

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS  
FACULTAD 1

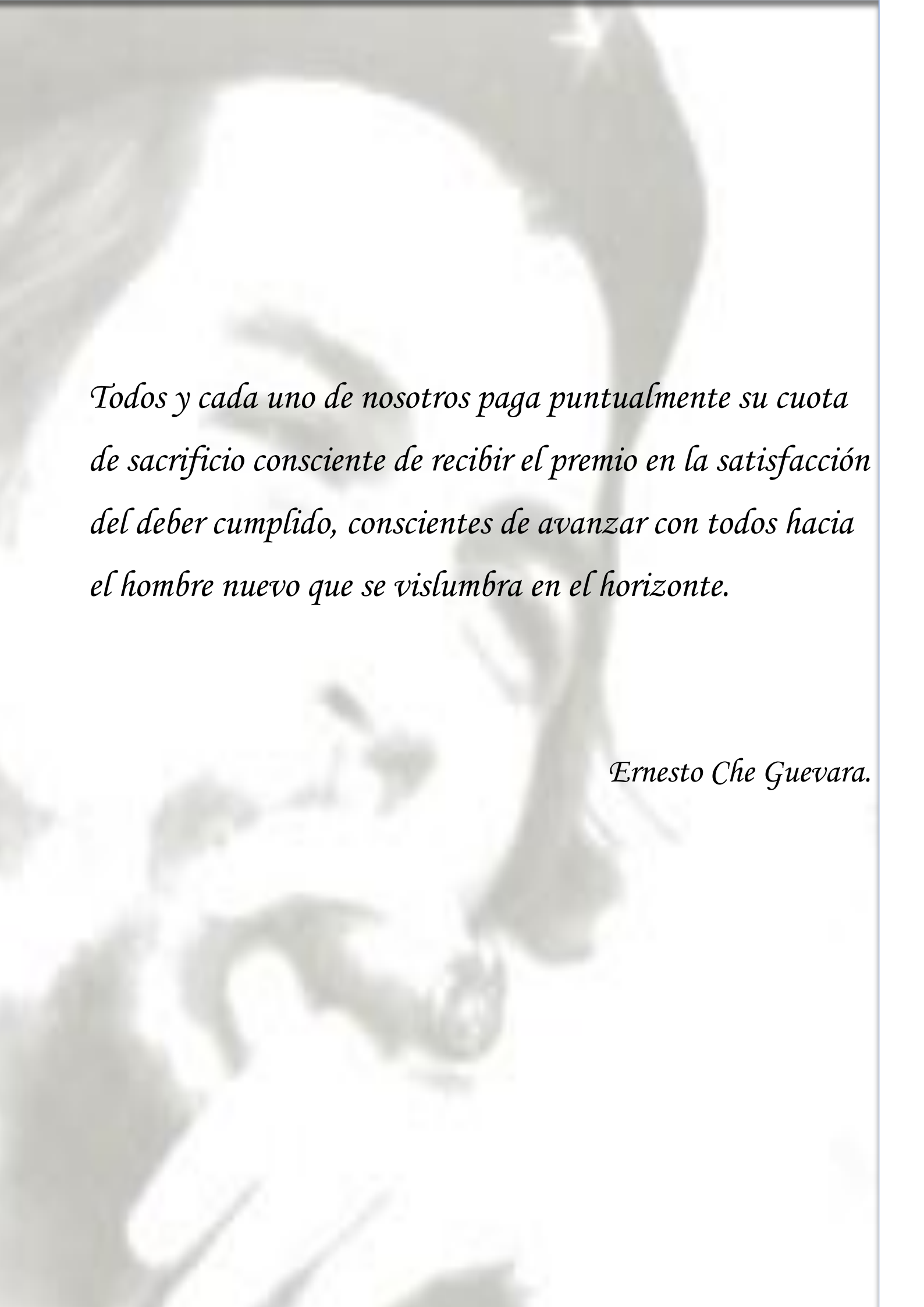
*Trabajo de Diploma para optar por el*  
*Título de*  
*Ingeniero en Ciencias Informáticas*

**Título:** Solución *web* para pruebas de algoritmos de identificación biométrica sobre huellas dactilares.

**Autor(es):** René Miní Rodríguez.  
Lieter Piñero Bermudez.

**Tutor:** Ing. Yainier Labrada Nueva.

Ciudad de La Habana. Junio, 2012



*Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el hombre nuevo que se vislumbra en el horizonte.*

*Ernesto Che Guevara.*

## **Declaración de autoría**

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 1 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los \_\_\_\_ días del mes de \_\_\_\_\_ del año

\_\_\_\_\_.

\_\_\_\_\_  
Lieter Piñero Bermudez

\_\_\_\_\_  
René Miní Rodríguez

\_\_\_\_\_  
Ing. Yainier Labrada Nueva

## Resumen

En la actualidad las aplicaciones biométricas tienen un gran auge e importancia, debido que al emplear y utilizar estas técnicas se muestran resultados certeros y eficientes, por ello en gran parte los sistemas de seguridad a nivel mundial se utilizan patrones biométricos para su ejecución

El Centro de Identificación y Seguridad Digital (CISED) está involucrado en el despliegue de plataformas y soluciones biométricas. Actualmente en el CISED se cuenta con una aplicación de escritorio que posibilita probar algoritmos biométricos. La misma no cuenta con un entorno *web* para que las pruebas sean realizadas en línea, de ahí que la presente investigación surge como resultado de la necesidad de la elaboración de una plataforma o solución *web* para algoritmos biométricos de identificación dactilar. Como resultado positivo los usuarios que deseen probar sus algoritmos lo podrán hacer de forma fácil, rápida y también observar sus resultados en línea.

Es necesario disponer de una aplicación *web* para probar algoritmos de identificación dactilar en la universidad y en el país. Para el logro de este objetivo se realizó la implementación de una plataforma *web* titulada: **“Solución Web para Pruebas de Algoritmos de Identificación Biométrica sobre huellas dactilares”**, en la cual el usuario pueda probar algoritmos biométricos de identificación dactilar, además de mostrar sus resultados con una interfaz amigable y verlos publicados con los reportes que se generan y las gráficas correspondientes.

Palabras clave:

Aplicaciones biométricas, algoritmo biométrico, pruebas, plataforma *web*.

## Índice

Introducción .....	1
Capítulo1: Fundamentación teórica.....	5
1. Introducción.....	5
1.1 Algoritmo biométrico.....	5
1.2 Prueba de precisión a un algoritmo biométrico.....	5
1.2.1 Precisión de un sistema biométrico .....	5
1.2.2 Fases del procedimiento biométrico .....	6
1.3 Reconocimiento biométrico .....	6
1.4 Verificación biométrica.....	6
1.5 Identificación biométrica .....	9
1.6 Punto de operación (OP).....	11
1.7 Curva característica operativa del receptor .....	11
1.8 Estándares biométricos internacionales .....	12
1.8.1 Interfaz de programación de aplicaciones .....	13
1.8.2 Interfaz de programación estándar (BAPI) .....	13
1.8.3 Formato común de ficheros para el intercambio biométrico (CBEFF) .....	14
1.8.4 Estándar ANSIX9.84 .....	14
1.9 Estado de los sistemas de evaluación biométrica a nivel internacional.....	16
1.9.1 NIST .....	16
1.9.2 Reconocimiento facial gran desafío (FRGC).....	16
1.9.3 FRVT.....	17
1.9.4 Campaña de evaluación multimodal de bioseguridad del 2007 (BMEC'2007) .....	17
1.9.5 FVC- OnGoing.....	18
1.10 Metodología, herramientas y lenguajes empleados.....	19
1.10.1 Proceso Unificado de Rational (RUP) .....	19
1.10.2 Lenguaje Unificado de Modelado (UML).....	21
1.10.3 Herramienta de modelado. <i>Visual Paradigm</i> v6.4 .....	22
1.10.4 Lenguaje de programación.....	23
1.10.4.1 Herramienta <i>Visual Studio Ultimate</i> 2010 .....	24
1.10.5 ASP.NET MVC 2.....	25

1.10.6	Base de datos.....	26
1.10.6.1	Base de datos de referencia .....	27
1.10.6.2	SQL Server Express 2008.....	28
1.10.6.3	Lenguaje integrado de consultas (LinQ) .....	29
1.10.6.4	ADO.NET Entity Framework .....	29
1.10.7	Artisteer .....	30
1.10.8	Librería telerik.....	30
1.10.9	JavaScript .....	31
1.10.10	Librería Jqplot.....	32
1.10.11	Librería JQuery.....	33
1.10.12	Macromedia Flash 8.....	33
1.10.13	Fireworks.....	34
1.10.14	Hojas de estilo en cascada (CCS) .....	34
1.10.15	Lenguaje de Marcación de Hipertexto (HTML) .....	35
1.11	Conclusiones .....	35
Capítulo 2: Características del sistema.....		36
2.1	Introducción .....	36
2.3	Modelo de dominio .....	36
2.4	Requisitos .....	38
2.4.1	Requisitos funcionales .....	38
2.4.2	Requisitos no funcionales.....	40
2.5	Diagrama de caso de uso del sistema .....	40
2.6	Especificaciones de casos de uso .....	41
2.7	Modelo de análisis .....	48
2.7.1	Diagrama de clases del análisis .....	49
2.7.2	Diagramas de colaboración.....	53
2.8	Conclusiones .....	55
Capítulo 3: Elaboración y construcción del sistema.....		56
3.1	Introducción .....	56
3.2	Diseño del sistema .....	56

3.3	Modelo de clases del diseño.....	56
3.4	Diagramas de secuencia del diseño.....	59
3.5	Modelo de datos.....	63
3.6	Descripción de la arquitectura .....	63
3.7	Patrones del diseño.....	64
3.8	Diagrama de componentes .....	67
3.9	Modelo de despliegue.....	68
3.10	Pruebas del sistema.....	69
3.10.1	Pruebas de caja blanca .....	70
3.10.2	Pruebas de caja negra.....	76
	Conclusiones Generales.....	82
	Recomendaciones.....	83
	Glosario de términos .....	84
	Referencias bibliográficas .....	85
	Bibliografía.....	87
	Anexos.....	89
	Anexo 1: Prototipo de interfaz de usuario para el caso de uso Registrar usuario. ....	89
	Anexo 2: Prototipo de interfaz de usuario para el caso de uso Realizar autenticación. ....	89
	Anexo 3: Prototipo de interfaz de usuario para el caso de uso Realizar prueba.....	90
	Anexo 4: Prototipo de interfaz de usuario para el caso de uso Realizar reporte de pruebas. ....	90
	Anexo 4.1: Prototipo de interfaz en el caso que el usuario acceda ver todos los reportes. ....	90
	Anexo 4.2: Prototipo de interfaz en el caso que el usuario acceda ver los reportes del usuario seleccionado.....	91
	Anexo 4.3: Prototipo de interfaz en el caso que el usuario acceda las gráficas generadas (FAR, FRR, FAR vs FRR, ROC y los Resultados). ....	91
	Anexo 5: Prototipo de interfaz para el caso de uso Gestionar base de datos. ....	94
	Anexo 5.1: Prototipo de interfaz para el caso de uso Gestionar base de datos sección adicionar..	94
	Anexo 5.2: Prototipo de interfaz para el caso de uso Gestionar base de datos sección eliminar. ....	95
	Anexo 6: Modelo de datos.....	96

## Índice de figuras

Figura 1: Proceso de Captura y Verificación de usuario .....	7
Figura 2: Gráfica de tasa de error igual. ....	8
Figura 3: Curva FAR vs FRR. ....	9
Figura 4: Curva característica operativa del receptor. ....	12
Figura 5: Ecuación ROC.....	12
Figura 6: Ejemplo de una gráfica en <i>Jqplot</i> . ....	32
Figura 7: Modelo de dominio .....	37
Figura 8: Diagrama de casos de uso del sistema.....	41
Figura 9: Diagrama de clases del análisis para el caso de uso Registrar usuario.....	51
Figura 10: Diagrama de clases del análisis para el caso de uso Realizar autenticación.....	51
Figura 11: Diagrama de clases del análisis para el caso de uso Realizar prueba.....	52
Figura 12: Diagrama de clases del análisis para el caso de uso Realizar reporte de pruebas. ....	52
Figura 13: Diagrama de clases del análisis para el caso de uso Gestionar base de datos. ....	53
Figura 14: Diagrama de colaboración para el caso de uso Registrar usuario. ....	54
Figura 15: Diagrama de colaboración para el caso de uso Realizar autenticación. ....	54
Figura 16: Diagrama de colaboración para el caso de uso Realizar prueba. ....	54
Figura 17: Diagrama de colaboración para el caso de uso Realizar reporte de pruebas. ....	55
Figura 18: Diagrama de colaboración para el caso de uso Gestionar base de datos (escenario 1). ....	55
Figura 19: Diagrama de colaboración para el caso de uso Gestionar base de datos (escenario 2). ....	55
Figura 20: Patrón Modelo Vista Controlador. ....	64
Figura 21: Diagrama de clases del diseño para el caso de uso Registrar usuario. ....	57
Figura 22: Diagrama de clases del diseño para el caso de uso Realizar autenticación. ....	57
Figura 23: Diagrama de clases del diseño para el caso de uso Realizar prueba. ....	58
Figura 24: Diagrama de clases del diseño para el caso de uso Realizar reporte de pruebas. ....	58
Figura 25: Diagrama de clases del diseño para el caso de uso Gestionar base de datos. ....	59
Figura 26: Diagrama de secuencia del diseño para el caso de uso Registrar usuario. ....	60
Figura 27: Diagrama de secuencia del diseño para el caso de uso Realizar autenticación. ....	60
Figura 28: Diagrama de secuencia del diseño para el caso de uso Realizar prueba.....	61
Figura 29: Diagrama de secuencia del diseño para el caso de uso Realizar reporte de pruebas. ....	61
Figura 30: Diagrama de secuencia del diseño para el caso de uso Gestionar base de datos (escenario 1). ....	62
Figura 31: Diagrama de secuencia del diseño para el caso de uso Gestionar base de datos (escenario 2). ....	62
Figura 32: Representación del patrón Creador en la aplicación. ....	65
Figura 33: Representación del patrón Controlador en la aplicación.....	66
Figura 34: Representación del patrón Bajo acoplamiento en la aplicación.....	67
Figura 35: Diagrama de componentes. ....	68
Figura 36: Modelo de despliegue.....	69
Figura 37: Prueba de caja blanca para la funcionalidad <i>RunBiometricSystem</i> . ....	71



Figura 38: Grafo de flujo para la funcionalidad <i>RunBiometricSystem</i> .....	71
Figura 39: Prueba de caja blanca para la funcionalidad <i>Register</i> . ....	72
Figura 40: Grafo de flujo para la funcionalidad <i>Register</i> . ....	73
Figura 41: Prueba de caja blanca para la funcionalidad <i>LoadFile</i> .....	74
Figura 42: Grafo de flujo para la funcionalidad <i>LoadFile</i> .....	74
Figura 43: Gráfico de resultados de las 4 iteraciones de pruebas unitarias realizadas hasta el momento. .....	75
Figura 44: Resultados de las 4 iteraciones de pruebas de validación realizadas hasta el momento. ....	81

## Índice de tablas

Tabla 1: Actores del sistema.....	42
Tabla 2: Descripción del caso de uso 1. ....	42
Tabla 3: Descripción del caso de uso 2. ....	43
Tabla 4: Descripción del caso de uso 3. ....	44
Tabla 5: Descripción del caso de uso 4. ....	45
Tabla 6: Descripción del caso de uso 5. ....	46
Tabla 7: Complejidad ciclomática para la funcionalidad <i>RunBiometricSystem</i> . ....	71
Tabla 8: Caminos independientes obtenidos y casos de pruebas que obliga a la ejecución de cada camino. ....	72
Tabla 9: Complejidad ciclomática para la funcionalidad <i>Register</i> .....	73
Tabla 10: Caminos independientes obtenidos y casos de pruebas que obliga a la ejecución de cada camino. ....	73
Tabla 11: Complejidad ciclomática para la funcionalidad <i>LoadFile</i> . ....	74
Tabla 12: Caminos independientes obtenidos y casos de pruebas que obliga a la ejecución de cada camino. ....	75
Tabla 13: Prueba de caja negra para el caso de uso Registrar usuario.....	76
Tabla 14: Descripción de las variables para el caso de prueba Registrar usuario. ....	77
Tabla 15: Prueba de caja negra para el caso de uso Realizar autenticación. ....	77
Tabla 16: Descripción de las variables para el caso de prueba Realizar autenticación. ....	78
Tabla 17: Prueba de caja negra para el caso de uso Realizar prueba. ....	78
Tabla 18: Descripción de las variables para el caso de prueba Realizar prueba. ....	79
Tabla 19: Prueba de caja negra para el caso de uso Realizar reporte de pruebas.....	80
Tabla 20: Descripción de las variables para el caso de prueba Realizar reporte de pruebas. ....	80

## Introducción

En los últimos años la biometría, muestra un crecimiento que se manifiesta desde el uso de la huella dactilar, hasta el empleo de los más complejos métodos de identificación teniendo en cuenta varias medidas físicas y de comportamiento biopsicosocial. Las aplicaciones de la biometría evolucionan y trascienden la simple identificación hasta sistemas de seguridad más diversos por su composición. (1)

El término biometría clásicamente se aplicaba de forma general a la ciencia que se dedica al estudio estadístico de las características cuantitativas de los seres vivos: peso, longitud, entre otras. Sin embargo, en épocas más recientes este concepto se utiliza también para referirse a los métodos automáticos que analizan determinadas características humanas con el fin de identificar y autenticar a personas. (1)

El concepto tradicional de la biometría se refiere a la aplicación de las técnicas matemáticas y estadísticas para el análisis de datos en las ciencias biológicas. En el contexto tecnológico no es más que la aplicación automatizada de técnicas biométricas como son: autenticación e identificación de personas en sistemas de seguridad. (2)

Los rasgos biométricos se agrupan en dos conjuntos: los de características físicas (estáticas), y de comportamiento (dinámicas). Pertenecen al primer grupo la huella dactilar, la retina, el iris, la geometría de la palma de la mano, los rasgos faciales, mientras que en el segundo se hallan la firma, el tecleo y la forma de caminar. La voz es una mezcla de características físicas y de comportamiento, además cada rasgo biométrico tiene en común aspectos físicos y de comportamiento.

Los países desarrollados a partir de los notorios adelantos que vislumbran tanto en la industria, como en la ciencia y en la tecnología, se erigen como los pilares del desarrollo de los sistemas biométricos, sin embargo el uso de estas técnicas no es exclusivo de estas naciones. Cuba a pesar de ser un país subdesarrollado no está ajena al conocimiento, ni al uso e implementación de éstas. El país transita por el camino de la industrialización y el avance informático, ya sea en la implementación de estas costosas técnicas o en la producción y exportación de *software*.

Se han creado centros de investigaciones y universidades que indagan y trabajan en estos aspectos informáticos y tecnológicos. La Universidad de las Ciencias Informáticas (UCI) es uno de estos ejemplos, cuyo objetivo principal consiste en desarrollar la industria del *software* y de esta forma contribuir al desarrollo económico del país. La UCI está constituida por 7 facultades y centros de

desarrollo de *software* asociados a estas facultades, un ejemplo es el Centro de Identificación y Seguridad Digital (CISED) que pertenece a la facultad 1, el cual está dividido en diferentes departamentos: Tarjetas Inteligentes, Identificación, Seguridad Digital, Soluciones Integrales y Biometría.

El departamento de Biometría, se encarga del desarrollo de componentes, productos y soluciones relacionados con el procesamiento digital de imágenes en el campo de la identificación y seguridad. En el mismo se realizan procesos de pruebas a los algoritmos biométricos. Actualmente existe una plataforma para realizar las pruebas a los algoritmos desarrollados en el CISED, pero esta no cuenta con un entorno para que las pruebas sean realizadas en línea, por lo que los usuarios que deseen probar sus algoritmos, tienen que dirigirse personalmente a los encargados de realizar este proceso. Además no se cuenta con una capa permita gestionar la conexión con distintas bases de datos de referencia y no se permite el acceso a la plataforma de diferentes lugares.

Debido a lo anteriormente expuesto surge el siguiente **problema de investigación**: ¿Cómo mejorar el proceso de pruebas de precisión de los algoritmos biométricos realizados en el CISED?

Se precisa como **objeto de estudio**, los sistemas de evaluación y control de los algoritmos biométricos.

Para dar solución a la problemática planteada se define como **objetivo general** de la investigación: Desarrollar una solución *web*, en el (CISED) para la plataforma pruebas de precisión de algoritmos biométricos. Derivándose los siguientes **objetivos específicos**:

1. Caracterizar los sistemas existentes en la actualidad que realicen pruebas de algoritmos biométricos.
2. Realizar un análisis de las herramientas, metodologías y lenguajes a utilizar.
3. Realizar la elaboración y construcción del sistema.
4. Implementar una plataforma de pruebas para algoritmos biométricos en la cual los usuarios puedan realizar pruebas y observar sus resultados.
5. Realizar pruebas de caja blanca y caja negra al sistema.

Teniendo como **campo de acción**, los sistemas de evaluación y control de los algoritmos biométricos desarrollados en el CISED.

Para dar cumplimiento total a los objetivos se definen las siguientes **tareas de investigación**:

1. Realización del análisis de la documentación asociada a la Plataforma de pruebas de algoritmos biométricos.
2. Caracterización y comparación de las Plataformas de pruebas existentes.
3. Fundamentación de las tecnologías, herramientas, metodología y lenguaje necesarios para el desarrollo de la aplicación.
4. Identificación de los requisitos funcionales y no funcionales del sistema.
5. Realización de un diseño de una arquitectura que exponga las funcionalidades para que sea integrable a otras aplicaciones desarrolladas en CISED.
6. Implementación de una Solución para Pruebas de Algoritmos de Identificación Biométrica sobre huellas dactilares en línea.
7. Realización de pruebas de caja blanca y caja negra al sistema.

Entre los **métodos científicos teóricos** utilizados en la investigación se encuentran:

**Analítico – Sintético**: permitió realizar un análisis de las teorías y documentos existentes, posibilitando una amplia indagación con todo lo relacionado con los diferentes tipos de algoritmos biométricos y permitiendo la extracción de datos importantes para esta investigación relacionados de forma general con el objeto de estudio definido.

**Inductivo – Deductivo**: permite a partir de conocimientos generales, inferir casos particulares por un razonamiento lógico, así como la elaboración de conclusiones a partir de conocimientos adquiridos de los procesos de pruebas analizados.

Por otra parte el **método empírico** utilizado para explicar las características fenomenológicas del objeto fue el de **observación** para obtener información acerca de las plataformas de pruebas existentes y así investigar los diferentes procesos de manera externa.

**Justificación de la investigación**: con el desarrollo de la presente investigación se pretende proporcionar al departamento Biometría del CISED, la UCI en general y todas las entidades que

utilicen algoritmos de este tipo, una plataforma en línea para algoritmos biométricos, en la cual se puedan probar los algoritmos biométricos de forma fácil y entendible para los usuarios.

La investigación está estructurada en tres capítulos:

**Capítulo 1 Fundamentación teórica:** este capítulo aborda el respaldo teórico de los temas tratados en el informe, necesarios para el entendimiento correcto de la solución planteada. Se describen los conceptos fundamentales asociados al dominio del problema, el objeto de estudio, haciéndose un análisis de la situación actual. Se incluye el estado del arte de la investigación, la fundamentación de la metodología, herramientas y tecnologías utilizadas para el diseño del sistema.

**Capítulo 2 Características del sistema:** en este capítulo se expondrán los requisitos funcionales y no funcionales con los cuales el sistema debe cumplir para satisfacer las necesidades del problema planteado, además de brindar una descripción detallada de los actores y casos de uso involucrados en el sistema. Se realiza una descripción a través de un modelo de dominio que permite mostrar al usuario los principales conceptos que manejan en el dominio de la aplicación en desarrollo, a partir de este, se comienza hacer el análisis del sistema a desarrollar. Se muestra el diagrama de clases del análisis para cada uno de los casos de uso del sistema, así como el diagrama de colaboración del análisis.

**Capítulo 3 Elaboración y construcción del sistema:** este capítulo tiene como objetivo mostrar la descripción del sistema implementado a través del diagrama de componentes y se visualiza el diagrama de despliegue con la distribución de sus nodos necesario para el despliegue del sistema. Se realiza un modelo de datos que describe la estructura de la base de datos y los elementos que intervienen. Se modelan los diagramas de clases del diseño y secuencia del diseño. Además se definen los principios de diseño de la interfaz *web* así como los patrones de diseño empleados para beneficiar el desarrollo y la calidad del *software*.

# Capítulo 1: Fundamentación teórica

## 1. Introducción.

En el presente capítulo se describen conceptos que servirán de base para el resto del documento. Se realiza un estudio del estado del arte, profundizando en los principales estándares biométricos internacionales. Se hará énfasis en el estudio de los lenguajes, tecnologías, herramientas y la metodología en las que se apoya el desarrollo del sistema, en función de las tendencias actuales.

En el presente epígrafe se describen los conceptos básicos que son necesarios para llevar a cabo la investigación, son muy importantes debido a que servirán de base para el entendimiento posterior del análisis del capítulo 1.

### 1.1 Algoritmo biométrico

**Un algoritmo biométrico** es una secuencia de instrucciones que indican a un sistema biométrico cómo resolver un problema particular. Los sistemas biométricos utilizan secuencias de reglas para interpretar los datos que han sido extraídos de la fuente original. No es más que el conjunto finito de reglas e instrucciones bien definidas y ordenadas, que permiten comprobar la identidad de una persona mediante una característica estática o dinámica que pueda ser reconocida o verificada de una manera automática. (3)

### 1.2 Prueba de precisión a un algoritmo biométrico

Partiendo del concepto de **prueba**: que es un examen o experimentación para comprobar el buen funcionamiento de alguna cosa o de su adecuación a un determinado fin, (4) y del concepto de **precisión**: que no es más que la exactitud con la que se realiza una medición o cálculo dados. Se puede concluir que las pruebas de precisión a un algoritmo biométrico no son más que una especie de experimentos que se realizan con el fin de determinar la exactitud de los valores obtenidos tras realizarle ensayos a los mismos. (5)

#### 1.2.1 Precisión de un sistema biométrico

La precisión de un sistema biométrico es determinada mediante una serie de pruebas, primero, una evaluación de la precisión del algoritmo de coincidencia (evaluación de tecnología); luego, una evaluación del rendimiento en un ambiente de imitación (evaluación específica); seguida de una

prueba en el lugar (evaluación operativa), antes de comenzar con las operaciones completas. Cada evaluación cumple un fin diferente e incluye distintos tipos de análisis. Se realizan con el fin de determinar la exactitud de los valores obtenidos tras realizarle ensayos a los mismos. (5)

### **1.2.2 Fases del procedimiento biométrico**

El procedimiento biométrico consta necesariamente de 2 períodos:

- Fase de registro: el sujeto entrega al sistema un elemento biométrico del cual se extrae el "*template*" o "modelo" o bien una representación matemática de los datos biométricos (el modelo puede ser memorizado en una base de datos centralizada).
- Fase de verificación: durante la cual, el dato biométrico, ya adquirido por el sistema, es confrontado con el que está siendo sometido a análisis, ya sea para una autenticación o para una identificación. (1)

### **1.3 Reconocimiento biométrico**

Aparece como una de las alternativas más efectivas para enfrentar la problemática de la identificación de personas, es un término general, y no necesariamente implica verificación ni identificación. Todos los sistemas biométricos realizan "reconocimiento" para "volver a conocer" a una persona que ya ha sido inscrita. Tiene como objetivo el lograr que, a través de sistemas computacionales, se reconozca con certeza a cada persona sobre la base de patrones personales. (1)

### **1.4 Verificación biométrica**

Se define como verificación biométrica a la tarea durante la cual el sistema biométrico intenta confirmar la identidad declarada de un individuo, al comparar la muestra suministrada con una o más plantillas registradas con anterioridad. (1)

Un sistema biométrico que trabaja en el modo de verificación comprueba la identidad de algún individuo, comparando la característica sólo con los *templates* del individuo. Por ejemplo, si una persona ingresa su nombre de usuario entonces no será necesario revisar toda la base de datos buscando el *template* que más se asemeje al de él, sino que bastará con comparar la información de entrada sólo con el *template* que está asociado al usuario. Esto conduce a una comparación uno-a-uno para determinar si la identidad reclamada por el individuo es verdadera o no.



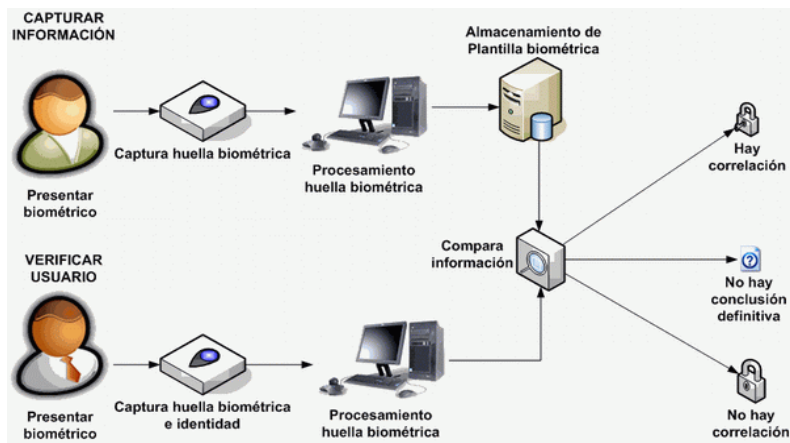


Figura 1: Proceso de captura y verificación de usuario

En el proceso de verificación intervienen varios conceptos fundamentales:

Cliente: usuario genuino del sistema, denominado también inscrito.

Impostor: una persona que se somete a una muestra biométrica, ya sea en un intento deliberado o inadvertido para reclamar la identidad de otra persona a un sistema biométrico. (1)

Las respuestas del sistema son:

- Una persona autorizada es aceptada.
- Una persona autorizada es rechazada.
- Un impostor es rechazado.
- Un impostor es aceptado. (6)

Falsa aceptación: si se acepta un impostor. (1)

Falso rechazo: si un cliente verdadero es rechazado. (1)

Entre las medidas que reflejan la eficiencia de un sistema de autenticación biométrica se encuentran:

Tasa de falsos aceptados (FAR): es la probabilidad de que un individuo no autorizado sea autenticado (1). Por ejemplo: María declara ser Pedro, y el sistema afirma esta declaración. En general entre más bajo sea el valor de la tasa de falsa aceptación, más alta es la precisión del sistema biométrico. A nivel de fabricantes, la mayoría tiene esta tasa entre el 0.0001% y el 0.1%. En la práctica, el verdadero FAR puede ser estimado de la forma que muestra la Ecuación 1. (7)

$$FAR = \frac{\text{Número de Falsos Aceptados}}{\text{Número de intentos del impostor}} \text{ Ecuación 1}$$

Tasa de falsos rechazados (FRR): es la probabilidad de que un individuo autorizado sea inapropiadamente rechazado. Por ejemplo Lieter declara ser Lieter y el sistema deniega esta petición. En general entre más bajo sea el valor de la tasa de falso rechazo, más alta es la precisión del sistema biométrico. Comercialmente su valor varía entre el 0.00066% y el 1%. Sucediendo lo mismo que con el FAR, en la práctica puede ser estimado como muestra la Ecuación 2: (7)

$$FRR = \frac{\text{Número de Falsos Rechazados}}{\text{Número de intentos del cliente}} \text{ Ecuación 2}$$

Tasa de error igual (EER): algunas veces se llama tasa de error cruzada (CER). Es una estadística que muestra la actuación del sistema biométrico, típicamente cuando opera en la tarea de verificación. En general entre más bajo sea el valor de la tasa de error igual, más alta es la precisión del sistema biométrico.

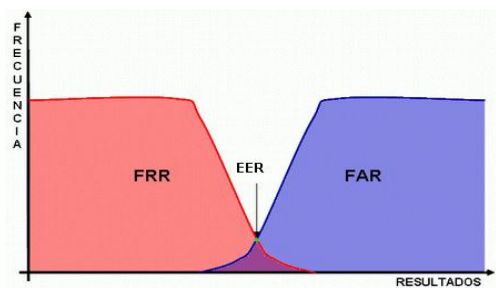


Figura 2: tasa de error igual.

La tasa de error se calcula como el punto en que las  $FAR(t) = FRR(t)$  (figura 3 a). En la práctica, las distribuciones de resultados no son continuas y un punto de cruce podría no existir. En este caso (figura 3 b, c), el valor de EER se calcula de la siguiente manera: (1)

$$EER = \frac{\frac{FAR(t1)+FRR(t1)}{2}}{\frac{FAR(t2)+FRR(t2)}{2}} \text{ if } FAR(t1) - FRR(t1) \leq FRR(t2) - FAR(t2) \text{ Ecuación 3}$$

$$\text{otherwise}$$

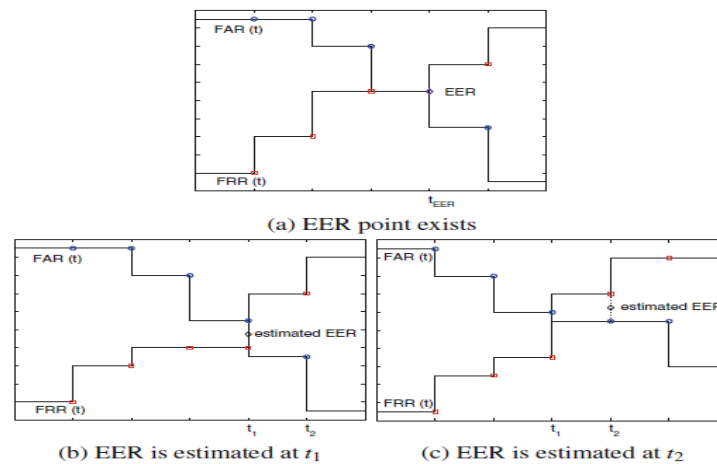


Figura 3: Curva FAR vs FRR.

(a) Ejemplo donde el punto de EER existe, (b) y(c) ejemplos donde el punto EER no existe. (1)

Tasa de falsa coincidencia (FMR): la probabilidad de que un sistema biométrico identifique incorrectamente un individuo o que falle para rechazar un impostor. La alternativa a tasa de falsos aceptados (FAR). (1)

Tasa de falsa no-coincidencia (FNMR): es parecida a la tasa de falso rechazo (FRR), con la diferencia de que esta incluye la tasa de falla para capturar el error. (1)

Error tipo 1: este tipo de error ocurre en una prueba estadística cuando una reclamación válida es rechazada. Por ejemplo Lieter reclama ser Lieter, pero el sistema niega el reclamo de manera incorrecta. (1)

Error Tipo 2: este tipo de error ocurre en una prueba estadística cuando una reclamación falsa es aceptada. Por ejemplo Lieter reclama ser René y el sistema acepta el reclamo de manera incorrecta. (1)

Tasa de verificación: tasa en la cual los usuarios finales legítimos son correctamente verificados.

## 1.5 Identificación biométrica

Tarea durante la cual el sistema biométrico intenta determinar la identidad de un individuo. Se recopila información biométrica y se compara con todas las plantillas en la base de datos, puede ser de tipo identificación abierta o identificación cerrada. Cuando es encontrada una combinación, el usuario es "identificado" como un usuario preexistente. (1)

**Identificación cerrada:** en el proceso de comparación de uno a muchos, el usuario presenta su(s) dato(s) biométrico(s) y el dato biométrico se compara contra la base de datos, donde se sabe que existe, buscando la identidad más probable del usuario.

**Tasa de identificación:** tasa en la cual un individuo que es parte de la base de datos es correctamente identificado.

**Identificación abierta:** es un proceso híbrido entre la verificación y la identificación cerrada, donde la persona no reclama una identidad específica, entonces se compara contra toda la base de datos para verificar si existe en la base de datos, una vez que se verifica que posiblemente existe, dentro de las coincidencias más probables, se determina quién es el usuario.

**Tasa de falsa alarma:** representa el porcentaje de veces que se emite una alarma ante un individuo que no es parte de la base de datos del sistema biométrico ejemplo (el sistema emite alarma ante Lieter, y Lieter no es parte de la base de datos), o cuando se emite una alarma, pero se identifica a la persona equivocada ejemplo (el sistema emite alarma ante René, cuando René es parte de la base de datos, pero el sistema piensa que René es José).

**Tasa de detección e identificación:** tasa en la cual los individuos que son parte de la base de datos hacen que suene una alarma del sistema, y son correctamente identificados en una aplicación de identificación de grupo abierto (listas de vigilancia).

Para la toma de decisiones el resultado de cualquiera de las comparaciones que se realicen, puede presentar una de tres posibilidades. Depende de la puntuación que se alcance en la comparación de la plantilla, el dato biométrico y del umbral que se le haya dado al sistema; las tres posibles alternativas son:

- Hay correlación: es decir que al comparar el dato biométrico capturado con la(s) plantilla(s) almacenada(s), la puntuación está dentro de los umbrales de coincidencia.
- No hay correlación: es decir que al comparar el dato biométrico capturado con la(s) plantilla(s) almacenada(s) la puntuación está fuera de los umbrales de coincidencia.
- Imposibilidad de alcanzar conclusión definitiva: es decir que hay falta de información para poder hacer una comparación adecuada.

## 1.6 Punto de operación (OP)

En la práctica, los sistemas biométricos operan en un nivel bajo del FAR en lugar del EER a fin de proporcionar alta seguridad. Este punto de funcionamiento se define en términos de FRR (%) conseguidos con un valor de FAR fijo. El valor fijo  $\alpha$  del FAR depende de la modalidad. En la práctica, el OP se calcula de la siguiente manera:

$$OP_{\{FAR=\alpha\}} = FRR(t_{OP}) \mid t_{OP} = \max_{t \in S} \{t \mid \alpha \leq FAR(t)\} \quad \text{Ecuación 4}$$

Donde S es el conjunto de los umbrales utilizados para calcular las distribuciones de resultados. (8)

## 1.7 Curva característica operativa del receptor

Para arribar a una comparación efectiva de los diferentes sistemas y una descripción independiente de la escala umbral se requiere la característica de operativa del receptor (ROC), que valora la FRR directamente en contra de los valores de las FAR, eliminando así los parámetros de umbral. El ROC, como la FRR, sólo puede tomar valores entre 0 y 1 y se limita a valores entre 0 y 1 en el eje x (FAR). (9)

Esta curva se utiliza para resumir el rendimiento de un sistema de verificación biométrica. La **Curva ROC** es independiente del umbral, lo que permite la comparación de resultados diferentes en sistemas donde las condiciones son similares. Otra interpretación de este gráfico es la representación de la razón de verdaderos positivos (VPR) frente a la razón de falsos positivos (FPR), también según varíe el umbral de discriminación (valor a partir del cual se decide que un caso es positivo). Es decir, ROC como una comparación de dos características operativas (VPR y FPR) según se cambia el umbral para la decisión. (1)

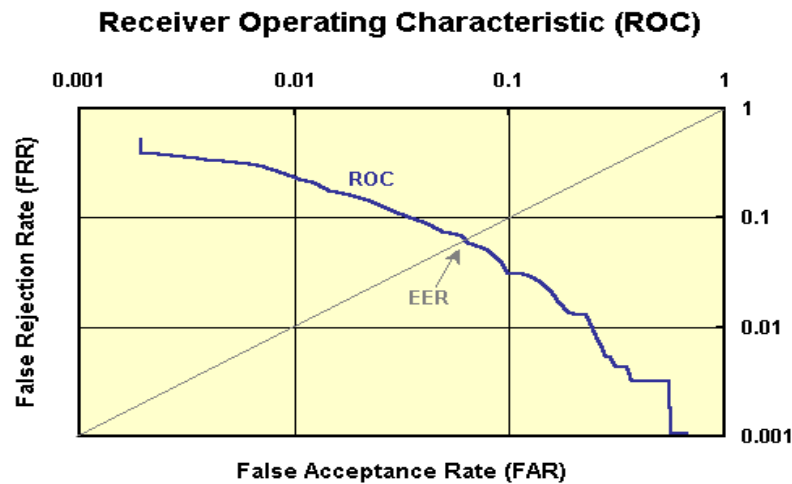


Figura 4: Curva característica operativa del receptor.

Matemáticamente, se implementa este método mediante la introducción de una FRR generalizada (GFRR) y FAR generalizada (GFAR). El cálculo de GFRR y GFAR es bastante simple, se asume que cada ensayo de autenticación es precedido por su ensayo de inscripción propia, debido a que el rendimiento de la autenticación no es independiente de la inscripción. ROC se calcula como se muestra en la figura 5.

$$ROCA = \sum_{n=1}^K FRR(n)p_N(n-1)$$

$$p_n = p_b \Rightarrow FAR = 1 - FRR \Rightarrow$$

$$EER = 1/2$$

Figura 5: Ecuación ROC

## 1.8 Estándares biométricos internacionales

**Un estándar** es un conjunto de reglas que deben cumplir los productos, procedimientos o investigaciones que afirmen ser compatibles con el mismo producto. Puede ser conceptualizado como la definición clara de un modelo, criterio, regla de medida o de los requisitos mínimos aceptables para la operación de procesos específicos, con el fin asegurar la calidad en la prestación de los servicios de salud. Los estándares requieren ser establecidos con el fin de contar con una referencia que permita identificar oportunamente las variaciones presentadas en el desarrollo de los procesos y aplicar las medidas correctivas necesarias. (1)

Un **estándar biométrico** representa las reglas que deben cumplir los productos, procedimientos o investigaciones biométricas. La carencia de estándares biométricos a nivel industrial ha dificultado el desarrollo de algunos tipos de sistemas biométricos y el crecimiento de este sector industrial. Sin embargo, la industria biométrica está teniendo un papel muy activo para solucionar el problema de la carencia de estándares generando resultados que empiezan a ser ampliamente aceptados por los industriales y marcando el camino a seguir en un futuro próximo. (8)

### 1.8.1 Interfaz de programación de aplicaciones

Interfaz de programación de aplicaciones (API por sus siglas en inglés). Nace en abril de 1998 durante la conferencia *CardTech/SecureTech* con el apoyo de algunas de las compañías informáticas más importantes a nivel internacional como IBM y *Hewlett-Packard*. La primera especificación apareció en septiembre de 2000 y la especificación final en marzo de 2001. La idea era desarrollar una alternativa a otras iniciativas de estandarización. **BioAPI** ha llegado a ser uno de los esfuerzos más relevantes en la generación de estándares biométricos con varios objetivos como: desarrollar una Interfaz de Programación de Aplicaciones, independiente del parámetro biométrico y del *hardware* de los distintos fabricantes. (10)

Desde un punto de vista general, **BioAPI** intenta estandarizar el modo en el que las aplicaciones se comunican con los dispositivos biométricos y la forma en la que los datos son almacenados y utilizados. Ofrece a los desarrolladores un conjunto común de llamadas a funciones para interactuar modularmente con los distintos dispositivos biométricos y algoritmos. Sin embargo, no pretende estandarizar el modo en el que los datos son generados por los dispositivos biométricos, ni entrometerse en los rasgos distintivos que define la tecnología biométrica de cada fabricante. El resultado de ello es que, en algunos casos, obliga a los usuarios a “reentrenarse” en el uso de estos dispositivos. Las funciones de BioAPI cubren aspectos como el entrenamiento, la verificación e identificación de usuarios, la captura de datos, el proceso de los mismos, la comparación de patrones y el almacenamiento de la información biométrica. Establece un alto nivel de abstracción que permite a los desarrolladores olvidarse de los detalles particulares de fabricación de los distintos productos y de las tecnologías empleadas por los diferentes fabricantes. (10)

### 1.8.2 Interfaz de programación estándar (BAPI)

**BAPI** es un nuevo estándar biométrico desarrollado y planeado por un vendedor de soluciones biométricas llamado *I/O Software* en lugar de un consorcio de compañías e instituciones. En mayo de

2000 *Microsoft* licenció BAPI, con la intención de incluirlo en las futuras versiones de sus sistemas operativos (*Windows*). BAPI se fusionó con su predecesor BioAPI llegando casi a reemplazarlo. La idea de que la tecnología biométrica forme parte de un sistema operativo ha hecho madurar esta área tecnológica, dejando de ser considerada como una tecnología del futuro y formado parte de un panorama actual de posibilidades a tener en cuenta a la hora de desarrollar aplicaciones. Arrastrados por la iniciativa de *Microsoft*, otras compañías como *Intel* han apostado por BAPI, licenciando este estándar para incluirlo en sus plataformas PC móviles y dotarlas de aspectos de seguridad. (8)

### **1.8.3 Formato común de ficheros para el intercambio biométrico (CBEFF)**

El objetivo es definir los formatos de los patrones biométricos para facilitar el acceso y el intercambio de diferentes tipos de datos biométricos, a los sistemas que integran esta tecnología, o entre diferentes componentes de un mismo sistema. Establece un formato para la cabecera de los ficheros. Define campos obligatorios y opcionales que proporcionan elementos comunes (opciones de seguridad, de integridad de los datos, fecha de creación del fichero, firma y tipo de parámetro biométrico) para el intercambio de información entre los dispositivos biométricos y los sistemas que hacen uso de los mismos. Favorece la interoperabilidad entre las aplicaciones biométricas y los sistemas. Simplifica la integración del *software* y el *hardware*, y posibilita la compatibilidad futura frente a los nuevos avances tecnológicos que se vayan produciendo. CBEFF no busca soluciones de interacción con los dispositivos o con los procesos, sino un método común para manejar los datos biométricos. (8)

La definición e implementación de este estándar está siendo considerada para su incorporación en dispositivos como las tarjetas inteligentes bajo los auspicios del grupo de trabajo *NIST/BC Biometric Interoperability, Performance and Assurance Working Group*.

### **1.8.4 Estándar ANSIX9.84**

La organización acreditada por el Instituto Nacional de Estándares de Estados Unidos (*American National Standards Institute, ANSI*) conocida como X9 es responsable del desarrollo y la publicación de los consensos alcanzados en temas de estandarización para la industria de servicios financieros. Entre las tareas encomendadas a esta organización se destacan:

- La comprobación de procesos.
- La comprobación de las transacciones electrónicas comerciales y personales.



- La gestión y la protección de los códigos de autenticación personal (PIN).
- El uso de técnicas criptográficas.
- Los pagos a través de internet.
- El intercambio de imágenes financieras.
- Los aspectos de seguridad de la banca en línea.

Dentro de X9, el grupo de trabajo X9.84-2000 (*Biometric Information Management and Security*) se encarga de la estandarización biométrica, preocupados por la seguridad y la gestión de los datos biométricos de los usuarios durante el tiempo de su existencia. Aspectos como la seguridad en la transmisión y en el almacenamiento de esta información biométrica sensible, o la seguridad e integridad en el *hardware* y *software* utilizado, constituyen algunos de sus objetivos. Su misión es desarrollar los estándares biométricos necesarios antes de empezar a realizar grandes inversiones en esta tecnología para incluirla en sus propios servicios. (8)

El **estándar X9.84** se aprobó en marzo de 2011. Su idea es que, si el sistema es consistente, los bancos e instituciones similares pueden implementar soluciones con la confianza suficiente en las fuentes de datos, los resultados de la verificaciones que se produzcan, la integridad y la confidencialidad de los datos que intervienen en el proceso de autenticación, en los procesos de transmisión y almacenamiento de esos datos. (11)

El estándar X9.84 proporciona integridad y privacidad. La privacidad se consigue mediante técnicas criptográficas aplicadas sobre los datos biométricos específicos del usuario, la integridad se alcanza aplicando de igual manera técnicas criptográficas como las firmas digitales o un mensaje con un código de autenticación (*Message Authentication Code, MAC*) sobre todos los datos biométricos, es decir, la cabecera y el bloque de datos opaco con la información biométrica específica de ese usuario. (8)

El estándar que se utilizará en la aplicación para representar las reglas que deben cumplir los productos es ANSIX9.84, debido a que es muy actual (2011), contiene mejoras que los demás no incluyen, proporciona integridad y privacidad.

## **1.9 Estado de los sistemas de evaluación biométrica a nivel internacional**

El análisis de la documentación ha permitido la identificación de las principales técnicas biométricas existentes. La mayoría aún se encuentran en estados de madurez insuficientes, muchas otras se encuentran suficientemente maduras y ofrecen soluciones de autenticación en muy diversos ámbitos. A continuación se representa un resumen de las aplicaciones biométricas y campañas de evaluación actuales a nivel internacional.

### **1.9.1 NIST**

El *U.S. National Institute of Standards and Technology* (NIST) es el líder de las actividades de evaluación biométrica. NIST ha estado coordinando las evaluaciones de *Speaker Recognition Evaluations* desde 1996. Cada campaña se inicia con el anuncio del plan de evaluación oficial, que establece claramente las reglas y las tareas involucradas en la misma, la cual culmina con un taller de seguimiento, donde NIST informa los resultados oficiales y que además los investigadores puedan compartir sus hallazgos. Los protocolos de NIST y bases de datos utilizadas en *Speaker Recognition Evaluations 2005* forman parte de la evaluación comparativa de *BioSecure*. (1)

NIST también es muy activa en las huellas dactilares. Incluso en 2003 se desarrollaron campañas en esta modalidad. NIST organizó la primera campaña de *Iris Challenge Evaluation* (ICE) en el 2006, además de lanzar el *Multiple Biometric Grand Challenge* (MBGC) en el 2008.

### **1.9.2 Reconocimiento facial gran desafío (FRGC)**

La meta del FRGC (*Face Recognition Grand Challenge*) fue promover y adelantar la tecnología de reconocimiento facial diseñada para dar soporte a los esfuerzos existentes del gobierno de los Estados Unidos. El FRGC procuró desarrollar nuevas técnicas de reconocimiento facial y desarrollar sistemas prototipo mientras que aumenta el rendimiento mediante un orden de magnitud.

El objetivo de FRGC es evaluar la actuación de reconocimiento del rostro y los sistemas para alta definición 2D y 3D. (1)

### 1.9.3 FRVT

Las pruebas de reconocimiento facial (*Face Recognition Vendor Tests*) se llevaron a cabo en 2000, 2002 y 2006. Estas evaluaciones fueron construidas sobre el trabajo del *The Facial Recognition Technology* (FERET) y en coincidencia con el inicio general de productos de reconocimiento facial comercialmente disponibles.

**EL FRVT 2000** tuvo dos metas: evaluar las capacidades de los sistemas de reconocimiento facial comercialmente disponibles y educar la comunidad de biometría, el público general sobre cómo presentar y analizar resultados apropiadamente.

**FRVT 2002** fue diseñado para medir progresos técnicos desde el año 2000, para evaluar el rendimiento en bases de datos a gran escala de la vida real, y para introducir nuevos experimentos para ayudar a entender mejor el rendimiento del reconocimiento facial. El FRVT 2002 incluyó experimentos con barras de error mostrando variaciones en los rendimientos al intercambiar imágenes similares. Son resultados clave del FRVT 2002, la iluminación de interiores razonable controlada dada, la tecnología de punta de reconocimiento facial es de verificación del 90% a una tasa de falsa aceptación de 1%.

El **FRVT 2006** es una campaña de evaluación independiente a gran escala que tiene como objetivo principal buscar en el rendimiento en las modalidades 2D de alta resolución y 3D la cual estaba abierta a universidades, institutos de investigación y empresas. (1)

### 1.9.4 Campaña de evaluación multimodal de bioseguridad del 2007 (BMEC'2007)

Una de las aspiraciones fundamentales de la red de excelencia de bioseguridad fue investigar en profundidad el potencial y las ventajas de la biometría multimodal. Con el fin de crear las condiciones que permitan a dichos estudios llevarse a cabo de una manera significativa. El objetivo inicial de esta evaluación, fue poner a prueba el impacto real de la multimodalidad en escenarios bien elegidos, es decir, móviles y de control de acceso. Mientras tanto, la campaña permitió a los participantes evaluar el desempeño de sus algoritmos además de comparar y evaluar los beneficios de las distintas soluciones de investigación. (1)

El objetivo principal del escenario móvil era probar la solidez de los sistemas biométricos monomodal y multimodal contra las condiciones de adquisición degradadas. Estas condiciones se pueden encontrar cuando la verificación biométrica de la identidad se realiza ya sea en interiores o al aire libre utilizando un dispositivo con capacidades limitadas como *webcam*, un PDA o un teléfono móvil. El escenario móvil incluyó dos tipos diferentes de evaluación:

Evaluación monomodal: esta evaluación fue en realidad una continuación del trabajo iniciado durante el taller de bioseguridad residencial. Las modalidades en cuenta para esta evaluación incluyen la huella digital, firma, el habla y el rostro en 2D en las secuencias de video. (1)

Evaluación multimodal: se presenta a menudo como una manera de mejorar el rendimiento del sistema monomodal, especialmente en el caso de las condiciones degradadas, y para mejorar la resistencia a las falsificaciones. (1)

De esta manera, esta evaluación permitirá probar y evaluar los dos puntos siguientes: o la mejora de rendimiento en condiciones degradadas (en relación a los obtenidos con los sistemas de monomodal), o la solidez de las falsificaciones. El último taller de bioseguridad, que se celebró a finales de septiembre de 2007. Se reunieron a todos los participantes para discutir en gran medida los resultados de BMEC'2007, donde se dieron a conocer por primera vez los resultados de las evaluaciones de bioseguridad. (1)

### **1.9.5 FVC- OnGoing**

Tras el éxito de los primeros cuatro concursos de verificación de huellas dactilares (FVC2000, FVC2002, FVC2004 y FVC2006), el Laboratorio de Sistema Biométrico (Universidad de Bolonia) decidió organizar una nueva campaña de evaluación en línea para las tecnologías de reconocimiento de huellas dactilares: FVC-OnGoing. (12)

Es un sistema *web* basado en la evaluación automatizada de algoritmos de reconocimiento de huellas dactilares. Las pruebas se llevan a cabo en un conjunto de bases de datos y los resultados se reportan en línea mediante el uso de conocidos indicadores de desempeño y métricas. (12)

El objetivo es realizar un seguimiento de los avances en las tecnologías de reconocimiento de huellas dactilares, a través de pruebas independientes actualizándose continuamente. Brinda la notificación de actuaciones en los puntos de referencias dados. Los algoritmos se evalúan utilizando métodos fuertemente supervisados para maximizar la confiabilidad. (12)

A partir de la investigación realizada sobre las diferentes plataformas de pruebas y eventos biométricos de identificación dactilar se concluyó lo siguiente:

La plataforma FVC-onGoing, desarrollada en el laboratorio de sistema biométrico de la Universidad de Bolonia, cumple con las características necesarias para la evaluación de algoritmos de reconocimiento de huellas dactilares en línea por lo que es un punto de referencia importante para la investigación, utiliza como estándar biométrico ANSIX9.84, el mismo se preocupa por la seguridad y la gestión de los datos biométricos de los usuarios durante el tiempo de su existencia. Por consiguiente se planteó la necesidad que posee el CISED, de contar con un sistema *web* basado en la evaluación automatizada de algoritmos de comparación de huellas dactilares, permitiendo realizar pruebas y mostrar resultados en línea.

Actualmente en el CISED se cuenta con una aplicación escritorio que realiza pruebas a los algoritmos biométricos de huellas dactilares. Se pretende desarrollar una aplicación *web* que presente las mismas funcionalidades con que cuenta esta aplicación de escritorio, además que utilice los indicadores de desempeño (FAR, FRR, EER, entre otros) y que muestre los resultados en gráficas similares a FVC onGoing.

## **1.10 Metodología, herramientas y lenguajes empleados**

Las metodologías proporcionan una organización lógica en el proceso de desarrollo del *software*, apoyándose de las herramientas necesarias y más convenientes para lograr obtener el resultado esperado con la eficiencia requerida. Por lo que el estudio de las mismas permitirá seleccionar las correctas para el desarrollo de la aplicación.

### **1.10.1 Proceso Unificado de Rational (RUP)**

RUP es un proceso de desarrollo de *software*, orientado a objetos, preparado para desarrollar grandes y complejos proyectos, unifica los mejores elementos de metodologías anteriores y utiliza el Lenguaje Unificado de Modelado (UML) (13)

Brinda la posibilidad a cada miembro del equipo de desarrollo de *software* un fácil acceso a una base de conocimientos a través de guías, plantillas y herramientas para realizar las actividades críticas del desarrollo de *software*. La metodología de desarrollo RUP presenta tres características fundamentales: dirigido por casos de uso, centrado en la arquitectura e iterativo-incremental. (13)

En RUP las actividades se dividen en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos de apoyo.

### **Flujos de trabajo:**

- Modelado del negocio.
- Requerimientos.
- Análisis y diseño.
- Implementación.
- Prueba (Testeo).
- Instalación.
- Administración del proyecto.
- Administración de configuración y cambios.
- Ambiente.

RUP divide el proceso en cuatro fases: inicio, elaboración, construcción y transición; dentro de las que se realizan un número variable de iteraciones en las que se hacen un mayor o menor énfasis de las actividades de acuerdo al proyecto. (14)

### **Fases:**

#### Conceptualización (Concepción o Inicio):

Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema, que orientarán la funcionalidad.

#### Elaboración:

Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requisitos (funcionales y no funcionales), identificados de acuerdo con el alcance definido.

#### Construcción:

Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene uno o varios *release* del producto que han pasado las pruebas. Se ponen estos *release* a consideración de un subconjunto de usuarios. Es la fase más prolongada de todas.

### Transición:

El *release* ya está listo para su instalación en las condiciones reales y se corrigen los últimos errores. Se llama transición porque se transfiere a las manos del usuario, pasando del entorno de desarrollo al de producción.

Cada una de estas fases tiene objetivos específicos y una serie de artefactos<sup>1</sup> que deben alcanzarse. La metodología de desarrollo RUP presenta diversas ventajas entre las que se puede citar:

### **Ventajas**

- Reconoce que las necesidades del usuario y sus requerimientos no se pueden definir completamente desde el principio.
- Permite evaluar tempranamente los riesgos en lugar de descubrir problemas en la integración final del sistema.
- Reduce el costo del riesgo a los costos de un solo incremento.
- Acelera el ritmo del esfuerzo de desarrollo en su totalidad debido a que los desarrolladores trabajan para obtener resultados claros a corto plazo.
- Distribuye la carga de trabajo a lo largo del tiempo del proyecto ya que todas las disciplinas colaboran en cada iteración. (13)

A pesar que la propuesta de solución de la presente investigación no se considera un proyecto grande y complejo, se utilizará esta metodología de desarrollo de *software*, ya que es la definida por el Departamento Biometría, debido a que es muy organizada y genera desde sus inicios una documentación robusta, lo que facilita continuar el desarrollo de varios sistemas sin la necesidad de generar una nueva documentación y así poder contar con un expediente de proyecto claro y fácil de entender en el momento de ser consultado.

### **1.10.2 Lenguaje Unificado de Modelado (UML)**

El lenguaje de modelado seleccionado para guiar el diseño del sistema fue el Lenguaje Unificado de Modelado (por sus siglas en inglés UML).

---

<sup>1</sup> Producto tangible resultante del proceso de desarrollo de *software*.

Es una herramienta para la especificación de sistemas, para ayudar a describir y mapear visualmente el diseño de un sistema de *software* y su estructura. El uso de UML como una herramienta para la definición de la estructura de un sistema es una forma útil de manejar sistemas grandes y complejos, así como tener una estructura claramente visible hace que sea fácil introducir nuevas personas a un proyecto existente. La construcción de librerías de patrones UML simplifica la reutilización de modelos y el código, es importante resaltar que UML es un "lenguaje" para especificar y no un método o un proceso, se utiliza para definir un sistema de *software*, para detallar los artefactos en el sistema y para documentar y construir, y es a su vez, el lenguaje en el que está descrito el modelo. UML puede aplicar en una gran variedad de formas soportable una metodología de desarrollo de *software*, (tal como el Proceso Unificado de Rational), pero no especifica en sí mismo qué metodología o proceso utilizar. (15)

### **1.10.3 Herramienta de modelado. *Visual Paradigm v6.4***

Esta herramienta será utilizada para basado en el lenguaje de modelado definido, diagramar todos los procesos y flujos que se llevan a cabo entre los escenarios del sistema, así como representar las diferentes vistas de arquitectura, despliegue y datos de la herramienta a desarrollar.

Para el modelado de la aplicación se emplea la herramienta de diseño *Visual Paradigm*. La misma soporta todos los diagramas UML y el diagrama de entidad-relación. Provee documentación del sistema en múltiples formatos como *PDF* y *HTML*. *Visual Paradigm* para *UML* es multiplataforma, lo cual le permite al usuario utilizar esta herramienta en varios sistemas operativos como *Windows*, *Linux*, *Unix* y otros. (16)

#### **Características:**

- Producto de calidad.
- Soporta aplicaciones *web*.
- Las imágenes y reportes generados, no son de muy buena calidad.
- Generación de código para *Java* y exportación como *HTML*.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.

Además presenta varias ventajas entre las que se destacan:

- Contribuye a la rápida construcción de aplicaciones de calidad y a un menor costo.



- Poderosa herramienta de generación de PDF/HTML a partir de diagramas UML.
- Permite la sincronización entre el código fuente y el modelo en tiempo real.
- Soporte para toda la notación UML.
- Ofrece capacidades de ingeniería directa e inversa.
- Es una herramienta colaborativa, es decir, soporta a varios usuarios trabajando en un mismo proyecto.
- Permite el control de versiones.

Brinda la posibilidad de generar código a partir de los diagramas, para plataformas como .Net, Java y PHP, así como permite la obtención de diagramas a partir de código. (16)

### 1.10.4 Lenguaje de programación

Los lenguajes de programación surgieron un poco después de los ordenadores, su creación vino dada a la gran necesidad de programarle tareas fundamentales a los ordenadores. Los primeros eran muy difíciles de desarrollar, en la actualidad han evolucionado logrando facilitar su aprendizaje, uso y compilación. Entre los más usados se encuentran PHP, Java y C Sharp.

Este último será el lenguaje en el que se va a desarrollar la aplicación biométrica debido a que se emplea en el departamento de biometría como política de programación.

C Sharp (C#) es un lenguaje de programación orientado a objeto, el cual ha transitado un largo camino desde que salió a luz C, seguido por el C++ hasta llegar al C#. El C# es un híbrido entre el C++ y Java, mezclando la combinación de operadores del primero y la plena orientación a objetos del segundo. Actualmente se encuentra entre los diez lenguajes más utilizados. A pesar de su corta historia, ha recibido la aprobación de estándar de dos organizaciones: en el 2001 se aprueba el ECMA y en el 2003 el ISO. Las ventajas que ofrece respecto a otros lenguajes son varias, la declaración de datos tiene un rango más amplio y mejor definido que otros lenguajes como el C++ o el Java, cada miembro de las clase tiene atributos lo que pueden estar en diferentes niveles como: público, privado o protegido. (17)

#### Características

- Es autocontenido: un programa en C# no necesita de ficheros adicionales al código fuente.
- Es homogéneo: el tamaño de los tipos de datos básicos es fijo e independiente del compilador, lo que facilita la portabilidad del código.

- Es actual: C# incorpora en el propio lenguaje elementos que han demostrado ser muy útiles para el desarrollo de aplicaciones como el tipo básico decimal que representa valores decimales con 128 bits, lo que lo hace adecuado para cálculos financieros y monetarios. Incorpora la instrucción *foreach*, que permite una cómoda iteración por colecciones de datos.
- Está orientado a objetos: C# soporta todas las características propias del paradigma de la programación orientada a objetos: encapsulación, herencia y polimorfismo.
- Encapsulación: además de los modificadores de acceso convencionales: *public*, *private* y *protected*, añade el modificador *internal*, que limita el acceso al proyecto actual.
- Herencia: sólo admite herencia simple.
- Polimorfismo: los métodos son por defecto sellados y los métodos redefinibles han de marcarse obligatoriamente, con el modificador *virtual*.
- Delega la gestión de memoria: como todo lenguaje .NET, la gestión de la memoria se realiza automáticamente ya que tiene a su disposición el recolector de basura del CLR. Esto hace que el programador se desentienda de la gestión directa de la memoria (petición y liberación explícita) evitando que se cometan los errores habituales de este tipo de gestión.
- Proporciona seguridad con los tipos de datos. c# no admite ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código. (17)

#### 1.10.4.1 Herramienta *Visual Studio Ultimate 2010*

El IDE que se escogió para el desarrollo del sistema fue *Visual Studio Ultimate 2010* por las características y ventajas que se mencionan a continuación.

*Microsoft Visual Studio 2010 Ultimate* incluye potentes herramientas que simplifican todo el proceso de desarrollo de aplicaciones, de principio a fin. Los equipos pueden observar una mayor productividad y ahorro de costes al utilizar características de colaboración avanzadas, así como herramientas de pruebas y depuración integradas que le ayudarán a crear siempre un código de gran calidad. Entre las características del mismo están: (18)

##### **Compatibilidad con la plataforma de desarrollo**

*Visual Studio 2010 Ultimate* permite hacer realidad la idea en una gran variedad de plataformas, entre las que se incluyen *Windows*, *Windows Server*, *Web*, *Cloud*, *Office* y *SharePoint*, entre otras, todo en un único entorno de desarrollo integrado. (18)

## Entorno de desarrollo integrado

*Visual Studio 2010 Ultimate* posee características personalizables como, por ejemplo, compatibilidad con varios monitores, de modo que se pueda organizar y administrar el trabajo. (18)

Uno de los mayores logros de la versión 2010 de *Visual Studio* ha sido el de incluir las herramientas para desarrollo de aplicaciones para *Windows 7*. Es un entorno integrado que simplifica la creación, depuración e implementación de aplicaciones. Trabaja dentro de un entorno personalizado, dirigido a un número cada vez mayor de plataformas. ***Visual Studio Ultimate 2010*** se presenta con el firme objetivo de impulsar las ideas y la imaginación de los desarrolladores, facilitando su trabajo en los procesos de desarrollo y diseño, poniendo a su disposición herramientas de calidad que garanticen resultados óptimos. (18)

### Ventajas:

- Interpreta rápidamente el código
- Crea enriquecidas experiencias de usuario
- Utiliza las habilidades existentes: Trabaja el desarrollo de *SharePoint*, incluyendo herramientas para componentes *web*, listas, los flujos de trabajo, eventos y más.
- Dedicar menos tiempo a la depuración: La jerarquía de llamada en línea ayuda rápidamente rastrear el flujo de ejecución de un programa sin invocar al depurador. También puede utilizar la marca de etiquetas de punto de interrupción para realizar una depuración más sencilla. (18)

## 1.10.5 ASP.NET MVC 2

El marco de ASP.NET MVC proporciona una alternativa al modelo de formularios *Web Forms* de ASP.NET para crear aplicaciones *web*. El mismo es un marco de presentación de poca complejidad y fácil de comprobar (como las aplicaciones basadas en formularios *Web Forms*), se integra con las características de ASP.NET existentes, como páginas maestras y la autenticación basada en pertenencia. El marco de MVC se define en el ensamblado *System.Web.Mvc*. (19)

El marco de ASP.NET MVC ofrece las siguientes ventajas:

Separación de tareas de aplicación (lógica de entrada, lógica comercial y lógica de la interfaz de usuario), facilidad para pruebas y desarrollo basado en pruebas (TDD). Todos los contratos principales del marco de MVC están basados en interfaz y se pueden probar utilizando objetos ficticios (objetos ficticios que imitan el comportamiento de objetos reales en la aplicación). Puede hacer una prueba

unitaria de la aplicación sin tener que ejecutar los controladores en un proceso de *ASP.NET*, lo cual posibilita que las pruebas unitarias sean rápidas y flexibles. Puede utilizar cualquier marco de pruebas unitarias que sea compatible con *.NET Framework*. (19)

Un marco extensible y conectable. Los componentes del marco de *ASP.NET MVC* están diseñados para que se puedan reemplazar o personalizar con facilidad. Puede conectar su propio motor de vista, directiva de enrutamiento de URL, serialización de parámetros de método y acción y otros componentes. El marco de *ASP.NET MVC* también admite el uso de los modelos de contenedor Inyección de dependencia (DI) e Inversión de control (IOC). DI permite insertar objetos en una clase, en lugar de depender de que la clase cree el propio objeto. IOC especifica que si un objeto requiere otro objeto, el primer objeto debe obtener el segundo objeto de un origen externo como un archivo de configuración. Esto facilita las pruebas. (19)

Amplia compatibilidad para el enrutamiento de *ASP.NET*, un eficaz componente de asignación de direcciones URL que le permite compilar aplicaciones que tienen direcciones URL comprensibles y que admiten búsquedas. Las direcciones URL no tienen que incluir las extensiones de los nombres de archivo y están diseñadas para admitir patrones de nombres de direcciones URL que funcionan bien para la optimización del motor de búsqueda (SEO) y el direccionamiento de transferencia de estado representacional (REST, *Representational State Transfer*). (19)

Compatibilidad para utilizar el marcado en archivos de *ASP.NET* existentes (archivos.aspx), de controles de usuario (archivos.ascx) y de páginas maestras (archivos .master) como plantillas de vista. Puede utilizar las características de *ASP.NET* existentes con el marco de *ASP.NET MVC*, como páginas maestras anidadas, expresiones en línea (<%= %>), controles de servidor declarativos, plantillas, enlace de datos, localización, etcétera. (19)

Compatibilidad con las características de *ASP.NET* existentes. *ASP.NET MVC* permite utilizar características como autenticación de formularios y autenticación de *Windows*, autorización para URL, pertenencia y roles, almacenamiento en caché de resultados y datos, administración de estados de sesión y perfil, seguimiento de estado, el sistema de configuración y la arquitectura de proveedor. (19)

### **1.10.6 Base de datos**

Los sistemas de gestión de bases de datos (DBMSs) permiten almacenar, visualizar y modificar datos, así como hacer copias de seguridad y mantener la integridad de los mismos, proporcionando una serie

de funciones que facilitan el desarrollo de nuevas aplicaciones. Desde un punto de vista intuitivo, una base de datos no es más que un fondo común de información almacenada en una computadora para que cualquier persona o programa autorizado pueda acceder a ella, independientemente de su lugar de procedencia y del uso que se haga de ella. En un aspecto más formal, una base de datos es un conjunto de datos comunes a un "proyecto" que se almacenan sin redundancia para ser útiles en diferentes aplicaciones. El DBMS es el *software* con capacidad para definir, mantener y utilizar una base de datos. Un sistema de gestión de bases de datos que debe permitir definir estructuras de almacenamiento, así como acceder a los datos de forma eficiente y segura. Ejemplos: *Oracle*, IBM DB2, *Microsoft SQL Server*, *Interbase* y *PostgreSQL*. En una base de datos, los datos se organizan independientemente de las aplicaciones que los vayan a utilizar (independencia lógica) y de los ficheros en los que vayan a almacenarse (independencia física). Además, los datos deben ser accesibles a los usuarios de la manera más amigable posible, generalmente mediante lenguajes de consulta como SQL o *Query-by-example*. Por otro lado, es esencial que no exista redundancia (los datos no deben estar duplicados) para evitar problemas de consistencia e integridad. (20)

#### **1.10.6.1 Base de datos de referencia**

Una de las tareas más importantes de cualquier evaluación de los sistemas biométricos es la recopilación de datos y la comparación de los mismos. Las compañías biométricas que se realizan en el mundo, han creado una multibase de datos, las cuales contienen datos diferentes biométricos almacenados en dependencia del tipo de modalidad a evaluar (Huella digital, firma, rostro, etc.). Para realizar las pruebas a los algoritmos biométricos es necesaria la utilización de una base de datos que permita realizar las comparaciones. En la aplicación biométrica se utiliza la base de datos *Innovatrics* la cual presenta las siguientes características:

Los componentes *Innovatrics* ofrecen un rendimiento excepcional perfectamente conveniente para la alta calidad de aplicaciones biométricas.

*Innovatrics* está orientado en proveer *software* de reconocimiento de huella digital rápido, interoperable e independiente de sensores, para su incorporación a aplicaciones biométricas. (21)

Los componentes de *Innovatrics* ofrecen un excelente desempeño para su uso tanto en aplicaciones de bajo y de alto costo. Para la elaboración de la base de datos contribuyeron treinta individuos. En una primera sesión se obtuvo una huella del dedo pulgar de la mano derecha de cada individuo y en la

segunda se obtuvieron once muestras más de cada dedo pulgar por cada individuo, para un total de trescientos sesenta. (1)

### 1.10.6.2 SQL Server Express 2008

**Microsoft SQL Server** es un sistema de gestión de bases de datos relacionales (SGBD) basado en el lenguaje *Transact-SQL*, y específicamente en *Sybase IQ*. Es un sistema de base de datos empresarial, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.

*Microsoft SQL Server Express* es una versión de *SQL Server* gratuita. Lanzada por *Microsoft* como alternativa a versiones gratuitas de otros motores de base de datos. Es un sistema de administración de datos eficaz y confiable que ofrece un variado conjunto de características, protección de datos y rendimiento para clientes de aplicaciones incrustadas, aplicaciones *web* ligeras y almacenes de datos locales. Está diseñado para una implementación sencilla y una creación de prototipos rápida, está disponible de forma gratuita y su redistribución con aplicaciones también es gratuita. Está diseñado para integrarse a la perfección con otras inversiones de infraestructura de servidor. (22)

Algunas de las características de *Microsoft SQL Server Express*:

- Soporte de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los clientes de la red sólo acceden a la información.
- Además permite administrar información de otros servidores de datos.
- Vistas con particiones distribuidas. (22)

Se utilizará como gestor de base de datos **Microsoft SQL Server** debido a que es una versión gratuita de *SQL Server*, además administra los datos de forma eficaz y confiable. Presenta características

relevantes y se diseña para integrarse a la perfección con otras inversiones de infraestructura de servidor.

### **1.10.6.3 Lenguaje integrado de consultas (LinQ)**

Es un *framework* de consultas que habilita el uso de órdenes tipo SQL integradas en el lenguaje de programación a partir de una serie de operadores. El objetivo es permitir que todo el código hecho en *Visual Studio* sea también orientado a objetos. Define operadores de consulta que permiten a lenguajes habilitados filtrar, enumerar y crear proyecciones de varios tipos de colecciones usando la misma sintaxis. Tales conexiones pueden incluir arreglos, XML y conjunto de datos (bases de datos relacionales). Representa una nueva forma de ver las colecciones. De la manera convencional, es necesario escribir *bucles foreach* complejos que especifican cómo recuperar los datos de una colección. (23)

Tres ventajas principales respecto a los *bucles foreach* tradicionales:

- Son más concisas y legibles, sobre todo al filtrar varias condiciones.
- Proporcionan funcionalidad eficaz de filtrado, ordenación y agrupación con código de aplicación mínimo.
- Se pueden trasladar a otros orígenes de datos con pocas o ningunas modificaciones. (23)

En general, cuanto más compleja sea la operación que se deba realizar con los datos, se observará un número mayor de ventajas al utilizar *linQ* en lugar de las técnicas de iteración convencionales. (23)

### **1.10.6.4 ADO.NET *Entity Framework***

Es una tecnología desarrollada por *Microsoft*, que a través de ADO.NET genera un conjunto de objetos que están directamente ligados a una base de datos, permitiendo a los desarrolladores manejar dichos objetos en lugar de utilizar lenguaje SQL contra la base de datos. Da vida a los modelos conceptuales permitiendo a los programadores consultar las entidades y relaciones en el modelo de dominio (denominado modelo conceptual en *Entity Framework*). (24)

Permite a los programadores trabajar con datos en forma de objetos y propiedades específicos del dominio, por ejemplo, con clientes y direcciones, sin tener que pensar en las tablas de las bases de datos subyacentes y en las columnas en las que se almacenan estos datos. (24)

- Los desarrolladores de *software* pueden trabajar en un nivel más alto de abstracción cuando tratan con datos, pueden crear y mantener aplicaciones orientadas a datos con menos código que en las aplicaciones tradicionales. Pueden funcionar en términos de un modelo conceptual más centrado en la aplicación, que incluye tipos con herencia, miembros complejos y relaciones.
- Las asignaciones entre el modelo conceptual y el esquema específico de almacenamiento pueden cambiar sin tener que cambiar el código de la aplicación.
- Dado que es un componente de *.NET Framework*, las aplicaciones de *Entity Framework* se pueden ejecutar en cualquier equipo en el que esté instalado *.NET Framework* a partir de la versión 3.5 SP. (24)

Se empleará *ADO.NET Entity Framework* con el objetivo de reducir la cantidad de código, además es compatible con el lenguaje integrado de consultas (LinQ) que se utilizará para la aplicación y proporcionará la validación de la sintaxis en el momento de la compilación para consultas en un modelo conceptual.

### **1.10.7 Artisteer**

Es un programa para automatizar el diseño de plantillas destinadas a ser visualizadas en páginas *web*, que crea al instante una red de gran apariencia, plantillas únicas y entradas de *blog*. Permite la creación de plantillas para páginas *web* sencillas, para *Wordpress*, *Drupal* o *Joomla*. Y de esta forma ayuda en hacer nuevas ideas para el diseño *web* con un código *html* y *css* en conformidad con estándares *web*. Soporta los formatos: ARTX, HTML, JPG, PNG, GIF. Posee una interfaz simple e intuitiva. Brinda soporte en diferentes idiomas. (25)

Se utilizará para el diseño *Artisteer*, debido a que posee una interfaz simple y fácil de usar, además es una herramienta muy útil para aplicaciones *web*. Brinda la posibilidad de añadir imágenes, videos, *links* y *html*. Permite ajustar los diseños generados utilizando numerosos elementos incluidos, fondos, objetos fotográficos y botones, crea código profesional y plenamente compatible con HTML y CSS.

### **1.10.8 Librería telerik**

Es uno de los gigantes diseñadores y desarrolladores de componentes para plataformas *web*. El objetivo es hacer de la manera más fácil y agradable el desarrollo del *software*. Es la herramienta para



la gestión ágil de proyectos, colaboración, desarrollo y pruebas. Permite a las empresas de todos los tamaños crear *software* de una manera más rica, estable y estética. (26)

*Telerik* es una biblioteca de cliente estática. Es la herramienta que apoya el desarrollo de aplicaciones de *software* orientadas a datos, está dirigido a resolver la diferencia de impedancia objeto-relacional. La diferencia de impedancia objeto-relacional es un conjunto de dificultades conceptuales y técnicas que se encuentran a menudo cuando un sistema de base de datos relacional de gestión, está siendo utilizado por un programa escrito en un lenguaje de programación orientado a objetos. (26)

Permite a los desarrolladores trabajar con datos en la forma de dominio específico, objetos y gráficos de objetos sin tener que preocuparse por las tablas de bases de datos subyacentes y las columnas en las que los datos reales están almacenados.

Por lo que se aumenta el nivel de abstracción en el que los desarrolladores trabajan cuando se trata de datos, además reduce el código necesario para crear y mantener aplicaciones orientada a datos. (26)

Se escogió como herramienta para la gestión ágil de proyectos *Telerik*. Posee una gran cantidad de ventajas además trabajar con datos en la forma de dominio específico, objetos, gráficos de objetos que permiten despreocuparse por las tablas de bases de datos y reduce el código necesario para crear y mantener aplicaciones orientada a datos.

### **1.10.9 JavaScript**

Es un lenguaje de programación interpretado, establece la relación entre el mecanismo cliente-servidor y permite a los desarrolladores crear acciones en sus páginas *web*. Maneja objetos dentro de la página *web* y sobre ese objeto se definen diferentes eventos. Es dinámico y responde a eventos en tiempo real. Utilizado para crear pequeños programas que luego son insertados en una página *web* y en programas más grandes orientados a objetos. Con *Javascript* se pueden crear diferentes efectos e interactuar con los usuarios. No pueden construirse programas independientes, sólo pueden escribirse *scripts* que funcionarán en el entorno de una página *web*, interpretado por un explorador. *Javascript* no posee todas las características de los lenguajes orientados a objetos como *Java* o *C++*, pero si es capaz de manejar objetos e incluso crearlos. De hecho si un programa en este lenguaje es capaz de interactuar con el explorador es gracias a esta capacidad. Posee algunos objetos predefinidos u objetos intrínsecos como son: *array*, *boolean*, *date*, *function*, *global*, *math*, *number*, *object*, *regExp*, y

*string*. Además el programador puede crear objetos nuevos, con sus propios métodos y propiedades, adaptados a las necesidades concretas de su aplicación. (27)

Se escogió como lenguaje de programación interpretado *JavaScript* debido a que establece la relación entre el mecanismo cliente-servidor y permite a los desarrolladores crear acciones en sus páginas *web*. *JavaScript* posee muchas ventajas que argumentan el uso de este lenguaje como por ejemplo: es dinámico, responde a eventos en tiempo real, establece la relación entre el mecanismo cliente-servidor y permite a los desarrolladores crear acciones en sus páginas *web*.

### 1.10.10 Librería *Jqplot*

*Jqplot* es una solución potente para mostrar gráficos y eficaz para gestionar cualquier tipo de gráfico. Ofrece muchas posibilidades de configuración y funciona enteramente en *JavaScript*. Es una nueva herramienta que ayuda a generar gráficas lineales de forma sencilla usando el potencial de *jQuery*. Las imágenes generadas son de gran calidad y con muy buena presentación. Es un *plugin jQuery* que permite crear todo tipo gráficas con datos y con una apariencia muy acabada. *Jqplot* utiliza el elemento *canvas* para renderizar cliente gráficos dinámicos por medio de programación. Una de las ventajas fundamentales de usar este *plugin de jQuery* es la posibilidad de incluir en las páginas gráficos interactivos sin necesidad de sobrecargarlas con pesados *flash*.

Es un desarrollo *open source* que ofrece:

- Crear varios tipos de gráficos entre los que se encuentran de barras, puntos, etc.
- Agregar fechas personalizadas a los ejes cartesianos.
- Agregar sombras y puntos de arrastres en los gráficos. (28)



Figura 6: Ejemplo de una gráfica en *Jqplot*.

Se utilizará la librería *Jqplot* que permite la realización de los gráficos que se mostrarán en la aplicación, además es la herramienta que ayuda a generar gráficas lineales de forma sencilla usando el potencial de *jQuery* y permite crear gráficas de considerable calidad.

### **1.10.11 Librería *jQuery***

*jQuery* es *software* libre y de código abierto. Librería *framework* de *Javascript* basada en prototipo, que facilita a los desarrolladores crear interfaces de usuario enriquecidas e interactuar con los elementos *html* de forma sencilla. Maneja los elementos *DOM* o *CSS* de una página *HTML*. Otra de las características de *jQuery* es que se puede manipular las propiedades de los elementos a los cuales se ha accedido. (29)

Es una librería *Javascript* ligera, rápida y concisa que simplifica el tratamiento de documentos *HTML*, el manejo de eventos, la creación de animaciones y las interacciones vía *Ajax*, para agilizar el desarrollo de aplicaciones *web*. *jQuery* ayuda a escribir un código más limpio, separando el comportamiento del contenido. Como ventaja se tiene que es simple, potente y hay un gran número de *plugins* en torno a *jQuery* que permiten añadir comportamientos, *widgets* y efectos visuales a la interfaz. La característica principal de la biblioteca es que permite cambiar el contenido de una página *web* sin necesidad de recargarla, mediante la manipulación del árbol *DOM* y peticiones *AJAX*. (29)

Se utilizará la librería *jQuery* por ser empleada en grandes comunidades de desarrollo web, además de contar con mucho *plugins* y buen soporte.

### **1.10.12 *Macromedia Flash 8***

*Macromedia Flash 8* es una herramienta orientada a crear aplicaciones y contenidos dinámicos para Internet, es decir, utilidades interactivas y multimedia con una amplia posibilidad de animación. Es una herramienta muy compatible, cuyas aplicaciones abarcan cada vez un espectro más amplio, desde animaciones publicitarias *on-line*, presentaciones de proyectos, *webs* interactivos, hasta creación de juegos. (30)

*Macromedia Flash 8 Profesional* marca un importante lanzamiento que incluye avances en herramientas de expresión, video, calidad de las experiencias de usuario, y la creación de contenido móvil. La nueva herramienta de aceleración personalizada permite un control preciso sobre la animación. El *FlashType*, es un revolucionario motor de renderizado de fuentes, garantiza una visión

clara y de alta calidad de texto. Estos rasgos expresivos nuevos elevan el listón de la calidad de los negocios y sitios *web* individuales y mejoran la experiencia digital. (30)

### **1.10.13 Fireworks**

*Adobe Fireworks* es un programa versátil para crear, editar y optimizar gráficos *web*. Permite crear, editar imágenes de mapa de *bits* y vectoriales, diseñar efectos *web*, menús emergentes, recortar, optimizar elementos gráficos para reducir su tamaño de archivo y automatizar tareas repetitivas para ahorrar tiempo. Es posible exportar o guardar un documento como un archivo *JPEG*, un archivo *GIF* o un archivo de otro formato. Estos archivos pueden guardarse junto con archivos *HTML* que contengan tablas *HTML* y código *JavaScript* para facilitar su uso en Internet. (31)

Destinado para el manejo de gráficos vectoriales con gráficos en mapa de *bits* y que ofrece un ambiente eficiente tanto para la creación rápida de prototipos de sitios *web* e interfaces de usuario, como para la creación y optimización de imágenes para *web*. (31)

Proporciona las herramientas que necesita para crear gráficos de expresión, altamente optimizado para la *web*. *Fireworks* está disponible de forma individual o integrada en *Adobe CS3/CS4/CS5* y por tanto ha sido diseñado para integrarse con otros productos de *Adobe*, como *Dreamweaver* y *Flash*. En la *suite* de *Adobe* se identifica por usar el color amarillo, color que venía usando como representación desde que pertenecía a *Macromedia*. (31)

### **1.10.14 Hojas de estilo en cascada (CSS)**

Hojas de Estilo en Cascada (*Cascading Style Sheets (CSS)*), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

CSS se utiliza para dar estilo a documentos *HTML* y *XML*, separando el contenido de la presentación. Los estilos definen la forma de mostrar los elementos *HTML* y *XML*. CSS permite a los desarrolladores *web* controlar el estilo y el formato de múltiples páginas *web* al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento. (32)

### **1.10.15 Lenguaje de Marcación de Hipertexto (HTML)**

Es el lenguaje de marcado predominante para la elaboración 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. El término *HTML* se suele referir a ambas cosas, tanto al tipo de documento como al lenguaje de marcas. (33)

El entorno para trabajar *HTML* es simplemente un procesador de texto, como el que ofrecen los sistemas operativos *Windows* (Bloc de notas), *Unix* (el editor *vi* o *ed*) o el que ofrece *MS office* (*Word*). El conjunto de etiquetas que se creen, se deben guardar con la extensión *.htm* o *.html*. Estos documentos pueden ser mostrados por los visores o "*browsers*" de páginas *web* en Internet, como *Netscape Navigator*, *Mosaic*, *Opera* y *Microsoft Internet Explorer*. También existe el *HTML* Dinámico (*DHTML*), que es una mejora de *Microsoft* de la versión 4.0 de *HTML* que le permite crear efectos especiales como, por ejemplo, texto que vuela desde la página, palabra por palabra o efectos de transición al estilo de anuncio publicitario giratorio entre página y página. (33)

## **1.11 Conclusiones**

Al realizar un estudio de las tendencias actuales de los procesos de prueba de algoritmos biométricos, se examinaron las principales campañas de evaluación de sistemas biométricos en sus diferentes modalidades, observando cómo se han ido incorporando cada año empresas y participantes, ampliándose de esta forma el número de competencias a realizarse, con el objetivo principal de mejorar cada uno de los sistemas biométricos sometidos a dichas evaluaciones. Posibilitando realizar un seguimiento de los avances en las tecnologías de reconocimiento de huellas dactilares, a través de pruebas independientes que van actualizándose continuamente. Con el estudio de las herramientas y tecnologías existentes fue posible realizar una adecuada selección de las mismas, para dar solución a la problemática planteada. Además, su análisis permitió conocer sus características y ventajas haciendo posible que los resultados obtenidos con su posterior aplicación sean lo más factible y eficiente posible.

## **Capítulo 2: Características del sistema**

### **2.1 Introducción**

En el presente capítulo se realiza la descripción de la propuesta de solución de la investigación. Para dar cumplimiento a lo anteriormente planteado se confecciona la descripción gráfica del Modelo de dominio, el mismo permite mostrar de manera visual los principales conceptos que se manejan en la aplicación. Se identifican los requisitos funcionales que posee el sistema, los cuales serán representados mediante diagramas de casos de uso, estos forman parte de los Diagramas de Caso de Uso del Sistema (DCUS), en ellos estarán presentes las relaciones de los actores con los casos de uso del sistema y las secuencias de acciones con las que interactúan. Se identifican además los requisitos no funcionales, los cuales especifican las propiedades o cualidades que el producto debe tener. Además se exponen los diagramas de colaboración por cada funcionalidad del sistema.

### **2.2 Propuesta del sistema**

La solución propuesta es una aplicación *web*, en la cual es posible probar algoritmos de identificación por huella dactilar. Al acceder a la aplicación *web*, el usuario debe llenar una solicitud con sus datos y de esta forma quedar registrado. Una vez introducidos los datos solicitados al usuario, seleccionado el tipo de biometría y la base de datos de referencia, el mismo genera distribuciones de puntajes (*scores*), las cuales van a ser utilizadas posteriormente para obtener el resultado final de la prueba. El reporte del resultado es almacenado en una base de datos, donde se pueden consultar posteriormente. Los datos obtenidos son criterios de evaluación, la Tasa de Falsos Aceptados (FAR), la Tasa de Falsos Rechazados (FRR) y la Tasa de Error Igual (EER). Estos criterios pueden ser consultados a través de sus valores o a través de una gráfica generada por los mismos.

### **2.3 Modelo de dominio**

El modelo de dominio ayuda a comprender los conceptos que utilizan los usuarios. Este modelo permite mostrar de manera visual los principales conceptos que se manejan, ayudando a los usuarios, desarrolladores e interesados; a utilizar un vocabulario común para poder entender el contexto en que se desarrolla el sistema. Además contribuye a identificar personas, eventos, transacciones y objetos involucrados en el sistema. (15)

Para la realización de la aplicación, fue necesaria la elaboración de un modelo de dominio, debido que no se identificaron procesos del negocio. Este modelo contribuye a describir las clases más importantes dentro del contexto de la aplicación. A continuación se identifican los conceptos más significativos que se usaron en el modelo de dominio:

**Usuario:** es el encargado de realizar la prueba del algoritmo, usualmente es la misma persona que implementa el algoritmo que desea probar. La primera acción que se realiza para hacer la prueba es acceder al portal *web*, para de esta forma ejecutar la **Aplicación Biométrica**, en la cual el usuario después de autenticado procede a cargar el algoritmo que se desea probar, para esta tarea, utiliza una base de datos de referencia (**BDReferencia**). Las plantillas biométricas almacenadas en la base de datos son pasadas manualmente al algoritmo en dependencia del valor que desea obtener. Las bases de datos de referencia (**BDReferencia**), son diferentes muestras biométricas que fueron elegidas en dependencia del algoritmo que se va a probar. Las pruebas que se realizan son solo de identificación dactilar, como se muestra en la figura 7.

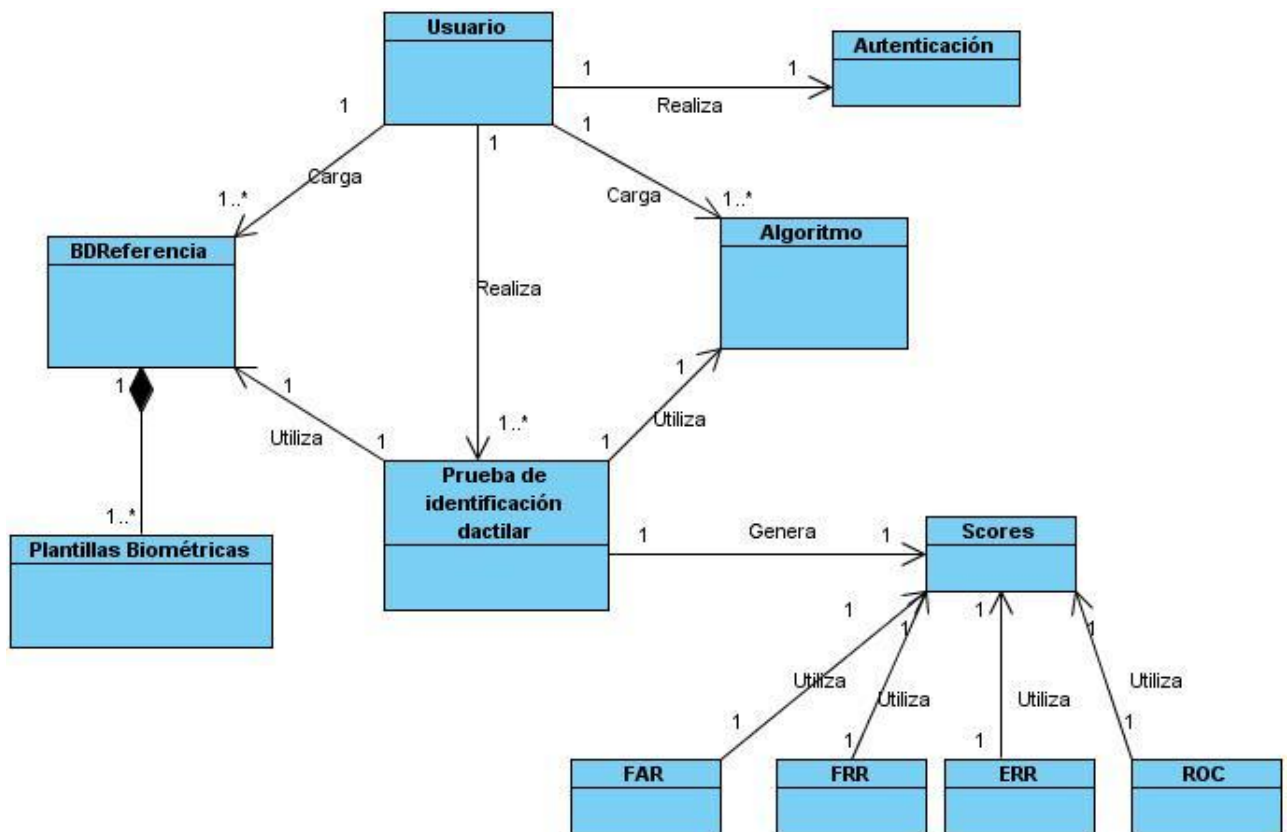


Figura 7: Modelo de dominio

## 2.4 Requisitos

Los requisitos son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que se especifiquen. (34)

Los objetivos del flujo requisitos son:

- Establecer y mantener un acuerdo entre clientes sobre lo que el sistema podría hacer.
- Proveer a los desarrolladores un mejor entendimiento de los requisitos del sistema.
- Definir el ámbito del sistema.
- Proveer una base para la planeación de los contenidos técnicos de las iteraciones.
- Proveer una base para estimar costos y tiempo de desarrollo del sistema.
- Definir una interfaz de usuarios para el sistema, enfocada a las necesidades y metas del usuario.(34)

Para capturar los requisitos es preciso entrevistar a todos los interesados en el proyecto, no sólo a los usuarios finales y luego anotar todas sus peticiones. A partir de ellas hay que descubrir lo que necesitan y expresarlo en forma de requisitos. En este flujo de trabajo, y como parte de los requisitos de facilidad de uso, se diseña la interfaz gráfica de usuario. Para ello habitualmente se construyen prototipos de la interfaz gráfica de usuario que se contrastan con el usuario final. (34)

### 2.4.1 Requisitos funcionales

Los requisitos funcionales especifican acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física. (13)

#### **RF 1 Registrar usuario.**

RF 1.1 Introducir usuario.

RF 1.2 Introducir correo.

RF 1.3 Validar los datos.

RF 1.4 Almacenar los datos.

RF 1.5 Enviar correo electrónico.



## **RF 2 Realizar Autenticación.**

RF 2.1 Introducir usuario.

RF 2.2 Introducir contraseña.

RF 2.3 Verificar usuario y contraseña.

## **RF 3 Realizar prueba.**

RF 3.1 Introducir nombre del algoritmo biométrico.

RF 3.2 Introducir la versión.

RF 3.3 Seleccionar la categoría del algoritmo que se desea probar (huellas).

RF 3.4 Seleccionar y cargar la base de datos de referencia la cual se utilizará para probar el algoritmo biométrico.

RF 3.5 Cargar la aplicación biométrica (se refiere al fichero que contiene al algoritmo.)

RF 3.6 Obtención de los scores generados.

RF 3.7 Obtener los valores del FAR, FRR y EER.

RF 3.8 Graficar los resultados.

RF 3.9 Notificar al usuario que ya la prueba se realizó.

## **RF 4 Realizar reporte de pruebas.**

RF 4.1 Cargar y visualizar los resultados de la base de datos.

## **RF 5 Gestionar base de datos.**

RF 5.1 Adicionar base de datos.

RF 5.2 Eliminar base de datos.

RF 5.3 Listar base de datos.

## 2.4.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. (13)

### RNF 1. Apariencia o interfaz externa

- La interfaz de esta aplicación debe ser sencilla y de fácil uso.
- La interfaz debe tener colores refrescantes a la vista del usuario.

### RNF 2. Usabilidad

- La aplicación *web* debe ser de fácil manejo para los usuarios, de este modo se logra una mejor interacción entre ella y el cliente.

### RNF 3. Rendimiento

- Los tiempos de respuesta deben ser rápidos con el fin de hacer más eficiente y veloz el proceso de pruebas a los algoritmos biométricos.

### RNF 4. Software

- La plataforma debe ser montada sobre el sistema operativo *Windows Server 2008*.

### RNF 5. Hardware

- El sistema requiere como mínimo 1Gb de RAM, un microprocesador con velocidad de 1.0 GHz y 32Mb de memoria de video.

## 2.5 Diagrama de caso de uso del sistema

Una vez capturados los requisitos funcionales del sistema, estos se representarán mediante un diagrama de casos de uso. Para ello primero se debe definir claramente cuáles son los actores que van a interactuar con el sistema y los Casos de Uso que representarán todas las funcionalidades del sistema.

Un **diagrama de casos de uso** muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa. Los elementos que pueden aparecer en un diagrama de casos de uso son: actores, casos de uso y relaciones entre casos de uso.

Un **actor** es una entidad externa al sistema que realiza algún tipo de interacción con el mismo. Un **caso de uso** es una descripción de la secuencia de interacciones que se producen entre un actor y el

sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. El nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema. (15)

La aplicación cuenta con dos actores el usuario y el administrador. La primera acción que se efectúa, antes de realizar cualquier operación en la aplicación es la autenticación. El usuario es el encargado de cargar la aplicación biométrica y probar los algoritmos. Luego de cargado el algoritmo y de realizada la prueba puede observar los reportes generados. El administrador se encarga de gestionar las bases de datos, como se observa en la figura 8.

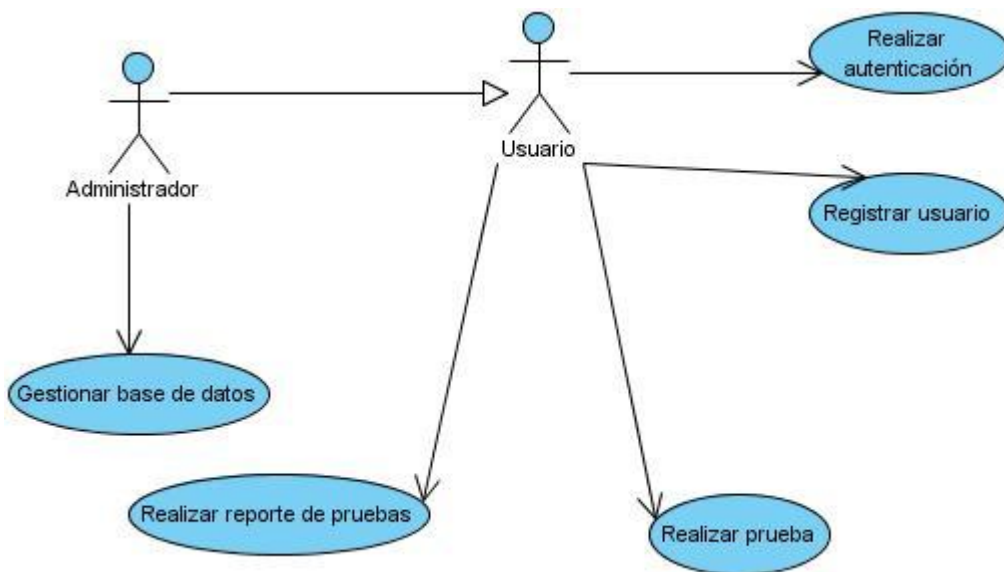


Figura 8: Diagrama de casos de uso del sistema

## 2.6 Especificaciones de casos de uso

Las especificaciones de casos de uso son una de las herramientas principales para el usuario, ya que posibilita la descripción detallada de cada uno de los casos de uso del sistema, describiendo paso a paso las acciones que realizan tanto el sistema como el actor. Las descripciones de las principales funcionalidades del sistema serán mostradas a continuación. (13)

En este caso con el sistema interactúan dos actores, el Usuario y el Administrador. El usuario es el que realiza las pruebas y el administrador se encarga de gestionar las bases de datos, como se muestra en la tabla 1.

Tabla 1: Actores del sistema.

Actor	Descripción
Usuario	Es la persona encargada de cargar el algoritmo al que se le desea realizar la prueba.
Administrador	Es la persona encargada de gestionar las bases de datos. Este proceso incluye adicionar y eliminar bases de datos.

Tabla 2: Descripción del caso de uso 1.

Nombre del caso de uso: Registrar usuario.	
<b>Actores</b>	Usuario.
<b>Propósito</b>	Registrar un usuario nuevo, para que después pueda acceder con su usuario y contraseña.
<b>Resumen</b>	El caso de uso se inicia cuando el usuario selecciona la opción "registrarse" y procede a llenar los datos que se le solicitan.
<b>Referencia</b>	RF1, RF1.1, RF1.2, RF1.3, RF1.4, RF1.5.
<b>Precondiciones</b>	El usuario no debe estar registrado con anterioridad.
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. El usuario selecciona la opción "Registrarse".	2. El sistema muestra una interfaz con los datos que el usuario debe ingresar. 2.1 Opción "Usuario". 2.2 Opción "Correo electrónico".
3. El usuario introduce los datos.	4. El sistema genera una contraseña para el usuario y la almacena junto con los datos introducidos por el usuario.
	5. El sistema envía un correo electrónico al usuario informándole la contraseña.

<b>Flujo Alternativo</b>	
1.1 Si el usuario deja algún campo vacío.	1.2 El sistema muestra un mensaje informando que debe llenar todos los campos.
<b>Poscondiciones</b>	Se realiza el registro, posibilitándole al usuario la autenticación.
<b>Prototipos de Interfaz de usuario</b> <a href="#">Ver Anexo 1.</a>	

Tabla 3: Descripción del caso de uso 2.

<b>Nombre del caso de uso:</b> Realizar autenticación.	
<b>Actores</b>	Usuario.
<b>Propósito</b>	Realizar la autenticación al sistema, para de esta forma tener acceso a todas las opciones que se brindan.
<b>Resumen</b>	El caso de uso se inicia cuando el usuario introduce los datos para acceder a la aplicación, se verifican si son correctos, dándole los permisos que tiene y habilitándole o denegándole la entrada al sistema.
<b>Referencia</b>	RF2, RF2.1, RF2.2, RF2.3.
<b>Precondiciones</b>	Que el usuario esté registrado en el sistema.
<b>Flujo normal de eventos</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
1. El usuario introduce su "Usuario" y "Contraseña".	2. El sistema recibe la solicitud y comprueba si la identidad es válida verificando que esté compuesta por los datos requeridos.
	3. El sistema posibilita el acceso a la aplicación.
<b>Flujo Alternativo</b>	
1.1 Si el usuario entra una identidad o contraseña no válida y solicita	1.2 El sistema verifica los datos y deniega el acceso.

entrar al sistema.	
1.3 Si el usuario no está registrado aún y quiere acceder al sistema.	1.4 El sistema verifica que el usuario no está registrado.
<b>Poscondiciones</b>	Se realiza la autenticación y se posibilita el acceso.
<b>Prototipos de Interfaz de usuario</b> <a href="#">Ver Anexo 2.</a>	

Tabla 4: Descripción del caso de uso 3.

<b>Nombre del caso de uso:</b> Realizar prueba.	
<b>Actores</b>	Usuario.
<b>Propósito</b>	Cargar la aplicación a la que se le desea realizar la prueba, almacenar los datos introducidos por el usuario y notificarle que su prueba se realizó.
<b>Resumen</b>	El caso de uso inicia cuando el usuario selecciona la opción “Subir algoritmo”, para ello tiene que llenar los datos que se le solicitan y cumplir con los parámetros establecidos en la creación del ejecutable a probar. El algoritmo correspondiente se analiza por el sistema. Luego de la obtención de todos los valores y graficados los resultados, el sistema le envía un correo electrónico al usuario notificándole que su prueba se realizó.
<b>Referencia</b>	RF 3, RF3.1, RF 3.2, RF 3.3, RF 3.4, RF 3.5, RF 3.6, RF 3.7, RF 3.8, RF 3.9.
<b>Precondiciones</b>	El usuario debe encontrarse autenticado.
<b>Flujo normal de eventos</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
1. El usuario selecciona la opción “Subir algoritmo”.	2. El sistema muestra una interfaz con los datos que el usuario debe ingresar y son necesarios para realizar este proceso. a) Opción “Algoritmo” (se refiere al tipo de algoritmo ejemplo: huella dactilar).

	<p>b) Opción “Base de datos” (se refiere a cargar la base de datos a utilizar en este caso es <i>Innovatrics</i>).</p> <p>c) Opción “Nombre del sistema”.</p> <p>d) Opción “Versión”.</p> <p>e) Opción “Aplicación “(se refiere al ejecutable .exe).</p>
3. El usuario introduce los datos.	4. El sistema almacena los datos en la base de datos.
	5. El sistema realiza la ejecución del algoritmo.
	6. El sistema analiza y carga las distribuciones de puntajes ( <i>scores</i> ) generados por el algoritmo.
	7. El sistema obtiene los valores de la FAR, FRR y EER.
	8. El sistema envía un correo electrónico al usuario informándole que ya su prueba está lista.
<b>Flujo Alternativo</b>	
3.1 Si el usuario deja algún campo vacío.	3.2 El sistema muestra un mensaje informando que debe llenar todos los campos.
6.1 Si el algoritmo cargado no cumple con los parámetros establecidos (devolver una distribución de puntaje en base a 100).	6.2 El sistema no realiza la prueba.
<b>Poscondiciones</b>	Se realiza la prueba correctamente y se notifica al usuario.
<b>Prototipos de Interfaz de usuario</b>  <a href="#">Ver Anexo 3.</a>	

Tabla 5: Descripción del caso de uso 4

<b>Nombre del caso de uso:</b> Realizar reporte de pruebas.	
<b>Actores</b>	Usuario.

<b>Propósito</b>	Mostrar los resultados de todas las pruebas que se han realizado.	
<b>Resumen</b>	El caso de uso inicia cuando el usuario selecciona la opción “Todos” del bloque reportes, donde podrá observar el reporte de todas las pruebas por usuario.	
<b>Referencia</b>	RF4, RF4.1.	
<b>Precondiciones</b>	El usuario debe encontrarse autenticado.	
<b>Flujo normal de eventos</b>		
<b>Acción del actor</b>	<b>Respuesta del sistema</b>	
1. El usuario selecciona la opción “Todos” del bloque reportes.	2. El sistema muestra una nueva interfaz con los resultados de las pruebas realizadas por los usuarios.	
3. El usuario selecciona opción “Ver”.	4. El sistema muestra todas las pruebas para el usuario que el seleccionó.	
5. El usuario selecciona la opción “Más”.	6. El sistema muestra todos los detalles de la prueba seleccionada (gráficas de FAR, FRR, EER, ROC y los resultados generales).	
<b>Flujo Alternativo</b>		
<b>Pos condiciones</b>	Mostrar los reportes correspondientes.	
<b>Prototipos de Interfaz de usuario</b>  <a href="#">Ver Anexo 4</a> , <a href="#">Anexo 4.1</a> , <a href="#">Anexo4.2</a> , <a href="#">Anexo 4.3</a> .		

Tabla 6: Descripción del caso de uso 5.

<b>Nombre del caso de uso:</b> Gestionar base de datos	
<b>Actores</b>	Administrador.
<b>Propósito</b>	Permitir realizar las operaciones del Gestionar base de datos como son



	listar, adicionar y eliminar.
<b>Resumen</b>	El caso de uso inicia cuando el administrador selecciona la opción “Gestionar base de datos”, posteriormente se muestra en forma de listado todas las bases de datos existentes, luego se efectúa la selección una de las acciones a realizar (“Adicionar” o “Eliminar”), con las acciones “Adicionar”, o “Eliminar” finaliza el Caso de Uso.
<b>Referencia</b>	RF 5, RF 5.1, RF 5.2, RF 5.3.
<b>Precondiciones</b>	El administrador tiene que estar autenticado en el sistema.
<b>Flujo normal de eventos</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
1. El caso de uso inicia cuando el administrador selecciona la opción “Gestionar base de datos”.	2. El sistema muestra la interfaz correspondiente a Gestionar base de datos con: a) Listado de todas las base de datos existentes. b) Opción “Adicionar”. c) Opción “Eliminar”.
3. El administrador realiza una de las siguientes acciones: a) Opción “Adicionar”. b) Opción “Eliminar”.	4. Según su selección: a) Si seleccionó “Adicionar”, ir a Sección 1: “Adicionar base de datos”. b) Si seleccionó “Eliminar”, ir a sección 2: “Eliminar base de datos”.
<b>Sección 1: “Adicionar base de datos”</b>	
<b>Sección del actor</b>	<b>Respuesta del sistema</b>
	4. El sistema muestra la interfaz para Adicionar una base de datos con el siguiente campo a llenar por el administrador. a) Campo “Examinar” (aquí el administrador carga la url donde se encuentra el archivo a subir, el mismo tiene que estar compactado).

5. El administrador carga el archivo compactado.	
6. Selecciona la opción "Subir".	
7. El administrador descompacta el archivo.	8. Continúa en el paso 2 del Flujo Normal.
<b>Sección 2: "Eliminar base de datos"</b>	
<b>Sección del actor</b>	<b>Respuesta del sistema</b>
	4. El sistema muestra un listado con las bases de datos existentes.
5. El administrador selecciona una base de datos de la lista.	
6. El administrador selecciona la opción "Eliminar".	7. El sistema elimina la base de datos seleccionada.
	8. Continúa en el paso 2 del Flujo Normal.
<b>Flujo Alternativo</b>	
4.1 Si el administrador adiciona un archivo que no esté compactado.	7.1 El sistema muestra un error, informándole que el archivo tiene que estar compactado.
<b>Poscondiciones</b>	Se adiciona o elimina la base de datos.
<b>Prototipos de Interfaz de usuario</b>	
<a href="#">Ver Anexo 5</a> , <a href="#">Anexo 5.1</a> , <a href="#">Anexo 5.2</a> .	

## 2.7 Modelo de análisis

El análisis consiste en obtener una visión del sistema, se preocupa de ver qué hace, de modo que sólo se interesa por los requisitos funcionales. Por otro lado el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva cómo cumple el sistema sus objetivos. Este modelo establece la realización de los casos de uso en clases y pasando desde una representación en

términos de análisis (sin incluir aspectos de implementación) hacia una de diseño (incluyendo una orientación hacia el entorno de implementación). Está constituido esencialmente por un Diagrama de Clases y algunos Diagramas de Estados para las clases que lo requieran. (34)

### **2.7.1 Diagrama de clases del análisis**

Un diagrama de clases es una descripción de las clases en un sistema y sus relaciones. No describe el comportamiento dinámico del sistema, por ejemplo el comportamiento de objetos individuales. El primer elemento de un diagrama de clases es una descripción de clases individuales. (13)

Las clases de análisis se centran principalmente en los requisitos funcionales, y son evidentes en el dominio del problema, porque representan conceptos y relaciones del dominio. Tienen atributos y se establecen relaciones entre ellas. Encajan en tres estereotipos básicos: (13)

**Clase Interfaz:** Modelan la interacción entre el sistema y sus actores.

**Clase Entidad:** Modelan información que posee larga vida y que es a menudo persistente.

**Clase Control:** Coordinan la realización de uno o unos pocos Casos de Uso coordinando las actividades de los objetos que implementan su funcionalidad. (13)

#### **Descripción de las clases del análisis:**

CI\_Index: se refiere a la interfaz principal de la aplicación.

CI\_Register: se refiere a la interfaz donde se muestra el formulario que permite registrar un usuario.

CI\_LogOn: en esta clase interfaz se muestra el formulario donde el usuario realiza la autenticación.

CI\_Upload: se refiere al proceso realizar prueba, en la cual se muestra un formulario donde se realiza el proceso cargar el algoritmo.

CI\_Reports: se refiere al proceso realizar reporte, en el cual el usuario observa los resultados de las pruebas efectuadas.

CI\_DeleteDB: en esta clase interfaz se realiza el proceso de eliminar la base de datos en el cual el usuario selecciona de la lista la base de datos a eliminar.

CI\_AddDB: en esta clase interfaz se adiciona una nueva base de datos, la cual posteriormente se mostrara en el listado de base de datos.

CI\_ListDB: en esta interfaz se listan las bases de datos existentes.

CC\_AccountController: en la presente controladora se encuentran los métodos de registrar y autenticar usuario.

CC\_TestController: en esta clase controladora se realiza en proceso de realizar la prueba.

CC\_ReportsController: se controla el proceso de realizar los reportes.

CC\_ManageController: en la presente controladora se gestiona todo el proceso de adicionar, eliminar y listar base de datos.

CE\_Users: se refiere a la clase entidad relacionada a los datos de los usuarios registrados en la base de datos y los reportes generados.

CE\_Membership: en la siguiente entidad se refiere a la relación con los usuarios autenticados en la aplicación.

CE\_Algorithm: la entidad algorithm se refiere a la relación de los datos para realizar la prueba y los reportes.

CE\_Test: la siguiente entidad se relaciona al proceso de realizar reporte.

### **Descripción de los actores:**

Usuario: se refiere al usuario de la aplicación, es el encargado de realizar todo el proceso relacionado con la prueba del algoritmo.

Administrador: se refiere al administrador del sistema, es el encargado de gestionar el proceso de las base de datos.

### **Diagramas de clases del análisis:**

El usuario accede a la interfaz principal y selecciona una operación de módulo derecho en la aplicación, luego se muestra la opción registrarse, en la cual debe llenar un formulario. Para la

realización de este proceso utiliza la entidad *Users*, para posteriormente acceder a la aplicación, como se muestra en la figura 9.

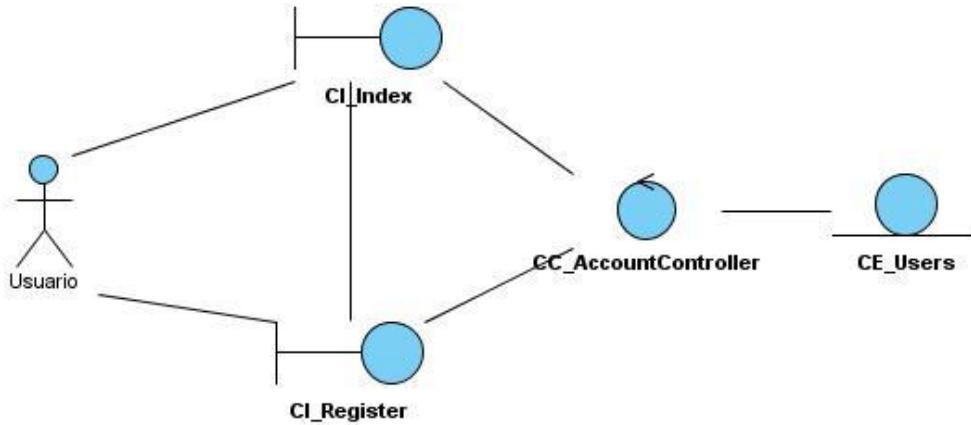


Figura 9: Diagrama de clases del análisis para el caso de uso **Registrar usuario**

El usuario accede a la interfaz principal, luego solicita alguna opción del módulo derecho y seguido se muestra el formulario para realizar la autenticación. Para realizar esta operación debe estar registrado. Luego de que el usuario introduzca los datos que se le solicitan puede realizar las operaciones que desee, como se muestra en la figura 10.

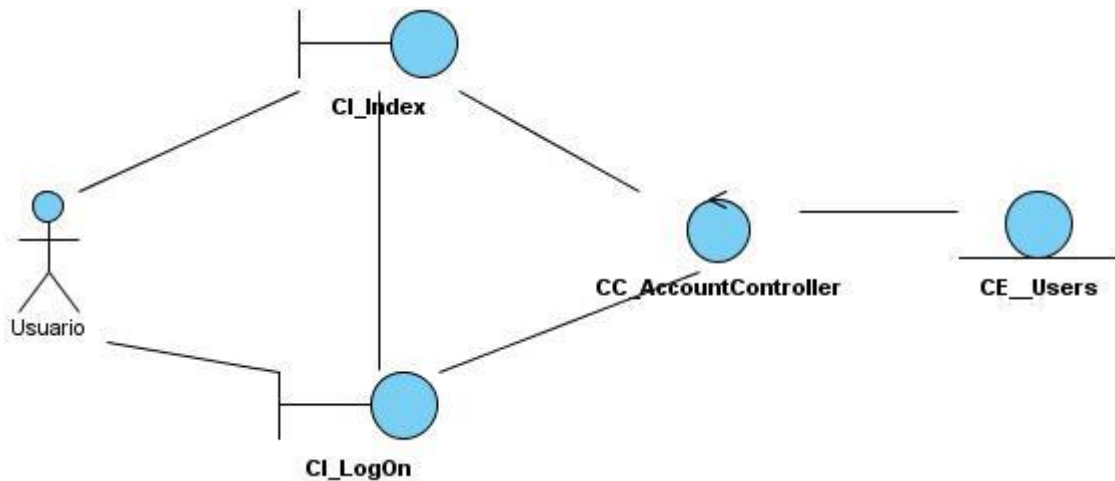


Figura 10: Diagrama de clases del análisis para el caso de uso **Realizar autenticación**.

El usuario accede a la interfaz principal, y selecciona la opción subir algoritmo para la cual debe llenar una serie de campos, utiliza como entidad *Algorithm*, como muestra en la figura 11.

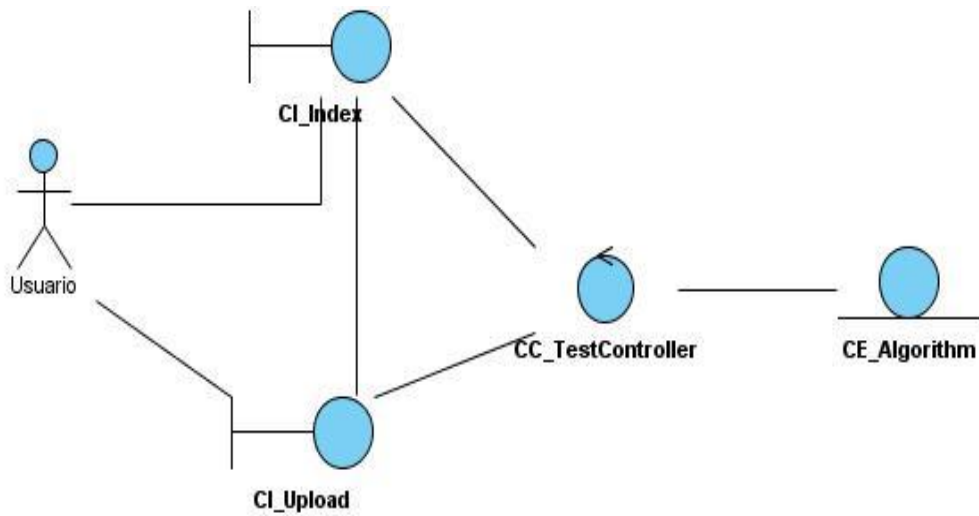


Figura 11: Diagrama de clases del análisis para el caso de uso **Realizar prueba**.

El usuario accede a la interfaz principal, y luego se procede a la realización del reporte en la cual se muestran todos los reportes de las pruebas realizadas, para esta tarea utilizará las entidades *Users*, *Algorithm* y *Test*, como se muestra en la figura 12

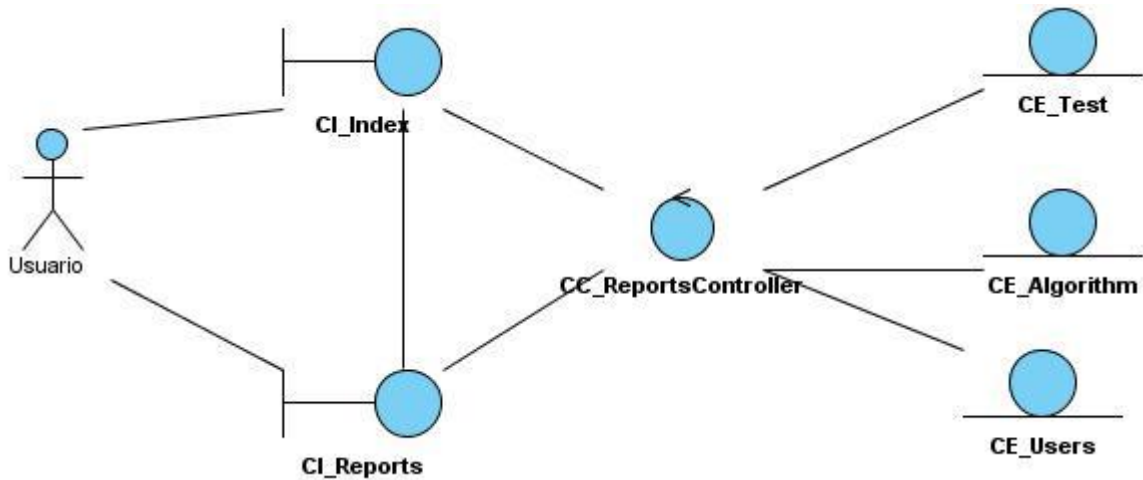


Figura 12: Diagrama de clases del análisis para el caso de uso **Realizar reporte de pruebas**.

El administrador desea gestionar las bases de datos, y para ello se muestra un listado de las base de datos existentes, luego puede eliminar alguna base de datos o adicionar base de datos. Para ello utiliza la entidad *ReferenceDB*, como se muestra en la figura 13.

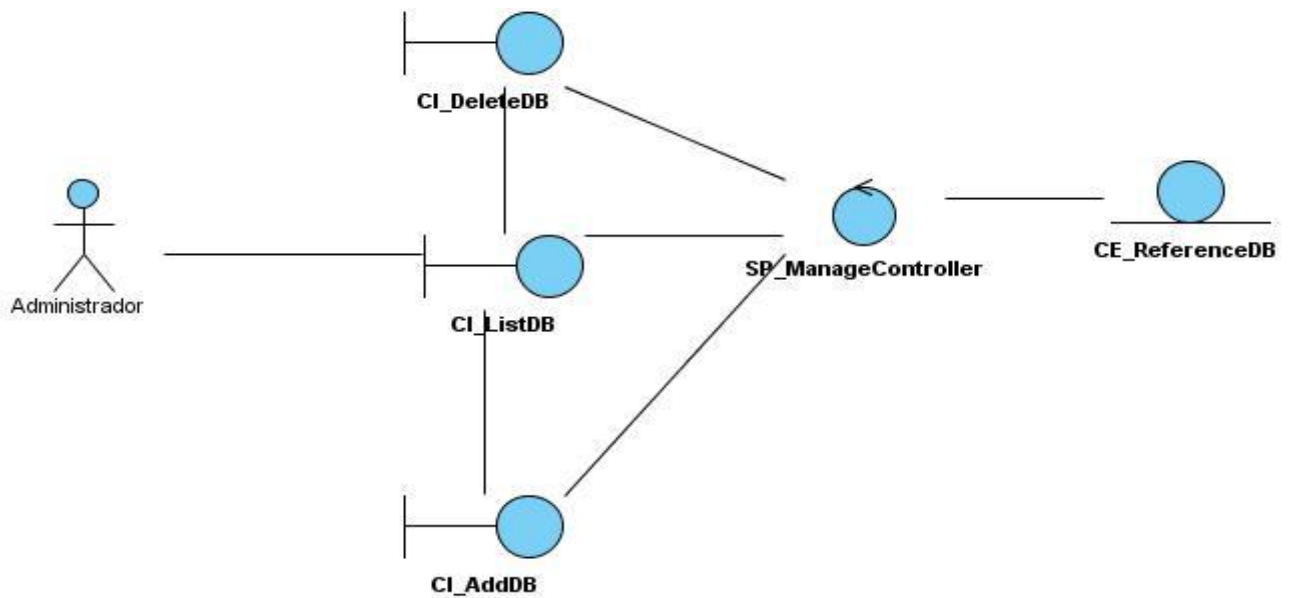


Figura 13: Diagrama de clases del análisis para el caso de uso **Gestionar base de datos**.

## 2.7.2 Diagramas de colaboración

Un diagrama de colaboración muestra una interacción organizada basándose en los objetos que toman parte en la interacción y los enlaces entre los mismos (en cuanto a la interacción se refiere). A diferencia de los diagramas de secuencia, los diagramas de colaboración muestran las relaciones entre los roles de los objetos. La secuencia de los mensajes y los flujos de ejecución concurrentes deben determinarse explícitamente mediante números de secuencia. (15)

En cuanto a la representación, un diagrama de colaboración muestra a una serie de objetos con los enlaces entre los mismos, y con los mensajes que se intercambian dichos objetos. Los mensajes son flechas que van junto al enlace por el que “circulan”, y con el nombre del mensaje y los parámetros (si los tiene) entre paréntesis. Cada mensaje lleva un número de secuencia que denota cuál es el mensaje que le precede, excepto el mensaje que inicia el diagrama, que no lleva número de secuencia. (15)

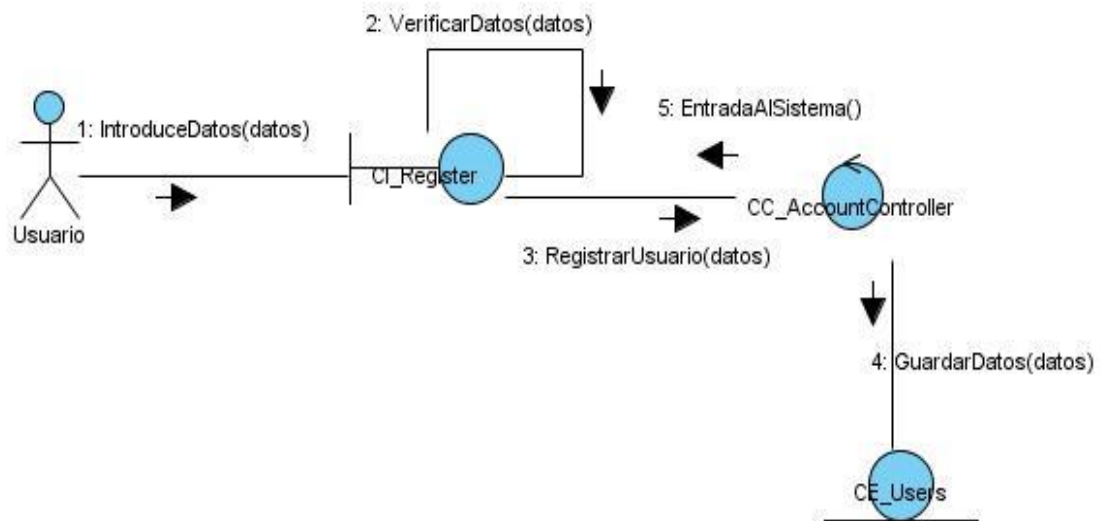


Figura 14: Diagrama de colaboración para el caso de uso **Registrar usuario**.

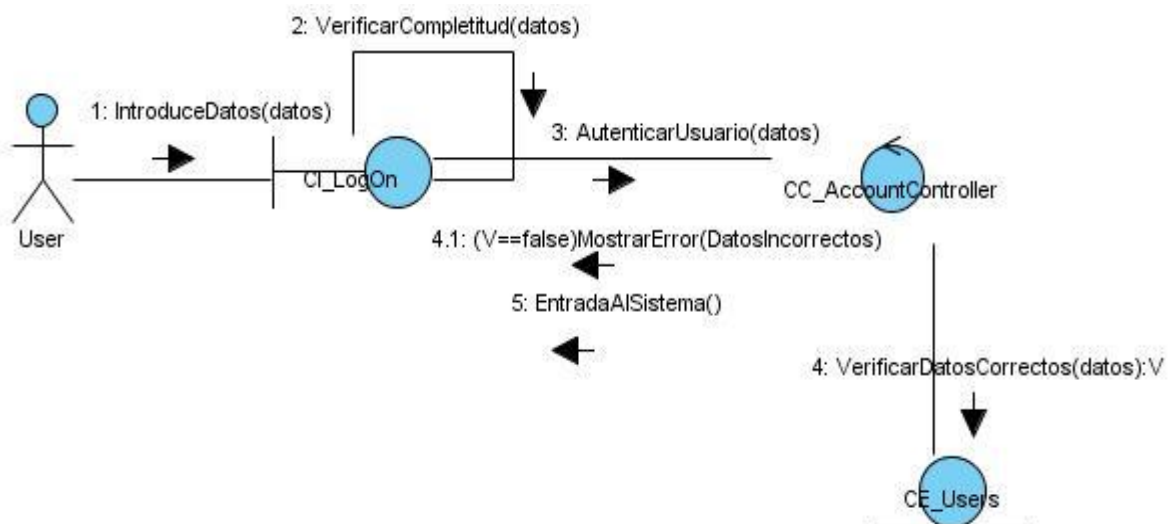


Figura 15: Diagrama de colaboración para el caso de uso **Realizar autenticación**.

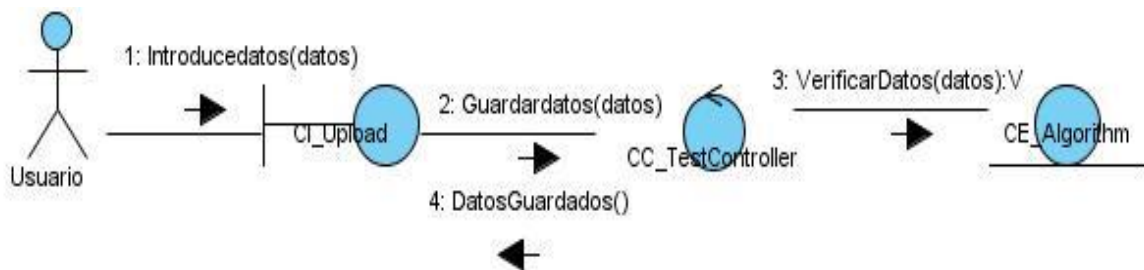


Figura 16: Diagrama de colaboración para el caso de uso Realizar prueba.



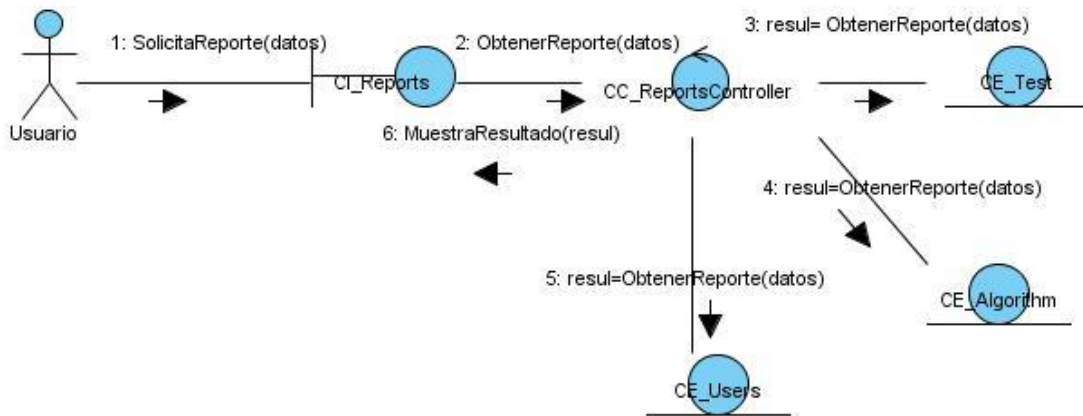


Figura 17: Diagrama de colaboración para el caso de uso **Realizar reporte de pruebas**.

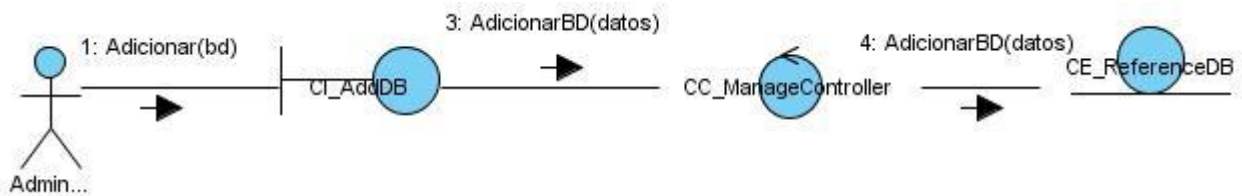


Figura 18: Diagrama de colaboración para el caso de uso **Gestionar base de datos** (escenario 1).

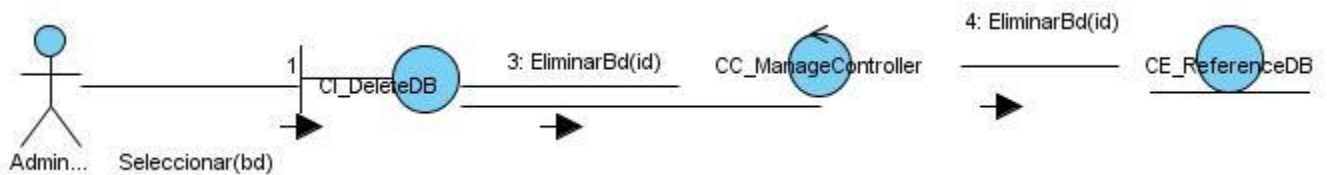


Figura 19: Diagrama de colaboración para el caso de uso **Gestionar base de datos** (escenario 2).

## 2.8 Conclusiones

Se realizó un modelo del dominio debido a que no existen procesos del negocio. Se efectuó un estudio detallado que permitió la confección de los requisitos funcionales y no funcionales del sistema, los cuales permitieron una visión más clara del producto a realizar debido a un análisis profundo y necesario para llevar a cabo el objetivo planteado. Se realizaron las correspondientes descripciones de los actores y casos de usos que intervienen en el sistema a partir de los diagramas de clase del análisis por cada caso de uso.

## Capítulo 3: Elaboración y construcción del sistema

### 3.1 Introducción

Durante la elaboración y construcción del sistema, se realiza un análisis detallado de la propuesta de solución. Se exponen los diagramas de clases del diseño y los de secuencia por funcionalidad. Se realiza un modelo de datos para describir la base de datos que almacena la información del sistema y una descripción de la arquitectura que se emplea. Este capítulo tiene como objetivo mostrar la descripción del sistema implementado a través del diagrama de componentes y visualizarlo mediante el modelo de despliegue, con la distribución de los nodos necesarios para el despliegue del sistema. Se explican a partir de escenarios las pruebas del sistema.

### 3.2 Diseño del sistema

El diseño del sistema desarrolla el detalle arquitectónico requerido para construir un sistema o producto. Contribuye a una arquitectura estable, sólida y crea un plano del modelo de implementación. Durante el diseño se tiene en cuenta los requisitos funcionales, formando el punto de partida para posteriormente realizar la implementación del sistema. (14)

El proceso de diseño del sistema abarca las siguientes actividades:

- Partición del modelo de análisis en subsistemas.
- Identificar la concurrencia dictada por el problema.
- Asignar subsistemas a procesadores y tareas.
- Desarrollar un diseño para la interfaz de usuario.
- Elegir una estrategia básica para implementar la administración (gestión) de datos.
- Identificar recursos globales y los mecanismos de control requeridos para su acceso. (14)

### 3.3 Modelo de clases del diseño

El **modelo de diseño** es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. El modelo de diseño sirve de abstracción a la implementación y se utiliza como una entrada fundamental de las actividades de implementación. (13)

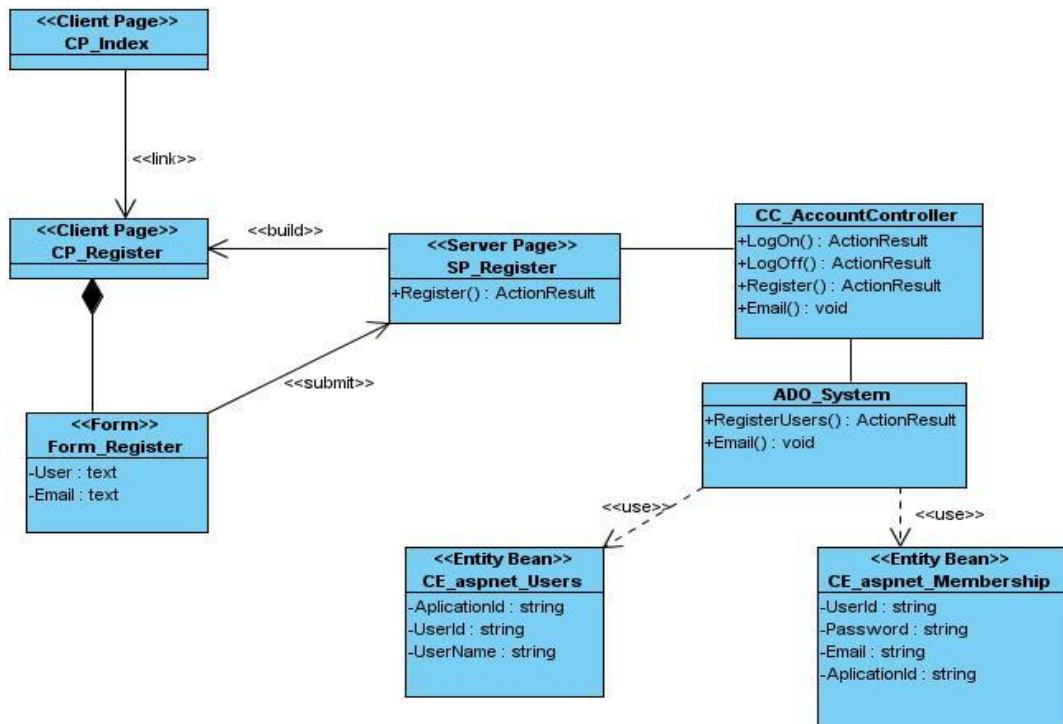


Figura 20: Diagrama de clases del diseño para el caso de uso **Registrar usuario**.

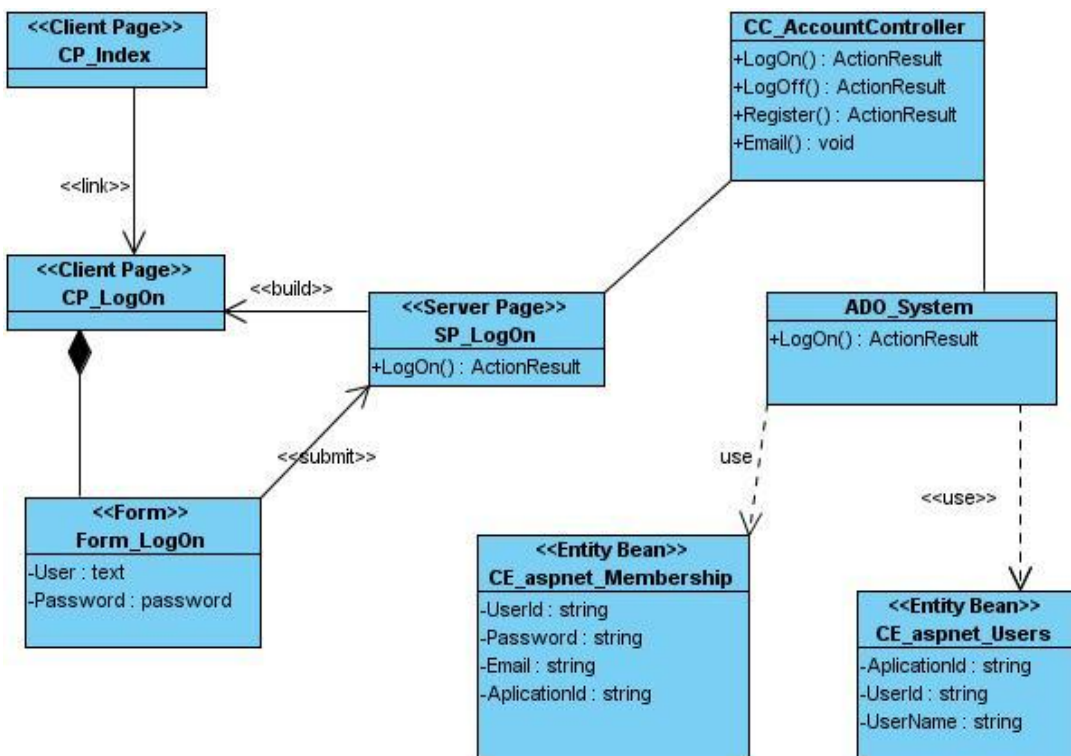


Figura 21: Diagrama de clases del diseño para el caso de uso **Realizar autenticación**.

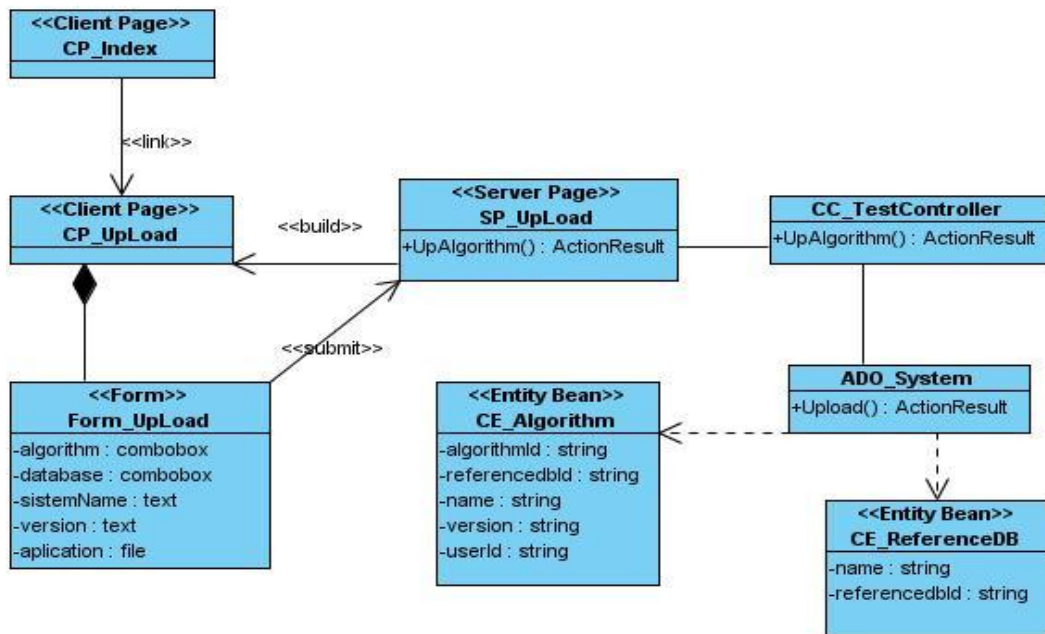


Figura 22: Diagrama de clases del diseño para el caso de uso **Realizar prueba**.

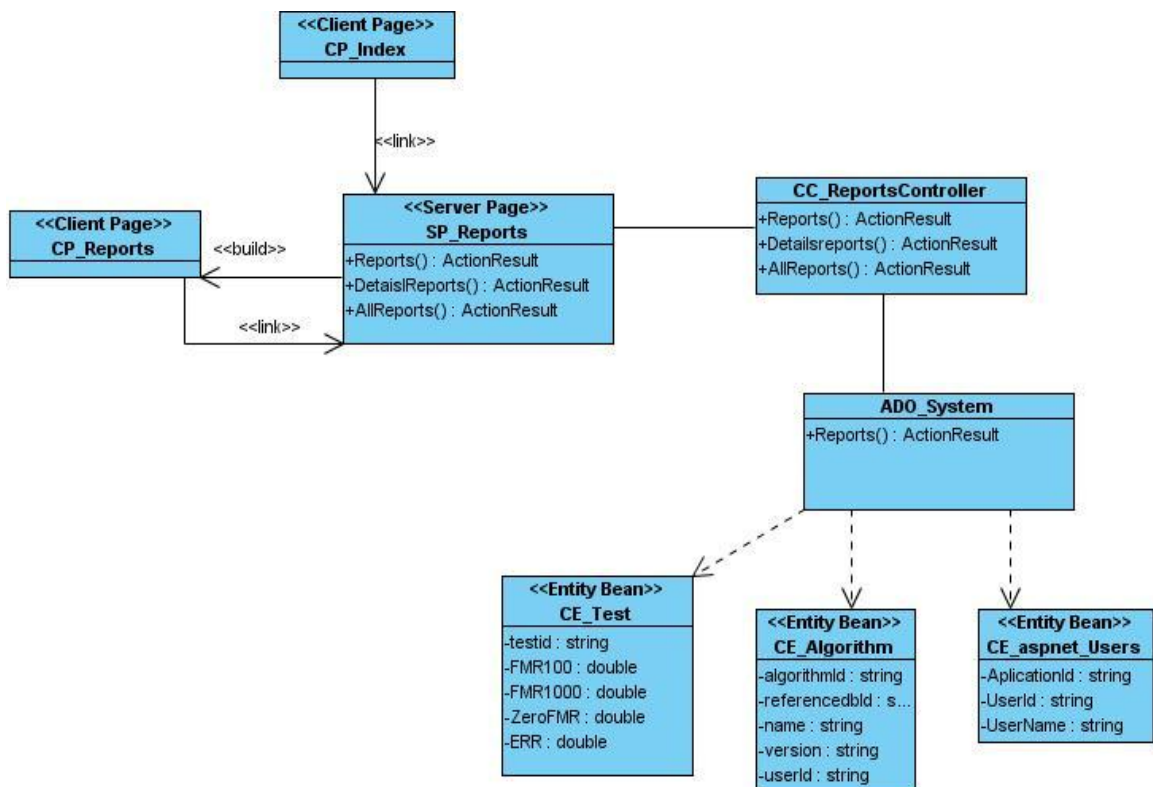


Figura 23: Diagrama de clases del diseño para el caso de uso **Realizar reporte de pruebas**.

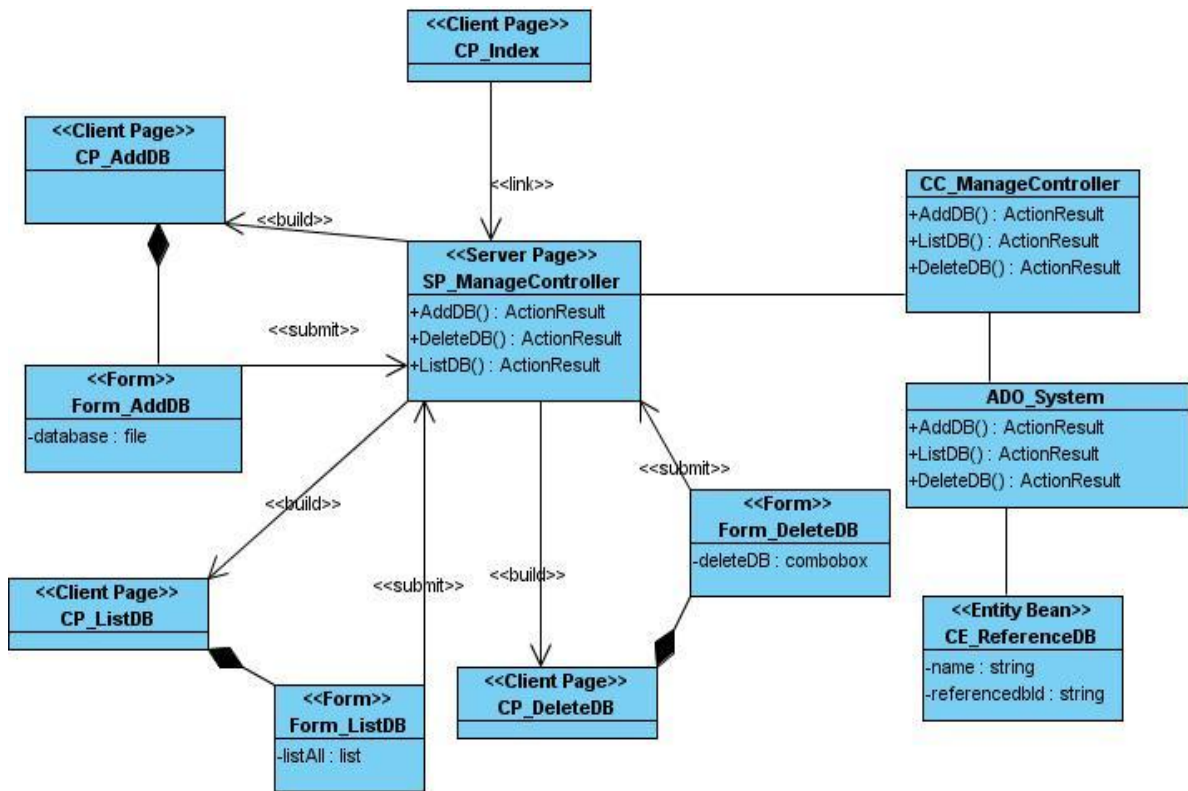


Figura 24: Diagrama de clases del diseño para el caso de uso **Gestionar base de datos**.

### 3.4 Diagramas de secuencia del diseño.

Un diagrama de secuencia muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo. El eje vertical representa el tiempo y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba abajo. Se pueden colocar etiquetas (como restricciones de tiempo, descripciones de acciones, etc.). (35)

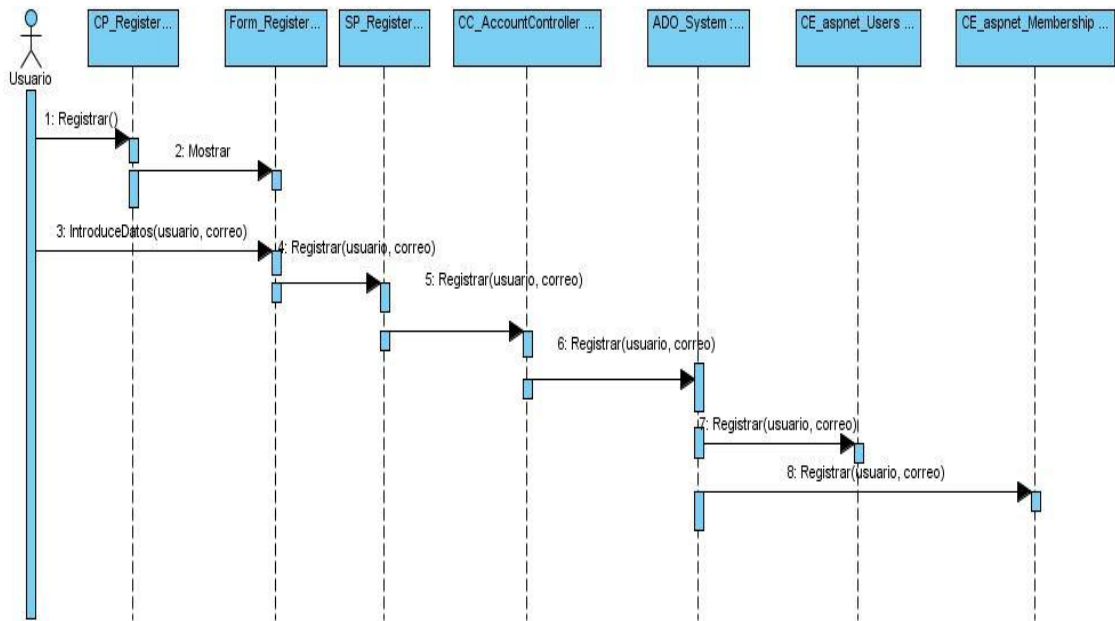


Figura 25: Diagrama de secuencia del diseño para el caso de uso **Registrar usuario**.

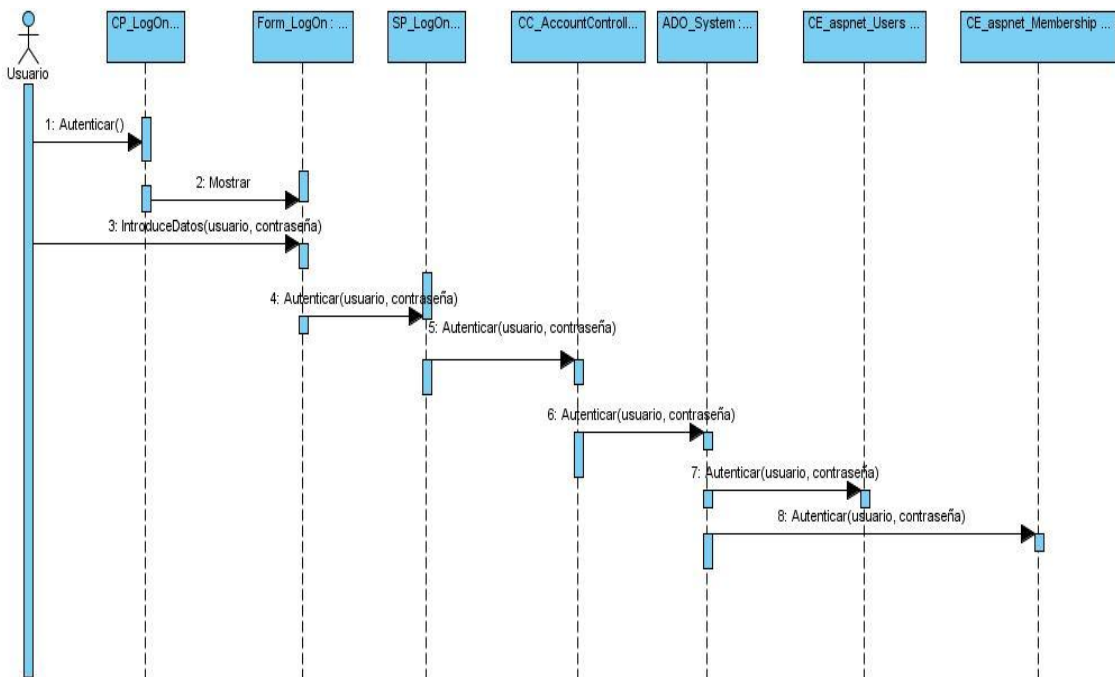


Figura 26: Diagrama de secuencia del diseño para el caso de uso **Realizar autenticación**.

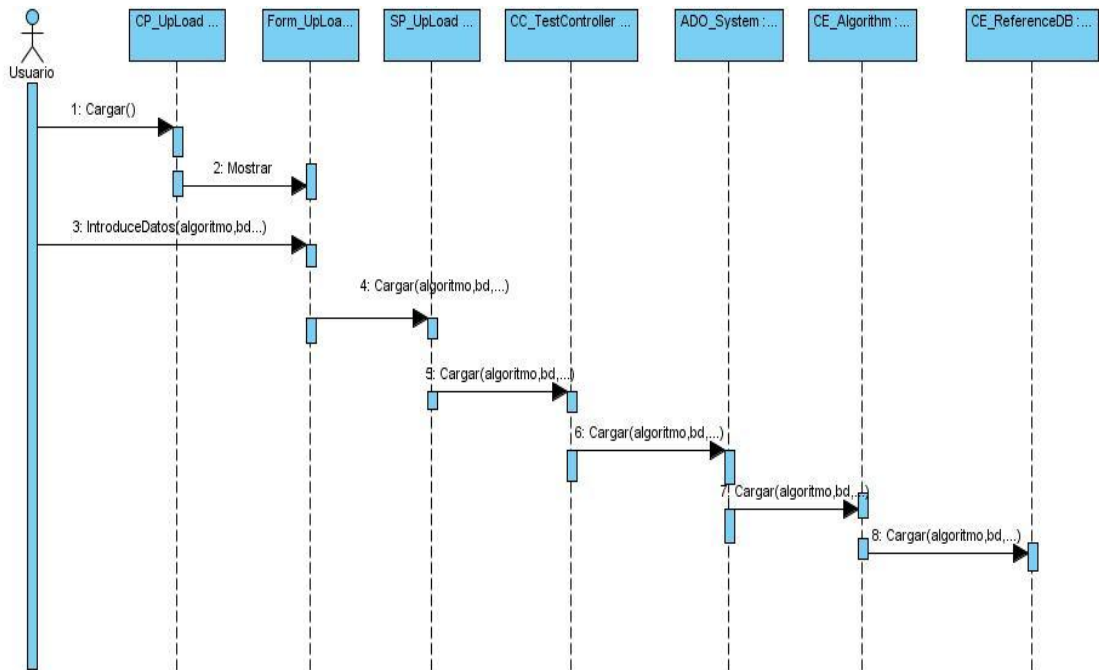


Figura 27: Diagrama de secuencia del diseño para el caso de uso **Realizar prueba**.

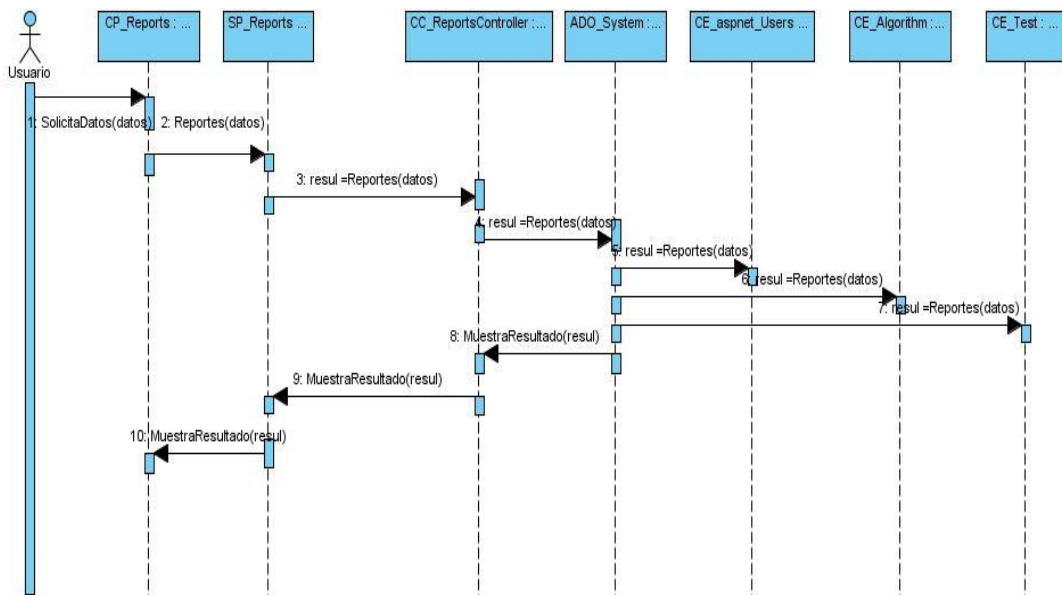


Figura 28: Diagrama de secuencia del diseño para el caso de uso **Realizar reporte de pruebas**.



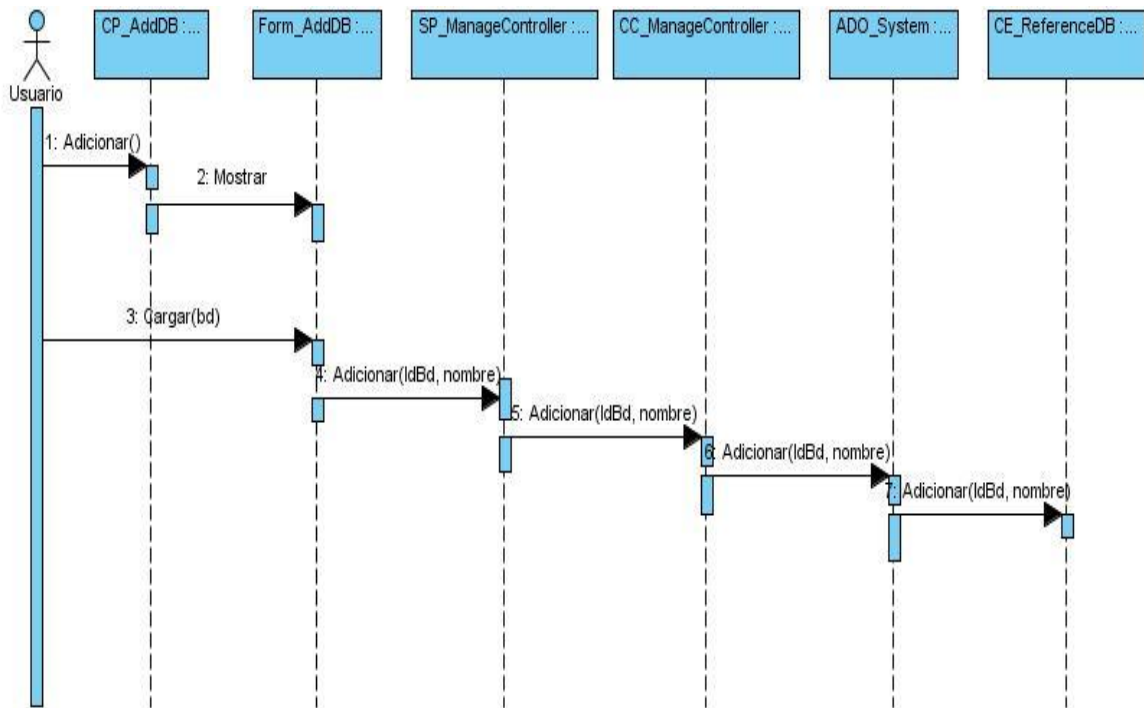


Figura 29: Diagrama de secuencia del diseño para el caso de uso **Gestionar base de datos** (escenario 1).

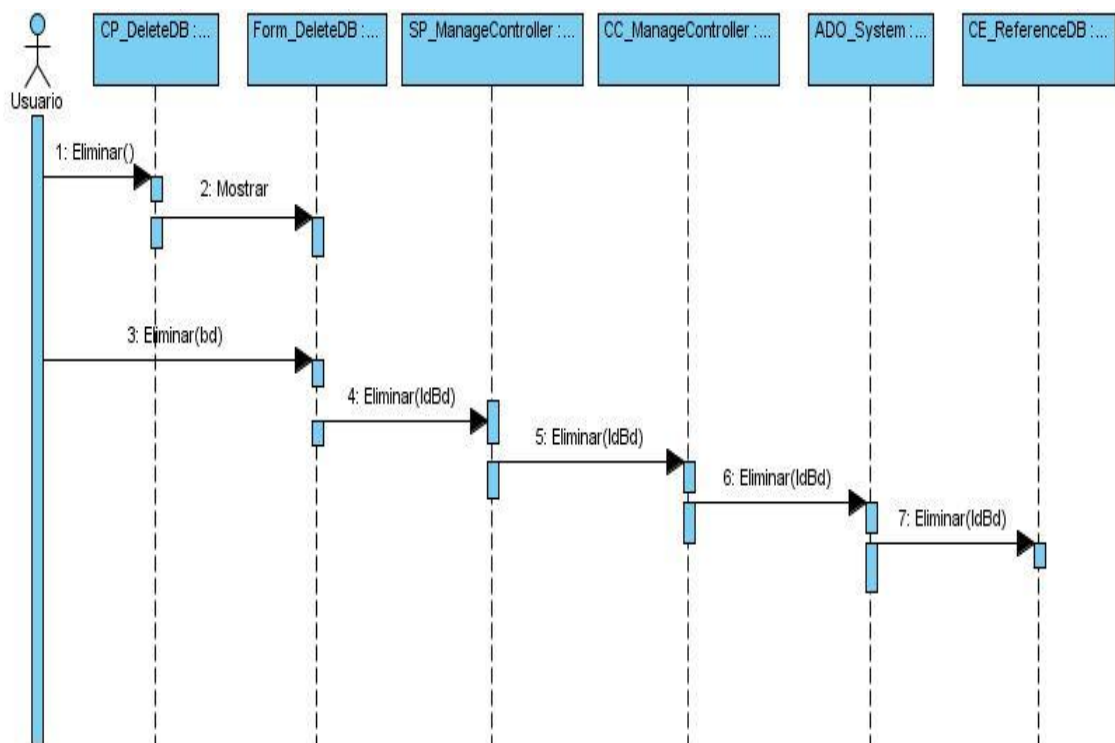


Figura 30: Diagrama de secuencia del diseño para el caso de uso **Gestionar base de datos** (escenario 2).



### 3.5 Modelo de datos

Un modelo de datos es un lenguaje utilizado para la descripción de una base de datos, permitiendo describir las estructuras de la base de datos, los elementos que intervienen en una realidad o en un problema dado y la forma en que se relacionan dichos elementos entre sí. A continuación se muestra el modelo físico de la base de datos que se utiliza para almacenar la información del sistema. Ver [anexo 6](#). (20)

### 3.6 Descripción de la arquitectura

El patrón Modelo Vista Controlador permite separar los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. El patrón de llamada y retorno MVC, se utiliza frecuentemente en aplicaciones *web*, donde la **vista** es la página HTML y el código que provee de datos dinámicos a la página. El **modelo** es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el **controlador** es el responsable de recibir los eventos de entrada desde la vista. (15)

**El modelo** incorpora la capa del dominio y persistencia, es el encargado de guardar los datos en un medio persistente (ya sea una base de datos, un archivo de texto, XML, registro, etcétera). Se refiere a la lógica del negocio o servicio y los datos asociados con la aplicación. La vista se encarga de presentar la interfaz al usuario.

**En la vista**, sólo se deben realizar operaciones simples. Es la encargada de la presentación de los datos.

**El controlador** es el responsable de recibir los eventos de entrada desde la vista. Es el que atiende las peticiones y componentes para la toma de decisiones de la aplicación. El propósito del MVC es aislar los cambios. Es una arquitectura preparada para los cambios, que separa los datos y lógica de negocio de la lógica de presentación. (15)

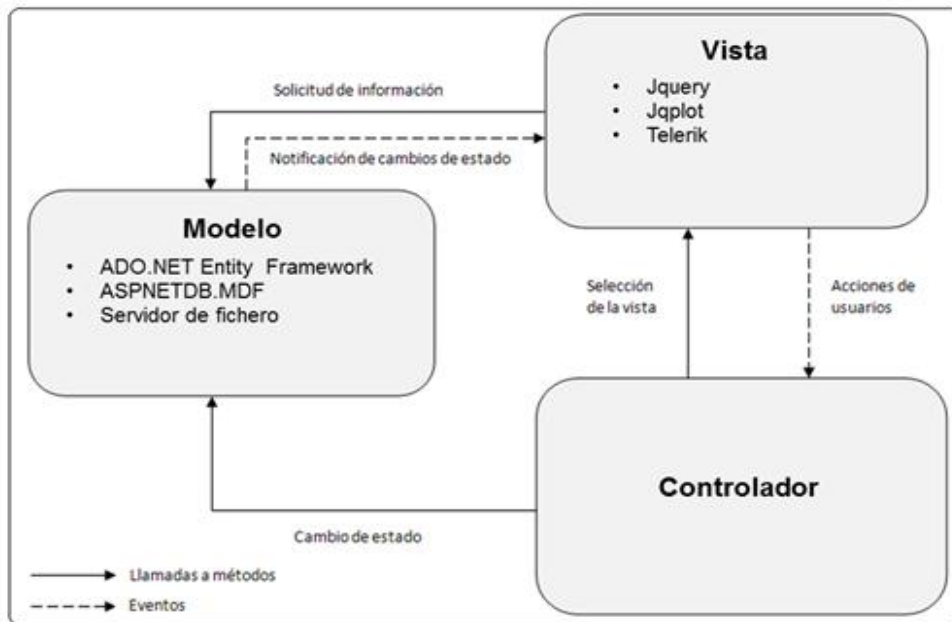


Figura 31: Patrón Modelo Vista Controlador.

### 3.7 Patrones del diseño

Los patrones de diseño son el dúo problema-solución a inconvenientes del diseño. Los patrones de diseño son los que brindan una solución ya probada y documentada a los problemas que pueden aparecer en el diseño del *software*. (15)

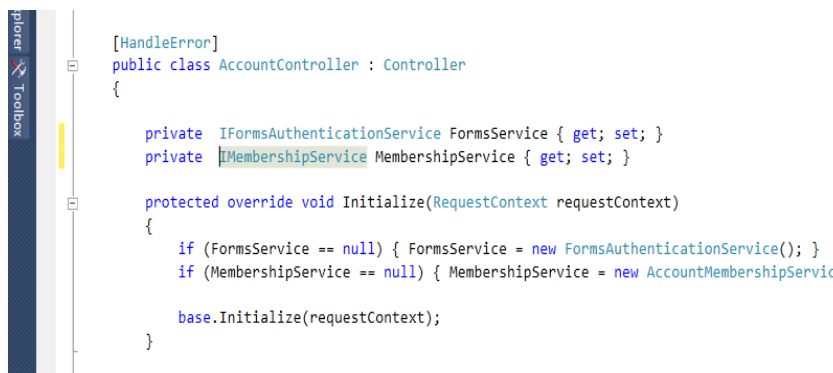
Una forma de reutilizar las soluciones a los problemas de diseño comúnmente presentes en el desarrollo de *software* es mediante el uso de los patrones de diseño. Estos describen un problema que ocurre en múltiples ocasiones en un mismo entorno, así como la solución al mismo, de tal modo que se puede utilizar esta solución varias veces sin tener que volver a pensarla. (35)

El uso de los patrones de diseño beneficia en gran medida el desarrollo y la calidad del *software*. Permite evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y corregidos anteriormente, además formaliza un vocabulario común entre diseñadores y estandariza el modo en que se realiza el diseño. (35)

Durante el desarrollo de la aplicación se han usado los patrones de diseño **GRASP** (*General Responsibility Assignment Software Patterns*), traducido al español como Patrones de Principios Generales para Asignar Responsabilidades:

**Creador:** Indica quién debe ser el responsable de la creación de una nueva instancia de alguna clase. Su uso en el desarrollo de este sistema brinda la posibilidad de mantenimiento, mayor claridad y reutilización. (35)

La Figura 32 presenta un ejemplo del patrón Creador en el sistema. La clase *AccountController* es la clase encargada de crear las instancias que serán necesarias en el transcurso de la implementación.

The image shows a screenshot of a code editor window in Visual Studio. The code is for a class named AccountController, which inherits from Controller. It has two private properties: FormsService of type IFormsAuthenticationService and MembershipService of type IMembershipService. The Initialize method is overridden and contains logic to instantiate FormsService and MembershipService if they are null. The code is as follows:

```
[HandleError]
public class AccountController : Controller
{
    private IFormsAuthenticationService FormsService { get; set; }
    private IMembershipService MembershipService { get; set; }

    protected override void Initialize(RequestContext requestContext)
    {
        if (FormsService == null) { FormsService = new FormsAuthenticationService(); }
        if (MembershipService == null) { MembershipService = new AccountMembershipService(); }

        base.Initialize(requestContext);
    }
}
```

Figura 32: Representación del patrón Creador en la aplicación.

**Controlador:** Asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Este funciona como intermediario entre los datos del usuario y las clases según los métodos necesarios. Ha sido usado en la implementación del sistema separado por capas, aislando las funciones: lógica del servicio, lógica del negocio y lógica de datos. (35)

La figura 33 presenta un ejemplo del patrón Controlador en el sistema. La clase *TestController*, es la encargada de controlar el evento que atiende una petición del usuario que luego responde mediante clases y métodos.

```

[Authorize]
public class TestController : Controller
{

public ActionResult Upload()
{

string[] bdr = Directory.GetFileSystemEntries(ConfigurationManager.ConnectionStrings["bdreferencia"].ConnectionString);
string bd = "";

foreach (var item in bdr)
{
bd += item.Split('\\')[1] + "/";
}
ViewData["bd"] = bd;
return View();
}

[HttpPost]
public ActionResult Upload(string algoritmo,string bd,string nombre_sis,string version, HttpPostedFileBase consola)
{

if (String.IsNullOrEmpty(algoritmo) || String.IsNullOrEmpty(bd) || String.IsNullOrEmpty(nombre_sis) || String.IsNu
{
ViewData["Error"] = "Debe cubrir todos los datos!!!";
return View();
}
if (consola.FileName.Substring(consola.FileName.Length - 4) != ".exe")
{
ViewData["Error"] = "Debe escoger un fichero ejecutable!!!";
return View();
}
}
}

```

Figura 33: Representación del patrón Controlador en la aplicación.

**Bajo acoplamiento:** Indica la independencia existente entre módulos, es decir, tener las clases lo menos aglomeradas entre sí. Su uso garantiza que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. (35)

La figura 34 expone la presencia del patrón Bajo Acoplamiento, donde se muestra la independencia de los paquetes *Controllers*, *Models* y *Views*. En caso de alguna modificación en alguna de las clases, las demás no se verán afectadas, esto sucede debido a que las clases de las capas inferiores, no son conscientes de ningún detalle de clases superiores.

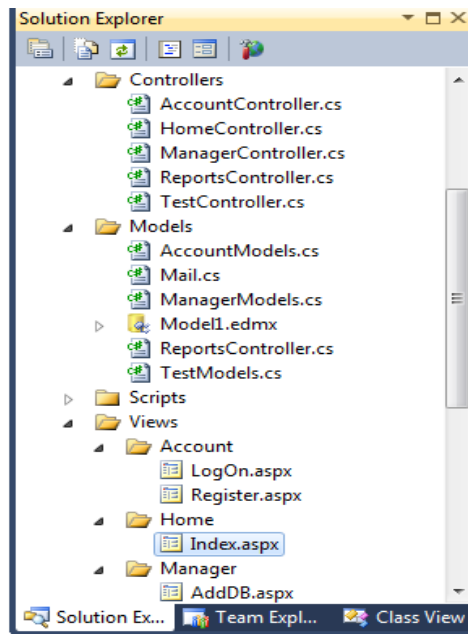


Figura 34: Representación del patrón Bajo acoplamiento en la aplicación.

### 3.8 Diagrama de componentes

Los componentes pertenecen al mundo físico, es decir, representan un bloque de construcción al modelar aspectos físicos de un sistema. Los diagramas de componentes describen los elementos lógicos del sistema y sus relaciones, incluyendo código fuente, binario y ejecutable. Los componentes son la representación de todos los tipos de elementos de *software* que intervienen en la creación de aplicaciones informáticas, estos pueden ser archivos, paquetes, etc. (15)

Descripción de los elementos:

**Model1.edmx:** archivo generado mediante el uso del *Entity Framework*. Un archivo .edmx define un modelo conceptual, un modelo de almacenamiento y la asignación entre estos modelos.

**ASPNETDB.MDF:** base de datos utilizada en la aplicación web.

**PerformanceEvaluation.exe:** aplicación de consola mediante la cual se le realiza las pruebas a los algoritmos cargados por los usuarios de la aplicación *web*.

El resto de los componentes representan clases y bibliotecas que se emplearon en el desarrollo de la aplicación.

A continuación se expone el diagrama de componentes:

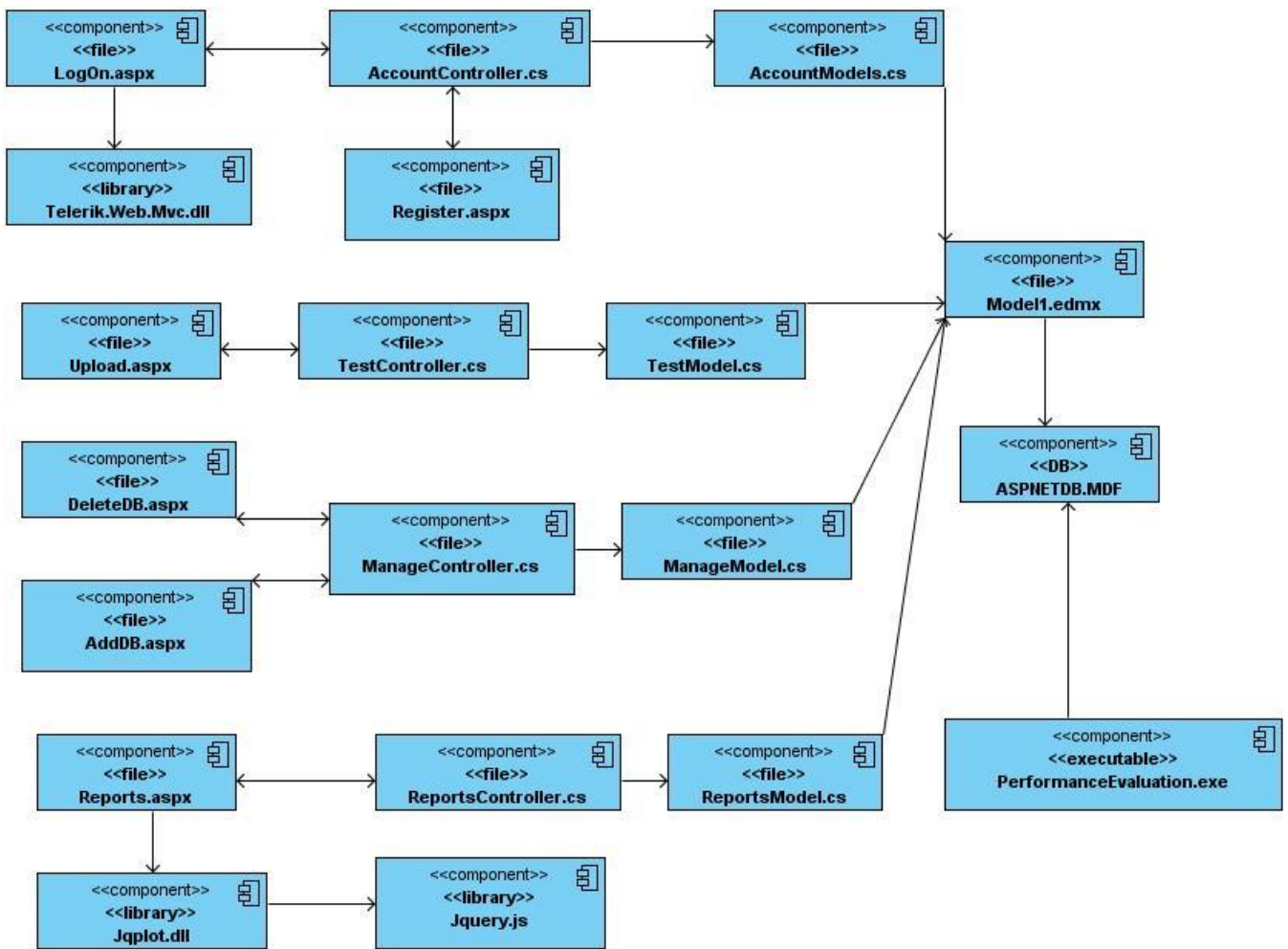


Figura 35: Diagrama de componentes.

### 3.9 Modelo de despliegue

El Modelo de despliegue se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos, visualizando la distribución de los componentes de *software* en los nodos físicos. Los diagramas de despliegue son útiles para facilitar la comunicación entre los ingenieros de *hardware* y los de *software*.

El modelo de despliegue muestra las relaciones físicas entre los componentes *hardware* y *software* en el sistema, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los

componentes *software* (procesos y objetos que se ejecutan en ellos). Estarán formados por instancias de los componentes *software* que representan manifestaciones del código en tiempo de ejecución. (15) El siguiente diagrama muestra un nodo PC Cliente donde se encontrará la computadora del usuario con la aplicación, esta se conectará con un servidor de aplicaciones, consumiendo los servicios de petición al mismo mediante el protocolo HTTPS. En el nodo Servidor de Aplicaciones se encuentran todos los componentes que implementan las funcionalidades del sistema, este a su vez, interactúa con el nodo Servidor de Base de datos, utilizando la conexión mediante el protocolo TCP/IP, como se puede observar en la figura 36.

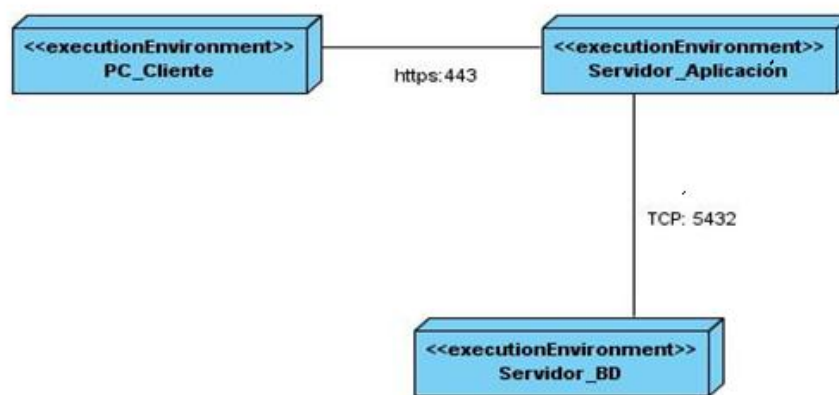


Figura 36: Modelo de despliegue.

### 3.10 Pruebas del sistema

Este flujo de trabajo es el encargado de evaluar la calidad del producto que se está desarrollando, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida. (34)

Sus objetivos son:

- Encontrar y documentar defectos en la calidad del *software*.
- Generalmente asesora la calidad del *software* percibida.
- Proveer las validaciones realizadas en el diseño y especificación de requisitos por medio de demostraciones concretas.
- Verificar las funciones del producto de *software* según lo diseñado.
- Verificar que los requisitos tengan su apropiada implementación. (34)

El desarrollo del flujo de trabajo consistirá en planificar que es lo que hay que probar, diseñar cómo se va a hacer, implementar lo necesario para llevarlos a cabo, ejecutarlos en los niveles necesarios y obtener los resultados, de forma que la información obtenida sirva para ir refinando el producto a desarrollar.

Para cada Caso de uso se establecen pruebas de aceptación que validarán la correcta implementación del caso de uso. Cada prueba es especificada mediante un documento que establece las condiciones de ejecución, las entradas de la prueba, y los resultados esperados. (13)

Las pruebas del sistema son el proceso de analizar una característica de un *software*, para detectar diferencia entre las condiciones existentes y requeridas. Existen varios tipos de pruebas, en dependencia del tipo de técnica que aplican (pruebas de cajas blancas o pruebas de caja negra). La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del *software*, los casos de prueba pretenden demostrar que las funciones del *software* son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. La prueba de la caja blanca del *software* comprueba los caminos lógicos del *software* proponiendo casos de prueba que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coinciden con el esperado o mencionado. (13)

### **3.10.1 Pruebas de caja blanca**

La prueba de la **caja blanca** del *software* comprueba los caminos lógicos del mismo proponiendo casos de prueba que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coinciden con el esperado o mencionado. (14)

Se realizó una prueba de caja blanca a la funcionalidad *RunBiometricSystem* como se muestra en la figura 37, a la misma se le confeccionó el grafo de flujo, la complejidad ciclomática y los caminos independientes, como se pueden observar en la figura 38, tabla 7 y 8.



```

public float RunBiometricSystem(string[] args)
{
    string tempArgs = "";
    ProcessStartInfo processInfo = new ProcessStartInfo();

    processInfo.FileName = fileName; //1
    for (int i = 0; i < args.Length; i++) //2
    {
        tempArgs += " " + args[i]; //3
    }
    processInfo.Arguments = tempArgs;
    processInfo.RedirectStandardOutput = true;
    processInfo.UseShellExecute = false;
    processInfo.CreateNoWindow = true;
    Process process1 = System.Diagnostics.Process.Start(processInfo);

    process1.EnableRaisingEvents = false;
    string value = ""; //4

    return float.Parse(value); //5
}

```

Figura 37: Prueba de caja blanca para la funcionalidad *RunBiometricSystem*.

A continuación se muestra el grafo de flujo para la funcionalidad *RunBiometricSystem*, donde se muestran mediante nodos y aristas todos los caminos posibles que se ejecutan en el código.

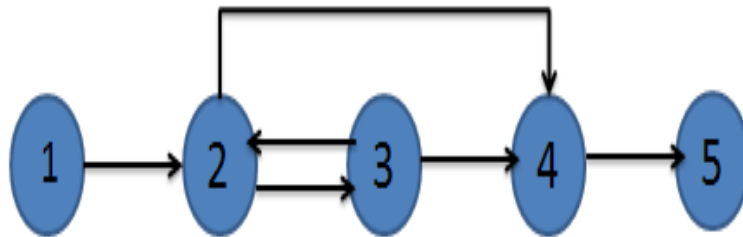


Figura 38: Grafo de flujo para la funcionalidad *RunBiometricSystem*.

Tabla 7: Complejidad ciclomática para la funcionalidad *RunBiometricSystem*.

Fórmula 1	Fórmula 2	Fórmula 3
$V(G) = (A(\text{Aristas}) - N(\text{Nodos})) + 2$	$V(G) = P(\text{Nodos Predicados}) + 1$	$V(G) = R = 3$
$V(G) = (6 - 5) + 2$	$V(G) = 2 + 1 = 3$	
$V(G) = 1 + 2 = 3$		

Tabla 8: Caminos independientes obtenidos y casos de pruebas que obliga a la ejecución de cada camino.

Caminos	Descripción
1-2-3-4-5	Camino cuando todo se ejecuta.
1-2-4-5	Cuando el valor de la variable args.Length sea 0.
1-2-3-2-4-5	Cuando el valor de la variable args.Length sea mayor que 0.

Se realizó una prueba de caja blanca a la funcionalidad *Register* como se muestra en la figura 39, a la misma se le realizó el grafo de flujo, la complejidad ciclomática y los caminos independientes, como se pueden observar en la figura 40, tabla 9 y 10.

```

public ActionResult Register(RegisterModel model)
{
    string user = Convert.ToString(TempData["user"]);
    ViewData["user"] = user;
    TempData["user"] = user; //1
    if (ModelState.IsValid) //2
    {
        MembershipCreateStatus createStatus = MembershipService.CreateUser
            (model.UserName, model.Password, model.Email);//3

        if (createStatus == MembershipCreateStatus.Success) //4
        {
            FormsService.SignIn(model.UserName, false /* createPersistentCookie */);
            TempData["user"] = model.UserName; //5
            return RedirectToAction("Index", "Home");//6
        }
        else //7
        {
            ModelState.AddModelError("", AccountValidation.ErrorCodeToString(createStatus));//8
        }
    }

    ViewData["PasswordLength"] = MembershipService.MinPasswordLength;//9
    return View(model);//10
}

```

Figura 39: Prueba de caja blanca para la funcionalidad *Register*.

La figura que se muestra a continuación representa el grafo de flujo para la funcionalidad *Register*, en el mismo se muestran en nodos y aristas los caminos posibles de la ejecución del método.

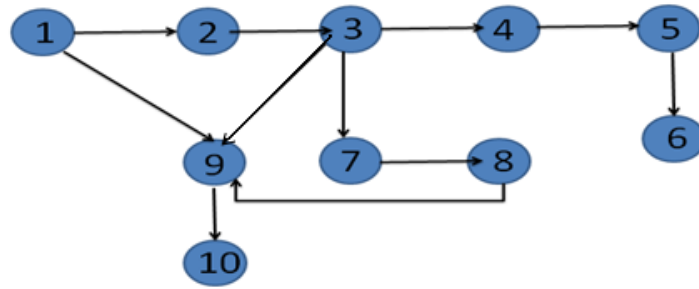


Figura 40: Grafo de flujo para la funcionalidad *Register*.

Tabla 9: Complejidad ciclomática para la funcionalidad *Register*.

Fórmula 1	Fórmula 2	Fórmula 3
$V(G)=(A(\text{Aristas})- N(\text{Nodos})) +2$	$V(G)=P(\text{Nodos Predicados})+1$	$V(G)= R= 3$
$V (G) = 11-10+2 = 3$	$V(G) = 2+1 =3$	

Tabla 10: Caminos independientes obtenidos y casos de pruebas que obliga a la ejecución de cada camino.

Camino	Descripción
1-2-3-4-5-6	Camino en el primer ciclo cuando <i>createStatus</i> sea igual <i>MembershipCreateStatus.Success</i> .
1-2-3-7-8-9-10	Camino en el primer ciclo cuando <i>createStatus</i> sea distinto <i>MembershipCreateStatus.Success</i> .
1-9-10	Cuando <i>ModelState.IsValid</i> devuelve falso.

Se realizó una prueba de caja blanca a la funcionalidad *LoadFile* como se muestra en la figura 41, a la misma se le realizó el grafo de flujo, la complejidad ciclomática y los caminos independientes, como se pueden observar en la figura 42, tabla 11 y 12.

```

public List<float> LoadFile(string filePath)
{
    tmp = new List<float>();

    TextReader reader = new StreamReader(filePath);
    string var; //1

    while ((var = reader.ReadLine()) != null) //2
    {
        string[] tmpSplitter = var.Split(); //3
        if (tmpSplitter[0].Contains(".")) //4
            tmpSplitter[0].Replace('.', ','); //5
        tmp.Add(float.Parse(tmpSplitter[0])); //6
    }

    reader.Close();//7

    return tmp;//8
}

```

Figura 41: Prueba de caja blanca para la funcionalidad *LoadFile*.

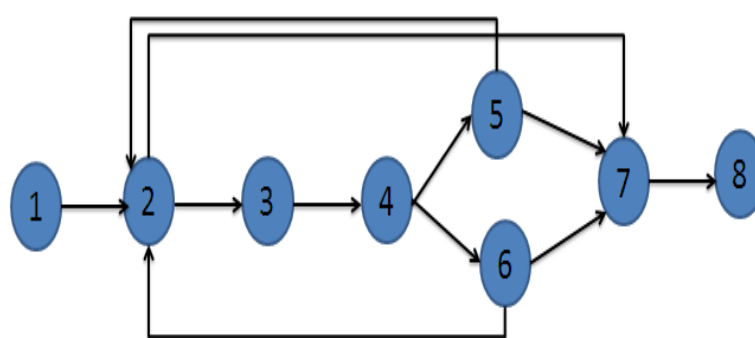


Figura 42: Grafo de flujo para la funcionalidad *LoadFile*.

Tabla 11: Complejidad ciclomática para la funcionalidad *LoadFile*.

Fórmula 1	Fórmula 2	Fórmula 3
$V(G) = (A(\text{Aristas}) - N(\text{Nodos})) + 2$	$V(G) = P(\text{Nodos Predicados}) + 1$	$V(G) = R = 5$
$V(G) = (11 - 8) + 2$	$V(G) = 4 + 1 = 5$	
$V(G) = 3 + 2 = 5$		

Tabla 12: Caminos independientes obtenidos y casos de pruebas que obliga a la ejecución de cada camino.

Camino	Descripción
1-2-3-4-5-2-7-8 1-2-3-4-6-2-7-8	Camino cuando se ejecuta todo el código.
1-2-7-8	Que el valor de la variable var sea igual a null.
1-2-3-4-5-2-7-8	Que el valor de la variable var sea distinto de null y tmpSplitter contenga un punto y coma.
1-2-3-4-6-2-7-8	Que el valor de la variable var sea distinto de null y tmpSplitter no contenga un punto y coma.

Los resultados obtenidos de cada prueba en la última iteración fueron satisfactorios lo que demuestra la correcta implementación de dichas funcionalidades. A continuación se muestra un gráfico que representa los resultados de las pruebas unitarias realizadas en las cuatro iteraciones de pruebas efectuadas al sistema, el cual muestra como en la primera iteración se le realizaron pruebas unitarias a 4 funcionalidades críticas de la aplicación, devolviendo 3 de ellas los resultados esperados, mientras que ya en la última iteración se le realizaron la prueba a 6 funcionalidades, devolviendo todas, los resultados esperados.

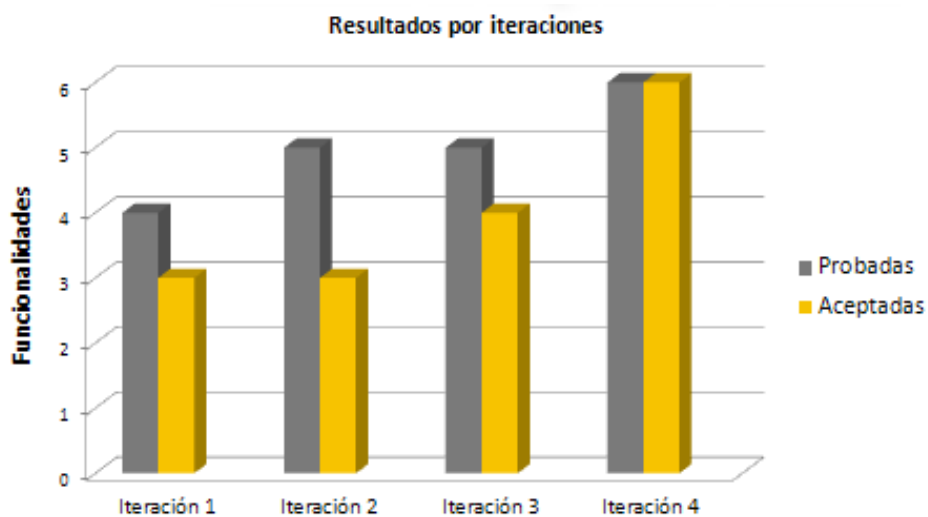


Figura 43: Gráfico de resultados de las 4 iteraciones de pruebas unitarias realizadas hasta el momento.

### 3.10.2 Pruebas de caja negra

La prueba de **caja negra** se refiere a las pruebas que se llevan a cabo sobre la interfaz del *software*. Los casos de prueba pretenden demostrar que las funciones del *software* son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. (14)

El primer caso de prueba se refiere al caso de uso Registrar usuario, en el mismo se describe el proceso de insertar un nuevo usuario en el sistema, donde apoyado de las variables y la descripción de las mismas se facilita este proceso, mostrando así la respuesta del sistema y el flujo central que le corresponde, como se observa en la tabla 13 y 14.

Tabla 13: Prueba de caja negra para el caso de uso Registrar usuario.

Escenarios	Descripción	Variable	Variable	Respuesta del Sistema	Flujo Central
EC 1 Registrar usuario.	Mediante este escenario se inserta en el sistema un nuevo usuario	Usuario	Correo electrónico	El sistema guarda el nuevo usuario y permite el acceso.	1. El usuario selecciona la opción Registrar. 2. Se muestra la interfaz para registrar usuario. 3. Se introducen los datos. 4. Se selecciona la opción registrarme y se guardan los datos.
		Confirmar contraseña	Contraseña		

Tabla 14: Descripción de las variables para el caso de prueba Registrar usuario.

Nombre de campo	Clasificación	Valor Nulo	Descripción
Usuario	Campo de texto	No	Se aceptan letras minúsculas y mayúsculas.
Contraseña	Campo de texto	No	Letras, números y caracteres especiales.
Confirmar contraseña	Campo de texto	No	Letras, números y caracteres especiales
Correo electrónico	Campo de texto	No	Se aceptan letras minúsculas y mayúsculas, el carácter @ es obligatorio luego el dominio al que pertenece el nombre del usuario del correo electrónico.

A continuación se muestra la prueba de caja negra para el caso de uso Autenticar usuario, se describe el proceso de acceso a la aplicación. Se realiza la descripción de las variables y de las mismas se muestra la respuesta del sistema y el flujo central que le corresponde, como se observa en la tabla 15 y 16.

Tabla 15: Prueba de caja negra para el caso de uso Realizar autenticación.

Escenarios	Descripción	Variable	Respuesta del sistema	Flujo central
EC 2 Realizar autenticación.	Mediante este escenario se accede al sistema, y el usuario debe estar registrado.	Usuario	El sistema verifica el usuario y contraseña y si están correctos posibilita el acceso.	<ol style="list-style-type: none"> <li>1. El usuario selecciona cualquier opción del módulo derecho de la aplicación.</li> <li>2. Se muestra la interfaz para realizar la autenticación.</li> <li>3. Se introducen los datos.</li> <li>4. Se selecciona la opción ingresar y se verifican los datos.</li> </ol>
		Contraseña		

				Si los datos son los correctos se posibilita el acceso.
--	--	--	--	---

Tabla 16: Descripción de las variables para el caso de prueba Realizar autenticación.

Nombre de campo	Clasificación	Valor nulo	Descripción
Usuario	Campo de texto	No	Se aceptan letras minúsculas y mayúsculas.
Contraseña	Campo de texto	No	Letras, números y caracteres especiales.

A continuación se realiza la descripción de la prueba de caja negra para en caso de uso Realizar prueba, donde se realiza una breve descripción de este proceso y se muestran las variables relacionadas con el mismo, posibilitándole una respuesta del sistema y un flujo central del proceso que es descrito con detalle, como se puede observar en la tabla 17 y 18.

Tabla 17: Prueba de caja negra para el caso de uso Realizar prueba.

Escenarios	Descripción	Variable	Variable	Respuesta del sistema	Flujo central
EC 3 Realizar prueba.	Mediante este escenario se carga el algoritmo al que se le va a realizar la prueba.	Algoritmo (huella dactilar).	Base de datos.	El sistema almacena los datos en una base de datos.	<ol style="list-style-type: none"> <li>1. El usuario selecciona la opción "Subir algoritmo".</li> <li>2. Se muestra la interfaz con los datos que el usuario debe introducir.</li> <li>3. Se selecciona la opción "Registrarme" y se guardan los</li> </ol>
		Versión.	Cargar aplicación (.exe)		
		Nombre del sistema.			



					<p>datos.</p> <p>4. El sistema obtiene los scores generados en la aplicación.</p> <p>5. Se obtienen los valores de FAR, FRR, EER.</p> <p>6. Se grafican los resultados o valores anteriores.</p> <p>7. Se notifica al usuario mediante un correo electrónico que su prueba se realizó.</p>
--	--	--	--	--	--

Tabla 18: Descripción de las variables para el caso de prueba Realizar prueba.

Nombre de campo	Clasificación	Valor nulo	Descripción
Algoritmo	Seleccionar	No	Se selecciona el tipo de algoritmo, huella dactilar o firmas manuscritas.
Base de datos	Seleccionar	No	Se selecciona la base de datos a utilizar.
Nombre del sistema	Campo de texto	No	Letras, números y caracteres especiales
Versión	Campo de texto	No	Se aceptan letras, números y caracteres especiales.
Aplicación	Botón	No	Se acepta la dirección ( <i>url</i> ) de donde se encuentra el ejecutable (.exe).

A continuación se realiza la prueba de caja negra para el caso de uso Realizar reporte de pruebas que pertenece al escenario 5. Se realiza una breve descripción del mismo, donde se muestran las variables

asociadas y, la respuesta del sistema, además se describe el proceso detalladamente como un flujo central, como se puede observar en la tabla 19 y 20.

Tabla 19: Prueba de caja negra para el caso de uso Realizar reporte de pruebas.

Escenarios	Descripción	Variable	Respuesta del sistema	Flujo central
EC 5 Realizar reporte de la prueba ejecutada.	Mediante este escenario se le informan al usuario los resultados de la prueba realizada.	Seleccionar opción ver. Seleccionar opción más.	El sistema muestra todos los resultados de la prueba.	<ol style="list-style-type: none"> <li>1. El usuario selecciona la opción ver.</li> <li>2. Se muestra una interfaz con los resultados de las pruebas realizadas por los usuarios.</li> <li>3. El usuario selecciona la opción más.</li> <li>4. Se muestran las gráficas de FAR, FRR, ERR y los resultados finales (ERR, FMR 100, FMR 1000, Zero FMR, Zero FNMR).</li> </ol>

Tabla 20: Descripción de las variables para el caso de prueba Realizar reporte de pruebas.

Nombre de campo	Clasificación	Valor nulo	Descripción
Ver	hipervínculo	Sí	Se acepta un clic.
Más	hipervínculo	Sí	Se acepta un clic.

Como resultado de las diferentes pruebas de validación de requisitos realizadas, se detectaron un conjunto de no conformidades que se han ido resolviendo gradualmente. A continuación se muestran una figura que representa los resultados de las pruebas de validación que se han realizado en cada una de las iteraciones hasta el momento. Importante destacar que en la última iteración oficial de pruebas, como muestra la figura, fueron detectadas 4 no conformidades, las cuales fueron mitigadas,

dejando hasta el momento el sistema libre de no conformidades y listo para una nueva iteración de pruebas.

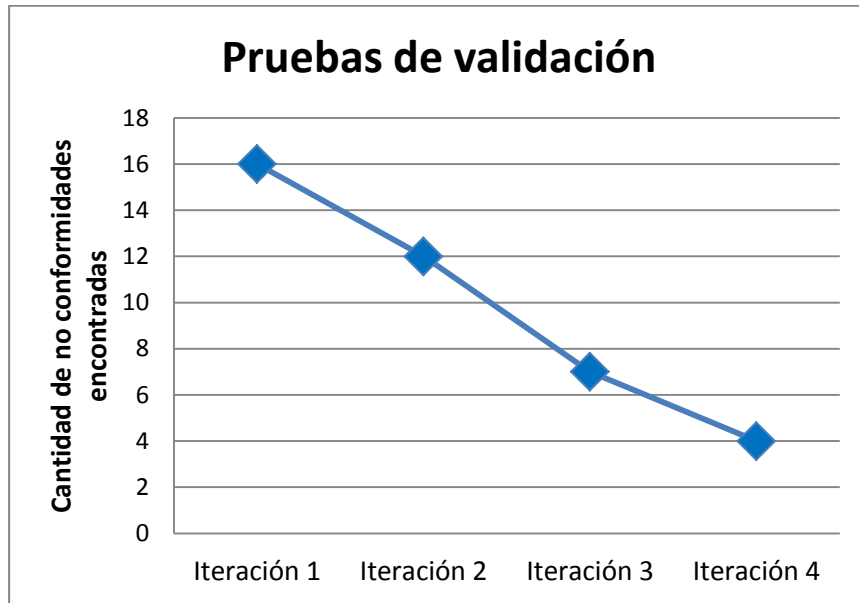


Figura 44: Resultados de las 4 iteraciones de pruebas de validación realizadas hasta el momento.

### 3.11 Conclusiones

En la realización del capítulo “Elaboración y construcción del sistema”, se explica el diseño de la solución. Se elaboraron los diagramas de clases del diseño y de secuencia, lo que ha permitido obtener una visión del sistema, y mostrar el flujo de operación presente durante cada acción solicitada.

Se muestra el modelo físico de la base de datos el cual se utiliza para almacenar la información del sistema, el patrón arquitectónico presente en la aplicación y los patrones de diseño usados durante el desarrollo del sistema. Se realizó el diagrama componentes y un modelo de despliegue donde se muestran las relaciones físicas entre los componentes *hardware* y *software* del sistema. Luego de realizar las pruebas al sistema, se aseguró la ejecución correcta de la solución en todo el período de implementación a partir de cada una de las iteraciones realizadas.

## Conclusiones Generales

Del análisis de los resultados se concluye que:

- Al realizar un estudio de las interfaces de plataformas de pruebas, profundizar en la historia, evolución y tendencias actuales de las aplicaciones web permitió elegir las herramientas y tecnologías necesarias para guiar el proceso de desarrollo.
- El análisis y diseño de la solución dio paso al entendimiento de los procesos asociados al sistema, a partir de las descripciones de los casos de uso y los modelos de clases.
- La plataforma *web* posee una interfaz amigable y de fácil uso para los usuarios. Permite la realización de las pruebas a los algoritmos biométricos y de esta forma mide la efectividad de los mismos mostrando sus resultados en línea.
- El objetivo planteado en el diseño de la investigación fue cumplido, dando como resultado la implementación de la aplicación “Solución *Web* para Pruebas de Algoritmos de Identificación Biométrica sobre huellas dactilares.”
- Las pruebas del sistema posibilitaron evaluar que las funcionalidades de la aplicación fueron desarrolladas correctamente, detectando los errores para realizar el despliegue del sistema.

## Recomendaciones

Se recomienda:

- Agregar otras funcionalidades que permitan realizar pruebas a otros algoritmos biométricos, que no sean sólo de huella dactilar; como por ejemplo firma manuscrita.
- Realizar un monitoreo del rendimiento del sistema para verificar el límite donde colapsa la solución propuesta.
- Aumentar la seguridad en el sistema mediante el uso de certificados digitales.
- La utilización del manual de usuarios, para el correcto funcionamiento y administración de la aplicación.

## Glosario de términos

**Algoritmo:** secuencia limitada de instrucciones o pasos que indica a un sistema computarizado cómo resolver un problema en especial. Un sistema biométrico utiliza múltiples algoritmos; por ejemplo, para el procesamiento de imágenes, la generación de plantillas, comparaciones, etcétera. (1)

**Comparación:** proceso de comparación de una referencia biométrica almacenada con anterioridad, para tomar una decisión sobre identificación o verificación. (1)

**Extracción:** proceso de conversión de una muestra biométrica capturada en datos biométricos para que puedan ser comparados con una referencia. (1)

**Plantilla biométrica:** es el modelo binario del patrón biométrico del sujeto, es decir, los equipos "no almacenan" huellas como tal, sino después de un enrolamiento y gracias al Algoritmo, la muestra tomada es transformada en un número binario que es lo que realmente se almacena en los lectores biométricos. (1)

**Precisión:** término general utilizado para describir cuál es el rendimiento de un sistema biométrico. La estadística real que determina dicho rendimiento varía según la tarea a realizar (verificación e identificación). (1)

**Sistema biométrico:** componentes individuales múltiples (tales como sensor, algoritmo de coincidencia y visualización de resultado) que se combinan para crear un sistema totalmente funcional.

**Tasa de error igual (EER):** algunas veces se llama tasa de error cruzada (CER). Es una estadística que muestra la actuación del sistema biométrico, típicamente cuando opera en la tarea de verificación.

**Tasa de falsa coincidencia (FMR):** es la probabilidad de que un sistema biométrico identifique incorrectamente un individuo o que falle para rechazar un impostor. (1)

**Tasa de falsa no-coincidencia (FNMR):** es parecida a la Tasa de Falso Rechazo (FRR), con la diferencia de que esta incluye la tasa de falla para capturar el error. (1)

**Tasa de falsos aceptados (FAR):** es la probabilidad de que un individuo no autorizado sea autenticado. (1)

**Tasa de falsos rechazados (FRR):** es la probabilidad de que un individuo autorizado sea inapropiadamente rechazado. (1)

## Referencias bibliográficas

1. **Dorizzi, Dijana Petrovska-Delacrétaz. Gérard Chollet Bernadette.** Guide to Biometric Reference Systems. Michigan, State : s.n., 2009.
2. **Janices, Pedro.** [Online] 2011. <http://www.biometria.gov.ar>.
3. **Elmadany Younis, Hamed.** *Performance Evaluation of Bbiometric System.* 2007.
4. **Real Academia Española.** Diccionario de la lengua española. [Online] 2011. <http://www.rae.es/rae.html>.
5. **Muhammad Younas, Javed and Umer, Munir.** *Fingerprint Matching using Gabor Filters.* 2007.
6. **Biometrics.** [Online] 2011. <http://www.biometria.gov.ar/acerca-de-la-biometria/glosario/a-e.aspx>.
7. **Bolle, Ruud and Pankanti, Sharath.** *Introduction to biometrics.* Michigan State University East Lansing : s.n., 2009.
8. **Herve, Brendin, Aversano, Guido and Chollet, Gerard.** *Verification Reference System In 2nd Workshop on Multimodal User .* Francia : s.n., 2006.
9. **Responsible for the Biometrics FAQ's.** Bioidentification. [Online] 1999-2000. <http://www.bromba.com/faq/biofaq.htm>.
10. **BIOAPI Consortium.** BIOAPI. [Online] 1998-2012. <http://www.bioapi.org>.
11. **American National Standards Institute.** ANSI. [Online] 2001. <http://www.ansi.org/>.
12. **Universidad de Bolonia, Italia.** FVC on-Going. [Online] junio 22, 2009. <https://biolab.csr.unibo.it/fvcongoing/UI/Form/Home.aspx..>
13. **Pressman, Roger S.** *Software Engineering.* New York : Previous editions, 2005.
14. **Sommerville, Ian.** *Ingeniería del Software. Séptima edición.* Madrid : s.n., 2005.
15. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : s.n., 2006.
16. **Visual paradigm company.** [Online] 2005. <http://www.visual-paradigm.com/>.
17. **Ferguson, Jeff, Patterson, Brian and Beres, Jason.** *La biblia de C#.* Madrid : s.n., 2003.
18. **Gousset, Mickey; Keller, Brian; Krishnamoorthy, Ajoy.** *Professional Application Lifecycle Management with Visual Studio 2010.* 2010.
19. **Espósito, Dino.** *Programing Microsoft ASP.Net MVC y MVC2 con Visual Studio 2010.* s.l. : Microsoft Press, 2010.

20. **Wesley Longman, Addison.** *Introducción a los sistemas de base de datos.* México : s.n., 2001.
21. **Biometría Aplicada México.** Soluciones biométricas de huella digital. [Online] 2012.  
<http://www.biometriaaplicada.com/huella.html>.
22. **Berzal Galiano, Fernando.** *Acceso a Base de Datos con ADO.NET.* 2007.
23. **Hernández Leal, Octavio.** *C# 3.0 y LinQ.* s.l. : Editorial Krasis Press, 2007.
24. **Zorrilla Castro, Unai.** *ADO.NET Entity Framework.* 2008.
25. **Artisteer Standard Edition.** Artisteer Web design revolution. 2001-2008. [Online] <http://www.artisteer.com>.
26. **Developer Productivity, Team Productivity and Automated Testing.** Telerik. [Online] 2002-2012.  
<http://www.telerik.com>.
27. **Crockford, Douglas.** *JavaScript: The Good Parts.* 2008.
28. **Corporation Jqplot.** JqPlot. [Online] junio 4, 2009. <http://www.jqplot.com/>.
29. **Murphey, Rebecca and Padolsey, James .** *Fundamentos de jQuery.* s.l. : Creative Commons, 2011.
30. **Creativaint Corporation.** Estilo Flash. [Online] novienmbre 15, 2008. <http://www.estiloflash.com>.
31. **Adobe Digital Marketing Suite.** Adobe Fireworks CS6. [Online]  
<http://www.adobe.com/es/products/fireworks.html>.
32. **Schmitt, Christopher.** *CSS Hojas de Estilo en Cascada para el Diseño Web.* 2005.
33. **Duckett, Jon.** *HTML and CSS: Design and Build Websites.* 2011.
34. **Rational Software Corporation.** *Rational Unified Process (RUP).* 2002.
35. **Pressman, R.** *Ingeniería del Software: Un enfoque práctico.* Madrid : McGraw Hill, 2001.
36. **Dorizzi, Dijana Petrovska-Delacrétaz. Gérard Chollet Bernadette.** Guide to Biometric Reference Systems.  
*Guide to Biometric Reference Systems.* Michigan,State : s.n., 2009.
37. **Pérez, Pedro Carlos.** *El diseño metodológico de la investigación científica.*



## Bibliografía

1. **Dorizzi, Dijana Petrovska-Delacrétaz. Gérard Chollet Bernadette.** Guide to Biometric Reference Systems. Michigan,State : s.n., 2009.
2. **Elmadany Younis, Hamed.** *Performance Evaluation of Bbiometric System.* 2007.
3. **Muhammad Younas, Javed y Umer, Munir.** *Fingerprint Matching using Gabor Filters.* 2007.
4. **Bolle, Ruud y Pankanti, Sharath.** *Introduction to biometrics.* Michigan State University East Lansing : s.n., 2009.
5. **Herve, Brendin, Aversano, Guido y Chollet, Gerard.** *Verification Reference System In 2nd Workshop on Multimodal User .* Francia : s.n., 2006.
6. **Pressman, Roger S.** *Software Engineering.* New York : Previous editions, 2005.
7. **Sommerville, Ian.** *Ingeniería del Software. Séptima edición.* Madrid : s.n., 2005.
8. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : s.n., 2006.
9. **Ferguson, Jeff, Patterson, Brian y Beres, Jason.** *La biblia de C#.* Madrid : s.n., 2003.
10. **Gousset, Mickey; Keller, Brian; Krishnamoorthy, Ajoy.** *Professional Application Lifecycle Management with Visual Studio 2010.* 2010.
11. **Espósito, Dino.** *Programing Microsoft ASP.Net MVC y MVC2 con Visual Studio 2010.* s.l. : Microsoft Press, 2010.
12. **Wesley Longman, Addison.** *Introducción a los sistemas de base de datos.* México : s.n., 2001.
13. **Berzal Galiano, Fernando.** *Acceso a Base de Datos con ADO.NET.* 2007.
14. **Hernández Leal, Octavio.** *C# 3.0 y LinQ.* s.l. : Editorial Krasis Press, 2007.
15. **Zorrilla Castro, Unai.** *ADO.NET Entity Framework.* 2008.
16. **Crockford, Douglas.** *JavaScript: The Good Parts.* 2008.
17. **Corporation Jqplot.** JqPlot. [En línea] 4 de junio de 2009. <http://www.jqplot.com/>.
18. **Murphey, Rebecca y Padolsey, James .** *Fundamentos de jQuery.* s.l. : Creative Commons, 2011.
19. **Creativaint Corporation.** Estilo Flash. [En línea] 15 de noviembre de 2008. <http://www.estiloflash.com>.
20. **Schmitt, Christopher.** *CSS Hojas de Estilo en Cascada para el Diseño Web.* 2005.

21. **Duckett, Jon.** *HTML and CSS: Design and Build Websites.* 2011.
22. **Rational Software Corporation.** *Rational Unified Process (RUP).* 2002.
23. **Pressman, R.** *Ingeniería del Software: Un enfoque práctico.* Madrid : McGraw Hill, 2001.
24. **Dorizzi, Dijana Petrovska-Delacrétaz. Gérard Chollet Bernadette.** *Guide to Biometric Reference Systems. Guide to Biometric Reference Systems.* Michigan,State : s.n., 2009.
25. **Pérez, Pedro Carlos.** *El diseño metodológico de la investigación científica.*

## Anexos

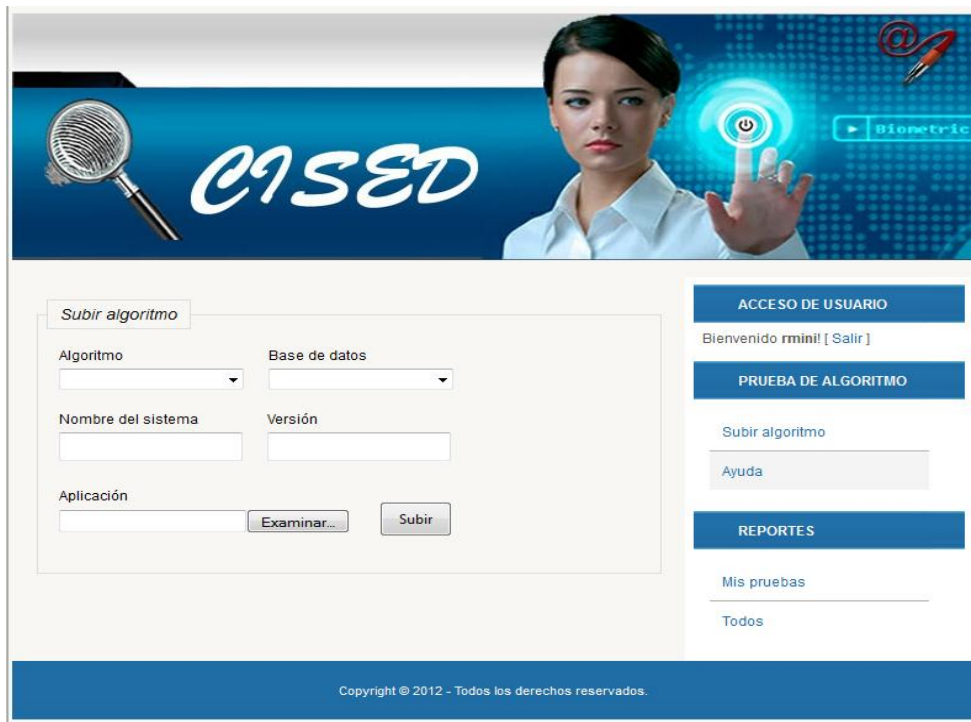
Anexo 1: Prototipo de interfaz de usuario para el caso de uso Registrar usuario.

The image shows a user registration interface for a system named 'e95ED'. The header features a woman's face and a hand interacting with a biometric scanner, with the text 'Bionic' and 'e95ED' visible. The main content area is divided into two sections. On the left, under the heading 'Información de la cuenta', there are two input fields for 'Usuario' and 'Correo', and a 'Registrarme' button. On the right, there are two main sections: 'PRUEBA DE ALGORITMO' with a 'Subir algoritmo' button and an 'Ayuda' link, and 'REPORTES' with 'Mis pruebas' and 'Todos' links. The footer contains the text 'Copyright © 2012 - Todos los derechos reservados.'

Anexo 2: Prototipo de interfaz de usuario para el caso de uso Realizar autenticación.

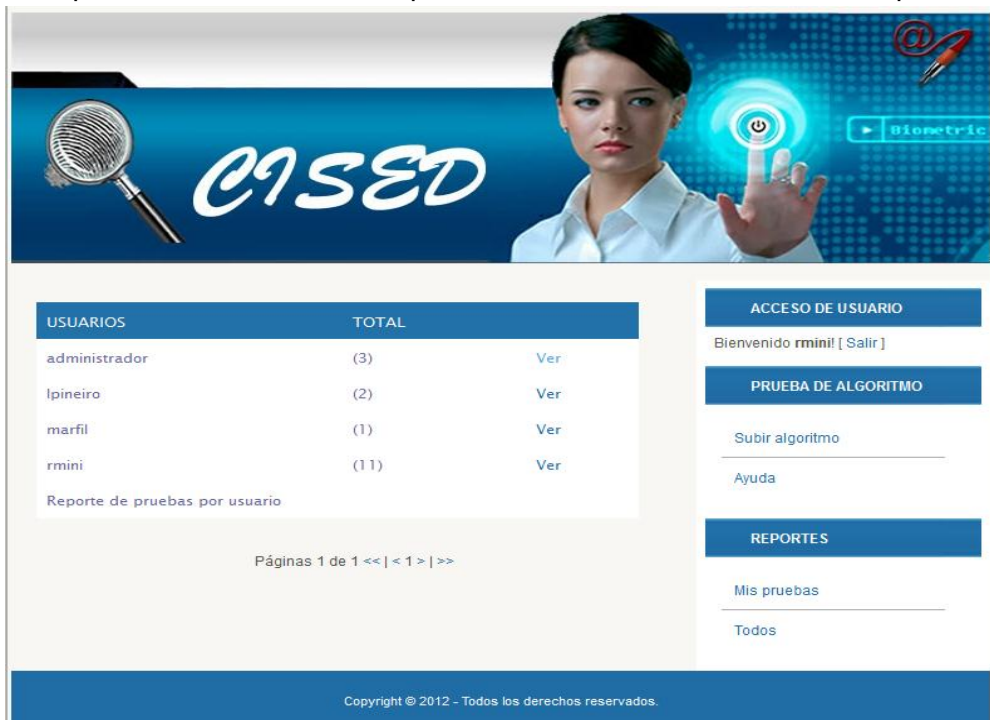
The image shows a user authentication interface for the same system 'e95ED'. The background is identical to the registration screen. A modal window titled 'Autenticarse' is overlaid in the center. It contains two input fields for 'Usuario:' and 'Contraseña:', a 'Registrarme' link, and an 'Ingresar' button. The background interface is dimmed, showing the 'PRUEBA DE ALGORITMO' and 'REPORTES' sections. The footer contains the text 'Copyright © 2012 - Todos los derechos reservados.'

**Anexo 3:** Prototipo de interfaz de usuario para el caso de uso Realizar prueba.



**Anexo 4:** Prototipo de interfaz de usuario para el caso de uso Realizar reporte de pruebas.

**Anexo 4.1:** Prototipo de interfaz en el caso que el usuario acceda ver todos los reportes.



**Anexo 4.2:** Prototipo de interfaz en el caso que el usuario acceda ver los reportes del usuario seleccionado.

The screenshot shows the CASSED user interface. At the top, there is a header with a magnifying glass icon, the logo 'CASSED', and a woman's face with a hand gesture over a biometric sensor. Below the header, there is a table with the following data:

CATEGORIA	BD	NOMBRE	VERSION	ERR	FMR1000	
Huella	innovatrics	joil	3.0	0,57	1,14	Más
Huella	innovatrics	looa	2.0	0,57	1,14	Más

Below the table, it says 'Nombre de usuario: lpineiro'. There are navigation links for 'Páginas 1 de 1 << | < 1 > | >>'. On the right side, there are navigation buttons for 'ACCESO DE USUARIO', 'PRUEBA DE ALGORITMO', and 'REPORTES'. The 'ACCESO DE USUARIO' section shows 'Bienvenido rmini! [ Salir ]'. The 'PRUEBA DE ALGORITMO' section has 'Subir algoritmo' and 'Ayuda' links. The 'REPORTES' section has 'Mis pruebas' and 'Todos' links. At the bottom, there is a copyright notice: 'Copyright © 2012 - Todos los derechos reservados.'

**Anexo 4.3:** Prototipo de interfaz en el caso que el usuario acceda las gráficas generadas (FAR, FRR, FAR vs FRR, ROC y los Resultados).

The screenshot shows the CASSED user interface with a graph. The header is the same as in the previous screenshot. Below the header, there is a navigation bar with tabs for 'FAR', 'FRR', 'FAR vs FRR', 'ROC', and 'Resultados'. The 'FAR' tab is selected, and the graph shows FAR on the y-axis (0 to 100) and 'Umbral' on the x-axis (0 to 100). The graph shows a curve that starts at (0, 100) and drops sharply to near 0 by a threshold of 10. On the right side, there are navigation buttons for 'ACCESO DE USUARIO', 'PRUEBA DE ALGORITMO', and 'REPORTES'. The 'ACCESO DE USUARIO' section shows 'Bienvenido rmini! [ Salir ]'. The 'PRUEBA DE ALGORITMO' section has 'Subir algoritmo' and 'Ayuda' links. The 'REPORTES' section has 'Mis pruebas' and 'Todos' links. At the bottom, there is a copyright notice: 'Copyright © 2012 - Todos los derechos reservados.'



**ACCESO DE USUARIO**  
 Bienvenido rmini! [ Salir ]

**PRUEBA DE ALGORITMO**  
 Subir algoritmo  
 Ayuda

**REPORTES**  
 Mis pruebas  
 Todos

FAR   FRR   FAR vs FRR   ROC   Resultados



Umbral

Copyright © 2012 - Todos los derechos reservados.



**ACCESO DE USUARIO**  
 Bienvenido rmini! [ Salir ]

**PRUEBA DE ALGORITMO**  
 Subir algoritmo  
 Ayuda

**REPORTES**  
 Mis pruebas  
 Todos

FAR   FRR   FAR vs FRR   ROC   Resultados



Umbral

Copyright © 2012 - Todos los derechos reservados.



**eSSED**

Biometric

ACCESO DE USUARIO  
Bienvenido rmini! [ Salir ]

PRUEBA DE ALGORITMO  
Subir algoritmo  
Ayuda

REPORTES  
Mis pruebas  
Todos

FAR FRR FAR vs FRR ROC Resultados

Copyright © 2012 - Todos los derechos reservados.

**eSSED**

Biometric

ACCESO DE USUARIO  
Bienvenido rmini! [ Salir ]

PRUEBA DE ALGORITMO  
Subir algoritmo  
Ayuda

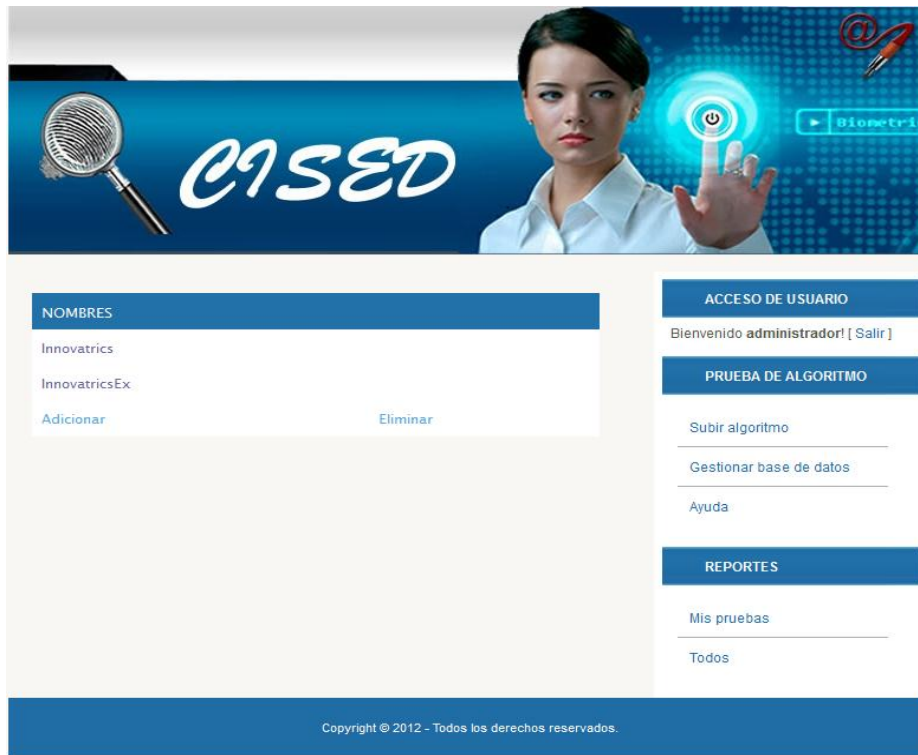
REPORTES  
Mis pruebas  
Todos

FAR FRR FAR vs FRR ROC Resultados

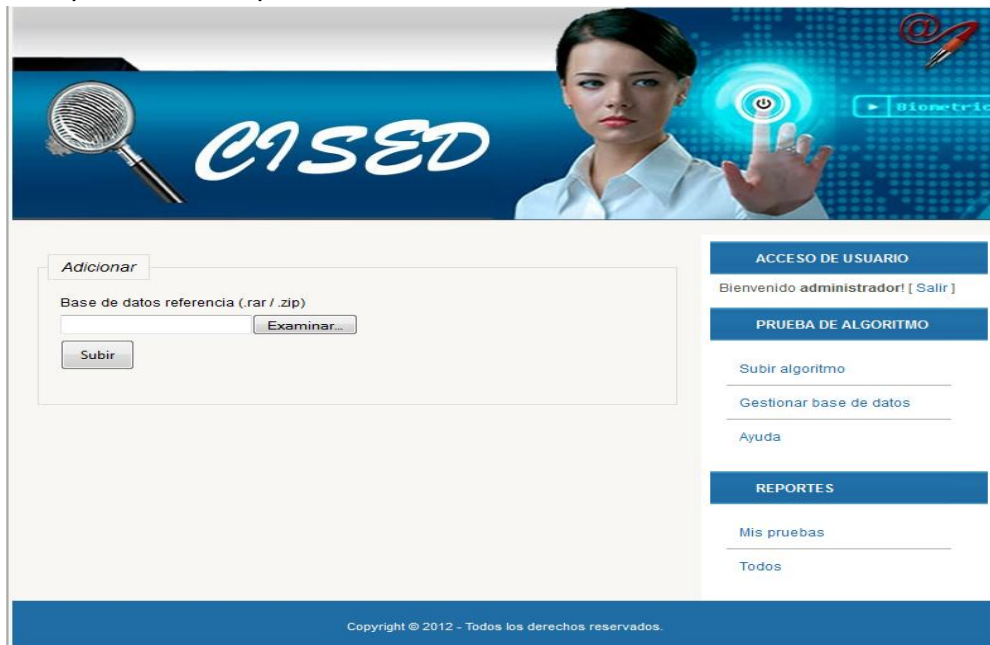
ERR	FMR100	FMR1000	FMR10000	ZeroFMR	ZeroFNMR
3,28	4,55	9,39	9,39	9,39	100

Copyright © 2012 - Todos los derechos reservados.

**Anexo 5:** Prototipo de interfaz para el caso de uso Gestionar base de datos.

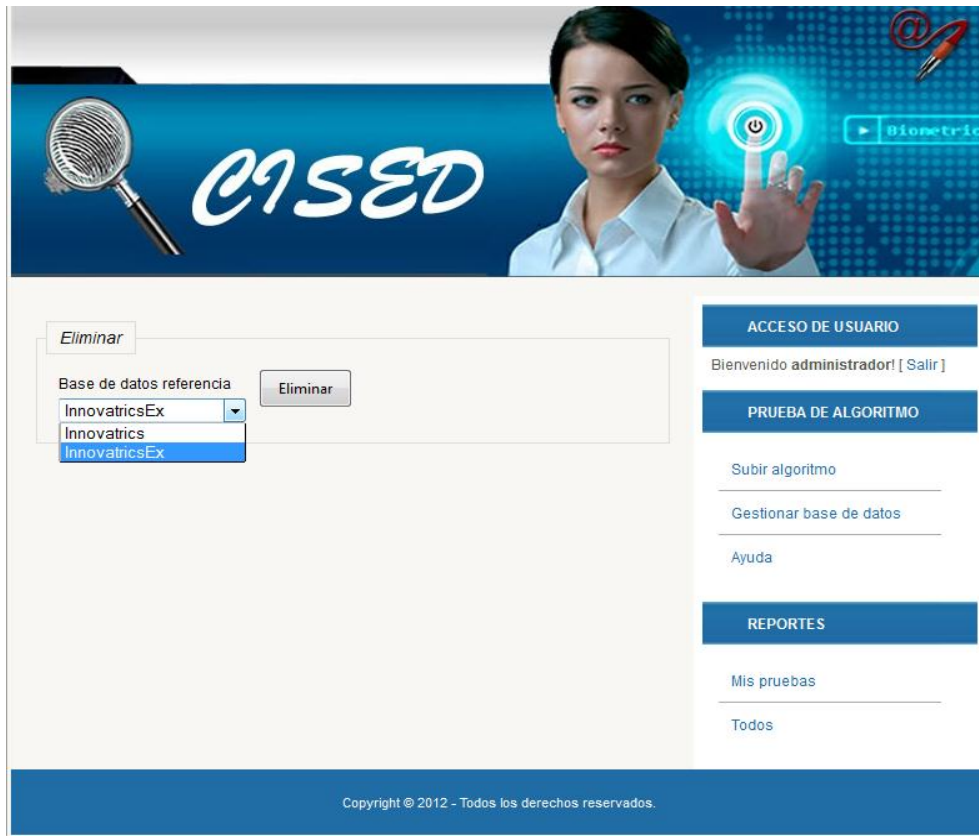


**Anexo 5.1:** Prototipo de interfaz para el caso de uso Gestionar base de datos sección adicional.





Anexo 5.2: Prototipo de interfaz para el caso de uso Gestionar base de datos sección eliminar.



## Anexo 6: Modelo de datos.

