UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Título: Plataforma de pruebas para algoritmos biométricos.

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

Autores: Giraldo David Uffo Lima.

Carlos Achang Rivas.

Tutores: Ing. Yainier Labrada Nueva.

Ing. Yisel Avila Portales.

Consultante: Miguel Ángel Hernández de la Rosa.

Curso 2010-2011

Ciudad de la Habana

Año 53 de la Revolución.

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 _____.

Giraldo David Uffo Lima

_____

Carlos Achang Rivas

_____

Ing. Yanier Labrada Nueva

_____

Ing. Yisel Avila Portales

_____

Datos de Contacto

Ing. Yisel Avila Portales. Graduada de Ingeniero en Ciencias Informáticas en la Universidad de Ciencias Informáticas en el año 2009. Profesora del Departamento de Ciencias Básicas en la Facultad # 1 y adjunto a la producción en el Centro de Identificación y Seguridad Digital (CISED), Departamento de Biometría.

Correo electrónico: yaportales@uci.cu

Ing. Yainier Labrada Nueva. Graduado de Ingeniero en Ciencias Informáticas en la Universidad de Ciencias Informáticas en el año 2008. Profesor del Departamento de Ciencias Básicas en la Facultad # 1 y adjunto a la producción en el Centro de Identificación y Seguridad Digital (CISED), Departamento de Biometría.

Correo electrónico: ylabrada@uci.cu

Agradecimientos

Agradecemos a:

Todas las personas que me han ayudado, apoyado, comprendido y guiado durante estos 5 años de mi carrera.
En especial, a todos mis profesores, mis amigos, mis dos tutores, a mi novia, a mi familia, mis padres y mi abuelita.

Giraldo David Uffo Lima.

Todas las personas las cuales han estado cerca de mí durante estos 5 años de carrera, mis amistades, mis profesores, y esta última recta mis dos tutores, a toda mi familia, en especial a mi hermana, a mi vieja y a mi abuelita.

Carlos Achang Rivas.

Dedicatoria

Dedico:

A todos mis amigos de aquí de la UCI, a Luis Orlando, Albany, Ernesto, Liset, Yidian, Neyke, Reinier, Isidro, Jose Eduardo, Adrian, Darién, Lázaro, Frank, Anita, Williams, Darío, Lily, Carlitos, Jose Augusto y Rosa por siempre apoyarme y estar a mi lado en las buenas y las malas.

A todos mis profesores, en especial: a Humberto, el Piti, Liván, Aniel, Yamila, Brisey, por tener tanta paciencia todos estos años.

A mis amigos de toda la vida: Jaime, Rogelito, Hernández, Yanni, Ernesto, Ariel, a ellos y a todas sus familias.

A toda pero toda mi familia,

A mis tíos: Alain, Marita, Tere, Yamito, Miriam, Caruca, Omar, Fernando, Sandra, Bertico.

A mis primos: Osmelito, David, Yanire, Albert, Dayan, Alian, Yaima, mi ahijado Henry.

A mi tata Yudith.

A mi padrastro Disney.

A Maité y Mayelín.

Muy especial también para mis dos tutores Labrada y Yisel, y los profesores Miguel Angel y Leo. Gracias por comprenderme, ayudarme y luchar conmigo hasta el final.

A mi novia Yaliana, siempre pero siempre recordaré todo lo que luchaste conmigo para que esto saliera adelante, gracias a Laurita por lograr que me sintiera querido como un padre y la vez permitirme quererla como una hija, a mi suegra.

A mi compañero de tesis Carlos por soportarme todo este tiempo y decir que me ponía como su papá regañándolo.

Y el rincón más especial de todos, que han sido mi ejemplo a seguir y a los que les debo el tamaño y lo que hoy soy, a mis dos madres, a mi abuelita y a mi papá.

A todos muchas gracias.

Giraldo David Uffo Lima.

Dedico:

Este trabajo de diploma y mi carrera en general, a mi mamá por haberme apoyado a pesar de lo regado que soy y por haber estado junto a mí toda la vida incluyendo estos 5 años y que va a tener que seguir cargando conmigo toda la vida.

A René mi padrastro por comportarse como un padre para mí en los años que llevamos viviendo juntos y por haber apoyando tanto a mi mamá. A mi hermana la mayor que aunque no se encuentra cerca de mí en estos momentos, supo guiarme y apoyarme todo este tiempo.

A mi papá a pesar de no haberlo tenido tan cerca como me hubiera gustado en la vida supo estar presente y aconsejarme en algunos momentos.

A mi abuela, que aunque a veces me pase sin verla años, que sepa que la quiero con la vida y la tengo presente en este momento como en otros.

A mi hermanita pequeña, a toda la familia de ella, a mi madrastra, a mis hermanastros que me ha apoyado muchísimo.

A todos mis amigos de la UCI, a los que están y los que no están, siempre los voy a recordar, ya que fueron una familia más.

A mis tutores Yainier y Yisel, por estar siempre arriba de nosotros para lograr al final este trabajo de tesis.

A Madelaine por ser mi compañera todos estos años.

A mi compañero de tesis Uffo que junto logramos llegar hasta lo último de esta recta.

A todas mis amistades del barrio: Julio, Rafael, Luis Ángel, Osniel en fin a todos.

A mis primos, en especial a Yanni y Jordan por preocuparse, a mi tía Bárbara, Vicente, a Tite.

A todos muchas gracias.

Carlos Achang Rivas.

Resumen

Las aplicaciones biométricas tienen gran importancia en el mundo actual. Los sistemas de seguridad más eficientes del mundo utilizan en alguna medida patrones biométricos para su ejecución.

En la Universidad de las Ciencias Informáticas (UCI) existe el Centro de Identificación y Seguridad Digital (CISED) y dentro del mismo, el Departamento de Biometría, donde la principal tarea es la producción de aplicaciones biométricas.

La presente investigación surge como resultado de la necesidad de contar con una aplicación en el CISED que sea capaz de realizarle pruebas de precisión a los algoritmos biométricos que son implementados.

El objetivo principal de la investigación es crear una herramienta que reduzca el tiempo de pruebas y su vez aumente la efectividad de los algoritmos biométricos implementados. Para el logro de este objetivo se implementó un sistema llamado "**Plataforma de pruebas para algoritmos biométricos**" el cual de una forma automatizada recibe el algoritmo en forma ejecutable de consola y lo procesa hasta obtener los indicadores necesarios para precisar la efectividad del mismo.

Estos resultados son almacenados en forma de reportes en una base de datos donde posteriormente pueden ser visualizados por cualquier usuario.

Palabras Claves:

Algoritmo biométrico, sistema, aplicación, prueba.

Índice

Introducción.

La biometría es el análisis de distintos procedimientos automáticos para el reconocimiento seres humanos basados en uno o más rasgos físicos o conductuales. El término se deriva de las palabras griegas "bios" de vida y "metron" de medida. [26]

La biometría informática es la aplicación de técnicas matemáticas y estadísticas sobre los rasgos físicos o de conducta de un individuo, para verificar identidades o para identificar individuos.

Las huellas dactilares, las retinas, el iris, los patrones faciales, las venas de la mano o la geometría de la palma de la misma, representan ejemplos de características físicas (estáticas), mientras que entre los ejemplos de características del comportamiento se incluye la firma, el paso y el tecleo (dinámicas). La voz se considera una mezcla de características físicas y del comportamiento, pero todos los rasgos biométricos comparten aspectos físicos y del comportamiento.

Las aplicaciones biométricas en su mayor escala son producidas por países del llamado primer mundo donde generalmente existe un gran desarrollo industrial, tecnológico en general y específicamente de las Tecnologías de la Informática y las Comunicaciones (TIC's).

Cuba, a pesar de ser un país subdesarrollado y potentemente bloqueado, no está ajena al progresivo auge de las TIC's, además aprecia la importancia que tienen para el desarrollo económico e intelectual del país y constantemente adopta medidas para implantarlas en cada sector de la sociedad. Uno de estos sectores es el Ministerio del Interior.

El desarrollo de *software* en Cuba ha evolucionado en gran medida con el surgimiento de la Universidad de las Ciencias Informáticas (UCI), centro de altos estudios creado por el líder de la Revolución cubana Fidel Castro Ruz, como parte del programa de la Batalla de Ideas.

La UCI está organizada por facultades según distintos perfiles y a su vez estas por centros de producción y desarrollo, encargados fundamentalmente de la creación de *software* para el desarrollo tecnológico y económico de la nación cubana. Uno de ellos es el Centro de Identificación y Seguridad Digital (CISED). El mismo cuenta con varios departamentos: Identificación, Tarjetas Inteligentes, Seguridad Digital, Soluciones Integrales y Biometría.

Los procesos de pruebas a los algoritmos biométricos implementados en el CISED son lentos y poco efectivos, dedicando un tiempo considerable a esta tarea, limitándose de

esta forma el desarrollo de futuros productos biométricos, así como la calidad que los mismos deben poseer.

Después de argumentada la situación problémica, el **problema científico** queda formulado de la siguiente manera: ¿Cómo acelerar el proceso de pruebas de los algoritmos biométricos implementados en el CISED?

El **objeto de estudio** de la presente investigación lo constituye: Sistemas de evaluación y control de los algoritmos biométricos, y el **campo de acción** es: Plataforma de prueba para algoritmos biométricos.

Para dar solución a la problemática planteada se define como **objetivo general** de la investigación: Automatizar el proceso de pruebas de precisión de algoritmos biométricos incluidos en las tecnologías biométricas desarrolladas en CISED.

Derivándose los siguientes objetivos específicos:

1- Caracterizar las plataformas de pruebas existentes utilizadas en pruebas a algoritmos biométricos.
2- Utilizar una metodología de desarrollo de *software* para la plataforma de pruebas dirigida a evaluar los algoritmos biométricos.
3- Implementar una Plataforma de prueba para algoritmos biométricos que se ajuste a la arquitectura anteriormente diseñada.

Idea a defender:

Con el desarrollo de una Plataforma de pruebas para algoritmos biométricos, se agilizará el proceso de pruebas de los algoritmos biométricos implementados en el CISED.

Para dar cumplimiento a los objetivos se definen las siguientes **Tareas científicas:**

1. Elaborar un mapa conceptual mediante la investigación de los conceptos fundamentales asociados a Plataformas de pruebas para algoritmos biométricos, con el objetivo de identificar las características generales de las mismas.
2. Caracterizar las plataformas de pruebas existentes utilizadas en pruebas a algoritmos biométricos (Ejemplo: FVC2006), en cuanto a:
    i. Integración a bases de datos de referencia biométrica.
    ii. Protocolos de pruebas.
    iii. Sistemas y coeficientes de evaluación.
    iv. Reportes de los resultados de las pruebas a algoritmos biométricos.
3. Utilizar una metodología de desarrollo de *software* para la plataforma de pruebas, incluyendo:
    i. Fases y sus actividades.

ii. Roles que intervienen en cada fase o actividad.
iii. Comunicación entre los distintos roles.
iv. Características del ciclo de desarrollo.
v. Artefactos generados en las distintas fases del desarrollo.

4. Diseñar una arquitectura que exponga las funcionalidades para que pueda ser integrable a otras aplicaciones desarrolladas en el CISED.

i. Arquitectura multicapas.
ii. Exponer la funcionalidad a sistemas externos mediante las interfaces necesarias.
iii. Flexibilidad para conectarse a bases de datos de referencia biométrica.

5. Implementar una Plataforma de prueba para algoritmos biométricos que se ajuste a la arquitectura anteriormente diseñada y que brinde, entre otras, las funcionalidades siguientes:

i. Conexión a bases de datos de referencia biométrica.
ii. Ejecución de plan de pruebas y cálculos de indicadores.
iii. Reportes de los resultados.

6. Documentar la Plataforma de prueba para algoritmos biométricos.

Para llevar a cabo las tareas científicas se emplearon métodos teóricos y empíricos de la investigación científica. Los **métodos teóricos** utilizados para cumplir con las tareas a desarrollar fueron:

Inductivo-Deductivo: se pasa de un conocimiento particular a un conocimiento más general que va a reflejar lo que hay en común entre esos fenómenos individuales. Además permite inferir conclusiones y predicciones a partir de los conocimientos adquiridos de los procesos de pruebas para algoritmos biométricos.

Histórico Lógico: este método se utiliza con el fin de posibilitar un mejor análisis histórico de los procesos de pruebas para algoritmos biométricos, es decir, permite analizar la trayectoria de dichos procesos, desde su desenvolvimiento hasta las conexiones históricas más importantes.

Analítico-Sintético: posibilitó el análisis de los procesos de pruebas para determinar con exactitud cómo este funciona. Como resultado, se toman todas las características principales para lograr modelar un sistema que logre una integración eficaz y una armonía dentro de los procesos que rigen su comportamiento.

El **método empírico** utilizado para obtener información sobre el objeto de estudio fue:

Observación: este método se utiliza con el objetivo de permitir investigar los procesos de manera externa sin tener que llegar a la esencia de los mismos, lo que ayudó al planteamiento del problema científico. Además permitió conocer bien el proceso delimitado como objeto de estudio, lo cual propició un conocimiento más detallado de lo que se quiere, lo que hace falta hacer y cómo hay que hacerlo.

La investigación está estructurada en tres capítulos:

Capítulo 1: **Fundamentación teórica**: contiene la fundamentación teórica de la propuesta presentada, así como un análisis crítico y la tendencia de las tecnologías, técnicas y las metodologías a emplear para dar solución al problema. También se realiza una investigación sobre la evaluación y pruebas de precisión de algoritmos biométricos, haciéndose una descripción y exponiendo los conceptos fundamentales de las mismas.

Capítulo 2: **Características del sistema**: se describe el negocio a través de un modelo de Dominio y, a partir de este, se comienza a hacer el análisis del sistema a desarrollar. Se enumeran los requisitos funcionales y no funcionales que debe tener la aplicación, y se muestran en forma de diagrama, los Casos de Usos derivados de ellos.

Capítulo 3: **Construcción y elaboración del sistema**: se construye la solución que se propone, se modelan los diagramas de clases de análisis, los diagramas de clases del diseño y los diagramas de interacción. Se definen los principios de diseño de la interfaz y se aborda la implementación. Por último contiene los diagramas de despliegue y de componentes, así como una breve descripción de estos procesos.

Capítulo 1. Fundamentación teórica.

1.1 Introducción.

En el presente capítulo se expone una visión general de los aspectos más significativos relacionados con los procesos de pruebas para algoritmos biométricos. Además se describen los principales conceptos asociados al dominio del problema. Se hace un amplio análisis acerca de las tecnologías, metodologías, estándares y herramientas utilizadas en la actualidad para el desarrollo de sistemas informáticos (Plataforma de pruebas a algoritmos biométricos). Este capítulo contiene la fundamentación teórica de la propuesta de solución a presentar, así como el estado de los procesos de pruebas a algoritmos biométricos en la actualidad.

1.2 Conceptos asociados al dominio del problema.

1.2.1 ¿Qué es un algoritmo biométrico?

Un **algoritmo biométrico** es un conjunto preescrito de instrucciones o reglas bien definidas, ordenadas y finitas [26] que permiten comprobar la identidad de una persona mediante una característica estática o dinámica que pueda ser reconocida o verificada de una manera automatizada.

1.2.2 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 [31]. Y del concepto de **precisión**: que no es más que la exactitud con la que se realiza una medición o cálculo dados [32]. Podemos 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.

1.3 ¿Cómo se determina la precisión de un sistema biométrico?

La precisión de un sistema biométrico se determina 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 vivo en el lugar (evaluación

operativa), todo esto antes de comenzar con las operaciones completas. Cada evaluación cumple un fin diferente e incluye distintos tipos de análisis.

Los términos biométricos como reconocimiento, verificación e identificación, suelen utilizarse al azar. Esto no sólo resulta confuso, sino que es incorrecto ya que cada término tiene un significado diferente. [1]

1.3.1 ¿Qué se entiende por reconocimiento, verificación e identificación?

Reconocimiento: 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 inscripta.

Verificación: es una 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.

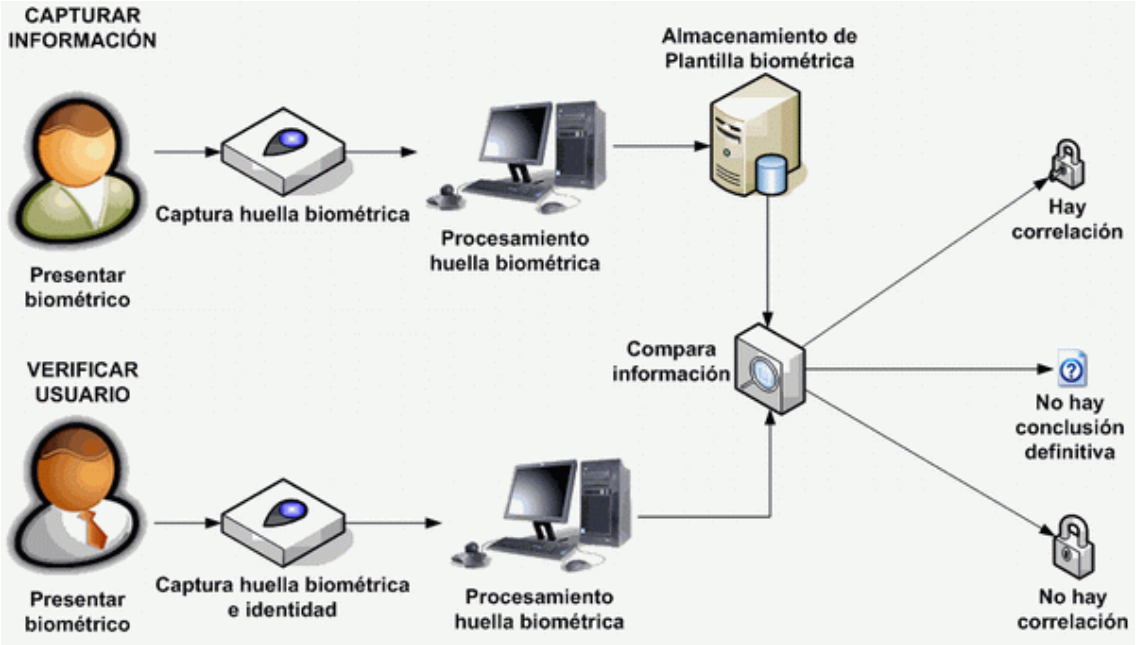

Figura 1 Proceso de Captura y Verificación de usuario. [2]

Identificación: es una 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.

Debido a esta falta de concordancia, se deben utilizar diferentes estadísticas para cada tarea. [2]

1.3.1.1 Verificación.

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. [2]

Falsa aceptación: si se acepta un impostor. [3]

Falso rechazo: si un cliente verdadero es rechazado. [3]

Tasa de Falsos Aceptados (FAR):

Es la probabilidad de que un individuo no autorizado sea autenticado. Por ejemplo: Giraldo declara ser Carlos, y el sistema afirma esta declaración [4]. En general entre más bajo sea el valor de la tasa de falsa aceptación, más alta es la precisión del sistema biométrico. El valor depende de la sensibilidad del área o sistema a proteger y de la necesidad del usuario. 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 Fórmula 1: [2]

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

Fórmula 1 Tasa de Falsos Aceptados.

Tasa de falsos rechazados (FRR):

Es la probabilidad de que un individuo autorizado sea inapropiadamente rechazado. Por ejemplo Giraldo declara ser Giraldo 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. El valor depende de lo sensible del área o sistema a proteger y de la necesidad del usuario. 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 Fórmula 2: [4]

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

Fórmula 2 Tasa de Falsos Rechazados.

El punto de intersección entre la tasa de falsa aceptación y la tasa de falso rechazo se conoce como:

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. [2]

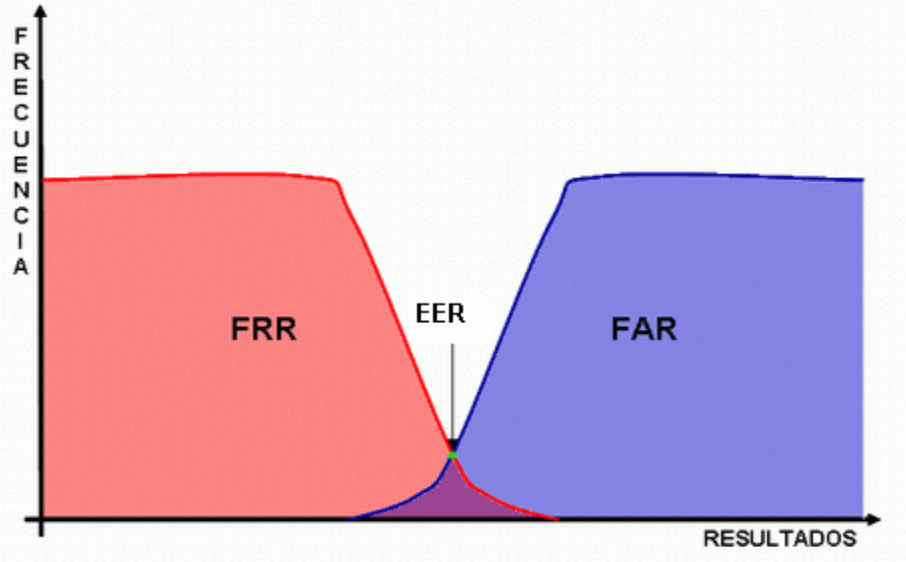

Figura 2 Gráfica de Tasa de error igual. [2]

La tasa de error igual 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: [3]

$$EER = \begin{cases} \frac{FAR(t_1)+FRR(t_1)}{2} & \text{if } FAR(t_1) - FRR(t_1) \leqslant FRR(t_2) - FAR(t_2) \\ \frac{FAR(t_2)+FRR(t_2)}{2} & \text{otherwise} \end{cases}$$

Fórmula 3 Cálculo del EER.

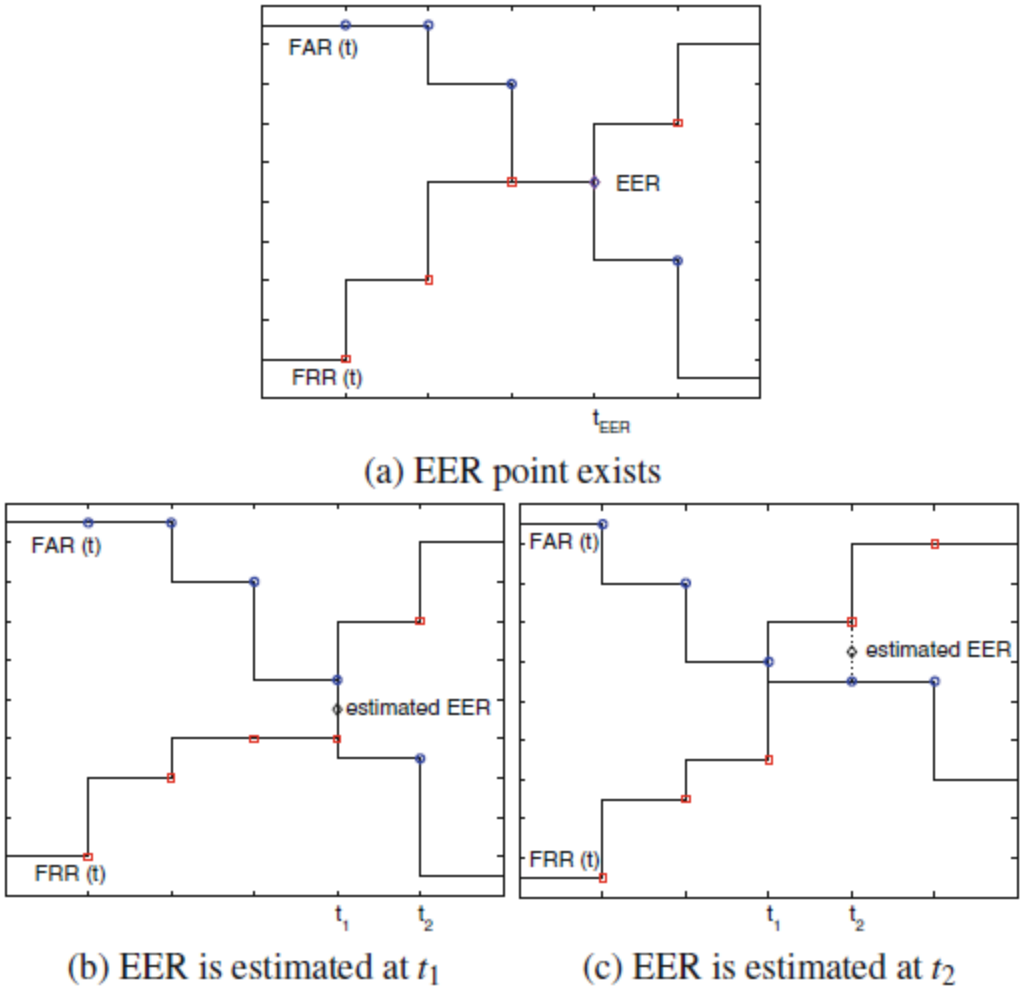

Figura 3 Curva FARvsFRR: (a) ejemplo donde el punto de EER existe, (b) y(c) ejemplos donde el punto EER no existe. [3]

Tasa de Falsa Coincidencia (FMR):

La probabilidad de que un sistema biométrico identifique incorrectamente un individuo o que falle para rechazar un impostor. Alternativa a Tasa de Falsa Aceptación (FAR). [2]

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. [2]

Error tipo I: este tipo de error ocurre en una prueba estadística cuando una reclamación válida es rechazada. Es decir cuando falla al rechazar una reclamación válida. Por ejemplo Giraldo reclama ser Giraldo, pero el sistema niega el reclamo de manera incorrecta.

Error Tipo 2: este tipo de error ocurre en una prueba estadística cuando una reclamación falsa es aceptada. Es decir cuando falla al aceptar una reclamación falsa.

Por ejemplo Carlos reclama ser Giraldo y el sistema acepta el reclamo de manera incorrecta. [2]

Tasa de verificación:

Tasa en la cual los usuarios finales legítimos son correctamente verificados.

1.3.1.2 Identificación.

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:

Porcentaje de veces que una alarma suena incorrectamente ante un individuo que no es parte de la base de datos del sistema biométrico (el sistema emite alarma ante Carlos, y Carlos no es parte de la base de datos), o cuando suena una alarma, pero se identifica a la persona equivocada (el sistema emite alarma ante Giraldo, cuando Giraldo es parte de la base de datos, pero el sistema piensa que Giraldo es Yanier).

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 hagan, puede presentar una de tres posibilidades dependiendo de la puntuación que se alcance en la comparación de la plantilla y 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. [2]

1.4 El 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 α del FAR depende de la modalidad. En la práctica, el OP se calcula de la siguiente manera:

$$OP_{\{FAR=\alpha\}} = FRR(t_{OP}) \mid t_{OP} = \max_{t \in S}\{t \mid \alpha \leqslant FAR(t)\}$$

Fórmula 4 Cálculo de OP.

Donde S es el conjunto de los umbrales utilizados para calcular las distribuciones de resultados. [28]

1.5 Curva *Receiver Operating Characteristic* (ROC).

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. [28]

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 decidimos que un caso es un positivo). Es decir ROC como una comparación de dos características operativas (VPR y FPR) según cambiamos el umbral para la decisión. [29]

1.6 Estándares.

Los estándares ayudan a los usuarios a utilizar y a mantener los sistemas de manera fácil, al mismo tiempo que promueven una mayor duración de los sistemas y permiten operaciones entre los mismos. [5]

1.6.1 ¿Qué es un estándar?

Un estándar es un conjunto de reglas que deben cumplir los productos, procedimientos o investigaciones que afirmen ser compatibles con el mismo producto. Los estándares ofrecen muchos beneficios, reduciendo las diferencias entre los productos y generando un ambiente de estabilidad, madurez y calidad en beneficio de consumidores e inversores. [5]

1.6.2 Estándares Biométricos.

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. Los esfuerzos que se están realizando y los ya realizados han perseguido distintos objetivos, por ejemplo la definición de API (Interfaz de Programación de Aplicaciones), los formatos de los ficheros con la información de parámetros biométricos, la encriptación de la información biométrica y la interacción entre dispositivos biométricos diferentes. [5]

1.6.3 Estándares Biométricos Internacionales.

1.6.3.1 BioAPI.

El consorcio BioAPI 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. Crear un estándar independiente del sistema operativo y trabajar de forma coordinada con desarrolladores e integradores de aplicaciones con elementos biométricos para generar una API de fácil uso y de fácil convivencia con otros estándares ya existentes.

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, ofreciendo 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 're-entrenarse' 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. [5]

1.6.3.2 Estándar BAPI.

BAPI es un estándar biométrico desarrollado y planeado por un vendedor de soluciones biométricas llamado I/O *Software* en lugar de un consorcio de compañías e instituciones como fue el caso de BioAPI. En Mayo de 2000, Microsoft licenció BAPI, aunque había sido uno de los primeros en apostar por BioAPI, 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. En la actualidad el mundo de los estándares biométricos se encuentra dividido entre BAPI (apoyado por el consorcio Microsoft / Intel, en el que han colaborado otras compañías que también participan en el consorcio BioAPI), y BioAPI (considerado como el estándar de facto por agencias del gobierno de Estados Unidos para sus aplicaciones de seguridad). Esto nos conduce a una situación indeseada, contraria a la propia naturaleza de los esfuerzos encaminados a la generación de un único estándar. En los próximos años, BioAPI y BAPI deberán tener que ser considerados por todos los desarrolladores de aplicaciones hasta que ambos converjan en un único y definitivo estándar biométrico. [5]

1.6.3.3 CBEFF (*Common Biometric Exchange File Format*).

Se ha desarrollado un estándar conocido como Formato de Ficheros Común para el Intercambio Biométrico (CBEFF), cuyo 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. CBEFF establece un formato para la cabecera de los ficheros definiendo campos obligatorios y opcionales que proporcionan elementos comunes (opciones de seguridad, de integridad de los datos, fecha de creación del fichero, firma, tipo de parámetro biométrico) para el intercambio de información entre los dispositivos biométricos y los sistemas que hacen uso de los mismos, además 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.

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.*

Actualmente BioAPI y CBEFF se han unido para construir un frente común a la estandarización biométrica. Muchos fabricantes están adoptando este estándar ofreciendo soluciones compatibles CBEFF, lo que implica que los ficheros que contienen los datos de los patrones biométricos tienen esta cabecera común. [5]

1.7 Ámbito Internacional.

1.7.1 Campañas de Evaluación

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.

NIST también es muy activa en las huellas dactilares. Incluso en 2003 se desarrollaron campañas en esta modalidad.

Las actividades de evaluación reconocimiento facial fueron impulsadas en 1993 por la disponibilidad de *Face Recognition Technology* (FERET) [6]. Esta actividad fue trasladada al NIST, que organizó la primera prueba a vendedores de reconocimiento facial (FRVT) en 2000. El *Face Recognition Grand Challenge* (FRGC) se llevó a cabo a continuación, para facilitar la investigación sobre esta tecnología. El FRVT continúo su desempeño en otras convocatorias en 2004 y 2006.

NIST organizó la primera campaña de *Iris Challenge Evaluation* (ICE) en el 2006 [7], además de lanzar el *Multiple Biometric Grand Challenge* (MBGC) en el 2008.

Pero NIST no es la única organización que participan en la evaluación biométrica. La *Fingerprint Verification Competitions* (FVC) fueron organizadas por la Universidad de Bologna, cada dos años desde 2000.

La *BioSecure Network of Excellence* a respaldado la recogida de datos, desarrollo de *software* y la infraestructura necesaria para llevar a cabo la *BioSecure Multimodal Evaluation Campaign* in 2007 [8].

1.7.1.1 *The Face Recognition Vendor Tests* (FVRT).

Las pruebas a vendedores de reconocimiento facial (***The 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 [9]:

Evaluar las capacidades de los sistemas de reconocimiento facial comercialmente disponibles y educar la comunidad de biometría y el público general sobre cómo presentar y analizar resultados apropiadamente. FRVT 2002 [10] 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 radiaciones 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 uso de modelos moldeables, los cuales mapean una imagen 2D sobre una grilla 3D en un intento de superar radiaciones posturales y de iluminación, puede mejorar significantemente el reconocimiento facial no frontal. El rendimiento de la lista de

vigilancia decrece como función del tamaño de una galería el rendimiento, utilizando listas de vigilancia más pequeñas lo cual es mejor que utilizando las más grandes.
En aplicaciones de reconocimiento facial, las ubicaciones deben ser hechas para la información demográfica ya que características como la edad y el sexo pueden afectar significativamente el rendimiento.
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 [11] la cual estaba abierta a universidades, institutos de investigación y empresas. Los algoritmos enviados fueron probados en datos recogidos de 330 sujetos.

1.7.1.2 *Face Recognition Grand Challenge* (FRGC).

La meta del FRGC fue promover y adelantar la tecnología de reconocimiento facial diseñada para dar soporte a los esfuerzos existentes de reconocimiento facial 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.

1.7.1.3 *The Fourth International Fingerprint Verification Competition* (FVC2006).

La competencia se centra en la evaluación de *software* de huellas digitales de verificación. Un subconjunto de impresiones de huellas dactilares adquiridas con varios sensores se proporcionó a los participantes inscritos, que les permitió ajustar los parámetros de sus algoritmos. A los participantes se les pidió inscribirse y hacer coincidir los archivos ejecutables de los algoritmos. La evaluación se llevó a cabo en las instalaciones de los organizadores con los expedientes presentados ejecutable en una base de datos, adquiridos con los mismos sensores como el conjunto de entrenamiento. [12]

¿Cómo se organiza y se participa en el FVC?

El FVC comienza con una llamada dentro de la comunidad biométrica para la inscripción en el evento. Los participantes inscritos reciben un número de identificación confidencial para sus algoritmos.
Después de formalizar la inscripción, los participantes reciben una pequeña muestra de las imágenes de cada una de las bases de datos que serán utilizadas en la competición. Estas muestras les permiten a los participantes conocer con anticipación

las características del tipo de sensor y de las imágenes que su algoritmo tendrá que enfrentar. Las imágenes que forman parte de las muestras no son utilizadas posteriormente en la competición.

Una vez que termina el período de recepción de los algoritmos, comienza el proceso de ejecución, evaluación y agregación de resultados que demora varios meses. Cuando este proceso concluye, cada participante recibe confidencialmente los resultados de su algoritmo junto con los promedios de los diez algoritmos mejor colocados en cada base. A partir de estos resultados cada participante decide si su nombre debe aparecer explícitamente o de forma anónima en la publicación de los resultados finales.

El desempeño y precisión de un algoritmo va a ser diferente dependiendo del tipo de sensor utilizado pues las imágenes de sensores diferentes que presentan características diferentes. Por esta razón existen dos tipos de resultados o métricas que el FVC evalúa en los algoritmos participantes: resultados medios que son calculados considerando todas las bases de imágenes y resultados por base, calculados en cada base de imágenes de manera individual.

Los resultados o métricas medias miden el comportamiento promedio cuando el algoritmo es aplicado tanto en las imágenes de uno u otro sensor. Estos resultados describen cómo el algoritmo resuelve de manera universal el problema de la verificación de huellas dactilares, o sea, describen la robustez y capacidad del algoritmo de funcionar si se cambia el lector de huellas por otro de otra característica. Estos resultados medios son los de mayor interés para la comunidad académica e industrial y guían los avances del estado del arte en esta área.

Por otro lado, los resultados o métricas por base describen cómo el algoritmo se comporta para cada tipo específico de sensor y de imagen. Estos resultados permiten valorar y premiar algoritmos que no son buenos de forma universal para todos los sensores, pero que obtienen buenos resultados para sensores de tecnologías específicas. Los resultados medios son calculados a partir da integración de los resultados por base. [20]

1.7.1.4 *BioSecure Multimodal Evaluation Campaign* de 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, la red ha desarrollado un marco de evaluación de rendimiento,

incluidos los sistemas de referencia y se recoge una importante base de datos biométrica multimodal. 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.

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 la cara en 2D en las secuencias de video.

Evaluación multimodal: Multimodal se presenta a menudo como una manera de mejorar el rendimiento de los sistemas de monomodal, especialmente en el caso de las condiciones degradadas, y para mejorar la resistencia a las falsificaciones. 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.

En total, más de 20 organizaciones no sólo de bioseguridad, participaron en estas evaluaciones. Sólo el desarrollo de los datos se distribuyó a los participantes, mientras que las pruebas finales se han realizado en privado con el fin de evitar la divulgación de los datos biométricos sensibles.

El último taller de bioseguridad, que se celebró a finales de septiembre de 2007, reunió 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 están a disposición del público en la dirección del BMEC, a través de la entrega pública de bioseguridad titulado "Informe sobre las campañas de evaluación de bioseguridad multimodal" y a través de varias publicaciones en revistas y conferencias. [13]

1.7.1.5 *The BioSecure Signature Evaluation Campaign* (BSEC) 2009.

Para participar en *The BioSecure Signature Evaluation Campaign* (BSEC) 2009 cada sistema tiene que estar compuesto por 3 ejecutables y 2 documentos descriptivos que se definen en la parte I y la parte II respectivamente. En la Parte III se presentan los archivos de entrada para cada tarea.

I Ejecutables:

Los participantes deberán presentar tres ejecutables para cada sistema presentado. La entradas / salidas de estos ejecutables son impuestas por el organizador.

a) Extracción de características:

Descripción: Este ejecutable tiene como objetivo la extracción de características del archivo de firma de entrada.

Uso: featureExtraction.exe **inputData featuresFolder tmpDirectory**

Entrada:

InputData: ruta absoluta al archivo de firma de entrada para ser procesado.

Salidas:

featuresFolder: nombre de la carpeta que contendrá las características extraídas. Esta carpeta tiene que ser creada por el ejecutable. **tmpDirectory**: ruta absoluta al directorio en el que la carpeta. featuresFolder estará almacenada. Este directorio también se utiliza para almacenar todos los archivos temporales creados por el ejecutable.

Estos archivos tienen que ser eliminados al final de la etapa de extracción de características.

Ejemplo

./featureExtraction/home/database/u010s0001_sf1.txt

u010s0001_sf1/home/tmp/

b) La creación del Modelo

Descripción: Este ejecutable tiene como objetivo la construcción de un modelo de cliente que utiliza vectores de características extraídas de cinco firmas auténticas.

Uso: modelCreation.exe **targetIdList tmpDirectory**

Entrada:

targetIdList: ruta absoluta a un archivo de texto que asocia un nombre de modelo a las características.

La forma de este archivo se da en la figura 3. En la primera columna, modelFolder corresponde al nombre de la carpeta en la que el modelo se almacenará. Esta carpeta tiene que ser creada por el ejecutable mientras que las otras columnas mencionan las cinco carpetas que contienen las características utilizadas para construir el modelo. Estas carpetas de características deben haber sido creadas por el ejecutable featureExtraction.

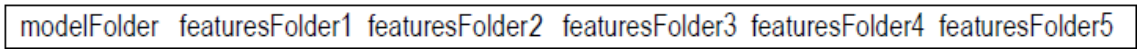

Figura 4 Forma del archivo targetIdList. [21]

Salida:

tmpDirectory: ruta absoluta al directorio en el que la carpeta modelFolder será almacenada. Este directorio también se utiliza para almacenar todos los archivos temporales creados por el ejecutable. Estos archivos tienen que ser eliminados al final de la etapa de creación de modelos. Este directorio es el mismo, utilizado en el paso anterior, por lo que contiene también la carpeta de características.

Ejemplo:

./modelCreation.pl /home/targetIdList/m080s0001_sg.txt /home/tmp/

Donde el contenido de /home/targetIdList/ m080s0001_sg.txt es:

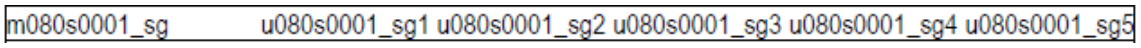

Figura 5 Formato del archivo m080s0001_sg.txt [21]

c) Matching

Descripción: Este ejecutable tiene como objetivo el macheo de las características de prueba y un modelo de cliente, así como proporcionar una puntuación.

Uso: Matching.exe **modelFolder featuresFolder tmpDirectory**

Entradas:

modelFolder: nombre de la carpeta que contiene el modelo cliente para probar.

featuresFolder: nombre de la carpeta que contiene las características de la prueba.

tmpDirectory: ruta absoluta al directorio que contiene las carpetas modelFolder y featuresFolder. Este directorio también se utiliza para almacenar todos los archivos temporales creados por el ejecutable. Estos archivos tienen que ser eliminados al final de la etapa correspondiente.

Salida:

el ejecutable proporciona una puntuación de similitud que se muestra en la pantalla.

Ejemplo:

./matching m04008s00 01_sg u04008s0002_sg6 /home/tmp/

II Documentos:

Cada participante debe enviar dos documentos descriptivos del sistema al organizador.

- Un documento teórico que contiene una descripción de alto nivel de su algoritmo (métodos de tratamiento previo, las características extraídas, el algoritmo usado para construir el modelo, estrategia de comparación).
- Un documento técnico que describe la forma y la arquitectura de sus ejecutables (archivos que precisa el ejecutable para que funcione correctamente).

III Ficheros de datos de entrada

Para la BSEC, un archivo de entrada sigue siendo un archivo de texto que contiene información sobre la dinámica de una firma. Esta información depende de la tarea y tiene que ser procesado por el ejecutable featuresExtraction. [21]

1.8 Lenguajes, herramientas, tecnologías y metodología.

1.8.1 UML.

UML es un lenguaje gráfico para especificar, construir y documentar los artefactos que modelan un sistema. Fue diseñado para ser un lenguaje de modelado de propósito general, por lo que puede utilizarse para especificar la mayoría de los sistemas basados en objetos o en componentes. También para modelar aplicaciones de muy diversos dominios de aplicación y plataformas de objetos distribuidos (como por ejemplo J2EE, .NET). El hecho de que UML sea un lenguaje de propósito general proporciona una gran flexibilidad y expresividad a la hora de modelar sistemas.

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.

1.8.1.1 UModel 2007 de Altova.

Es el punto de entrada para el desarrollo de software exitoso. Permite crear e interpretar diseños de software mediante la potencia del estándar UML 2.1. Dibuja el diseño de la aplicación y genera un código Java o C# a partir de sus planos. Es una herramienta que permite realizar la ingeniería inversa de programas existentes, a diagramas UML claros y precisos para abarcar rápidamente la arquitectura. Además puede corregir el código generado sincronizado con el modelo y viceversa.

1.8.2 Metodologías de desarrollo de software.

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Indican, paso a paso, todas las actividades a realizar para lograr el producto informático deseado, además de las personas que deben participar en el desarrollo de las actividades y qué papel deben tener. Detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. Las metodologías se encargan de elaborar estrategias de desarrollo de software que promuevan prácticas centradas en las personas o equipos, orientadas hacia la funcionalidad y la entrega de comunicación intensiva y que requieren implicación directa del cliente. Una metodología de desarrollo de software permite producir organizada y económicamente software de alta calidad, siguiendo una serie de pasos donde se utilizan un conjunto de técnicas, notación y normas de documentación preestablecidas.

1.8.2.1 RUP.

El Proceso Racional Unificado (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, como lenguaje de representación visual. En RUP se han agrupado las actividades 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.

El proceso de ciclo de vida de RUP se divide en cuatro fases bien conocidas, llamadas Concepción, Elaboración, Construcción y Transición.

Esas fases se dividen en iteraciones, cada una de ellas produce una pieza de software demostrable.

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 requerimientos (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 1 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.

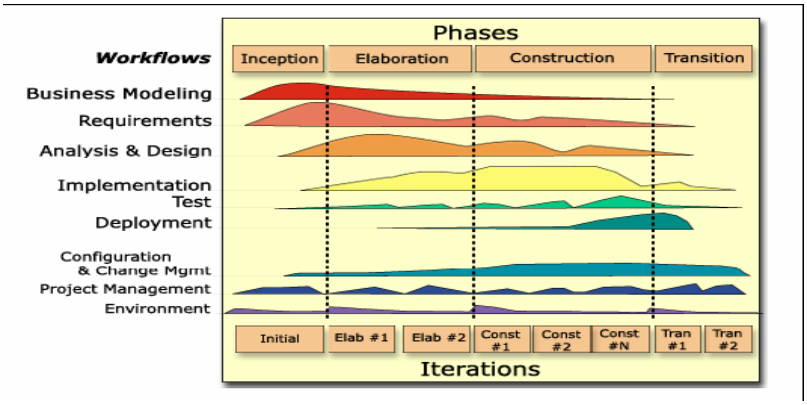

Figura 6 Ciclo de vida de RUP.

El ciclo de vida de RUP se caracteriza por estar:

- **Dirigido por Casos de Uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo pues los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).
- **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción. Por otra parte, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
- **Iterativo e incremental:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

Se utiliza RUP como metodología de desarrollo de software debido a las ventajas que la misma proporciona, aunque la versión inicial de la plataforma no requiera de complejos casos de uso. Las versiones posteriores estarán integradas por distintos módulos y las funcionalidades que se puedan tener al final de la implementación, serán ampliadas paulatinamente.

1.8.3 Herramientas para el modelado.

La misión de cualquier herramienta CASE[1], que utiliza UML como notación para elaborar los modelos, es comunicar, de la manera más eficiente posible, a los agentes del proyecto, todas aquellas decisiones que se toman con respecto a la arquitectura del sistema en discusión y que son determinantes para cumplir con los objetivos de las distintas fases de un proyecto.

[1] **CASE** (***C**omputer **A**ided **S**oftware **E**ngineering*, Ingeniería de Software Asistida por Computadora).

1.8.3.1 *Rational Rose Enterprise Suite*.

El *Rational Rose*, es una herramienta CASE desarrollada por *Rational Corporation*, basada en UML. La misma permite crear los diagramas que se van generando durante el proceso de Ingeniería en el desarrollo del software. Las personas que desarrollaron RUP son miembros de *Rational Corporation*, por lo que es completamente compatible con esta metodología y brinda muchas facilidades en la generación de la documentación del *software* que se está desarrollando. Además posee un gran número de estereotipos predefinidos que facilitan el proceso de modelación del *software*.

Dicha herramienta es capaz de generar el código fuente de las clases definidas en el flujo de trabajo de diseño, pero tiene la limitación de que aún hay varios lenguajes de programación que no soporta o que sólo lo hace a medias. Por otra parte, una vez que se tiene el diagrama de clases persistentes a partir del cual se genera la base de datos del sistema, no existe la posibilidad de exportar ese modelo hacia algún sistema gestor de bases de datos. [14]

1.8.4 Lenguaje C# 3.0.

C# es un lenguaje de programación que está basado en otro muy utilizado del que, en parte, toma su nombre. Ese lenguaje es C++ y no es algo que ocurra por primera vez, ya en el pasado otros desarrolladores tomaron C++ como base para crear nuevos lenguajes. Es el caso, por ejemplo, de Java.

El objetivo de Microsoft ha sido la creación del primer lenguaje orientado a componentes, al estilo de *Visual Basic*, pero con la flexibilidad y potencia de C++ y sin muchas de sus complejidades. C, C++ o *Visual Basic* están pensados para el desarrollo de aplicaciones en sistemas operativos que ofrecen sus servicios a través de cientos o miles de funciones independientes que, en conjunto, forman lo que se conoce coloquialmente como API (*Application Programming Interface*). En contraposición, C# ha sido diseñado para una plataforma, la Microsoft .NET, en la que los servicios son ofrecidos en forma de componentes. Aunque C# es un primo muy cercano de C++, también tiene influencias de *Visual Basic*, Modula 2, Smalltalk y Java. El resultado es un lenguaje orientado al trabajo con componentes que mantiene, en su mayor parte, los operadores y estructuras propias del lenguaje C++. [16]

C # 3.0 introduce varias extensiones del lenguaje que se basan en C # 2.0 a apoyar la creación y uso de orden superior, las bibliotecas de clases funcionales de estilo. Las extensiones permiten la construcción de la API de composición que tienen el mismo

poder expresivo de los lenguajes de consulta en dominios tales como bases de datos relacionales y XML. [15]

1.8.4.1 Orientación a componentes.

Además de un lenguaje orientado a objetos, C# es el primer lenguaje de la familia C/C++ que también está orientado al trabajo con componentes. Esto significa que cuenta con construcciones sintácticas nativas para la definición, implementación y consumo de propiedades, métodos y eventos, lo cual le diferencia claramente de C++ o Java.

Para definir una propiedad en el interior de una clase C# se utiliza una sintaxis especifica, no necesitando la simulación mediante métodos que se ajustan a un cierto patrón de nomenclatura. Igualmente, pueden utilizarse eventos sin necesidad de tratar con punteros a funciones o métodos. La existencia de palabras clave, como ***property, event*** *y* ***delegate*****,** simplifica la construcción de estos elementos que, como sabemos, son fundamentales a la hora de trabajar con componentes, no sólo con objetos.

En este aspecto, por tanto, C# asimila, e incluso supera, las tradicionales facilidades que da *Visual Basic* a la hora de usar y diseñar componentes sin por ello, perder un ápice de potencia y flexibilidad. Para construir un componente con C# no es necesario hacer nada especial aparte de crear una nueva clase. Dicho de otro modo, cualquier clase de objeto C# es un componente, sin necesidad de crear GUID (*Globally Unique Identifier)* para interfaces y clases de componentes, y así efectuar operaciones de registro, implementar interfaces como ***IUnknown*****,** etc., operaciones todas ellas habituales cuando se utiliza C++.

La documentación de las clases, especialmente interesante cuando se crean componentes para ser utilizados por otros miembros de un equipo o incluso por terceros, es generada automáticamente a partir de los comentarios incluidos en el código. Estos siguen una notación XML, siendo C# el primer lenguaje en utilizar este sistema con este fin concreto. [16]

1.8.5 *Visual Studio* 2008.

Visual Studio 2008 fue publicado (RTM) el 17 de Noviembre de 2007 en inglés, mientras que la versión en castellano no fue publicada hasta el 2 de Febrero de 2008. [22]

El nuevo *framework* (.Net 3.5) está diseñado para aprovechar las ventajas que ofrece el nuevo sistema operativo "Windows Vista" a través de sus subsistemas "*Windows Communication Foundation*" (WCF) y "*Windows Presentation Foundation*" (WPF). El

primero tiene como objetivo la construcción de aplicaciones orientadas a servicios, mientras que el último apunta a la creación de interfaces de usuario más dinámicas que las conocidas hasta el momento. [23]

A las mejoras de desempeño, escalabilidad y seguridad con respecto a la versión anterior, se agregan entre otras, las siguientes novedades.

• La mejora en las capacidades de Pruebas Unitarias permiten ejecutarlas más rápido independientemente de si lo hacen en el entorno IDE o desde la línea de comandos. Se incluye además un nuevo soporte para diagnosticar y optimizar el sistema a través de las herramientas de pruebas de *Visual Studio*. Con ellas se podrán ejecutar perfiles durante las pruebas para que ejecuten cargas, prueben procedimientos contra un sistema y registren su comportamiento; y utilizar herramientas integradas para depurar y optimizar.

• Con *Visual Studio Tools for Office* (VSTO) integrado con *Visual Studio* 2008 es posible desarrollar rápidamente aplicaciones de alta calidad basadas en la interfaz de usuario (UI) de Office que personalicen la experiencia del usuario y mejoren su productividad en el uso de Word, Excel, PowerPoint, Outlook, Visio, InfoPath y Project. Una completa compatibilidad para la implementación con ClickOnce garantiza el entorno ideal para una fácil instalación y mantenimiento de las soluciones Office.

• *Visual Studio* 2008 permite incorporar características del nuevo *Windows Presentation Foundation* sin dificultad tanto en los formularios de Windows existentes como en los nuevos. Ahora es posible actualizar el estilo visual de las aplicaciones al de Windows Vista debido a las mejoras en *Microsoft Foundation Class Library* (MFC) y Visual C++. *Visual Studio* 2008 quien permite mejorar la interoperabilidad entre código nativo y código manejado por .NET. Esta integración más profunda simplificará el trabajo de diseño y codificación.

• LINQ (*Language Integrated Query*) es un nuevo conjunto de herramientas diseñado para reducir la complejidad del acceso a Base de Datos, a través de extensiones para C++ y *Visual Basic* así como para Microsoft .NET *Framework*. Permite filtrar, enumerar, y crear proyecciones de muchos tipos y colecciones de datos utilizando todos la misma sintaxis, prescindiendo del uso de lenguajes especializados como SQL o XPath.

• *Visual Studio* 2008 ahora permite la creación de soluciones de multiplataforma adaptadas para funcionar con las diferentes versiones de .Net *Framework*: 2.0. (Incluido con *Visual Studio* 2005), 3.0 (incluido en Windows Vista) y 3.5 (incluido con *Visual Studio* 2008).

• .NET 3.5 incluye biblioteca ASP.NET AJAX para desarrollar aplicaciones web más eficientes, interactivas y altamente personalizadas que funcionen para todos los navegadores más populares y a su vez utilicen las últimas tecnologías y herramientas Web, incluyendo Silverlight y Popfly.

1.8.6 Base de Datos.

Cualquier aplicación de interés requiere el almacenamiento y posterior recuperación de los datos con los que este trabaje (pedidos en aplicaciones de comercio electrónico, datos del personal para las aplicaciones de recursos humanos, etc.). Los sistemas de gestión de bases de datos (DBMSs) nos permiten almacenar, visualizar y modificar datos, así como hacer copias de seguridad y mantener la integridad de los datos, 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 (esto es, los datos no deben estar duplicados) para evitar problemas de consistencia e integridad. [17]

1.8.6.1 PostgreSQL.

Es una base de datos relacional, distribuida bajo licencia BSD y con su código fuente disponible libremente. Es el motor de bases de datos de código abierto más potente del momento y en sus últimas versiones empieza a no tener que envidiarle nada a otras bases de datos comerciales.

Sus características técnicas la hacen una de las bases de datos más potentes y robustas del mercado. Su desarrollo comenzó hace más de 15 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. En los últimos años se han concentrado mucho en la velocidad de proceso y en características demandadas en el mundo empresarial.

PostgreSQL se puede ejecutar en la gran mayoría de sistemas operativos existentes en la actualidad, entre ellos Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows. Las características más importantes y soportadas son:

- Una base de datos 100% ACID.[2]
- Llaves ajenas (foreign keys).
- Joins.
- Vistas.
- Disparadores (triggers).
- Reglas (Rules).
- Funciones/procedimientos almacenados (*stored procedures*) en numerosos lenguajes de programación, entre otros PL/pgSQL (similar al PL/SQL de oracle).
- Numerosos tipos de datos, posibilidades de definir nuevos tipos.
- Soporta el almacenamiento de objetos binarios grandes, por ejemplo: gráficos, videos y sonido.
- Herencia de tablas.
- PITR - *point in time recovery.*
- Tablespaces.
- Replicación asíncrona.
- *Nested transactions* (savepoints).
- *Two-phase commit.*
- Copias de seguridad en caliente (Online/hot backups).
- Unicode.
- Juegos de caracteres internacionales.
- *Multi-Version Concurrency* Control (MVCC).
- Acceso encriptado vía SSL.

[2] En bases de datos se denomina ACID a un conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción.

- SQL92/SQL99.
- APIs para programar en C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, PHP y muchos otros lenguajes.
- Completa documentación.

Otra característica a tener en cuenta es lo bien que PostgreSQL funciona con grandes cantidades de datos y una alta concurrencia, con muchos usuarios accediendo a la vez al sistema. [18]

1.8.6.2 Base de datos de referencias.

Una de las tareas más importantes de cualquier evaluación de los sistemas biométricos es la recopilación de datos. Las compañas que se realizan en el mundo han creado una multi-base de datos, las cuales contiene datos biométricos diferentes almacenados en dependencia del tipo de modalidad a evaluar (Huella digital, firma, rostro, etc.).

1.9 Conclusiones

Durante la realización del presente capítulo se hizo un estudio de las tendencias actuales de los procesos de prueba de algoritmos biométricos, se analizaron las principales campañas de evaluación a los sistemas biométricos en sus diferentes modalidades, viendo cómo se han ido incorporando cada año más 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. Además se estudiaron los lenguajes, herramientas, metodología y tecnologías a utilizar y se decidió utilizar como metodología de desarrollo de *software* RUP, para el control y planificación de este trabajo; atendiendo a sus características y a las facilidades que aporta, y para el modelado el *Rational Rose Enterprise* y UModel 2007 de Altova. Como lenguaje de programación C#3.0, utilizándose para la implementación Microsoft *Visual Studio* 2008 y el gestor de bases de datos escogido fue PostgreSQL.

Capítulo 2 Características del sistema.

2.1 Introducción.

En el presente capítulo se realiza una descripción de la propuesta de solución de la investigación. Para dar cumplimiento a ello se identifican los procesos del negocio relacionados con el objeto de estudio. Se identifican los requisitos funcionales y no funcionales que debe tener el sistema, los cuales son representados mediante diagramas de casos de uso, estos forman parte de los Diagramas de Caso de Uso del Sistema (DCUS), en ellos están presentes las relaciones de los actores con los casos de uso del sistema, y las secuencias de acciones con las que interactúan.

2.2 Descripción de los procesos del negocio propuesto.

La solución se fundamenta en un diseño de fácil entendimiento para el usuario.
La aplicación debe ser capaz a partir de un ejecutable en consola (el cual contiene la implementación del algoritmo a probar) de desarrollar los procesos de pruebas necesarios. Al cargar el ejecutable una vez llenados los datos solicitados al usuario, seleccionado el tipo de biometría y la base de datos de referencia, el mismo genera distribuciones de puntajes[3] (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 (ERR). 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.

A partir de una breve descripción de los procesos del negocio expuestos, se concluye que el mismo tiene un bajo nivel de estructuración. Por lo que esta investigación se basará en un modelo de dominio, que permitirá mostrar al usuario los principales conceptos que se operan en el dominio de la aplicación en desarrollo. De esta forma

[3] El grado de similitud en la correspondencia biométrica se define como puntuación. Esta puntuación representa el éxito durante la extracción de las características de las muestras biométricas obtenidas.[19]

los usuarios utilizarán un vocabulario común, para lograr entender el contenido en que se enmarca la aplicación. 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 utilizarán en el modelo de dominio:

Probador de algoritmos, es el usuario que se encarga de realizar la prueba, usualmente es la misma persona que implementa el algoritmo que desea probar. La primera acción que se realiza para hacer la **Prueba** es ejecutar la **Aplicación Biométrica,** en la cual se contiene el algoritmo que se desea probar, para esta tarea utiliza una base de datos de referencia (**BDReferencia**), las plantillas biométricas almacenadas en esta son pasadas manualmente al algoritmo en dependencia del valor que desea obtener (distribuciones de puntaje cliente e implementación)**.** Las **Pruebas** realizadas se dividen en Categorías (Huella dactilar, rostro, iris, retina, palma de la mano, entre otros). Las bases de datos de referencia (**BDReferencia**), son diferentes muestras biométricas que son elegidas en dependencia del algoritmo que se va a probar.

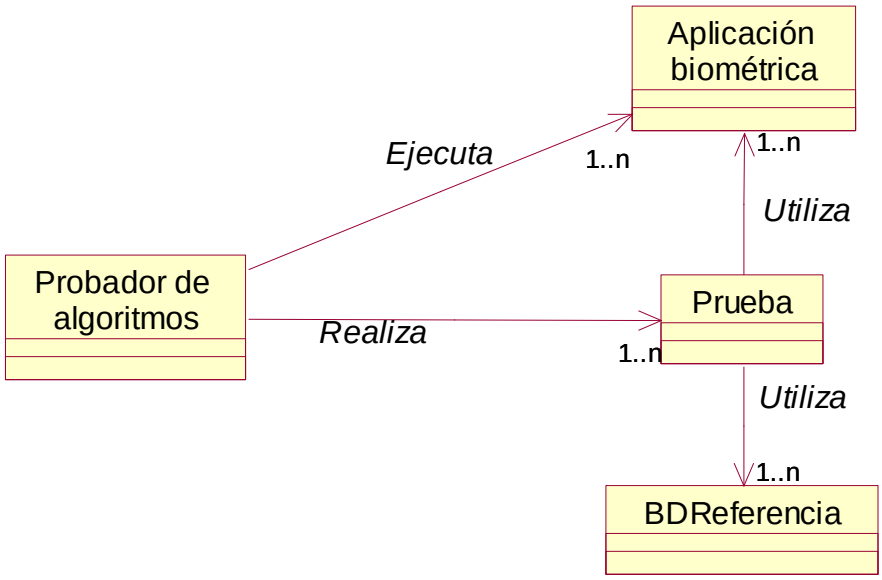

Figura 7 Modelo de Dominio.

2.4 Requisitos funcionales.

Los Requerimientos funcionales especifican acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física. [24]

RF 1 Cargar aplicación biométrica.

RF 1.1 Introducir nombres, apellidos y el nombre del algoritmo biométrico.

RF 1.2 Seleccionar la categoría del algoritmo que se desea probar (tipo de biometría, huellas, rostros iris, etcétera)**.**

RF 1.3 Seleccionar y cargar **la base de datos de referencia** la cual se utilizará para probar el algoritmo biométrico.

RF 1.4 Cargar la aplicación biométrica (es el fichero que contiene al algoritmo.)

RF 2 Realizar prueba.

RF 2.1 Cargar los **scores** generados.

RF 2.2 Obtener los valores del FAR, FRR y EER.

RF 2.3 Graficar los resultados.

RF 3 Realizar reporte de la prueba ejecutada.

RF 3.1 Almacenar los resultados en la base de datos.

RF 3.2 Visualizar resultados posteriormente.

2.5 Requisitos no funcionales.

Los requerimientos 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. [24]

RNF 1.Apariencia o interfaz externa

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

RNF 2.Usabilidad

- La plataforma 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 algoritmos biométricos. Estos tiempos pueden ser comparados, ya que la plataforma debe mostrar el tiempo que demora en realizar la prueba. Los tiempos varían en dependencia de la cantidad de muestras que contengan cada una de las base de datos de referencia.

RNF 4.Portabilidad

- La plataforma debe ser capaz de funcionar sobre los Sistemas Operativos Windows XP y Windows 7, en ambos casos en todas sus versiones.

RNF 5.*Hardware*

- El sistema requiere como mínimo 512 *Mega Bytes* (Mb) de *random-access memory (*RAM), un microprocesador con velocidad de 1.0 *gigahercio* (GHz), y 32Mb de memoria de video.

2.6 Modelo de Casos 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 **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. Un **actor** representa un conjunto coherente de roles que juegan los usuarios de los Casos de Uso cuando interactúan con estos. Los actores pueden ser personas (roles que desempeñan las personas), aparatos eléctricos o mecánicos, y otros sistemas de cómputo. En este caso con el sistema interactúan dos actores (Probador y Cliente).

Tabla 2.1 Actores del sistema.

Actor del sistema	Justificación
Probador	Es la persona encargada de realizarle las pruebas a los algoritmos y crear los reportes que serán mostrados.

Tabla 2.2 Resumen del Caso de Uso 1.

Caso de uso:	Probar aplicación biométrica
Actor :	Probador
Descripción:	Cuando el probador procede a probar la aplicación biométrica, debe cargar las distribuciones de puntajes que corresponden a los clientes y los impostores. Y de este modo obtener los resultados de la misma.
Referencia:	RF2.1, RF2.2, RF2.3

Ver Anexo 1.

Tabla 2.3 Resumen del Caso de Uso 2.

Caso de uso:	Realizar reporte
Actor :	Probador

Descripción:	El Probador almacena el resultado de la prueba en la base de datos obteniendo así lo que llamamos Reportes, los cuales pueden ser visualizados posteriormente filtrándolos por el tipo de biometría o simplemente mostrándolos todos.
Referencia:	RF3.1, RF3.2

Ver Anexo 2.

Tabla 2.4 Resumen del Caso de Uso 3.

Caso de uso:	Cargar aplicación biométrica
Actor :	Probador
Descripción:	El probador introduce el nombre, los apellidos, el nombre de la aplicación biométrica que desea someter a la prueba, la categoría y la base de datos que desea utilizar. Una vez obtenidos todos estos datos se procede a comenzar el proceso de obtención de distribuciones de puntajes de los clientes y los impostores.
Referencia:	RF 1.1, RF 1.2, RF 1.3, RF 1.4

Ver Anexo 3.

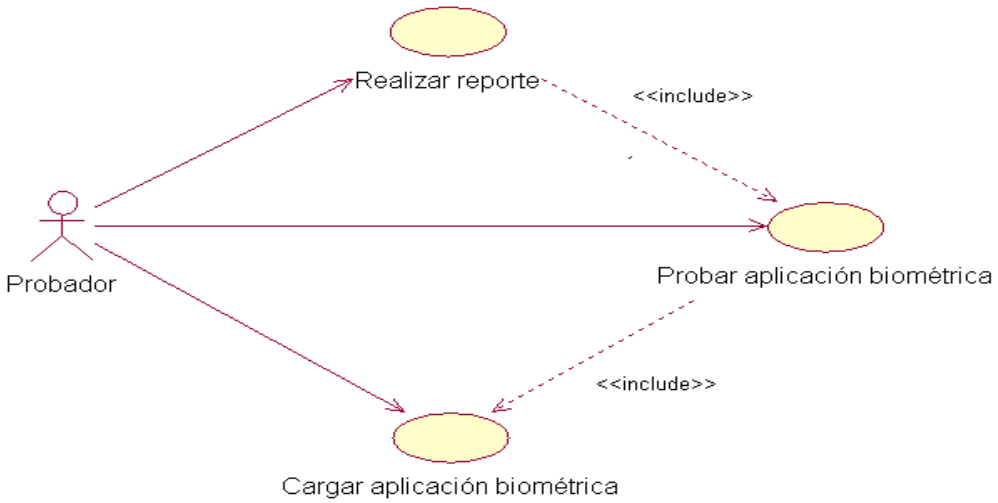

Figura 8 Diagrama de casos de uso del sistema.

2.7 Conclusiones.

Durante el desarrollo de este capítulo se comenzó la realización de la propuesta de solución para la aplicación que se desea implementar. Se realizó el modelado del negocio mediante un modelo de dominio. Además se expusieron los requisitos funcionales y no funcionales, así como la descripción de los casos de usos del sistema. Con el desarrollo de este capítulo es posible comenzar a realizar el diseño e implementación del sistema.

Capítulo 3. Construcción y elaboración del sistema.

3.1 Introducción.

Durante el desarrollo del presente capítulo se realiza el diseño de la propuesta de solución, se definen los principios de diseño gráfico y los estándares de interfaz de la aplicación. Se expondrán además el diagrama de despliegue y el diagrama de implementación. Ellos son sumamente importantes dentro del flujo de trabajo de implementación, pues ambos conforman lo que se conoce como modelo de implementación. Además se describen y explican los conceptos relacionados con dichos diagramas.

3.2 Modelo de Análisis.

En la construcción del modelo de análisis se identifican las clases que describen la realización de los Casos de Uso, los atributos y las relaciones entre ellas. Con esta información se construye el Diagrama de clases del análisis, que por lo general se descompone para agrupar las clases en paquetes. Esta descomposición tiene impacto comúnmente en el diseño e implementación de la solución. El Modelo de Análisis es el resultado de la actividad de analizar los Casos de Uso y su realización suaviza la transición al Diseño y se utiliza para tener una visión general de la propuesta de sistema [25].

3.2.1 Diagrama de Clases de Análisis.

Las clases de análisis se centran 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:

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.

Diagramas de clases del análisis para el actor Probador:

Figura 9 Diagrama de clases del análisis del caso de uso: Probar aplicación biométrica.

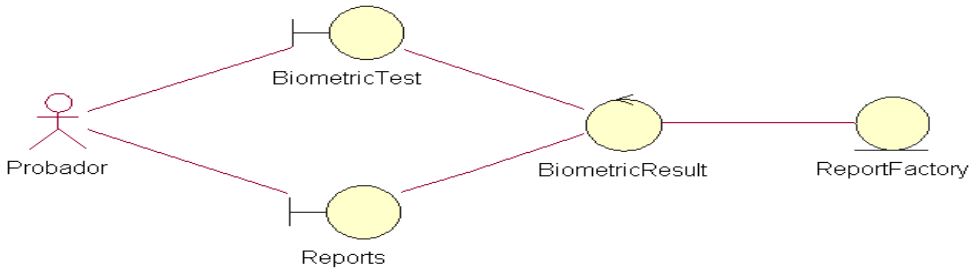

Figura 10 Diagrama de clases del análisis del caso de uso: Realizar reporte.

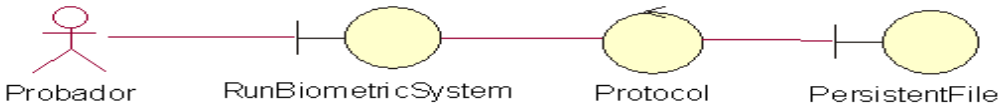

Figura 11 Diagrama de clases del análisis del caso de uso: Cargar aplicación biométrica.

3.3 Modelo de 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. [26]

3.3.1 Diagrama de Clases del Diseño.

Las clases de diseño se especifican utilizando la sintaxis del lenguaje de programación elegido y tienen correspondencia directa con los métodos en la implementación. Un diagrama de clases de diseño es una representación más concreta que el diagrama de clases del análisis, representa la parte estática del sistema, las clases y sus relaciones.

A continuación se muestran el diagrama de clases del diseño por paquetes y finalmente el diseño de diagrama de clases, de manera general.

Paquete 1. Diagrama de clases de *Performance Evaluation*.

Paquete 2. Diagrama de clases de DATAACCESS.

Paquete 3. Component.

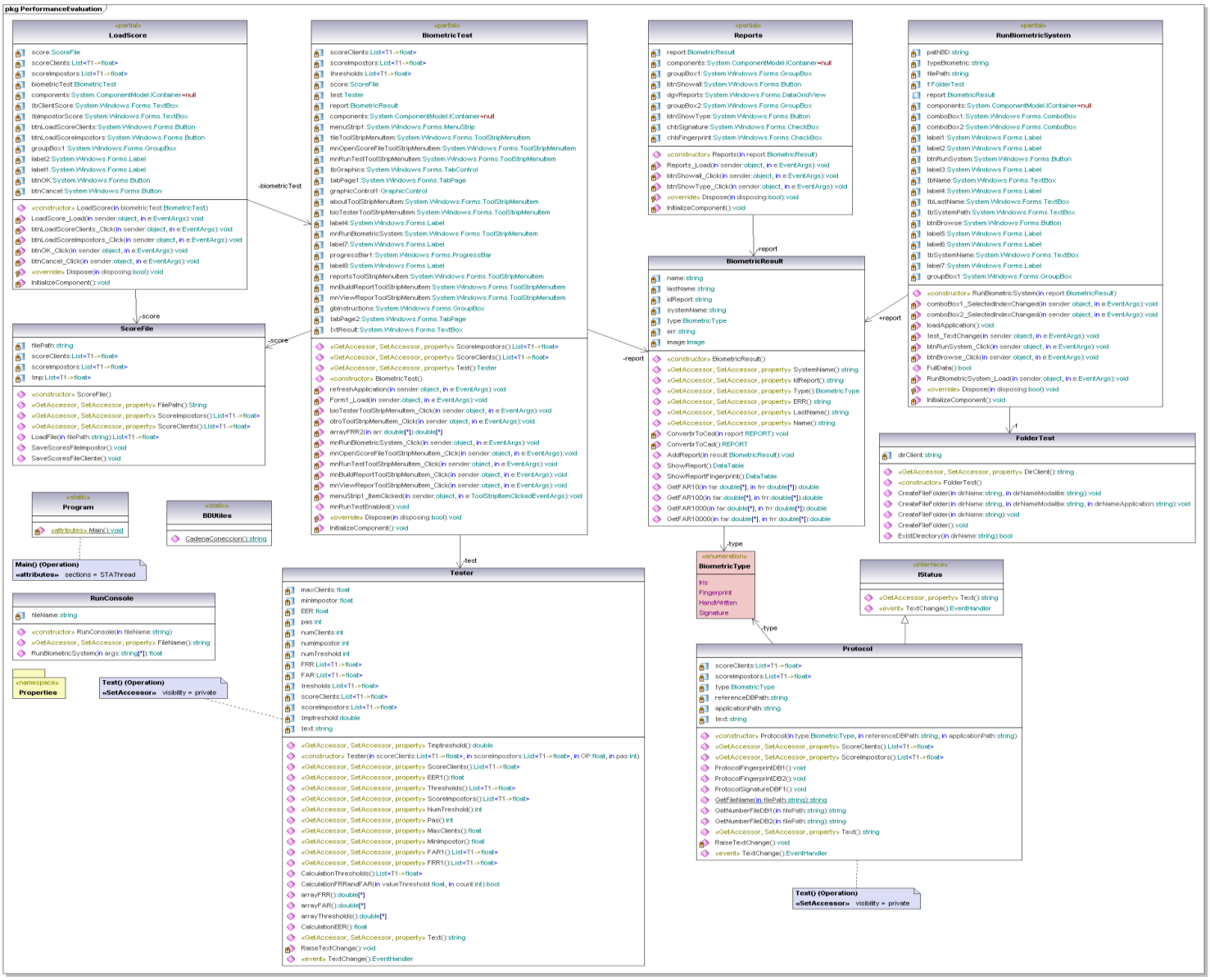

Figura 12 Diagrama de clases del diseño del paquete Performance Evaluation.

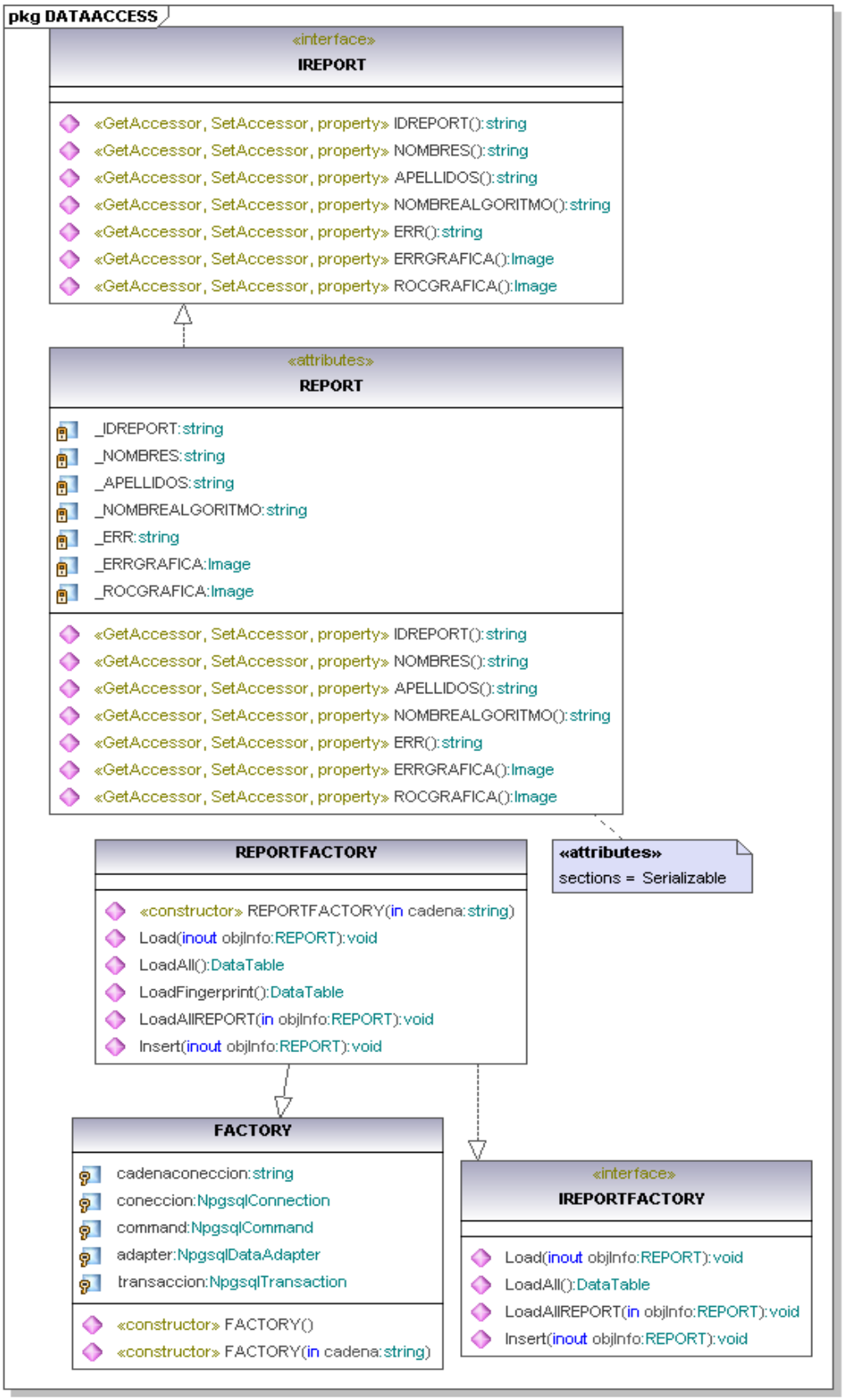

Figura 13 Diagrama de clases del diseño del paquete DATAACCESS.

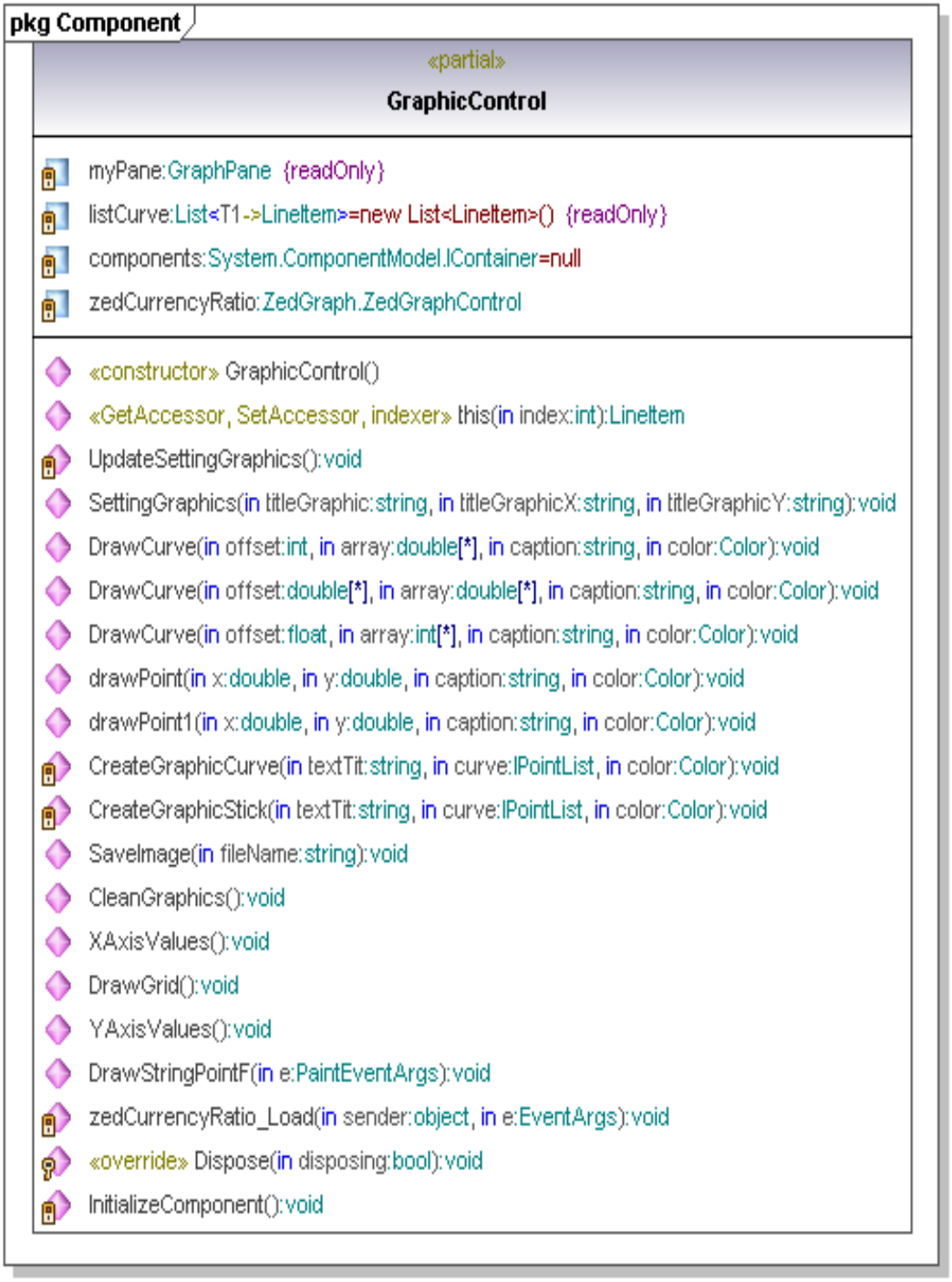

Figura 14 Diagrama de clases del diseño del paquete Component.

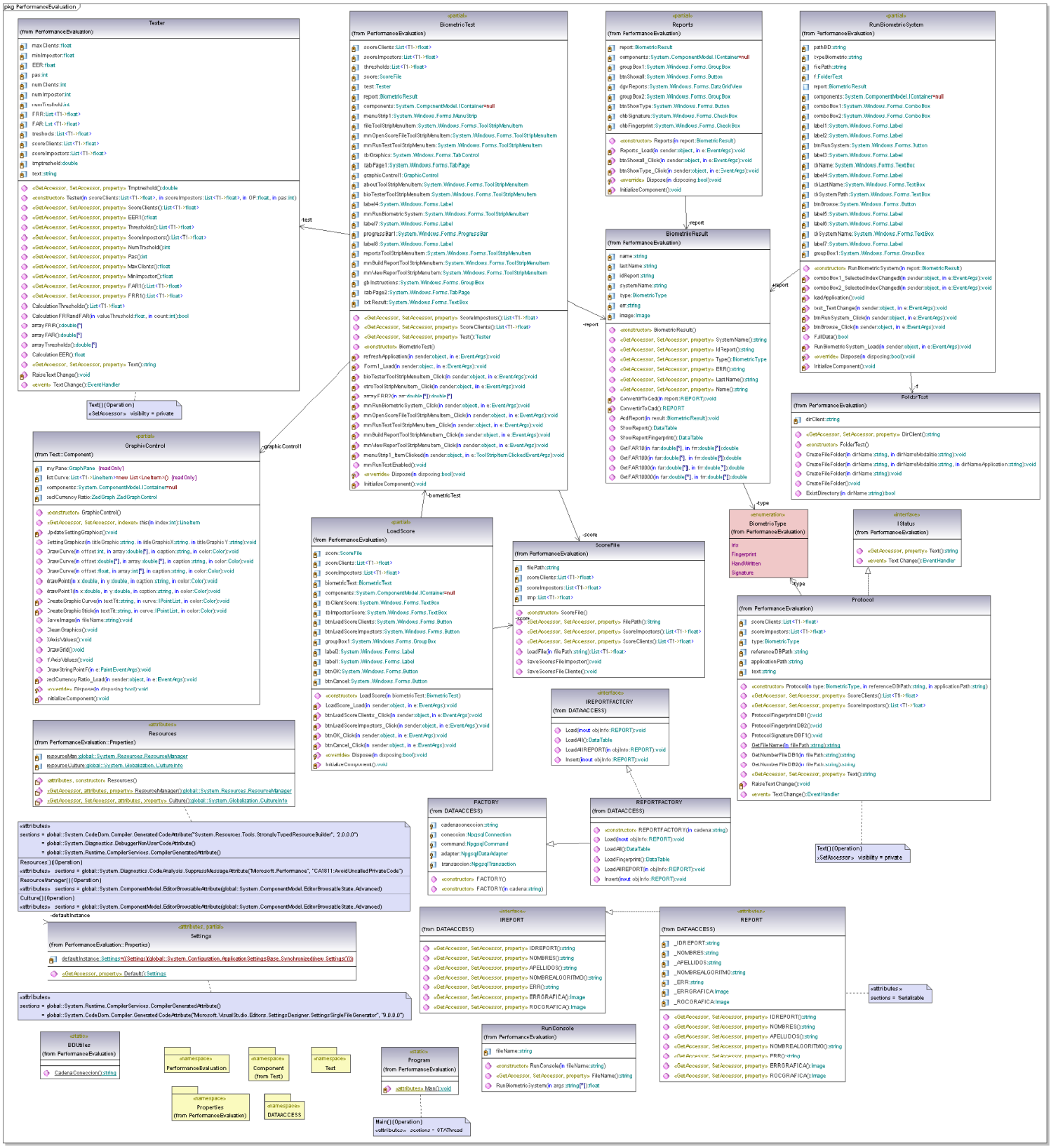

Figura 15 Diagramas de clases del diseño del paquete *Performance Evaluation* y de todos sus subpaquetes.

Tabla 3.1 Descripción de la clase BiometricResult.

Nombre: **BiometricResult**	
Tipo de Clase:	
Atributos	Tipo
Name	**string**
lastName	**string**
idReport	**string**
systemName	**string**
Type	**BiometricType**
Err	**string**
Image	**Image**
Responsabilidades	
Nombre	Descripción
BiometricResult()	**Constructor sin parámetros**
SystemName():string	**GetAccessor, SetAccessor, property**
IdReport():string	**GetAccessor, SetAccessor, property**
Type():BiometricType	**GetAccessor, SetAccessor, property**
ERR():string	**GetAccessor, SetAccessor, property**
LastName():string	**GetAccessor, SetAccessor, property**
Name():string	**GetAccessor, SetAccessor, property**
ConvertirToCed(in report:REPORT):void	
ConvertirToCad():REPORT	

AddReport(in result:BiometricResult):void	**Agregar un reporte a la base de datos**
ShowReport():DataTable	**Mostrar reporte de pruebas general**
ShowReportFingerprint():DataTable	**Mostrar reporte de pruebas con base de datos de huella**
ShowReportSignature():DataTable	**Mostrar reporte de pruebas con base de datos de firma**
GetFAR10(in far:double[*], in frr:double[*]):double	**Obtener FRR para un FAR de 10**
GetFAR100(in far:double[*], in frr:double[*]):double	**Obtener FRR para un FAR de 100**
GetFAR1000(in far:double[*], in frr:double[*]):double	**Obtener FRR para un FAR de 1000**
GetFAR10000(in far:double[*], in frr:double[*]):double	**Obtener FRR para un FAR de 10000**

Tabla 3.2 Descripción de la clase BiometricType.

Nombre: **BiometricType**	
Tipo de Clase: Enumerativo	
Atributos	Tipo
Iris	**enum**
Fingerprint	**enum**
HandWritten	**enum**
systemName	**enum**
Signatura	**enum**

Tabla 3.3 Descripción de la clase FolderTest.

Nombre: **FolderTest**	
Tipo de Clase:	
Atributos	Tipo
dirClient	**string**

Responsabilidades	
Nombre	Descripción
FolderTest()	**Constructor sin parámetros**
DirClient():string	**GetAccessor, SetAccessor, property**
CreateFileFolder(in dirName:string, in dirNameModalitie:string):void	**Crear directorio**
CreateFileFolder(in dirName:string, in dirNameModalitie:string, in dirNameApplication:string):void	**Crear directorio**
CreateFileFolder(in dirName:string):void	**Crear directorio**
CreateFileFolder():void	**Crear directorio**
ExistDirectory(in dirName:string):bool	**Si existe un directorio**

Tabla 3.4 Descripción de la clase Protocol.

Nombre: **Protocol**	
Tipo de Clase:	
Atributos	Tipo
scoreClients	**List<T1->float>**
scoreImpostors	**List<T1->float>**
Type	**BiometricType**
referenceDBPath	**string**
applicationPath	**string**
Text	**string**
Responsabilidades	
Nombre	Descripción
Protocol(in type:BiometricType, in referenceDBPath:string, in applicationPath:string)	**Constructor con parámetros**
ScoreClients():List<T1->flota>	**GetAccessor, SetAccessor, property**

ScoreImpostors():List<T1->float>	**GetAccessor, SetAccessor, property**
ProtocolFingerprintDB1():void	**Protocolo para DB1 de Fingerprint**
ProtocolFingerprintInnovatrics ():void	**Protocolo para DB2 de Fingerprint**
ProtocolSignatureDBF1():void	**Protocolo para DB1 de Signature**
GetFileName(in filePath:string):string	**Obtener el nombre del fichero de una ruta seleccionada**
GetNumberFileDB1(in filePath:string):string	**Obtener el parte del nombre de una imagen de la BD1**
GetNumberFileDB2(in filePath:string):string	**Obtener el parte del nombre de una imagen de la BD2**
Text():string	**GetAccessor, SetAccessor, property**
RaiseTextChange():void	**Controla el cambio de texto**
TextChange():EventHandler	**Evento**

Ver Anexo 4.

3.4 Diagramas de secuencia.

Es el diagrama que muestra las interacciones entre los objetos organizados en una secuencia temporal. En particular muestra los objetos participantes en la interacción y la secuencia de mensajes intercambiados. Dentro del conjunto de mensajes representados dispuestos en una secuencia temporal, cada rol en la secuencia se muestra como una línea de vida, es decir, una línea vertical que representa el rol durante cierto plazo de tiempo, con la interacción completa. Los mensajes se muestran como flechas entre líneas de vida. Un diagrama de secuencia puede mostrar un escenario, es decir, una historia individual de transacción. Un uso de un diagrama de secuencia es mostrar la secuencia del comportamiento de un caso de uso.

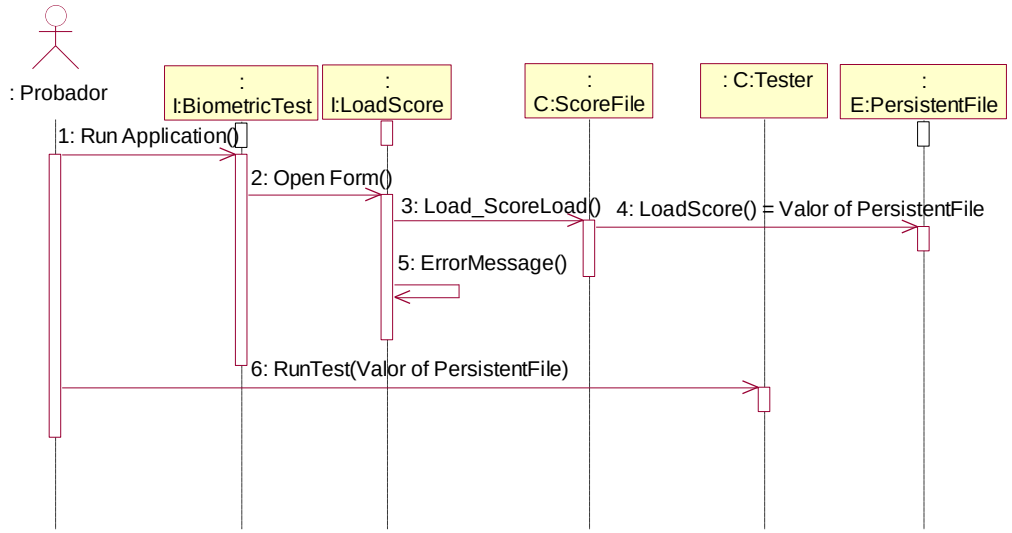

Figura 16 Diagrama de secuencia del caso de uso: Probar aplicación biométrica.

Ver Anexo 5.

3.5 Principios de diseño de interfaz.

La interfaz gráfica del usuario es el medio por el cual este interactúa con el sistema, por lo que su diseño debe ser amigable, consistente, necesita contener retroalimentación, minimizar las posibilidades de error y la memorización, debe proveer la recuperación de errores y que no requiera usuarios con un alto nivel informático.

Una aplicación con una interfaz bien diseñada, además de un buen diseño gráfico, debe tener una buena navegabilidad, usabilidad y distribución de los contenidos. Este trabajo utilizará en el diseño los principios antes mencionados.

3.5.1 Estándares de interfaz de la aplicación.

Las interfaces permiten la comunicación y el intercambio de información entre el usuario y el sistema. El aplicación que se presenta cuenta con una interfaz principal, en esta se encuentra un menú donde aparecen las funcionalidades fundamentales que esta debe ser capaz de realizar. Cuenta con las instrucciones básicas para el uso del sistema, además de 4 gráficas donde son reflejados los valores del FAR, el FRR, el EER y la curva ROC; así como una pestaña donde se puede observar el resultado. En el borde inferior izquierdo se encuentra una barra de progreso expresada en por ciento, la cual muestra el proceso de procesamiento de los scores.

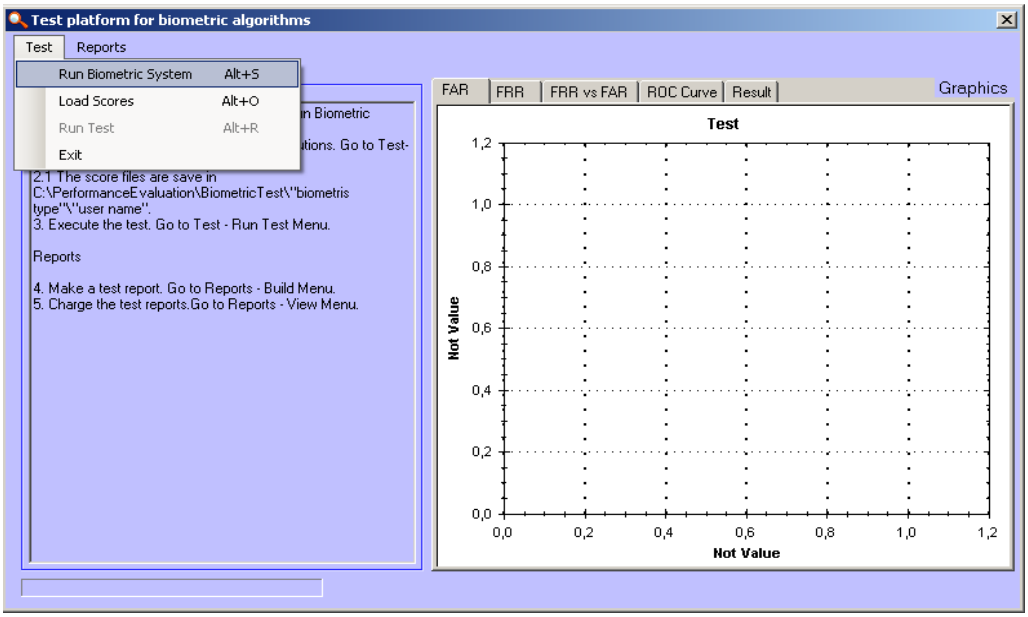

Figura 17 Formulario principal de la Plataforma de pruebas para algoritmos biométricos.

Para realizar la prueba, primeramente debemos seleccionar "**Test**"/ "**Run Biometric System**" en el menú principal.

Seguidamente se muestra la ventana para ingresar los datos necesarios para el primer paso de la prueba.

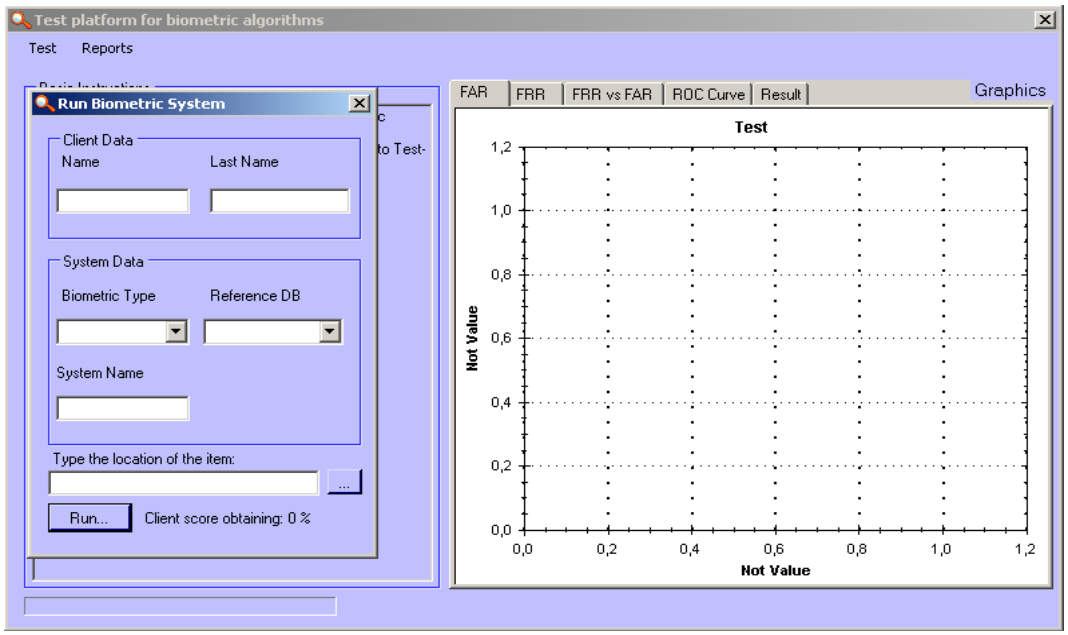

Figura 18 Formulario de Run Biometric System.

Los datos que son suministrados a la aplicación son validados, si detecta alguno erróneo, el sistema lanza una excepción informándolo.

Al terminar el proceso de ejecución de la aplicación biométrica la ventana se cierra automáticamente. Los scores de clientes e impostor han sido generados.

El paso siguiente es seleccionar "**Test**"/ "**Load Scores**" donde se buscan los scores generados.

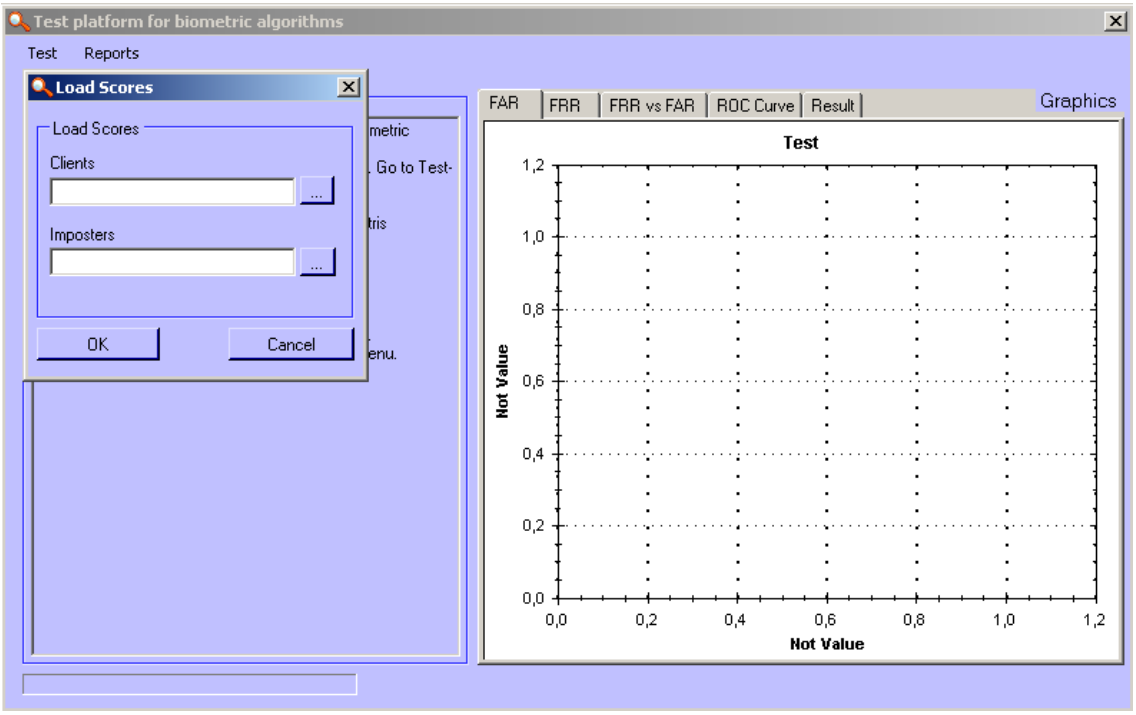

Figura 19 Formulario de Load Score.

Los scores son generados con extensión "**.txt**", si es cargado un archivo .txt incorrecto el sistema lanza una excepción indicándolo.

Una vez cargados los scores correctamente al sistema, se notará que se habilita el botón "**Test**"/ "**Run Test**", al seleccionarlo comienza a ejecutarse el procesamiento de los mismos.

Los valores obtenidos deben ser almacenados en reportes. Para crear los mismo se debe ir al menú "**Reports**"/ "**Build**".

Los reportes pueden ser mostrados filtrándolos por categoría biométrica o todos en general.

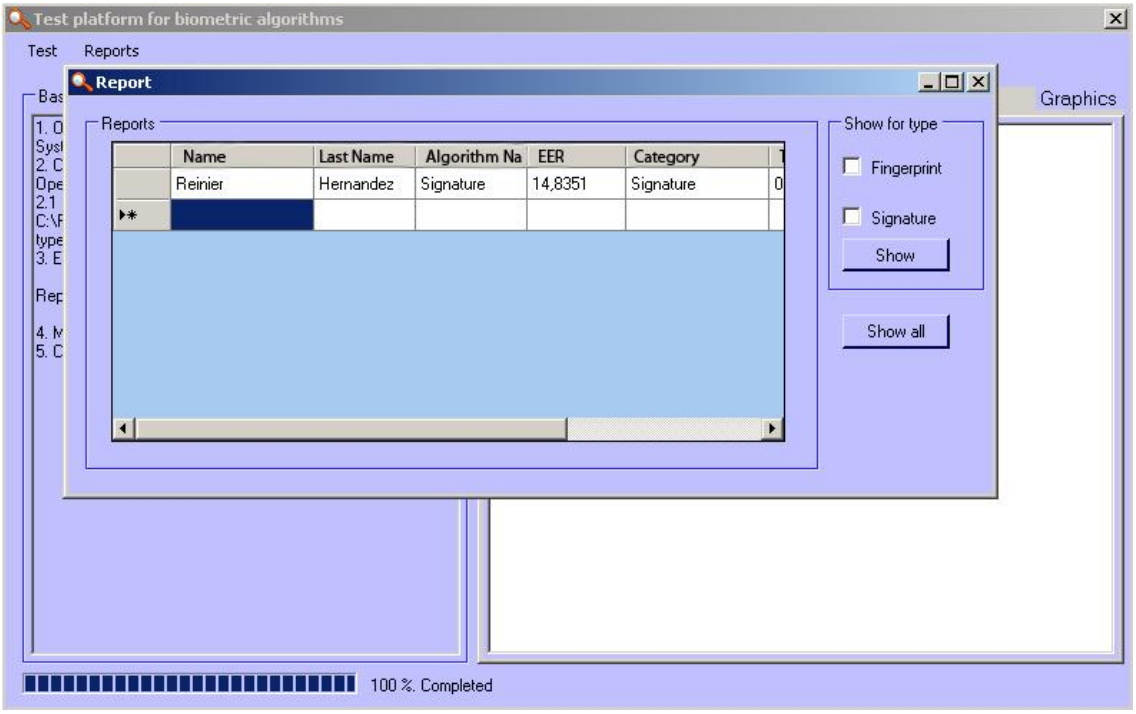

Figura 20 Reporte.

3.6 Descripción de las base de datos de referencia.

- ✓ ***Signature***

DBF_1: Cuenta con 28 imágenes (.tif) de firmas. Para conformarla participaron 14 clientes, a los cuales en una primera sesión se le tomó una muestra y en una segunda sesión se le tomó otra.

DBF_3: Para la obtención de esta base datos de referencia, se desarrolló una herramienta que accede a la "Base de Datos de referencia para la Prueba de Sistemas de Identificación de personas a través de Firmas Manuscritas". Almacenando las muestras de las firmas con un formato (.jpg) en una carpeta seleccionada.

Para su conformación se tomaron 3 muestras de firmas de 15 usuarios diferentes.

- ✓ ***Fingerprint***

Innovatrics: Para la elaboración de la base de datos contribuyeron 30 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 360.

DB_1: Para la elaboración de la base de datos contribuyeron dos individuos. En una primera sesión se obtuvo una huella por cada dedo de cada individuo y en la segunda se obtuvieron siete muestras más de cada dedo por cada individuo, para un total de 160.

3.7 Protocolo para realizar la prueba.

- ✓ ***Fingerprint***

Las bases de datos utilizadas para esta evaluación están descritas en el epígrafe 3.6.

Para **Innovatrics:** Debe proporcionar un ejecutable de consola que reciba por parámetros dos plantillas de huellas digitales y estas sean comparadas entre sí, debe devolver el valor de la similitud en base a 100. Si la huella no puede ser comparada por algún motivo debe devolver -1.

Para **DB_1:** Debe proporcionar un ejecutable de consola que reciba por parámetros dos imágenes de huellas digitales y estas sean comparadas entre sí, debe devolver el valor de la similitud en base a 100. Si la huella no puede ser comparada por algún motivo debe devolver -1.

- ✓ ***Signature***

Para **DBF_1** y **DBF_3:** Debe proporcionar un ejecutable de consola que reciba por parámetros dos imágenes de firmas manuscritas digitales y estas sean comparadas entre sí, debe devolver el valor de la similitud en base a 100. Si la huella no puede ser comparada por algún motivo debe devolver -1.

¿Cómo se realiza la obtención de scores a partir del ejecutable de consola?

Para obtener las **Scores de clientes** se compara la muestra tomada en la primera sesión con las muestras de la segunda sesión del mismo individuo

Para obtener las **Scores de impostores** comparan entre sí todas las muestras tomadas de la primera sesión.

3.8 Arquitectura.

Para la definición de la arquitectura de esta propuesta se pensó en el Modelo N Capas el cual utiliza como uno de sus principios, el desarrollo de aplicaciones basadas en componentes. Este tipo de arquitectura presenta varios beneficios:

- ✓ Aplicaciones más robustas debido al encapsulamiento.
- ✓ Mantenimiento y soporte más sencillo (es más fácil cambiar un componente que una aplicación monolítica).
- ✓ Mayor flexibilidad (se le pueden añadir más componentes que aumenten las funcionalidades del sistema).
- ✓ Soluciones altamente flexibles y reutilizables.
- ✓ Alta escalabilidad[4].

[4] Habilidad para extender el margen de operaciones sin perder calidad.

3.8.1 Modelo Vista Controlador (MVC).

La arquitectura del patrón MVC es un paradigma de programación bien conocido para el desarrollo de aplicaciones con interfaz gráfica. El patrón MVC es un patrón de arquitectura de *software* en el cual todo el proceso está dividido en tres capas, típicamente estas capas son el Modelo, la Vista y el Controlador.

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, etc.). 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 que escucha los cambios en la vista y los envía al modelo, este último retorna los datos a 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. [30]

3.9 Diseño de la base de datos.

Figura 21 Diagrama entidad-relación. Tabla dreporte.

Tabla 3.16 Descripción de la tabla dreporte.

Nombre: dreporte		
Descripción: Tabla donde son almacenados los reportes.		
Atributo	**Tipo**	**Descripción**
IDREPORTE	uuid	Identificador del reporte

NOMBRES	varchar	Nombres de la persona que está probando el algoritmo.
APELLIDOS	varchar	Apellidos de la persona que está probando el algoritmo.
CATEGORÍA	varchar	Categoría biométrica del algoritmo que se somete a la prueba.
NOMBRE_ALGORITMO	varchar	Nombre del algoritmo biométrico.
ERR	float	Valor de la Tasa de Error Igual.
TIEMPOPRUEBA	text	Tiempo en que se demora en realizar la prueba.
OPVALUE	char	Valor del punto OP.

3.10 Diagrama de despliegue.

El diagrama de despliegue representa cómo va a estar distribuido el sistema de forma física. Se utiliza una PC cliente en la cual va a estar situada la Plataforma de pruebas para algoritmos biométricos, la misma estará conectada a un servidor de base de datos de PostgreSQL a través de una conexión TCP/IP, con el objetivo de almacenar y visualizar los reportes.

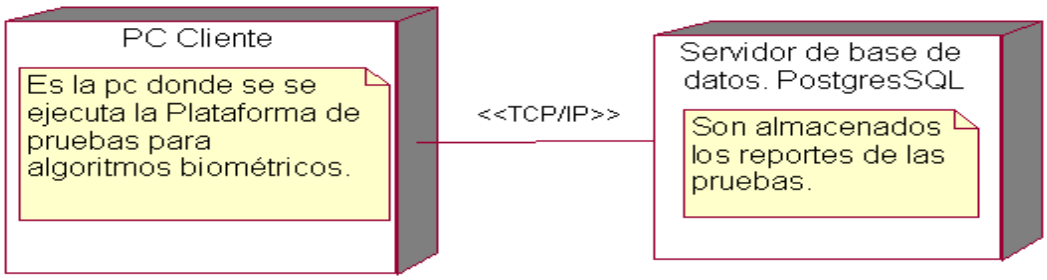

Figura 22 Diagrama de Despliegue.

3.11 Diagrama de componentes.

Los componentes representan todos los tipos de elementos *software* que entran en la fabricación de aplicaciones informáticas. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente. Un diagrama de componentes representa las dependencias entre componentes *software*, incluyendo componentes de código fuente, componentes del código binario y componentes ejecutables. Un módulo de *software* se puede representar como componente. Algunos componentes existen en tiempo de compilación, en tiempo de enlace, en tiempo de ejecución y otros en varias de éstas. Los mimos tienen dos características: empaquetan el código que implementa la funcionalidad de un sistema y también empaqueta algunas de sus propias instancias de objetos que constituyen el estado del sistema.

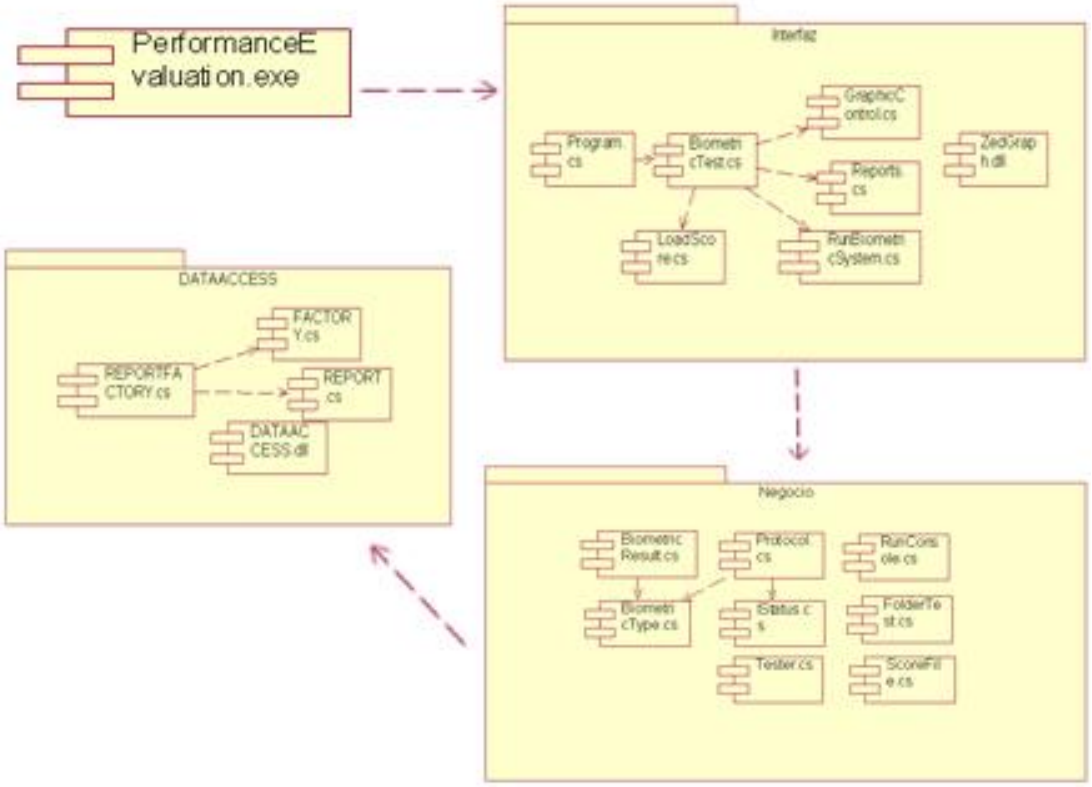

Figura 23 Diagrama de componentes.

3.12 Pruebas realizadas.

3.12.1 Descripción de variable.

Tabla 3.17 CU. Probar aplicación biométrica.

No	Nombre de Campo	Clasificación	Valor Nulo	Descripción
[1] Cargar Score Cliente	Score Cliente	Campo de Texto	No	Se debe introducir la dirección donde se encuentra ubicado el fichero de score de cliente en el disco duro.
[2] Cargar Score Impostor	Score Impostor	Campo de Texto	No	Se debe introducir la dirección donde se encuentra ubicado el fichero de score de impostor en el disco duro.

Tabla 3.18 CU. Realizar Reporte.

No	Nombre de Campo	Clasificación	Valor Nulo	Descripción
[1] Construir reporte.	Construir	Botón	No	Se crea el reporte una vez realizada la prueba.
[2] Visualizar reporte.	Visualizar	Tabla de datos	No	Se muestran los reportes en una tabla de datos.

Tabla 3.19 CU. Cargar aplicación biométrica.

No	Nombre de Campo	Clasificación	Valor Nulo	Descripción
[1] Introducir nombre del cliente.	nombre	Campo de Texto	No	Se introduce el nombre de la persona que va a realizarle la prueba al algoritmo biométrico.
[2] Introducir apellidos del cliente.	apellidos	Campo de Texto	No	Se introduce el apellido de la persona que va a realizarle la prueba al algoritmo biométrico.
[3] Seleccionar el tipo de	tipo de biometría	Lista desplegable	No	Se selecciona el tipo de biometría al que

biometría.				pertenece el algoritmo que va a ser sometido a la prueba.
[4] Seleccionar base de datos de referencia.	base de datos de referencia	Lista desplegable	No	Se selecciona la base de datos de referencia según el tipo de biometría seleccionado.
[5] Introducir nombre del algoritmo.	Nombre del algoritmo.	Campo de texto	No	Se introduce el nombre del algoritmo que va a ser sometido a prueba.
[6]Cargar aplicación biométrica.	Aplicación biométrica.	Campo de texto	No	Se carga la aplicación biométrica que se someterá a la prueba.

3.12.2 Matriz de Datos.

Tabla 3.20 CU. Probar aplicación biométrica.

Escenario	V1	V2	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Probar aplicación biométrica.	V	V	Se le realiza la prueba al algoritmo biométrico.	Satisfactorio.	1. Seleccionar el score de cliente. 2. Seleccionar el score de impostor. 3. Realizar la prueba al algoritmo biométrico.
	I	V	Índica que tiene algún campo vacío.	Satisfactorio.	
	V	I	Índica que tiene algún campo vacío.	Satisfactorio.	
	I	I	Índica que tiene algún campo vacío.	Satisfactorio.	

Leyenda.
V1. Cargar Score Cliente.
V2. Cargar Score Impostor.

Tabla 3.21 CU. Realizar Reporte.

Escenario	V1	V2	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Realizar Reporte	V	I	Se construye el reporte.	Satisfactorio.	1. Seleccionar la opción para construir el reporte
	I	V	No muestra el reporte sino no está	Satisfactorio.	

			construido.		2. Seleccionar la opción para visualizar los reportes.
	V	V	El reporte construido es visualizado.	Satisfactorio.	

Leyenda.
V1. Construir reporte.
V2. Visualizar reporte.

Ver Anexo 6.

3.12.3 Prueba al "Componente para la extracción y verificación de las características de la firma a través de un sistema de verificación de personas" propiedad del CISED.

Obteniéndose los siguientes resultados para DBF_1:

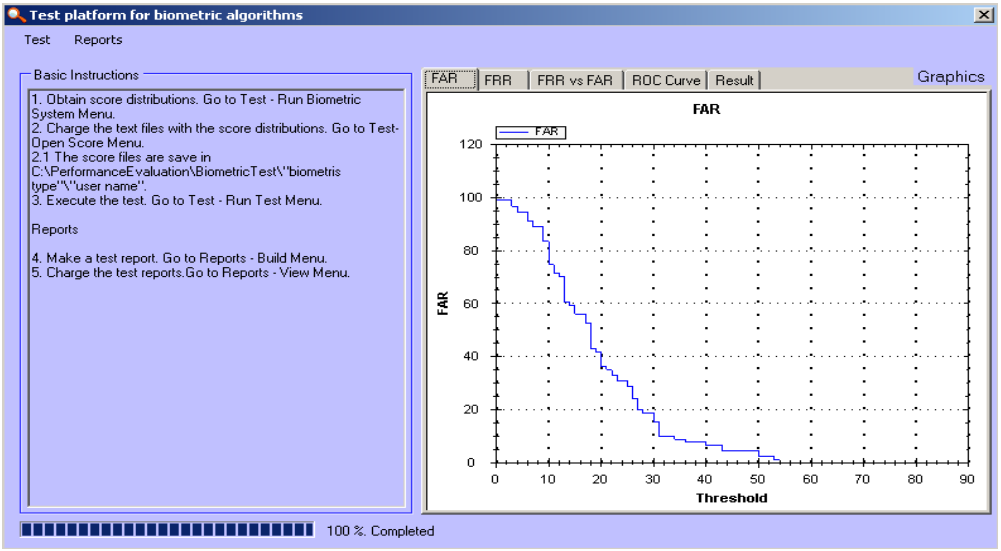

Figura 24 Gráfica del FAR.

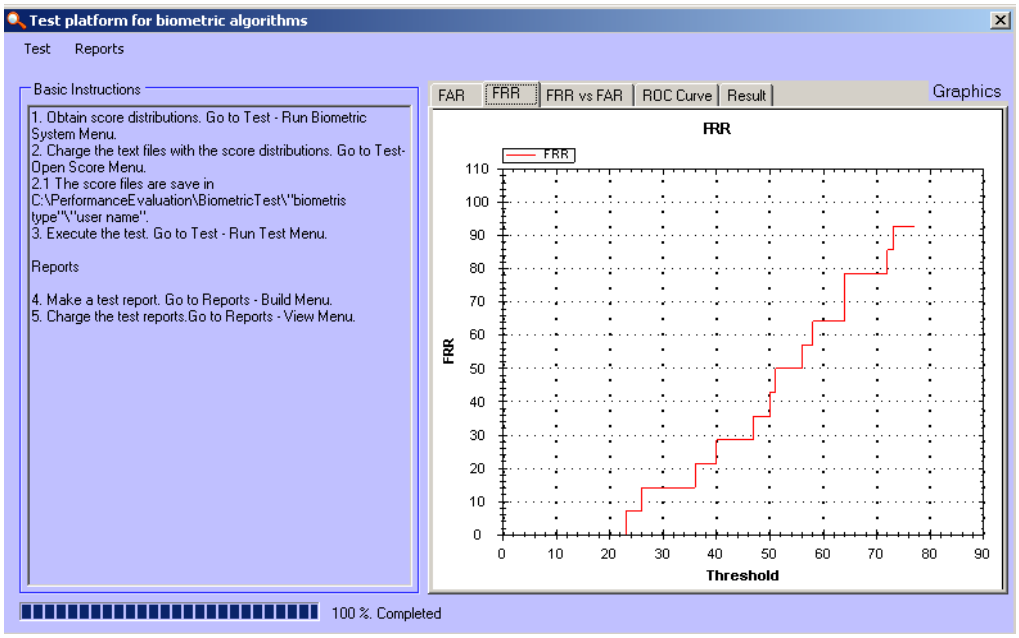

Figura 25 Gráfica del FRR.

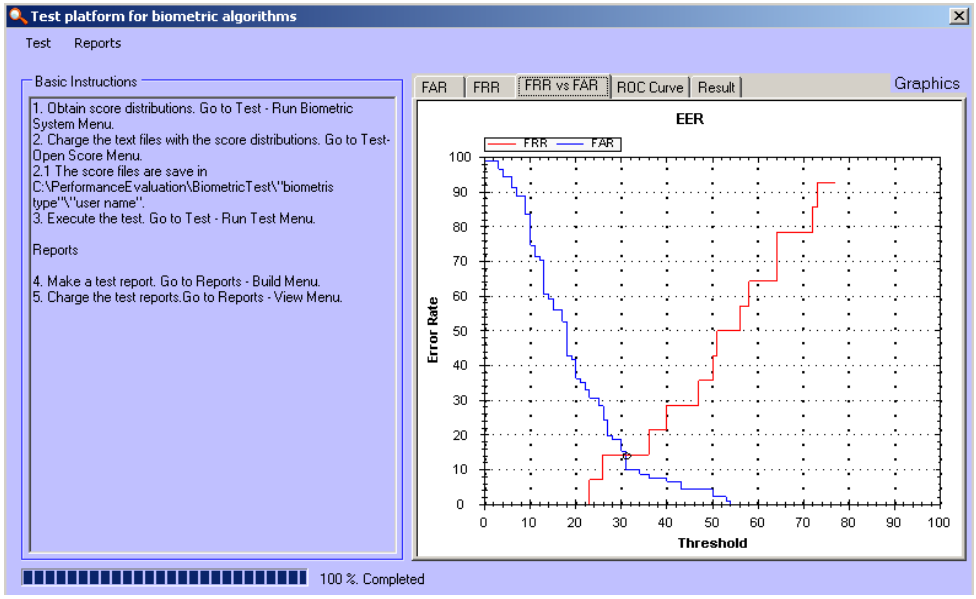

Figura 26 Gráfica del FAR vs FRR. EER.

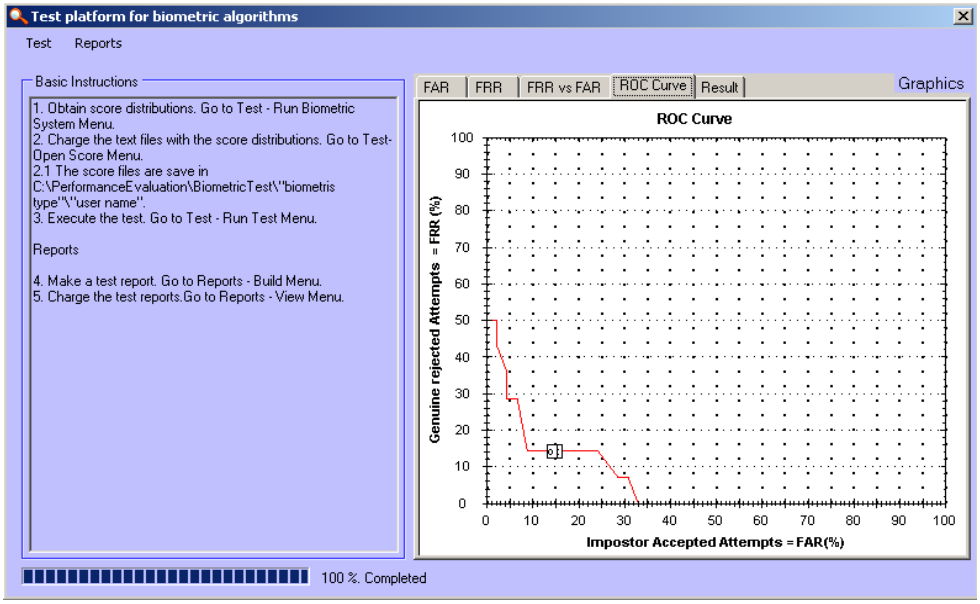

Figura 27 Gráfica del ROC.

El análisis del componente arrojó que para un valor fijo del FAR en 10% se obtiene un falso rechazo de 14,2857%. Este valor es utilizado para ser comparado con otros algoritmos de la misma modalidad en iguales condiciones de prueba.

Se obtuvo un valor de EER = 14,8351%

3.12.4 Prueba realizada a un componente desarrollado en CISED para la comparación de huellas dactilares.

Obteniéndose los siguientes resultados con Innovatrics:

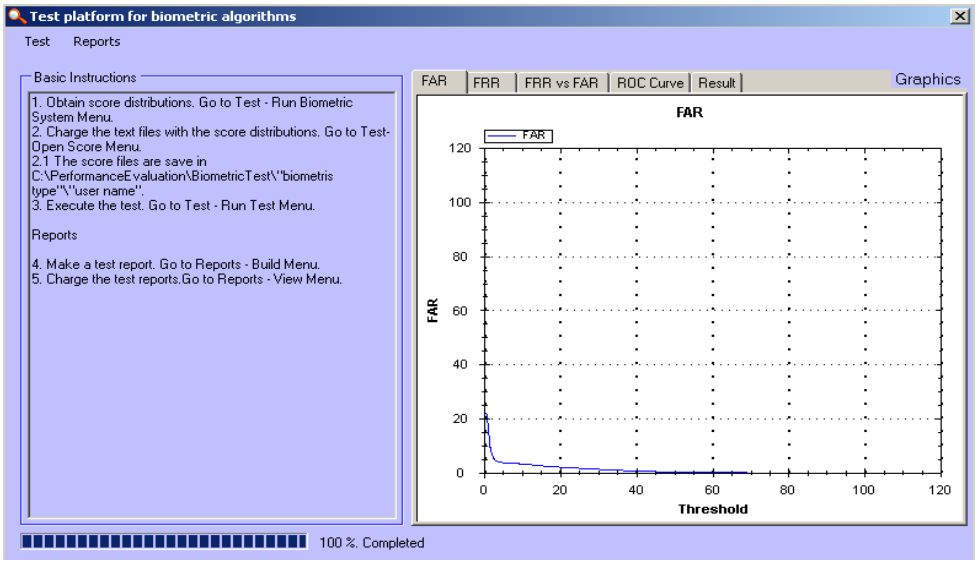

Figura 28 Gráfica del FAR.

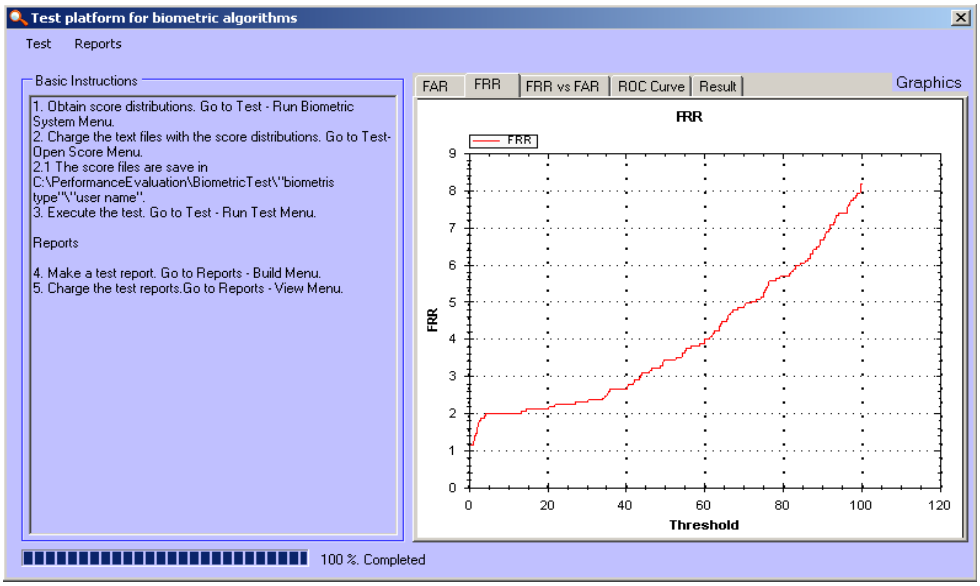

Figura 29 Gráfica del FRR.

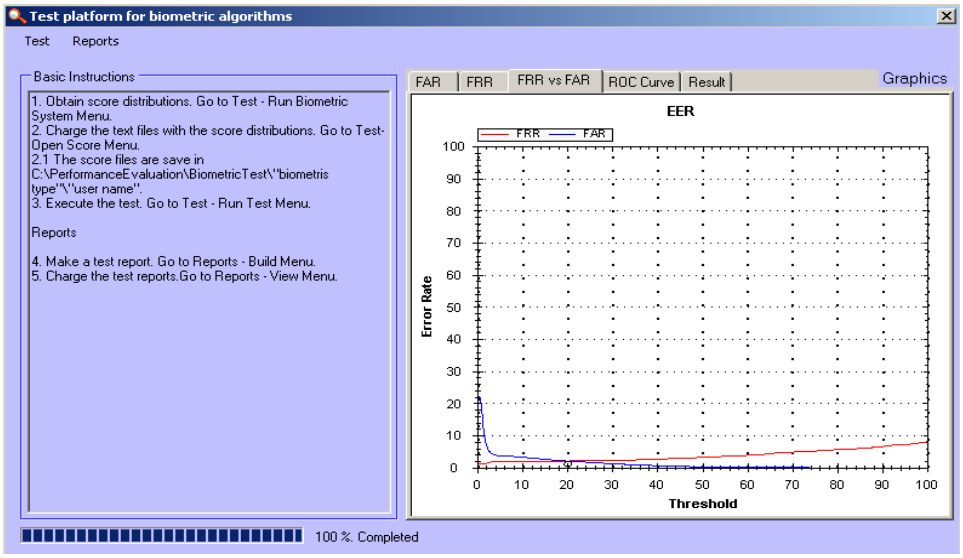

Figura 30 Gráfica del FAR vs FRR.

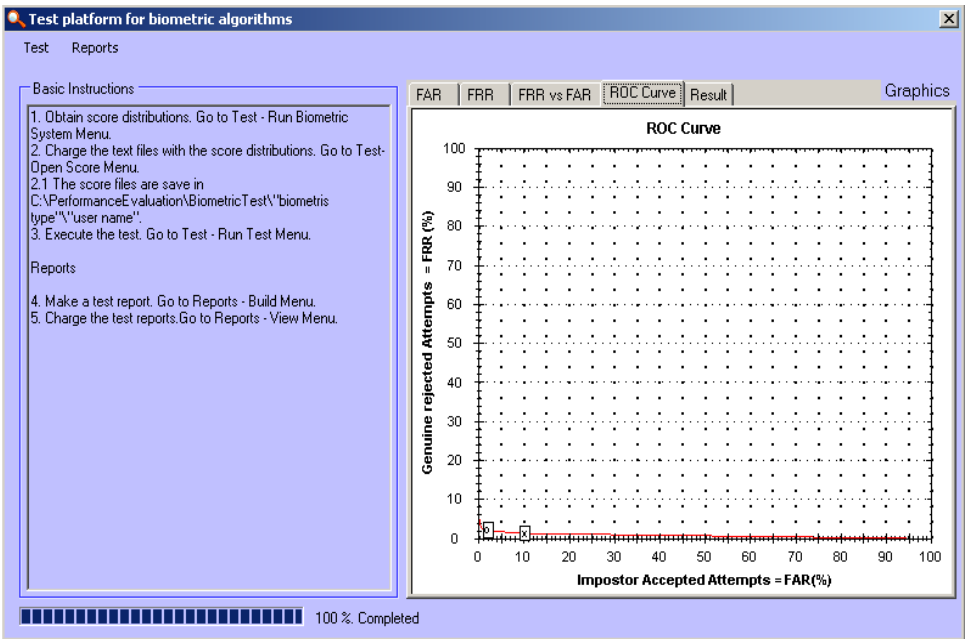

Figura 31 Gráfica del ROC.

El análisis del componente arrojó que para un valor fijo del FAR en 10% se obtiene un falso rechazo de 1,3939%. Este valor es utilizado para ser comparado con otros algoritmos de la misma modalidad en iguales condiciones de prueba.

Se obtuvo un valor de EER = 2,121%

3.13 Conclusiones.

Durante el presente capítulo se realizó el análisis y diseño de la solución propuesta. Para lograr un mayor entendimiento de la misma se expusieron los diagramas de clases del análisis y los diagramas de clases del diseño, los cuales fueron agrupados en diferentes paquetes; describiéndose detalladamente cada una de sus clases. Además se mostraron los diagramas de secuencia para cada uno de los casos de uso del sistema. Se dio una panorámica de cómo funcionada el sistema apoyándose en las interfaces con que cuenta el mismo. Se realizó la implementación del sistema y se mostraron los diagramas de despliegue y de componentes donde se evidencia cómo va a estar dividido el sistema. Además se hizo la descripción de la arquitectura utilizada en la implementación.

Se utilizó una base de datos de referencia generada en el CISED y se sometieron a pruebas dos componentes de comparación, uno de huellas dactilares y otro de firmas manuscritas, obteniéndose resultados satisfactorios.

Conclusiones.

Durante la presente investigación se realizó el estudio de la situación actual de los procesos de pruebas a algoritmos biométricos en el mundo. Notándose cómo las Campañas de Evaluación a algoritmos biométricos en todas sus modalidades se han venido incrementando desde sus inicios hasta la actualidad. Se obtuvo de las mismas un conocimiento profundo acerca de los procesos de pruebas a algoritmos biométricos en el mundo, demostrando la necesidad de automatizar dichos procesos en el CISED. Se analizaron las herramientas, metodología, tecnologías y lenguajes de programación a utilizar. Para un mejor entendimiento de los procesos que ocurren en el negocio se modeló el mismo a través de un Modelo del Dominio, donde se expresan de forma detallada los conceptos relacionados con el mismo. Se identificaron los requisitos del sistema a implementar, quedando plasmados en un Modelo de Casos de Uso del Sistema, siendo este, el punto de partida para comenzar el flujo de trabajo Análisis y Diseño. Con el mismo se definió qué deberá hacer el futuro sistema como resultado de analizar los requisitos funcionales identificados y cómo lograrlo, a partir de los requisitos no funcionales.

Como resultado, se obtuvieron los artefactos necesarios para dar inicio al siguiente flujo de trabajo propuesto por la metodología de desarrollo de *software* RUP, Implementación.

Los objetivos y las tareas planteadas al inicio de la investigación se cumplieron con éxito, obteniendo como producto la Plataforma de pruebas para algoritmos biométricos, luego de llevarse a cabo todas las investigaciones necesarias, así como todos los procesos involucrados en el análisis, el diseño y la implementación del mismo.

Se comprobó como el tiempo de prueba y la efectividad de los algoritmos biométricos mejoran notablemente.

Recomendaciones.

Se recomienda:

- Desarrollar el módulo de seguridad, con el objetivo de definir roles y tener un mejor control sobre los procesos que se desarrollen en el sistema, además de controlar el acceso no autorizado a la plataforma.
- Crear Campañas de Evaluación con el objetivo de impulsar el interés por la creación de aplicaciones biométricas y tener un Rankig de los competidores que participen.
- Introducir bases de datos para la realización de pruebas en la modalidad de iris, retina, palma de la mano y rostro humano.
- Desplegar el sistema en cualquier ordenador donde se desarrollen aplicaciones biométricas, en aras de ahorrar tiempo.

Referencias Citadas.

[1] http://www.biometria.gov.ar

[2] http://www.monografias.com/trabajos43/biometria/biometria.shtml

[3] Guide to Biometric Reference Systems and Performance Evaluation with a Foreword by Professor Anil K. Jain, Michigan State University, USA.

[4] Younis Elmadany Hamed. (Abril 2007). “Performance Evaluation of Biometric System”. Faculty of Electrical Engineering. Universiti Teknologi Malaysia.

[5] http://redyseguridad.fi-p.unam.mx/proyectos/biometria/estandares/estandar.html

[6] FERET database. URL: http://www.itl.nist.gov/iad/humanid/feret

[7] P. J. Phillips, W. T. Scruggs, A. J. OToole, P. J. Flynn, K. W. Bowyer, C. L. Schott, and M. Sharpe. FRVT 2006 and ICE 2006 Large-Scale Results (NISTIR 7408), March 2007.

[8] BioSecure Multimodal Evaluation Campaign 2007 (BMEC’2007).

http://www.biometrics.it-sudparis.eu/BMEC2007/

[9] D. M. Blackburn, J. M. Bone, and P. J. Phillips, "Facial Recognition Vendor Test 2000 Evaluation Report," February 2001. http://www.frvt.org

[10] P. J. Phillips, P. Grother, R. J. Micheals, D. M. Blackburn, E. Tabassi, and J. M. Bone, "Face Recognition Vendor Test 2002 Overview and Summary," March 2003. http://www.frvt.org

[11] P. Jonathon Phillips, W. Todd Scruggs, Alice J. OToole, Patrick J. Flynn, Kevin W. Bowyer, Cathy L. Schott, and Matthew Sharpe. FRVT 2006 and ICE 2006 Large-Scale Results (NISTIR 7408), March 2007.

[12] http://bias.csr.unibo.it/fvc2006/default.asp

[13] Website Oficial. http://biometrics.it-sudparis.eu/BMEC2007/

[14] Yoandro Hechevarría Toranzo, A. F. D. (2006). Sistema Automatizado para la Planificación Material y Financiera del MINFAR. Facultad de Ingeniería Industrial

Centro de Estudios de Ingeniería de Sistemas (CEIS). Ciudad de la Habana., Instituto Superior Politécnico "José Antonio Echeverría". 99.

[15] http://msdn.microsoft.com/en-us/library/bb308966.aspx

[16] Microsoft Visual C# .NET de Francisco Charte Ojeda.

[17] Acceso a Base de Datos con ADO.NET de Fernando Berzal Galiano.

[18] http://www.linux-es.org/node/536

[19] TECNOLOGÍAS BIOMÉTRICAS POR NIBALDO MUÑOZ V.

[20] http://www.griaulebiometrics.com/page/es/FVCFAQ

[21] Submission Instructions for the BioSecure *Signature* Evaluation Campaign. Version 1.0–26 November 2008.

[22] http://msdn2.microsoft.com/es-es/vstudio/default.aspx

[23] http://objetoide.blogspot.com/2008/02/ya-se-puede-adquirir-microsoft-visual.html

[24] Rational RequisitePro. Presentación de Power Point. Clase práctica 3. Ingeniería de *Software* I. Universidad de las Ciencias Informáticas.

[25] Larman, C. (2004). UML y Patrones. Introducción al análisis y diseño orientado a objetos. La Habana, Félix Varela.

[26] JACOBSON, Ivar, BOOCH, Grady and RUMBAUGH, James. El Proceso Unificado de Desarrollo de *Software*. 2000.

[27] Real Academia Española. Diccionario de la lengua española.

[28] Project N° IST-2002-507634 – BioSecure.

[29] Signal detection theory and ROC analysis in psychology and diagnostics: collected papers; Swets.

[30] http://www.scribd.com/doc/2539650/manual-microsoft-project-2003

[31] http://www.wordreference.com/definicion/prueba

[32] http://www.rpdp.net/mathdictionary/spanish/vmd/full/p/precision.htm

Bibliografía Consultada.

1. Universidad de las Ciencias Informáticas, Conferencias de Ingeniería de *Software* I, 2010.
2. Universidad de las Ciencias Informáticas, Conferencias de Ingeniería de *Software* II, 2010.
3. Jacobson, I.; Booch, G. y Rumbaugh, J.; “El Proceso Unificado de Desarrollo de *software*”. 2000. Addison-Wesley.
4. Anil Jain of Michigan State University East Lansing, Ruud Bolle and Sharath Pankanti of IBM T. J. Watson Research Center Yorktown Heights, NY; "INTRODUCTION TO BIOMETRICS”.
5. NIBALDO MUÑOZ V. “TECNOLOGÍAS BIOMETRICAS”.
6. P. Jonathon Phillips, Alvin Martin C.L. Wilson, Mark Przybocki of National Institute of Standards and Technology; "An Introduction to Evaluating Biometric Systems".
7. YOUNIS ELMADANY HAMED; "PERFORMANCE EVALUATION OF BIOMETRIC SYSTEM".
8. Introducción a la biometría; http://www.monografias.com/trabajos43/biometria/biometria.shtml
9. http://www.biometria.gov.ar

Anexos

Anexo 1.

Tabla 2.5 Descripción del Caso de Uso Probar aplicación biométrica.

Caso de uso:	Probar aplicación biométrica
Actores:	Probador
Resumen:	El caso de uso se inicia cuando el Probador selecciona la opción de probar aplicación biométrica, en ese punto selecciona las distribuciones de puntaje (scores) correspondientes a los clientes e impostores, y concluye cuando esta ha sido probada.
Precondiciones:	Tienen que estar generados los scores de cliente y de impostor de la aplicación biométrica que se desea someter a prueba.
Referencias:	RF2.1, RF2.2, RF2.3
Prioridad:	Crítico
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. El probador selecciona la opción cargar scores. 3. El probador ubica en la computadora donde se encuentran almacenados los scores de clientes e impostor, y preciosa el botón OK. 4. El probador selecciona la opción para comenzar el proceso de prueba.	2. El sistema muestra un formulario para cargar los scores de cliente e impostor. 5. El sistema procede a procesar los scores de clientes y de impostores cargados. 6. El sistema grafica los valores del FAR, FRR y EER.
Flujo alterno	
	3. En el caso de que no se haya generado ningún score, el usuario no puede continuar la prueba hasta que no cargue la aplicación biométrica al

	sistema. 4. Si no ha sido cargado ningún score, la opción para comenzar la prueba se encontrará deshabilitada hasta que sean cargados los scores de cliente e impostor respectivamente.
Poscondiciones:	Se obtuvo el resultado de la prueba.

Anexo 2.
Tabla 2.6 Descripción del Caso de Uso Realizar reporte.

Caso de uso:	Realizar reporte
Actores:	Probador
Resumen:	Una vez concluida la prueba se crea un reporte de la prueba el cual puede ser visualizado posteriormente.
Precondiciones:	La prueba tiene que haberse realizado.
Referencias:	RF3.1, RF3.2
Prioridad:	Crítico
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. El probador selecciona la opción de realizar reporte. 3. El probador puede visualizar los reportes almacenados en la base de datos. 5. El probador puede filtrar los reportes por categoría biométrica.	2. El sistema almacena el reporte en la base de datos. 4. El sistema muestra los reportes que han sido almacenados. 6. El sistema muestra los reportes que han sido solicitados por el usuario.
Flujo alterno	
	1. En el caso que no se haya realizado ninguna prueba, se mostrará un mensaje indicando que no se puede almacenar ningún reporte. 3. En el caso de que no haya sido creado ningún reporte el sistema emite

	un mensaje indicándolo.
Poscondiciones:	Se almacenó el reporte de la prueba en la base de datos para ser mostrado en el momento que sea solicitado.

Anexo 3.
Tabla 2.7 Descripción del Caso de Uso Cargar aplicación biométrica.

Caso de uso:	Cargar aplicación biométrica
Actores:	Probador
Resumen:	El caso de uso se inicia cuando el probador selecciona la opción Cargar aplicación biométrica, debe introducir su(s) nombres(s), sus apellidos, el tipo de biometría correspondiente a su algoritmo, la base de datos de referencia con la que desea realizar la prueba, el nombre del algoritmo y cargar el algoritmo que va a ser sometido a la prueba del sistema. El CU concluye cuando son generados los scores de cliente e impostor.
Precondiciones:	
Referencias:	RF 1.1, RF 1.2, RF 1.3, RF 1.4
Prioridad:	Crítico
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. El Probador selecciona la opción Cargar aplicación biométrica. 3. El probador debe introducir todos los datos solicitados y presionar el botón Run.	2. El sistema muestra un formulario solicitando: a) nombre(s) y apellidos de la persona. b) nombre del algoritmo. c) La opción de seleccionar la categoría del algoritmo y la base de datos de referencia con la cual

	desea realizar la prueba. d) la ubicación donde se encuentra el algoritmo biométrico. 4. Comienza el proceso de obtención de scores de cliente e impostor.
Flujo alterno	
	2. Deben estar llenos todos los campos solicitados para cargar la aplicación biométrica. Todos los datos introducidos deben estar validados. a) Ni el nombre y los apellidos del cliente deben contener números. d) La dirección suministrada de la ubicación de la aplicación biométrica debe ser correcta.
Poscondiciones:	Son generados y almacenados en el disco duro los score de clientes e impostor de la aplicación biométrica que se desea someter a prueba.

Anexo 4.

Tabla 3.5 Descripción de la clase RunConsole.

Nombre: **RunConsole**	
Tipo de Clase:	
Atributos	Tipo
fileName	**string**
Responsabilidades	
Nombre	Descripción
RunConsole(in fileName:string)	**Constructor con parámetros**
FileName():string	**GetAccessor, SetAccessor, property**
RunBiometricSystem(in args:string[*]):float	**Ejecuta una aplicación de consola**

Tabla 3.6 Descripción de la clase ScoreFile.

Nombre: **ScoreFile**	
Tipo de Clase:	
Atributos	Tipo
filePath	**string**
scoreClients	**List<T1->float>**
scoreImpostors	**List<T1->float>**
Tmp	**List<T1->float>**
Responsabilidades	
Nombre	Descripción
ScoreFile()	**Constructor sin parámetros**
FilePath():String	**GetAccessor, SetAccessor, property**
ScoreImpostors():List<T1->float>	**GetAccessor, SetAccessor, property**
ScoreClients():List<T1->flota>	**GetAccessor, SetAccessor, property**
LoadFile(in filePath:string):List<T1->float>	**Carga el fichero**

Tabla 3.7 Descripción de la clase Tester.

Nombre: **Tester**	
Tipo de Clase:	
Atributos	Tipo
maxClients	**float**
minImpostor	**float**
EER	**float**
Pas	**int**
numClients	**Int**
numImpostor	**Int**
numTreshold	**int**

FRR	**List<T1->float>**
FAR	**List<T1->float>**
Tresholds	**List<T1->float>**
scoreClients	**List<T1->float>**
scoreImpostors	**List<T1->float>**
Tmptreshold	**double**
Text	**string**
Responsabilidades	
Nombre	Descripción
Tester(in scoreClients:List<T1->float>, in scoreImpostors:List<T1->float>, in OP:float, in pas:int)	**Constructor con parámetros**
Tmptreshold():double	**GetAccessor, SetAccessor, property**
ScoreClients():List<T1->flota>	**GetAccessor, SetAccessor, property**
EER1():float	**GetAccessor, SetAccessor, property**
Thresholds():List<T1->float>	**GetAccessor, SetAccessor, property**
ScoreImpostors():List<T1->float>	**GetAccessor, SetAccessor, property**
NumTreshold():int	**GetAccessor, SetAccessor, property**
Pas():int	**GetAccessor, SetAccessor, property**
MaxClients():flota	**GetAccessor, SetAccessor, property**
MinImpostor():flota	**GetAccessor, SetAccessor, property**
FAR1():List<T1->float>	**GetAccessor, SetAccessor, property**
FRR1():List<T1->float>	**GetAccessor, SetAccessor, property**
Text():string	**GetAccessor, SetAccessor, property**
CalculationThresholds():List<T1->float>	**Calcula los valores del umbral de decisión**
CalculationFRRandFAR(in valueThreshold:float, in count:int):bool	**Devuelve true si calculó los valores de FRR y FAR para un umbral específico.**

arrayFRR():double[*]	**Guarda los valores de la lista FRR para un arreglo**
arrayFAR():double[*]	**Guarda los valores de la lista FAR para un arreglo**
arrayThresholds():double[*]	**Guarda los valores de la lista Thresholds para un arreglo**
CalculationEER():flota	**Calcula el valor del EER**
RaiseTextChange():void	**Controla el cambio de texto**
TextChange():EventHandler	**Evento**

Tabla 3.8 Descripción de la clase FACTORY.

Nombre: **FACTORY**	
Tipo de Clase:	
Atributos	Tipo
Cadenaconeccion	**string**
Coneccion	**NpgsqlConnection**
Command	**NpgsqlCommand**
Adapter	**NpgsqlDataAdapter**
Transacción	**NpgsqlTransaction**
Responsabilidades	
Nombre	Descripción
FACTORY()	**Constructor sin parámetros**
FACTORY(in cadena:string)	**Constructor con parámetros**

Tabla 3.9 Descripción de la clase REPORT.

Nombre: **REPORT**	
Tipo de Clase:	
Atributos	Tipo
_IDREPORT	**string**
_NOMBRES	**string**

_APELLIDOS	**string**
_NOMBREALGORITMO	**string**
_ERR	**string**
_ERRGRAFICA	**Image**
Responsabilidades	
Nombre	Descripción
IDREPORT():string	**GetAccessor, SetAccessor, property**
NOMBRES():string	**GetAccessor, SetAccessor, property**
APELLIDOS():string	**GetAccessor, SetAccessor, property**
NOMBREALGORITMO():string	**GetAccessor, SetAccessor, property**
ERR():string	**GetAccessor, SetAccessor, property**
ERRGRAFICA():Image	**GetAccessor, SetAccessor, property**

Tabla 3.10 Descripción de la clase REPORTFACTORY.

Nombre: **REPORTFACTORY**	
Tipo de Clase:	
Responsabilidades	
Nombre	Descripción
REPORTFACTORY(in cadena:string)	**Constructor con parámetros**
Load(inout objInfo:REPORT):void	
LoadAll():DataTable	**Carga todos los reportes**
LoadFingerprint():DataTable	**Carga los reportes de Fingerprint**
LoadSiganture():DataTable	**Carga los reportes de Signature**
LoadAllREPORT(in objInfo:REPORT):void	
Insert(inout objInfo:REPORT):void	**Inserta en la tabla**

Tabla 3.11 Descripción de la clase BiometricTest.

Nombre: **BiometricTest**

Tipo de Clase:	
Atributos	Tipo
scoreClients	**List<T1->float>**
scoreImpostors	**List<T1->float>**
Thresholds	**List<T1->float>**
Store	**ScoreFile**
Test	**Tester**
Report	**BiometricResult**
Components	**System.ComponentModel.IContainer= null**
menuStrip1	**System.Windows.Forms.MenuStrip**
fileToolStripMenuItem	**System.Windows.Forms.ToolStripMenuItem**
mnOpenScoreFileToolStripMenuItem	**System.Windows.Forms.ToolStripMenuItem**
mnRunTestToolStripMenuItem	**System.Windows.Forms.ToolStripMenuItem**
tbGraphics	**System.Windows.Forms.TabControl**
tabPage1	**System.Windows.Forms.TabPage**
graphicControl1	**GraphicControl**
aboutToolStripMenuItem	**System.Windows.Forms.ToolStripMenuItem**
bioTesterToolStripMenuItem	**System.Windows.Forms.ToolStripMenuItem**
label4	**System.Windows.Forms.Label**
mnRunBiometricSystem	**System.Windows.Forms.ToolStripMenuItem**
label7	**System.Windows.Forms.Label**
progressBar1	**System.Windows.Forms.ProgressBar**
label8	**System.Windows.Forms.Label**

reportsToolStripMenuItem	**System.Windows.Forms.ToolStripMenuItem**
mnBuildReportToolStripMenuItem	**System.Windows.Forms.ToolStripMenuItem**
mnViewReporToolStripMenuItem	**System.Windows.Forms.ToolStripMenuItem**
gbInstructions	**System.Windows.Forms.GroupBox**
tabPage2	**System.Windows.Forms.TabPage**
txtResult	**System.Windows.Forms.TextBox**
Responsabilidades	
Nombre	Descripción
BiometricTest()	**Constructor sin parámetros**
ScoreImpostors():List<T1->float>	**GetAccessor, SetAccessor, property**
ScoreClients():List<T1->float>	**GetAccessor, SetAccessor, property**
Test():Tester	**GetAccessor, SetAccessor, property**
refreshApplication(in sender:object, in e:EventArgs):void	**Refrescar aplicación**
Form1_Load(in sender:object, in e:EventArgs):void	**Crea la carpeta principal de la aplicación**
mnRunBiometricSystem_Click(in sender:object, in e:EventArgs):void	**Ejecuta la aplicación biométrica**
mnOpenScoreFileToolStripMenuItem_Click (in sender:object, in e:EventArgs):void	**Cargar la interfaz LoadScore**
mnRunTestToolStripMenuItem_Click(in sender:object, in e:EventArgs):void	**Ejecuta la prueba**
mnBuildReportToolStripMenuItem_Click(in sender:object, in e:EventArgs):void	**Crea un reporte**
mnViewReporToolStripMenuItem_Click(in sender:object, in e:EventArgs):void	**Carga la interfaz Reports**
mnRunTestEnabled():void	**Pone activo la opción de menú Run Test**
Dispose(in disposing:bool):void	**override**

InitializeComponent():void	**Inicializa todos los componentes visuales del formulario**

Tabla 3.12 Descripción de la clase LoadScore.

Nombre: **LoadScore**	
Tipo de Clase:	
Atributos	Tipo
Store	**ScoreFile**
scoreClients	**List<T1->float>**
scoreImpostors	**List<T1->float>**
biometricTest	**BiometricTest**
Components	**System.ComponentModel.IContainer=null**
tbClientScore	**System.Windows.Forms.TextBox**
tbImpostorScore	**System.Windows.Forms.TextBox**
btnLoadScoreClients	**System.Windows.Forms.Button**
btnLoadScoreImpostors	**System.Windows.Forms.Button**
groupBox1	**System.Windows.Forms.GroupBox**
label2	**System.Windows.Forms.Label**
label1	**System.Windows.Forms.Label**
btnOK:	**System.Windows.Forms.Button**
btnCancel	**System.Windows.Forms.Button**
Responsabilidades	
Nombre	Descripción
LoadScore(in biometricTest:BiometricTest)	**Constructor con parámetros**
btnLoadScoreClients_Click(in sender:object, in e:EventArgs):void	**Busca la ruta del score de cliente**
btnLoadScoreImpostors_Click(in sender:object, in e:EventArgs):void	**Busca la ruta del score de impostor**

btnOK_Click(in sender:object, in e:EventArgs):void	**Carga los scores y cierra el formulario**
btnCancel_Click(in sender:object, in e:EventArgs):void	**Cierra el formulario**
Dispose(in disposing:bool):void	**override**
InitializeComponent():void	**Inicializa todos los componentes visuales del formulario**

Tabla 3.13 Descripción de la clase Reports.

Nombre: **Reports**	
Tipo de Clase:	
Atributos	Tipo
Report	**BiometricResult**
Components	**System.ComponentModel.IContainer=null**
groupBox1	**System.Windows.Forms.GroupBox**
btnShowall	**System.Windows.Forms.Button**
dgvReports	**System.Windows.Forms.DataGridView**
groupBox2	**System.Windows.Forms.GroupBox**
btnShowType	**System.Windows.Forms.Button**
chbSignature	**System.Windows.Forms.CheckBox**
chbFingerprint	**System.Windows.Forms.CheckBox**
Responsabilidades	
Nombre	Descripción
Reports(in report:BiometricResult)	**Constructor con parámetros**
btnShowall_Click(in sender:object, in e:EventArgs):void	**Muestra todos los reportes**
btnShowType_Click(in sender:object, in e:EventArgs):void	**Muestra los reportes por tipo de biometría del algoritmo.**
Dispose(in disposing:bool):void	**override**
InitializeComponent():void	**Inicializa todos los componentes**

	visuales del formulario

Tabla 3.14 Descripción de la clase RunBiometricSystem.

Nombre: **RunBiometricSystem**	
Tipo de Clase:	
Atributos	Tipo
pathBD	**string**
typeBiometric	**string**
filePath	**string**
F	**FolderTest**
Report	**BiometricResult**
components:	**System.ComponentModel.IContainer =null**
comboBox1	**System.Windows.Forms.ComboBox**
comboBox2	**System.Windows.Forms.ComboBox**
label1	**System.Windows.Forms.Label**
label2	**System.Windows.Forms.Label**
label3	**System.Windows.Forms.Label**
label4	**System.Windows.Forms.Label**
label5	**System.Windows.Forms.Label**
label6	**System.Windows.Forms.Label**
label7	**System.Windows.Forms.Label**
btnRunSystem	**System.Windows.Forms.Button**
tbName	**System.Windows.Forms.TextBox**
tbLastName	**System.Windows.Forms.TextBox**
tbSystemPath	**System.Windows.Forms.TextBox**
btnBrowse	**System.Windows.Forms.Button**
tbSystemName	**System.Windows.Forms.TextBox**

groupBox1	System.Windows.Forms.GroupBox
Responsabilidades	
Nombre	Descripción
RunBiometricSystem(in report:BiometricResult)	Constructor con parámetros
comboBox1_SelectedIndexChanged(in sender:object, in e:EventArgs):void	Escoger el tipo de biometría y cargar bases de datos de referencia
comboBox2_SelectedIndexChanged(in sender:object, in e:EventArgs):void	Seleccionar la base de datos de referencia
loadApplication():void	Buscar la aplicación biométrica que se va a someter a prueba
test_TextChange(in sender:object, in e:EventArgs):void	Cambia texto
btnRunSystem_Click(in sender:object, in e:EventArgs):void	Ejecuta la aplicación biométrica
btnBrowse_Click(in sender:object, in e:EventArgs):void	Llamar al método loadApplication
FullData():bool	Verifica si todos los textbox del formulario están llenos
Dispose(in disposing:bool):void	override
InitializeComponent():void	Inicializa todos los componentes visuales del formulario

Tabla 3.15 Descripción de la clase GraphicControl.

Nombre: **GraphicControl**	
Tipo de Clase:	
Atributos	Tipo
myPane	GraphPane #{readOnly}
listCurve	List<T1->LineItem>=new List<LineItem>() #{readOnly}
Components	System.ComponentModel.IContainer =null

zedCurrencyRatio	ZedGraph.ZedGraphControl
Responsabilidades	
Nombre	Descripción
GraphicControl()	Constructor sin parámetros
this(in index:int):LineItem	GetAccessor, SetAccessor, indexer
UpdateSettingGraphics():void	Actualizar con nuevos datos la configuración del panel gráfico
SettingGraphics(in titleGraphic:string, in titleGraphicX:string, in titleGraphicY:string):void	Configuración del panel gráfico
DrawCurve(in offset:int, in array:double[*], in caption:string, in color:Color):void	Dibuja curva en la gráfica
DrawCurve(in offset:double[*], in array:double[*], in caption:string, in color:Color):void	Dibuja curva en la gráfica
DrawCurve(in offset:float, in array:int[*], in caption:string, in color:Color):void	Dibuja curva en la gráfica
drawPoint(in x:double, in y:double, in caption:string, in color:Color):void	Dibuja un punto en la gráfica
drawPoint1(in x:double, in y:double, in caption:string, in color:Color):void	Dibuja un punto en la gráfica
CreateGraphicCurve(in textTit:string, in curve:IPointList, in color:Color):void	Crear una curva
SaveImage(in fileName:string):void	Salva la imagen del gráfico
CleanGraphics():void	Limpia el gráfico
DrawGrid():void	Dibuja una rejilla en la gráfica
Dispose(in disposing:bool):void	override
InitializeComponent():void	Inicializa todos los componentes visuales del formulario

Anexo 5.

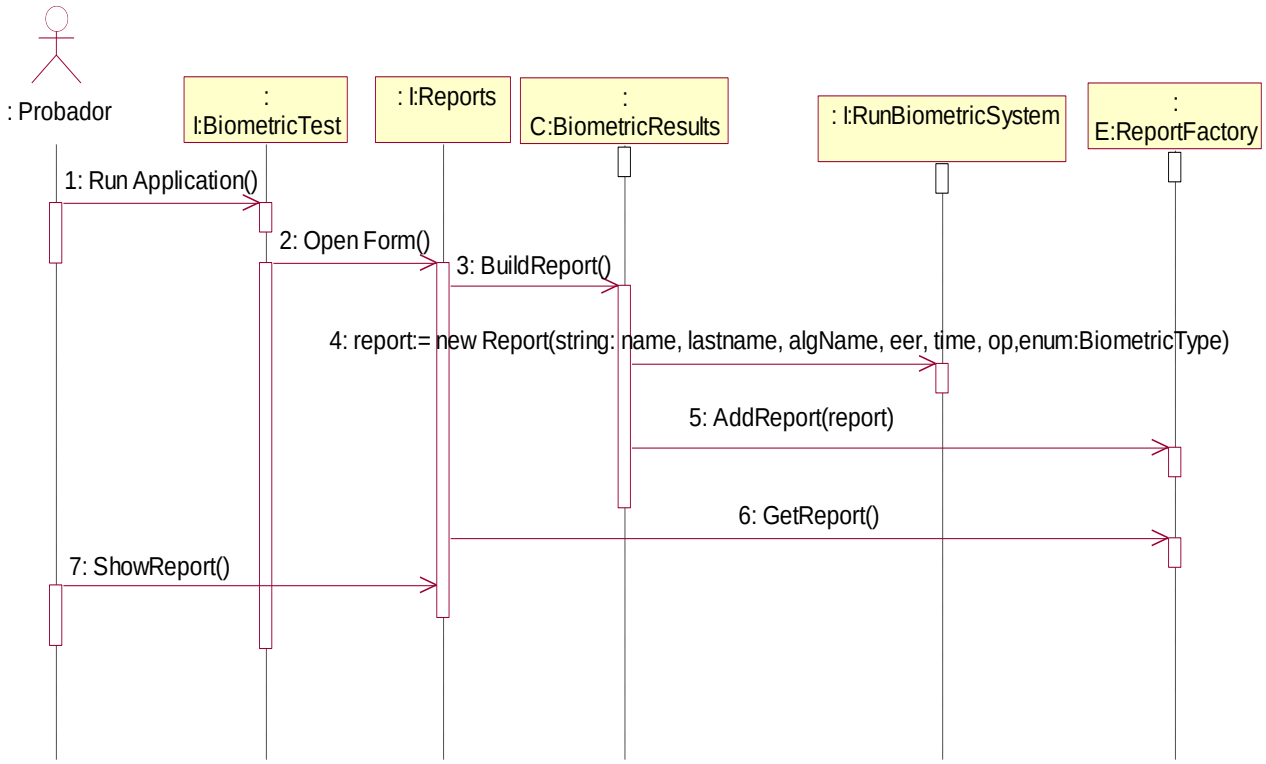

Figura 32 Diagrama de secuencia del caso de uso: Realizar reporte.

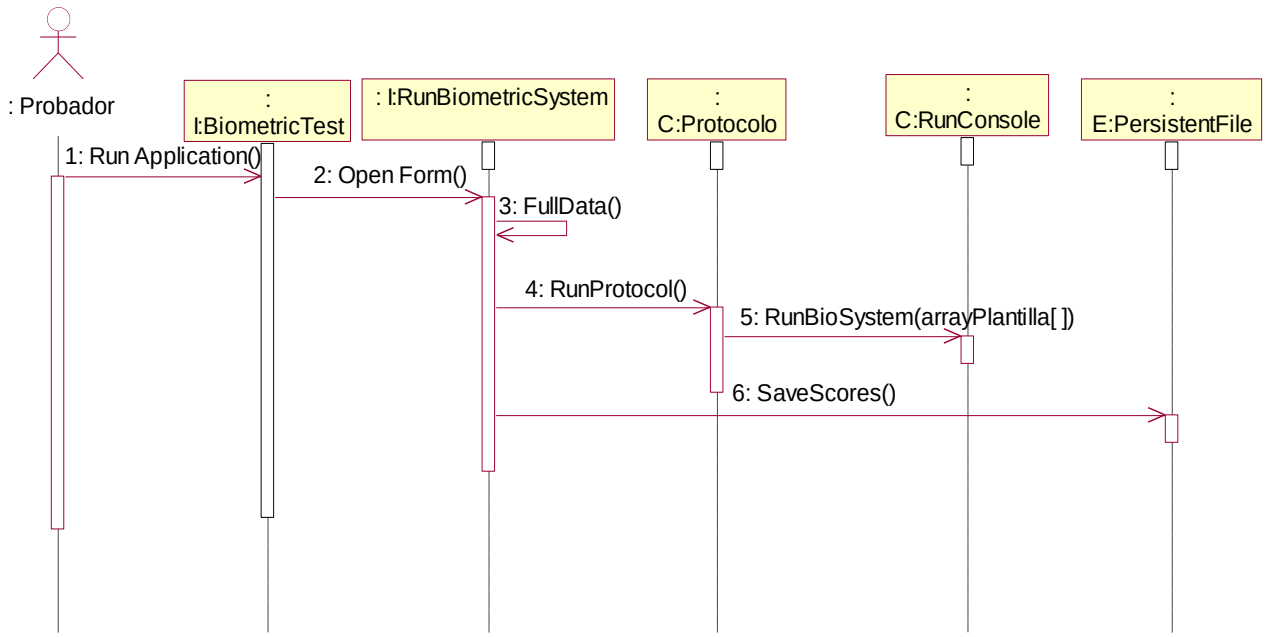

Figura 33 Diagrama de secuencia del caso de uso: Cargar aplicación biométrica.

Anexo 6.

Tabla 3.22 CU. Cargar aplicación biométrica.

Escenario	V1	V2	V3	V4	V5	V6	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Cargar aplicación biométrica.	V	V	V	V	V	V	Se obtienen los scores de cliente e impostor.	Satisfactorio.	1. Se introduce el nombre y los apellidos del cliente. 2. Se selecciona el tipo de biometría a la cual pertenece el algoritmo que se va a someter a prueba. 3. Se introduce el nombre del algoritmo biométrico. 4. Se selecciona la aplicación biométrica que se va a someter a la prueba.
	I	V	V	V	V	V	Indica que algún campo es incorrecto.	Satisfactorio.	
	V	I	V	V	V	V	Indica que algún campo es incorrecto.	Satisfactorio.	
	V	V	I	V	V	V	Indica que algún campo es incorrecto.	Satisfactorio.	
	V	V	V	I	V	V	Indica que algún campo es incorrecto.	Satisfactorio.	
	V	V	V	V	I	V	Indica que algún campo es incorrecto.	Satisfactorio.	
	V	V	V	V	V	I	Indica que algún campo es incorrecto.	Satisfactorio.	
	I	I	I	I	I	I	Indica que algún campo es incorrecto.	Satisfactorio.	

Leyenda.

V1. Introducir nombre del cliente.

V2. Introducir apellidos del cliente.

V3. Seleccionar el tipo de biometría.

V4. Seleccionar base de datos de referencia.

V5. Introducir nombre del algoritmo.

V6. Cargar aplicación biométrica.