo de negocios consultar información de solicitudes de aplicación de tarjetas de crédito.	
Pre Condiciones: • El Ejecutivo de Negocios debe haber iniciado sesión en el sistema. • La solicitud de aplicación de tarjeta de crédito debe estar registrada en el sistema.	
Post Condiciones: • El sistema despliega en pantalla la información de la solicitud de aplicación de tarjeta de crédito asociada al cliente seleccionado.	
Escenarios: **Flujo Normal:** 1. El actor selecciona el tipo de búsqueda (cédula de identidad, número de tarjeta de crédito) del cliente que se desea consultar los datos.	

2. El sistema despliega la información de la solicitud de aplicación de tarjeta de crédito asociada al cliente seleccionado.

3. Termina caso de uso.

Excepción:

1.1 El sistema no despliega información de la solicitud de aplicación de tarjeta de crédito debido a que aún no se ha registrado una solicitud para el cliente con la cédula ingresada.

2.3.1.2 Análisis y Diseño Preliminar

2.3.1.2.1 Diagrama de Robustez

Al actor Ejecutivo de Negocios le corresponde el diagrama de robustez descrito a continuación.

Diagrama de Robustez – Gestionar Solicitud de Tarjeta de Crédito

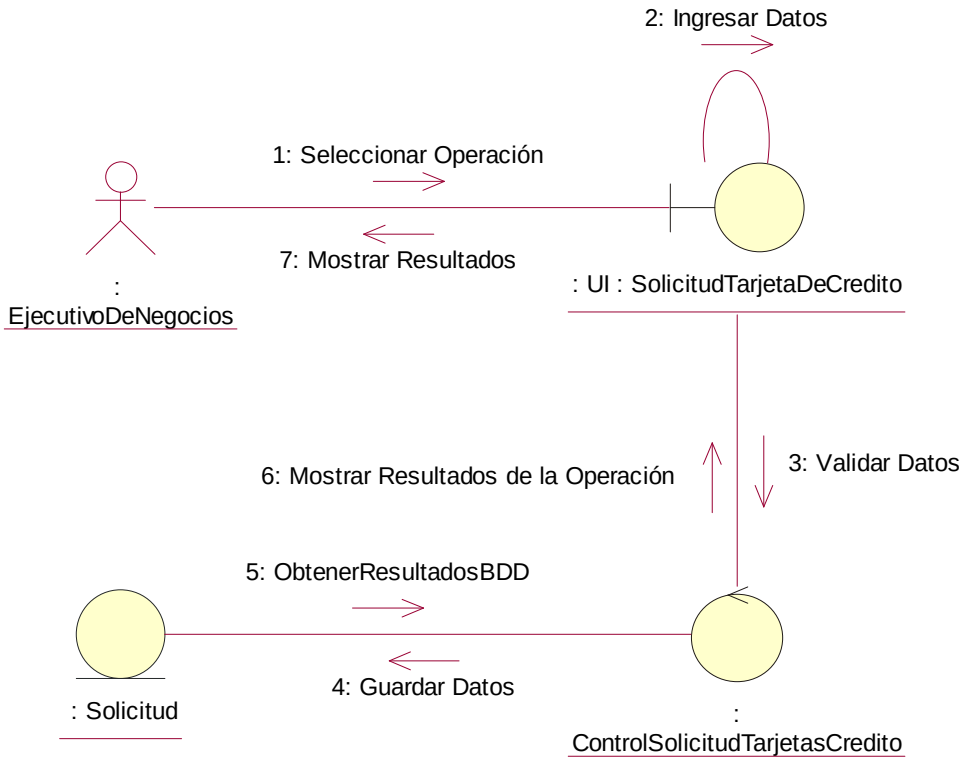

Figura 2-10 Diagrama de Robustez – Gestionar Solicitud de Tarjeta de Crédito

Elaborado por: El Autor.

Descripción del proceso:

1. Una vez ingresado el Ejecutivo de Negocios al sistema, se selecciona la operación que se desea realizar, ya sea buscar, ingresar, eliminar o actualizar.
2. Se procede a ingresar los datos respectivos a la solicitud de tarjeta de crédito.
3. Se valida la información que se ingresó, en caso de que la información validada sea la correcta se sigue con el paso 4, caso contrario se presenta un mensaje al usuario de la verificación de los datos ingresados.
4. Se realiza la operación seleccionada por el usuario ya sea buscar, ingresar, actualizar o eliminar en la base de datos.
5. Se obtiene el respectivo resultado de la Base de Datos.
6. Se muestra el resultado de la operación a la Interfaz correspondiente a la Gestión de Solicitud de Tarjeta de Crédito.
7. Se muestra el resultado de la operación solicitada por el Ejecutivo de Negocios.

2.3.1.3 Diseño Detallado

2.3.1.3.1 Diagrama de Clases

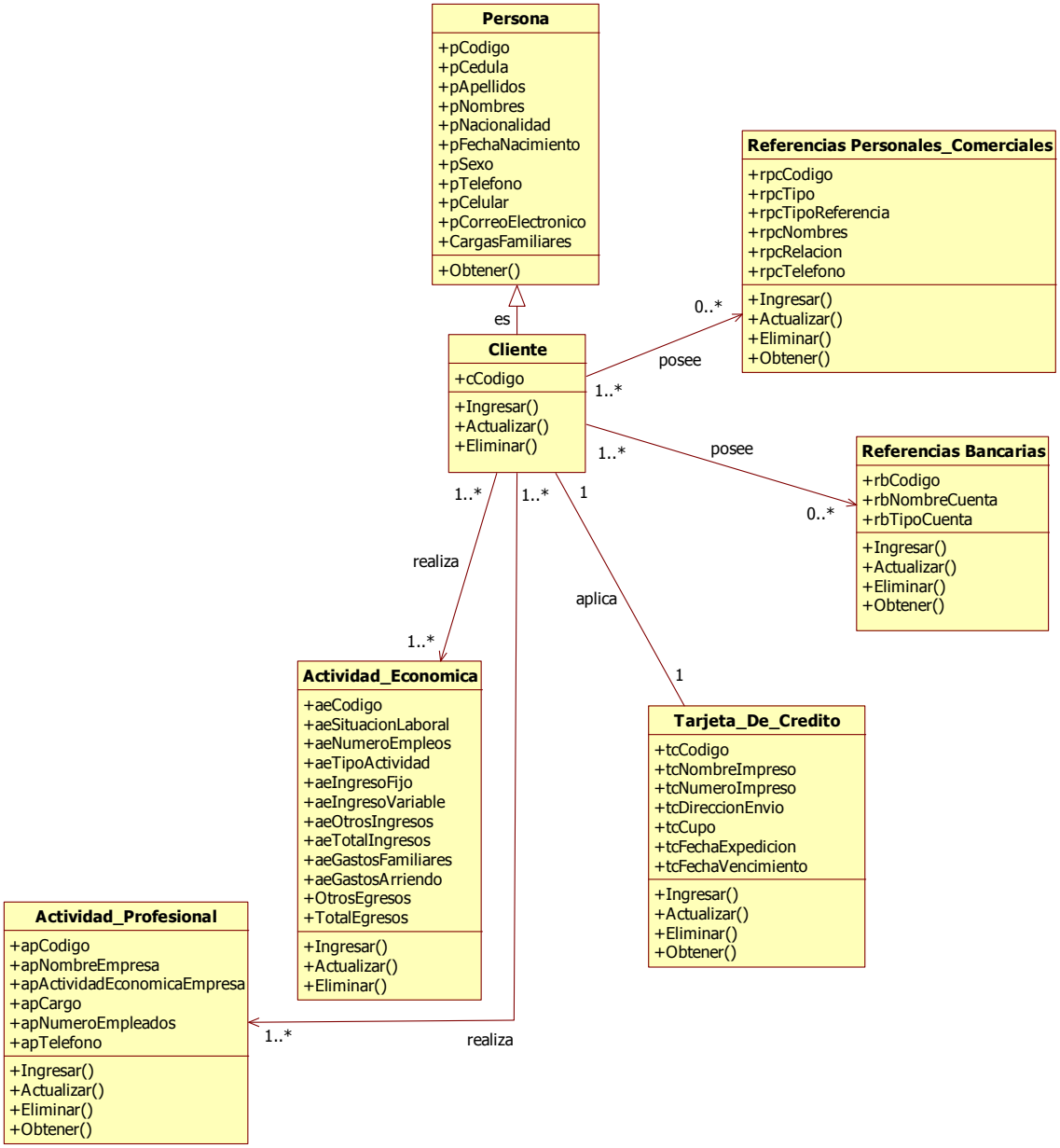

Figura 2-11 Modelo de Clases.

Elaborado por: El Autor.

2.3.1.3.2 Diagrama de Secuencia

Diagrama de Secuencia – Gestionar Solicitud de Tarjeta de Crédito

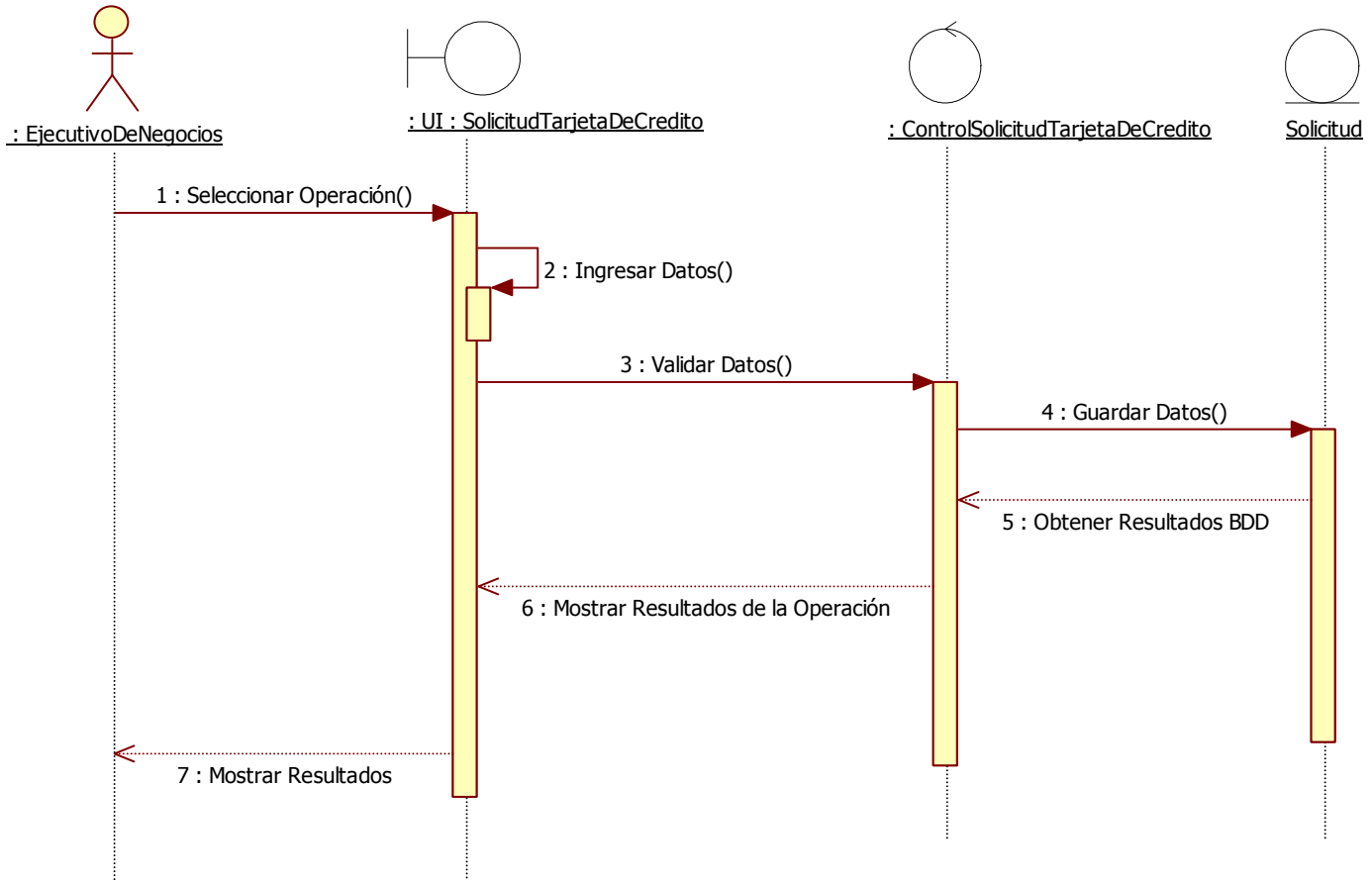

Figura 2-12 Diagrama de Secuencia – Gestionar Solicitud de Tarjeta de Crédito.

Elaborado por: El Autor.

2.3.1.4 Implementación y Pruebas

2.3.1.4.1 Escribir código

2.3.1.4.1.1 Desarrollo de Prototipo usando EJB 3

2.3.1.4.1.1.1 Gestionar Solicitud de Aplicación de Tarjeta de Crédito

Figura 2-13 Pantalla principal – Sistema de Gestión de Solicitud de Tarjeta de Crédito.

Elaborado por: El Autor.

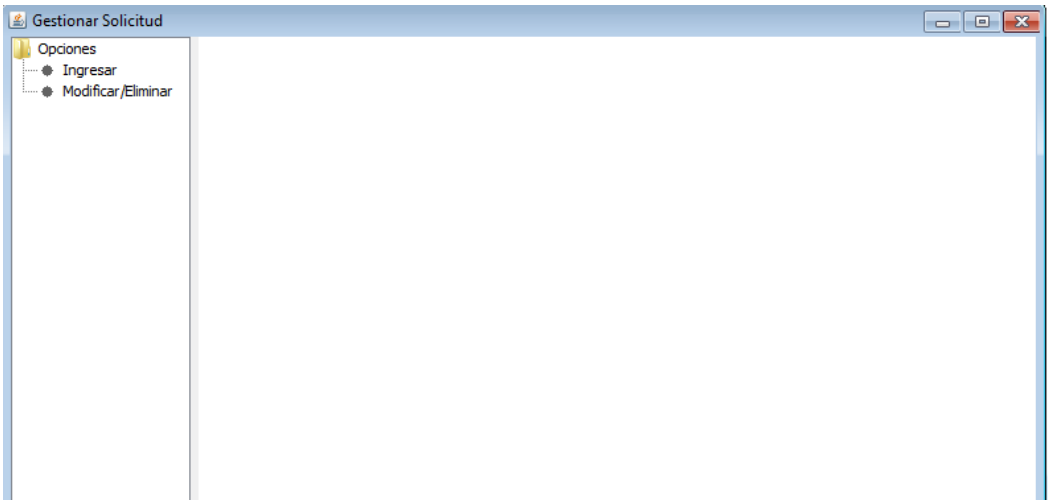

Figura 2-14 Pantalla principal – Sistema de Gestión de Solicitud de Tarjeta de Crédito.

Elaborado por: El Autor.

Ingresar Solicitud

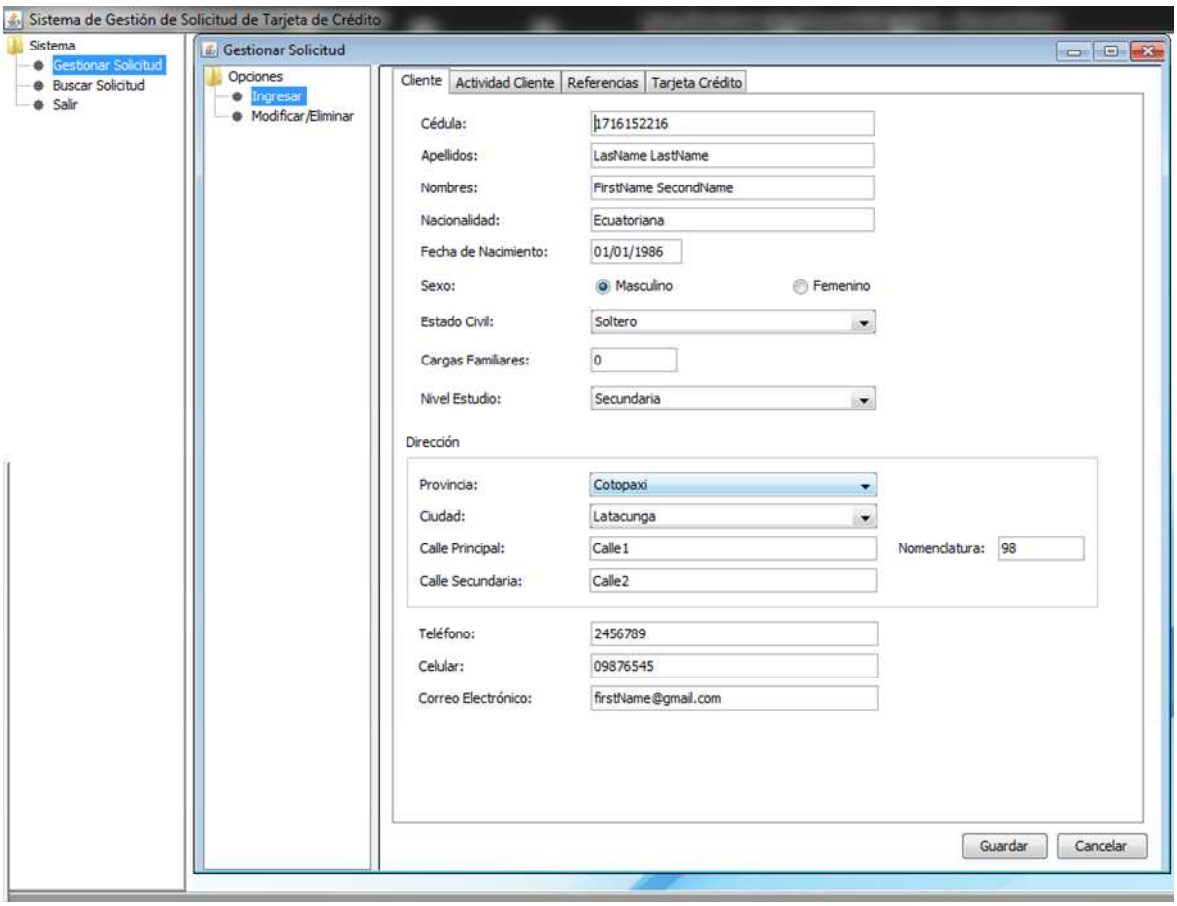

Figura 2-15 Pantalla de Ingreso de Solicitud.

Elaborado por: El Autor.

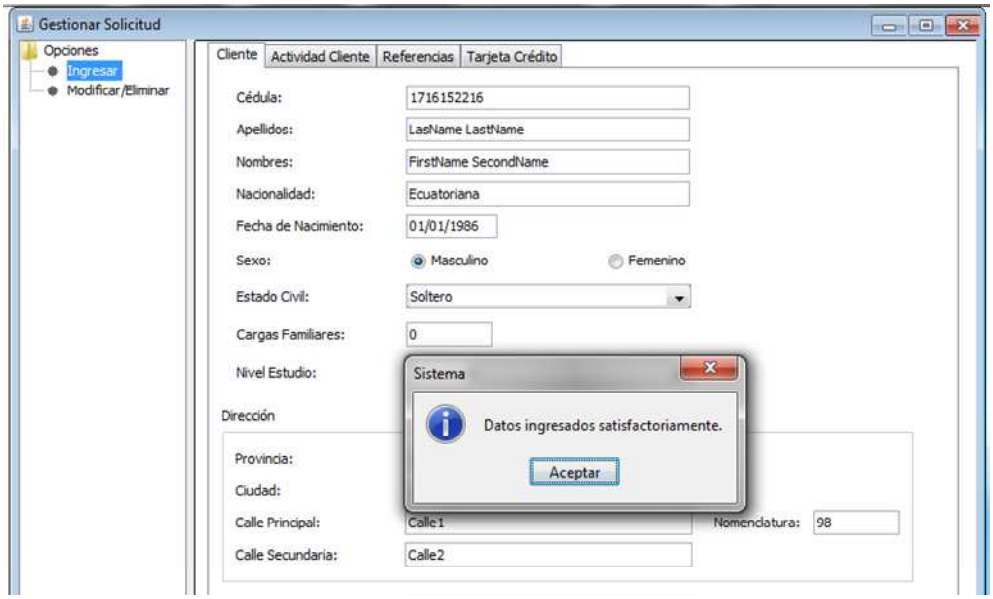

Figura 2-16 Pantalla de Solicitud ingresada con éxito. **Elaborado por:** El Autor.

Editar Solicitud

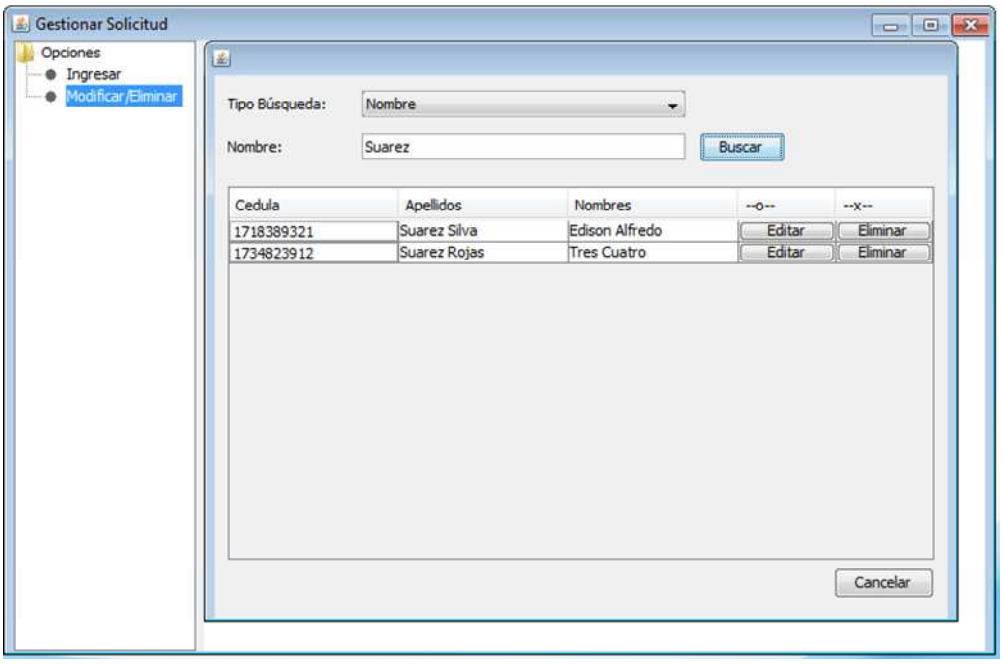

Figura 2-17 Pantalla de Edición y Eliminación de Solicitudes.

Elaborado por: El Autor.

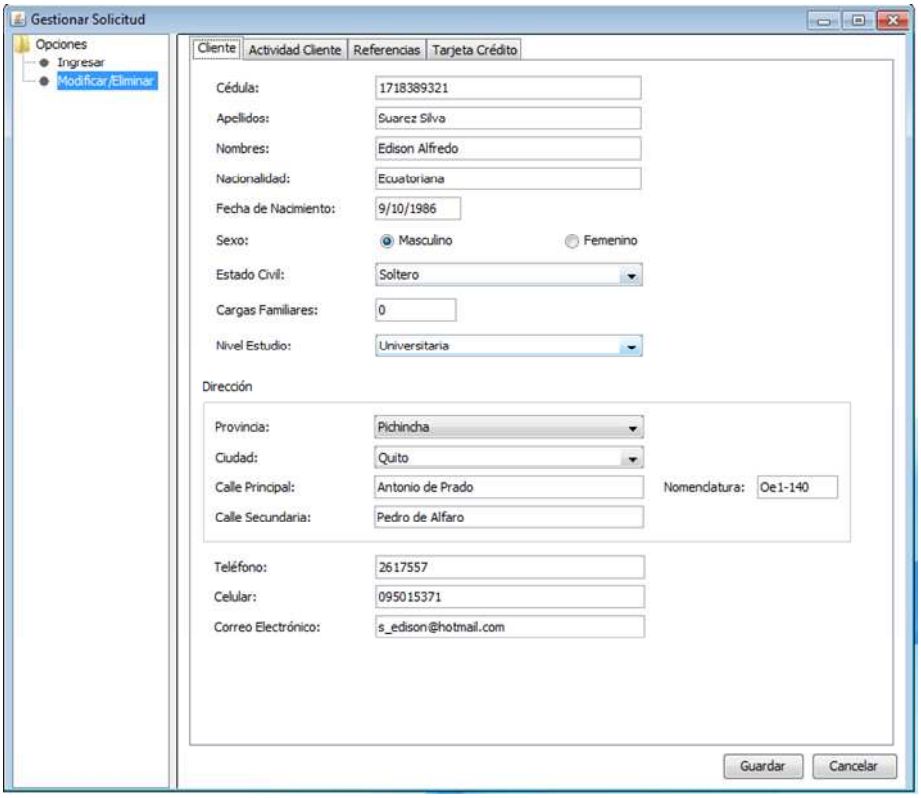

Figura 2-18 Pantalla de Actualización de datos de Solicitud.

Elaborado por: El Autor.

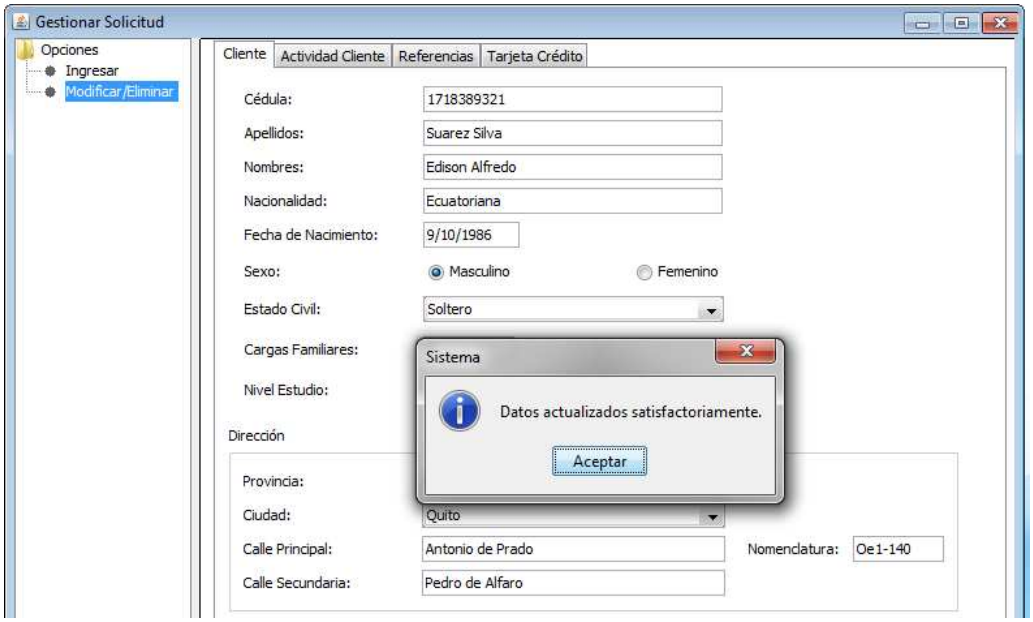

Figura 2-19 Pantalla de Solicitud actualizada con éxito.

Elaborado por: El Autor.

Eliminar Solicitud

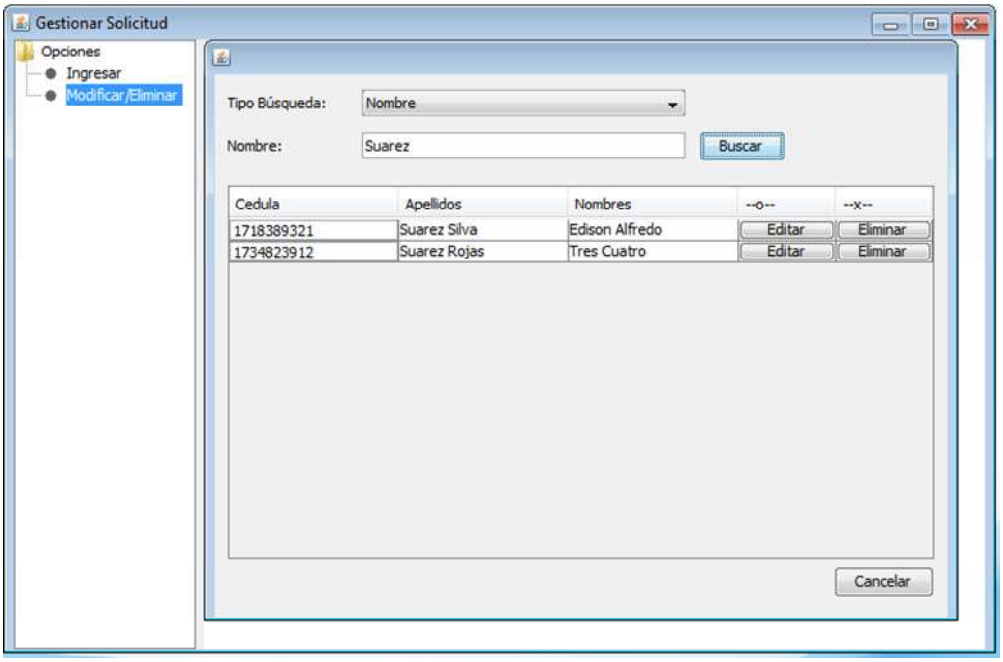

Figura 2-20 Pantalla de Edición y Eliminación de Solicitudes.

Elaborado por: El Autor.

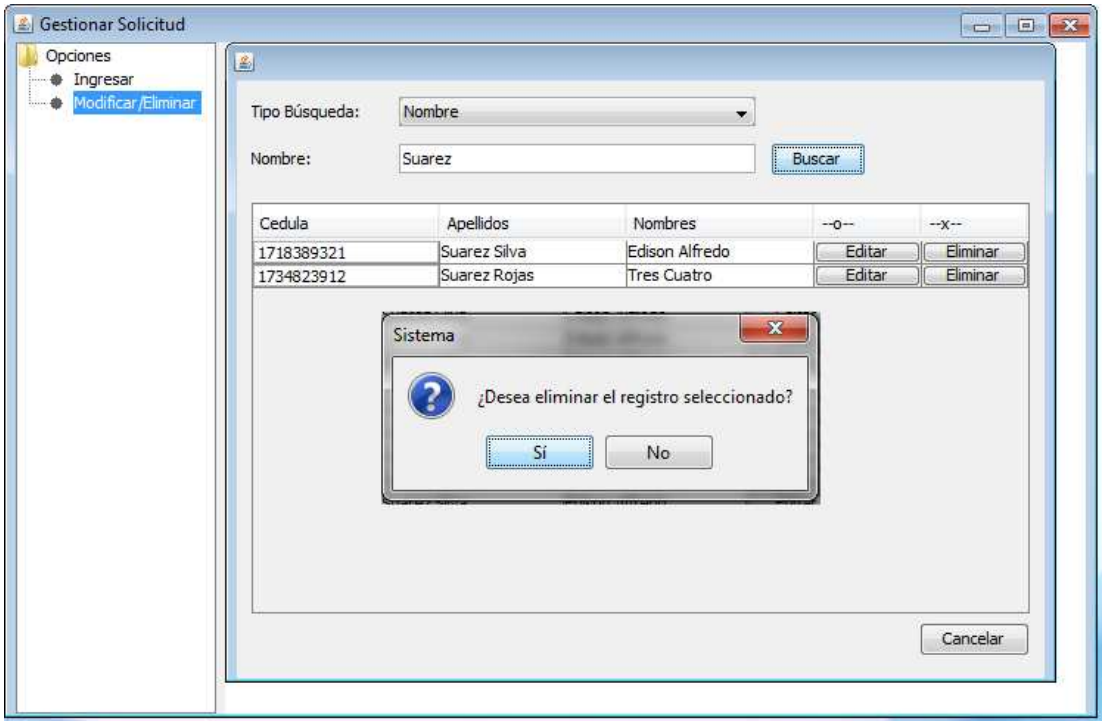

Figura 2-21 Pantalla – Desea Eliminar el registro seleccionado.

Elaborado por: El Autor.

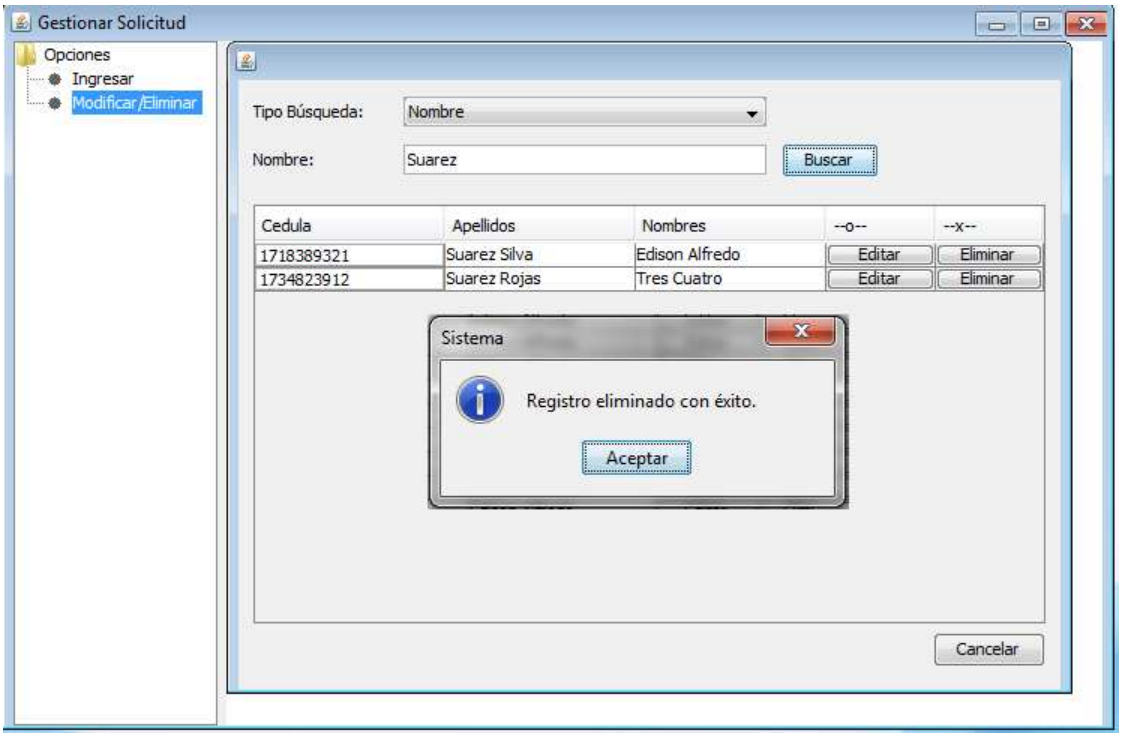

Figura 2-22 Pantalla Solicitud eliminada con éxito.

Elaborado por: El Autor.

2.3.1.4.1.1.2 Consultar información de Solicitud de Aplicación de Tarjeta de Crédito

Figura 2-23 Pantalla principal – Sistema de Gestión de Solicitud de Tarjeta de Crédito.

Elaborado por: El Autor.

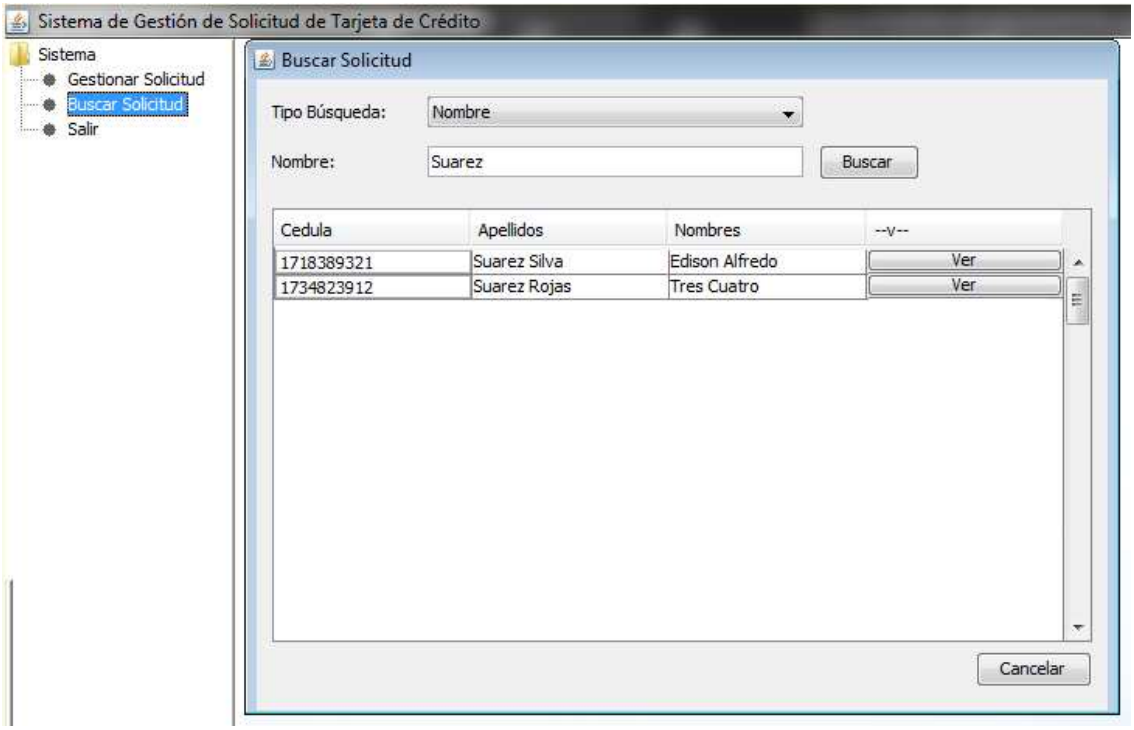

Figura 2-24 Pantalla de Búsqueda Solicitud.

Elaborado por: El Autor.

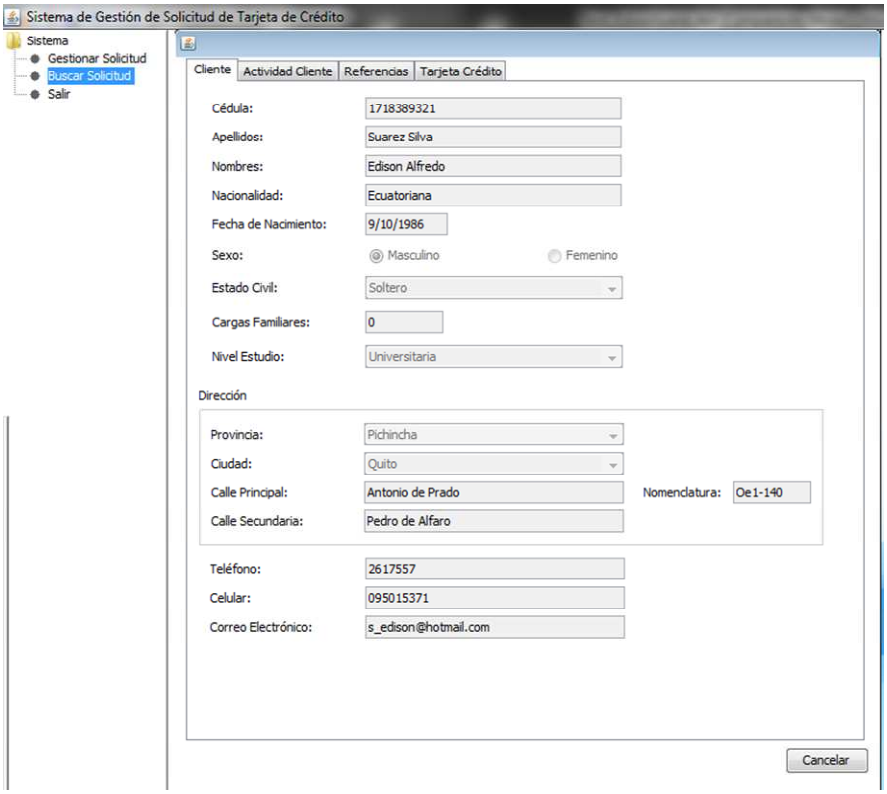

Figura 2-25 Pantalla de visualización de los datos de Solicitud.

Elaborado por: El Autor.

2.3.1.4.1.2 Desarrollo de Prototipo usando Entity Framework

2.3.1.4.1.2.1 Gestionar Solicitud de Aplicación de Tarjeta de Crédito

Figura 2-26 Pantalla principal – Sistema de Gestión de Solicitud de Tarjeta de Crédito.

Elaborado por: El Autor.

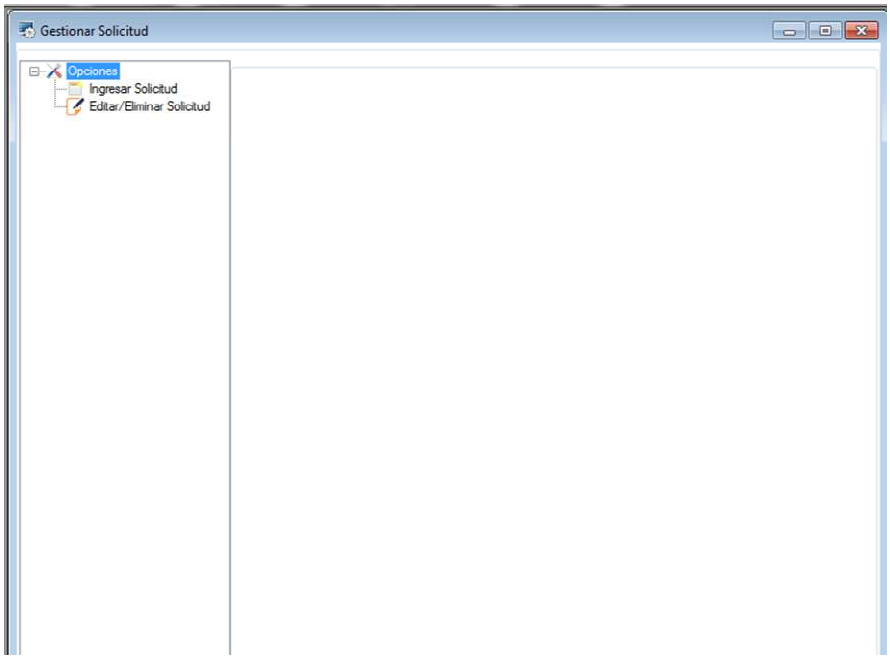

Figura 2-27 Pantalla de opciones para Gestionar Solicitud.

Elaborado por: El Autor.

Ingresar Solicitud

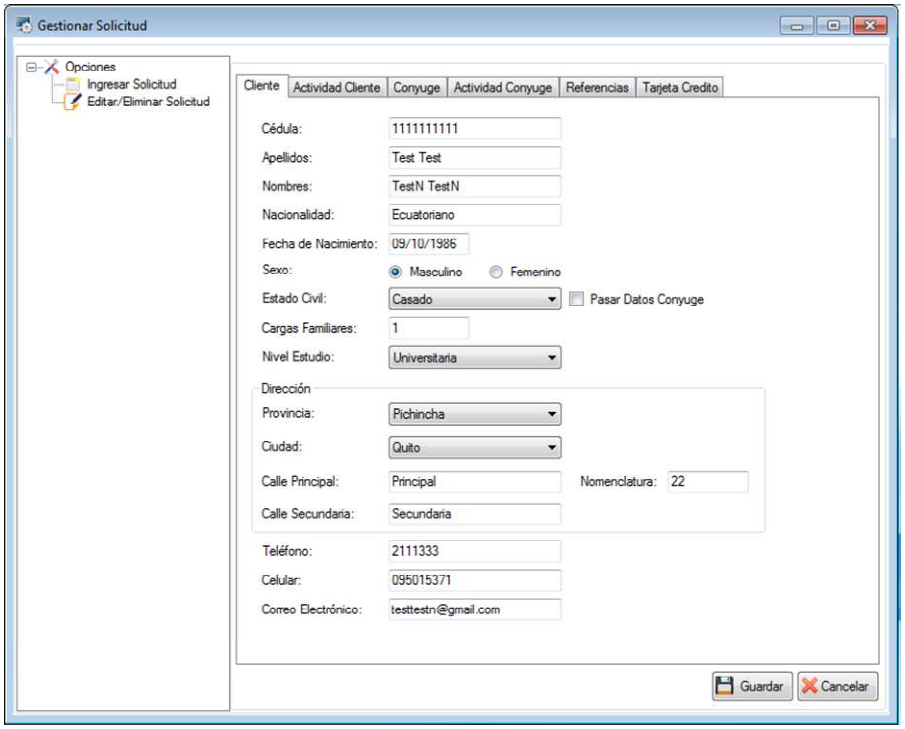

Figura 2-28 Pantalla de Ingreso de Solicitud.

Elaborado por: El Autor.

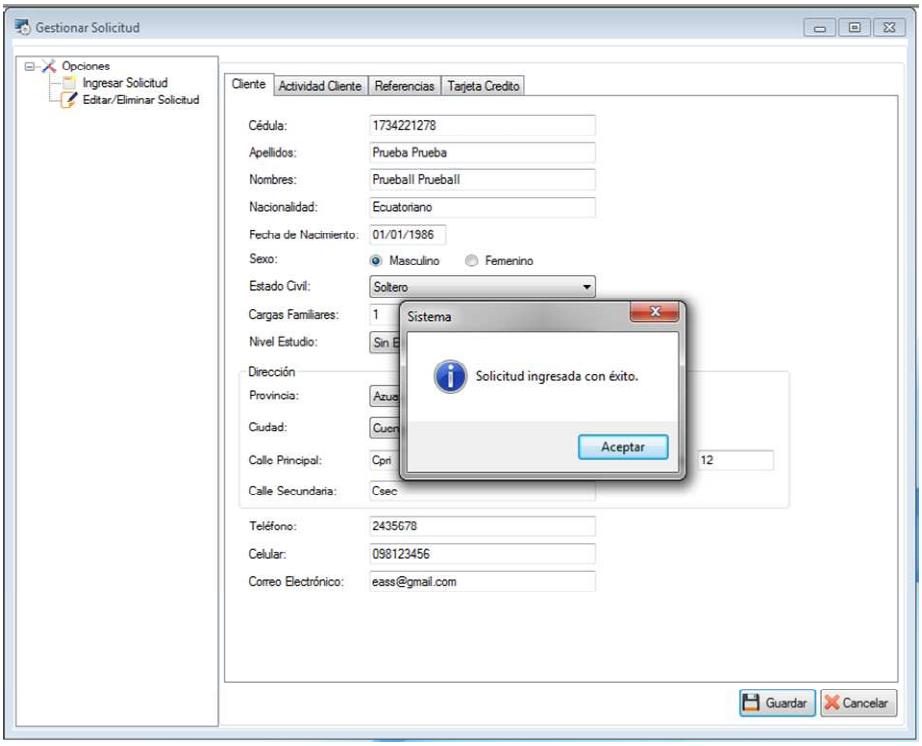

Figura 2-29 Pantalla de Solicitud ingresada con éxito.

Elaborado por: El Autor.

Editar Solicitud

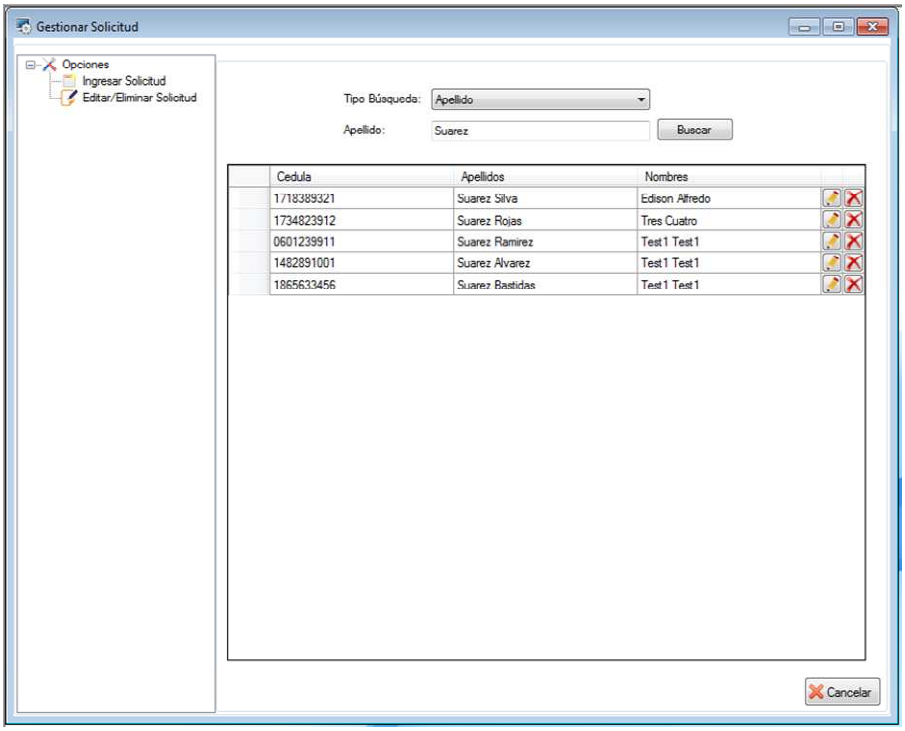

Figura 2-30 Pantalla de Edición y Eliminación de Solicitudes.

Elaborado por: El Autor.

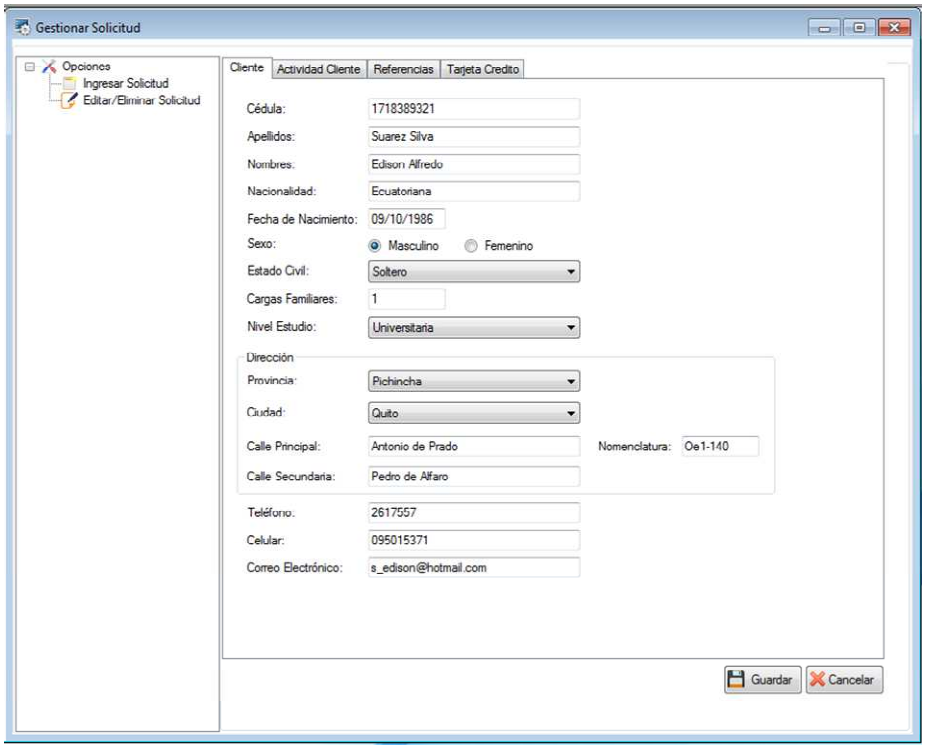

Figura 2-31 Pantalla de Actualización de datos de Solicitud.

Elaborado por: El Autor.

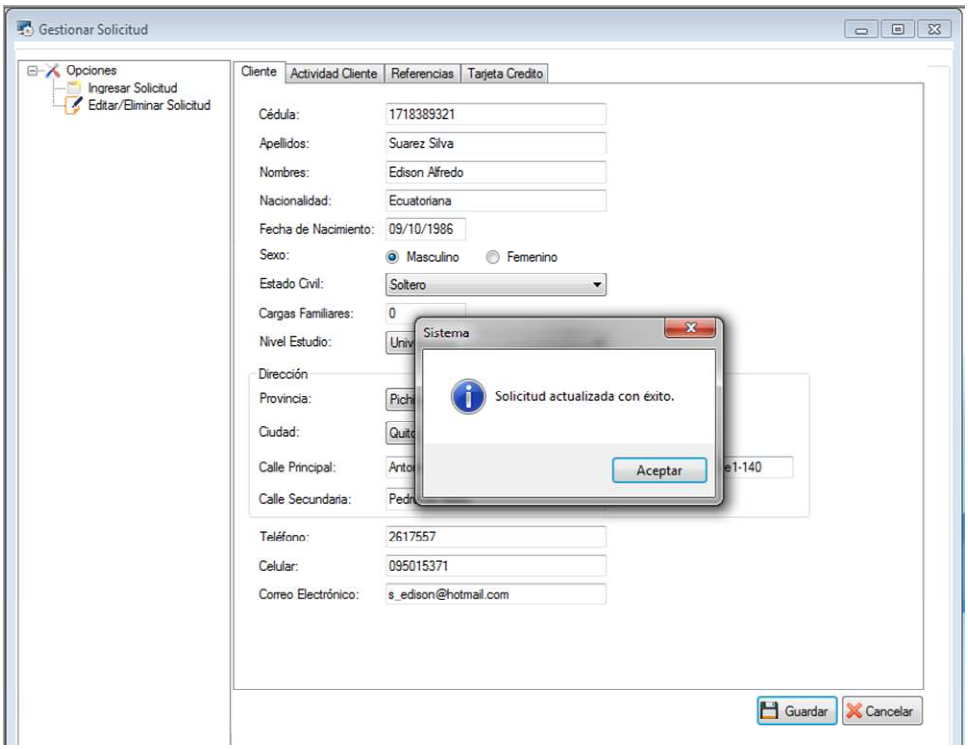

Figura 2-32 Pantalla de Solicitud actualizada con éxito.

Elaborado por: El Autor.

Eliminar Solicitud

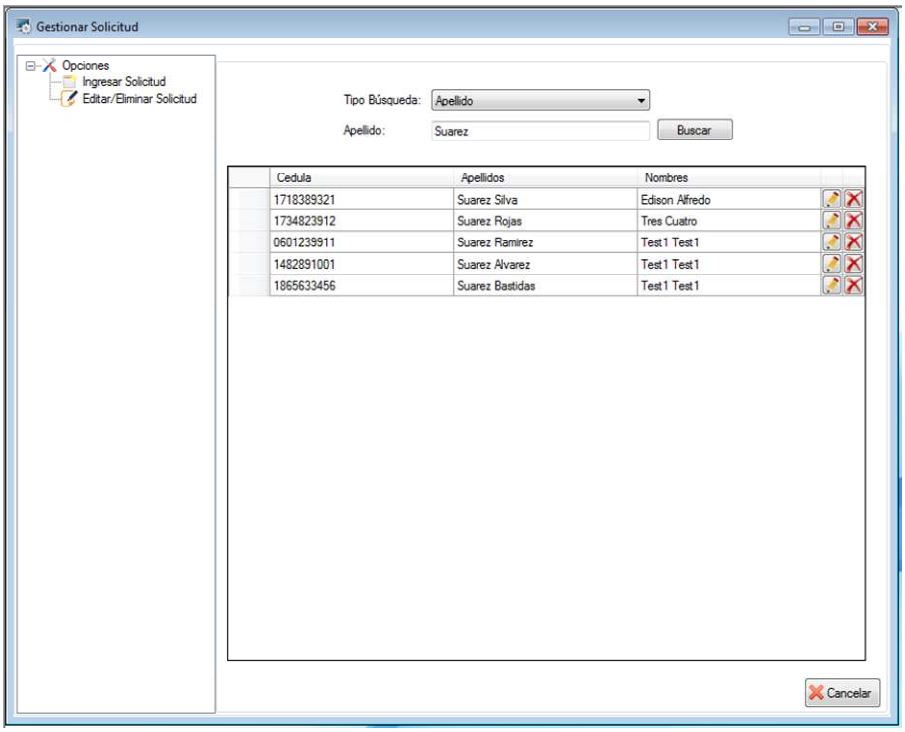

Figura 2-33 Pantalla de Edición y Eliminación de Solicitudes.

Elaborado por: El Autor.

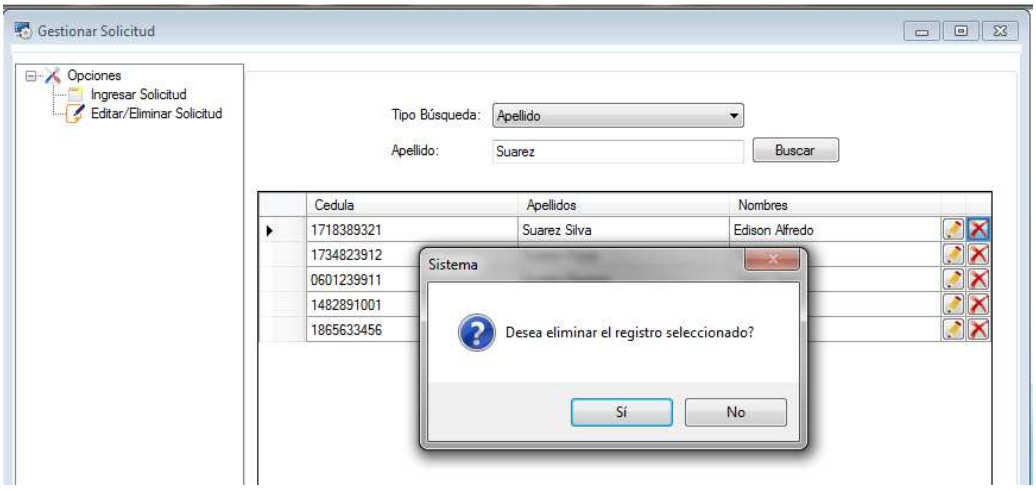

Figura 2-34 Pantalla – Desea Eliminar el registro seleccionado.

Elaborado por: El Autor.

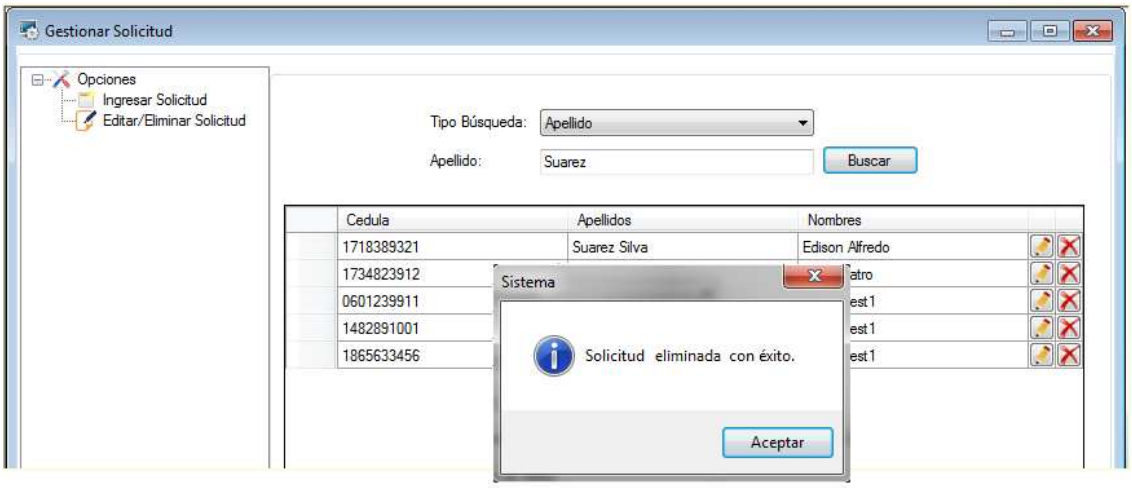

Figura 2-35 Pantalla Solicitud eliminada con éxito.

Elaborado por: El Autor.

2.3.1.4.1.2.2 Consultar información de Solicitud de Aplicación de Tarjeta de Crédito

Figura 2-36 Pantalla principal – Sistema de Gestión de Solicitud de Tarjeta de Crédito.

Elaborado por: El Autor.

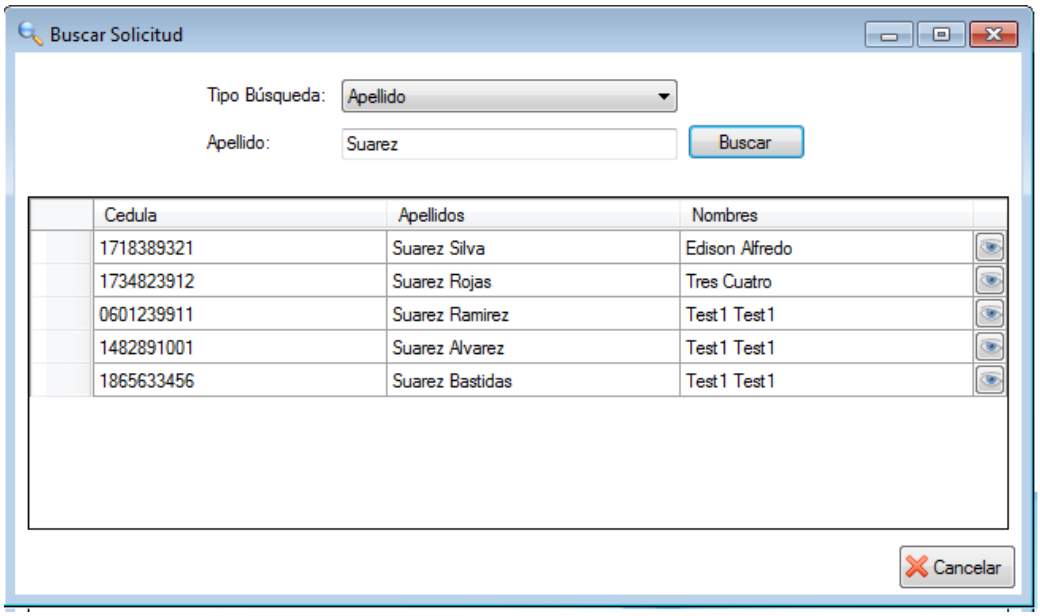

Figura 2-37 Pantalla de Búsqueda Solicitud.

Elaborado por: El Autor.

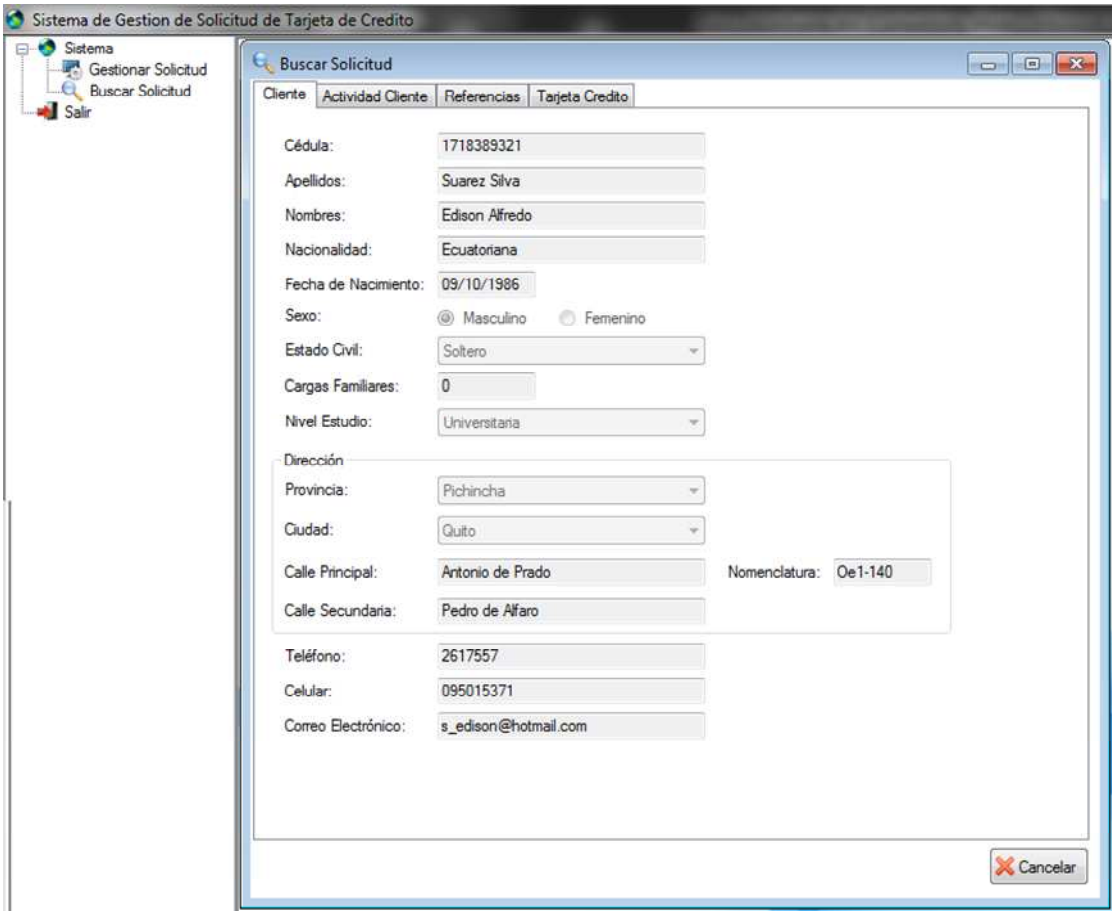

Figura 2-38 Pantalla de visualización de los datos de Solicitud.

Elaborado por: El Autor.

2.3.1.4.1.3 Pruebas de Sistema

Las pruebas realizadas se enfocaron en valorar los criterios de evaluación definidos en el Capítulo 1, obteniendo calificaciones definitivas para los criterios que no fueron valorados o cuya valoración fue preliminar.

2.3.1.4.1.4 Ambiente para el proceso de desarrollo

Durante el proceso de desarrollo de los prototipos se utilizaron los siguientes recursos:

Máquina de Desarrollo	
Sistema Operativo:	Microsoft Windows 7
Procesador:	Intel core i5 x64
Memoria RAM:	4GB
Ambiente de Desarrollo - EJB3/JPA	
IDE de Desarrollo:	NetBeans 6.9.1
Versión de JDK:	1.6 update 25
Servidor de Aplicaciones:	Glassfish 3.0.1
Servidor Web:	Apache Tomcat 6.0.26
Proveedor de Persistencia:	Eclipse Link – JPA2
Conector MySQL:	mysql-connector-java 5.1.6
Ambiente de Desarrollo – Entity Framework	
IDE de Desarrollo:	Visual Studio 2010
Conector MySQL:	mysql-connector-net 6.3.6
Servidor Web:	ASP .Net Developer Server
Máquina Servidor – Base de Datos	
Sistema Operativo:	Microsoft XP SP3
Procesador:	Intel core i5 x64
Memoria RAM:	1GB
Base de Datos:	MySQL Server 5.1

Tabla 19 Características de los recursos utilizados en el desarrollo de prototipos.

Elaborado por: El Autor.

El desarrollo de los prototipos se basó en un modelo multicapa dividiendo sus componentes de la siguiente forma:

1. *Arquitectura de desarrollo del prototipo utilizando EJB3*

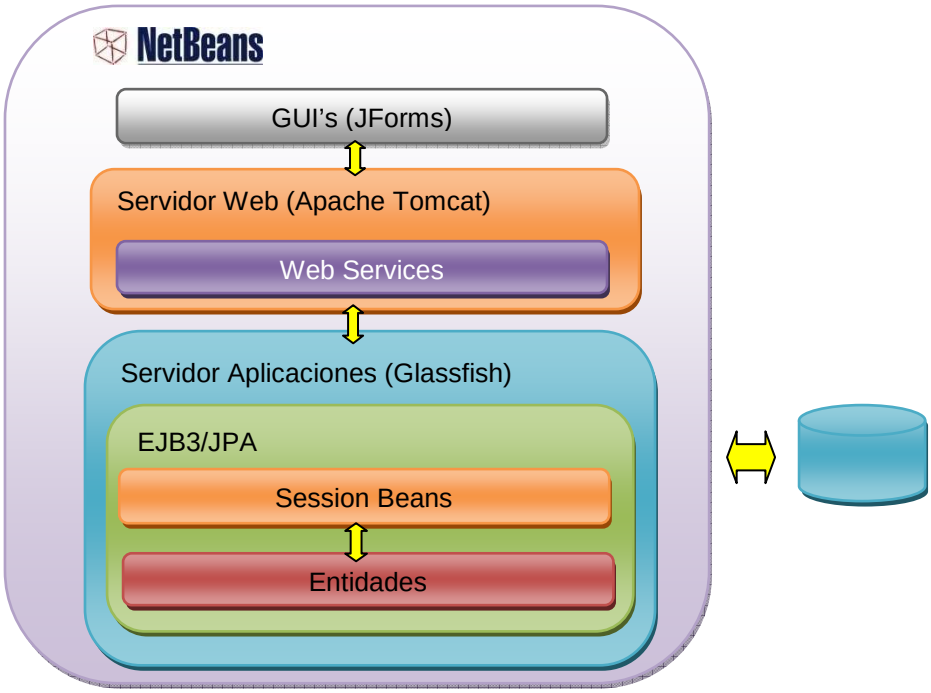

Figura 2-39 Arquitectura de desarrollo utilizando EJB3

Elaborado por: El Autor.

2. *Arquitectura de desarrollo del prototipo utilizando Entity Framework*

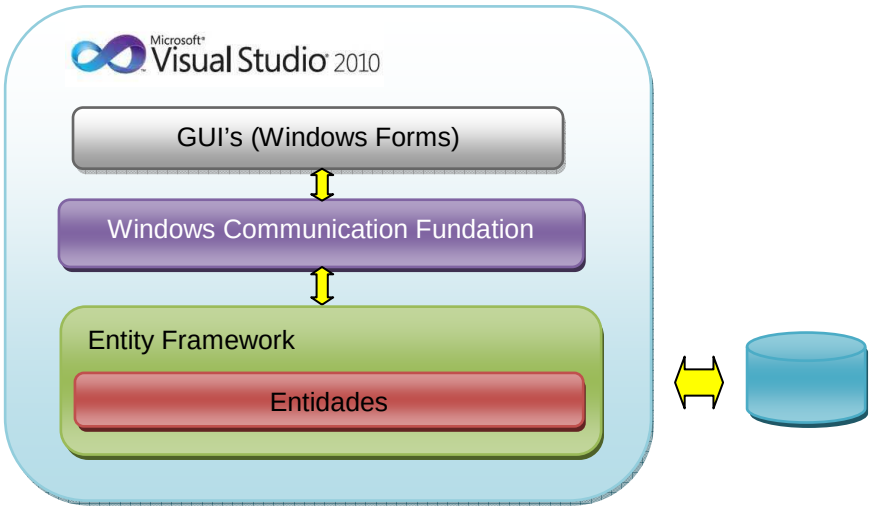

Figura 2-40 Arquitectura de desarrollo utilizando Entity Framework

Elaborado por: El Autor.

2.3.1.4.1.5 Ambiente para el proceso de evaluación

Con el fin de simular un ambiente de producción adecuado en el cual poner a prueba los prototipos frente a los criterios de evaluación referentes al rendimiento, se escogieron las herramientas que se describen a continuación.

JMeter

Se trata de una herramienta de prueba de carga desarrollada por Apache que se utiliza para analizar y medir el desempeño de una variedad de servicios. Entre sus características encontramos:

- No tiene costo de licencia.
- Capacidad de simular un número elevado de clientes conectándose simultáneamente o a intervalos a la aplicación.
- Ofrece varios tipos de resúmenes de las pruebas de carga ejecutadas (tabla de resumen, graficas de rendimiento, logs de ejecución, etc).
- Permite el uso de archivos “csv” para facilitar el uso de metadata en la ejecución de pruebas de carga.
- Ofrece una variedad de parámetros para realizar pruebas de rendimiento, carga y regresión con facilidad y eficiencia.

Adicionalmente a JMeter se utilizó como herramientas para la evaluación de rendimiento, Visual Studio 2010 y NetBeans para calcular el tiempo que tomó crear las entidades desde un esquema de base de datos y viceversa a través de los wizards que ofrecen cada uno de los IDEs.

A continuación se describe el diagrama de red utilizado para la valoración de los criterios de evaluación en lo que respecta al rendimiento de los frameworks:

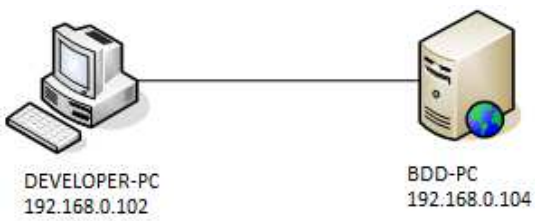

Figura 2-41 Diagrama de red. **Elaborado por:** El Autor.

Se detalla a continuación las máquinas que se utilizaron para la evaluación:

Nombre de la Máquina:	DEVELOPER-PC
Sistema Operativo:	Microsoft Windows 7
Procesador:	Intel Core i5
Memoria RAM:	4GB

Tabla 20 Características de los recursos utilizados en el desarrollo de prototipos.

Elaborado por: El Autor.

Nombre de la Máquina	BDD-PC
Sistema Operativo:	Microsoft Windows XP SP3
Procesador:	Intel Core i5
Memoria RAM:	1GB

Tabla 21 Características de los recursos utilizados en el desarrollo de prototipos.

Elaborado por: El Autor.

2.3.1.4.1.6 Configuración de las herramientas de prueba

JMeter

En el proceso de evaluación de los prototipos, esta herramienta permitió medir los tiempos de respuesta que las operaciones CRUD expuestas como servicios web consumen desde los ambientes .Net y Java, con las siguientes características:

- 5 usuarios con intervalos de un usuario cada 0.5 segundos.
- 100 transacciones por cada usuario.
- Fuente de datos con 100 registros configurados en un archivo *.csv.
- 1 caso de prueba para cada servicio web que encapsula una operación CRUD en cada uno de los ambientes .Net y Java.

La descripción detallada de la configuración de JMeter para las pruebas de rendimiento se encuentra descrita en el Anexo 3.

3 ANÁLISIS COMPARATIVO

3.1 Evaluación de prototipos

3.1.1 ADO .Net Entity Framework

3.1.1.1 Definición de metadatos

Entity Framework hasta la versión con la que se desarrolló el prototipo utiliza un mecanismo de definición de metadata que se lo expresa a través de archivos XML, que pueden ser autogenerados desde Visual Studio o creados de forma manual para configuraciones más específicas sobre el modelo de entidades.

Valoración final: 3.

3.1.1.2 Facilidades de generación de consultas (Queries)

Entiy Framework soporta en la generación de consultas lenguaje SQL y lenguajes orientados a objetos propios del framework como eSQL y Linq que permiten un trabajo más directo con el modelo de entidades. Esto fue comprobado mediante la implementación de consultas usando las distintas opciones en el desarrollo del prototipo.

Es importante mencionar que para los lenguajes de consulta orientados objetos que ofrece este framework, Visual Studio integra su aplicación Intellisense para gestionar su sintaxis y facilitar la creación de este tipo de consultas.

Valoración final: 4.

3.1.1.3 Control de transacciones y concurrencia

Entity Framework provee servicios para el control de transacciones y concurrencia través de su componente Object Services que de forma automática y transparente controla este tipo de operaciones. Este comportamiento se pudo observar en las pruebas de rendimiento realizadas al prototipo, en el cual las transacciones CRUD ejecutadas de forma concurrente se realizaron sin problemas frente a un éxito o error en la ejecución de las mismas en la base de datos.

Valoración final: 3.

3.1.1.4 Tiempo de ejecución de operación CRUD - CREATE

Esta prueba realiza la inserción de 500 clientes con 5 usuarios concurrentes. Cada operación de inserción realizada es considerada como una operación compuesta ya que la creación de un Cliente implica la creación de Tarjeta de Crédito, Actividad Profesional, Actividad Económica, Referencias Personales y Comerciales.

Una vez realizadas las mediciones se obtuvo que el tiempo promedio que a Entity Framework le toma ejecutar la operación CRUD – Create en el ambiente de pruebas fue de: 4,7193 milisegundos. Los reportes de las mediciones obtenidas se presentan en el Anexo 4.

Valoración final: 2.

3.1.1.5 Tiempo de ejecución de operación CRUD - RETRIEVE

Esta prueba realiza una consulta de 500 clientes con 5 usuarios concurrentes. Cada operación de consulta es considerada como una operación compuesta ya que la consulta de un Cliente implica la recuperación de datos en las tablas relacionadas pertenecientes a Tarjeta de Crédito, Actividad Profesional, Actividad Económica, Referencias Personales y Comerciales.

Una vez realizadas las mediciones se obtuvo que el tiempo promedio que a Entity Framework le toma ejecutar la operación CRUD – Retrieve fue de: 1,3716 milisegundos. Los reportes de las mediciones obtenidas se presentan en el Anexo 4.

Valoración final: 2.

3.1.1.6 Tiempo de ejecución de operación CRUD - UPDATE

Esta prueba realiza la actualización de 500 Clientes con 5 usuarios concurrentes. Cada operación de actualización es considerada como una operación compuesta ya que la actualización de un Cliente implica actualizar los datos de sus tablas relacionadas pertenecientes a Tarjeta de Crédito, Actividad Profesional, Actividad Económica, Referencias Personales y Comerciales.

Una vez realizadas las mediciones se obtuvo que el tiempo promedio que a Entity Framework le toma ejecutar la operación CRUD – Update fue de: 0,8443 milisegundos.

Los reportes de las mediciones obtenidas se presentan en el Anexo 4.

Valoración final: 1.

3.1.1.7 Tiempo de ejecución de operación CRUD - DELETE

Esta prueba realiza la eliminación de 500 Clientes con 5 usuarios concurrentes. Cada operación de eliminación es considerada como una operación compuesta ya que la eliminación de un Cliente implica eliminar los datos de sus tablas relacionadas pertenecientes a Tarjeta de Crédito, Actividad Profesional, Actividad Económica, Referencias Personales y Comerciales.

Una vez realizadas las mediciones se obtuvo que el tiempo promedio que a Entity Framework le toma ejecutar la operación CRUD – Delete fue de: 1,0443 milisegundos.

Los reportes de las mediciones obtenidas se presentan en el Anexo 4.

Valoración final: 1.

3.1.1.8 Tiempo en la generación del modelo de entidades

Esta prueba mide el tiempo que le tomó a Entity Framework generar el modelo de entidades que se puede ver en el Anexo 5 partiendo del esquema de base de datos descrito en el Anexo 6.

Una vez realizadas las mediciones se obtuvo que el tiempo que a Entity Framework le toma generar el modelo de entidades partiendo del esquema de base de datos fue de: 1.031 segundos.

Valoración final: 1.

3.1.1.9 Requerimientos de Hardware y Software para el desarrollo

Entity Framework no requirió de ningún tipo de software o hardware adicional al descrito en la sección 2.3.1.4.1.4, el cual se ajusta al requerido en la sección 1.2.1.5. Sin embargo es limitado en cuanto al soporte de sistemas operativos.

Valoración final 2.

3.1.1.10 Curva de aprendizaje

La valoración final a este criterio se dará en base a la experiencia personal durante el desarrollo del prototipo. Se debe considerar que se trata de un usuario con alrededor de 2 años de experiencia en el desarrollo de aplicaciones de escritorio, con conocimientos medios de lenguajes como C# y Visual Basic y base de datos, con un nivel bajo de conocimiento en conceptos web y manejo de servicios. Con estos antecedentes, se tiene que la experiencia con Entity Framework resultó satisfactoria ya que en un tiempo aproximado de 1 mes de trabajo empleando 5 horas por día se logro un nivel medio de conocimiento de los componentes que intervinieron en la arquitectura de desarrollo utilizada para este prototipo.

Gracias al apoyo de los recursos disponibles para el aprendizaje se logró desarrollar una aplicación haciendo uso del framework con integración a diferentes tecnologías (web, servicios, Windows forms, entidades, base de datos).

Valoración final: 3.

3.1.1.11 Herramientas de desarrollo

Entity Framework a través de Visual 2010 permite el desarrollo de aplicaciones de una forma bastante intuitiva y ágil al ofrecer un IDE en el que cada uno de los elementos que se necesita para crear y probar una aplicación empresarial, desde la capa de datos hasta la capa de presentación se los puede crear y utilizar desde la misma herramienta. Es por esa razón que para la creación del prototipo siguiendo la arquitectura de desarrollo que se puede observar en la figura 2-40 no se necesitó más que Visual Studio 2010.

Valoración final: 4.

3.1.1.12 Facilidad de interacción con otras tecnologías

Si bien en la evaluación previa de la sección 1.2.2.11 y la descripción del capítulo 1 se menciona el soporte a varias bases de datos de terceros, la mayoría no ofrece mecanismos directos de conexión requiriendo de un conocimiento más profundo del framework y del API para poder realizar actividades que lo haría de forma nativa y automática con SQL Server. Por otra parte para las demás capas existen APIs que permitan su integración con herramientas siempre y cuando pertenezcan al entorno Microsoft.

Valoración final: 2.

3.1.2 EJB 3/ JPA

3.1.2.1 Definición de metadatos

JPA permite el uso de varios tipos de mecanismos para expresar metadata en las configuraciones del modelo de objetos, seguridad, etc., por medio de anotaciones y/o archivos XML - heredados de los Entity Beans en versiones anteriores a EJB 3 - permitiendo que el desarrollador cuente con más opciones al momento de crear o dar soporte a la capa de persistencia.

Valoración final: 4.

3.1.2.2 Facilidades de generación de consultas (Queries)

JPA soporta diferentes lenguajes y librerías que facilitan la generación de consultas, ya sea con orientación a objetos como JPQL y Criteria API o basados en SQL los primeros enfocados al modelo de negocio y el último que permite un trabajo más directo con el esquema de base de datos.

Valoración final: 4.

3.1.2.3 Control de transacciones y concurrencia

JPA como parte de EJB3 permite que el control de transacciones y concurrencia se lo realice de forma nativa y transparente a través de los beans de sesión y en aplicaciones JEE como es el caso del prototipo realizado con este framework, el control transaccional y concurrente se lo delega al servidor de aplicaciones,

permitiendo que estos procesos se encuentren automatizados y que la atención se centre en el desarrollo del producto.

Valoración final: 3.

3.1.2.4 Tiempo de ejecución de operación CRUD - CREATE

Esta prueba realiza la inserción de 500 clientes con 5 usuarios concurrentes. Cada operación de inserción realizada es considerada como una operación compuesta ya que la creación de un Cliente implica la creación de Tarjeta de Crédito, Actividad Profesional, Actividad Económica, Referencias Personales y Comerciales.

Una vez realizadas las mediciones se obtuvo que el tiempo promedio que a JPA le toma ejecutar la operación CRUD – Create fue de: 3,5713 milisegundos.

Los reportes de las mediciones obtenidas se presentan en el Anexo 7.

Valoración final: 1.

3.1.2.5 Tiempo de ejecución de operación CRUD - RETRIEVE

Esta prueba realiza una consulta de 500 clientes con 5 usuarios concurrentes. Cada operación de consulta es considerada como una operación compuesta ya que la consulta de un Cliente implica la recuperación de datos en las tablas relacionadas pertenecientes a Tarjeta de Crédito, Actividad Profesional, Actividad Económica, Referencias Personales y Comerciales.

Una vez realizadas las mediciones se obtuvo que el tiempo promedio que a JPA le toma ejecutar la operación CRUD – Retrieve fue de: 0,6197 milisegundos.

Los reportes de las mediciones obtenidas se presentan en el Anexo 7.

Valoración final: 1.

3.1.2.6 Tiempo de ejecución de operación CRUD - UPDATE

Esta prueba realiza la actualización de 500 Clientes con 5 usuarios concurrentes. Cada operación de actualización es considerada como una operación compuesta ya que la actualización de un Cliente implica actualizar los datos de sus tablas

relacionadas pertenecientes a Tarjeta de Crédito, Actividad Profesional, Actividad Económica, Referencias Personales y Comerciales.

Una vez realizadas las mediciones se obtuvo que el tiempo promedio que a JPA le toma ejecutar la operación CRUD – Update fue de: 1,5908 milisegundos.

Los reportes de las mediciones obtenidas se presentan en el Anexo 7.

Valoración final: 2.

3.1.2.7 Tiempo de ejecución de operación CRUD - DELETE

Esta prueba realiza la eliminación de 500 Clientes con 5 usuarios concurrentes. Cada operación de eliminación es considerada como una operación compuesta ya que la eliminación de un Cliente implica eliminar los datos de sus tablas relacionadas pertenecientes a Tarjeta de Crédito, Actividad Profesional, Actividad Económica, Referencias Personales y Comerciales.

Una vez realizadas las mediciones se obtuvo que el tiempo promedio que a JPA le toma ejecutar la operación CRUD – Delete fue de: 1,7902 milisegundos.

Los reportes de las mediciones obtenidas se presentan en el Anexo 7.

Valoración final: 2.

3.1.2.8 Tiempo en la generación del modelo de entidades

Esta prueba mide el tiempo que le tomó a JPA generar el modelo de entidades que se puede ver en el Anexo 5 partiendo del esquema de base de datos descrito en el Anexo 6.

Una vez realizadas las mediciones se obtuvo que el tiempo que a JPA le toma generar el modelo de entidades partiendo del esquema de base de datos fue de: 2.14 segundos.

Valoración final: 2.

3.1.2.9 Requerimientos de Hardware y Software para el desarrollo

Como se puede ver en la sección 2.3.1.4.1.4 JPA requiere de diferentes tipos de programas para el desarrollo de una aplicación empresarial que no siempre van a

pertenecer a la misma casa comercial por lo que los requerimientos de Hardware y Software van a depender mucho de las herramientas que se seleccionen de toda la gama de productos que ofrece el ambiente Java.

Valoración final 3

3.1.2.10 Curva de aprendizaje

La valoración final a este criterio se dará en base a la experiencia personal durante el desarrollo del prototipo. Se debe considerar que se trata de un usuario con alrededor de 2 años de experiencia en el desarrollo de aplicaciones de escritorio, con conocimientos medios de lenguajes como C# y Visual Basic y base de datos, con un nivel bajo de conocimiento en Java, conceptos web y manejo de servicios. Con estos antecedentes, se tiene que la experiencia con EJB 3 y JPA exigió una mayor atención e investigación por el tiempo que tomó seleccionar y familiarizarse con las herramientas de desarrollo que se seleccionaron del gran conjunto de opciones que ofrece JAVA para la creación de aplicaciones empresariales. Para este caso si se necesitó de la guía de personas con experiencia en esta tecnología para crear una aplicación que a la larga no provoquen dificultades en el proceso de desarrollo. Por otro lado el desarrollo del prototipo con este framework se puede considerar satisfactorio ya que en un tiempo aproximado de un mes y medio de trabajo empleando 5 horas por día se logró un nivel aceptable de conocimiento con lo que se pudo desarrollar una aplicación haciendo uso del framework e integrándolo con diferentes tecnologías (web, servicios, JEE, JSE, entidades, base de datos).

Valoración final: 2.

3.1.2.11 Herramientas de desarrollo

JPA al ser parte del entorno Java tiene una gran variedad de herramientas que permiten crear y probar cada una de las capas que pueden participar en una aplicación empresarial, como el caso del IDE NetBeans que permitió crear e integrar los elementos que formaron parte del prototipo desarrollado con este framework siguiendo la arquitectura de desarrollo que se puede observar en la figura 2-39.

Valoración final: 4.

3.1.2.12 Facilidad de interacción con otras tecnologías

JPA para la interacción con la capa de datos cuenta en el mercado con varios APIs que permiten una conexión directa y sin problemas con bases de datos de terceros.

Por otra parte la interacción con otras capas no presenta problemas siempre y cuando pertenezcan al entorno Java, es así que de forma nativa JPA por ser parte de EJB3 ofrece mecanismos que facilitan la comunicación con la capa de datos y la capa de negocios, para las demás capas existe un gran número de APIs que el desarrollador puede seleccionar basándose en su experiencia y experticia.

Valoración final: 4.

3.2 Comparación de Frameworks

Se utilizará a continuación la matriz de priorización para determinar cuál de los frameworks es el más adecuado para la aplicación considerada.

Se presenta en primer lugar la matriz de priorización considerando los valores sin multiplicarlos por los pesos de los criterios obtenidos en la matriz de comparación descrita en la sección 1.1.4 (Elaboración de la matriz de comparación), luego la matriz normalizada con los valores multiplicados por los pesos de cada criterio, la cual será la considerada de forma definitiva en la priorización.

Matriz de Priorización Final sin normalizar

La matriz descrita a continuación está formada por los valores establecidos en cada uno de los criterios analizados en la sección 3.1 (Evaluación de prototipos).

	Madurez (G1)	Soporte y Documentación (G2)	Tipo de Licencia-miento (G3)	Definición de metadatos (F1)	Generación de consultas (F2)	Control de transacciones y concurrencia (F3)	Tiempo de ejecución de operaciones CRUD - Create	Tiempo de ejecución de operaciones CRUD - Retrieve	Tiempo de ejecución de operaciones CRUD - Update	Tiempo de ejecución de operaciones CRUD - Delete	Tiempo en la Generación del Modelo de Entidades	Requerimiento de HW y SW (R1)	Costo de desarrollo (R2)	Curva de Aprendizaje (FD1)	Herramientas de desarrollo (FD2)	Facilidad de interacción con otras tecnologías (FD3)	Total
Entity Framework	3	4	1	3	4	3	2	2	1	1	1	2	4	3	4	2	**41**
EJB3/JPA	4	4	2	4	4	3	1	1	2	2	2	3	4	2	4	4	**46**

Tabla 22 Matriz de Priorización Final sin normalizar.
Elaborado por: El Autor.

Matriz de Priorización Final normalizada

La Matriz de Priorización Final normalizada añade a su valoración los pesos de los criterios obtenidos en la matriz de comparación descrita en la sección 1.1.4 (Elaboración de la matriz de comparación).

	Madurez (G1)	Soporte y Documentación (G2)	Tipo de Licencia-miento (G3)	Definición de metadatos (F1)	Generación de consultas (F2)	Control de transacciones y concurrencia (F3)	Tiempo de ejecución de operaciones CRUD - Create	Tiempo de ejecución de operaciones CRUD - Retrieve	Tiempo de ejecución de operaciones CRUD - Update	Tiempo de ejecución de operaciones CRUD - Delete	Tiempo en la Generación del Modelo de Entidades	Requerimiento de HW y SW (R1)	Costo de desarrollo (R2)	Curva de Aprendizaje (FD1)	Herramientas de desarrollo (FD2)	Facilidad de interacción con otras tecnologías (FD3)	Total
Entity Framework	0.08	0.04	0.01	0.15	0.17	0.52	0.19	0.19	0.1	0.1	0.05	0.03	0.09	0.46	0.08	0.07	**2.33**
EJB3/JPA	0.11	0.04	0.03	0.2	0.17	0.52	0.1	0.1	0.19	0.19	0.1	0.05	0.09	0.31	0.08	0.14	**2.41**

Tabla 23 Matriz de Priorización Final normalizada.
Elaborado por: El Autor.

3.3 Análisis de Resultados

Realizadas las mediciones y tabulados los datos en la matriz de priorización se obtienen los siguientes análisis a partir de los resultados:

- Tomando en cuenta los resultados finales se puede observar que no existe una diferencia marcada entre los frameworks evaluados, por lo que ambos serían aptos para la creación de una aplicación empresarial, las características técnicas en los ambientes de cada framework son similares, por lo cual la decisión en una organización para optar por uno u otro queda a decisión y análisis de criterios administrativos para lo cual puede aplicar la misma metodología de comparación.

- Las características comunes de los frameworks son: buena documentación, soporte amplio por parte de sus fabricantes así como de la comunidad de desarrolladores, facilidad y versatilidad en la generación de consultas, control de transacciones y concurrencia de forma automática y transparente, costo de desarrollo por debajo de los $1000 y una variada gama de herramientas de desarrollo que permiten crear cada una de las capas de una aplicación empresarial.

- El punto más fuerte de Entity Framework es la curva de aprendizaje ya que la mayoría de elementos que forman la capa de persistencia pueden ser creados de forma automática o guiados a través de asistentes utilizando un único IDE de desarrollo, sin necesidad de que el desarrollador tenga experticia con el framework haciendo más fácil y transparente la creación de esta capa.

- El punto más fuerte de EJB3/JPA es la facilidad de interacción con otras tecnologías ofreciendo un gran conjunto de APIs que facilitan la comunicación de capas utilizando una gran variedad de especificaciones y tecnologías basadas en el entorno JAVA.

- En lo referente al rendimiento no se obtuvo un framework que predomine las pruebas CRUD que se realizaron en cada ambiente, cada uno de estos tomó tiempos que hicieron notar su buen desempeño sin una diferencia marcada el uno del otro. Lo que denota el crecimiento de los frameworks y su enfoque para conseguir consultas optimizadas que priorizan el rendimiento.

- Para el desarrollo de aplicaciones empresariales con EJB3/JPA además de conocer el framework de persistencia es necesario adquirir experiencia con la tecnología JEE ya que funciona de manera interrelacionada con componentes de esta arquitectura.

- JPA al formar parte de la especificación EJB, se acopla de manera natural en proyectos de mayor complejidad debido al respaldo de la arquitectura de aplicaciones JEE de la cual EJB es una parte fundamental.

4 CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- El documento guía obtenido del análisis de los frameworks del presente trabajo permite discriminar la aplicabilidad de los frameworks EJB3 y ADO.NET Entity Framework en el desarrollo de aplicaciones empresariales.

- Con la finalidad de escoger una plataforma adecuada para la implementación de aplicaciones empresariales es importante determinar sus beneficios y debilidades en base a criterios de evaluación. En este trabajo se han establecido criterios referenciales sobre: madurez, soporte, licenciamiento, rendimiento, funcionalidad, recursos y facilidades de desarrollo, que servirán como base para las organizaciones a fin de seleccionar el framework más idóneo de acuerdo a sus necesidades especificas.

- Con la implementación de los prototipos fue posible validar las características de cada framework así como valorar el cumplimiento de los criterios descritos y analizados teóricamente. Por ejemplo en el caso de Entity Framework, teóricamente se mencionaba el soporte a varias bases de datos de terceros, sin embargo en la práctica la mayoría no cumplió las expectativas durante el desarrollo al ofrecer mecanismos en los que la comunicación con el framework se lo debe realizar de forma manual o a través de librerías que implican costo al desarrollo.

- La experiencia del desarrollo de prototipos permitió notar que es necesario mejorar el soporte que hasta el momento dan las empresas y comunidades de los frameworks con respecto a la integración de estas herramientas con las demás capas de una aplicación empresaria, para de esta forma ayudar a los desarrolladores novatos a disminuir la curva de aprendizaje y los tiempos de implementación que conlleva este tipo de aplicaciones.

- De acuerdo al análisis de resultados de la evaluación, se pudo determinar:

    - Entity Framework es el framework que ofrece mayores facilidades de uso al crear una capa de persistencia ya que permite la creación y configuración de la mayoría de sus elementos de forma gráfica haciendo de este un desarrollo más transparente que abstrae los detalles técnicos del framework. Entre sus mayores desventajas está la dificultad de utilizar herramientas de terceros en la capa de datos y la dependencia hacia tecnologías de su propia empresa.

    - EJB3/JPA es el framework más solido para una arquitectura más compleja gracias al respaldo de un conjunto amplio de especificaciones como JEE y EJB para la interaccion de la capa de persistencia con el resto de capas de una aplicación empresarial.

    - Entity Framework al ser un framework especializado en la capa de persistencia necesita de otras herramientas que le permitan integrarse con la capa de negocio, caracteristica que por su lado

EJB 3 lo ofrece de forma nativa ya que esta especificación integra componentes para la parte del negocio como los beans de sesión y JPA para la parte de persistencia permitiendo que a través de un solo framework se pueda gestionar dos capas de una aplicación empresarial.

4.2 RECOMENDACIONES

- Para complementar el estudio realizado sería beneficioso extender el análisis de las herramientas que le permiten a estos frameworks de persistencia comunicarse con las otras capas con el objetivo de cubrir todo el espectro de una aplicación empresarial.

- Para futuros análisis de frameworks de desarrollo de aplicaciones empresariales se puede tomar como base los criterios de evaluación expuestos en este trabajo, así como la matriz de comparación para facilitar la selección de los más adecuados que se ajusten a necesidades particulares.

- Para complementar el estudio realizado se debería considerar el análisis de frameworks adicionales para aplicaciones empresariales, como el caso de Hibernate y Spring que cuentan con versiones para Java y .Net y que durante el periodo de desarrollo de este trabajo tuvieron notables avances y aceptación en el mercado.

- Para futuras comparaciones sería conveniente estudiar la forma de automatizar la toma de decisiones en base al conjunto de criterios seleccionados de tal forma que el tiempo y costo que puede tomar este análisis no sea de gran impacto y el enfoque se lo de a la selección de la aplicación que se ajuste a las necesidades de la organización.

- Desarrollar o analizar la existencia de software utilitario que permita automatizar la creación de las matrices utilizadas para la comparación de los frameworks para que el trabajo se centre en la valoración y comparación de estos, evitando la intervención manual y disminuir el tiempo que esto representa.

GLOSARIO

1.1 **Framework.-** Es una estructura conceptual y tecnológica, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

1.2 **Object Relational Impedance Mismatch.-** Es un conjunto de limitaciones técnicas y conceptuales que son frecuentemente encontradas cuando un sistema de base de datos relacional es usado por un programa escrito en un lenguaje de programación orientado a objetos.

1.3 **CRUD.-** Acrónimo utilizado para referirse a las funciones básicas en bases de datos; Crear, Obtener, Actualizar y Borrar (del original en inglés: Create, Read, Update and Delete).

1.4 **SQL.-** Lenguaje de consulta estructurado o SQL (por sus siglas en inglés Structured Query Language). Es un lenguaje declarativo y normalizado de acceso a bases de datos relacionales que permite efectuar consultas, con el fin de recuperar de una forma sencilla información de una base de datos, así como también hacer cambios sobre ella.

1.5 **ADO .NET.-** Es un conjunto de componentes de software que forman parte de la biblioteca de clases base del Framework .Net de Microsoft y permiten acceder y modificar los datos almacenados en un Sistema Gestor de Base Datos Relacional.

1.6 **Data Reader.-** Es un objeto utilizado para acceder a la información de una base de datos que permite realizar consultas únicamente de lectura.

1.7 **Data Set.-** Es un objeto de acceso a datos utilizado como un caché de memoria interna de datos recuperados de un origen de datos, representa un esquema (una base de datos entera o un subconjunto de esta) con características como tablas, relaciones, etc. Los datos de un Data Set pueden ser accedidos con o sin conexión al origen de datos.

1.8 **LINQ (Language Integrated Query).-** Es un componente del Framework de .Net de Microsoft que agrega a los lenguajes de .Net nuevas capacidades de consulta de datos, utilizando una sintaxis similar a la de SQL.

LINQ introduce patrones estándares fáciles de aprender para consultar y actualizar los datos, que le permiten a esta tecnología el soporte a cualquier tipo de base de datos. Visual Studio incluye librerías proveedores de LINQ que permiten el uso de LINQ con colecciones de .NET Framework (LINQ to Objects), bases de datos SQL Server (LINQ to SQL - DLINQ), DataSets de ADO.NET (LINQ to DataSets), documentos XML (LINQ to XML - XLINQ) y modelos conceptuales (LINQ to Entities).

1.9 **JEE (Java Enterprise Edition).-** Es una plataforma de programación para desarrollar y ejecutar aplicaciones con arquitectura multicapa en un ambiente distribuido. ejecutándose sobre un servidor de aplicaciones. La especificación JEE define los siguientes componentes: Aplicaciones clientes y Applets, Java Servlets y JSP's (Java Server Pages), Enterprise Java Beans (EJB).

1.10 **POJO (Plain Old Java Object). -** Es una clase simple de Java que no extiende o implementa ningún tipo de clase especial.

1.11 **RUP.-** Proceso Unificado Relacional o RUP (por sus siglas en inglés Rational Unified Process). Es un proceso de desarrollo de software iterativo creada por Rational Software Corporation que junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

1.12 **XP.-** La Programación Extrema, mejor conocida por su nombre en inglés Extreme Programming (XP), es una de las llamadas Metodologías Agiles de desarrollo de software más utilizadas en estos tiempos formulado por Kent Beck. Esta metodología define iteraciones frecuentes en cortos ciclos de desarrollo, que tiene por objeto mejorar la productividad e introducir puntos de control donde los nuevos requisitos de los clientes se pueden adoptar.

BIBLIOGRAFÍA

Bauer, C., & King, G. (2007). *Java Persistence with Hibernate.* Greenwich: Manning Publications.

IBM Corporation. (2004). *Concepts: Relational Databases and Object Orientation*. Retrieved 10 18, 2010, from Rational Unified Process: http://students.mimuw.edu.pl/~zbyszek/posi/ibm/RUP_Eval/process/workflow/ana_desi/co_rdbom.htm

Microsoft. (2011). *Visual Studio 2008 Professional* . Retrieved 2011, from Microsoft Download Center: http://www.microsoft.com/downloads/es-es/details.aspx?FamilyID=83C3A1EC-ED72-4A79-8961-25635DB0192B#SystemRequirements

Microsoft. (2011). *Visual Studio 2010 Editions*. Retrieved 2011, from Microsoft Visual Studio: http://www.microsoft.com/visualstudio/en-us/products/2010-editions

Pothu, S. (2008). *A Comparative Analysis of Object-relational mappings for Java, MSc. Thesis.* University of Applied Sciences at Braunschweig/Wolfenbuettel.

Sikora, M. (2008). *EJB3 Developer Guide.* Packt.

Tenny, L., & Hirani, Z. (2010). *Entity Framework 4.0 Recipes.* Apress.

Universidad Politécnica de Valencia. (2010). *Grupo de Valoración y Decisi'on Multicriterio.* Retrieved January 25, 2011, from PoliMedia: https://polimedia.upv.es/visor/?id=da298ce5-c481-0d4f-a40e-937380d3b4f5

Ávila, R. (2000). *El AHP (Proceso Analítico Jerárquico) y su aplicación para determinar los usos de las tierras*. Documento PDF. Diciembre 2000. http://www.rlc.fao.org/proyecto/139jpn/document/3dctos/sirtplan/infotec/2_AHP.pdf

ANEXOS

- Anexo 1 Matriz de comparación sin normalizar.
- Anexo 2 Matriz de comparación normalizada.
- Anexo 3 Configuración de JMeter para las pruebas de rendimiento de los Prototipos.
- Anexo 4 Pruebas de rendimiento Entitty Framework.
- Anexo 5 Modelo de Entidades.
- Anexo 6 Esquema de Base de Datos.
- Anexo 7 Pruebas de rendimiento EJB3 / JPA.
- Anexo 8 Plantilla de Solicitud de Tarjeta de Crédito.