# Universidad de las Ciencias Informáticas

## Facultad 1


# Subsistema Huésped Casero para la Sección de Enfrentamiento del Departamento Policía de Inmigración

**Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas**

**Autores:** Diana Milagros García Bazán.

Antonio Losada Martínez.

**Tutoras:** MSc. Martha Y. Ambruster Crespo.

Ing. Helen Arbelo Montero.

Ing. Bárbara Yamily Mansito Reyes.

Ciudad de la Habana, junio de 2010.

"Año de 51 Aniversario del Triunfo de la Revolución"

*"...Si nuestra obra es imperfecta, y no alcanza el fin propuesto, demostrará, sin embargo, que hay otro grado superior más próximo al fin deseado"*

*Juan Amos Comenio*

## DECLARACIÓN DE AUTORÍA

Declaro que somos los únicos autores de este trabajo titulado: Subsistema Huésped Casero para la Sección de Enfrentamiento del Departamento Policía de Inmigración, y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Diana Milagros García Bazán
Autor

Antonio Losada Martínez
Autor

MSc. Martha Y. Ambruster Crespo
Tutora

Ing. Helen Arbelo Montero
Tutora

Ing. Bárbara Y. Mansito Reyes
Tutora

## DATOS DE CONTACTO

**MSc. Martha Ambruster Crespo:** Graduada de Ingeniería Informática en el Instituto Superior Politécnico José Antonio Echeverría, en el año 2003. Actualmente se desempeña como profesora de la Universidad de Ciencias Informáticas, en la cual ha impartido clases de Programación *Web*, Estructura de Datos y Gráfico por Computadoras. Ha participado en proyectos productivos en el Grupo de Procesamiento de Imágenes de la Facultad 7 y actualmente forma parte del grupo de desarrollo del Proyecto Identidad Cuba del Centro de Identificación y Seguridad Digital.
mambruster@uci.cu

**Ing. Helen Arbelo Montero:** Graduada de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas, en el año 2007. Actualmente se desempeña como profesora de la universidad antes citada, en la cual ha impartido clases de Programación *Web*, Sistemas de Bases de Datos e Inteligencia Artificial. Ha participado en proyectos productivos en el Centro de Identificación y Seguridad Digital como SAIME, para Venezuela. Actualmente forma parte del grupo de desarrollo del Proyecto Identidad Cuba perteneciente al mismo centro.
harbelo@uci.cu

**Ing. Bárbara Yamily Mansito Reyes:** Graduada de Ingeniería Industrial en el Instituto Superior Politécnico José Antonio Echeverría, en el año 2009, con participación en proyectos de investigación como parte de la colaboración CUJAE-MININT. Actualmente pertenece al grupo de desarrollo del Proyecto Identidad Cuba que reside en la Universidad de las Ciencias Informáticas.
byamily@uci.cu

**AGRADECIMIENTOS**

*Agradecemos de manera muy especial a nuestros compañeros: Alfredo, Ernesto Dasa, Ernesto Kindelán, Reynaldo Mavilio, Reynier Blanco, Elizabeta, Angel Boloy, Manuel Quert y otros, por su ayuda y apoyo en todo momento.*

*A la Universidad de las Ciencias Informáticas, al MININT y a la Revolución por convertirnos en buenos profesionales.*

*A nuestras tutoras, por su apoyo incondicional.*

*A todos los que hicieron este momento real y especial.*

***Diana.***

*A mis padres, hermanos y abuelos, por la formación y el amor incondicional.*

*A mi novio Joan Manuel, por el amor, la paciencia, el incentivo que tuvo para mí en todo momento, a su familia, en especial a sus padres y hermano.*

*A mi compañero de tesis, porque en este tiempo de trabajo juntos se convirtió en un gran amigo.*

***Antonio.***

*A mis padres y hermana que son mis razón de ser, el motivo de cada uno de mis esfuerzos y a quien van dedicados todos mis logros como persona y profesional, a ellos mi vida, decir gracias es insuficiente.*

*A mis abuelos y tíos que me han brindado su apoyo y siempre han confiado en que este día podía hacerse realidad.*

*A Naila que es mi vida, mi amiga, mi hermana, la mujer que admiro y que siempre he tenido a mi lado en momentos duros y felices. Para ti mi corazón y mi eterna amistad.*

**DEDICATORIA**

***Diana.***

*A mi mamá Deysi, por ser el ser más especial en mi vida, por su confianza y porque su sacrificio nunca fue en vano.*

*A mi abuela materna, por enseñarme que todo el que persevera triunfa, por cuidarme con tanto esmero y cariño.*

*A mi primo Yunier porque quiero cumplir su sueño de convertirse en un gran profesional.*

***Antonio.***

*A mis padres, por su amor y apoyo…*

*A Lourdita, por quererme siempre y soportarme…*

**RESUMEN**

Dentro del Ministerio del Interior se encuentra la Dirección de Inmigración y Extranjería, la que se encuentra inmersa en el proceso de automatización de sus servicios y áreas con el objetivo de ejecutar y controlar el cumplimiento de la política migratoria. Una de las áreas a las que se hace referencia es el Departamento Policía de Inmigración y dentro del mismo a la Sección de Enfrentamiento.

En la actualidad, la gestión de estancias de extranjeros se apoya en un sistema obsoleto y la mayoría de las actividades se realizan de forma manual en algunos casos haciendo lento el proceso, sumándose a lo anterior que los registros operativos no están integrados.

El presente trabajo describe el desarrollo de un subsistema para automatizar la gestión de las estancias de la masa extranjera que se lleva a cabo por la Sección de Enfrentamiento del Departamento Policía de Inmigración.

El desarrollo de la aplicación facilitará el trabajo de los Inspectores de Enfrentamiento en la obtención de la información de la gestión de las estancias, la obtención de reportes, reducirá los tiempos de respuesta y existirá un mayor aprovechamiento de los datos digitalizados.

El Subsistema Huésped Casero, se realiza sobre la plataforma *.NET*, con el entorno de desarrollo integrado *Visual Studio TeamSystem 2008*, que incluye nuevas tecnologías, como son: el *Entity Framework*, *LINQ(Language-Integrated Query)*, *Windows Workflow Fundation*, *Windows Communication Fundation*, entre otros, lo que permite la creación de proyectos, brindando muchas facilidades para el diseño y la implementación.

**PALABRAS CLAVE**

**Áreas de atención, Sección de Enfrentamiento, Departamento Policía de Inmigración, Control territorial, Gestión de estancias.**

## ÍNDICE GENERAL

## ÍNDICE DE LAS TABLAS

## ÍNDICE DE LAS FIGURAS

## INTRODUCCIÓN

La Universidad de las Ciencias Informáticas (UCI), tiene como misión producir software y servicios informáticos a partir de la vinculación estudio-trabajo-investigación como modelo de formación. Hace algún tiempo se han realizado intercambios entre el Ministerio del Interior (MININT) y la UCI con el objetivo de establecer acuerdos, que permitan desarrollar proyectos. Entre ellos se encuentra el establecido con la Dirección de Inmigración y Extranjería (DIE), que es el órgano responsable de aplicar y controlar las normativas legales relacionadas con la migración, extranjería y ciudadanía en Cuba.

Durante el último período del 2008 e inicios del 2009, se realizó un diagnóstico del estado actual de las distintas áreas de la DIE, entre las que se encuentra el Departamento Policía de Inmigración; con el objetivo de proponer la modernización de los procesos y tecnologías, para facilitar la ejecución de los diversos trámites correspondientes a su objeto de trabajo, así como la informatización de estas áreas que son las que soportan su correcto funcionamiento.

En el diagnóstico se detecta la siguiente **situación problémica:**

- Los oficiales encargados del control de extranjeros en el terreno presentan limitaciones para consultar los registros de la DIE y del resto MININT, lo que provoca un uso excesivo de papel y modelajes para conformar expedientes y otros documentos.
- El sistema para el control del huésped casero, carece de herramientas que permitan obtener información de manera automatizada para realizar el enfrentamiento.
- El diseño funcional del sistema para el control del huésped casero está basado fundamentalmente en dejar un registro de los huéspedes alojados, sin tener en cuenta otras alternativas en función del enfrentamiento.
- El registro del huésped casero que se realiza en las oficinas del Carné de Identidad y Registro de Población (CIRP), tributa a las bases de datos de este órgano y posteriormente con este resultado se carga y actualiza el Sistema Integral Automatizado (SIA-DIE), lo que presenta dificultades pues no existe información real acerca de los extranjeros registrados.

Para dar solución a esta situación problémica que actualmente presenta la DIE, se plantea el siguiente **problema científico**:

¿Cómo mejorar el proceso para la gestión de estancias de la masa extranjera que se llevan a cabo por la Sección de Enfrentamiento del Departamento Policía de Inmigración?

Estableciendo como **objeto de estudio,** los procesos que se realizan para el control y supervisión de la masa extranjera en el Departamento Policía de Inmigración; y centrándose en el proceso para la

gestión de estancias de la masa extranjera que se desarrolla en la Sección de Enfrentamiento como **campo de acción**.

Para dar solución al problema científico, se define el siguiente **objetivo general:** desarrollar un subsistema que permita mejorar el proceso para la gestión de estancias de la masa extranjera que se llevan a cabo por la Sección de Enfrentamiento del Departamento Policía de Inmigración. A continuación se desglosa el objetivo general en los siguientes **objetivos específicos**:

- Analizar los procesos de negocio que se realizan para gestionar las estancias de la masa extranjera que se llevan a cabo por la Sección de Enfrentamiento del Departamento Policía de Inmigración.
- Modelar los procesos de negocio para la gestión de estancias de la masa extranjera que se llevan a cabo por la Sección de Enfrentamiento del Departamento Policía de Inmigración.
- Desarrollar el levantamiento de los requerimientos del Subsistema Huésped Casero para la Sección de Enfrentamiento del Departamento Policía de Inmigración.
- Diseñar e implementar el Subsistema Huésped Casero para la Sección de Enfrentamiento del Departamento Policía de Inmigración.
- Validar el Subsistema Huésped Casero para la Sección de Enfrentamiento del Departamento Policía de Inmigración en un ambiente de pruebas.

De todo lo planteando surge la siguiente **hipótesis**:

El desarrollo del Subsistema Huésped Casero, contribuye a la mejora del proceso para la gestión de estancias de la masa extranjera en la Sección de Enfrentamiento del Departamento Policía de Inmigración.

Se definieron las siguientes **variables:**

- Variables independientes: El Subsistema Huésped Casero para la Sección de Enfrentamiento del Departamento Policía de Inmigración.
- Variables dependientes: La mejora del proceso para la gestión de estancias de la masa extranjera.

La operacionalización de las variables definidas, se encuentra en el Anexo 1.

Para obtener la información necesaria para el desarrollo del subsistema se desarrollaron los siguientes **métodos investigativos:**

- Métodos Teóricos:
  - Analítico–Sintético: Con el objetivo de analizar la información y la documentación relevante para el desarrollo del software, enfatizando en los elementos más importantes que se relacionan con el objeto de estudio.

- Histórico–Lógico: Para constatar teóricamente cómo ha evolucionado el desarrollo de los sistemas de gestión a través de los años.

- Métodos Empíricos:
  - Entrevista: Con el fin de obtener información referente a los procesos que se desarrollan para determinar los requerimientos, el diseño e implementación del software.

Con el desarrollo del trabajo de diploma, se esperan los siguientes **resultados:**

Obtener un subsistema informático para desarrollar el proceso para la gestión de estancias de la masa extranjera que se llevan a cabo por la Sección de Enfrentamiento del Departamento Policía de Inmigración, que permita:

- Gestionar el trabajo de los oficiales de la Sección de Enfrentamiento del Departamento Policía de Inmigración, disminuyendo de esta forma el uso excesivo de papel.
- Garantizar un registro y control de la información con eficiencia, permitiendo que se puedan realizar las actividades en función del enfrentamiento.

Por lo que se proponen las siguientes **tareas** para medir la evolución y el desarrollo de la investigación:

- Analizar la situación actual de la gestión de estancias en la Sección de Enfrentamiento del Departamento Policía de Inmigración.
- Entrevistar al personal responsable de las principales actividades.
- Elaborar el diseño teórico de la investigación.
- Definir la situación problémica, problema a resolver, objetivos, novedad científica, aportes teóricos y prácticos.
- Realizar el estado del arte.
- Revisar la bibliografía del tema.
- Analizar la existencia de otras aplicaciones o soluciones similares.
- Elaborar la propuesta de solución.
- Modelar los procesos del negocio.
- Describir las funcionalidades que debe cumplir el subsistema.
- Asimilar las herramientas necesarias para el análisis, diseño e implementación del subsistema.
- Realizar el análisis y diseño de la aplicación.
- Asimilar la arquitectura de software definida por el Proyecto Identificación, Migración y Extranjería de la República de Cuba.
- Implementar una aplicación *web* que cumpla con los requerimientos propuestos.

El presente documento posee la siguiente **estructura**:

- **Capítulo 1. Fundamentación teórica**: Se describe la gestión de procesos del negocio, de los sistemas de gestión de la información tanto a nivel internacional como nacional, así como las tendencias técnicas, tecnológicas y metodológicas.
- **Capítulo 2. Características del subsistema**: Se estudia y modela el negocio del subsistema realizando mejoras potenciales al proceso para la gestión de estancias de la masa extranjera y se determinan las funcionalidades a reflejar en los requisitos funcionales y no funcionales.
- **Capítulo 3. Análisis y diseño del subsistema**: Contiene la descripción y fundamentación de la arquitectura, el análisis y el diseño del subsistema, logrando transformar los requisitos del usuario a una especificación que describe cómo implementar el mismo.
- **Capítulo 4. Implementación y pruebas**: Contiene la implementación, el diseño de casos de pruebas, los resultados de las pruebas realizadas, los diagramas de despliegue y componentes, y las interfaces de usuario de la propuesta realizada

# CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

## 1.1 Introducción.

En este capítulo se exponen los conceptos fundamentales para una mejor comprensión del problema y el desarrollo del trabajo de diploma. Se realiza un estudio de las soluciones que existen hasta el momento sobre el proceso en el mundo y en Cuba. Además se desarrolla una familiarización con las tecnologías y las diferentes herramientas, garantizando la utilización adecuada de las mismas para cada situación que se presente en el desarrollo del subsistema que se propone.

## 1.2 Conceptos asociados al dominio del problema.

A continuación se muestran los conceptos de mayor relevancia para que pueda ser comprendido el problema en cuestión.

### 1.2.1 Control territorial de la masa extranjera:

Es la inspección en el territorio de los extranjeros y cubanos residentes en el exterior, que se encuentren hospedados en casas de renta; así como la inspección de los responsables de la renta y su actitud ante posibles violaciones.

### 1.2.2 Registro de estancia:

Es el registro de un hospedaje en un sistema, en el que se guarda la información de la persona que renta la casa, el huésped y los posibles acompañantes que pueda llevar durante su estadía en el lugar.

## 1.3 Análisis de otras soluciones existentes.

En el mundo existen muchos sistemas informáticos para la gestión de la información que identifica a las personas, pero para cada país es diferente, pues se rigen por sus leyes específicas. Las aplicaciones desarrolladas por otros países para resolver situaciones migratorias son privadas, por lo que resulta casi imposible la obtención de información pública disponible.

### 1.3.1 Sistema para el Control de Estancias en Hoteles para la Dirección de Migración y Fronteras de la República Bolivariana de Venezuela.

El control de estancias de extranjeros en los hoteles de Venezuela, es el proceso que se lleva a cabo para controlar todos los extranjeros que se encuentran hospedados en los hoteles de Venezuela. El sistema procede de la siguiente forma: capta los datos de la persona que llega al hotel, guarda en un

archivo estos datos, para de esta manera controlar las estancias de los extranjeros en un plazo de días determinado, además genera algunos reportes de los extranjeros que han sido hospedados en los establecimientos. (1)

Este sistema no se ajusta a las necesidades de Cuba porque está desarrollado para cumplir características específicas del gobierno de la República Bolivariana de Venezuela, no obstante puede ser de ayuda para desarrollar el proceso gestión de estancias en el Sistema Huésped Casero, pues tienen gran similitud.

### 1.3.2 Sistema Huésped-Casero.

El Sistema Huésped-Casero, fue concebido para el control de las actividades relacionadas con el Decreto Ley 171/97, que es el que permite que un nacional pueda hospedar extranjeros en su hogar. Por medio de las estancias se establece la relación entre las personas alojadas en casas particulares, el propietario de la vivienda que hospeda y el nacional residente en el país que los acompañó.

Este sistema permite controlar:

- Personas que arriendan la vivienda de manera ilegal.
- Personas que reportan la presencia de extranjeros en su vivienda, como resultado del acto de Registro Migratorio por poseer Visa Consular o por cambio de clasificación migratoria, dejando constancia del vínculo familiar en caso de existir este.
- Situación de la licencia, en cuanto a vigencia y cancelación de esta, de los arrendadores amparados por el Decreto Ley 171/97 para el ejercicio de la actividad.
- Violaciones detectadas relacionadas con el arrendamiento de la vivienda, y medidas adoptadas, diferenciando las medidas que impone la DIE, de las que impone la vivienda u otros órganos que actúan en el enfrentamiento de estas actividades.
- Residentes en el país que en condición de acompañantes se hospedan junto con los huéspedes.
- Una o más direcciones de residencia, de una persona de tipo casero, cuando estas son de diferentes municipios.
- Condición del casero en el momento de realizar el alquiler o alojamiento, en cuanto a la posesión o no de licencia. (2)

#### 1.3.2.1 Objetivo del Sistema Huésped-Casero.

Aumentar el control de las personas que durante su estancia en el territorio nacional se hospedan en casas particulares y de las personas con las que se relacionan ya sea porque los alojan en su vivienda,

amparados o no por el Decreto Ley 171/97 para el ejercicio de esta actividad, o en condición de acompañante.

En el sistema se controlan los tipos de personas siguientes:

- **Casero**: Propietario de vivienda.
  - Acogidos al Decreto Ley 171/97, para ejercer el alquiler de su vivienda.
  - Que reporten la presencia de extranjeros en su vivienda (Visas A2 consulares o cambio clasificación migratoria).
  - Detectados con personas alojadas en su vivienda sin cumplir los requisitos establecidos para ello, ya sea por violaciones de lo establecido en el Decreto 171/97 en lo referente a la actividad de arrendamiento o por violación de normas de Inmigración y Extranjería, incurriendo así en ilegalidades.
- **Huésped:**
  - Extranjeros y cubanos residentes en el exterior que durante su estancia en el territorio nacional, se alojan en casas particulares, amparadas o no (con o sin licencia) por el Decreto Ley 171/97 para el ejercicio de la actividad.
- **Acompañantes:**
  - Nacionales residentes en el país, que comparten el alojamiento con los huéspedes.
  - En el sistema se controlan también las violaciones cometidas por propietarios de viviendas relativas al Decreto 171/97 o normas establecidas para el control migratorio.(2)

El Sistema Huésped-Casero actual no satisface las necesidades de la Sección de Enfrentamiento, por lo que se realiza un nuevo subsistema por interés de la DIE, con el objetivo de eliminar las deficiencias de la herramienta que emplean.

### 1.4 Propuesta de solución.

Al realizar un estudio entre el Sistema para el Control de Estancias en Hoteles para la Dirección de Migración y Fronteras de la República Bolivariana de Venezuela y el Sistema Huésped Casero, se encontró que existen características y funcionalidades que pueden contribuir al desarrollo de una aplicación que facilite la realización de las actividades para la gestión de estancias de una manera más óptima, por lo que se propone la siguiente solución: **Subsistema Huésped Casero para la Sección de Enfrentamiento del Departamento Policía de Inmigración.**

## 1.5 Necesidad de implementar el subsistema.

La Sección de Enfrentamiento del Departamento Policía de Inmigración, necesita una herramienta capaz de ofrecer calidad y rapidez para el trabajo con la información que obtiene de las estancias que se realizan en el territorio en el cual trabajan. Actualmente esta sección cuenta con un sistema que no suple con la siguiente necesidad: controlar la información exacta de los extranjeros y cubanos residentes en el exterior que se encuentren hospedados en casas de nacionales, por lo que el trabajo en las oficinas es engorroso, lento e inseguro. Debido a las causas mencionadas anteriormente, es que se hace necesaria la creación del Subsistema Huésped Casero para la Sección de Enfrentamiento del Departamento Policía de Inmigración.

## 1.6 Fundamentación de la metodología utilizada.

### 1.6.1 Metodología de desarrollo a utilizar.

*Microsoft Solution Framework (MSF)*[1] es un grupo de guías para lograr que una solución en sistemas de información pueda ser finalizada exitosamente, rápidamente y reduciendo la cantidad de personas y riesgos. Las características de *MSF* son:

- Adaptable
- Flexible
- Escalable
- Agnóstico a tecnologías (3)

#### 1.6.1.1 Funcionamiento del modelo del Microsoft Solution Framework.

El modelo de proceso *MSF*, propone una secuencia generalizada de actividades para la construcción de soluciones empresariales. Este proceso es flexible y se puede adaptar al diseño y desarrollo de una amplia gama de proyectos en una empresa. El modelo *MSF* está también basado en fases, puntos de transición y carga de forma iterativa que se puede aplicar en el desarrollo de aplicaciones tradicionales, soluciones empresariales para comercio electrónico así como aplicaciones *web* distribuidas. El modelo de proceso *MSF* combina los mejores principios del modelo en cascada y del modelo en espiral. Combina la claridad que planea el modelo en cascada y las ventajas de los puntos de transición del modelo en espiral. El modelo de proceso *MSF* consta de las siguientes fases, reflejadas en la Figura 1:

[1] *Microsoft Solution Framework*: Metodología de desarrollo de Microsoft .


**Figura 1. Fases de la metodología MSF para CMMI.**

El *MSF* proporciona las mejores prácticas para planear, diseñar, convertir y desarrollar exitosas soluciones empresariales. (4)

### 1.6.2 Lenguajes de modelado.

#### 1.6.2.1 Lenguaje Unificado de Modelado (UML).

Lenguaje Unificado de Modelado (*UML*, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el *Object Management Group*[2] (OMG). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. *UML* ofrece un estándar para describir un "plano" del sistema (es decir un modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Es importante resaltar que *UML* es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o *RUP*), pero no especifica en sí mismo qué metodología o proceso usar. (5)

[2] *Object Management Group*: Grupo de administración de objetos.

#### 1.6.2.2 BPMN.

El *Business Process Management Initiative (BPMI)*[3] ha desarrollado una notación estándar llamada *Business Process Modeling Notation* (*BPMN*)[4]. El objetivo principal de los esfuerzos de *BPMN* era dar una notación rápidamente comprensible por toda esa gente de negocios, desde el analista de negocio que hace el borrador inicial de los procesos, pasando por los desarrolladores técnicos responsables de implementar la tecnología que llevarán a cabo dichos procesos, llegando finalmente a la gente de negocio que gestionará y monitorizará esos procesos. Así, *BPMN* crea un puente estandarizado para el hueco entre el diseño de los procesos de negocio y la implementación de procesos. *BPMN* define un *Business Process Diagram* (*BPD*)[5], que se basa en una técnica de grafos de flujo para crear modelos gráficos de operaciones de procesos de negocio. Un modelo de procesos de negocio, es una red de objetos gráficos, que son actividades y controles de flujo que definen su orden de rendimiento. (6)

### 1.6.3 Herramienta de modelado.

#### 1.6.3.1 Altova UModel 2009 Enterprise Edition.

*UModel* – Herramienta *UML* para el modelado y desarrollo de aplicaciones. *Altova UModel® 2009* es el punto de salida para el desarrollo de software de éxito. Diseña visualmente modelos de aplicaciones en *UML* y genera código *Java, C#, o Visual Basic .NET* y documentación del proyecto. Puede realizar ingeniería inversa de los programas existentes pasándolos a diagramas *UML 2*, luego afina los diseños y completa el viaje de ida y vuelta con la generación de código. *UModel* es la herramienta *UML* que hace el diseño visual de software práctico para cualquier proyecto.

Combina un rico interfaz visual, con funciones de usabilidad superiores para ayudar a nivelar la curva de aprendizaje de *UML*, además de incluir las más altas funcionalidades para potenciar a los usuarios con las más completas ventajas del desarrollo de software *UML*. Las características de *UModel® 2009* para el desarrollo de software basado en las capacidades de modelado avanzado son:

- Soporte para los catorce tipos de diagramas *UML* (Lenguaje Unificado de Modelado).
- Modelado de esquemas *Extensible Markup Lenguaje* (*XML*)[6] en diagramas *UML*.
- Diagramas de proceso de negocio (*BPMN*).
- Generación de código fuente en lenguajes *Java, C#, y VisualBasic.NET*.
- Ingeniería inversa de código fuente y ficheros binarios *Java, C# y VisualBasic.NET*.
- Sincronizado de modelo y código a través de ingeniería de ida y vuelta.

[3] *Business Process Management Initiative*: Iniciativa de la gestión de procesos de negocio.
[4] *Business Process Modeling Notation*: Notación para el modelado de los procesos de negocio.
[5] *Business Process Diagram*: Diagrama de procesos de negocio.
[6] XML (*Extensible Markup Language*): Lenguaje de marcado extensible.

- Crea diagramas de secuencia desde el código fuente de la ingeniería inversa.
- Generación de documentación personalizable de proyecto.
- Compartir subproyectos para colaboración o reutilización.
- Capas de diagramas con visibilidad selectiva.
- *Hyperlinks* entre diagramas, documentos, o páginas *web*.
- Integración con sistemas de control de versiones.
- *Application Programming Interface* (*API*)[7] extendida para permitir manipulaciones externas.
- Estrecha integración con *Visual Studio y Eclipse*. (7)

## 1.7 Frameworks utilizados en la solución.

### 1.7.1 Microsoft .NET framework 3.5.

*Microsoft .NET Framework 3.5* agrega de forma incremental las nuevas características de *Microsoft .NET Framework 3.0*. Por ejemplo, los conjuntos de características de *Windows Workflow Foundation* (WWF)[8], *Windows Communication Foundation* (WCF)[9] y *Windows Presentation Foundation* (WPF)[10]. Además, *.NET Framework 3.5* contiene una serie de características nuevas en distintas áreas tecnológicas que se han agregado como nuevos ensamblados para evitar cambios. Algunas de estas características son:

- Integración total de *Language Integrated Query* (*LINQ*)[11] y del reconocimiento de los datos. Esta nueva característica le permitirá escribir código en idiomas habilitados para *LINQ* para filtrar, enumerar y crear proyecciones de varios tipos de datos *SQL* (*Structured Query Language*)[12], colecciones, *XML* (*Extensible Markup Language*) y conjuntos de datos usando la misma sintaxis.
- Nueva compatibilidad con el protocolo *web* para generar servicios *WCF*, y distintos estándares nuevos.
- Compatibilidad total con las herramientas de *Visual Studio 2008* para *WWF*, *WCF* y *WPF*, incluida la nueva tecnología de servicios habilitados para flujos de trabajo. (8)

[7] API (*Application Programming Interface*): Interfaz de programación de aplicaciones.
[8] *Windows Workflow Foundation*: Fundación de flujos de trabajo de Windows.
[9] *Windows Communication Foundation*: Fundación de comunicación de Windows.
[10] *Windows Presentation Foundation*: Fundación de presentación de Windows.
[11] *Language Integrated Query*: Lenguaje de consulta integrada.
[12] SQL (*Structured Query Language*): Lenguaje de consulta estructurado.

### 1.7.2 Bison framework.

Bison es un marco de trabajo basado en la automatización de flujos de trabajo para sistemas *.NET* que responde a una arquitectura de referencia dentro del Centro de Identidad y Seguridad Digital. Representa el marco de trabajo común, y su principal objetivo es el de encapsular en un componente todas aquellas actividades y servicios que son de vital importancia para la arquitectura.

Este está estructurado por los siguientes paquetes:

- **Actividades.**

En este paquete se definen todas las actividades que por su comportamiento son necesarias en la arquitectura base. La actividad clave de este marco de trabajo es la representada por la clase *ClientActivity*[13] que permite definir la navegación de todo el proceso de forma gráfica.

- **Servicios.**

En este paquete se encuentran todos las interfaces e implementaciones de servicios pertenecientes a la arquitectura base, como son el servicio de navegación, servicio de acceso a la información del proceso, entre otros. Estos servicios representan un factor clave dentro de la aplicación ya que son los que permiten la interacción con las instancias de proceso creadas. Estos servicios son de tipo *Runtime*[14].

- **Hosting.**[15]

En este *namespace*[16] se encuentra la clase *BisonHost* la cual contiene la instancia de *Runtime* que maneja todas las instancias de procesos y servicios. (9)

### 1.7.3 ADO.NET Entity framework.

El *ADO.NET Entity Framework* está diseñado para permitir a los desarrolladores crear aplicaciones de acceso a datos mediante programación, en contra de un modelo de aplicación conceptual en lugar de programar directamente en contra de un esquema de almacenamiento relacional. El objetivo es disminuir la cantidad de código y el mantenimiento necesario para las aplicaciones de datos orientada. El *Entity Framework*[17] proporciona los siguientes beneficios:

- Las aplicaciones pueden trabajar en términos de un modelo conceptual más centrado en las aplicaciones, incluidos los tipos de herencia, los miembros del complejo, y las relaciones.

[13] *ClientActivity*: Actividad del cliente.
[14] *Runtime*: Tiempo de ejecución.
[15] *Hosting*: Manejador, Organizador.
[16] *Namespace*: Nombre de espacio.
[17] Entity Framework: Marco para entidades.

- Las solicitudes se liberan de las dependencias no modificables en un motor de datos particulares o esquema de almacenamiento.
- Asignaciones entre el modelo conceptual y el esquema de almacenamiento específico puede cambiar sin alterar el código de la aplicación.
- Los desarrolladores pueden trabajar con un modelo de objetos de aplicaciones compatibles que se pueden asignar a diferentes esquemas de almacenamiento, posiblemente en marcha en diversos sistemas de bases de datos de gestión.
- Múltiples modelos conceptuales se pueden asignar a un esquema de almacenamiento único.
- Lenguaje de consulta integrada (*LINQ*) proporciona en tiempo de compilación validación de la sintaxis de consultas en un modelo conceptual.(10)

## 1.8 Tecnologías y herramientas utilizadas.

Para el desarrollo de un software es necesario utilizar algunas tecnologías y herramientas que permiten su creación y que suelen hacer el trabajo más fácil. Se utilizan herramientas para modelar, implementar y para el trabajo con bases de datos.

El entorno de desarrollo, metodología, lenguajes de programación, lenguajes de modelado y la herramienta CASE[18], definidos para la construcción del Subsistema Huésped Casero para la Sección de Enfrentamiento del Departamento Policía de Inmigración, fue producto de un estudio realizado por el equipo de arquitectura, y establecido como políticas del Proyecto Identificación, Migración y Extranjería de la República de Cuba, por lo que la selección del mismo queda fuera del alcance del presente trabajo.

En este trabajo de diploma solo se brinda una descripción de cada herramienta que se va a utilizar para desarrollar el Subsistema Huésped Casero para la Sección de Enfrentamiento del Departamento Policía de Inmigración.

### 1.8.1 La plataforma a utilizar.

#### 1.8.1.1 Plataforma .NET.

*.NET*[19] es un proyecto de *Microsoft* para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de *hardware* y que permite un rápido desarrollo de aplicaciones. Su objetivo es simplificar el desarrollo de aplicaciones *web*. Provee las herramientas y tecnologías para transformar a *Internet* en una plataforma de computación

[18] Herramienta CASE (*Computer Aided Engineering*): Ingeniería de asistida por ordenador.
[19] NET: Abreviación para "red" (*network*), ".net".

distribuida en gran escala. Además soporta los estándares sobre los cuales se basan los servicios *web* y utiliza tecnologías existentes, productos modificados para su uso dentro de la plataforma y elementos nuevos. (11)

*.NET* puede considerarse en ciertos aspectos como la respuesta de *Microsoft a Java*, aunque tiene bastantes diferencias. Reúne en una misma plataforma un conjunto interesante de características, como independencia de plataforma, independencia de lenguaje, soporte de bases de datos, soporte para *XML* (*Extensible Markup Language*), servicios *web* y aplicaciones *web*, entre otras. De entre todas estas, quizás una de las más importantes sean las dos primeras, referentes a la independencia tanto del lenguaje como de la plataforma.

- **Independencia del lenguaje.**

Se pueden desarrollar aplicaciones en múltiples lenguajes dentro de la plataforma *.NET*, pero lo más interesante, es que una aplicación puede tener diferentes partes desarrolladas en diferentes lenguajes, y todas estas pueden comunicarse entre sí, transparentemente, sin tener que utilizar ningún tipo de capa intermedia que posibilite esta comunicación. Esto permite a su vez una gran reutilización de código, ya que las clases desarrolladas para un proyecto en un lenguaje concreto, podrán ser reutilizadas en un nuevo proyecto, independientemente del lenguaje en el que se desarrolle éste.

- **Independencia de plataforma.**

Al igual que en *Java*, el código *.NET* no se compila a código máquina, sino a un código en un formato intermedio, independiente de la plataforma. Esto permite llevar los binarios producidos de una plataforma a otra, tal como sucede en *Java*. En la plataforma *.NET* existe el *Common Language Runtime (CLR)*[20], que se encarga de ejecutar el código intermedio o *Common Intermediate Language (CIL)*[21]. (12)

### 1.8.2 Servicios web.

Los servicios *web* son la revolución informática de la nueva generación de aplicaciones que trabajan colaborativamente, en las cuales el software está distribuido en diferentes servidores. (13) Un servicio *web* (en inglés *web service*) es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios *web* para intercambiar datos en redes de ordenadores como *Internet*. La interoperabilidad se consigue mediante la adopción de estándares abiertos. (14)

[20] *Common Language Runtime*: Lenguaje de ejecución común.
[21] *Common Intermediate Language*: Lenguaje intermedio común.

### 1.8.3 AJAX.

*Ajax* no es una tecnología. Es realmente muchas tecnologías, cada una floreciendo por su propio mérito, uniéndose en poderosas y nuevas formas.

#### 1.8.3.1 ¿Cómo es diferente AJAX?

Una aplicación *AJAX*[22] elimina la naturaleza "arrancar-frenar- arrancar-frenar" de la interacción en la *web* introduciendo un intermediario -un motor *AJAX*- entre el usuario y el servidor. Parecería que sumar una capa a la aplicación la haría menos reactiva, pero la verdad es lo contrario. En vez de cargar una página *web*, al inicio de la sesión, el navegador carga al motor *AJAX* (escrito en *Java Script* y usualmente "sacado" en un *frame*[23] oculto). Este motor es el responsable por renderizar la interfaz que el usuario ve y por comunicarse con el servidor en nombre del usuario.

El motor *AJAX* permite que la interacción del usuario con la aplicación suceda asincrónicamente (independientemente de la comunicación con el servidor). Así el usuario nunca estará mirando una ventana en blanco del navegador y un ícono de reloj de arena esperando a que el servidor haga algo. Cada acción de un usuario que normalmente generaría un requerimiento *HTTP* toma la forma de un llamado Java Script al motor *AJAX* en vez de ese requerimiento. Cualquier respuesta a una acción del usuario que no requiera un viaje de vuelta al servidor (como una simple validación de datos, edición de datos en memoria, incluso algo de navegación) es manejada por su cuenta. Si el motor necesita algo del servidor para responder (sea enviando datos para procesar, cargar código adicional, o recuperando nuevos datos) hace esos pedidos asincrónicamente, usualmente usando *XML*, sin frenar la interacción del usuario con la aplicación. (15)

### 1.8.4 Asp.NET.

*ASP.NET*[24] es usado por programadores para construir sitios *web* dinámicos, aplicaciones *web* y servicios *web XML*. Apareció en enero de 2002 con la versión 1.0 del *.NET Framework*, y es la tecnología sucesora de la tecnología *Active Server Pages (ASP). ASP.NET* está construido sobre el *Common Language Runtime*, permitiendo a los programadores escribir código *ASP.NET* usando cualquier lenguaje admitido por el *.NET Framework*. (16) Las páginas de *ASP.NET*, conocidas oficialmente como "*web forms*"[25], son el principal medio de construcción para el desarrollo de aplicaciones *web*. Los formularios *web* están contenidos en archivos con una extensión *ASPX*; en jerga

[22] AJAX (*Asynchronous JavaScript And XML*): *JavaScript* asíncrono y *XML*.
[23] *Frame*: Marco.
[24] *ASP (Active Server Pages)*: Servidor de páginas activas.
[25] Web forms: Formularios web.

de programación, estos archivos típicamente contienen etiquetas *HTML*[26] o *XHTML*[27] estático, y también etiquetas definiendo controles *web* que se procesan del lado del servidor y controles de usuario donde los desarrolladores colocan todo el código estático y dinámico requerido por la página *web*.

El modelo de *ASP.NET* marca la separación del *ASP* clásico y alienta a los desarrolladores a construir aplicaciones con la idea de presentación y contenido separados en mente. En teoría, esto permite a un diseñador *web*, por ejemplo, enfocarse en la creación del diseño con menos posibilidades de alterar el código de programación mientras lo hace. Las aplicaciones *ASP.NET* son alojadas en un servidor w*eb* y se tiene acceso a ellas mediante el protocolo sin estado *HTTP*[28], que no guarda ninguna información sobre conexiones anteriores. Por lo tanto, si la aplicación requiere interacción entre conexiones, tiene que implementar su propia administración del estado. *ASP.NET* proporciona varias maneras de administrar el estado de las aplicaciones *ASP.NET*. (17)

### 1.8.5 Windows Workflow Foundation (WWF).

*Windows Workflow Foundation* provee un único modelo unificado para crear soluciones de punta a punta que abarcan categorías de aplicaciones, incluyendo flujos de trabajo tanto humanos como de sistemas.

Un *workflow* (flujo de trabajo) es un modelo de proceso humano o de sistema que es definido como un mapa de actividades. Una actividad es un paso en un flujo de trabajo y es la unidad de ejecución. El mapa de actividades expresa reglas, acciones, estados y sus relaciones. Diseñadas las secuencias de actividades, un flujo de trabajo de *WWF* es entonces compilado en un *assembly .NET*, y es ejecutado sobre el *runtime* del flujo de trabajo y el *Common Language Runtime (CLR)*.

Esta tecnología de *WWF* complementa al *Framework .NET* con un grupo de componentes basados en flujos de trabajos que brindan a los desarrolladores la habilidad de definirlos, compilarlos, instanciarlos, depurarlos y rastrearlos.

Esta tecnología será parte de *WinFX* junto con *Windows Presentation Foundation (Avalon)* y *Windows Communication Foundation (Indigo)*.

Usando *WWF*, los desarrolladores pueden incorporar conceptos tales como *scheduling*, coordinación de tareas y escalabilidad en sus aplicaciones existentes sin costo alguno. *WWF* provee la plataforma base donde se pueden desarrollar aplicaciones con muchos procesos. (18)

[26] *HTML (HyperText Markup Language)*: Lenguaje de marcado de hipertexto.
[27] *XHTML (eXtensible HyperText Markup Language*): Lenguaje extensible de marcado de hipertexto.
[28] *HTTP (HyperText Transfer Protocol)*: Protocolo de transferencia de hipertexto.

### 1.8.6 Windows Communication Foundation (WCF).

*Windows Communication Foundation* (anteriormente conocida con el nombre en clave "Indigo") es un conjunto de tecnologías *.NET* para la creación y puesta en marcha de sistemas interconectados. Es una nueva generación de infraestructura de comunicaciones que gira en torno a la arquitectura de servicios *web*. El soporte para servicios *web* avanzados en *WCF* proporciona una mensajería segura, fiable y organizada en transacciones, además de interoperabilidad. El modelo de programación orientado a servicios de *WCF* se basa en *Microsoft .NET Framework* y simplifica el desarrollo de sistemas interconectados. *WCF* unifica una gran variedad de funcionalidades de sistemas distribuidos en una arquitectura organizable y extensible, que abarca transportes, sistemas de seguridad, patrones de mensajería, sistemas de codificación, topologías de red y modelos de alojamiento. Los desarrolladores pueden crear aplicaciones utilizando *WCF* en *Microsoft Visual Studio 2008* en su entorno de desarrollo integrado. (19)

### 1.8.7 Lenguajes de programación a utilizar.

Un lenguaje de programación es un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura, así como el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina.

Un lenguaje de programación es una construcción mental del ser humano para expresar programas. Está constituido por un grupo de reglas gramaticales, un grupo de símbolos utilizables, un grupo de términos monosémicos (es decir, con sentido único) y una regla principal que resume las demás. Para que ésta construcción mental sea operable en un computador debe existir otro programa que controle la validez o no de lo escrito. A éste se le llama traductor.

Los lenguajes pueden ser de alto o bajo nivel. En los de bajo nivel las instrucciones son simples y cercanas al funcionamiento de la máquina, como por ejemplo el código máquina y el ensamblador. En los lenguajes de alto nivel hay un alto grado de abstracción y el lenguaje es más próximo a los humanos, como por ejemplo Léxico, PASCAL, Cobol o Java. (20)

#### 1.8.7.1 C Sharp.

C# (pronunciado si *sharp* en inglés) es un lenguaje de programación orientado a objetos desarrollado y estandarizado por *Microsoft* como parte de su plataforma *.NET*, que después fue aprobado como un estándar por la *ECMA*[29] e *ISO*[30]. Su sintaxis básica deriva de *C/C++* y utiliza el modelo de objetos de la plataforma *.NET* el cual es similar al de *Java* aunque incluye mejoras derivadas de otros lenguajes

[29] ECMA: Empresa certificadora medioambiental.
[30] ISO: Organización internacional para la estandarización.

(entre ellos *Delphi*). Aunque este lenguaje forma parte de la plataforma.*NET*, resulta de programación independiente diseñado para generar programas sobre dicha plataforma. (21)

#### 1.8.7.2 Java Script.

*Java Script* es un lenguaje de *scripting* basado en objetos, utilizado para acceder a objetos en aplicaciones. Principalmente, se utiliza integrado en un navegador *web* permitiendo el desarrollo de interfaces de usuario mejoradas y páginas *web* dinámicas. *Java Script* se caracteriza por ser un lenguaje basado en prototipos, con entrada dinámica y con funciones de primera clase. Ha tenido influencia de múltiples lenguajes y se diseñó con una sintaxis similar al lenguaje de programación *Java*, aunque más fácil de utilizar para personas que no programan. Todos los navegadores modernos interpretan el código *Java Script* integrado dentro de las páginas *web*. *Java Script* se ejecuta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código *HTML (HyperText Markup Language).* (22)

#### 1.8.7.3 HyperText Markup Language (HTML).

HTML, siglas de *HyperText Markup Language* (Lenguaje de Marcas de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas *web*. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. *HTML* se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). (23)

*HTML* también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un *script* (por ejemplo *Java script*), el cual puede afectar el comportamiento de navegadores *web* y otros procesadores de *HTML*. Por convención, los archivos de formato *HTML* usan la extensión *.htm o .html*.

#### 1.8.7.4 CSS (Cascading Style Sheets).

CSS[31] es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con *HTML* y *XHTML.* CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas *web* complejas. Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos bien definidos y con significado completo (también llamados "documentos semánticos"*).* Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. La información de estilo puede ser adjuntada tanto como un documento separado o en el mismo

[31] *CSS (*Cascading Style Sheets*)*: Hojas de estilo en cascada.

documento *HTML*. En este último caso podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo "*style*". (24)

### 1.8.8 Microsoft Visual Studio Team System 2008.

*Visual Studio* es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones *web ASP.NET*, servicios *web XML* y aplicaciones de escritorio.

Ventajas:

- Facilita la creación rápida de aplicaciones interconectadas.
- Se incluyen todas las funciones de las ediciones de *Visual Studio 2008 Team*, además del contenido en *Visual Studio 2008 Professional Edition*.
- Contiene herramientas de diseño de arquitectura que le ayudan a centrarse en la mejora del diseño y la validación de sistemas distribuidos, herramientas de desarrollo para identificar códigos no eficientes, no seguros o de poca calidad.
- Permite especificar procedimientos recomendados de codificación y automatizar las pruebas de unidades.
- Contiene herramientas de prueba para modificar, ejecutar y administrar pruebas y elementos de trabajo relacionados con *Visual Studio*.
- Posee herramientas de desarrollo de base de datos para la administración de cambios de la base de datos y la realización de pruebas para aumentar la calidad del nivel de los datos.

Desventajas:

- Es un software propietario.
- Diseñado para soluciones corporativas.
- Exige más *hardware*.

Se selecciona como herramienta para el desarrollo *Visual Studio Team System 2008* ya que posibilita la creación rápida de aplicaciones interconectadas, y además por ser la establecida por líderes y arquitectos para el desarrollo del proyecto. (25)

## 1.9 Herramientas para bases de datos.

### 1.9.1 Sistema Gestor de Bases de Datos.

#### 1.9.1.1 ¿Por qué usar Oracle Database 11g?

La última versión de *Oracle* es la versión 11g, cuenta con administración de usuarios así como la administración de roles, además soporta *triggers*[32] y *store procedure*[33], cuenta con conectividad JDBC[34] y ODBC[35], siempre y cuando se tengan los *drivers*[36] adecuados para la misma. Es un DBMS[37] seguro ya que cuenta con un proceso de sistema de respaldo y recuperación de información. Soporta *Data Warehouse*[38] por lo que facilita el acceso a la información y da mayor versatilidad. La mayor parte de las empresas de telecomunicaciones en Latinoamérica utilizan *Oracle*, por lo que se puede decir que es un DBMS confiable, seguro para ser utilizado en una empresa y sobre todo permite reducir costos por su accesibilidad en el mercado. Se considera a *Oracle* como uno de los sistemas de bases de datos más completos, destacando su:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Soporte multiplataforma.

La base de datos *Oracle* en w*indows* ha evolucionado desde un nivel básico de integración del sistema operativo hasta utilizar servicios más avanzados en la plataforma w*indows*. Como siempre, *Oracle* continúa innovando y aprovechando las nuevas tecnologías de *windows*. (26)

### 1.9.2 Herramienta de modelado de bases de datos, Erwin Studio.

Erwin Studio es una herramienta de modelado de datos fácil de usar y multinivel, es usada para el diseño y construcción de bases de datos a nivel físico y lógico. Direcciona las necesidades diarias de los administradores de bases de datos, desarrolladores y arquitectos de datos que construyen y mantienen aplicaciones de bases de datos grandes y complejos. *ER/Studio* está equipado para crear y

[32] *Triggers:* Es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).
[33] *Store procedure:* O procedimiento almacenado, es un programa que es almacenado físicamente en una base de datos.
[34] JDBC (*Java Database Connectivity*): Conectividad de Java para Bases de datos.
[35] ODBC (*Open Database Connectivity*): Conectividad abierta para Bases de datos.
[36] *Drivers:* Es un programa que permite al sistema operativo interactuar con un periférico, haciendo una abstracción del hardware y proporcionando una interfaz.
[37] DBMS: Sistemas de Gestión de Bases de datos.
[38] *Data Warehouse:* un almacén de datos (del inglés data warehouse) es una colección de datos orientada a un determinado ámbito (empresa u organización), integrado, no volátil y variable en el tiempo, que ayuda a la toma de decisiones en la entidad en la que se utiliza.

manejar diseños de bases de datos funcionales y confiables. Ofrece fuertes capacidades de diseño lógico, sincronización bidireccional de los diseños físicos y lógicos, construcción automática de bases de datos, documentación y fácil creación de reportes. (27)

### 1.9.3 Language Integrated Query (LINQ).

*Language-Integrated Query (LINQ)* es un conjunto de características en *Visual Studio 2008* que agrega eficaces capacidades de consulta a la sintaxis de los lenguajes *C# y Visual Basic*. LINQ incluye patrones estándar y de fácil aprendizaje para consultar y actualizar datos, y su tecnología se puede extender para utilizar potencialmente cualquier tipo de almacén de datos. *Visual Studio 2008* incluye ensamblados de proveedores para LINQ que habilitan el uso de LINQ con colecciones de *.NET Framework*, bases de datos de *SQL Server*, conjuntos de datos de *ADO.NET* y documentos *XML (eXtensible Markup Language).* (28)

## 1.10 Conclusiones parciales.

En este capítulo se describe una serie de elementos que servirán como punto de partida para la construcción de un subsistema que cumpla con los objetivos propuestos. La utilización de la metodología facilita el seguimiento y monitoreo de un proceso de desarrollo que agilice el trabajo de los implicados en el mismo, de ahí que la obtención de un producto con la calidad requerida sea un hecho seguro. La utilización de las herramientas y las definiciones arquitectónicas mencionadas permitirá el desarrollo claro y fluido de un subsistema construido sobre bases sólidas y un entorno de desarrollo bien preciso.

# CAPÍTULO 2: CARACTERÍSTICAS DEL SUBSISTEMA

## 2.1 Introducción.

En este capítulo se realiza un análisis y modelado del negocio, además se determinan las necesidades del subsistema que se quiere desarrollar, mejorando los procesos más importantes y especificando estas necesidades mediante los requerimientos funcionales y no funcionales. Por lo que resulta de gran importancia conocer la definición de negocio.

### Definición de negocio:

El negocio es cualquier ambiente o entorno en el cual está enmarcado el problema; y los procesos de negocio son las funciones que se desarrollan en el ambiente o entorno que definimos.

Durante el negocio se debe cumplir con los siguientes objetivos:

- Comprender la estructura y la dinámica de la organización en la cual se va a implantar un sistema.
- Comprender los problemas actuales de la organización e identificar las mejoras potenciales.
- Asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización.
- Derivar los requerimientos del sistema que va a soportar la organización. (29)

## 2.2 Descripción del flujo actual de los procesos.

- **Subproceso Gestión de estancia.**

En las Figuras 2 y 3 se muestra el diagrama de proceso del negocio para el subproceso Gestión de estancia dividido en dos secciones, pues resulta el más importante en el desarrollo del Subsistema Huésped Casero para la Sección de Enfrentamiento del Departamento Policía de Inmigración, el mismo se encuentra asociado a la siguiente descripción, las descripciones y digramas de los demás procesos identificados se encuentran en el Anexo 2:

| | |
|---|---|
| **Objetivo:** | Registrar una estancia. |
| **Eventos que lo generan:** | Puede comenzar por dos eventos diferentes; cuando un casero va a reportar la estancia de extranjeros alquilados en su vivienda a un Inspector de Enfrentamiento o cuando una persona entra al país y se crea un vínculo familiar mediante un funcionario de Extranjería, también puede presentarse un extranjero |

| | |
|---|---|
| | con Visa consular a realizar un registro de estancia a través de un funcionario de Extranjería |
| **Responsable:** | Inspector de Enfrentamiento y un funcionario de Extranjería. |
| **Clientes:** | Caseros acogidos al Decreto Ley 171/97 y las personas con Visa familiar. |
| **Descripción:** | Para el casero que va a reportar la estancia de extranjeros alquilados en su vivienda es obligatorio presentar los siguientes documentos para que el proceso pueda comenzar:<br>• Libro de registro de arrendatario.<br>• Visa-Tarjeta del turista.<br>• Información sobre el casero, huésped y acompañante.<br>Luego de haber reportado la estancia el casero se marcha con el libro de registro de arrendatario revisado y acuñado. |

**Tabla 2.1. Descripción del subproceso Gestión de estancia.**

**(Sección I)**

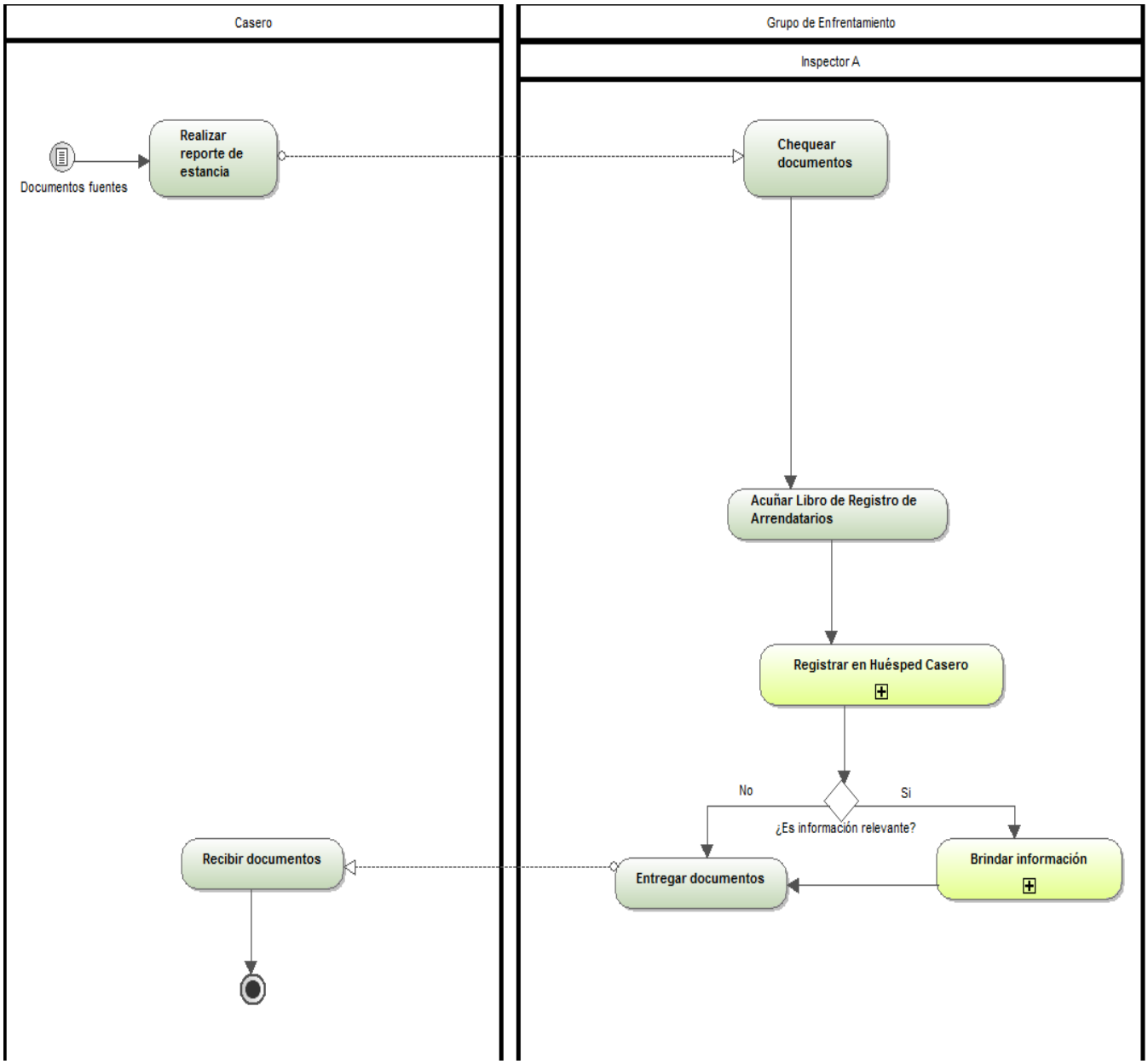



**Figura. 2 Diagrama de flujo del subproceso Gestionar estancia (Sección I)**

**(Sección II)**

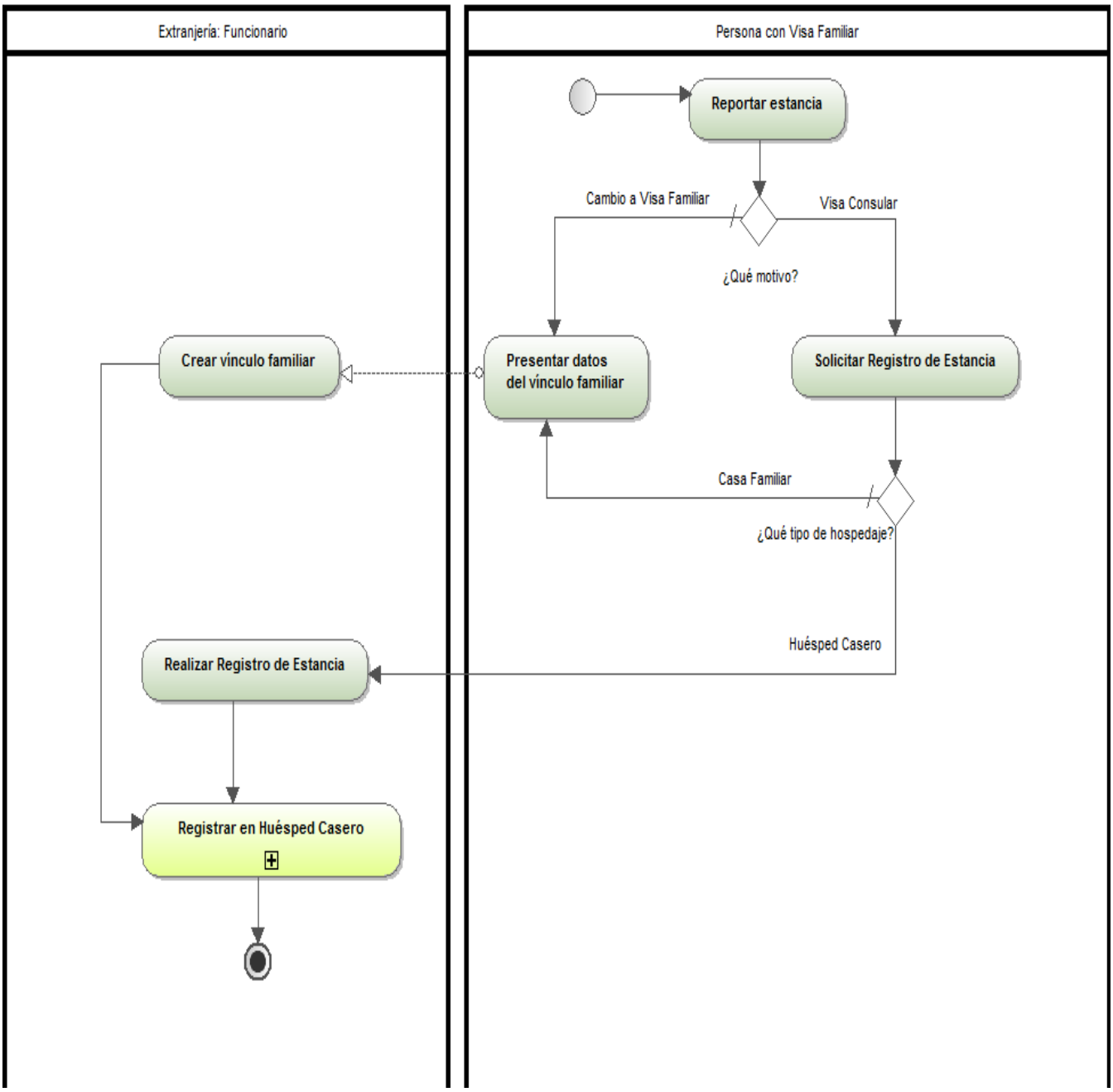



**Figura. 3 Diagrama de flujo del subproceso Gestionar estancia (Sección II)**

## 2.3 Reglas del negocio.

Las reglas del negocio o conjunto de reglas de negocio describen las políticas, normas, operaciones, definiciones y restricciones presentes en una organización. Se tienen en cuenta a la hora de modelar el negocio. A continuación se clasifican y describen los tipos de reglas de negocio que están presentes en el negocio según las necesidades específicas del subsistema que se desarrolla. De cada regla se detalla el nombre y una descripción que complemente en qué consiste la regla.

- **Reglas textuales.**

Contienen "instrucciones", se expresan de forma libre (no estructurada) en lenguaje natural.

- **Reglas del modelo de datos.**

Engloba todas aquellas reglas que se encargan de controlar que la información básica almacenada para cada atributo o propiedad de una entidad u objeto sea válida.

- **Reglas de relación.**

Incluye todas aquellas reglas que controlan las relaciones entre los datos.

Especifican, por ejemplo, que todo pedido debe ser realizado por un cliente, al que se debe atender. Además, una vez que un cliente haya hecho algún pedido, se debe garantizar que no es posible eliminar el pedido, a menos que previamente se eliminen todos los que haya realizado anteriormente.

- **Reglas de derivación.**

Es frecuente que a partir de cierta información se pueda derivar otra, este conjunto de reglas especifican y controlan la obtención de información que se puede calcular a partir de la ya existente.

- **Reglas de acción.**

Las reglas de acción pueden describir como fluyen los datos en un proceso, restricciones que deben tener las operaciones y respuestas ante determinados estímulos que deben tenerse en cuenta.

- **Reglas de flujo de datos.**

Determinan y limitan como fluye la información a través del subsistema.

- **Reglas de restricción de operaciones.**

Especifican condiciones que deben ser ciertas para que una determinada operación se ejecute correctamente.

La relación de reglas del negocio se muestra en el Anexo 3.

## 2.4 Diagramas de procesos mejorados.

Teniendo en cuenta cómo se desarrolla el proceso actual y los resultados que se quieren alcanzar, se identificaron mejoras que se reflejan en los diagramas de procesos mejorados. A continuación en la Figura 4 se muestra el diagrama del proceso Buscar persona que es similar al proceso Buscar casero que se encuentra en el Anexo 5 pero resulta más general, en la Figura 5 se muestra el diagrama del proceso Gestionar estancia para una persona con Visa familiar y en la Figura 6 se muestra el diagrama del proceso Gestionar estancia para un huésped casero pues son los procesos en los que se centra el estudio.

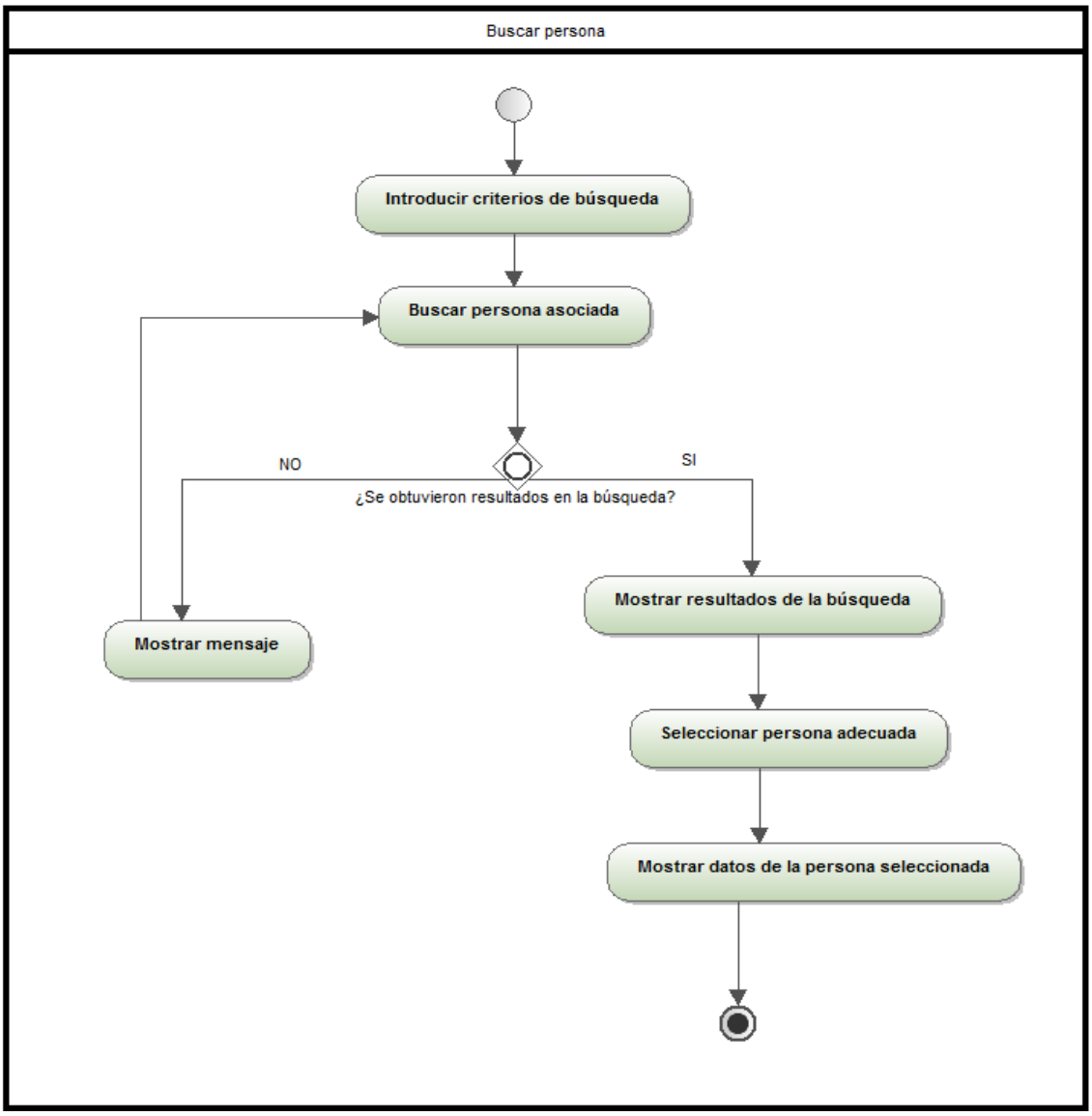



**Figura. 4 Diagrama del proceso Buscar persona.**

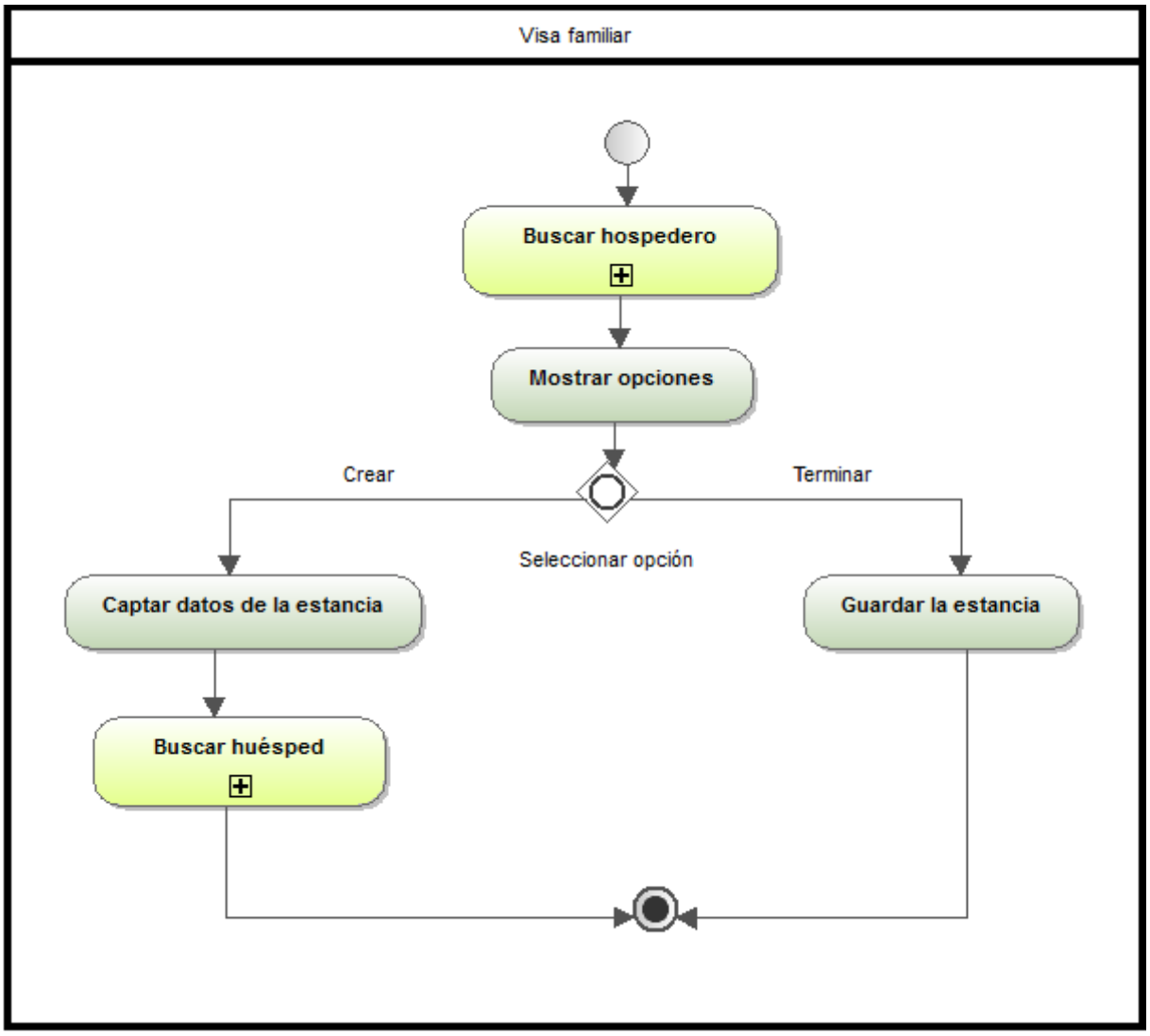



**Figura. 5 Diagrama del proceso Gestionar estancia para una persona con Visa familiar.**

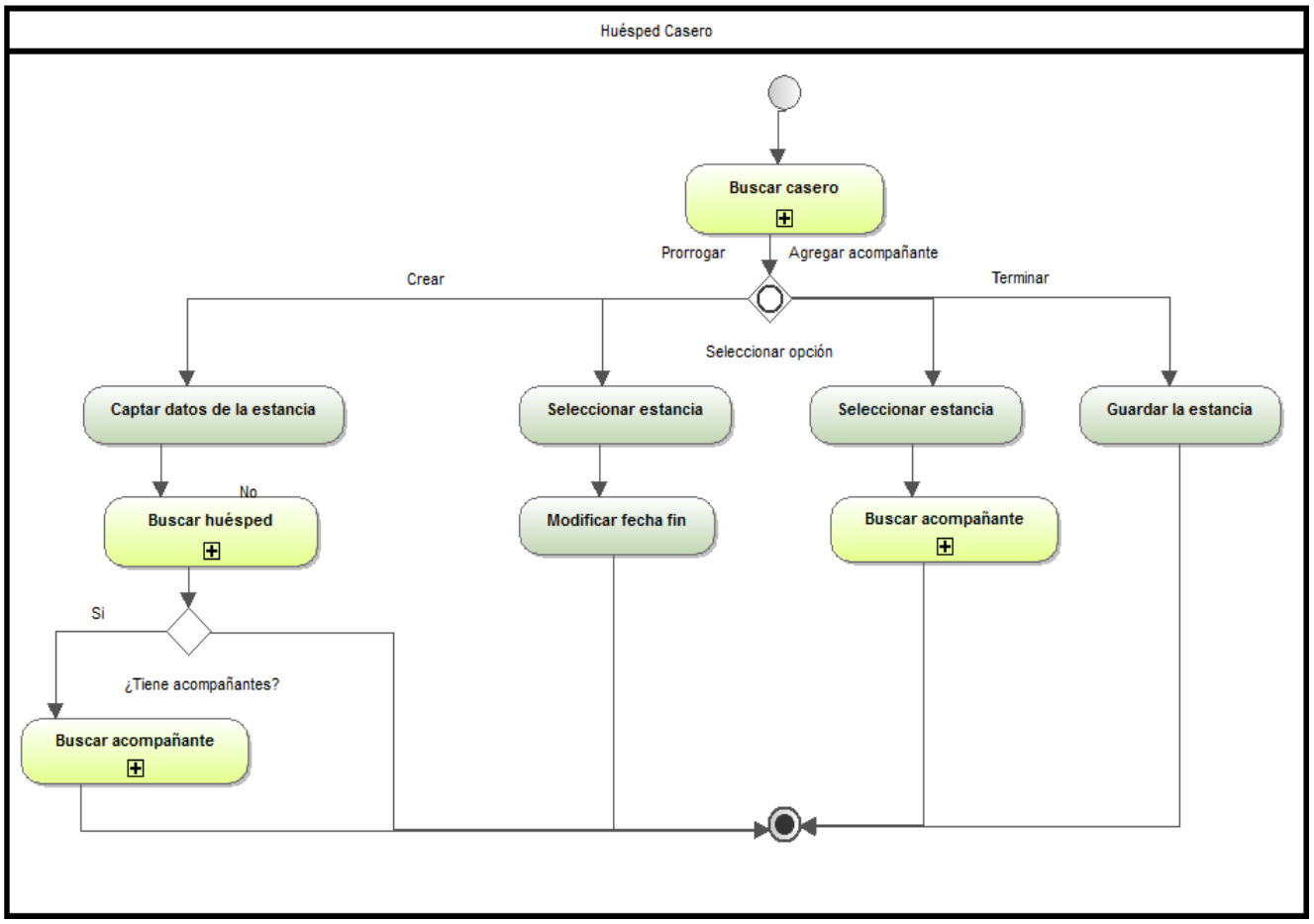



**Figura. 6 Diagrama del proceso Gestionar estancia para un huésped casero.**

## 2.5 Descripción del subsistema.

El subsistema permite la búsqueda de personas, ya sean nacionales o extranjeros, también garantiza buscar los caseros acogidos al Decreto Ley 171/97, y gestionar las estancias, ya sean mediante una visa familiar o mediante un hospedaje en viviendas de caseros acogidos al Decreto Ley 171/97, que se realicen en las áreas de atención llevadas por la Sección de Enfrentamiento del Departamento Policía de Inmigración.

## 2.6 Descripción de los módulos y procesos del subsistema.

El Subsistema Huésped Casero para la Sección Enfrentamiento del Departamento Policía de Inmigración, está integrado por los siguientes módulos:

- **Módulo Huésped Casero**.

- Buscar casero: Permite buscar caseros.
- Gestionar estancia para un huésped casero: Permite crear, modificar y terminar las estancias para los caseros.

- **Módulo Visa familiar.**
  - Buscar persona: Permite la búsqueda de personas, que pueden ser nacionales o extranjeros.
  - Gestionar estancia para una Visa familiar: Permite crear y terminar las estancias para las personas con Visa familiar.

## 2.7 Descripción de los roles.

En la Tabla 2.2, se realiza una descripción de los roles implicados en el negocio.

| Rol | Objetivo |
|---|---|
| Inspector del Enfrentamiento | Es el encargado de gestionar las estancias y realizar la búsqueda de personas, ya sean nacionales o extranjeros, que pertenezcan a su área de atención. |
| Jefe del Grupo de Enfrentamiento | Puede gestionar las estancias y realizar la búsqueda de personas, ya sean nacionales o extranjeros, además de observar todo el trabajo realizados por los inspectores subordinados a él, pero no puede modificar nada, generalmente los resultados que observa son estadísticas de las áreas. |
| Jefe de la Sección Enfrentamiento | Puede ver las estancias realizadas por todos los grupos de enfrentamiento, subordinados a él, pero no puede modificar nada, generalmente los resultados que observa son estadísticas de las áreas. |
| Jefe del Departamento Policía de Inmigración | Puede ver las estancias generadas por toda la Sección Enfrentamiento, generalmente los resultados que observa son estadísticas de las áreas. |
| Administrador | Puede acceder a todas las opciones y es el encargado de crear los diferentes usuarios y darle los permisos de acuerdo a su nivel de acceso a la información. |

**Tabla 2.2. Descripción de los roles.**

## 2.8 Especificación de los requisitos de software.

Un requerimiento es una necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio. Condición o capacidad que un usuario necesita para poder resolver un problema o lograr un objetivo. Condición o capacidad que debe exhibir o poseer un sistema para satisfacer un contrato, estándar, especificación, u otra documentación formalmente impuesta. Una condición o capacidad que debe ser conformada por el sistema. Algo que el sistema debe hacer o una cualidad que el sistema debe poseer. (29)

### 2.8.1 Definición de los requisitos funcionales.

Los requerimientos funcionales especifican acciones que el subsistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física. Es decir, especifican el comportamiento de entrada y salida del subsistema y surgen de la razón fundamental de la existencia del producto. (30)

A continuación se listan las funcionalidades en forma de requisitos funcionales del subsistema:

RF.1 Buscar persona.

RF.2 Buscar casero.

RF2.1 Mostrar estancias activas.

RF.3 Gestionar estancia para un huésped casero.

RF3.1 Insertar estancia para un huésped casero.

RF3.2 Modificar estancia para un huésped casero.

RF3.2.1 Prorrogar estancia.

RF3.2.2 Agregar acompañante.

RF3.3 Terminar estancia para un huésped casero.

RF3.3.1 Terminar estancia por arribo de la fecha de fin.

RF3.3.2 Terminar estancia por reporte del casero.

RF.4 Gestionar estancia para una persona con Visa familiar.

RF4.1 Insertar estancia para una persona con Visa familiar.

RF4.2 Terminar estancia para una persona con Visa familiar.

RF4.2.1 Terminar estancia por arribo de la fecha de fin.

RF4.2.2 Terminar estancia por reporte del casero.

### 2.8.2 Descripción de los requisitos funcionales.

A continuación en las Tablas 2.3 y 2.4 respectivamente se muestran las descripciones del RF1. Buscar persona y del RF 4. Gestionar estancia de una persona con Visa familiar, porque son las descripciones

que reflejan a las demás descripciones de los requisitos funcionales; en el interior de las descripciones mencionadas en las Figuras 7 y 8 respectivamente se encuentran los prototipos no funcionales asociados a las mismas. Las descripciones de los demás requisitos funcionales se encuentran en el Anexo 4.

RF 1. Buscar persona.

| | | |
|---|---|---|
| **Propósito** | Permite buscar personas a partir de una información determinada. | |
| **Roles** | Inspector de Enfrentamiento | |
| **Precondiciones** | El usuario debe estar autenticado en el subsistema. | |
| **Entidades tratadas** | **Entidad** | **Atributos** |
| | Persona | Ver entidades tratadas |
| | Inspector de Enfrentamiento | Ver entidades tratadas |
| **Descripción** | 1.1 Seleccionar la opción “Buscar Persona”.<br>1.2 Mostrar los campos para realizar la búsqueda.<br>a. Nombre(s).<br>b. Apellidos(s).<br>c. Tipo de documento de identificación.<br>d. Nacionalidad.<br>e. Número de documento.<br>f. Raza.<br>g. Sexo.<br>h. Fecha de nacimiento.<br>i. Fecha de entrada a Cuba.<br>1.3 Insertar los criterios de búsqueda.<br>1.4 Mostrar la opción “Buscar”<br>1.5 Mostrar los resultados de la búsqueda en caso de ser positiva.<br>1.5.1 Seleccionar la persona deseada.<br>1.5.2 Mostrar los datos de la persona seleccionada.<br>a. Nombre(s)<br>b. Apellido(s)<br>c. Ciudadanía<br>d. Nacionalidad<br>e. Tipo de documento<br>f. No. documento<br>g. Fecha de nacimiento | |

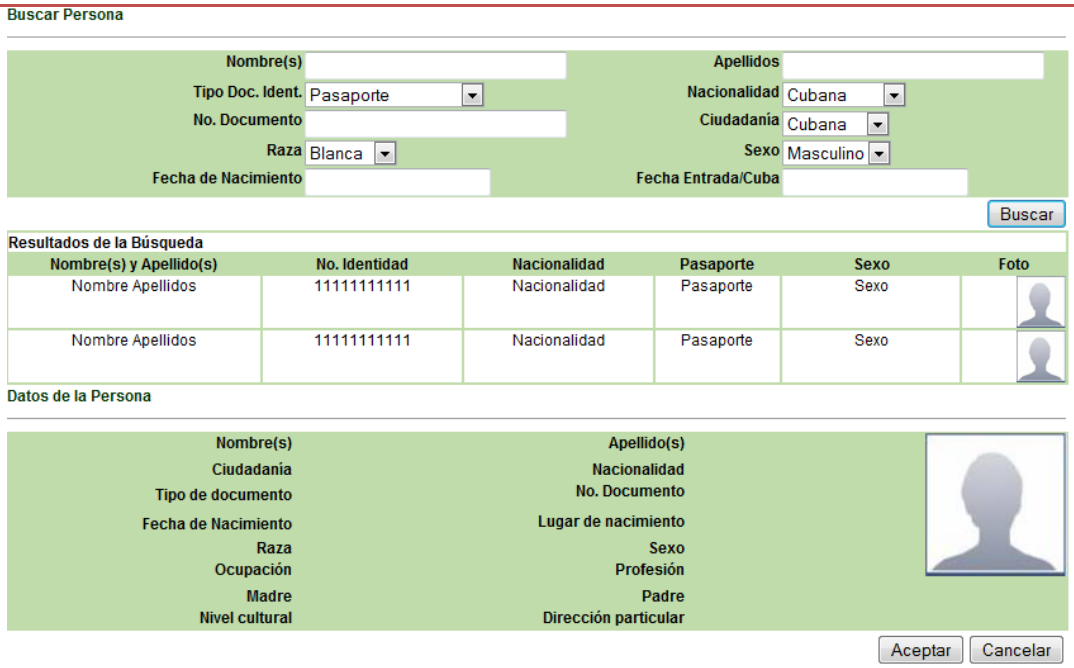
| | |
|---|---|
| | h. Lugar de nacimiento<br>i. Raza<br>j. Sexo<br>k. Ocupación<br>l. Profesión<br>m. Padre<br>n. Madre<br>o. Nivel cultural<br>p. Dirección particular<br>q. Foto<br>1.6 Mostrar mensaje informativo si el resultado de la búsqueda es negativo.<br>1.7 Seleccionar la opción “Aceptar” si se desea.<br>1.8 Seleccionar la opción “Cancelar” si se desea. |
| **Validaciones** | Para realizar las operaciones:<br>1. Ver validaciones de las entidades tratadas en el Diccionario de datos. |
| **Postcondiciones** | 1. Se seleccionó una persona.<br>2. Se mostraron los datos de la persona. |
| **Prototipo** |  |

**Figura. 7 Prototipo no funcional del proceso Buscar persona.**

**Tabla 2.3. Descripción del RF 1.Buscar persona.**

RF 4. Gestionar estancia para una persona con Visa familiar.

<table>
<tr><td>Propósito</td><td colspan="2">Permite gestionar la estancia para una persona con Visa familiar.</td></tr>
<tr><td>Roles</td><td colspan="2">Inspector de Enfrentamiento.</td></tr>
<tr><td>Precondiciones</td><td colspan="2">1. El usuario debe estar autenticado en el subsistema.<br>2. La búsqueda de la información de las personas de interés para la estancia, debe estar realizada.</td></tr>
<tr><td rowspan="5">Entidades tratadas</td><td>Entidad</td><td>Atributos</td></tr>
<tr><td>Persona</td><td>Ver entidades tratadas</td></tr>
<tr><td>Estancia</td><td>Ver entidades tratadas</td></tr>
<tr><td>Casero</td><td>Ver entidades tratadas</td></tr>
<tr><td>Huésped</td><td>Ver entidades tratadas</td></tr>
<tr><td>Descripción</td><td colspan="2">4.1 Buscar hospedero, ver RF 1.<br>4.2 Mostrar las opciones “Crear”, “Terminar”.<br>4.2.1 Seleccionar la opción “Crear”.<br>4.2.1.1 Captar datos de la estancia.<br>a. Fecha de inicio.<br>b. Fecha de fin.<br>c. Tipo de vínculo.<br>4.2.1.2 Buscar huésped, ver RF 1.<br>4.2.1.3 Mostrar los opciones “Aceptar”,” Cancelar”.<br>4.2.1.3.1 Seleccionar “Aceptar”, para registrar la estancia.<br>4.2.1.3.2 Seleccionar “Cancelar”, para no registrar la estancia.<br>4.2.2 Seleccionar la opción “Terminar”.<br>4.2.2.1 Guardar estancia.<br>4.2.2.2 Finalizar estancia.</td></tr>
</table>

| | |
|---|---|
| **Validaciones** | Para realizar las operaciones:<br>1. Ver validaciones de las entidades tratadas en el Diccionario de datos. |
| **Postcondiciones** | 1. Debe quedar registrada la información de la estancia para una persona con visa familiar. |
| **Prototipo** |  <br>**Figura 8. Prototipo no funcional del proceso Gestionar estancia para una persona con Visa familiar.** |

**Tabla 2.4. Descripción del RF 4. Gestionar estancia para una persona con Visa familiar.**

### 2.8.3 Definición de los requisitos no funcionales.

Los requerimientos no funcionales son las propiedades o cualidades que el producto debe tener para que este sea atractivo, usable, rápido y confiable. (30)

A continuación se describe uno de los requisitos no funcionales, en este caso el de la usabilidad, porque es en el que se explica como se debe usar el subsistema, quien lo puede usar entre otras características. El resto de los requisitos no funcionales se encuentran en el Anexo 4.

**Usabilidad**

RnF1. Nivel:

El subsistema puede ser utilizado por cualquier tipo de usuario con conocimientos básicos de computación.

RnF2. Acceso:

El subsistema debe permitir la conexión concurrente, por sus características de , desde todos los puestos de trabajo de la Sección de Enfrentamiento.

RnF3. Validaciones:

El subsistema debe validar que la información introducida sea correcta. Además debe validarse que el tipo de dato se corresponda con lo previsto para el campo.

RnF4. Especificación de la terminología utilizada:

El subsistema debe adaptarse al lenguaje y términos utilizados por los clientes en la rama abordada con vista a una mayor comprensión por parte del cliente de la herramienta de trabajo.

RnF5. Perfiles de usuario**:**

Diferenciar las interfaces y opciones para los usuarios que accedan al subsistema con diferentes roles.

RnF6. Menús:

El subsistema debe presentar una serie de menús tanto laterales como desplegables que permitan el acceso rápido a la información por parte de los usuarios, aprovechando así las potencialidades de estas estructuras.

## 2.9 Entidades conceptuales.

En la Tabla 2.5, se realiza una descripción de las entidades conceptuales.

| Conceptos | Atributos |
|---|---|
| **Viviendas acogidas al Decreto Ley 171/97** | • Consejo popular<br>• Número<br>• Número de licencia<br>• Nombres y apellidos<br>• Dirección particular<br>• Advertida<br>• Multada<br>• Teléfono<br>• Observación |
| **Casero** | • Total habitaciones<br>• Licencia<br>• Estado licencia<br>• Caracterizaciones |

| | |
|---|---|
| | • Descripción de la vivienda |
| **Estancia** | • Hospedero<br>• Huésped<br>• Visa familiar<br>• Listado acompañantes<br>• Fecha inicio<br>• Fecha fin<br>• Oficial que registró<br>• Fecha de registro |
| **Persona** | • Nombre y apellidos<br>• Documentos de identificación<br>  ○ Número de identificación<br>  ○ Pasaporte<br>  ○ Código de identificación(C/I)<br>  ○ Tipo de Visa<br>• Tipo de persona (Residentes temporales, Residentes permanentes, Refugiados, Secuestradores de vuelos, Cubanos residentes en el exterior)<br>• Edad<br>• Sexo (Masculino, Femenino)<br>• Fecha de nacimiento<br>• Lugar de nacimiento<br>• Nacionalidad<br>• Ciudadanía<br>• Raza<br>• Complexión<br>• Foto (3)<br>• Huellas<br>• Dirección particular.<br>• Estancia en Cuba<br>• Ocupación.<br>• Profesión<br>• Madre |

| | |
|---|---|
| | • Padre<br>• Nivel cultural (Primario, Pre. universitario, Universitario, Secundario, Técnico medio, Sin instrucción) |
| **Inspector de Enfrentamiento** | • Grupo<br>• Cargo en el grupo(Jefe del grupo, Inspector)<br>• Área<br>• Fuerzas auxiliares<br>• Incidencias |

**Tabla 2.5. Descripción de las entidades conceptuales.**

## 2.10 Conclusiones parciales.

El análisis y modelado de los procesos del negocio así como las reglas que definen el flujo de los mismos permitió alcanzar un amplio dominio del negocio analizado, teniéndolo como punto de partida para determinar los requisitos funcionales y no funcionales.

# CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SUBSISTEMA

## 3.1 Introducción.

En este capítulo se hace especial énfasis en una de las etapas más importantes en la vida de desarrollo de un software; pues se describe la arquitectura de la solución y las diferentes capas por las que está integrada, además se determina como debe ser el subsistema, realizando su análisis y diseño; para lo que se especifican las clases, así como los principales servicios. Se presenta el diseño de los flujos de trabajo y el diagrama de las clases entidades.

## 3.2 Arquitectura de la solución.

La arquitectura de software es el diseño de más alto nivel de la estructura de un sistema, donde se establecen los fundamentos para que analistas, diseñadores y programadores, trabajen en una línea común que permita alcanzar los objetivos del subsistema de información, cubriendo todas las necesidades. El subsistema se encuentra basado en una solución cliente servidor, desarrollada en la plataforma *.Net*, con el *.Net Framework 3.5*. El núcleo principal del servidor de aplicaciones es un motor de procesos que utiliza la tecnología *Workflow Foundation* que centra su funcionamiento en una arquitectura orientada a servicios.

El subsistema en su vista más abstracta es una solución Cliente - Servidor que se caracteriza por que describe la estructura tradicional de los sistemas distribuidos, y sus principales componentes son: un conjunto de servidores locales que ofrecen servicios a otros subsistemas, un conjunto de clientes que invocan los servicios ofrecidos por los servidores y una red que permite que los clientes accedan a los servicios.

En el subsistema se puede ver el uso del estilo de llamada y retorno, haciendo uso del patrón arquitectónico Modelo Vista Controlador, donde las peticiones del usuario son atendidas por un controlador y este desencadena un flujo guiado por el flujo de trabajo, definiendo la siguiente vista a mostrar, y esta a su vez posee los mecanismos para visualizar el estado del modelo. El uso del patrón arquitectura en capas permite que puedan asignar correctamente las responsabilidades a cada una de ellas, evitando que cambios en una de estas afecte directamente al resto. El subsistema se desarrolla bajo el paradigma arquitectónico de orientación a servicios, la Arquitectura Orientada a Servicios (SOA) es un paradigma arquitectónico que se basa en procesos de negocio y sistemas heterogéneamente distribuidos de diferente procedencia, está basado fundamentalmente en un modelo en el que los nodos son expuestos como servicios independientes o autónomos que contienen

un conjunto de funcionalidades técnicas, cuya principal característica es que son reutilizables por cualquier sistema o componente.

### 3.2.1 Capas de la vista lógica de la arquitectura.

La arquitectura se encuentra representada por cinco capas lógicas que dan un alto nivel de encapsulamiento de las responsabilidades, permitiendo reducir al máximo el acoplamiento y aumentar la reutilización entre las mismas. Esta distribución permite que se realicen grandes cambios sin siquiera tener que modificar en las demás capas. Una vez que estas estén bien definidas la comunicación entre ellas se realizará solo a nivel de interfaces que permiten trabajar de manera transparente a las instancias reales.

En la Figura 9 se muestra la Vista lógica de la arquitectura de la solución.

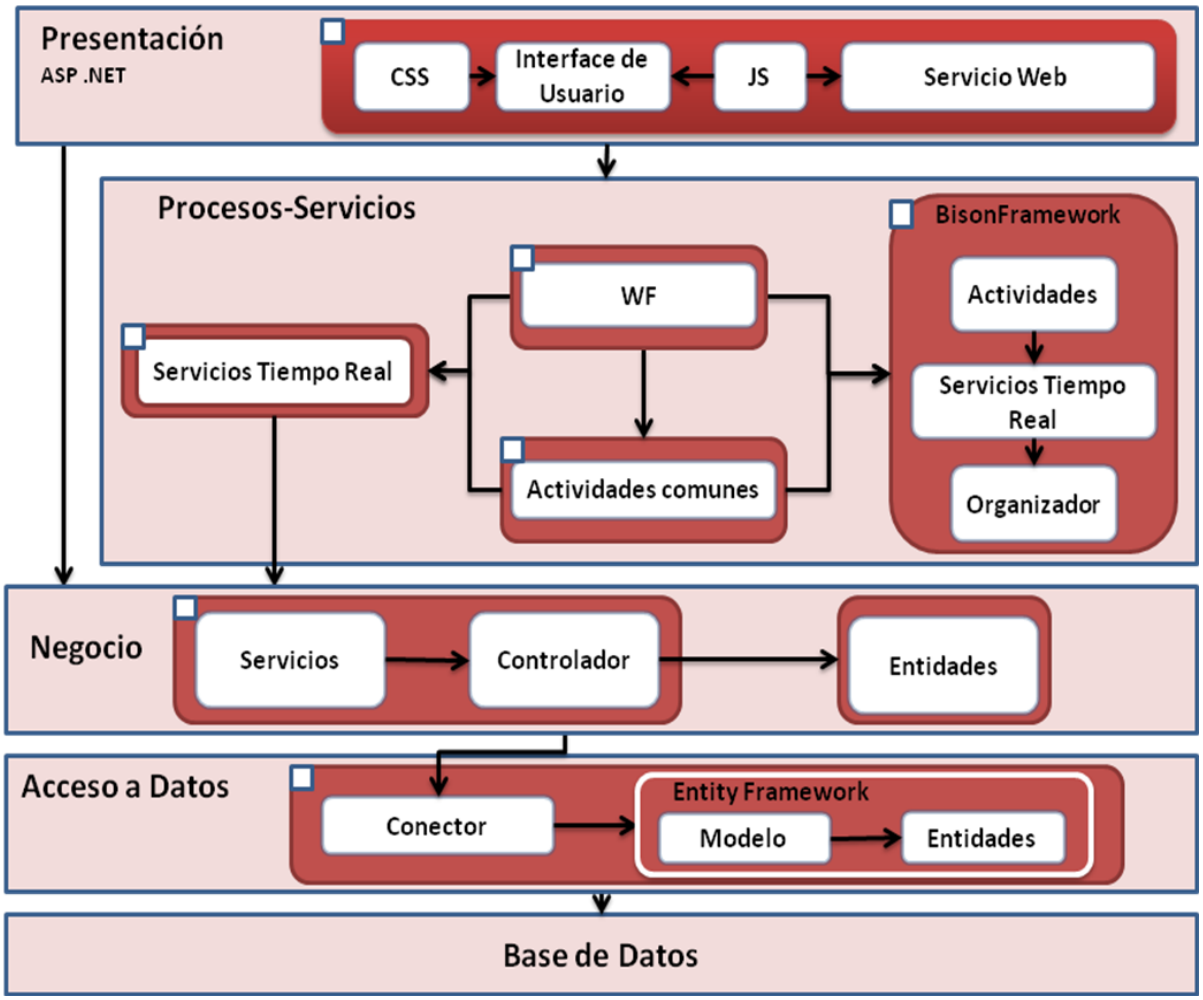



**Figura 9. Capas de la vista lógica de la arquitectura de la solución.**

#### 3.2.1.1 Capa presentación (Presentation Layer):

Es la interfaz de comunicación de la aplicación con un usuario determinado. Está compuesta por todas las interfaces de usuario y los componentes necesarios para su correcto funcionamiento. Estos elementos pueden ser ficheros *Java Script, CSS* o servicios consumidos por *Java Script*.

Esta capa se encuentra representada por el proyecto *web* de la aplicación, y tiene interacción directa con la capa Procesos y Servicios y con la capa Negocio.

#### 3.2.1.2 Capa procesos y servicios (Process Service Layer):

Es el modelado del negocio mediante procesos haciendo uso de *Workflow .NET*. Define además los servicios que darán cumplimiento a las actividades modeladas en el proceso. Tiene contenida la lógica de los procesos de negocio representada por flujos de trabajo, las actividades que por su nivel de reutilización o importancia lógica se encapsulan en una actividad propia del proyecto y los servicios de *Runtime*, los cuales son los encargados de la interacción con los flujos de trabajo definidos; estos a su vez interactúan con los servicios de la capa Negocio haciendo uso para ello de una fábrica de servicios que le da la instancia correcta. Al mismo nivel se encuentra definida una fábrica para los servicios de *Runtime* que le permite a la capa de Presentación interactuar con estos sin necesidad de que sepan la instancia concreta del servicio que utilizan. Todos estos elementos se encuentran vinculados directamente con el *Framework Bison*.

Esta capa se encuentra relacionada con la capa Presentación a la que le brinda servicios y con la capa Negocio de la que consume servicios. Los proyectos que se encuentran relacionadas son *Project. Workflows* donde se encuentran definidos los flujos de trabajo y las actividades comunes, *Project*. *Services* donde se encuentran los servicios de flujo de trabajo, la fábrica correspondiente a estos y el *Bison Framework*.

#### 3.2.1.3 Capa negocio (Business Layer):

En esta capa se recogen todos los servicios necesarios para darle solución a los requerimientos de negocio que no pueden ser satisfechos por el flujo de trabajo. Los servicios se encuentran definidos según el contexto en el que se desenvuelven. Tienen la responsabilidad de manejar todas las operaciones sobre una entidad de negocio en específico, así como todas las entidades que por conceptos de composición se encuentran relacionadas con esta. Por cada entidad de negocio, se crea un controlador y una interfaz que debe ser implementada por el acceso a datos que le dará soporte.

Se encuentra constituida por tres proyectos que agrupan los diferentes componentes: *Project.Entities, Project. Services, Services.ServiceName*.

#### 3.2.1.4 Capa acceso a datos (Data Access Layer):

La capa Acceso a datos está directamente relacionada con los servicios definidos en el negocio. Para establecer esta relación hace uso de la interfaces de conectores y de la fábrica de conectores que define la capa Negocio. De esta manera es posible realizar cambios en esta capa sin que se vean afectadas las demás. Su principal función es realizar una implementación de las interfaces definidas en la capa Negocio y al mismo tiempo trabajar directamente con las fuentes de datos establecidas.

#### 3.2.1.5 Capa base de datos (Data Base Layer):

Está constituida por todo el conjunto de tablas y procedimientos que permiten el almacenamiento de la información recolectada y procesada por los procesos. (9)

### 3.2.2 Patrones de diseño.

El desarrollo de un subsistema conlleva en la mayoría de las ocasiones darle solución a problemas muy complejos que ya alguien más ha resuelto. Por esta razón uno de los pasos a tener en cuenta cuando se decide desarrollar un proyecto de software, es identificar que patrones pueden ser utilizados. Donde se entiende por patrón de diseño a:

- Una solución estándar para un problema común de programación.
- Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios.
- Un proyecto o estructura de implementación que logra una finalidad determinada.
- Un lenguaje de programación de alto nivel.
- Una manera más práctica de describir ciertos aspectos de la organización de un programa.
- Conexiones entre componentes de programas.

Los patrones de diseño poseen las siguientes características:

- Pueden incrementar o disminuir la capacidad de comprensión de un diseño o de una implementación.
- Disminuir la capacidad al añadir accesos indirectos o aumentar la cantidad de código.
- Se disminuye también al regular la modularidad, separar mejor los conceptos y simplificar la descripción.
- Además de esto es necesario tener en cuenta que hacer uso de patrones significa en muchos casos disminución en el rendimiento de la aplicación, por ello es necesario lograr un equilibrio entre la flexibilidad y el rendimiento dentro del subsistema.
- Implica que no es razonable utilizar patrones hasta que no se tenga totalmente claro el dominio y el problema a tratar.

Para el desarrollo del subsistema en cuestión se han considerando un conjunto de patrones que permiten darle flexibilidad y no constituyen cambios grandes en el rendimiento del mismo. Además se identificaron un conjunto de patrones específicos para el desarrollo de los flujos de trabajo que brindan claridad y fortaleza a los diseños de estos, los cuales se describen a continuación.

- **Encapsulación*:*** Facilita esconder algunos componentes, permitiendo sólo accesos estilizados al objeto. Se hace uso de este patrón en casi todas las clases que componen al subsistema permitiendo que estas solo posean como elementos públicos aquellos que son exclusivamente necesarios.
- **Subclase**: Propone heredar miembros por defecto de una superclase, seleccionando la implementación correcta. Se puede encontrar este patrón con más fuerza en las entidades de negocio que por su conceptualización las funciones y la información que almacenan pueden estar diferenciadas en cierta medida.
- **Excepciones*:*** Permite introducir estructuras de lenguaje para arrojar e interceptar excepciones. Identifican los diferentes tipos de errores a tratar dentro del subsistema creando clases que permitan identificar cada tipo de error en el momento de ejecución.
- **Fábrica:** Provee de una interfaz para crear familias de objetos relacionados o dependientes sin especificar los tipos concretos de clases. Su uso se encuentra centrado a la creación de los conectores correspondientes al acceso a datos que se esté utilizando, así como en la obtención de los servicios a utilizar.
- ***Singleton***: Se asegura que solo se pueda crear una instancia de la clase y ofrece un punto global de acceso a esta instancia. El uso de este patrón permite que los servicios puedan ser creados solo una vez.
- ***GRASP***[39]: El uso de este patrón está totalmente ligado a cada componente desarrollado en el subsistema, donde cada uno de ellos posee solo las funcionalidades acorde a las particularidades que lo caracterizan.

#### 3.2.2.1 Patrones de flujos de trabajo.

Los patrones van desde los más simples como el patrón secuencial hasta los complejos, por ejemplo, el patrón de sincronización. Los patrones de flujos de trabajo pueden ser clasificados en las siguientes categorías:

[39] GRASP (General Responsibility Assignment Patterns): Patrones Generales de para Asignación de Responsabilidades.

- **Patrones de control de flujo básicos:** Están presentes en la mayoría de los lenguajes de flujo de trabajo, y sirven para modelar procesos secuenciales, paralelos, o aquellos que incluyan alguna decisión.
- **Patrones de ramificación avanzada y sincronización:** Estos superan a los patrones de control de flujo básico al permitir tipos avanzados de bifurcación y sincronización.
- **Patrones estructurales:** Permiten terminar un subproceso cuando ya no haya nada que hacer, o permiten definir ciclos de forma arbitraria.
- **Patrones que manejan múltiples instancias:** Cuando se le da seguimiento a un caso, algunas veces es necesario que el proceso sea instanciado muchas veces.(9)

## 3.3 Especificación de clases.

### 3.3.1 Diagrama de las clases entidades del diseño.

Las clases entidades son las que permiten crear objetos del dominio del negocio para persistir determinada información, y para que la misma sea duradera debe ser guardada en una base de datos. Una base de datos es un conjunto de datos que tiene las siguientes propiedades implícitas:

- Representa algún aspecto del mundo real, llamado mini mundo o universo de discurso. Las modificaciones del mini mundo se reflejan en la base de datos.
- Es un conjunto de datos lógicamente coherentes, con un cierto significado inherente. Una colección aleatoria de datos no puede considerarse propiamente una base de datos.
- Una base de datos se diseña, construye y puebla con datos para un propósito específico. Está dirigida a un grupo de usuarios y tiene ciertas aplicaciones preconcebidas que interesan a distintos usuarios. (31)

En la Figura 10 se muestra el diagrama de las clases entidades del diseño.

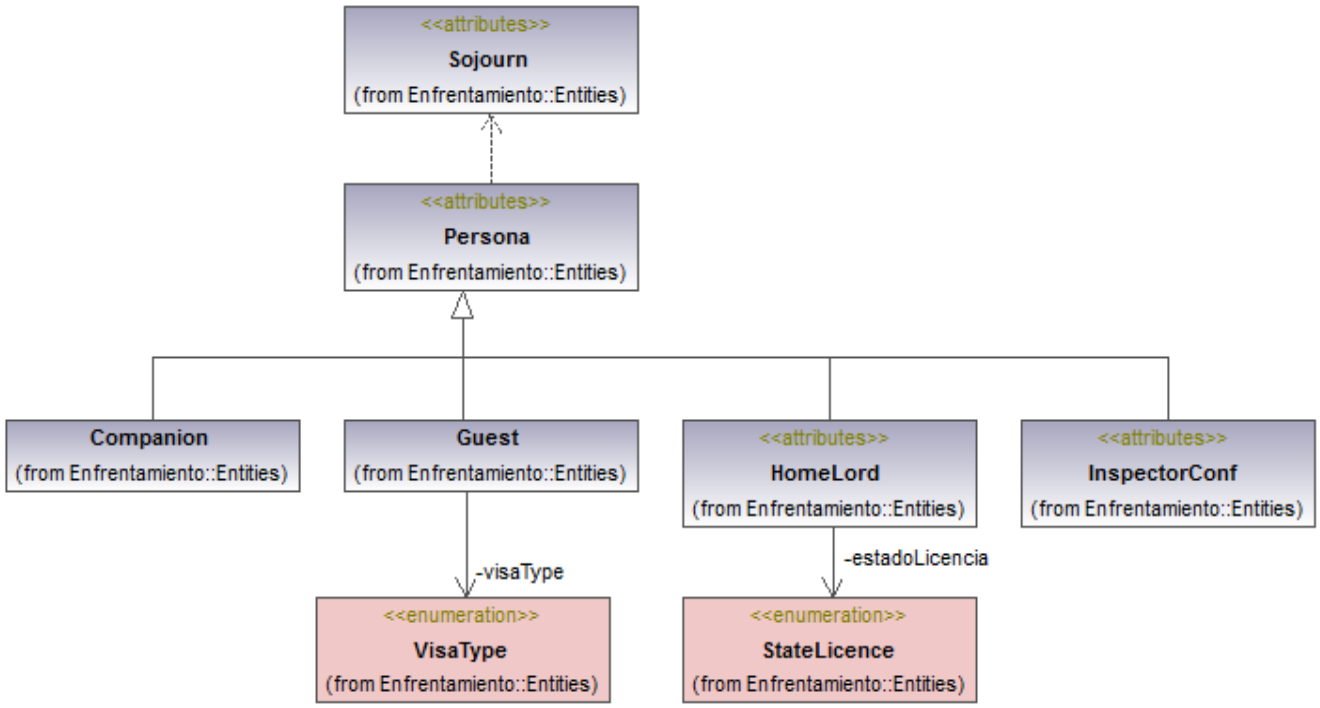



**Figura 10. Diagrama de clases entidades del diseño.**

Para el caso específico de este subsistema, se definieron varias clases entidades, a continuación en la Tabla 3.1 se muestra la descripción correspondiente a la entidad *Person* que contiene toda la información referente a una persona pues todos los procesos que se desarrollan dependen de lo que se haya realizado sobre esta entidad. El resto de las descripciones se encuentran en el Anexo 6.

| **Nombre** | Person | |
|---|---|---|
| **Descripción** | Entidad que contiene los datos generales de la persona. | |
| **Atributos** | **Tipo de dato** | **Definición** |
| IdentityNumber | string | Carné de identidad. |
| RegisterYear | int | Año en que se registra. |
| Tomo | int | Tomo del registro. |
| Folio | int | Folio del registro. |
| FirstName | string | Primer nombre. |
| SecondName | string | Segundo nombre. |
| FirstSurname | string | Primer apellido. |
| SecondSurname | string | Segundo apellido. |

| | | |
|---|---|---|
| MotherName | string | Nombre de la madre. |
| FatherName | string | Nombre del padre. |
| BirthDate | DateTime | Fecha de nacimiento. |
| Sex | char | Sexo. |
| Donante | byte | Donante o no. |
| Talla | int | Tamaño. |
| Weigh | string | Peso. |
| Dead | bool | Fallecida o no. |
| MigrationCondition | string | Condición migratoria. |
| Country | Country | Información del país. |
| EyeColor | EyeColor | Color de ojos. |
| SkinColor | SkinColor | Color de piel. |
| HearColor | HearColor | Color de pelo. |
| Address | Address | Direcciones de la persona. |
| PersonType | PersonType | Tipo de persona. |
| CivilRegister | CivilRegister | Información del registro civil. |
| OptionalDate | OptionalDate | Datos opcionales. |
| PersonImage | PersonImage | Imágenes de la persona. |

**Tabla 3.1. Descripción de la entidad *Person*.**

## 3.4 Servicios del subsistema.

La implementación de la arquitectura en capas permite que se puedan asignar correctamente las responsabilidades a cada una de las clases conectoras y gestoras de la información para llevar a cabo un buen acceso a datos. Las clases conectoras son las que tienen en su haber el acceder directamente a la base de datos, estas manipulan entidades de acceso a datos, y cuentan con la estructura de la lógica de acceso a datos, permitiendo una independencia total del gestor de base de datos a utilizar, el cambio en la bases de datos solo genera la actualización de estas clases. Dichas entidades son utilizadas por los servicios con los que cuenta el negocio de la solución.

En la Figura 11 se muestran los servicios utilizados por el subsistema para la interacción con los procesos.

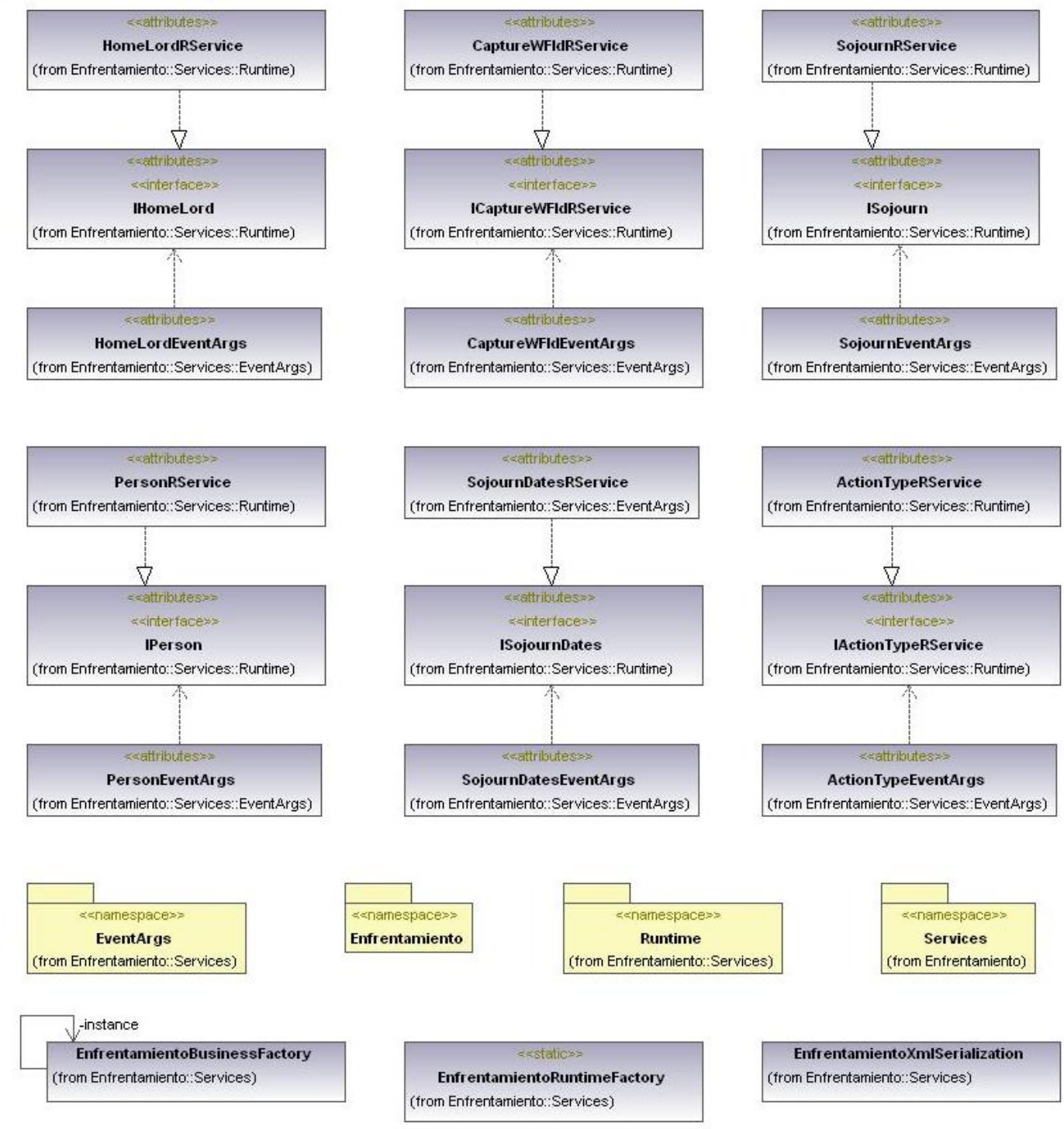



**Figura11. Servicios utilizados por el subsistema.**

### 3.4.1 Servicios de interacción con los procesos:

Estos servicios son los únicos que pueden comunicarse con los procesos permitiendo de esta manera lograr una independencia de la capa en la que se encuentran.

- HomeLordRService: Permite guardar los datos captados de un casero.
- CaptureWFIdRService: Permite guardar el identificador de un flujo de trabajo.
- SojournRService: Permite guardar una estancia.
- PersonRService: Permite guardar los datos captados de una persona.
- SojournDatesRService: Permite almacenar las fechas de una estancia.
- ActionTypeRService: Permite obtener el tipo de acción a realizar en un flujo de trabajo.

### 3.4.2 Servicios de negocio:

Estos servicios son los encargados de dar solución a los requerimientos del flujo de negocio que no pueden ser plasmados en el modelo gráfico de los procesos:

SearchPersonDIEService: Es uno de los servicios de mayor importancia en el desarrollo de la aplicación, es genérico a la hora de buscar, puede buscar por los parámetros definidos por el usuario, además de ser reutilizable; mediante el mismo tanto la dirección del MININT como la del país tendrán conocimiento de aquellas personas que tengan antecedentes migratorios o policiales, así como información detallada de cualquier persona con ciudadanía nacional o extranjera que resida en el país.

SearchHomeLordDIEService: Es el encargado de realizar las búsquedas de los caseros, brindando los datos específicos de la persona, además de las violaciones que pueda haber cometido, la cuales pueden afectar que mantenga o no su licencia.

## 3.5 Diseño de los flujos de trabajo del subsistema.

En el diseño de un flujo de trabajo se utilizan una serie de actividades, que llevan a cabo la manipulación y el procesamiento de la información, así como la comunicación del flujo de trabajo con el mundo de externo. Las actividades más utilizadas en el diseño se describen a continuación:

- ***ClientActivity*:**

Permite definir las interacciones del proceso con los usuarios de la aplicación. Estas pantallas pueden ser de diferentes tipos. Para su uso es necesario especificarle la *url* del componente representado.

- ***AssociationActivity*:**

Representa la asociación de un valor a una propiedad dentro del flujo de trabajo. El valor de origen puede ser una variable de entrada o bien un valor *string* que sea pasado en la propiedad *value*. Además de representar la asociación de valores de manera gráfica también se encarga de dejar una traza de los valores que han sido modificados a lo largo del proceso siempre que la propiedad *IsTreaceble* tenga valor *true*.

- ***HandleExternalEventActivity*:**

Esta actividad bloquea el flujo de trabajo hasta que el evento especificado por la propiedad *InterfaceType* sea lanzado, generalmente es utilizada para la interacción con el usuario y provee la principal fuente de extracción de información con el ámbito exterior al flujo de actividades. Es a través de esta actividad que se obtienen todas las informaciones requeridas por el flujo de trabajo desde las interfaces de usuario de los módulos.

- ***IfElseActivity*:**

Permite ejecutar condicionalmente una de varias bifurcaciones. Coloca una condición en cada rama del bloque *ifElse*. Si la condición se evalúa como *true*, se ejecutan las actividades contenidas en la actividad *IfElseBranchActivity* quien representa una bifurcación de una actividad *IfElseActivity*. Esta actividad permite en el diseño de la solución propuesta la toma de decisiones entre uno o más caminos para la ejecución de un flujo correspondiente al resultado obtenido de la búsqueda de la persona.

- ***CallExternalMethodActivity***:

Se utiliza para la comunicación de entrada y salida con un servicio local externo al flujo de trabajo. La clase *CallExternalMethodActivity* invoca el método especificado por las propiedades *InterfaceType* y *MethodName* quienes indicarán el lugar donde se aloja el servicio y método a invocar. Es mediante dicha actividad que se realizan las llamadas a los métodos encargados de realizar todas las verificaciones concernientes al resultado de la búsqueda de una persona.

- ***WhileActivity*:**

Permite que el flujo de trabajo se ejecute en *bucle* hasta que se cumpla la condición indicada. Es la actividad mediante la cual es posible realizar la ejecución de un mismo flujo para procesos diferentes sin necesidad de duplicar el flujo en el diseño del mismo. (9)

En las Figuras 12 y 13 se muestra en dos secciones el workflow del Proceso estancia de una Visa familiar que hace un llamado al workflow de la Figura 14 que pertenece al Proceso estancia de una Visa familiar auxiliar, porque este flujo de trabajo es muy similar al del Proceso estancia para un huésped casero, que al igual que los demás se puede encontrar en el Anexo 7.

**(Sección I)**

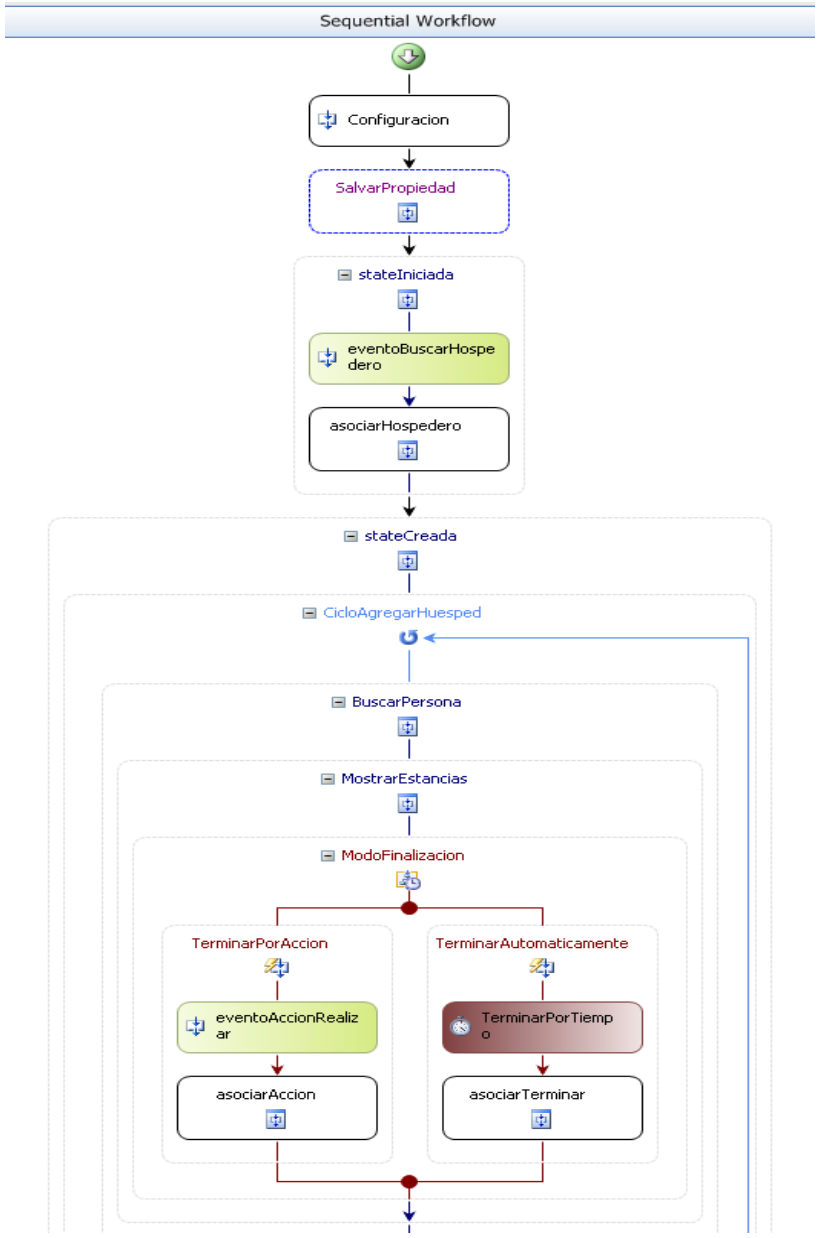



**Figura 12. Diseño del flujo de trabajo del Proceso estancia de una Visa familiar (Sección I).**

**(Sección II)**

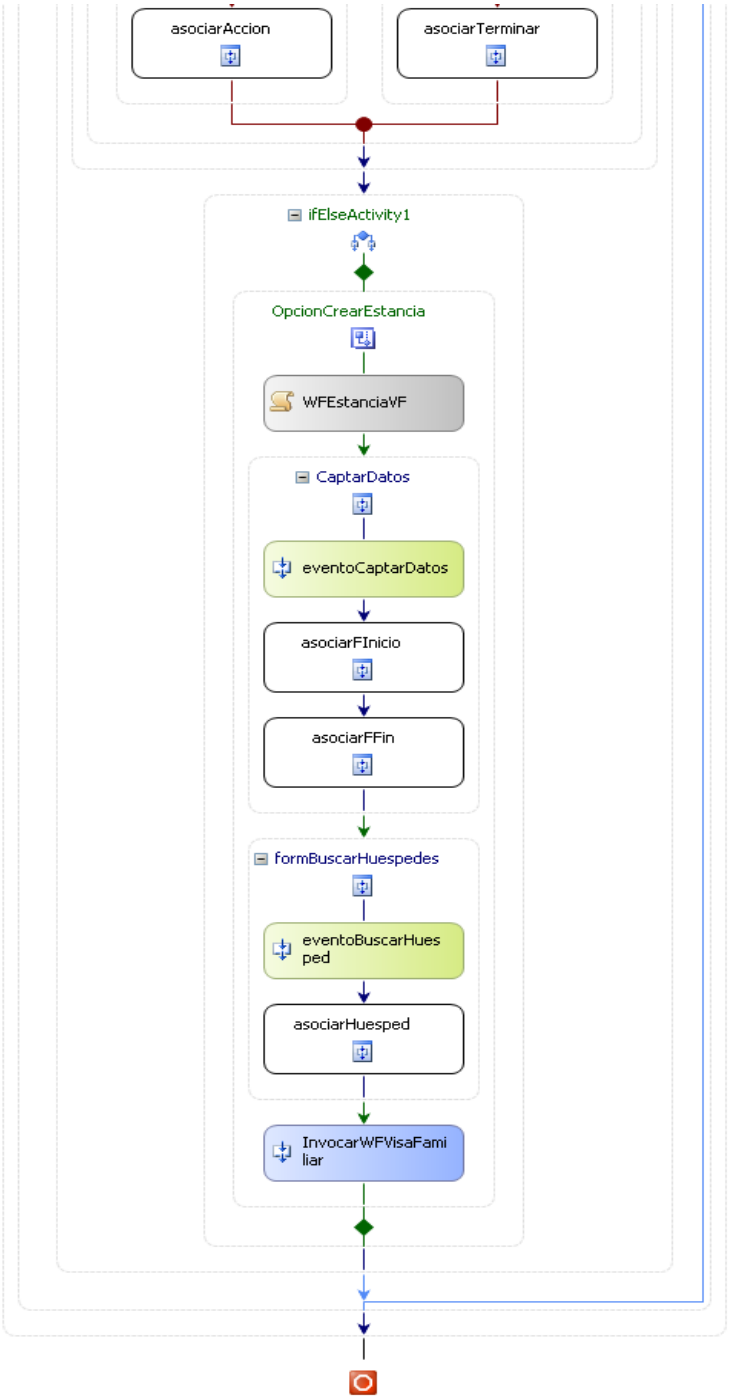



**Figura 13. Diseño del flujo de trabajo del Proceso estancia de una Visa familiar (Sección II).**

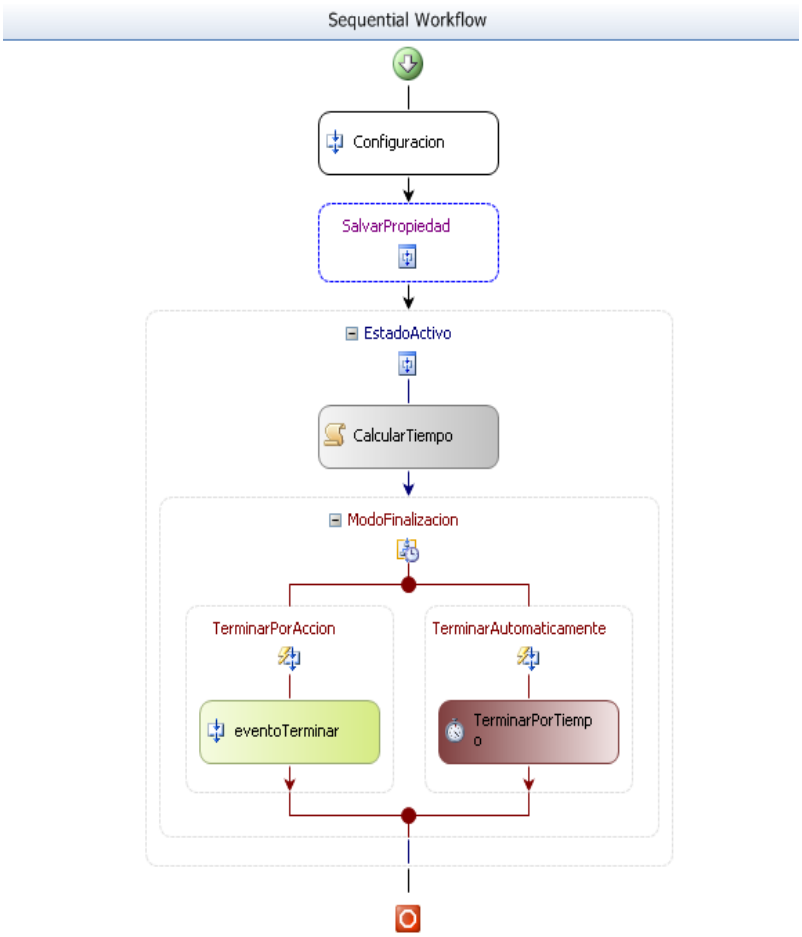



**Figura 14. Diseño del flujo de trabajo del Proceso estancia de una Visa familiar auxiliar.**

## 3.6 Conclusiones parciales.

En este capítulo, se realizó el análisis y diseño del subsistema, haciendo una familiarización a fondo con la arquitectura en capas y los patrones de diseño entre los que destacan el de encapsulación, subclase, *singleton y otros*, se identificaron las clases entidades, los principales servicios que se utilizan y se definieron los flujos de trabajo para los procesos estancia para un huésped casero y para una Visa familiar.

# CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS

## 4.1 Introducción.

Este capítulo realiza la implementación y pruebas del subsistema, haciendo énfasis en los estándares de codificación y el tratamiento que se le dará a los posibles errores. Además se definen los diagramas de componentes y despliegue, así como las interfaces de usuario. Se realiza también el diseño de los casos de pruebas, y se obtienen los resultados de los mismos.

## 4.2 Estándares de codificación y tratamiento de errores.

Un patrón de nomenclatura coherente es uno de los elementos más importantes de la previsibilidad y el descubrimiento de una biblioteca de clases administradas. El uso generalizado y la comprensión de estas directrices de denominación deberían eliminar muchas de las preguntas más comunes.

### 4.2.1 Estilos para la capitalización.

Se podrán utilizar los siguientes tres convenios para la capitalización de los identificadores:

- **Pascal.**

La primera letra en el identificador y la primera letra de cada subsiguiente palabra concatenada se capitalizan. Puede utilizar los identificadores de Pascal *case* en caso de tres o más caracteres. Por ejemplo: *BackColor*.

- **Camello.**

La primera letra en el identificador está en minúscula y la primera letra de cada subsiguiente palabra concatenada es mayúscula. Por ejemplo: *backColor*.

- **Mayúscula.**

Todas las letras en el identificador se capitalizan. Esta convención se utilizará sólo para los identificadores que constan de dos o menos letras. Por ejemplo: *System.IO, System.Web.UI*.

### 4.2.2 Sensibilidad a mayúsculas.

- No se debe utilizar nombres o identificadores que requieran ser *case sensitivity*[40].
- No se debe crear dos *namespaces*[41] que se diferencien solo en el uso de las mayúsculas.

[40] Case Sensitivity: Es una expresión usada en jerga informática que se aplica a los textos en los que tiene alguna relevancia escribir un carácter en mayúsculas o minúsculas.
[41] Namespaces: Espacios de Nombres.

- No crear funciones con nombres de parámetros que se diferencian solo en el uso de la mayúscula.
- No se debe crear *namespaces* con nombres de clases que se diferencien solo en el uso de las mayúsculas.
- No crear clases con propiedades que se diferencien solo en el uso de las mayúsculas.
- No crear clases con métodos que se diferencien solo en el uso de las mayúsculas.

### 4.2.3 Abreviaturas.

Para evitar confusiones y garantizan la interoperabilidad entre lenguajes, se siguen las siguientes reglas sobre el uso de abreviaturas:

- No usar abreviaturas o contracciones como parte del nombre de un identificador. Por ejemplo, se debe usar *GetWindows* en lugar de *GetWin*.
- No utilizar siglas que no son generalmente aceptadas en el campo de la informática.
- Cuando sea apropiado utilizar las siglas para remplazar frases o nombres largos. Por ejemplo: *UI* por *User Interface* y *OLAP* por *On-line Analytical Processing*.
- Cuando se usen acrónimos, debe utilizarse notación Camello o Pascal para aquellos que son de más de dos caracteres. Por ejemplo *HtmlButton* o *htmlButton*. Sin embargo se capitalizará los que consisten en solo dos caracteres, como *System.IO* en lugar de *System.Io*.
- No se debe usar abreviaturas en los identificadores o en los nombres de los parámetros. En caso de ser necesario usar siempre notación Camello para las abreviaturas de más de dos caracteres, aún cuando se contradiga el estándar de la abreviatura de la palabra.

### 4.2.4 Elección de las palabras.

Evitar el uso de los nombres de las clases o *namespaces* comúnmente usados en el *.Net Framework* para usar como nombre de las clases. Por ejemplo no usar los siguientes nombres para clases: *System, Collections, Forms o UI*. Evitar el uso de palabras reservadas en los nombres de los identificadores.

### 4.2.5 Para evitar confusión de nombre y tipo.

Se deben utilizar nombres que describan a sus identificadores en vez de nombres que describen el tipo de identificador. (32)

## 4.3 Tratamiento de errores.

Los mensajes de error que se emiten en el subsistema se muestran en un lenguaje de fácil comprensión para los usuarios. Cuando se introduce información en un formulario y faltan datos, sale un mensaje de alerta indicando el campo o dato que falta. Además en caso de que el usuario introduzca un dato que no corresponda con el valor que debe entrar, cambia de color el componente indicando que está mal.

A continuación en la Figura 15 se muestra un ejemplo del tratamiento de errores.

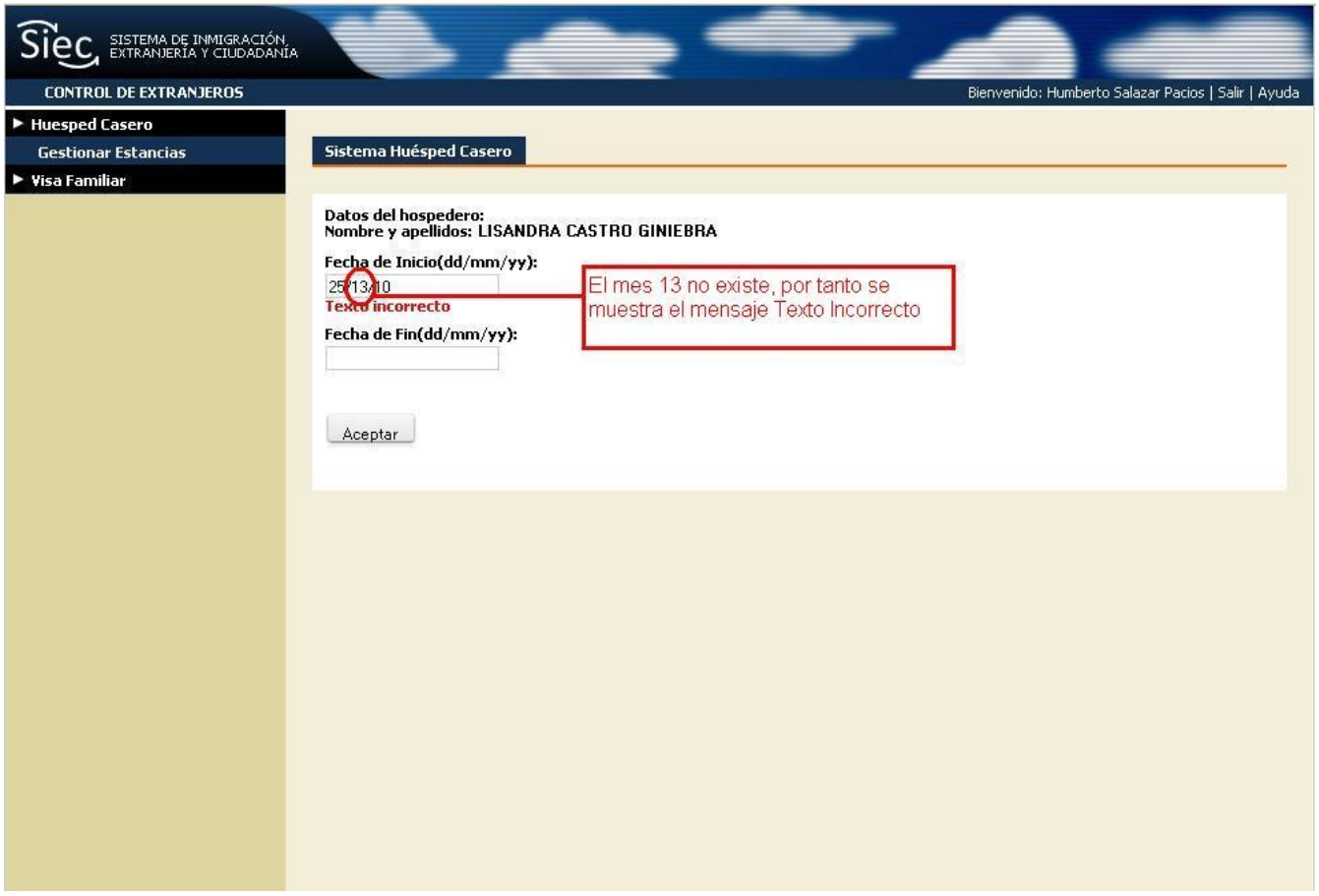



**Figura 15. Ejemplo del tratamiento de errores.**

## 4.4 Diagrama entidad-relación de la base de datos.

La base de datos que se utiliza cuenta con características diferentes respecto a las tradicionales. La información no se almacena en tablas relacionadas, sino que persiste en el flujo de trabajo de cada proceso llevado a cabo.

En la construcción de la base de datos se definen varias tablas las cuales se encargan de guardar toda la información, entre las principales se encuentran: dEstancia, dCasero y dInspector.

Se definieron además nomencladores, como por ejemplo nEstEstancia, nCantHabitCasero y nEstLicCasero; que contienen los valores predeterminados a tomar por una entidad. El diccionario de datos donde se describen las tablas que integran la base de datos se muestra en el Anexo 6.

En la Figura 16 se muestra el modelo entidad-relación con todas las tablas que lo componen.

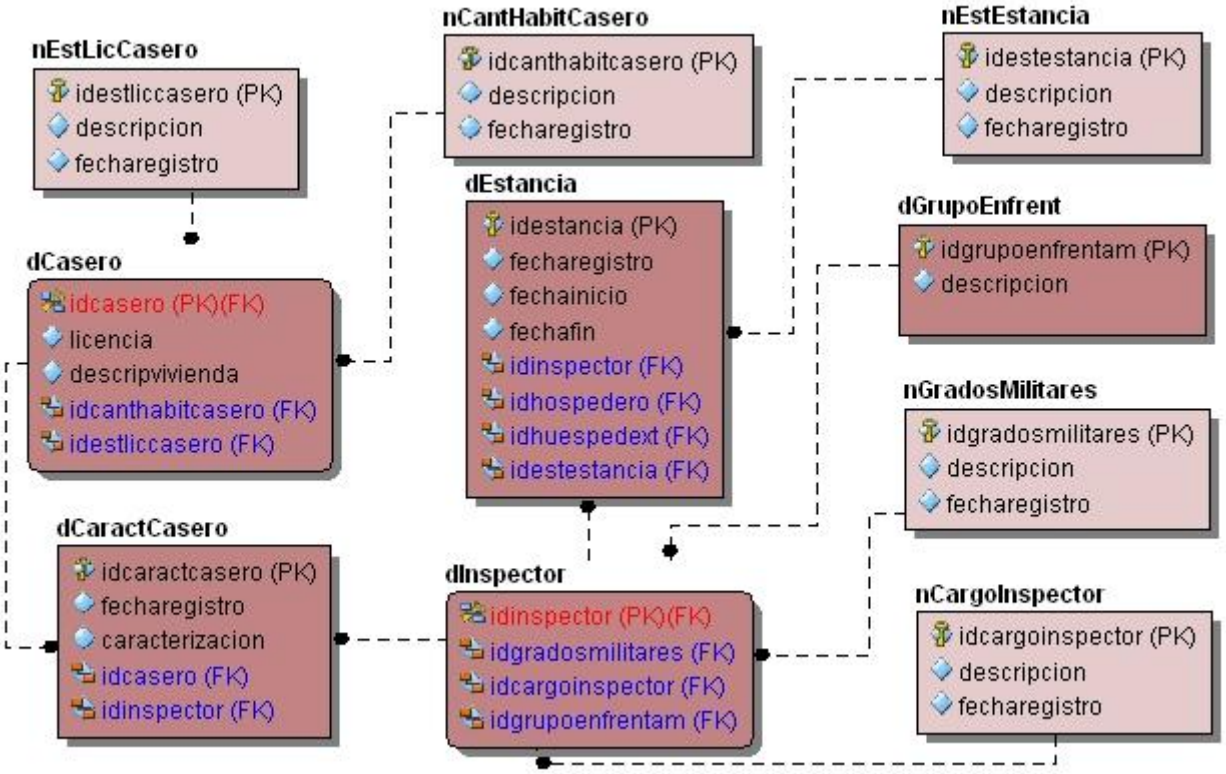



**Figura 16. Diagrama entidad-relación de la base de datos.**

## 4.5 Diagrama de componentes.

Los diagramas de componentes ilustran las piezas del software que conforman un sistema. Un diagrama de componentes tiene un nivel más alto de abstracción que un diagrama de clase, usualmente un componente se implementa por una o más clases en tiempo de ejecución. Estos son como bloques de construcción.

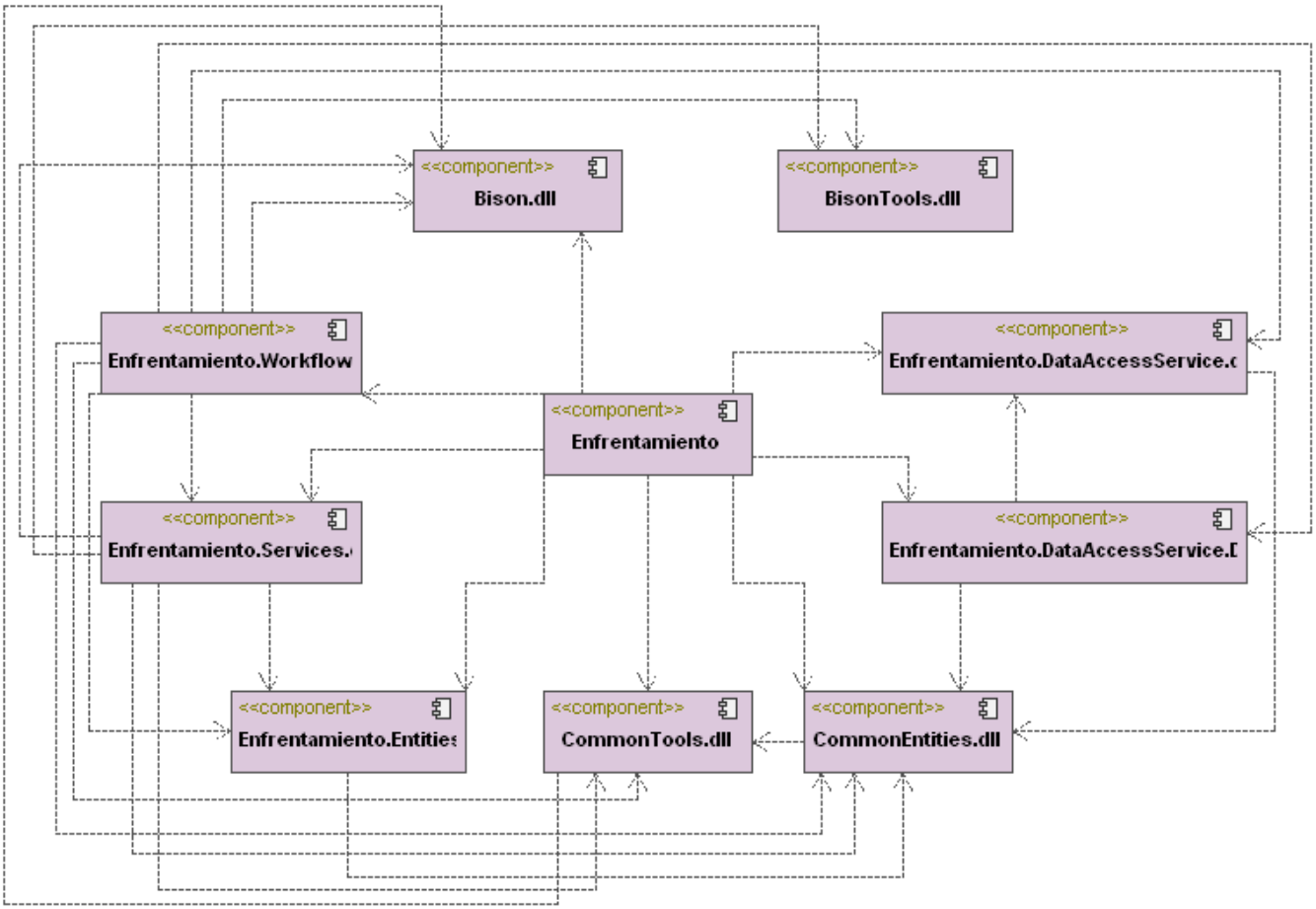



**Figura 17. Diagrama de componentes.**

### 4.5.1 Descripción del diagrama de componentes.

El diagrama de la Figura 17 muestra los principales componentes para la arquitectura del proyecto en general. A pesar de ser complicado de apreciar, hay una influencia del patrón de arquitectura orientado a n capas. Primeramente el paquete Enfrentamiento recoge todos los componentes los cuales están agrupados en un proyecto individual, allí existe un paquete con todos los módulos y en cada uno de ellos se encuentran los llamados *UserControls*, ficheros estructurados *<nombre.ascx>* que tienen los componentes gráficos y los formularios de las páginas *web*. El flujo de interfaces es gestionado por la capa Procesos y Servicios, posibilitando que cualquier cambio que se realice no afecte directamente a esta capa, el primer paquete es el *Enfrentamiento*.*Workflow*, este contiene el paquete *Bison*, el cual actúa como principal guía de los procesos realizados en los *workflows*, porque él tiene las funcionalidades especificas como las actividades y servicios importantes en la arquitectura. El *Enfrentamiento.Workflow* también está relacionado con el *Enfrentamiento.Entities*, componente

encargado de agrupar a todas las entidades con sus atributos que le darán solución a lo requerido en las clases controladoras y conectoras, además darán respuestas a los servicios implementados. *Enfrentamiento.Services* también se incluye en este paquete, este recoge los servicios, que son los que permiten la interacción con las instancias de proceso creadas.

El componente *CommonTools*, es un componente externo pero relacionado con el proyecto y se encarga de contener argumentos comunes para la utilización de todos los módulos, como los elementos de la *Master Page*.

Los componentes *Enfrentamiento.DataAccessService.dll* y *Enfrentamiento.DataAccessService.DAL*, son los encargados del proceso de búsqueda y la conexión a la base de datos, los cuales tienen una base común que es el *CommonEntities*.

## 4.6 Diagrama de despliegue.

El modelo de despliegue describe la distribución física del subsistema, muestra como están distribuidos los componentes de software entre los distintos nodos de cómputo. Permite comprender la correspondencia entre la arquitectura software y la arquitectura *hardware*.

### 4.6.1 Descripción del diagrama de despliegue.

Durante el despliegue la distribución a seguir es la siguiente: un servidor de aplicaciones al que se podrán conectar varias computadoras clientes mediante el protocolo de comunicación *HTTPs* y un servidor de bases de datos para persistir toda la información que se genere mediante la aplicación, la que se conectará por medio del protocolo *Oracle TNS*.

En la Figura 18 se muestra el diagrama de despliegue del subsistema desarrollado.

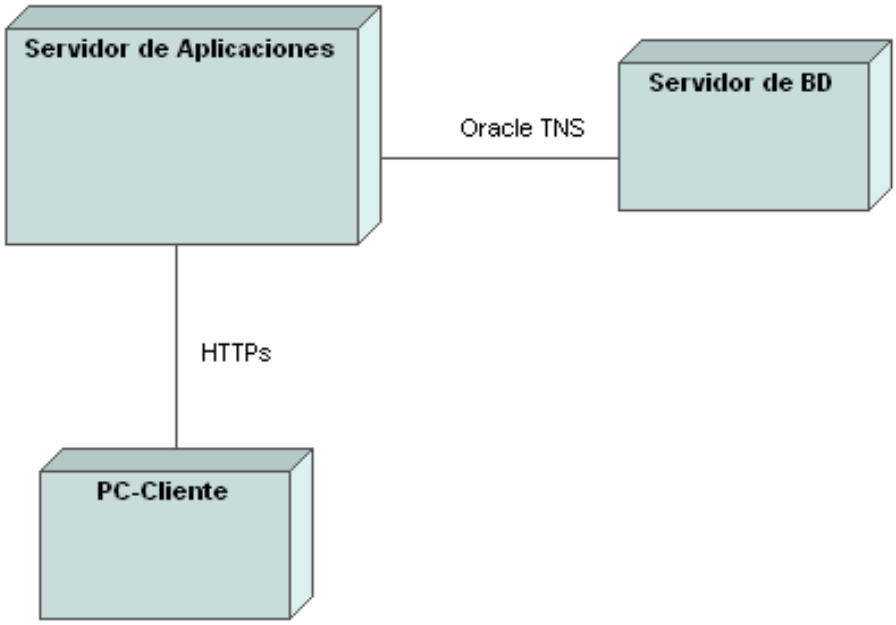



**Figura 18. Diagrama de despliegue del subsistema.**

## 4.7 Interfaces de usuarios.

La interfaz de usuario (*IU*) es la parte del programa que permite a éste interactuar con el usuario. Pueden adoptar muchas formas, que van desde la simple línea de comandos hasta las interfaces gráficas que proporcionan las aplicaciones más modernas, estas resultan ser el aspecto más importante de cualquier aplicación ya que sin una interfaz fácil, impide que los usuarios saquen el máximo rendimiento del programa.

En la Figura 19 se muestra la interfaz de usuario de inicio del subsistema, las demás interfaces se encuentran en el Anexo 8.

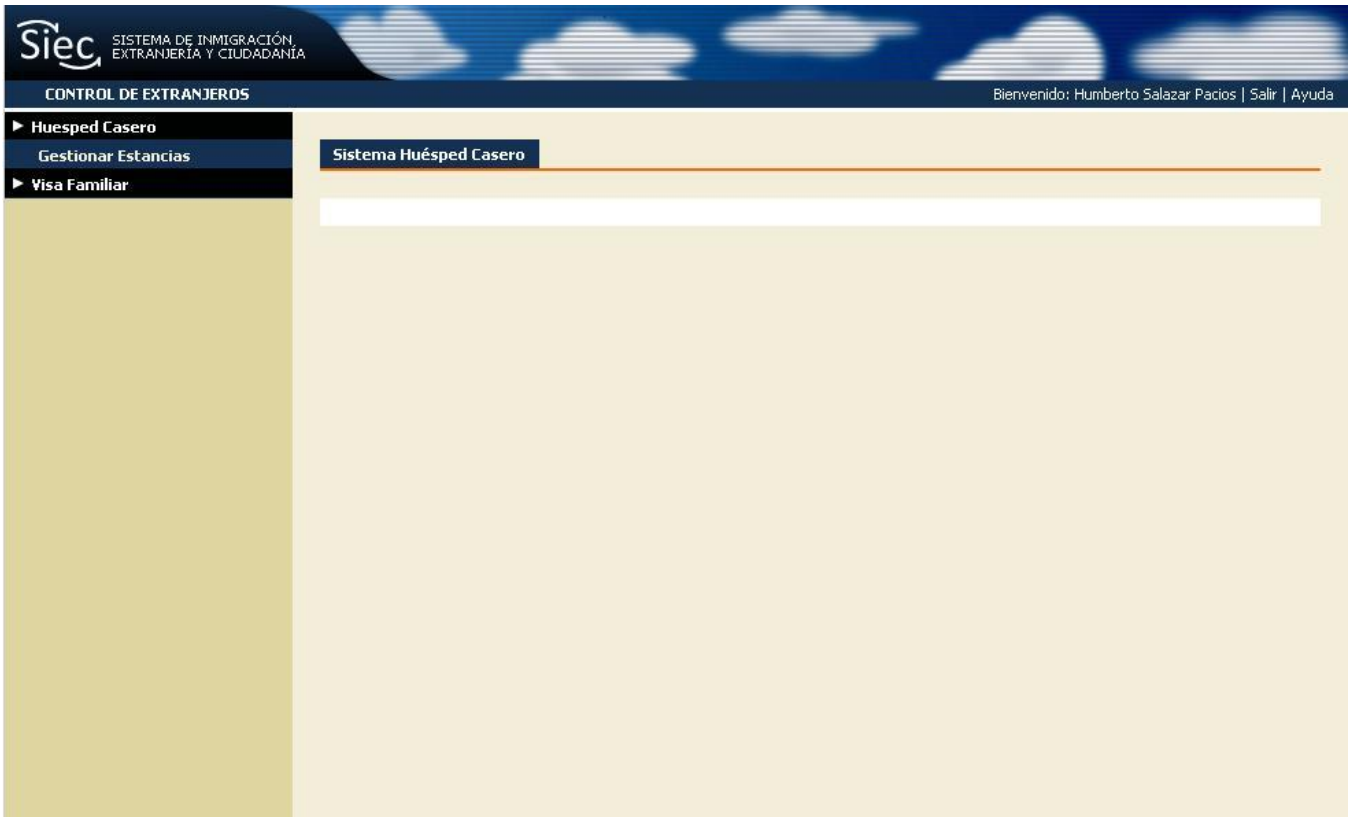



**Figura 19. Interfaz de usuario de inicio.**

## 4.8 Pruebas.

Los proyectos de desarrollo de software han padecido tradicionalmente problemas de calidad, tanto en el propio proceso de desarrollo como en los productos que entregan. Esta problemática tiene su origen en las habituales desviaciones de plazos y esfuerzo sobre los valores previstos y en la frecuente aparición de fallos durante la implantación y operación de los productos resultantes.

La validación de software es de vital importancia ya que proporciona un alto grado de confianza y seguridad no solo en el producto sino en los resultados que se obtienen al implantarlo. Validar el software que se produce es un requisito que debe ser cumplido con rigor. (33)

### 4.8.1 Diseño de casos de pruebas.

El diseño de casos de pruebas consiste en la confección de los mismos según la técnica identificada previamente. Cada uno va acompañado del resultado que ha de producir el software al ejecutarlo para detectar un posible fallo en el programa. Definen un conjunto de entradas, condiciones de ejecución y

resultados esperados para un objetivo particular. Cada técnica de pruebas proporciona criterios distintos para generar estos casos o datos de prueba.

### 4.8.2 Pruebas unitarias. Pruebas de caja blanca.

Las pruebas unitarias, comienzan con la prueba de cada módulo. Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno funcione correctamente por separado. Su objetivo es aislar cada parte del programa y mostrar que las partes individuales son correctas. Proporcionan un contrato escrito que el fragmento de código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas: fomentan el cambio, simplifica la integración, documenta el código, separa la interfaz del código y hace que los errores estén más acotados y sean fáciles de localizar. (34)

Las pruebas de caja blanca; denominan a un tipo de prueba de software que se realiza sobre las funciones internas de un módulo. Así como las pruebas de caja negra ejercitan los requisitos funcionales desde el exterior del módulo, las de caja blanca están dirigidas a las funciones internas. Se llevan a cabo en primer lugar, sobre un módulo concreto, para luego realizar las de caja negra sobre varios subsistemas (integración). A continuación se muestra una imagen que refleja el proceso de las pruebas de caja blanca. (35)

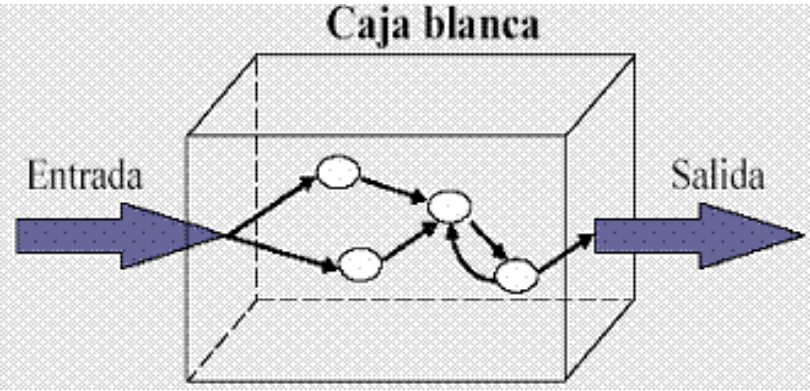



**Figura 20. Imagen del proceso de las pruebas de caja blanca.**

La siguiente prueba se realiza al método que permite devolver la cantidad de personas que existan en la base de datos dado un criterio de búsqueda determinado. Como parámetros del método se pasan *bool fonetic* que es el tipo de búsqueda y *Dictionary<object, object> parameters* que es un objeto que contiene todos los parámetros de la búsqueda. Para el ejemplo a probar se utiliza el siguiente juego de datos:

bool fonetic = false;

Dictionary<object, object> parameters = new Dictionary<object,object>();

**Método a probar:**

```
[WebMethod]
    public int GetTotal(bool fonetic, Dictionary<object, object> parameters)
    {
      parameters.Add("Fonetico", fonetic);
      try
      {
        return
EnfrentamientoBusinessFactory.Instance.SearchPersonDIEService.count(parameters);
      }
      catch (System.Exception ex)
      {

        throw;
      }

    }
```

**Método de prueba:**

```
 [TestMethod()]
    public void GetTotalTest()
    {
      SearchService target = new SearchService();
      bool fonetic = false;
            Dictionary<object, object> parameters = new Dictionary<object,
            object>();
      int expected =822;
      int actual;
            actual = target.GetTotal(fonetic, parameters);
            Assert.AreEqual(expected, actual);

    }
  }
```

Las demás pruebas unitarias realizadas se encuentran en el Anexo 9.

### 4.8.3 Pruebas de sistema. Pruebas de caja negra.

Las pruebas de sistema consisten en el siguiente procedimiento: el software es ensamblado totalmente con cualquier componente *hardware* que requiera, se prueba para comprobar que se cumplen los requisitos funcionales. Cualquier pieza de software completo, desarrollado o adquirido, puede verse como un sistema que debe probarse, ya sea para decidir acerca de su aceptación, para analizar defectos globales o para estudiar aspectos específicos de su comportamiento, tales como seguridad o rendimiento. (36)

Las pruebas de caja negra denominan a aquel elemento que es estudiado desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno. En otras palabras, de una caja negra interesa su forma de interactuar con el medio que le rodea (en ocasiones, otros elementos que también podrían ser cajas negras) entendiendo qué es lo que hace, pero sin dar importancia a cómo lo hace. Por tanto, de una caja negra deben estar muy bien definidas sus entradas y salidas, es decir, su interfaz. En cambio, no se precisa definir ni conocer los detalles internos de su funcionamiento. A continuación se muestra una imagen que representa el proceso de las pruebas de caja negra. (37)

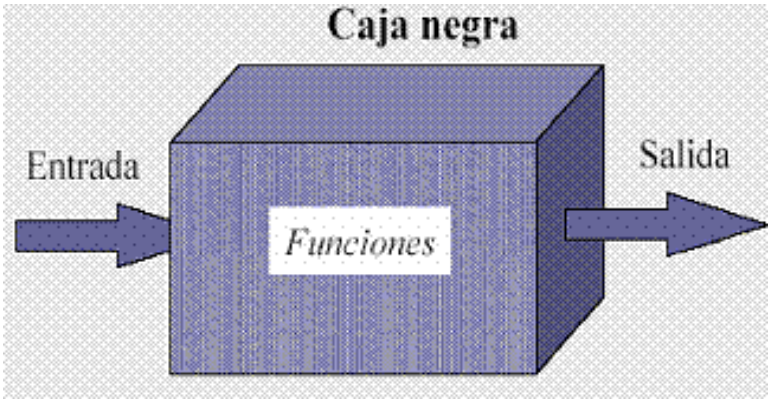



**Figura 21. Imagen de las pruebas de caja negra.**

En la Tabla 4.1 se muestra la descripción del caso de prueba del RF 1 Buscar persona y en la Tabla 4.2 se muestran una de las iteraciones realizadas, por ser este el proceso del cual depende la realización de los demás procesos para la gestión de estancias. Las demás pruebas realizadas se encuentran en el Anexo 10.

### CPR1 Buscar persona

**Condiciones de ejecución:**

- El usuario debe estar autenticado en el subsistema.

| Nombre del Flujo | Escenarios del flujo | Descripción de la funcionalidad | Flujo central |
|---|---|---|---|
| Buscar persona | 1.1 Seleccionar la opción “Buscar Persona”. | Se muestra la opción Buscar Persona | 1- Seleccionar la opción Buscar Persona. |

| | | | |
|---|---|---|---|
| | 1.2: Mostrar los campos para realizar la búsqueda. | Muestra los campos para la búsqueda.<br>a. Nombre(s).<br>b. Apellido(s).<br>c. Tipo de documento de Identificación.<br>d. Nacionalidad.<br>e. Número de documento.<br>f. Raza.<br>g. Sexo.<br>h. Fecha de nacimiento.<br>i. Fecha de entrada a Cuba. | 1- Seleccionar la opción Buscar Persona. |
| | 1.3: Insertar los criterios de búsqueda. | Permite insertar el/los criterios por los que se puede realizar la búsqueda de una persona. | 1- Seleccionar la opción Buscar. |
| | 1.4: Mostrar la opción “Buscar” | Se muestra un botón con la opción Buscar. | 1- Seleccionar la opción Buscar. |
| | 1.5: Mostrar los resultados de la búsqueda en caso de ser positiva.<br>1.5.3 Seleccionar la persona deseada.<br>1.5.4 Mostrar los datos de la persona seleccionada. | Se muestran los resultados de la búsqueda.<br>Se muestra la persona seleccionada resaltada.<br>Se muestran los datos de la persona seleccionada.<br>a. Nombre(s)<br>b. Apellido(s)<br>c. Ciudadanía<br>d. Nacionalidad<br>e. Tipo de documento<br>f. No. documento<br>g. Fecha de nacimiento<br>h. Lugar de nacimiento<br>i. Raza<br>j. Sexo | 1- Dar un clic para que comience la búsqueda.<br>2- Dar enter.<br>3- Dar un clic sobre la persona |

| | | | |
|---|---|---|---|
| | | k. Ocupación<br>l. Profesión<br>m. Padre<br>n. Madre<br>o. Nivel cultural<br>p. Dirección particular<br>q. Foto | |
| | 1.7: Seleccionar la opción “Aceptar” si se desea. | Se muestran los datos de la persona seleccionada nada más. | 1- Seleccionar la opción Aceptar. |
| Buscar persona | 1.6: Mostrar mensaje informativo si el resultado de la búsqueda es negativo. | En caso de no encontrar ningún resultado en la búsqueda se muestra un mensaje informativo. | 1- Seleccionar la opción Buscar. |
| Buscar persona | 1.8: Seleccionar la opción “Cancelar” si se desea. | Se restablece el formulario. | 1- Seleccionar la opción Cancelar. |

**Tabla 4.1. Descripción del caso de prueba del RF 1Buscar persona.**

### Iteraciones de la prueba:

| ID del Escenario | Clases Válidas | Clases Inválidas | Resultado Esperado | Resultado Real de la Prueba | Observaciones |
|---|---|---|---|---|---|
| **1.5.1 Seleccionar la persona deseada.** | Clase válida: Debe seleccionarse al menos una persona. | Clase Inválida: No se seleccionó ninguna persona. Ejemplo: No se introdujo dato. | Se espera que se muestren los datos de una persona seleccionada, luego de haber mostrado otras personas con los datos | Clase válida: Se muestran los datos de la persona.<br><br>Clase inválida: No se muestra ningún dato, solamente las opciones Terminar y | |

| | | | | | |
|---|---|---|---|---|---|
| | | | similares. | Cancelar. | |

**Tabla 4.2. Iteraciones del caso de prueba del RF 1Buscar persona.**

### 4.8.4 Resultados de las pruebas.

Un buen resultado de las pruebas cuando son aplicadas al software es determinante en la etapa final del proceso de desarrollo de un sistema. Mediante ellas es posible comprobar el correcto funcionamiento de la aplicación tomando como referencia los requisitos funcionales establecidos. En la Figura 22 se muestra el resultado de la prueba unitaria descrita en el epígrafe 4.8.3 y en la Figura 23 se muestra el resultado de la prueba de sistema descrita en el epígrafe 4.8.2. Se realizaron dos iteraciones de pruebas de sistema que se muestran en la Figura 23, en la primera etapa se detectaron 10 no conformidades, en la segunda 5 y estas últimas fueron corregidas en una tercera iteración que aún se encuentra en etapa de revisión.

**Resultado de la prueba unitaria.**

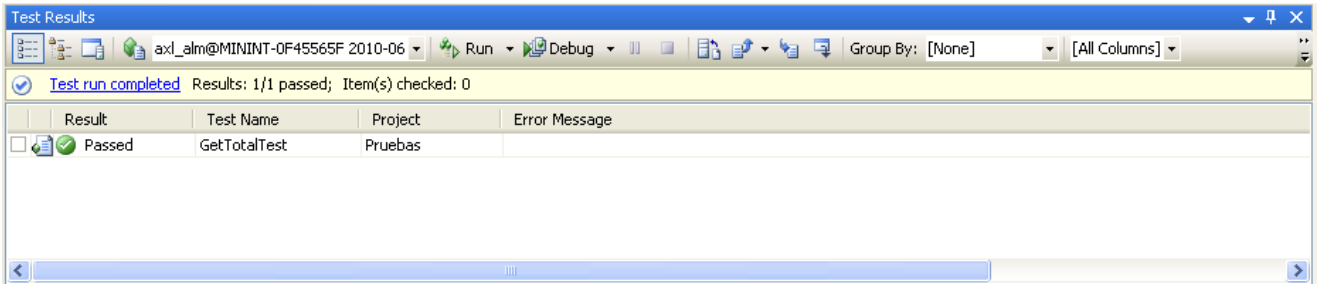



**Figura 22. Resultado de la prueba unitaria descrita en el epígrafe 4.8.3.**

**Resultados de las pruebas de sistema.**

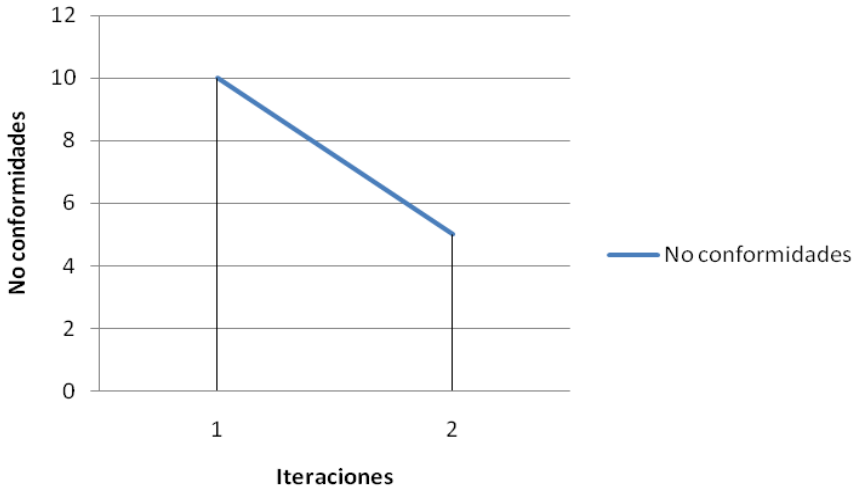



**Figura 23. Resultados de las pruebas de sistema.**

## 4.9 Análisis de los beneficios.

Al poner en práctica la aplicación aporta los siguientes beneficios:

- Eliminar pasos innecesarios en el proceso de negocio.
- Reducir la cantidad de documentación en papel que se maneja.
- Visualizar el expediente digital de la persona.
- Mayor agilidad en la búsqueda y registro de la información.

## 4.10 Conclusiones parciales.

El uso de estándares de codificación permitió un desarrollo común y organizado, la estrategia para el tratamiento de errores empleada evitó la introducción de datos incompletos e incorrectos al subsistema. Se realizó la distribución de componentes, la estructura del despliegue y la construcción del modelo entidad-relación. Se validaron las funcionalidades del Subsistema Huésped Casero para la Sección de Enfrentamiento del Departamento Policía de Inmigración a través de las pruebas unitarias y de sistema.

## CONCLUSIONES

Con el desarrollo de este trabajo de diploma se han cumplido los objetivos propuestos:

- El análisis y modelado de los procesos permitió alcanzar un amplio dominio del negocio que se estaba tratando, así como las reglas que definen el flujo de los mismos.
- Se identificaron cuatro requisitos generales y varios subrequisitos incluidos.
- Con el diseño e implementación del Subsistema Huésped Casero, se le proporciona a la Sección Enfrentamiento del Departamento Policía de Inmigración una herramienta para realizar el proceso gestión de estancias.

## RECOMENDACIONES

Para las siguientes versiones de la solución propuesta, se recomienda:

- Incluir la funcionalidad Gestionar casero que actualmente se desarrolla por Vivienda la Dirección de Identificación y Registro.
- Permitir la integración Subsistema Huésped Casero para la Sección Enfrentamiento del Departamento Policía de Inmigración con el módulo de reportes del Sistema de Inmigración, Extranjería y Ciudadanía.

## REFERENCIAS BIBLIOGRÁFICAS

1. **Rodríguez Peláez, Yadira Lázara.** Sistema para la informatización del proceso Control de Estancias en Hoteles para la Dirección de Migración y Fronteras de la República Bolivariana de Venezuela. 2009.

2. **Departamento de Informática, Comunicaciones y Cifras de la Direcció de Inmigración y Extranjería.** Manual del Sistema Huésped Casero.

3. DIGiriendo. *DIGiriendo.* [En línea] 30 de Junio de 2009. [Citado el: 3 de Febrero de 2010.] http://www.digiriendo.com/manejo-de-proyectos/msf-manejo-de-proyectos/que-es-msf-microsoft-solution-framework.

4. Intro Ingeniería de . *Intro Ingeniería .* [En línea] 12 de Noviembre de 2007. [Citado el: 3 de Febrero de 2010.] http://srivera334.blogspot.es/1195000200/.

5. Ignovan Blog. *Ignovan Blog.* [En línea] 1 de Noviembre de 2008. [Citado el: 3 de Febrero de 2010.] http://www.igovan.com/?p=16.

6. **Sánchez Barrientos, Manuel.** Apprender Gratis.com. *Aprender Gratis.com.* [En línea] 2 de Noviembre de 2008. [Citado el: 4 de Febrero de 2010.] http://www.aprendergratis.com/introduccion-a-bpmn.html.

7. Danysoft, Haciendo Visible lo Invisible. *Danysoft, Haciendo Visible lo Invisible.* [En línea] [Citado el: 4 de Febrero de 2010.] http://shop.danysoft.com/epages/danyshop_com.sf?ObjectPath=/Shops/danyshop_com/Products/%22Altova%20UModel%22/SubProducts/%22Altova%20UModel-0001%22.

8. Vagos.es. *Vagos.es.* [En línea] 17 de Agosto de 2008. [Citado el: 4 de Febrero de 2010.] http://www.vagos.es/showthread.php?t=320418.

9. **Blanco Zambrano, Reynier.** *Documento de Arquitectura del Proyecto Identificación, Migración y Extranjería de la República de Cuba.* 2010.

10. **Corporation, Microsoft.** MSDN. *MSDN.* [En línea] Junio de 2006. [Citado el: 6 de Febrero de 2010.] http://msdn.microsoft.com/en-us/library/aa697427%28VS.80%29.aspx.

11. Solo Disenio.com. *Solo Disenio.com.* [En línea] 29 de Julio de 2006. [Citado el: 4 de Febrero de 2010.] http://www.solodisenio.com/net-de-microsoft/.

12. **Patxi.** eslomas.com. *eslomas.com.* [En línea] 11 de Mayo de 2005. [Citado el: 4 de Febrero de 2010.] http://www.eslomas.com/index.php/archives/2005/05/11/introduccion-plataforma-net-y-mono/.

13. Sistema de Inventario Web. Proyecto ELO-330: Programación de Sistemas. *Sistema de Inventario Web. Proyecto ELO-330: Programación de Sistemas.* [En línea] [Citado el: 8 de Abril de 2010.] http://profesores.elo.utfsm.cl/~agv/elo330/2s08/projects/CaroDiazToro/tecnologia.html.

14. **Juan Antonio Breña Moral.** Web Taller Com*. Web Taller Com.* [En línea] [Citado el: 8 de Abril de 2010.] http://www.webtaller.com/maletin/articulos/que_son_web_services.php.

15. **James Garret, Jesse.**Maestros del Web. *Maestros del Web.* [En línea] 11 de Junio de 2005. [Citado el: 8 de Abril de 2010.] http://www.maestrosdelweb.com/editorial/ajax/.

16. Facebook. *Facebook.* [En línea] [Citado el: 8 de Abril de 2010.] http://es.facebook.com/pages/ASPNET/109271279091557.

17. FOROS.COM.VE: Foros Venezuela. *FOROS.COM.VE: Foros Venezuela.* [En línea] 10 de Septiembre de 2007. [Citado el: 8 de Abril de 2010.] http://www.foros.com.ve/post121488.html.

18. **Valenzuela, Juan Andres.**Mosca.org.*Mosca.org***.** [En línea] 20 de Agosto de 2007. [Citado el: 8 de Abril de 2010.] http://www.mossca.org/Documentos%20compartidos/Forms/AllItems.aspx?RootFolder=http%3a%2f%2fwww.mossca.org%2fDocumentos%20compartidos%2fDESARROLLO%20%28Development%29&FolderCTID=0x0120003A18DBE51089CE46AE85D4EF57D15831

19. **Pestrepo, Josse.** Facebook. *Facebook* [En línea] 16 de Julio de 2008. [Citado el: 10 de Abril de 2010.] http://www.microsoft.com/spain/interop/indigo.mspx.

20. Enciclopedia Libre Universal en Español. *Enciclopedia Libre Universal en Español* [En línea] 11 de Diciembre de 2008. [Citado el: 10 de Abril de 2010.] http://enciclopedia.us.es/index.php/Lenguaje_de_programaci%C3%B3n

21. FlupsNet. *FlupsNet* [En línea] [Citado el: 10 de Abril de 2010.] http://flups.net/e-books-y-tutoriales-f11/c-sharp-el-lenguaje-del-futuro-t515029.html.

22. Css Plantillas Recursos web. *Css Plantillas Recursos web.* [En línea] [Citado el: 10 de Abril de 2010.] http://www.cssplantillas.es.tl/glosario.htm.

23. Glosario. *Glosario.* [En línea] [Citado el: 10 de Abril de 2010.] http://www.codebox.es/glosario.

24. masadelante.com. *masadelante.com.* [En línea] [Citado el: 10 de Abril de 2010.] http://www.masadelante.com/faqs/css.

25. danysoft.com. *danysoft*. [Online] http://www.danysoft.com/bol/visualstudio2008.htm.

26. Territorio Scuola. *Territorio Scuola.* [En línea] [Citado el: 10 de Abril de 2010.] http://www.territorioscuola.com/wikipedia/es.wikipedia.php?title=Oracle_11g.

27. Scribd. *Scribd.* [En línea] 21 de Junio de 2009. [Citado el: 10 de Abril de 2010.] http://www.scribd.com/doc/16626706/Embarcadero-ER.

28. MSDN. *MSDN.* [En línea] Noviembre de 2007. [Citado el: 10 de Abril de 2010.] http://msdn.microsoft.com/es-es/library/bb397926.aspx.

27. Dale Gas: Somos Uruguay. *Microsoft Visual Studio 2008.* [En línea] 30 de Marzo de 2010. [Citado el: 11 de Abril de 2010.] http://www.dalegas.com.uy/windows/microsoft-visual-studio-2008-professional-%28con-serial%29/?PHPSESSID=nbmdadbm552dvtcsmvuoji6ks7.

29. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de .* 2000.

30. **Pressman, Roger.** *Ingenieria del , un enfoque práctico.* 2002.

31. **Ramírez, Isreal.** Universidad de los Andes. *Facultad de Ciencias Económicas y Sociales.* [En línea] [Citado el: 2010 de Abril de 11.] http://isis.faces.ula.ve/computacion/Israel/Bases.pdf.

32. **Blanco Zambrano, Reynier**. *Documento de Estándaresy Tratamiento de Errores.* 2010.

33. **Morales, Luisa**. Baquia knowledge center. *Baquia knowledge center*. [En línea] 13 de Junio de 2005.[Citado el: 2010 de Abril de 11.] http://www.baquia.com/noticias.php?id=9778

34. monografias.com. *monografias.com.* [En línea] [Citado el: 2010 de Abril de 11.] http://www.monografias.com/trabajos75/proyectos-informaticos/proyectos-informaticos3.shtml

35. Scribd. *Scribd*. [En línea] 14 de abril 2010 [Citado el: 2010 de Abril de 11.] http://www.scribd.com/doc/29908555/prueba-de-

36. **Fernández Peña, Juan Manuel**. [En línea] [Citado el: 2010 de Abril de 11.] http://www.uv.mx/jfernandez/cursos_archivos%5CPruebas%20de%20sistema.pdf

37. IHMC Public Cmaps. *IHMC Public Cmaps.* [En línea] [Citado el: 2010 de Abril de 11.] http://cmapspublic.ihmc.us/

## BIBLIOGRAFÍA CONSULTADA


1. **Española, Diccionario Manual de la Lengua.** Diccionario Manual de la Lengua Española. [Online] 2007. [Cited: Enero 26, 2010.]
2. **AM, Alejandro Afonso Spinola.** .net for your information. [Online] Agosto 4, 2009. http://dotnetfyi.wordpress.com/2009/08/04/utilidad-de-windows-workflow-foundation/.
3. **msdn.microsoft.com.** msdn. [Online] http://msdn.microsoft.com/es-es/library/kx37x362.aspx.
4. **qualitrain.com.** [Online] http://www.qualitrain.com.mx/ventajas-y-Desventajas-de-.NET/Page-5.htmlVentajas y Desventajas de .NET.
5. **Avanzis.** Todoexpertos.com. [Online] 2010. http://www.todoexpertos.com/categorias/tecnologia-e-internet/comercio-electronico/respustas/252736/caracteristicas-oracle.

## GLOSARIO DE TÉRMINOS

A continuación se muestra organizado alfabéticamente el significado de algunas palabras usadas en este documento que pueden tener problemas para su comprensión dado su poco uso.

- **Agnóstico:** El término "agnóstico" se deriva de la palabra griega "agnostos," que significa "no saber." Un agnóstico es aquel que admite: "No sé." El término es aplicado específicamente a aquellos quienes no saben con seguridad si Dios existe o no.
- **Áreas de atención:** Son las áreas que atienden los Inspectores de Enfrentamiento y los Grupos de Enfrentamiento a los que estos pertenecen.
- **Arquitectura de software:** La arquitectura de software establece los fundamentos para que analistas, diseñadores y programadores trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.
- **Casos de prueba:** Es un formulario con una serie de actividades para comprobar las funcionalidades y observar las respuestas que se obtienen.
- **CIRP:** Carné de Identidad y Registro de la Población.
- **Código fuente:** El código fuente de un programa informático (o ) es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa. Por tanto, en el código fuente de un programa está descrito por completo su funcionamiento.
- **Código máquina:** El código binario es el sistema de representación de textos, o procesadores de instrucciones de ordenador, utilizando el sistema binario (sistema numérico de dos dígitos, o *bit*: el "0" y el "1").
- **Control territorial:** Inspecciones que se realizan en las áreas de atención con el objetivo de detectar posibles violaciones por parte de extranjeros o nacionales.
- **Departamento Policía de Inmigración (DPI):** Departamento que forma parte de la DIE y es el encargado de la inmigración en el país.
- **Dirección de Inmigración y Extranjería (DIE):** Órgano responsable e de las normativas legales relacionadas, con la migración, extranjería y ciudadanía en Cuba.
- **Engorroso:** Algo difícil, molesto de realizar.
- **Estancia:** Es cuando un extranjero decide hospedarse en la casa de un nacional con o sin acompañante.
- **Flujo de trabajo:** El flujo de trabajo (*workflow* en inglés) es el estudio de los aspectos operacionales de una actividad de trabajo: cómo se estructuran las tareas, cómo se realizan, cuál

es su orden correlativo, cómo se sincronizan, cómo fluye la información que soporta las tareas y cómo se le hace seguimiento al cumplimiento de las tareas.

- ***Framework*:** La palabra inglesa *framework* define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.
- ***Hardware*:** Corresponde a todas las partes físicas y tangibles de una computadora: sus componentes eléctricos, electrónicos, electromecánicos y mecánicos; sus cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado.
- ***Hyperlinks*:** Son los elementos que asignan a un trozo de texto la facultad de que conecte o otro objeto (texto, imagen).
- **Ingeniería inversa:** Esta tipo de ingeniería se realiza cuando se ha realizado la implementación de un sistema o aplicación y luego es que se realiza el diseño de la misma y otras etapas.
- **Lenguaje de *scripting*:** Un lenguaje interpretado es un lenguaje de programación que está diseñado para ser ejecutado por medio de un intérprete.
- **Levantamiento de requisitos:** Se definen las funcionalidades y cualidades de un sistema o aplicación informática.
- **Masa extranjera:** Extranjeros y cubanos residentes en el exterior, que se encuentren de visita en Cuba.
- **Metodología:** La metodología, es una de las etapas específicas de un trabajo o proyecto que nace a partir de una posición teórica y conlleva a una selección de técnicas concretas (o métodos) de cómo se va a realizar la investigación.
- ***Microsoft*:** *Microsoft Corporation* es una empresa multinacional estadounidense, fundada en 1975 por *Bill Gates* y *Paul Allen*. Dedicada al sector de la informática. *Microsoft* desarrolla, fabrica, licencia y produce software y equipos electrónicos.
- **Modelo en cascada:** En Ingeniería de software el desarrollo en cascada, también llamado modelo en cascada, es el enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior.
- **Modelo en espiral:** Desarrollo en espiral es un modelo de ciclo de vida del software, utilizado generalmente en la Ingeniería de software. Las actividades de este modelo se conforman en una espiral, en la que cada *bucle* o iteración representa un conjunto de actividades. Las actividades no

están fijadas a priori, sino que las siguientes se eligen en función del análisis de riesgo, comenzando por el *bucle* interior.

- **Proceso:** Un proceso (del latín *processus*) es un conjunto de actividades o eventos (coordinados u organizados) que se realizan o suceden (alternativa o simultáneamente) con un fin determinado. Este término tiene significados diferentes según la rama de la ciencia o la técnica en que se utilice.
- **Procesos de negocio:** Son los conjuntos de actividades que se realizan en la actualidad en un Ambiente de Trabajo, dígase oficina, empresa o departamento.
- **Protocolo:** Un protocolo es un conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red. Un protocolo es una convención o estándar que controla o permite la conexión, comunicación, y transferencia de datos entre dos puntos finales.
- **Requerimiento:** Condición o capacidad que tiene que ser alcanzada por un sistema o los componentes de un sistema para satisfacer un contrato o documento formalmente.
- **Requerimiento funcional:** Capacidades o condiciones que el sistema debe cumplir.
- **Requerimiento no funcional:** Propiedades o cualidades que el sistema debe tener.
- **Situación problémica:** Es el conjunto de problemas que se presentan en un negocio que se analiza para realizar mejoras en sus procesos.
- **Sección de Enfrentamiento:** Es una de las partes del DPI y se encarga del control territorial de la masa extranjera en Cuba.
- **Software:** Se refiere al equipamiento lógico o soporte lógico de una computadora digital, y comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de tareas específica.
- **UML:** Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modelling Language*) es el lenguaje de modelado de sistemas de software más conocido en la actualidad; aún cuando todavía no es un estándar oficial.
- Usuario: Un usuario es la persona que utiliza o trabaja con algún objeto o que es destinataria de algún servicio público, privado, empresarial o profesional.