UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

PROYECTO FIN DE CARRERA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

BACK-END FOR A BIOMETRIC EXTENDED EXPERIMENT PLATFORM (BEEP)

AUTORA: **ELENA Mª ARRIBAS GONZÁLEZ**
TUTORA: **BELÉN RUÍZ MEZCUA**
DIRECTOR: **LUIS PUENTE RODRÍGUEZ**

Julio de 2009

Proyecto Fin de Carrera

Back-End for a Biometric Extended Experiment Platform (BEEP)

AUTORA:

Elena Mª Arribas González

TUTORA:

Belén Ruíz Mezcua

DIRECTOR:

Luis Puente Rodríguez

La defensa del presente Proyecto Fin de Carrera se realizó el día 17 de Julio de 2009 y fue evaluada por el siguiente tribunal:

PRESIDENTE: Ángel García Crespo

SECRETARIO: Vicente Palacios Madrid

VOCAL: Francisco Javier Calle Gómez

AGRADECIMIENTOS

Quiero agradecer este trabajo a toda la gente que ha estado apoyándome incondicionalmente y no ha dudado de mí nunca.

En primer lugar, agradezco a mis tutores Luis y Belén por su ayuda y apoyo durante el desarrollo del proyecto. A mi compañero Diego, por el trabajo conjunto que hemos realizado y todas las horas que hemos pasado codo con codo.

En segundo lugar, a mis padres, César y Pilar, por todo el esfuerzo que han hecho para que yo pueda llegar a este momento y a mi hermano César, porque siempre han estado ahí motivándome y ayudándome en todo.

Tampoco quiero olvidarme del resto de la familia, mis abuelos, tíos, primos, que han estado pendientes de mi formación. Gracias a Cuenca por ser única. Allí están todos los míos y me ha permitido desconectar durante todos estos años.

Gracias a Marga y a Germán, porque sabéis lo importantes que sois para mí y porque habéis estado conmigo siempre, y lo que nos queda.

También quería dar las gracias a todos los amigos que he conocido durante la carrera, gracias a mis “gañanes” porque las tardes de estudio no las olvidaré jamás. Al comando Industriales: J.J., Andrea, Hasself, Marta, Franconettis, Manuel,…..y a todos los demás por los buenos momentos que hemos pasado juntos. A Alberto y a Vicen por ponerme el café todas las mañanas y darme los buenos días con una sonrisa grande.

El CAU ha sido fundamental para mi, mil gracias a todos por enseñarme tantas cosas, por aguantar mis canciones y tonterías varias. Gracias a Ure por ser mi líder espiritual aquí, a Piticli, Bameda y Amand por introducirme en el mundo “friki” y al resto de compis porque ha sido una etapa que no olvidaré jamás.

Gracias a ti, por escucharme, por motivarme y por estar a mi lado cuando me he venido abajo, por creer que era capaz y ayudarme para que esto fuera posible.

Romi gracias por darme ánimos y estar ahí, la siguiente celebración espero que sea la tuya o la de Gonza!

Gracias de corazón a todos y espero que todos y cada uno de vosotros sigáis conmigo en la nueva etapa que empiezo como Ingeniera.

PRÓLOGO

En los últimos años, la demanda de sistemas fiables de identificación humana ha sufrido un importante incremento, debido a la aparición de un gran número de aplicaciones, tanto civiles como militares. Este crecimiento ha estimulado el interés de la comunidad científica en la evaluación de los sistemas biométricos de forma controlada.

Existen tres problemas principales derivados de la evaluación de los sistemas biométricos. En primer lugar, la comparación de los resultados de diferentes investigaciones no es rigurosa si el marco de trabajo no es común a la ejecución de todos los experimentos. En segundo lugar, los resultados presentados no son comparables si los datos biométricos utilizados en las evaluaciones de los sistemas no proceden de la misma base de datos. Por último, la realización de experimentos requiere la gestión de un amplio conjunto de tareas para adecuar el entorno de ejecución a las necesidades de los mismos.

Por los motivos expuestos, se hace necesario un contexto de experimentación repetible y bien definido que permita comparar las prestaciones de sistemas biométricos.

A lo largo de la presente memoria se describe el desarrollo del proyecto Back-Office de la plataforma BEEP (Biometric Extended Experiment Platform) que establece un marco controlado de experimentación. Ésta permite tanto la creación, edición y ejecución de dichos experimentos, como la obtención y comparación de resultados sobre bases de datos biométricos comunes.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En este capítulo introductorio se ofrece una presentación general del proyecto desarrollado a lo largo de esta memoria. Por un lado, se dan a conocer las motivaciones que han llevado a la creación de la plataforma y por otro, los objetivos buscados.

1.1. MOTIVACIÓN

Dentro del Departamento de Informática de la Universidad Carlos III de Madrid se planteó el desarrollo de una plataforma de experimentación biométrica que permitiese la investigación y obtención de resultados de manera fiable bajo un mismo entorno. La idea surgió dentro de SoftLab, perteneciente al grupo Sintonía. Entre sus líneas de investigación se contempla la evaluación y el desarrollo de sistemas de reconocimiento biométrico de personas.

Las tecnologías que soportan los sistemas biométricos se han convertido en uno de los principales puntos de atención por parte de muchas empresas y entidades. Como consecuencia, la investigación en este área ha crecido notablemente en los últimos años en un intento de ofrecer implementaciones de sistemas más precisos y seguros.

La comunidad científica y las organizaciones comerciales hasta la fecha, han venido ofreciendo medidas de prestaciones para los sistemas biométricos. La mayoría de los resultados no podían ser comparados, puesto que se obtenían de sistemas biométricos soportados en bases de datos propias y protocolos de evaluación no estandarizados. Actualmente, [Capelli et al, 2006], los procesos de evaluación más importantes se encuentran administrados por grupos independientes y la información utilizada para testear los algoritmos no ha sido utilizada nunca antes. Para que un protocolo de evaluación sea aceptado por la comunidad científica, han de estar publicados: los detalles de sus procedimientos, el protocolo de medidas de prestaciones y algunos ejemplos representativos del conjunto de datos.

Para conseguir un entorno de pruebas controlado, diferentes organizaciones comerciales y la comunidad científica han desarrollado competiciones que permiten la comparación de algoritmos biométricos, pero ninguna de ellas ha propuesto el desarrollo de una plataforma como la planteada en este proyecto.

A los inconvenientes anteriores hay q añadir los problemas derivados de la gestión de la información biométrica. Este tipo de información según la Ley Orgánica de Protección de Datos de Carácter Personal [LOPD, 1999], tiene la condición de carácter privado y por tanto, debe garantizarse a cualquier individuo su derecho de cancelación. La distribución de este tipo de información a través de un medio físico, tal como se realiza habitualmente, imposibilita al usuario ejercer dicho derecho.

1.2. OBJETIVOS

Una vez expuestas las motivaciones que han llevado a la implementación de la plataforma, se muestran los objetivos a cumplir con el desarrollo de la misma.

BEEP (Biometric Extended Experiment Platform) surge de las motivaciones presentadas en la sección anterior. El objetivo de la plataforma es dar solución a los problemas derivados de la experimentación biométrica al tiempo que ofrece a los investigadores un marco de trabajo que resulte sencillo, repetible, modificable, intuitivo y donde la información biométrica se encuentre protegida del libre acceso.

El planteamiento y desarrollo de la plataforma de experimentación BEEP implica la creación de un entorno estable y controlado de experimentación que permita a los investigadores obtener resultados, compartirlos y compararlos. A partir de las motivaciones descritas, los principales objetivos del proyecto son la creación de un metamodelo que normalice la definición de experimentos y el desarrollo de la plataforma que soporte la definición, ejecución y obtención de resultados de dichos modelos. A continuación se detallan cada uno de los objetivos:

- Se desarrollará un metamodelo capaz de describir las operaciones y los datos implicados en la ejecución de experimentos biométricos. El metamodelo permitirá definir un experimento como una sucesión de operaciones que transforman una información biométrica en otra con el fin de obtener las prestaciones de los propios procesos

- Se desarrollará un sistema que permitirá crear y modificar experimentos biométricos de acuerdo al metamodelo propuesto y con independencia de la ubicación del usuario.

- El sistema será capaz de almacenar descripciones de datos biométricos en distintos Sistemas Gestores de Bases de Datos.

- Se implementará un sistema capaz de generar descripciones de algoritmos basados en librerías de código para el entorno de ejecución .NET.

- Se desarrollará un sistema capaz de interpretar las definiciones de los experimentos con el objetivo de ejecutar los procesos necesarios para

evaluar los sistemas biométricos modelados.

- El sistema será capaz de generar y presentar informes de prestaciones a partir de los resultados obtenidos tras la ejecución de las definiciones de los experimentos.

- Se desarrollará una arquitectura de comunicación para permitir la comunicación entre los subsistemas de la plataforma, asegurando la independencia de los mismos.

- Al tratarse de información de carácter personal, según la LOPD, el sistema gestionará los datos biométricos acorde con su nivel de privacidad. De este modo, se salvaguarda la identidad del sujeto del cual proviene la información sensible de las muestras biométricas empleadas en la ejecución de los experimentos.

Dada la complejidad de la plataforma, el desarrollo de la misma se ha dividido en dos proyectos Fin de Carrera. El primero de ellos se corresponde con el subsistema de interacción con el usuario o Front-End, siendo el segundo el encargado de la ejecución de experimentos y elaboración de informes de resultados o Back-End.

Concretamente, la presente memoria recoge el desarrollo del subsistema de ejecución de experimentos o Back-End. El objetivo de este subsistema consiste en la comprobación de la coherencia del modelo de los experimentos, para su posterior ejecución y evaluación. Los resultados del mismo se almacenarán en un informe que será presentado al usuario por el Front-End.

1.3. ESTRUCTURA DEL DOCUMENTO

Esta memoria recoge en las próximas secciones la evolución del proyecto BEEP. Comienza con un capítulo en el que se describe el marco del proyecto y continúa con la exposición de las diferentes fases en las que se ha dividido el desarrollo del mismo. Este documento se divide en los siguientes capítulos:

- Capitulo 1. Introducción: Presentación del proyecto. Quedan reflejadas tanto las motivaciones como los objetivos a alcanzar.

- Capitulo 2. Estado del Arte: Situación actual de los sistemas biométricos y los diferentes marcos de experimentación, junto con los problemas asociados a los mismos.

- Capitulo 3. Definición del Metamodelo: Definición de los elementos involucrados en la ejecución de experimentos.

- Capitulo 4. Plataforma BEEP: Arquitectura funcional de la plataforma y descripción de los diferentes componentes arquitectónicos de la misma.

- Capítulo 5. Back-End. Análisis: Presentación de su análisis funcional y

descripción de perfiles de usuario y casos de uso.

- Capítulo 6. Back-End. Diseño: Exposición de las principales decisiones de diseño e implementación tomadas para el desarrollo de la aplicación.

- Capítulo 7. Gestión del Proyecto: Costes y plazos de ejecución del proyecto.

- Capitulo 8. Conclusiones y trabajo futuro. Especificación de las conclusiones. Nivel de consecución de los objetivos iniciales y líneas futuras de ampliación y mejora.

- Referencias y anexos.

ESTADO DEL ARTE

A lo largo de este capítulo se presenta una visión general de la biometría. En primer lugar, se describen las etapas del proceso biométrico, a continuación se enumeran las principales tecnologías y finalmente, se muestran los diferentes esquemas de experimentación y las medidas de evaluación más utilizadas.

2.1. INTODUCCIÓN

La biometría es el estudio mensurativo o estadístico de los fenómenos o procesos fisiológicos. Es un término que acompaña a la aplicación de métodos estadísticos en la medición de procesos biológicos [Dessimoz et al, 2005]. La biometría hace referencia a todas aquellas tecnologías que se emplean para analizar las características fisiológicas o antropométricas y de comportamiento de los individuos.

Se consideran rasgos útiles para los objetivos de la biometría aquellos que son susceptibles de ser medidos o caracterizados mediante parámetros computables. En la Tabla 2.1 se muestran los identificadores más utilizados en la actualidad. De manera ideal, la parametrización de cualquier rasgo debería satisfacer las siguientes propiedades [Jain et al, 2004]:

- Disponibilidad: El rasgo debería estar presente en todos los sujetos de la población.
- Diferenciación: Su variación debería ser elevada entre los diferentes sujetos de la población.
- Invariabilidad: Su variación debería ser mínima para un único sujeto de la población.
- Accesibilidad: La captura de la información del rasgo debería ser una operación sencilla.

Sin embargo, en un sistema biométrico existen otras cuestiones que deberían ser consideradas [Jain et al, 2004]:

- Rendimiento: Hace referencia a la precisión y velocidad del proceso de reconocimiento y a los recursos que son necesarios para conseguir los

niveles de rendimiento deseados.

- Aceptabilidad: Expresa la capacidad de aceptación de un determinado rasgo biométrico por parte de la usuarios.

- Elusión: Refleja la facilidad con la que el sistema puede ser engañado mediante métodos fraudulentos.

TIPO	RASGOS BIOMÉTRICOS
Fisiológicos	• Voz. • Huella dactilar. • Geometría facial. • Análisis del iris. • Geometría de la palma de la mano. • Huella de la palma de la mano. • Venas del dorso de la mano.
De comportamiento	• Voz[1]. • Firma. • Dinámica de tecleo. • Movimiento y/o cadencia del paso.

Tabla 2.1. Características Personales como Identificadores Biométricos.

El concepto de reconocimiento automático de personas pertenece a los últimos años de la década de los setenta [Raphael et al, 1974]. Los rasgos biométricos presentados en la Tabla 2.1 son características intrínsecas de las personas y por lo tanto, pueden ser empleados en aplicaciones relacionadas con el reconocimiento de la identidad. En los últimos años se han desarrollado ampliamente las tecnologías de los sistemas biométricos y el número de implementaciones ha aumentado notablemente. La biometría se encuentra en continua evolución y su futuro promete una industria de costes asequibles y un amplio abanico de aplicaciones.

2.2. PROCESO BIOMÉTRICO

Un sistema biométrico es un proceso que consta de tres fases [Blackburn, 2004]. En un primer paso, uno o varios sensores adaptadores reciben una señal biométrica. A continuación, esta señal se caracteriza mediante un algoritmo o función matemática para obtener un conjunto de datos que se pueden manipular y comparar. En el tercer y último paso, se comparan los

[1] La voz no es sólo un rasgo de comportamiento sino que también posee un componente fisiológico.

datos generados en el paso anterior con la información existentes en una base de datos y se obtiene un resultado que indica si el individuo generador de la señal es aceptado o rechazado.

2.2.1. Etapas

El proceso biométrico consiste en la adquisición y transformación de datos para realizar cualquier tipo de tarea relacionada con la determinación de la identidad [Dessimoz et al, 2005].

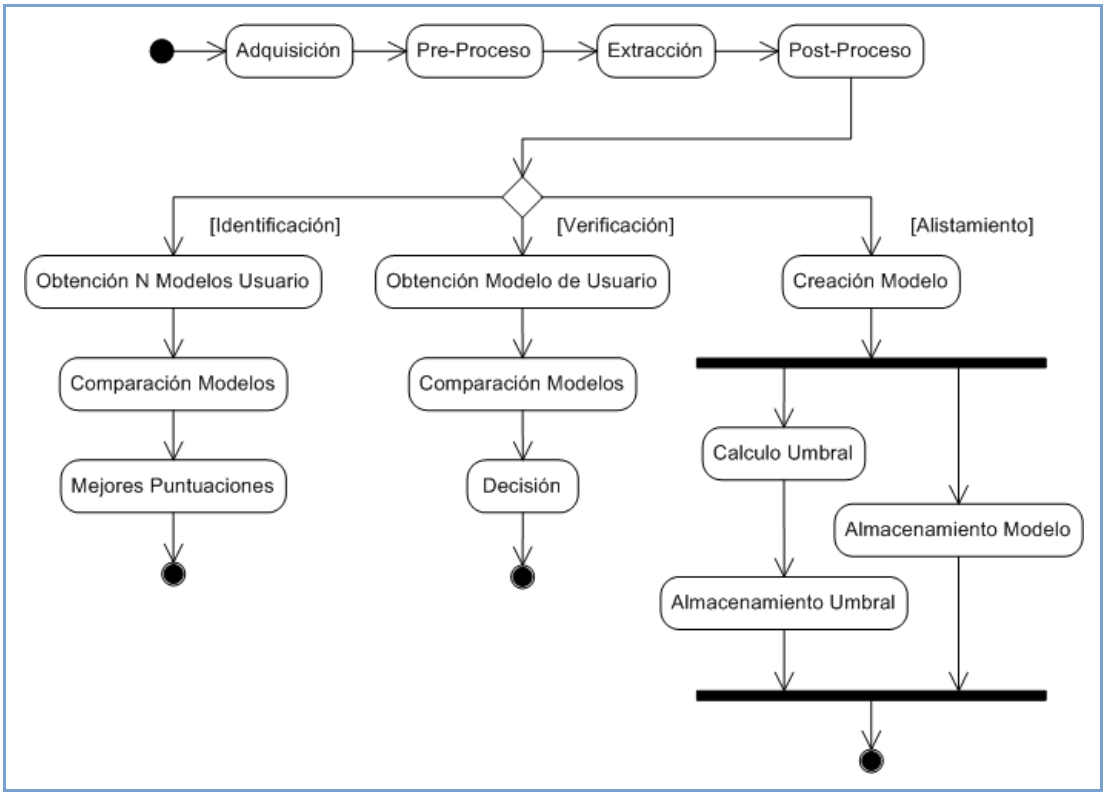


Figura 2.1. Diagrama de Flujo del Proceso Biométrico.

En [Dessimoz et al, 2005] se describe el proceso biométrico como una sucesión lineal de operaciones. El diagrama de flujo de la Figura 2.1 [Dessimoz et al, 2005] ofrece una visión esquemática del proceso biométrico. De manera general y para todas las modalidades biométricas, la información se transforma de acuerdo a los siguientes pasos:

- Captura o adquisición: La información se recoge por medio de un sensor y se almacena en memoria.

- Pre-procesado: La información adquirida en el dominio de la señal se prepara para la extracción de características. Normalmente, se realiza una normalización como paso previo a la extracción.

- Extracción de características: Una vez pre-procesada la información biométrica en el dominio de la señal, se obtiene una nueva representación de la misma en un nuevo dominio. Este proceso reduce la

dimensionalidad de la señal y crea una representación a nivel de características del patrón de entrada. Esta nueva representación de la señal se emplea en el posterior reconocimiento de patrones.

- Post-procesado: Las características obtenidas de la señal de entrada se normalizan a fin de adaptarlas al sistema de clasificación.

- Creación de modelos: Mediante un conjunto de datos de entrenamiento se genera un modelo matemático para obtener una representación genérica de un determinado usuario. En algunos sistemas es necesario crear un modelo de usuarios de fondo con el fin de normalizar las puntuaciones.

- Almacenamiento de modelos: Una vez se han estimado los parámetros que definen la identidad de un usuario, se almacena el modelo de tal forma que pueda ser empleado en etapas posteriores del proceso.

- Comparación de modelos: En esta etapa se compara un conjunto de vectores de características con el modelo biométrico de un determinado usuario. Esta comparación, con uno o varios modelos, resulta en una puntuación que recoge el parecido entre los datos biométricos de entrada y las identidades de los modelos con los que han sido cotejados.

- Cálculo de umbrales: A partir de datos de usuarios reales e impostores se calcula un umbral de decisión que marca la aceptación de la identidad en el sistema.

2.2.2. Operaciones

Las etapas descritas anteriormente se emplean en tres operaciones de alto nivel [Dessimoz et al, 2005]: alistamiento, verificación e identificación.

- Alistamiento: Con esta operación se añade un nuevo usuario al sistema. Mediante un cierto número de representaciones biométricas se entrena un modelo para el usuario y se adapta el modelo de los usuarios de fondo en caso de ser necesario.

- Verificación: La verificación consiste en comparar la representación biométrica de un usuario donante frente al modelo de un usuario pretendido. Una vez contrastada la información con el modelo se obtiene una puntuación que se coteja con un umbral para aceptar o denegar la identidad pretendida.

- Identificación: Mediante la identificación se busca la representación biométrica más parecida a la representación de entrada en una base de datos de modelos de usuarios. La información de entrada se compara con todos los modelos de interés y aquel que obtiene la mayor puntuación se sugiere como identidad.

2.3. BIOMETRÍA UNIMODAL

Una vez introducido el concepto de biometría, a continuación se presentan las características más relevantes de los principales rasgos.

2.3.1. Huella Dactilar

Los sistemas basados en huella dactilar son una de las tecnologías con mayor grado de madurez y actualmente, son los sistemas más utilizados y aceptados a nivel mundial. Estos sistemas tienen su base en los desarrollos realizados por Galton y Purkinje a finales del siglo XVIII. De forma general, la extracción de características de una huella dactilar sigue el esquema presentado en la Figura 2.2.

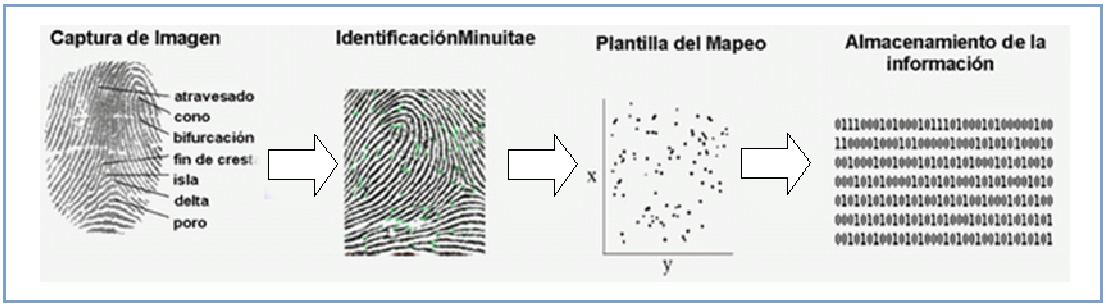


Figura 2.2. Extracción de Características de la Huella Dactilar.

La extracción de características en estos sistemas se basa en el análisis de una imagen en escala de grises. Principalmente existen dos procedimientos. Por un lado, en los sistemas basados en correlación [Maltoni et al, 2003] se superponen dos imágenes de la huella dactilar y se calcula la correlación entre ambas para diferentes posiciones mediante traslación y rotación. Por otro lado, en los sistemas basados en la extracción de puntos característicos [Yager et al, 2004] se segmenta la imagen y se localizan puntos singulares (bifurcaciones, fines de cresta, islas,…) denominados minucias. La disposición espacial de estas singularidades ofrece una elevada capacidad de diferenciación.

2.3.2. Geometría Facial

La mayoría de los sistemas actuales de reconocimiento facial determinan la apariencia de un sujeto a través de la obtención de puntos nodales de la cara. A partir de estos puntos, se determina la distancia entre los ojos, la anchura de la nariz, la distancia del ojo a la boca o la longitud de la línea de la mandíbula [Zhao et al, 2003]. La Figura 2.3 recoge el proceso de extracción de características de la geometría facial.

Aunque existen diversos métodos aplicables al reconocimiento facial, en la actualidad se utilizan principalmente los métodos basados en las características geométricas de la cara.

Land Marks se basa en la localización de diferentes características geométricas como la distancia entre ojos o la anchura de la nariz, de tal forma que el sujeto

queda caracterizado por el conjunto de distancias. En las técnicas de correlación se toman dos imágenes y se calcula el índice de correlación entre ambas para obtener un indicador del parecido. Tiene un elevado coste computacional. Finalmente, la técnica de Eigen Faces codifica una imagen facial de tal forma que puede ser representada como una combinación lineal de otras imágenes [Sirovich et al, 1987]. Permite optimizar el sistema y aumentar la velocidad de procesado.

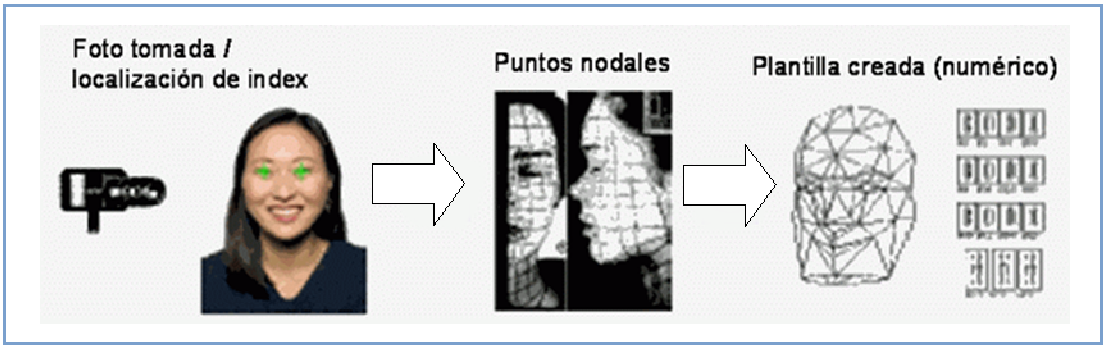


Figura 2.3. Extracción de Características de la Geometría Facial.

2.3.3. Iris

El iris es la región anular del ojo que se encuentra limitada por la pupila en la parte interior y por la esclera en la exterior. Los patrones del iris se forman durante el desarrollo del feto y se estabilizan durante los dos primeros años de vida. En la actualidad se emplean varias aproximaciones al problema del reconocimiento basado en iris [Wildes, 2005].

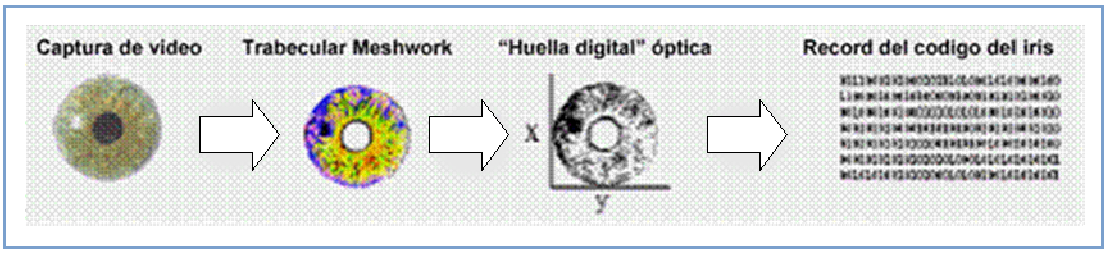


Figura 2.0.4. Extracción de Características de Iris.

Los métodos de ajuste basados en modelos de histogramas procesan la imagen del iris mediante filtros paso-banda de dos dimensiones utilizan el discriminante lineal de Fisher para realizar el proceso de comparación. Otros sistemas [Tisee, 2003] emplean la transformada de Hough para localizar el iris y mediante la transformada de Hilbert obtienen las características de éste. El último tipo de sistemas [Masek, 2003] utiliza transformadas de Hough circulares para localizar el iris y filtros de Gabor para extraer el conjunto de características.

2.3.4. Geometría de la Palma de la Mano

El reconocimiento biométrico por medio de la geometría de la mano se fundamenta en el cálculo de las distancias entre diferentes puntos

característicos de la mano [Sánchez et al, 2000 y Sánchez, 2000] a partir de una imagen bidimensional o tridimensional de la misma. Entre las medidas obtenidas se incluye normalmente la longitud de los dedos y el ancho y alto de la mano.

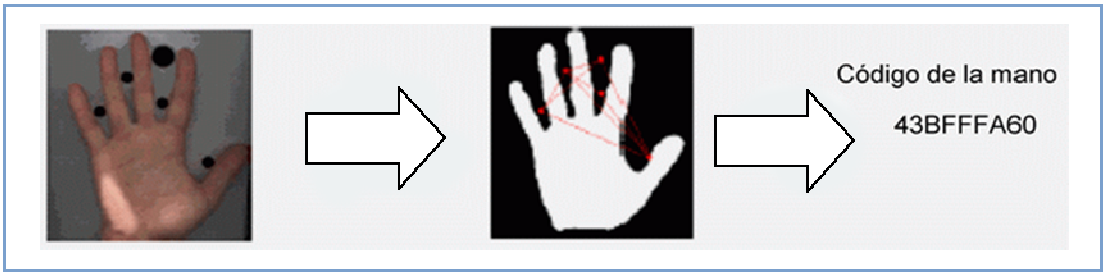


Figura 2.5. Extracción de Características de la Geometría de la Mano.

Estos sistemas no están orientados a aplicaciones de seguridad ya que la información obtenida de la geometría de la mano no permite identificar unívocamente a una persona.

2.3.5. Huella de la Palma de la Mano

Al igual que el reconocimiento de huellas dactilares, el reconocimiento de la palma de la mano se basa en la información representada por las crestas y valles de su superficie de la palma [Duta et al, 2002]. La información de la palma incluye entre otras características el sentido de las crestas y la presencia o ausencia de minucias.

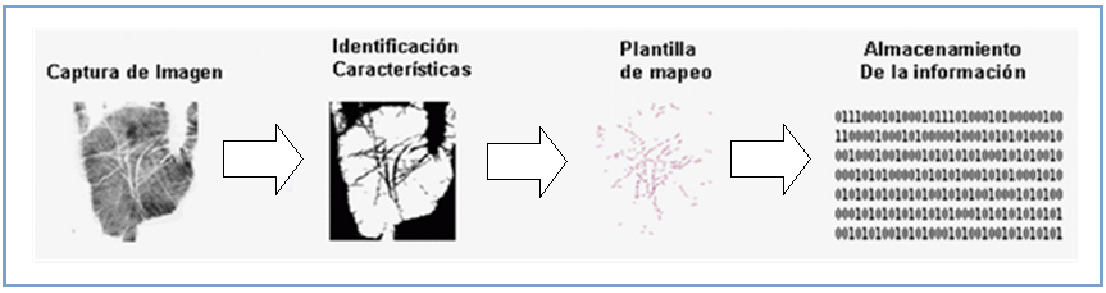


Figura 2.6. Extracción de Características de la Palma de la Mano.

2.3.6. Firma

La firma es una característica de comportamiento que se origina por los movimientos rápidos de la mano debidos a la actividad del sistema neuromuscular.

El reconocimiento de firma ofrece dos tipos de aproximaciones. En primer lugar, los métodos on-line obtienen características dinámicas del rasgo biométrico. Los principales métodos de verificación de firma on-line se basan en Dynamic Time Warping [Sato et al, 1982], HMM con distribuciones de probabilidad Gaussianas [Richiardi et al, 2003] y redes neuronales [Chang et al, 1993]. En segundo lugar, los métodos off-line procesan una imagen para obtener características estáticas de la misma. Algunos de estos sistemas

utillizan características geométricas de la imagen y redes neuronales para realizar la verificación [Huang et al, 1997]. Por su parte, otros sistemas [Justino et al, 2001] hacen uso de HMM para caracterizar la distribución espacial de la firma.

2.3.7. Voz

El reconocimiento de personas mediante voz se soporta tanto en la estructura física del sujeto como en las características de comportamiento del mismo. El esquema de extracción de características se puede ver en la Figura 2.7. Existen dos tipos de sistemas [Reynolds, 1995] de reconocimiento. Mientras que los sistemas dependientes del texto se basan en la repetición de un texto específico, los sistemas no dependientes no imponen ningún tipo de limitación.

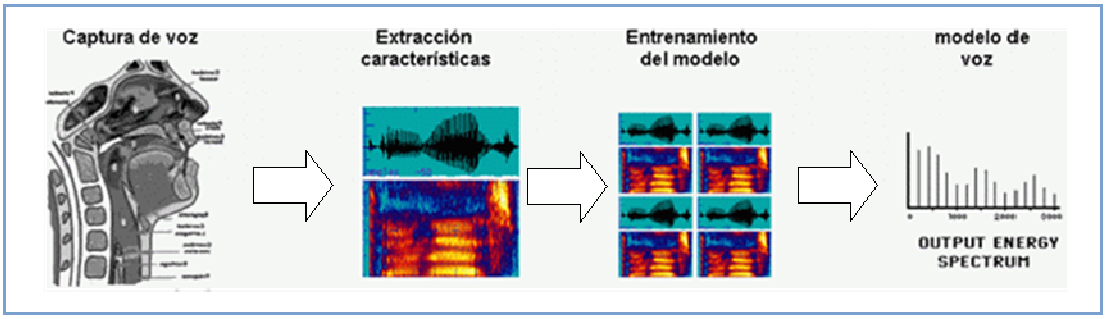


Figura 2.7. Extracción de Características de la Voz.

A lo largo de los años se han propuesto diferentes algoritmos para solventar el problema del reconocimiento de locutores [Dessimoz et al, 2005]. Los métodos probabilísticos modelan de forma estadística el comportamiento de la voz [Ruíz,1998]. A este tipo pertenecen los Modelos Ocultos de Markov [Poritz, 1982] y los Modelos de Mezclas Gaussianas [Reynolds, 1995]. Otros métodos de clasificación empleados en el reconocimiento de locutores son las Redes Neuronales [Oglesby et al, 1988], las Máquinas de Vectores Soporte [Schmidt et al, 1996] y Dynamic Time Warping [Pandit et al, 1998].

2.3.8. Dinámica del Tecleo

Es un tipo de rasgo biométrico conductual empleado en la verificación de la identidad de un individuo mediante su cadencia de escritura en un teclado [Monrose et al, 2000]. Esta tecnología se sostiene sobre la premisa de que cada individuo exhibe un patrón y una cadencia distintivos.

En la mayoría de los sistemas se emplea la latencia entre pulsaciones como característica, sin embargo en otros se utiliza también el tiempo que permanece la tecla presionada. Esta tecnología no requiere de hardware o dispositivos adicionales ya que se soporta sobre software de captura de la dinámica de tecleo. Esa tecnología utiliza clasificadores bayesianos, redes neuronales y sistemas basados en lógica difusa.

2.3.9. Movimiento

El movimiento es un tipo de rasgo biométrico conductual que se usa para verificar la identidad de un individuo examinando su patrón de marcha al caminar [Nixon et al, 2006 y Boyd et al, 2005]. Mediante este rasgo se puede realizar el reconocimiento a larga distancia e incluso con imágenes de baja resolución. Los sistemas se pueden clasificar en estáticos (basados en la figura humana) y dinámicos (basados en el movimiento del individuo).

Figura 2.8. Extracción de Características de la Silueta.

2.4. BIOMETRÍA MULTIMODAL

Los sistemas biométricos que trabajan utilizando un único rasgo tienen una serie de limitaciones:

- Ruido: La información capturada puede ser ruidosa o puede contener distorsión. Estos hechos son el resultado de la utilización de sensores defectuosos y de condiciones ambientales adversas.

- Variación intra-clase: Los datos capturados de un individuo en la operación de autenticación pueden ser muy diferentes de los empleados en la generación de su modelo durante la operación de alistamiento.

- Variación inter-clase: La escasa variación de un rasgo biométrico entre los sujetos de una población limita su capacidad de discriminación.

- Universalidad: Es posible que un determinado subconjunto de usuarios no posea un rasgo biométrico particular.

- Ataques de impostores: Un usuario ilegítimo puede intentar falsificar el rasgo biométrico de otro legítimo para eludir el sistema.

Las limitaciones impuestas por los sistemas unimodales pueden ser solventadas mediante la utilización de múltiples modalidades biométricas [Puente et al,2008]. Los sistemas biométricos multimodales utilizan varios rasgos para realizar el reconocimiento de individuos [Hong et al, 1999].

La Figura 2.9 [Jain et al, 2004] recoge los diferentes escenarios de operación de los sistemas multimodales. Estos sistemas se diseñan para trabajar de cinco

formas diferentes y aunque algunas de estas formas no implican múltiples modalidades, implican la fusión de información en algunos puntos. Los posibles esquemas de combinación son los siguientes:

- Un rasgo, múltiples sensores: Se adquiere un único rasgo mediante diferentes sensores y se combina para mejorar el proceso de reconocimiento.

- Múltiples rasgos: Se obtienen y combinan diferentes características biométricas de una persona .

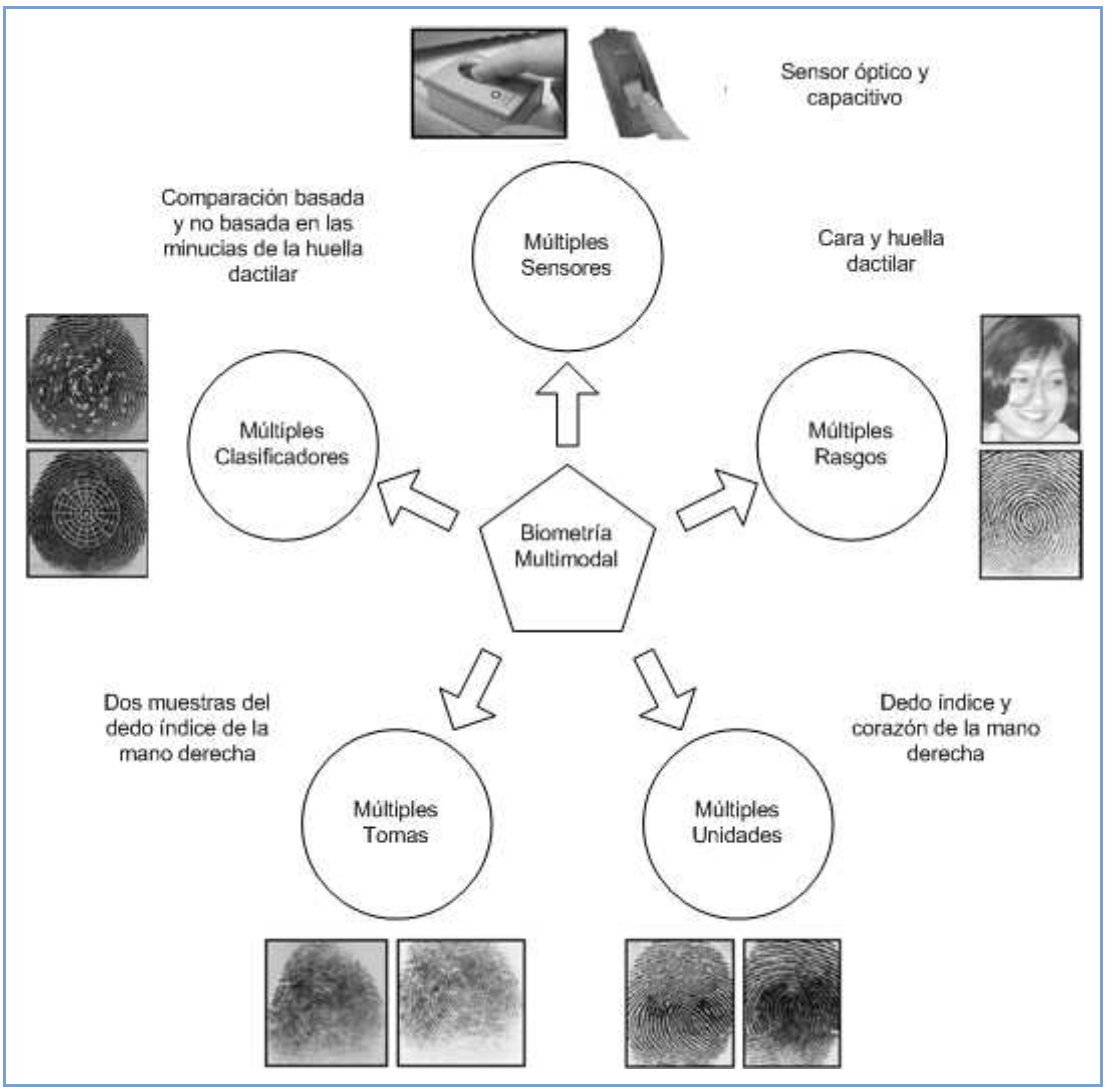


Figura 2.9. Escenarios de Operación de los Sistemas Multimodales.

- Un rasgo, múltiples unidades: Se adquieren y combinan diferentes unidades del mismo rasgo.

- Un rasgo, múltiples representaciones: Se adquieren múltiples tomas de un mismo rasgo por el mismo sensor.

- Un rasgo, múltiples clasificadores: Se obtiene un rasgo a través de un

único sensor y mediante diferentes técnicas de extracción de características y/o de comparación de patrones se realiza el proceso de reconocimiento.

A partir de los escenarios descritos y de los componentes de un sistema biométrico existen diferentes posibilidades de fusión [Jain et al, 2005]:

- Fusión a nivel de extracción de características: En este nivel se combinan las características extraídas de los rasgos biométricos en un único vector.

- Fusión a nivel de puntuaciones: En este nivel se combinan las puntuaciones que describen las similitudes entre los datos biométricos adquiridos y los modelos de cada sistema biométrico. Este modo de fusión requiere que las puntuaciones de los subsistemas se encuentren normalizadas en dominio común. La fusión a este nivel es la preferida debido a la facilidad de acceso y de combinación que ofrecen las puntuaciones.

- Fusión a nivel de decisiones: En este nivel se combinan las decisiones tomadas por cada sistema biométrico para obtener la decisión final.

2.5. EVALUACIÓN DE LOS SISTEMAS BIOMÉTRICOS

Desde hace varios años, la comunidad científica y las organizaciones comerciales han venido ofreciendo medidas de prestaciones para los sistemas biométricos basadas en bases de datos propias y protocolos de evaluación no estandarizados [Capelli et al, 2006]. Por este motivo, la mayoría de los resultados no son susceptibles de ser comparados y a menudo carecen de importancia.

Un protocolo de evaluación [Phillips et al, 2000] determina tanto la forma en la que se testea y se miden las prestaciones de un sistema biométrico como la información utilizada para evaluarlo. En la actualidad, los procesos de evaluación más importantes se encuentran administrados por grupos independientes y la información utilizada para testear los sistemas no ha sido utilizada nunca antes. Para que un protocolo de evaluación sea aceptado por la comunidad científica, los detalles de sus procedimientos han de estar publicados junto con el protocolo de medidas de prestaciones y algunos ejemplos representativos del conjunto de datos.

2.5.1. Iniciativas

En los últimos años se han desarrollado diferentes iniciativas dentro de la comunidad científica en un claro esfuerzo por ofrecer un marco común de evaluación para los sistemas biométricos. Las principales iniciativas son las siguientes:

- El Instituto Nacional de Estándares y Tecnología[2] de Estados Unidos viene desarrollando desde hace años diferentes competiciones para evaluar algoritmos y tecnologías de reconocimiento facial [Phillips et al, 2003], de locutor [Martin et al, 2004] y de iris.

- La Universidad de Bolonia desarrolla desde el año 2000 una competición bianual, la Fingerprint Verification Competition[3], para evaluar el estado del arte de los sistemas de verificación basados en huella dactilar [Capelli et al, 2006].

- El Instituto de Ciencias de la Academia China de las Ciencias desarrolla desde el año 2004 competiciones para evaluar las prestaciones de los sistemas de verificación de identidad basados en iris. Así mismo, desarrolla la Signature Verification Competition para evaluar los sistemas de verificación de identidad basados en firma on-line.

- El Face Recognition Vendor Test[4] desarrolla competiciones para la evaluación de sistemas comerciales basados en geometría facial desde el año 2000.

2.5.2. Tipos de Evaluaciones

La evaluación de un sistema biométrico requiere la captura de información. Ésta se utiliza tanto en la generación de modelos en la etapa de entrenamiento como en la obtención de puntuaciones en la etapa de reconocimiento [Capelli et al, 2006]. La información capturada puede ser empleada inmediatamente o puede ser almacenada para ser utilizada posteriormente. Los tests de evaluación de los sistemas biométricos se pueden clasificar en dos grupos [Mansfield et al, 2002]:

- Evaluación on-line: La realización de las pruebas requiere la presencia del usuario y su alistamiento o el cálculo de su puntuación se realiza inmediatamente tras la captura de la información biométrica. Como ventaja, el sistema solamente almacena la información estrictamente necesaria puesto que la muestra se elimina tras su utilización. Desafortunadamente, una misma operación no puede ser repetida debido a la falta de la muestra descartada.

- Evaluación off-line: En este segundo tipo, las pruebas se basan en información capturada y almacenada previamente. Este tipo de evaluación requiere un mayor almacenamiento de información pero a cambio permite reproducir un mismo test bajo idénticas condiciones.

Por su parte, los tests off-line se pueden clasificar en uno de los siguientes grupos [Capelli et al, 2006]:

[2] http://www.nist.gov/index.html
[3] http://bias.csr.unibo.it/fvc2006/
[4] http://www.frvt.org/

- Test propio en entorno local (In-house – Self-defined): La recolección de los datos biométricos y la definición del protocolo de prueba son realizadas directamente por el investigador. Este entorno no permite realizar ningún tipo de comparación puesto que el test no es reproducible por terceras partes.

- Test predefinido en entorno local (In-house – Existing benchmark): El test se realiza sobre una base de datos de carácter público de acuerdo a protocolos existentes. En este caso, los resultados obtenidos son comparables con otros obtenidos mediante el mismo protocolo y la misma base de datos. El principal inconveniente de este tipo de tests reside en la imposibilidad de predecir las verdaderas prestaciones del sistema en aplicaciones reales debido al posible sobreajuste sobre los datos utilizados.

- Test independiente y débilmente supervisado (Independent – Weakly supervised): La base de datos no se hace pública hasta el comienzo de la evaluación. Por un lado, el test se realiza en el entorno del investigador mediante datos sin etiquetar y con limitaciones temporales. Por otro lado, las prestaciones del sistema se determinan por evaluadores independientes a partir de las puntuaciones obtenidas por el investigador durante la prueba del sistema.

- Test independiente y supervisado (Independent – Supervised): La presente aproximación es muy similar a la anterior. Sin embargo, en este caso el test se realiza en la localización del evaluador sobre el hardware del investigador.

- Test independiente y fuertemente supervisado (Independent – Strongly supervised): En este último escenario, la base de datos no se hace pública hasta la conclusión de la evaluación. Los algoritmos de los investigadores se evalúan en la ubicación del evaluador sobre su hardware. Los algoritmos se ejecutan en un entorno totalmente controlado, en el que las operaciones de entrada y salida se definen mediante un protocolo que los algoritmos han de cumplir.

Una evaluación es correcta cuando las puntuaciones que ofrece permiten diferenciar entre las tecnologías y enfoques existentes. Las fortalezas y debilidades encontradas durante un proceso de evaluación indican las aplicaciones para las que las tecnologías son más aptas. A continuación se describen los principales tipos de evaluación [Phillips et al, 2000]:

- Evaluación de tecnología: Es el más general de los tipos de evaluación. Normalmente se realiza sobre algoritmos o prototipos de laboratorio para determinar el estado del arte, progresos tecnológicos o identificar los avances más prometedores.

- Evaluación de escenarios: Valora las prestaciones generales de un sistema para un prototipo de escenario que modela un dominio de aplicación concreto. El principal objetivo de este tipo de evaluación es

determinar si una tecnología biométrica es lo suficientemente madura para alcanzar los requisitos de prestaciones en un tipo de aplicaciones. Uno de sus objetivos es determinar las prestaciones de diferentes combinaciones de sensores y algoritmos.

- Evaluación operacional: Es similar a la evaluación de escenarios. Mientras que la evaluación de un escenario valora un determinado tipo de aplicación, la operacional mide las prestaciones de un algoritmo específico en una aplicación concreta. El principal objetivo de una evaluación operacional es determinar si un sistema biométrico alcanza los requisitos de prestaciones de una aplicación específica.

2.5.3. Medidas de Rendimiento

Existen diferentes indicadores que permiten evaluar el rendimiento de los sistemas biométricos. Es posible obtener medidas de prestaciones en cualquiera de las tres operaciones del proceso biométrico.

En la operación de alistamiento pueden aparecer dificultades en la generación de modelos debido a problemas en la adquisición de los datos biométricos [Bolle et al, 2003]. Los siguientes parámetros ofrecen una medida de las prestaciones del proceso de alistamiento [Renesse, 2002]:

- FTA (Failure to Acquire): Porcentaje de usuarios para los que el sistema no es capaz de adquirir muestras utilizables en etapas posteriores.

- FTE (Failure to Enrollment): Porcentaje de usuarios para los que el sistema no es capaz de generar un modelo de suficiente calidad.

- TTE (Time to Enroll): Tiempo transcurrido desde la captura de la muestra a la generación del modelo de usuario.

La operación de reconocimiento se ve influenciada por factores como las condiciones de adquisición, la variabilidad del rasgo biométrico o la calidad del modelo de usuario [Prabhakar et al, 2003]. Debido a estos factores se pueden producir errores a la hora de verificar o determinar la identidad de un sujeto. En este caso, las principales medidas de calidad son las siguientes:

- FRR (False Rejection Rate): Probabilidad de que el sistema rechace a un usuario legítimo porque no es capaz de identificarlo correctamente.

- FAR (False Acceptance Rate): Probabilidad de que el sistema autentique correctamente a un usuario ilegítimo.

- TTM (Time to Match): Tiempo transcurrido desde el final de la captura de la muestra a la decisión del sistema.

Para obtener un umbral de decisión óptimo, se puede representar el comportamiento de los ratios de falsa aceptación y falso rechazo en función del umbral, tal y como muestra la Figura 2.10.

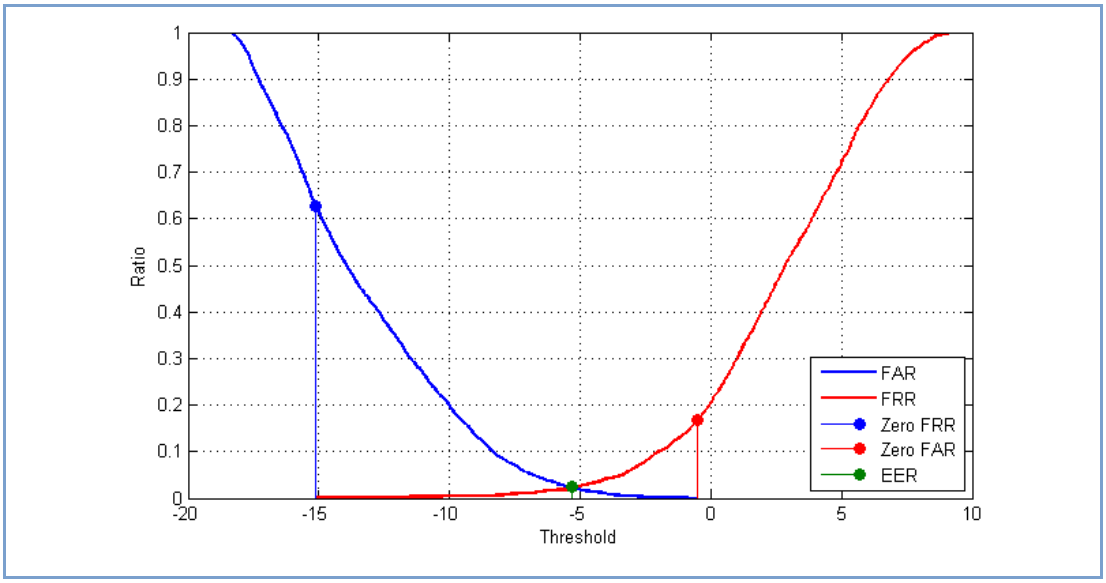


Figura 2.10. Curvas de Falso Rechazo y Falsa Aceptación.

Existen otras medidas que pueden ser usadas como una medida general de las prestaciones de un sistema biométrico:

- EER (Equal Error Rate): Hace referencia al valor del umbral de decisión para el que los valores de FAR y FRR son iguales.
- HTER (Half Total Error Rate): Es la semisuma de los valores de FAR y FRR.
- Coste: Es la suma ponderada de los valores de FAR y FRR. En este cálculo se asignan costes a los diferentes tipos de errores del sistema.

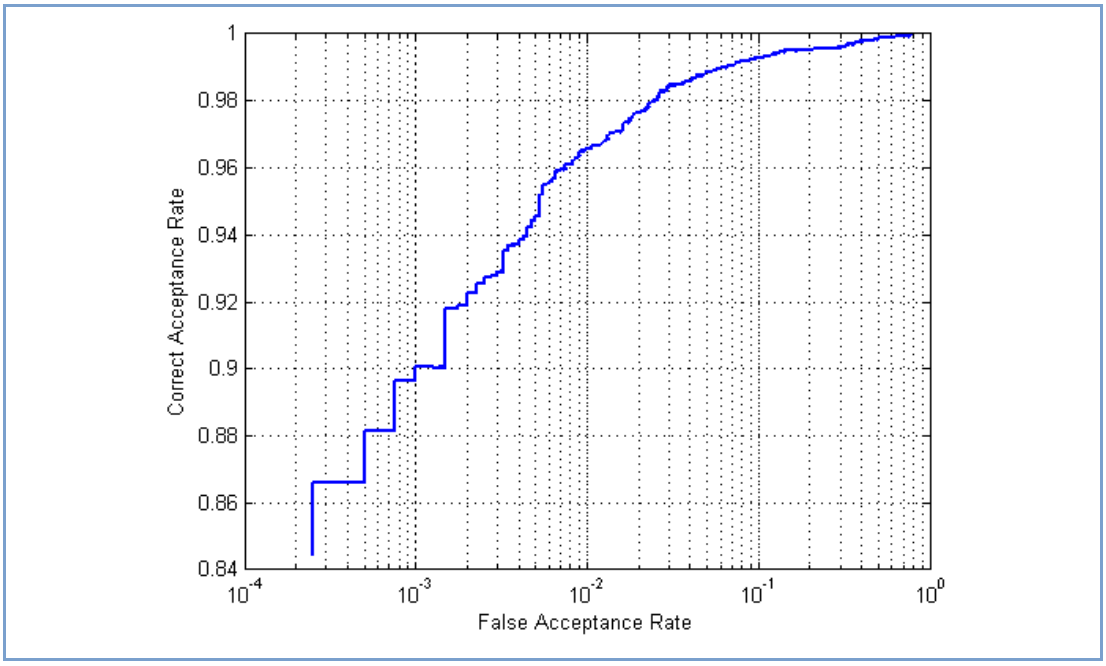


Figura 2.11. Curva ROC.

Una manera alternativa de ofrecer las prestaciones de un sistema biométrico es mediante la curva ROC (Receiver Operating Curve) [Fogarty et al, 2005]. Esta curva consiste en la representación gráfica del ratio de falsa aceptación (FAR) frente al ratio de correcta aceptación (1-FRR) en función del umbral de decisión. La Figura 2.11 muestra la curva ROC obtenida a partir de los datos de la Figura 2.10. Una de las principales ventajas de esta curva es la posibilidad de contrastar las prestaciones de diferentes sistemas en un solo gráfico.

METAMODELO

El objetivo del capítulo es ofrecer el metamodelo que defina la sintaxis, semántica e interpretación con las que construir las secuencias de operaciones que constituyen un experimento.

3.1. INTRODUCCIÓN

Un modelo es un conjunto de afirmaciones acerca de una realidad particular. Se emplean como herramientas descriptivas, pero alternativamente, se utilizan para especificar de manera formal sistemas cuyo objetivo es la generación de nuevos modelos [Seidewitz, 2003]. Su aplicación en el desarrollo de aplicaciones no requiere el uso de ningún lenguaje de programación específico para representar los detalles del sistema; simplemente se ha de describir su funcionalidad. Este hecho mejora la productividad ya que se aíslan los elementos que no son de interés [Atkinson et al, 2003].

Un metamodelo es una herramienta de especificación de modelos y es por sí mismo el modelo de la descripción de un dominio particular. La infraestructura necesaria para su creación tiene que definir los siguientes conceptos:

- Interpretación: Criterios que relacionan las afirmaciones contenidas en los modelos con los hechos del mundo real.
- Semántica: Conceptos necesarios para la creación de los modelos.
- Sintaxis: Notación para construir los modelos.
- Transformaciones: Elementos que permiten la correspondencia entre modelos de diferentes ámbitos.

Es importante describir las aplicaciones independientemente de su implementación y puesto que éstas son dinámicas, su formalización conceptual ha de representar su comportamiento [Saake et al, 1993]. Los elementos básicos del modelado conceptual de aplicaciones independientes de la situación son los siguientes:

- Abstracción de los detalles de la implementación tanto en estructuras de datos como en funcionalidades computacionales.

- Semántica formal para conseguir una verificación consistente de la implementación.
- Marco formal para modelar la estructura y el comportamiento.
- Integración de funciones heterogéneas con las que construir sistemas.
- Formalización de las conexiones existentes entre los diferentes componentes.

3.2. OBJETIVO

Tal y como se ha presentado en la sección 2.2, el proceso biométrico puede ser descrito como una serie de operaciones sobre un flujo lineal de datos. Un experimento pretende simular este proceso en un entorno controlado con el objetivo de valorar los algoritmos utilizados. En otras palabras, es una sucesión de operaciones que transforman una información biométrica en otra con el fin de obtener las prestaciones de los propios procesos. Conforme a esto, un experimento se fundamenta en dos elementos básicos: datos y operaciones.

Los datos engloban a cualquier tipo de información que se utiliza o genera durante la ejecución de un experimento. A medida que avanza la ejecución, éstos se vuelven menos voluminosos y más específicos. De forma más concreta, cualquier tipo de información presente en un experimento se puede clasificar como:

- Muestra: Información obtenida directamente de los rasgos fisiológicos o comportamentales de un sujeto. Un fragmento de voz o la captura de una imagen de iris son ejemplos de datos pertenecientes a este tipo.
- Vector de Características: Datos obtenidos a partir de la transformación matemática de las muestras biométricas de un individuo.
- Modelo: Conjunto de datos que definen el tipo o la identidad de un usuario.
- Puntuación: Valoración numérica que asigna un sistema de comparación de patrones a un conjunto de características biométricas.
- Decisión: Evaluación realizada por un sistema de clasificación acerca de la identidad de un sujeto.
- Resultado: Datos que contienen medidas de prestaciones para un sistema. A este tipo pertenecen las curvas ROC o las matrices de confusión.

Conceptualmente, los datos pueden ser agrupados en flujos con el fin de modelar su transporte entre las distintas operaciones. La información biométrica de los flujos puede ser estructurada como conjuntos de vectores de

datos.

Las operaciones representan a todos aquellos procedimientos que manipulan datos dentro de un experimento. Es posible diferenciar dos tipos de operaciones:

- De control: Procedimientos que manipulan flujos de datos para modificar su estructura.
- De transformación: Operaciones que procesan información mediante la aplicación de funciones o algoritmos matemáticos.

Las operaciones son elementos que generan datos a partir de los recibidos de uno o varios flujos. Por este motivo, pueden ser asimiladas como nodos dentro de un experimento.

La Figura 3.1 muestra el diagrama esquemático del entorno de evaluación. A partir de una base de datos biométricos y un protocolo de pruebas se puede construir un modelo para definir la ejecución un experimento. Su objetivo es simular el comportamiento del sistema biométrico y evaluar los algoritmos empleados.

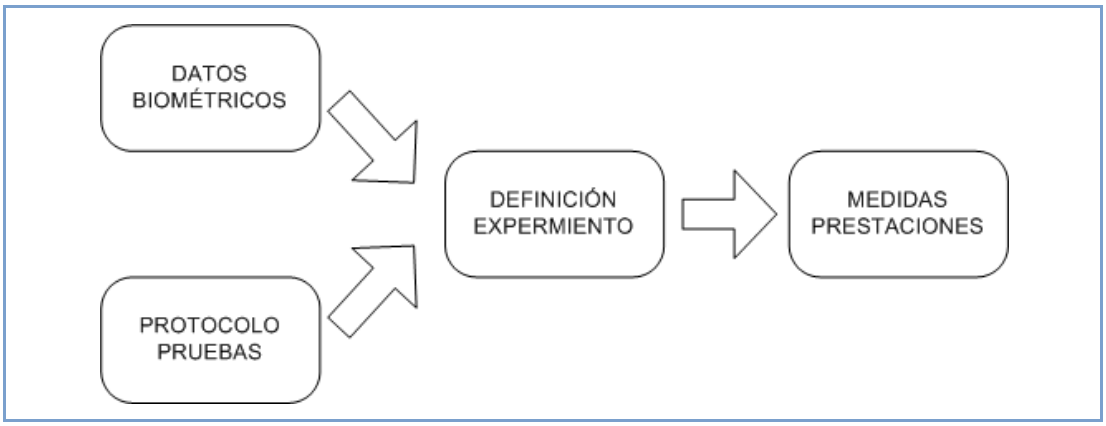


Figura 3.1. Marco de Evaluación de un Sistema Biométrico.

El objetivo del metamodelo propuesto es establecer las bases para la descripción de los datos y las operaciones que componen los experimentos de las evaluaciones de tecnología off-line (véase la sección 2.5.2). Las bases del proceso de experimentación tienen que establecer métodos capaces de describir las etapas de extracción de características, comparación de patrones, generación de modelos y en general, todas aquellas que forman parte de las operaciones de alistamiento, verificación y reconocimiento (véase sección 2.2.2). Puesto que el objetivo es modelar la evaluación off-line, no se pretende reproducir la adquisición de los datos biométricos (véase sección 2.2.1).

3.3. CONTEXTO ALGEBRAICO

Si se analiza un experimento biométrico desde el punto de vista algebraico, es

posible describirlo como una secuencia de operaciones sobre un conjunto de vectores de datos. Un vector es un conjunto ordenado de datos que puede contener valores de tipo escalar, texto, tuplas, conjuntos de vectores, etc.

Un experimento biométrico no es más que la transformación de un conjunto de vectores de datos en otro conjunto de vectores de resultados [Puente, 2007]. En su contexto, Σ determina el espacio de vectores que se pueden formar de manera homogénea, Ω define un conjunto cualquiera de vectores y por último, Δ establece el dominio de un espacio. La Figura 3.2 recoge la descripción algebraica de un experimento mediante la notación propuesta.

Figura 3.2. Definición Algebraica de un Experimento.

La transformación de los datos durante un experimento se puede dividir mediante descomposición en operaciones básicas [Puente, 2007]. Estas operaciones son procesos de alto nivel y por lo tanto, pueden ser descritas mediante una serie de primitivas con el fin de soportarlas.

3.3.1. Identidad

La transformación primitiva obtiene un conjunto de vectores idéntico al dado asignando a cada vector de su mismo vector .

Figura 3.3. Primitiva Identidad.

3.3.2. Proyección

La transformación primitiva obtiene el conjunto de vectores a partir de la reducción dimensional del conjunto según los índices de los campos especificados en el conjunto .

Figura 3.4. Primitiva Proyección.

3.3.3. Selección

La transformación primitiva obtiene el subconjunto a partir de los vectores del conjunto que cumplen la condición dada.

Figura 3.5. Primitiva Selección.

3.3.4. Asociación

La transformación primitiva genera un nuevo conjunto de vectores a partir de N conjuntos . Los vectores de se obtienen como la concatenación de aquellos vectores de los conjuntos de partida que cumplen la condición dada.

Figura 3.6. Primitiva Asociación.

3.3.5. Agrupación

La transformación primitiva permite obtener un recubrimiento del área de datos según una determinada condición .

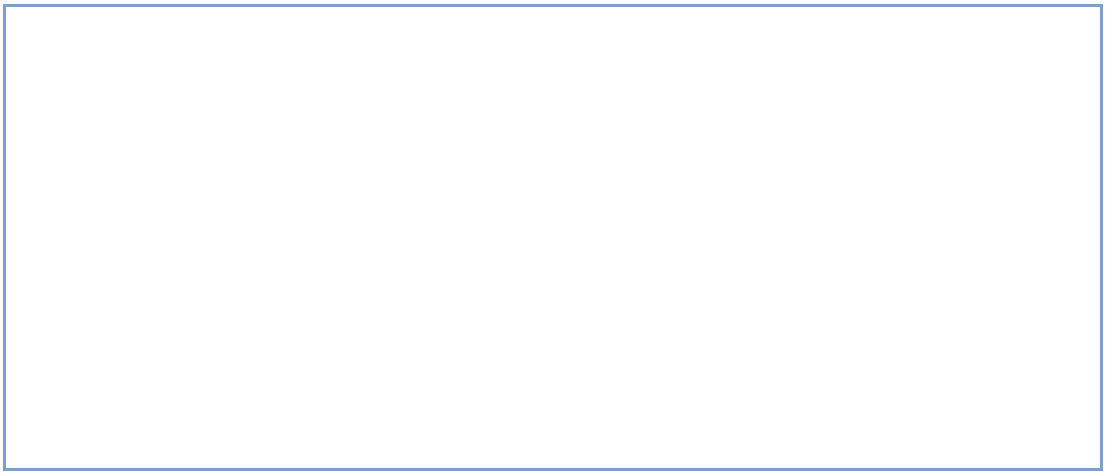


Figura 3.7. Primitiva Agrupación.

3.3.6. Procesado

La transformación primitiva genera un conjunto a partir del procesado individual, según la función , de cada uno de los vectores del conjunto . Cada uno de los vectores de salida contiene los elementos de los vectores de partida y además, incorpora los nuevos elementos generados por la función.

Figura 3.8. Primitiva Proceso.

3.4. DEFINICIÓN

El metamodelo persigue la sencillez en las definiciones de los experimentos a través de la notación gráfica. Un mismo modelo consta de dos vistas: una gráfica y otra formal.

La representación gráfica de un experimento se denomina EPC (Experiment Process Chart). Su principal objetivo es simplificar la construcción y edición de modelos. EPC se centra en la descripción de la cadena de procesos que suceden durante la ejecución y se basa en los diagramas de flujo o actividad de UML (Unified Modeling Language) [OMG, 2009].

La representación detallada se designa mediante el acrónimo BED (Biometric Experiment Definition). El objetivo de ésta es ofrecer la descripción completa de las operaciones y los datos implicados en la ejecución de un experimento y se fundamenta en el lenguaje XML (eXtended Markup Language) [W3C, 2004].

Tal y como se ha presentado en la sección 3.1, es necesario definir una serie

de conceptos con el objetivo de obtener un metamodelo consistente. En las siguientes páginas se desarrolla la interpretación, la semántica y la sintaxis.

3.4.1. Interpretación

De manera introductoria, en el modelo de ejemplo de la Figura 3.9 se representa el proceso de entrenamiento y verificación de la identidad para uno o varios usuarios.

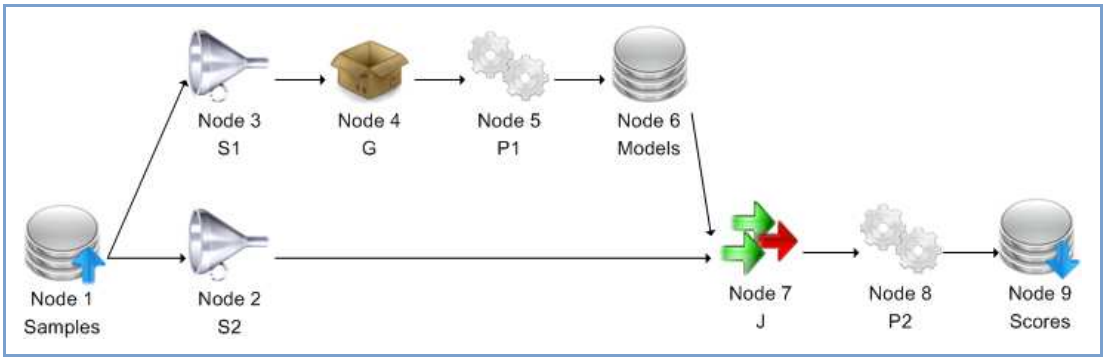


Figura 3.9. Ejemplo del Modelo de un Experimento.

Este experimento no es más que la transformación de un conjunto de muestras biométricas (“Samples”) en un conjunto de puntuaciones (“Scores”) y puede ser descrito a través de los siguientes pasos:

1. Se parte de las muestras almacenadas de cierta característica biométrica en forma de vectores de datos (Nodo 1 “Samples”).

2. De todos los vectores almacenados, se seleccionan unas muestras para la etapa de entrenamiento y otras para el proceso de identificación (Nodo 2 “S_2” y Nodo 3 “S_1”).

3. Los vectores de entrenamiento son agrupados (Nodo 4 “G”), procesados mediante un algoritmo de generación de modelos (Nodo 5 “P_1”) y almacenados (Nodo 6 “Models”).

4. Se fusionan los vectores de características seleccionados por el nodo 2 con los modelos generados (Nodo 7 “J”).

5. Los vectores de características se procesan junto con los modelos mediante un algoritmo de clasificación de patrones (Nodo 8 “P_2”).

6. Las puntuaciones generadas por el algoritmo son almacenadas (Nodo 9 “Scores”).

El metamodelo se define de acuerdo al modelo algebraico presentado en la sección 3.3 con el objetivo de dar soporte conceptual a las operaciones presentadas en el ejemplo anterior. En la Figura 3.10 se presenta la descripción formal del metamodelo mediante un diagrama de clases. Cada una de las operaciones definidas lleva asociado un conjunto de puntos de entrada y

de salida, cuya cardinalidad depende de la propia operación.

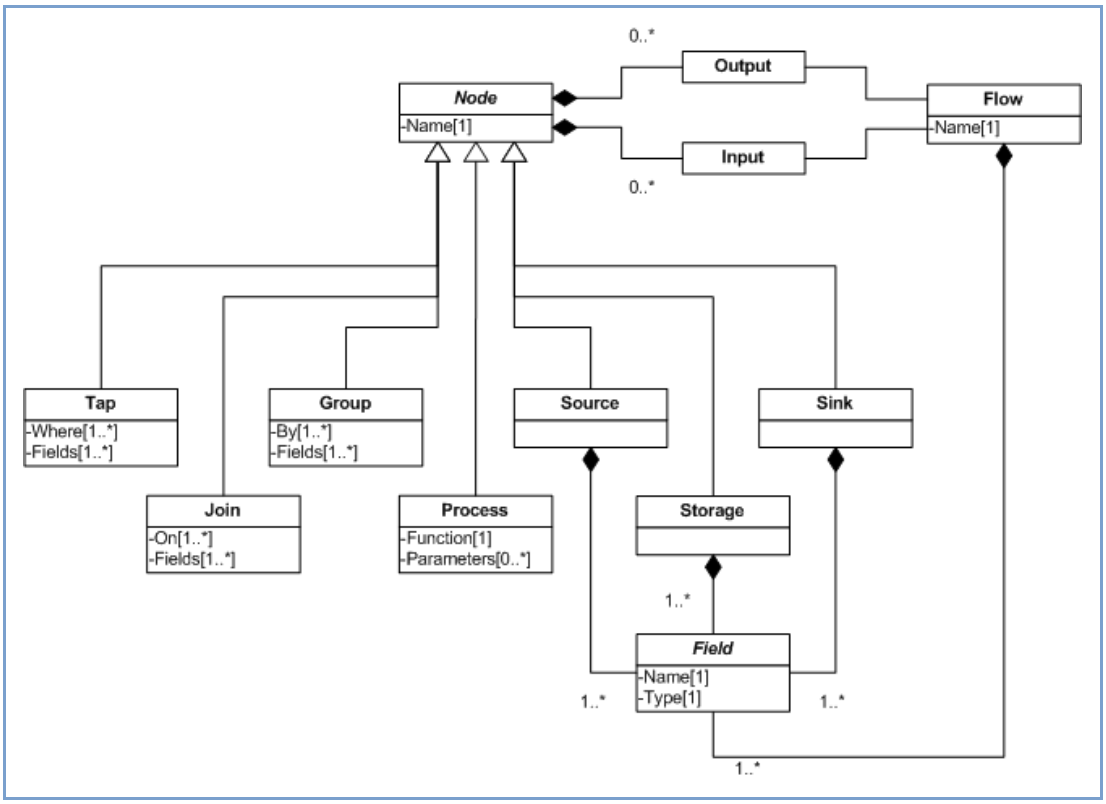


Figura 3.10. Diagrama de Clases del Metamodelo.

3.4.2. Semántica

El metamodelo propuesto se orienta hacia la descripción de la ejecución de un experimento mediante la sucesión de operaciones sobre flujos de vectores. En el modelo de un experimento se pueden distinguir dos tipos de elementos. Por un lado, los nodos (asociados a las operaciones) establecen puntos para el control, procesado y almacenamiento de la información y por otro, los flujos (relacionados con los datos) definen caminos para la transferencia de vectores entre dos nodos.

3.4.3. Sintaxis

Finalmente se presenta la sintaxis propuesta para describir los elementos del metamodelo.

3.4.3.1. Flujo

Un flujo es un elemento que modela el transporte de datos desde la salida de un nodo hacia la entrada de cualquier otro. Un flujo queda caracterizado por:

- El nodo origen que inyecta vectores.
- El nodo destino que los recibe.

- La estructura del flujo y los propios datos.

3.4.3.2. Nodos

Un nodo es un elemento que modela la generación, el control, el procesado o el almacenamiento de datos en un experimento. A continuación se ofrece la descripción de cada uno de los nodos definidos en el metamodelo. La Figura 3. 11 ofrece los elementos visuales asociados a las operaciones.

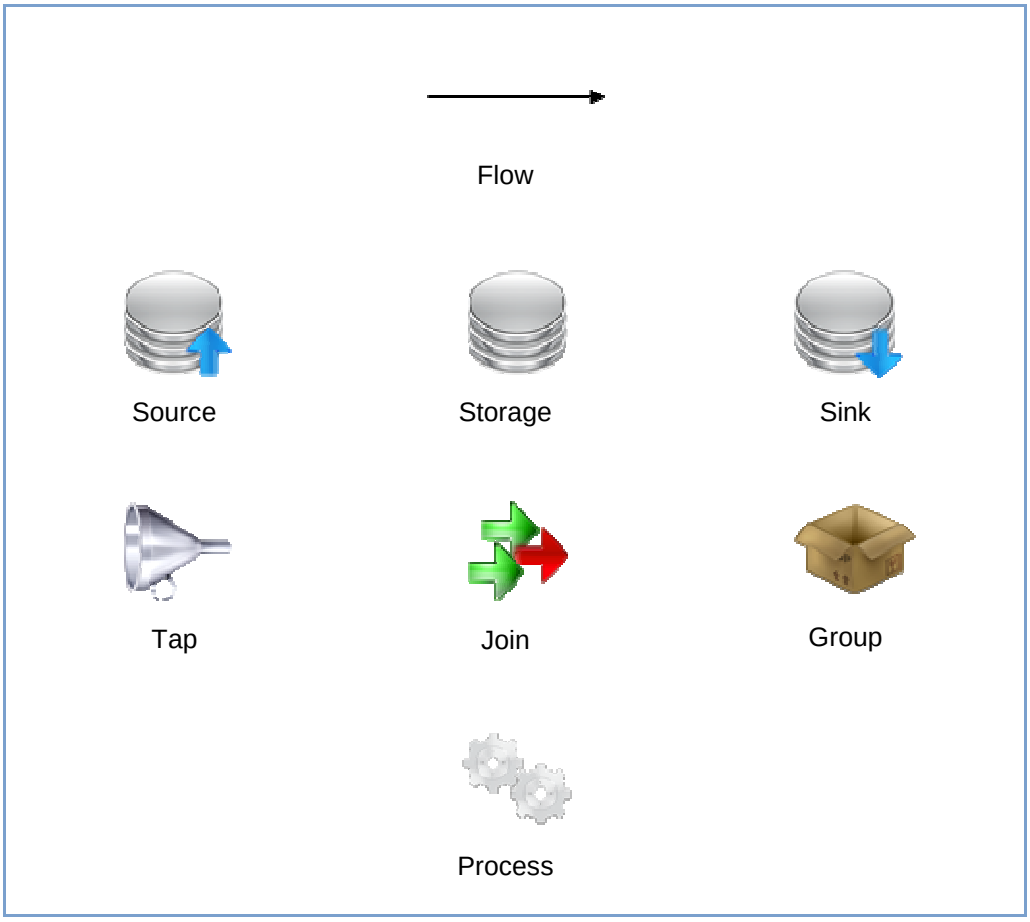


Figura 3. 11. Elementos Visuales Asociados a las Operaciones.

Nodos Source, Storage y Sink

Estos nodos modelan las propiedades tanto de las bases de datos que se emplean para evaluar un sistema como de la información biométrica presente en el experimento. Los almacenes de información biométrica representan la abstracción del lugar físico donde se almacenan vectores homogéneos. Un almacén de información no es más que una tabla desde la que se extraen o insertan datos durante la ejecución de un experimento.

El nodo Source establece la obtención de datos desde un almacén de información. Este nodo solamente posee flujos de salida puesto que es un punto de inicio en la ejecución. Así mismo, dentro de un mismo modelo pueden aparecer varias fuentes de datos representadas por diferentes nodos de tipo Source.

El nodo Sink modela el volcado de datos hacia un almacén de información. Como su propio nombre indica, es un punto final en la ejecución de un experimento y por lo tanto, no posee flujos de salida.

El nodo Storage define el almacenamiento de información temporal durante un experimento. El nodo presenta un único flujo de entrada, cuyo contenido se almacena hasta el final de la ejecución, y uno o más flujos de salida en los que se replica su contenido. Así mismo, ofrece la posibilidad de volcar el flujo de entrada en un almacén de información tras la ejecución.

Nodo Tap

El nodo Tap modela la aplicación de un filtro sobre un flujo de datos y por lo tanto, posee una entrada y una salida. Esta última contiene aquellos vectores que cumplen los criterios de selección definidos en el atributo . Así mismo, permite realizar una proyección de los vectores mediante el atributo .

Figura 3. 12. Definición Algebraica del Nodo Tap.

Nodo Join

El nodo Join establece las propiedades de la asociación de varios flujos. Las relaciones de igualdad o criterios de combinación que se tienen que dar entre los vectores se definen mediante el atributo . Al igual que el nodo Tap, el presente nodo permite definir una proyección de los vectores de salida mediante el atributo

Figura 3. 13. Definición Algebraica del Nodo Join.

Nodo Group

El nodo Group define la agrupación de los vectores de un flujo según unos determinados criterios. La agrupación de los vectores se realiza de acuerdo a los valores de los campos definidos por el atributo . El nodo permite agrupar los vectores de un flujo, según los valores que toman ciertos campos, para generar diferentes subconjuntos a la salida. De nuevo es posible realizar una proyección de los vectores de entrada de acuerdo a los campos listados en el atributo .

Figura 3. 14. Definición Algebraica del Nodo Group.

Nodo Process

El nodo Process modela el procesado de los vectores contenidos en un flujo mediante la aplicación de algoritmos y funciones. Este nodo es el elemento principal en la definición de cualquier experimento puesto que representa la aportación externa de los algoritmos a evaluar. Mientras que los nodos anteriores se centran en la selección y la agrupación de los datos biométricos, este nodo lo hace en el procesado individual de cada uno de los vectores que forman parte de un determinado flujo.

Figura 3. 15. Definición Algebraica del Nodo Process.

De forma general el nodo Process posee una entrada y una salida; sin embargo, puede comportarse como una fuente de datos. En estos casos, el nodo no necesita el flujo de entrada.

PLATAFORMA BEEP

Este capítulo ofrece la visión general de la plataforma de experimentación y presenta las restricciones impuestas a su implementación. Así mismo, recoge la definición de los subsistemas y los elementos comunes a ellos.

4.1. INTRODUCCIÓN

BEEP surge como una aplicación web capacitada para ofrecer servicios de experimentación biométrica a los usuarios bajo un entorno común. La plataforma está compuesta por dos subsistemas: el Front-End y el Back-End.

El Front-End es el subsistema encargado de la interacción con el usuario. Ofrece las funcionalidades necesarias para crear y modificar modelos de experimentos y presentar informes de resultados de los sistemas evaluados. Así mismo, el Front-End brinda mecanismos para incorporar descripciones de procesos y de datos biométricos a los modelos.

El Back-End es el subsistema responsable de la ejecución de los experimentos. Contiene las funcionalidades necesarias para interpretar y ejecutar los modelos de los experimentos. De la misma manera, es responsable de la generación de los informes de prestaciones.

El diagrama de la Figura 4. 1 muestra la interacción entre los elementos de la plataforma. A continuación se describe, de forma general, la relación entre el usuario y la aplicación:

1. El usuario define el modelo de un experimento al que incorpora las descripciones de datos biométricos y procesos.

2. El Front-End almacena la información relacionada con el experimento en el modelo de datos de la plataforma.

3. El Front-End notifica al Back-End de la solicitud de ejecución del experimento a través de un interface de comunicación.

4. El Back-End recupera del modelo de datos los recursos necesarios para la ejecución.

5. El Back-End ejecuta el experimento y almacena en el modelo de datos los resultados generados.

6. El Front-End accede al modelo de datos para recuperar el informe de prestaciones del sistema evaluado.

7. El usuario obtiene el informe de prestaciones del sistema biométrico evaluado.

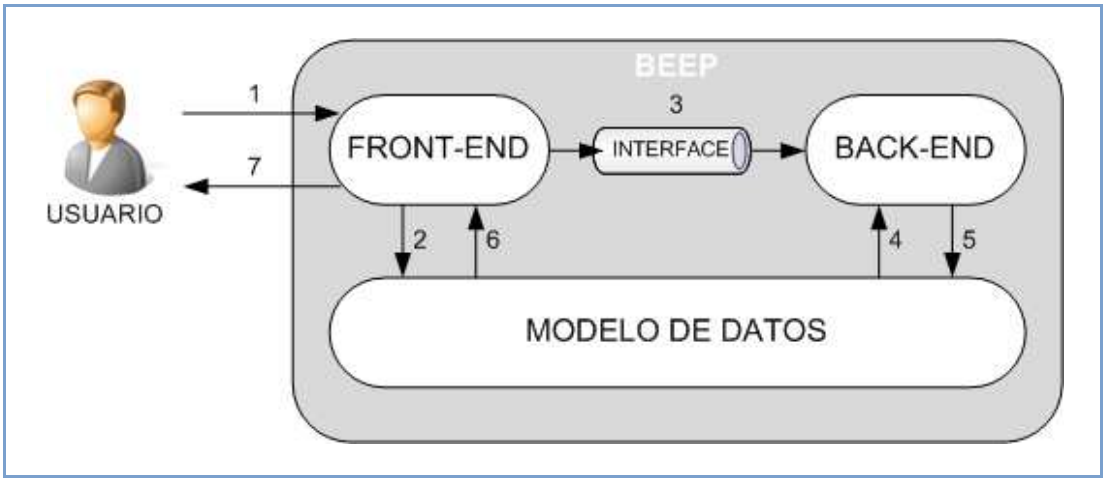


Figura 4. 1. Visión General de la Plataforma.

Para dar soporte a los dos subsistemas existe una serie de componentes comunes:

- Modelo de datos: La capa de persistencia alberga los datos de la plataforma y es utilizada por ambos subsistemas (véase la sección 4.4).

- Interface de datos: El metamodelo presentado define el interface de datos entre el Front-End y el Back-End (véase la sección 4.5).

- Interface de comunicación: Establece el protocolo de mensajes entre los dos subsistemas (véase la sección 4.6).

4.2. ESPECIFICACIÓN DE REQUISITOS

En las siguientes páginas se ofrece la especificación de requisitos de la plataforma. Ésta recoge los requisitos de capacidad y de restricción para la plataforma y establece el punto de partida para el análisis funcional del Back-End (véase el *Capítulo 5: BACK-END. ANÁLISIS*).

4.2.1. Requisitos de Capacidad

Los requisitos de capacidad son aquellas funciones requeridas por los usuarios para resolver un problema o alcanzar un objetivo [ESA, 1991]. La Tabla 4.1 ofrece los requisitos de capacidad de la plataforma (denominados RC).

ID	ESENCIAL	MODIFICABLE	PRIORIDAD
RC-001	Si	No	Alta
DESCRIPCIÓN: La plataforma ofrecerá servicios con independencia de la ubicación del usuario.			
RC-002	Si	No	Alta
DESCRIPCIÓN: La plataforma ofrecerá mecanismos para crear y modificar modelos de experimentos biométricos			
RC-003	Si	Si	Alta
DESCRIPCIÓN: La plataforma ofrecerá mecanismos para modelar datos biométricos con independencia de su origen.			
RC-004	Si	No	Alta
DESCRIPCIÓN: La plataforma tendrá capacidad para asegurar la información biométrica almacenada.			
RC-005	Si	No	Alta
DESCRIPCIÓN: La plataforma ofrecerá mecanismos para modelar algoritmos de tratamiento de información biométrica.			
RC-006	Si	No	Alta
DESCRIPCIÓN: La plataforma tendrá capacidad para interpretar y ejecutar modelos de experimentos biométricos.			
RC-007	Si	Si	Alta
DESCRIPCIÓN: La plataforma tendrá capacidad para generar y presentar informes de prestaciones tras la ejecución de los experimentos.			

Tabla 4.1. Requisitos de Capacidad de la Plataforma.

4.2.2. Requisitos de Restricción

Los requisitos de restricción engloban a las limitaciones impuestas por los usuarios sobre la forma en la que se resuelve un problema o alcanza un objetivo [ESA, 1991]. En Tabla 4.2 se muestran las restricciones de la plataforma (denominadas RP) impuestas por el cliente.

ID	ESENCIAL	MODIFICABLE	PRIORIDAD
RP-001	Si	No	Alta
DESCRIPCIÓN: La plataforma establece su arquitectura física como cliente-servidor.			
RP-002	Si	No	Alta
DESCRIPCIÓN: El modelo de datos de la plataforma tiene que ser común para los dos subsistemas.			
RP-003	Si	Si	Alta
DESCRIPCIÓN: El interface de datos entre los dos subsistemas queda establecido por la definición de los modelos de los experimentos.			
RP-004	Si	No	Alta
DESCRIPCIÓN: Es necesario un protocolo de comunicación entre los subsistemas.			

Tabla 4.2. Requisitos de Restricción de la Plataforma.

4.3. ARQUITECTURA

La división física de la plataforma, según establece el requisito RP-001, se asienta en una arquitectura cliente-servidor tal y como muestra la Figura 4. 2:

- Cliente: Máquina que inicia peticiones de servicio. La petición inicial puede convertirse en múltiples requerimientos de trabajo a través de peticiones basadas en HTTP (Hyper Text Transfer Protocol).
- Servidor: Máquina que ofrece los recursos de cómputo dedicados a

responder a las peticiones del cliente. El servidor está conectado a Internet con el objetivo de proveer servicios de forma simultánea a los clientes.

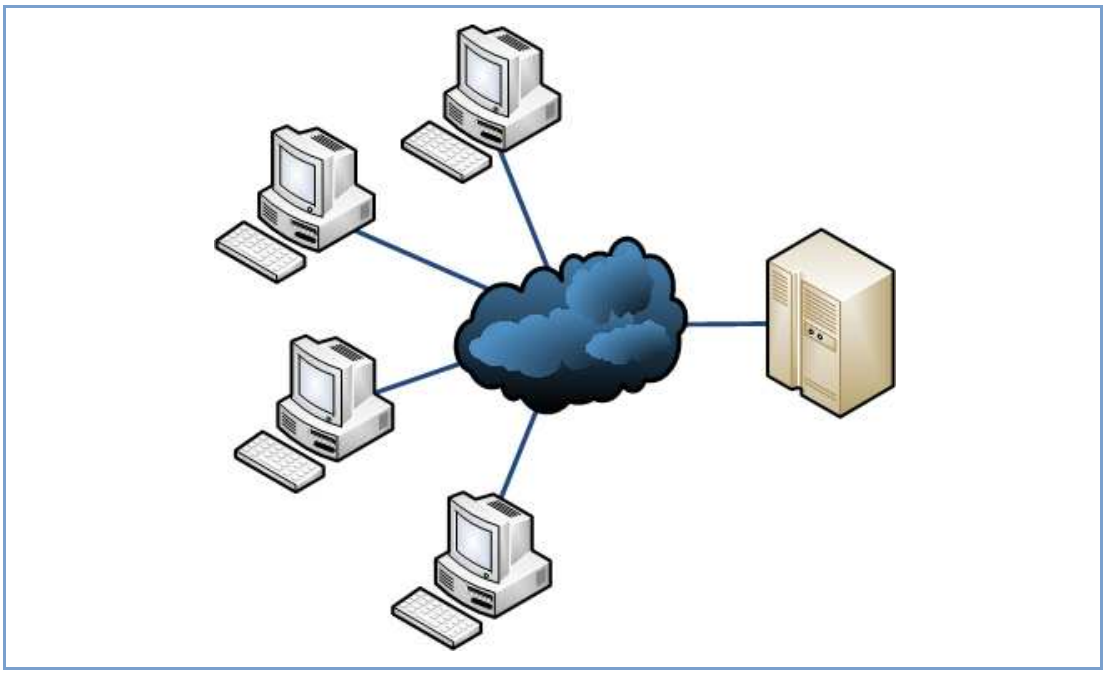


Figura 4. 2. Arquitectura Física de la Plataforma.

La infraestructura hardware de la máquina servidor se basa en un servidor HP Proliant 115. El servidor incorpora un procesador AMD Opteron Dual a 1.8 GHz, 4 GB de memoria RAM y un disco duro de 160 GB. El equipo se encuentra ubicado en un entorno estable con las medidas de seguridad adecuadas con el fin de garantizar la disponibilidad y la integridad de los servicios de la aplicación. Por su parte, la infraestructura software del servidor se basa en Microsoft Windows Server 2003[5]. Éste es un sistema operativo de propósito múltiple capaz de manejar una gran gama de funciones de servidor, tanto de manera centralizada como distribuida.

El diagrama de despliegue mostrado en la Figura 4. 3 ofrece las relaciones entre los componentes lógicos desplegados sobre la arquitectura física impuesta.

La arquitectura lógica de la máquina cliente se compone de un navegador web y de la aplicación de edición de experimentos. Las funciones principales de estos elementos son el acceso a la aplicación web y la construcción de los modelos respectivamente.

Dentro de la arquitectura lógica del servidor aparecen el Front-End y el Back-End. Así mismo, se encuentran las capas de infraestructura y persistencia. La infraestructura de la plataforma se basa en MySQL Connector para acceder a los datos de la plataforma y ODBC para aislar los orígenes de los datos

[5] http://www.microsoft.com/spain/windowsserver2003/default.mspx

biométricos de la lógica de la aplicación. Finalmente, la persistencia se consigue mediante sistemas de bases de datos como MySQL, SQL Server y Microsoft Access.

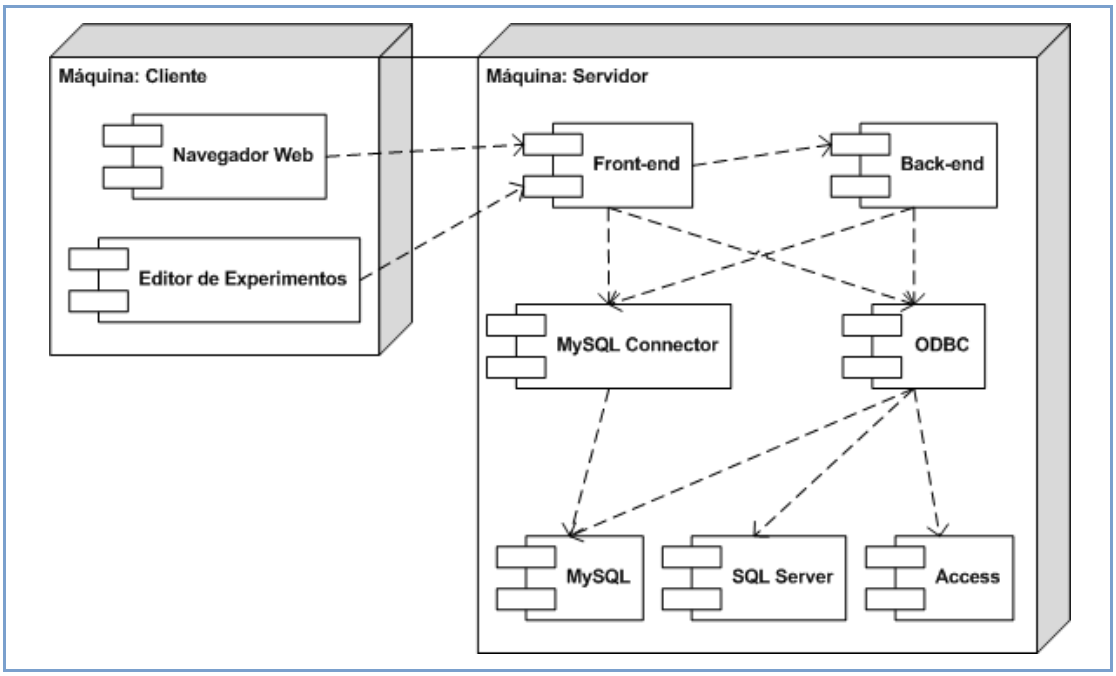


Figura 4. 3. Arquitectura Lógica de la Plataforma.

4.4. MODELO DE DATOS

En la actualidad, Internet se está convirtiendo en una plataforma de computación y en un interface común para compartir datos [Atzeni et al, 1998].

Dentro de cualquier organización son necesarias tareas como la gestión y la transmisión de información y varios de los problemas que se presentan a la hora de realizar estas tareas son las redundancias y las incongruencias en los datos almacenados. Durante la etapa de diseño, una base de datos ha de pasar por el proceso de normalización. Existen diferentes reglas de normalización y cada una de ellas se denomina forma normal [Codd, 1970]. Aunque existen diferentes niveles de normalización, la tercera forma normal se considera el máximo nivel necesario para la mayor parte de las aplicaciones [MSDN, 2007].

El modelo E-R es un conjunto de conceptos y reglas para representar de forma global los aspectos lógicos de los diferentes tipos de datos existentes en un sistema [Piattini et al, 2006]. El modelo de datos de la plataforma se ha implementado mediante diferentes bases de datos. Por un lado, se encuentra la base de datos del sistema y por otro, se encuentran las bases de datos de las entidades de investigación.

La base de datos del sistema contiene la información de las entidades de

investigación, de la cola de ejecución y de los procesos y almacenes de información públicos.

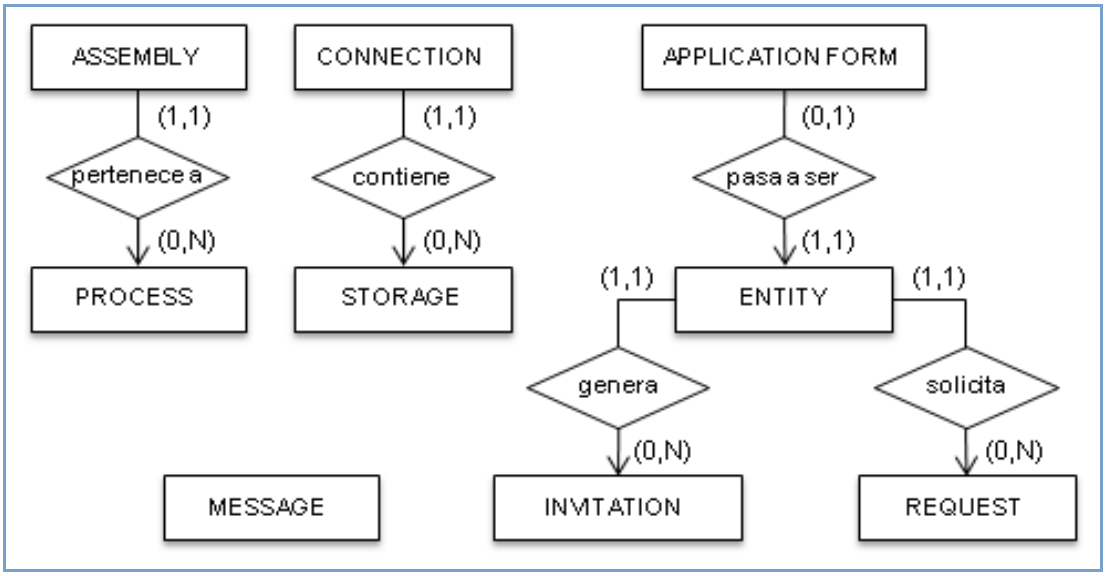


Figura 4. 4. Diagrama E-R para el Modelo de Datos del Sistema.

El modelo E-R propuesto para la base de datos del sistema, cuya representación recoge la Figura 4. 4, está formado por un total de nueve entidades conceptuales.

- Entidad Application Form: La entidad Application Form representa la solicitud de creación de una entidad investigadora.
- Entidad Entity: En esta entidad se engloba el concepto de entidad u organización investigadora, formada por usuarios y grupos de trabajo, cuyos objetivos son la realización de experimentos y la obtención de resultados.
- Entidad Invitation: La entidad Invitation engloba las invitaciones de creación de nuevos usuarios de la plataforma.
- Entidad Request: En la presente entidad se engloba el concepto de solicitud de ejecución de un experimento por parte de una entidad investigadora.
- Entidad Assembly: La entidad Assembly representa a las librerías de código que necesita un proceso público para realizar sus operaciones
- Entidad Process: La entidad Process abarca a todos aquellos elementos que intervienen en la ejecución de un experimento modificando o tratando los datos biométricos. La presente entidad modela los procesos ofrecidos por la plataforma y que por lo tanto, son de carácter público.
- Entidad Connection: La entidad Connection representa la localización de un almacén de información biométrica.

- Entidad Storage: La entidad Storage recoge el concepto de almacén público de información biométrica.

- Entidad Message: La última entidad del modelo representa a los mensajes públicos de la plataforma.

Dentro del presente modelo de datos, se han definido un total de cinco relaciones:

- Relación Application Form – pasa a ser – Entity: Esta primera relación establece la correspondencia entre las solicitudes de creación y las entidades investigadoras. La relación Application Form – pasa a ser – Entity posee una cardinalidad (0,1) ya que la creación de la entidad depende de la valoración de la solicitud de creación. En el camino inverso, se puede apreciar que la cardinalidad de la relación Entity – procede de – Application Form es (1,1) ya que una entidad investigadora solamente proviene de una solicitud de creación.

- Relación Entity – genera – Invitation: Esta relación presenta la dependencia existente entre las invitaciones de nuevos usuarios y las entidades de investigación. Una entidad de investigación puede generar cero o más invitaciones, por lo que la cardinalidad de la relación es (0,N). En el camino inverso, la relación Invitation – pertenece a – Entity posee una cardinalidad (1,1) ya que una invitación pertenece solamente a una entidad de investigación.

- Relación Entity – solicita – Request: Mediante la presente relación se asocian las entidades de investigación y las solicitudes de ejecución de experimentos. Una entidad puede poseer cero o más solicitudes de ejecución, por lo que la cardinalidad de la relación es (0,N). Si se comprueba la cardinalidad en el camino inverso, la relación Request – pertenece a – Entity posee una cardinalidad (1,1) puesto que una solicitud de ejecución pertenece a una única entidad investigadora.

- Relación Connection – contiene - Storage: Mediante esta relación se asocia cada uno de los almacenes de información con su localización física. La cardinalidad de la relación es (0,N) ya que una conexión puede no contener ningún almacén de información. La relación inversa Storage – pertenece a – Connection posee una cardinalidad (1,1) ya que un almacén de información solamente se localiza mediante una conexión.

- Relación Assembly – pertenece a – Process: La presente relación asocia los procesos biométricos y las librearías de código necesario para ejecutarlos. La cardinalidad de la relación es (0,N) puesto que una librería puede no usarse en ningún proceso. En la relación inversa Process – necesita – Assembly, la cardinalidad de la relación es (1,N) debido a que un proceso necesita como mínimo una librería de código.

La base de datos de la entidad de investigación contiene la información de sus usuarios y grupos de trabajo, así como la información de sus experimentos y

procesos y almacenes privados.

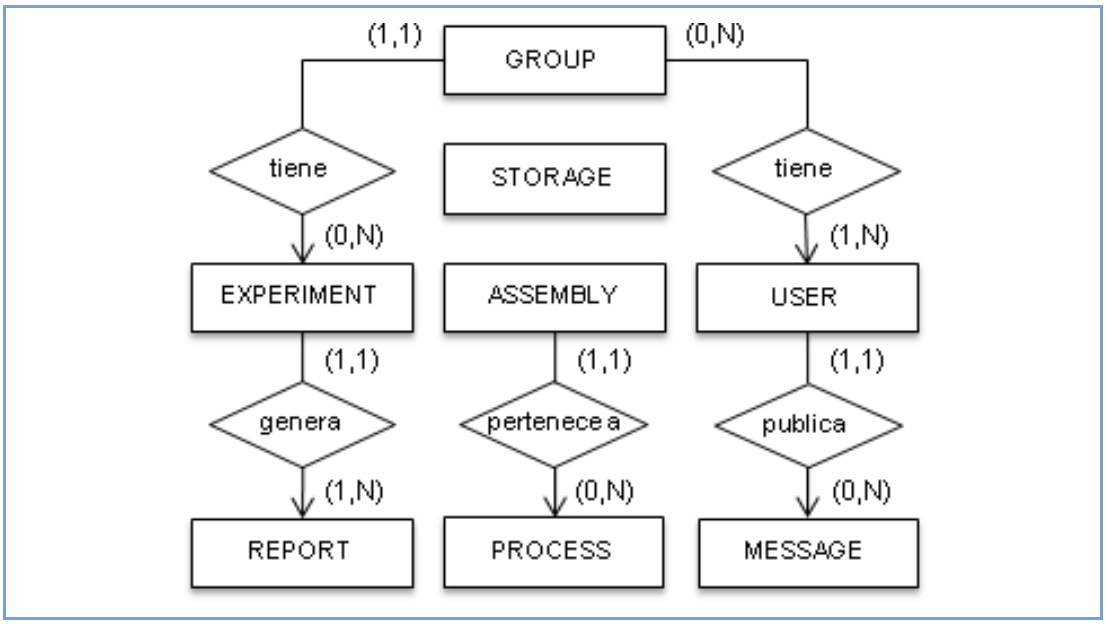


Figura 4.5. Diagrama E-R para el Modelo de Datos de la Entidad.

Junto a las descripciones de las entidades y las relaciones propuestas, se adjunta el diagrama relacional en la Figura 4.5 para completar de forma precisa la definición del modelo de datos. El modelo E-R propuesto para las bases de datos de las entidades está formado por un total de ocho entidades conceptuales:

- Entidad Group: Esta entidad hace referencia al conjunto de usuarios de una determinada entidad investigadora que trabajan de forma conjunta sobre una serie de experimentos.

- Entidad User: La entidad User recoge el concepto de personaje que accede a la plataforma, bien sea un usuario de administración o de investigación.

- Entidad Message: La entidad Message hace referencia a los mensajes privados de la entidad de investigación.

- Entidad Experiment: Un experimento biométrico queda representado dentro del modelo relacional por la entidad Experiment.

- Entidad Report: La entidad Report modela los informes de resultados generados tras la ejecución de un experimento.

- Entidad Assembly: La entidad Assembly modela las librerías de código que necesita un proceso privado para realizar sus operaciones

- Entidad Process: La presente entidad define a los procesos creados por los miembros de la entidad de investigación y que por lo tanto, son de carácter privado.

- Entidad Storage: La entidad Storage recoge el concepto de almacén de información biométrica generado tras la ejecución de un experimento. De esta forma, los almacenes privados quedan modelados por la presente entidad.

Una vez presentadas las distintas entidades conceptuales que conforman el modelo de datos, a continuación se describen las relaciones existentes entre ellas. Dentro del modelo conceptual, se han definido cinco interrelaciones:

- Relación Group – tiene – User: La presente relación describe la correspondencia que existe entre los grupos de trabajo y sus miembros. La cardinalidad de la relación es (0,N) ya que un grupo de trabajo puede no poseer ningún miembro. Por su parte, la relación User – pertenece a – Group posee una cardinalidad de (0,N) ya que un usuario de la entidad puede no pertenecer a ningún grupo de trabajo.

- Relación User – publica – Message: Mediante la relación actual se asocian los usuarios de la entidad de investigación y los mensajes privados de la misma. La cardinalidad de la relación es (0,N) puesto que un usuario puede publicar varios mensajes. En el camino inverso, la relación Message – pertenece a – User posee una cardinalidad (1,1) ya que un mensaje solamente pertenece a un usuario.

- Relación Group – tiene – Experiment: La actual relación recoge la asociación entre un grupo de trabajo y sus experimentos. La cardinalidad de la relación es (0,N) puesto que un grupo de trabajo puede no tener experimentos asociados. La relación inversa Experiment – pertenece a – Group posee una cardinalidad (1,1) ya que un experimento pertenece a un solo grupo de trabajo.

- Relación Experiment – genera – Report: La presente relación define la asociación entre los experimentos y los informes de prestaciones. La cardinalidad de la relación es (1,N) ya que como mínimo, tras la ejecución de un experimento se genera un informe de prestaciones. En la relación inversa, la cardinalidad de la relación Report – proviene de – Experiment es (1,1) debido a que un informe solamente proviene de un experimento.

- Relación Assembly – pertenece a – Process: Esta relación asocia los procesos biométricos y las librearías necesarias para ejecutarlos. La cardinalidad de la relación es (0,N) puesto que una librería puede no usarse en ningún proceso. En la relación inversa Process – necesita – Assembly, la cardinalidad de la relación es (1,N) debido a que un proceso necesita como mínimo una librería de código.

4.5. INTERFACE DE DATOS

El interface de datos queda definido por el metamodelo desarrollado (véase el *Capítulo 4.5 INTERFACE DE DATOS*) La representación detallada de un

experimento se denomina BED (Biometric Experiment Definition) y su objetivo es ofrecer la descripción completa de las operaciones y datos involucrados en la ejecución de un experimento.

Para la implementación del interface de datos se ha empleado XML por las siguientes razones:

- Es posible extender el metamodelo con nuevas etiquetas. Este hecho permite añadir funcionalidades manteniendo la compatibilidad con versiones anteriores.

- El analizador léxico es un componente estándar y no es necesario crear uno específico para cada versión del metamodelo. Este hecho posibilita el empleo de cualquier analizador disponible, se evitan errores y se acelera el desarrollo.

- Es sencillo entender y procesar la estructura del metamodelo puesto que XML está basado en texto.

La sintaxis de los modelos queda definida a través de un esquema XML, cuya representación se puede consultar en el CD adjunto. A continuación se describen sus principales elementos.

Elemento <BED>

El archivo XML posee un elemento raíz denominado <BED>. Dentro de este elemento raíz aparecen las etiquetas <Validation>, <Execution> y <Definition> para describir diferentes aspectos del modelo.

En primer lugar, el elemento <Validation> orece los resultados de las operaciones de validación. El siguiente elemento es la etiqueta <Execution>, cuyo cometido es recoger los resultados de ejecución. Su información se encuentra disponible una vez el experimento ha sido ejecutado. Finalmente, la etiqueta <Definition> describe los elementos que forman parte del modelo. Dentro del presente elemento aparecen las etiquetas <Flows> y <Nodes>. El cometido de estas etiquetas es recoger las propiedades de los flujos y nodos respectivamente.

Elemento <Flow>

El elemento <Flow> describe un flujo de datos. Ofrece una lista de componentes que detallan las propiedades de cada uno de los campos que forman parte de un flujo. Es posible diferencia dos tipos de campos:

- Value: Son valores o datos por sí mismos que se transportan directamente en un flujo. Un campo de este tipo se etiqueta como un elemento de tipo <Value>.

- Reference: No son valores por sí mismos, sino referencias a archivos donde se almacenan datos que pueden ser utilizados por los nodos. Básicamente, un campo de tipo Reference contiene la URL de un recurso

y se etiqueta como un elemento <Reference>.

Elemento <Source>

El nodo Source se representa mediante el elemento <Source>. Esta etiqueta posee un total de dos subelementos. Por un lado, el elemento <SDescriptor> describe el contenido del almacén y por otro, el elemento <Mapping> mapea los campos del almacén en los flujos de salida.

Elemento <Storage>

La etiqueta <Storage> describe las propiedades del nodo Storage. Al igual que el nodo anterior, el elemento XML asociado contiene las etiquetas <SDescriptor> y <Mapping>.

Elemento <Sink>

El nodo Sink se define mediante el elemento <Sink>. A diferencia de los anteriores nodos, su esquema XML asociado solamente posee el elemento <SDescriptor>.

Elemento <Tap>

El nodo Tap se modela mediante el elemento <Tap>. Posee un elemento <Mapping> para definir el contenido del flujo de salida y un elemento <WhereSetup> para configurar cada una de las condiciones del filtro de selección.

Elemento <Join>

El elemento <Join> recoge las propiedades del nodo Join y contiene un total de tres elementos. El elemento <Mapping> define la estructura del flujo de salida y el mapeo de campos desde las entradas. La etiqueta <OnSetup> establece los campos sobre los que se aplica la condición de asociación. Por último, el elemento <WhereSetup> configura el filtro aplicado sobre el flujo de salida.

Elemento <Group>

La etiqueta <Group> presenta los elementos <Mapping> y <BySetup> para describir el comportamiento del nodo Group. De igual forma que en los anteriores nodos, la etiqueta <Mapping> implementa la proyección de los vectores de datos. Mientras, la etiqueta <BySetup> contiene la lista de campos que se emplea como criterio de agrupación.

Elemento <Process>

El nodo Process se implementa mediante el elemento XML <Process>. La etiqueta <Process> contiene un total de tres elementos: <PDescriptor>, que describe los mecanismos de intercambio de información; <Loading>, que determina los campos de datos que recibe el proceso; y <Mapping> que establece los campos presentes a la salida del nodo.

Elemento <SDescriptor>

El contenido de los almacenes de información se describe mediante el uso de la etiqueta <SDescriptor>. Ésta define el conjunto de campos que forman parte de una fuente de datos biométricos.

Elemento <PDescriptor>

El elemento <PDescriptor> define el interface de datos de un proceso. En él se establecen los parámetros de las entradas y las salidas, así como los de configuración.

Elemento <Mapping>

Todos los nodos (salvo el nodo Sink) poseen un elemento común llamado <Mapping>. Este elemento implementa la primitiva proyección y el mapeo de campos desde las entradas a las salidas. Este elemento permite cambiar el nombre y la descripción de los campos mediante los atributos que contiene.

Elemento <WhereSetup>

El elemento <WhereSetup> implementa la primitiva selección. Está presente en los nodos Tap y Join para definir las condiciones del filtro que se aplica sobre los campos del flujo de entrada. Dentro de la etiqueta <WhereSetup> aparece una lista de elementos <Where>, donde cada uno contiene una de las condiciones del filtro.

Elemento <OnSetup>

El elemento <WhereSetup> implementa la primitiva selección. Se encuentra presente en los nodos Tap y Join para definir las condiciones del filtro de selección de los campos del flujo de entrada. Dentro de la etiqueta <WhereSetup> aparece una lista de elementos <Where>, donde cada uno contiene una de las condiciones del filtro.

Elemento <BySetup>

La primitiva de agrupación se implementa mediante el elemento <BySetup>. Esta etiqueta recoge el conjunto de campos que se emplea como criterio de agrupación. Dentro de ella aparece un elemento <By> por cada uno de los campos que forman parte del conjunto.

4.6. INTERFACE DE COMUNICACIÓN

La comunicación entre procesos es una función básica de los sistemas operativos [Silberschatz et al, 2006]. Los procesos se pueden comunicar entre sí mediante espacios compartidos de memoria o a través de las rutinas ofrecidas por el sistema operativo. La comunicación entre procesos se puede clasificar según varias propiedades:

- Sincronía: La sincronía hace referencia a la espera de los procesos durante la comunicación. Si la comunicación es síncrona, el proceso emisor permanece bloqueado hasta que recibe la respuesta del proceso receptor.

- Persistencia: La persistencia establece la disponibilidad del receptor. Si la comunicación es persistente, el proceso receptor no tiene porqué estar operativo al mismo tiempo que se realiza la comunicación. El mensaje se almacena tanto tiempo como sea necesario hasta ser entregado.

- Tipo: Si la comunicación es directa, se ha de explicitar el nombre del proceso destino. Sin embargo, si la comunicación es indirecta, el proceso destino recibe los datos a través de un puerto.

- Simetría: La simetría hace referencia a la capacidad de comunicación de los procesos. Si la comunicación es simétrica, todos los procesos pueden enviar y recibir mensajes; en cambio, si la comunicación es asimétrica, un proceso actúa como emisor y el resto como receptores.

Debido a que los subsistemas de la plataforma poseen contextos de ejecución independientes, se ha hecho necesaria la implementación de un sistema de comunicación. La comunicación entre el Front-End y el Back-End se ha implementado mediante una tubería. Ésta no es más que una sección de memoria compartida que varios procesos utilizan para su comunicación [MSDN, 2009c]. Por un lado, se encuentra el proceso que la crea (servidor) y por otro lado, aparecen los procesos que se conectan a ella (clientes). Dentro de las tuberías aparecen las tuberías con nombre [MSDN, 2009d] con el objetivo de ofrecer comunicación entre un proceso servidor y uno o varios procesos clientes. La manera de utilizar la tubería es mediante su nombre, de tal forma que cualquier proceso que conozca el nombre de la tubería puede conectarse a ella. Así mismo, cualquier proceso puede actuar como cliente y como servidor en la comunicación.

Para conseguir la máxima independencia entre los dos subsistemas se propone una comunicación asíncrona, no persistente, indirecta y asimétrica:

- Interface asimétrico: El Front-End se encarga de generar notificaciones y el Back-End se encarga de recibirlas, por lo que la comunicación es en un único sentido.

- Interface asíncrono: El Front-End no espera respuesta por parte del Back-End cada vez que envía una notificación.

- Comunicación no persistente: Las notificaciones son descartadas tanto si el Back-End no está disponible como si se encuentra en ejecución.

- Comunicación indirecta: Para realizar la comunicación es necesario conocer el nombre de la tubería y no basta con conocer el nombre del proceso asociado al Back-End.

El Back-End se encarga de crear la tubería y por lo tanto, actúa como servidor. Por su parte, el Front-End actúa como cliente y la utiliza cada vez que envía una notificación. A continuación se muestran varios diagramas que explican el proceso de comunicación entre ambos subsistemas. Los elementos implicados en la comunicación de los dos subsistemas se detallan a continuación:

- User: Usuario que interactúa con la plataforma.
- Queue: Tabla que contiene las solicitudes de ejecución.
- ExecutionMngr: Componente del Front-End que recoge las funcionalidades de gestión de experimentos.
- Pipe: Objeto que representa la tubería de comunicación.
- Listener: Subproceso que se mantiene a la espera de notificaciones en la tubería.
- Runner: Subproceso encargado de la ejecución de los experimentos.
- Controller: Subproceso controlador de los recursos asignados a la ejecución.

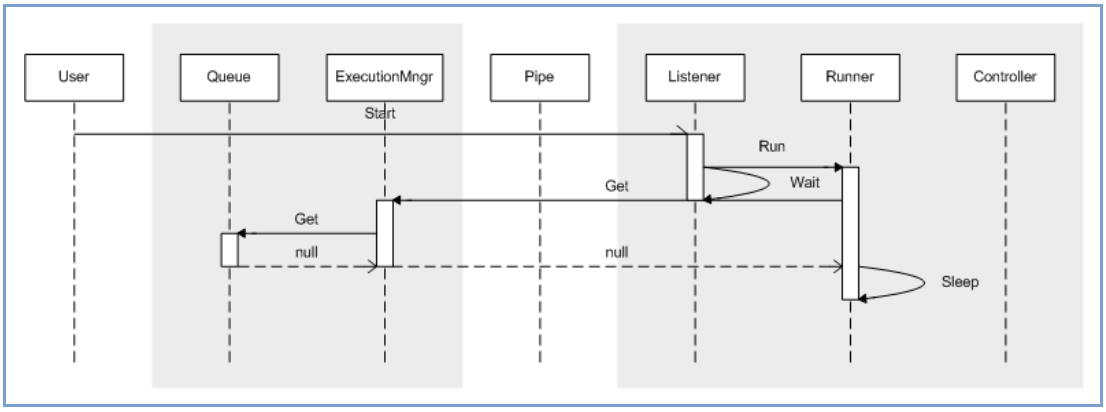


Figura 4. 6. Estado Inicial del Interface de Comunicación.

Una vez presentados los elementos, a continuación se describen los principales escenarios de comunicación. En la Figura 4. 6 se muestra el estado inicial del interface de comunicación. El Back-End se pone en funcionamiento y arranca el subproceso Listener. Éste inicia al subproceso Runner y queda a la espera de una notificación en la tubería. A su vez, Runner comprueba la existencia de solicitudes para obtener la información del siguiente experimento y si la cola se encuentra vacía, se duerme.

En el caso representado en la Figura 4. 7, el usuario interactúa con el Front-End a través de la aplicación web y solicita la ejecución de un experimento. El gestor de experimentos del Front-End almacena la solicitud en la cola de ejecución y envía la notificación al Back-End. Listener recibe la notificación y tras despertar a Runner, queda de nuevo a la escucha. Una vez despierto, Runner consulta la cola para obtener la información necesaria del experimento

y prepara su ejecución. A continuación, inicia el subproceso Controller con el fin de controlar los límites de tiempo y espacio.

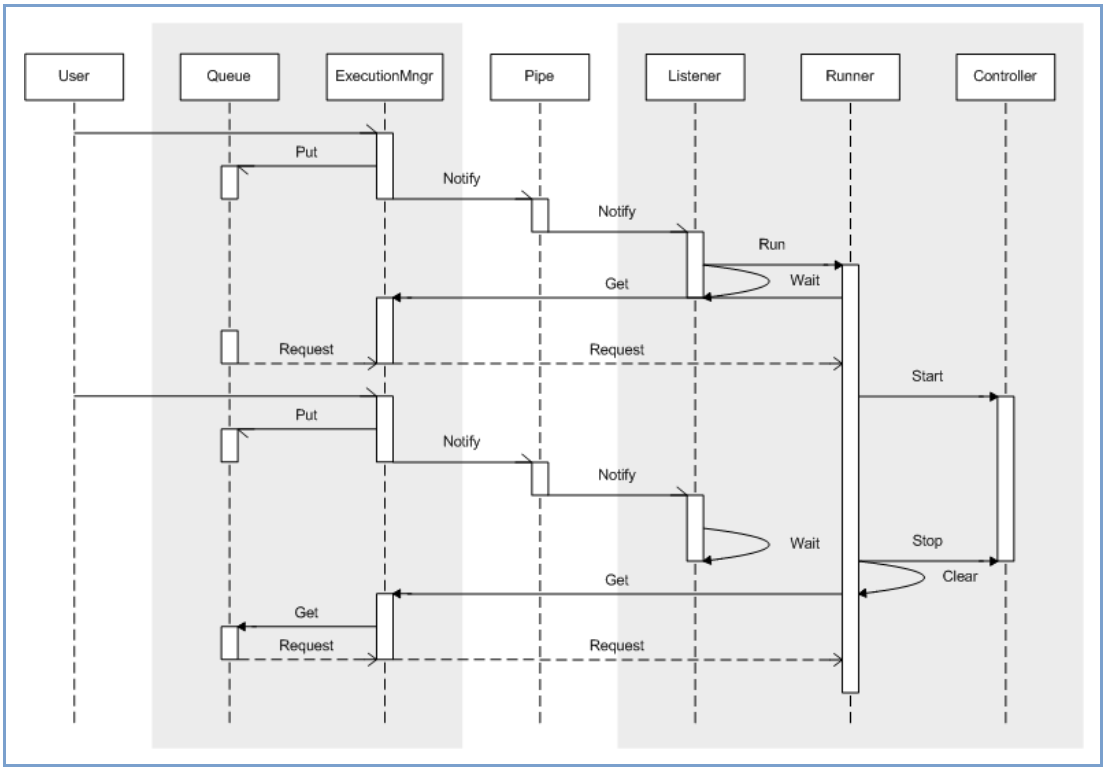


Figura 4. 7. Estado Intermedio del Interface de Comunicación.

Durante la ejecución de un experimento, el Back-End puede recibir solicitudes de por parte del Front-End. En estos casos, las notificaciones recibidas por el subproceso Listener son desechadas puesto que el Back-End se encuentra ocupado.

Si la ejecución termina correctamente, el hilo Runner para al hilo Controller, almacena los resultados de la ejecución y limpia el entorno de trabajo de datos temporales. Por el contrario, si se exceden las limitaciones impuestas, el hilo Controller notifica al hilo Runner de la detención de la ejecución por el consumo de los recursos asignados. Finalizada la ejecución del experimento, el hilo Runner vuelve a consultar la cola para obtener la siguiente solicitud.

La Figura 4.8 muestra la interacción de los componentes cuando el Back-End no se encuentra disponible. En este caso, el Front-End intenta acceder a una tubería que no existe y se produce una excepción. El control de esta excepción es suficiente para que el Front-End pueda almacenar solicitudes sin necesidad de que el Back-End se encuentre disponible.

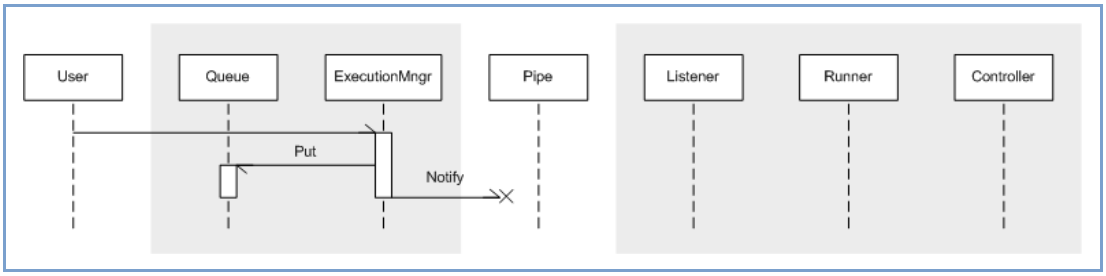


Figura 4.8. Estado de Indisposición del Interface de Comunicación.

4.7. DECISIONES TECNOLÓGICAS

En este punto se presentan las tecnologías elegidas para implementar la plataforma. Se describe el entorno de ejecución, el servidor web y el sistema de gestión de las bases de datos. Así mismo, se presenta el lenguaje elegido para modelar los experimentos biométricos.

4.7.1. Microsoft .NET Framework

La plataforma .NET[6] de Microsoft es un componente software que se puede añadir al sistema operativo Windows. Provee un extenso conjunto de soluciones predefinidas para cubrir las necesidades generales de la programación de aplicaciones. .NET hace énfasis en la transparencia de redes y la independencia de la plataforma hardware con el objetivo de ofrecer un rápido desarrollo de aplicaciones.

.NET es el encargado de proveer lo que se conoce como código administrado, es decir, un entorno que ofrece servicios automáticos al código que se ejecuta. Los principales servicios que ofrece son un cargador de clases que permite localizar clases en tiempo de ejecución, un recolector de basura, un motor de seguridad para administrar la seguridad del código, etc. La plataforma está compuesta por el conjunto de lenguajes de programación, la biblioteca de clases base y el entorno común de ejecución.

Soporta más de 20 lenguajes de programación y es posible desarrollar cualquier tipo de aplicación en la plataforma con cualquiera de ellos. Algunos de los lenguajes desarrollados para este marco de trabajo son C#, Visual Basic, Delphi, C++, J#, Perl, Python, Fortran, Cobol y PowerBuilder.

.NET se ha tomado como plataforma de desarrollo para los subsistemas de la aplicación. Para la elaboración de la aplicación web se ha utilizado ASP .NET junto a C#, el Back-End se ha implementado mediante un servicio de Windows codificado en C# y finalmente, el editor de experimentos también ha sido codificado mediante C#. Su elección como lenguaje de programación se fundamenta en sus amplias funcionalidades, el número de bibliotecas ya

[6] http://www.microsoft.com/NET/

implementadas que ofrece y la capacidad de generar código de forma rápida. Aunque C# y .NET son tecnologías propietarias de Microsoft, existe un compilador implementado que provee el Framework de DotGNU[7] - Mono capaz de generar programas para distintas plataformas como UNIX y Linux por lo que en principio, es posible ofrecer una aplicación multiplataforma.

4.7.2. Internet Information Services

IIS[8] (Internet Information Services), es una serie de servicios para los sistemas basados en Windows. Convierte a un sistema en un servidor de Internet o Intranet de tal manera que los equipos que tienen este servicio instalado pueden publicar páginas web tanto de forma local como remota. El servidor web se basa en varios módulos que ofrecen capacidad para procesar páginas ASP y ASP.NET. Así mismo, es posible agregar módulos para ofrecer servicios sobre PHP o Perl.

IIS ofrece servicios de seguridad y métodos de autenticación que se basan en las últimas tecnologías de cifrado y autenticación mediante certificados de cliente y servidor. Una de las formas que tiene IIS de asegurar los datos es mediante SSL, por lo que se proporciona un método para transferir datos entre el cliente y el servidor de forma segura.

4.7.3. MySQL

MySQL[9] es un motor de bases de datos desarrollado por Sun Microsystems que se ofrece bajo licencia GNU GPL para fines no comerciales. Sus principales características son su disponibilidad multiplataforma, la escalabilidad de las operaciones de almacenamiento, y el soporte tanto de transacciones como de claves ajenas. Así mismo, ofrece conectividad segura, replicación de tablas y búsqueda e indexación por campos de texto. MySQL se utiliza principalmente en aplicaciones web puesto que su funcionamiento en muy bueno en entornos de lectura intensiva de datos.

MySQL se ha elegido como sistema gestor de las bases de datos de la plataforma por su escalabilidad, conectividad, versatilidad y disponibilidad como software libre. La única limitación en cuanto al tamaño de una base de datos viene determinada por los recursos del sistema; así mismo, permite gestionar bases de datos del orden de seis mil tablas y alrededor de cincuenta millones de registros.

[7] http://www.gnu.org/projects/dotgnu/
[8] http://www.iis.net/
[9] http://www.mysql.com/

4.7.4. Open Data Base Connector

ODBC[10] (Open Data Base Connector) es un estándar de acceso a Bases de datos desarrollado por Microsoft. Su objetivo es hacer posible el acceso a cualquier dato sea cual sea la aplicación con independencia del sistema de gestión de la base de datos. ODBC se ha empleado como driver de conexión para aislar el acceso a la información biométrica localizada en orígenes de datos como Microsoft SQL Server y Microsoft Access.

4.7.5. XML

XML[11] es un metalenguaje extensible de etiquetas desarrollado por el W3C (World Wide Web Consortium). No es realmente un lenguaje en particular sino una manera de definir lenguajes para diferentes necesidades. Se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas y no solo para su aplicación en Internet.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan con posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

[10] http://msdn.microsoft.com/en-us/library/ms710252(VS.85).aspx
[11] http://www.w3.org/TR/xml-infoset/

BACK-END. ANÁLISIS

En este capítulo se describe el análisis funcional del sistema Back-End desarrollado en este proyecto. Se comienza con la descripción general del sistema para la posterior captura de requisitos software. También se describe la función y rendimiento que se espera del sistema y los criterios de validación que se necesitan.

5.1. INTRODUCCIÓN

La Ingeniería de Requisitos cumple un papel primordial en el proceso de producción de software, ya que se enfoca en la definición de lo que se desea producir. Su principal tarea consiste en la generación de especificaciones correctas para describir con claridad, sin ambigüedades, de forma consistente y compacta, el comportamiento del sistema con el fin de minimizar los problemas relacionados con el desarrollo de sistemas [Boehm et al, 2007].

Un requisito de software es una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo. Así mismo, se puede considerar requisito a aquella condición o capacidad que debe estar presente en un sistema o en algún componente del sistema para satisfacer un contrato, estándar, especificación u otro documento formal [IEEE, 1998]. Entre los principales beneficios que ofrece la Ingeniería de Requisitos se encuentra las capacidades para predecir los cronogramas de los proyectos y gestionar las necesidades del proyecto, la disminución de los costes, el aumento de la calidad del software, la reducción del rechazo de los usuarios finales y la mejora de la comunicación entre los equipos del proyecto.

5.2. IDENTIFICACIÓN DE LOS STAKEHOLDERS

El primer paso del análisis funcional consiste en identificar las necesidades de los usuarios potenciales de la plataforma y cuáles son las funcionalidades requeridas por el mismo.

De forma indirecta, los stakeholder del Back-End son los propios usuarios que interactúan con el Front-End y son investigadores del campo de la biometría, cuyas necesidades se centran en la evaluación de los sistemas biométricos.

Existen usuarios para el soporte de la plataforma. Aunque no son receptores de los servicios de la misma, su interacción es indispensable para su gestión y mantenimiento.

5.3. ESPECIFICACION DE REQUISITOS INFORMALES DEL BACK-END

En este epígrafe se detalla de forma general la descripción del sistema mediante reuniones con el tutor del proyecto. En él se describen las tareas a realizar por el Back-End y las funcionalidades de la misma.

Así pues, la aplicación y su entorno deben ofrecer las siguientes prestaciones:

- Gestión de los experimentos: se encarga de interpretar el modelo de experimento, ejecutar y generar el informe del mismo propuesto por el usuario.
- Sistema Autónomo: Ejecuta la secuencia de experimentos definida por el Front-End.
- Conexión con el Interface: se establece un protocolo de comunicación.
- Los criterios de seguridad sobre los datos biométricos descritos se mantienen durante todo el experimento.

5.4. PERFILES DE USUARIOS

El perfil de usuario es un modelo que define características básicas de un usuario tales como su tipo, sus permisos y en definitiva, las funcionalidades que la plataforma puede ofrecerle.

El subsistema Back-End cuenta con dos tipos de usuarios.

- El primero de ellos, es un usuario que interactúa con la plataforma a través de la interface. Las características que debe cumplir para poder acceder a todas las funcionalidades de la misma, están recogidas en el documento que describe dicho interface. [Carrero, 2009]
- Usuario Administrador: Para la gestión y administración de la base de datos biométricos, configuración de la aplicación y gestión de los recursos implicados se define un perfil específico de usuario Administrador. Éste interactúa con el subsistema para configurar sus parámetros y controlar el correcto funcionamiento de la aplicación.

5.5. REQUISITOS DE USUARIO

Tras presentar las especificaciones informales del sistema, se analizan y se obtienen los requisitos de usuario, es decir, la funcionalidad que debe proporcionar el sistema a desarrollar. Desde un punto de vista general, la aplicación debe soportar la realización de experimentos biométricos dentro de los límites establecidos por el sistema, permitiendo obtener resultados sobre los mismos.

- Resumen del sistema:

El Back-End se encarga de interpretar, ejecutar y evaluar el experimento propuesto por el usuario. El modelo de experimentación está contenido en un archivo XML según el formato definido por BEEP-EPC ,(anteriormente citado y explicado, *CAPÍTULO 3: METAMODELO*).

El sistema tiene como entrada dicho fichero. Comprueba la coherencia de la secuencia de las operaciones que contiene, las ordena y las ejecuta dentro de los límites establecidos en el sistema. Obtiene resultados que almacena para generar el informe de la ejecución.

Las restricciones impuestas durante la ejecución del experimento corresponden a valores que otorga la plataforma. Estas imposiciones marcan el tiempo y espacio de ejecución para cada experimento.

- Usuarios de la aplicación:

Los usuarios que utilizan la aplicación son investigadores de las distintas entidades que han decidido formar parte de BEEP, para poder realizar experimentos biométricos bajo un mismo entorno de ejecución.

También, existe un usuario administrador, puede configurar y gestionar todos los datos de configuración de la plataforma.

- Subsistema autónomo:

El modelo de datos de la plataforma contiene una tabla denominada “cola de ejecución”. En ella se encuentra la información de los experimentos pendientes de ejecutar y el Back-Office accede a ese almacén mientras queden los experimentos. (*CAPÍTULO 4: 4.5* INTERFACE DE DATOS)

- Ensamblaje con el interface gráfico Front-End:

Debido al trabajo conjunto que deberá realizar Front-End y Back-End, su interconexión es un punto crítico en el desarrollo eficaz de todas sus funcionalidades. Para este fin, se ha definido un protocolo de comunicación asíncrono cómo se ha visto en el *CAPÍTULO 4: 4.6 INTERFACE DE COMUNICACIÓN*

- Gestión de la base de datos biométricos.

La información biométrica que utiliza el sistema proviene de personas que cedan dicha información con fines investigadores.

Según la LOPD [LOPD, 1999], las muestras biométricas se consideran datos personales, privados, y están regulados a nivel nacional e internacional. Por ello se garantizará en todo momento la confidencialidad y el cumplimiento de la Ley.

En el experimento propuesto no se trabaja directamente con las muestras biométricas, sino que se utiliza una copia de las mismas con un identificador único para cada una. De esta manera se trabaja con los datos requeridos para la realización de la aplicación, sin quebrantar la Ley Orgánica de Protección de Datos de Carácter Personal y garantizando el anonimato de los donantes. Esta solución tiene como ventaja que en caso de que la información sea sustraída, solamente se ve afectado el servicio relacionado con la aplicación ya que no está relacionada con los datos oficiales de la identificación.

- Seguridad de la base de datos biométricos.

El hecho de almacenar datos físicos de carácter biométrico en cualquier sistema supone un alto nivel de responsabilidad sobre la información, aún estando algunos de ellos, fuera del alcance del presente proyecto. En este apartado se apuntan algunos aspectos relacionados con su protección:

El primer aspecto importante a tener en cuenta es que estos datos deben estar declarados en la Agencia de Protección de Datos (APD).

En segundo lugar se deben establecer una serie de medidas de nivel básico y medio. Por otro lado, se debe asegurar que la información contenida en el sistema biométrico sólo se usará para el fin a la que ha sido destinada.

En el *ANEXO B: ESPECIFICACIÓN DE REQUISITOS* se enumeran los requisitos de usuario de la aplicación de forma detallada, dando una visión más específica.

5.6. ESPECIFICACIÓN DE REQUISITOS SOFTWARE

El objetivo del apartado es definir los requisitos software partiendo de los requisitos de usuario anteriores. De este modo, se genera un modelo de análisis del sistema, que incluye un conjunto más detallado dentro de la fase de análisis previo al diseño del componente Back-End.

Los requisitos software que ha de ofrecer el Back-End se detallan a continuación:

- Independencia del Back-End: El subsistema debe realizar su funcionalidad mientras haya experimentos pendientes de ejecución.

- Gestión de las notificaciones: El sistema recibe notificaciones de inserción

de experimentos en la cola de ejecución por parte del Front-End.

- Gestión de los experimentos por ejecutar: La aplicación debe proveer de un mecanismo para gestionar la información de los experimentos pendientes de ejecutar. En el caso que hubiera varios el sistema elige el experimento que más tiempo lleve en la cola de ejecución y cuya entidad propietaria tenga mayor prioridad.

- Gestión del experimento a ejecutar: El sistema recupera el experimento de la carpeta de la entidad propietaria.

- Validación de los modelos de experimentos: La aplicación debe validar el experimento antes de ejecutarlo, con el fin de comprobar la coherencia de los datos que componen cada una de las operaciones incluidas en el experimento.

- Algoritmo de planificación: El sistema debe planificar el orden en el que se deben ejecutar las operaciones. Para ello busca entre todos los nodos, uno de inicio y continua con su siguiente y así sucesivamente, hasta llegar a un nodo final.

- Ejecución de los modelos de experimentos: La aplicación debe ejecutar el experimento mediante un algoritmo de planificación, puesto que el orden de ejecución de las operaciones es un factor importante.

- Gestión de las operaciones: El sistema diferencia entre los distintos tipos de operaciones y ejecuta cada nodo según su tipología.

- Gestión de múltiples SGBD: La aplicación debe permitir acceder a información contenida en distintos Sistemas Gestores de Bases de Datos, como MySQL, SQL Server y Access.

- Gestión de las notificaciones de ejecución: El sistema debe enviar notificaciones al Front-End para indicarle el inicio y finalización de la ejecución de un experimento, indicando si ha sido correcto o no.

- Gestión de los límites establecidos durante la ejecución del experimento: El sistema debe controlar que durante su ejecución no sobrepasa un tiempo máximo otorgado a la entidad ni un espacio destinado para el mismo. Ambos valores son establecidos por la plataforma en el fichero de configuración.

- Gestión de los resultados: El sistema debe ser capaz de obtener resultados al final de la ejecución de un experimento, para que el Front-End se los muestre al usuario.

- Gestión de informes: Debe ser capaz de obtener un informe con los resultados obtenidos tras la ejecución.

- Gestión de la información biométrica: La aplicación ha de ser capaz de

modelar cualquier tipo de dato biométrico con el fin de agregar su descripción a los modelos de los experimentos y ha de permitir la incorporación de datos con independencia de su localización o formato.

- Seguridad de los datos biométricos: El sistema ha de asegurar que la información individual de cada uno de los donantes de las bases de datos biométricos no es accedida o modificada por usuarios no autorizados. Así mismo, el sistema debe permitir la incorporación y la eliminación de bases de datos biométricos asegurando el correcto mantenimiento de las mismas dentro de la plataforma.

5.7. MODELO DE ANÁLISIS

Una parte importante del desarrollo de software es la exploración de los requisitos del sistema. A través del modelado de los diferentes usos se puede explorar como trabajan los usuarios con el sistema. Esta información es vital para construir de forma satisfactoria un sistema que cubra las necesidades de los usuarios y por esto, se hace crítico conocer el comportamiento de los usuarios en su interacción con el propio sistema.

Una vez presentados los perfiles de usuario y sus atributos funcionales, a lo largo del siguiente epígrafe se describen los casos de uso y los diagramas de casos de uso. Mediante los casos de uso se describe, desde el punto de vista del usuario, el trabajo del sistema para ofrecer una determinada funcionalidad [Fernández, 2001]. En otras palabras, los modelos de casos de uso se centran en la representación de los requisitos que ha de cumplir el sistema frente a la utilización de los usuarios [Amber, 2004]. Por su parte, a través de los diagramas de casos de uso se muestra la totalidad de actores y casos de uso del sistema así como las interacciones entre ambos y los límites de cada uno de los subsistemas que forman parte del sistema [Kobrun, 2001].

5.7.1. Gestión de Experimentos

Este grupo de funcionalidades se corresponde con la gestión de experimentos. La Figura 5.1 muestra los casos de uso relacionados con el presente grupo de funcionalidades.

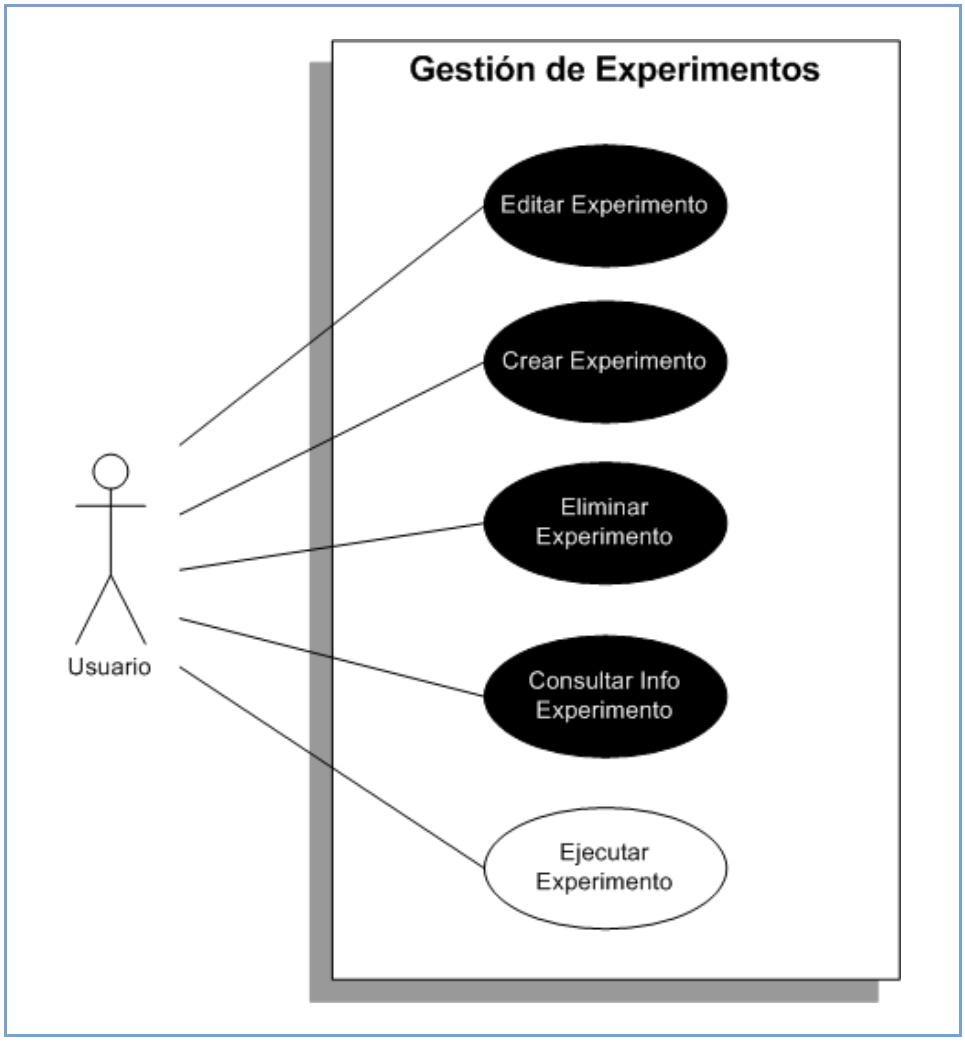


Figura 5.2. Caso de Uso de Gestionar Experimento.

Este diagrama representa las funcionalidades básicas que debe cumplir el sistema con los experimentos, la siguiente tabla relaciona el caso de uso con los requisitos funcionales que representa. Dichos requisitos funcionales se pueden consultar en el *ANEXO B: ESPECIFICACIÓN DE REQUISITOS*.

Ejecutar un Experimento
DESCRIPCIÓN: Esta operación ofrece la posibilidad de enviar un experimento a la cola de ejecución de la plataforma para su posterior ejecución.
ACTORES: Usuario de Grupo de Trabajo. Usuario Propietario.
PRECONDICIONES: Para poder efectuar esta operación, el actor tiene que haber realizado el

caso de uso Consultar la Información de un Experimento y ha de poseer los permisos necesarios dentro del grupo de trabajo. Para que se produzca el desarrollo normal de la operación, el experimento no se puede encontrar en ejecución ni en uso por ningún usuario.

FLUJO NORMAL:

1. El Front-End ofrece una barra de herramientas en la página de información del experimento que contiene un botón para enviar el experimento a la cola de ejecución. Este botón solamente está presente si el actor posee los permisos necesarios.
2. El actor pulsa sobre el botón.
3. El sistema comprueba el estado del experimento. Si el experimento se encuentra libre, el sistema procede a ponerlo en la cola de ejecución y cambia el estado del experimento en la cola de ejecución a Queued. A continuación, el sistema notifica al actor de la realización de la operación.
4. El actor pulsa sobre el botón mostrado en la notificación.
5. El sistema devuelve al actor a la página de información del experimento.
6. El Front-End notifica al Back-End que hay un nuevo experimento en la cola de ejecución.
7. El Back-End obtiene de la cola de ejecución la información del siguiente experimento a ejecutar.
8. Valida la definición del experimento.
9. Notifica al Front-End el inicio de la ejecución del experimento.
10. Cambia el estado de ejecución del experimento en la cola de ejecución a Running.
11. Ejecuta el experimento.
12. Notifica al Front-End el fin de la ejecución del experimento.
13. Cambia el estado de ejecución del experimento en la cola de ejecución a Executed.

FLUJO ALTERNATIVO 1:

3. El sistema comprueba el estado del experimento. Si el experimento no se encuentra libre, el sistema notifica al actor de la imposibilidad de ejecutar el experimento.

FLUJO ALTERNATIVO 2:

6. Si la cola de ejecución está vacía, el Back-End se mantiene a la espera de más notificaciones.

FLUJO ALTERNATIVO 3:

9. Si la validación del experimento no es correcta, se obtendrán los errores correspondientes y se notificará al Front-End y no se ejecutará el experimento.

POSTCONDICIONES:
Si se produce el desarrollo normal, el sistema ejecutará el experimento.

Tabla 5.1. Caso de uso Ejecutar un experimento.

5.8. MAPA GENERAL DE PROCESOS

La gestión de experimentos por parte del Back-End se ha documentado mediante la aplicación de varias herramientas conceptuales habituales en tecnologías de la información. Como punto de partida, la elaboración de la gestión de experimentos se toma como monolítica.

La aplicación reiterada de las herramientas sobre los procesos resultantes crea un modelo formado por varias capas. En cada una de ellas se deben utilizar las herramientas de modelado más adecuadas en función del nivel de detalle requerido.

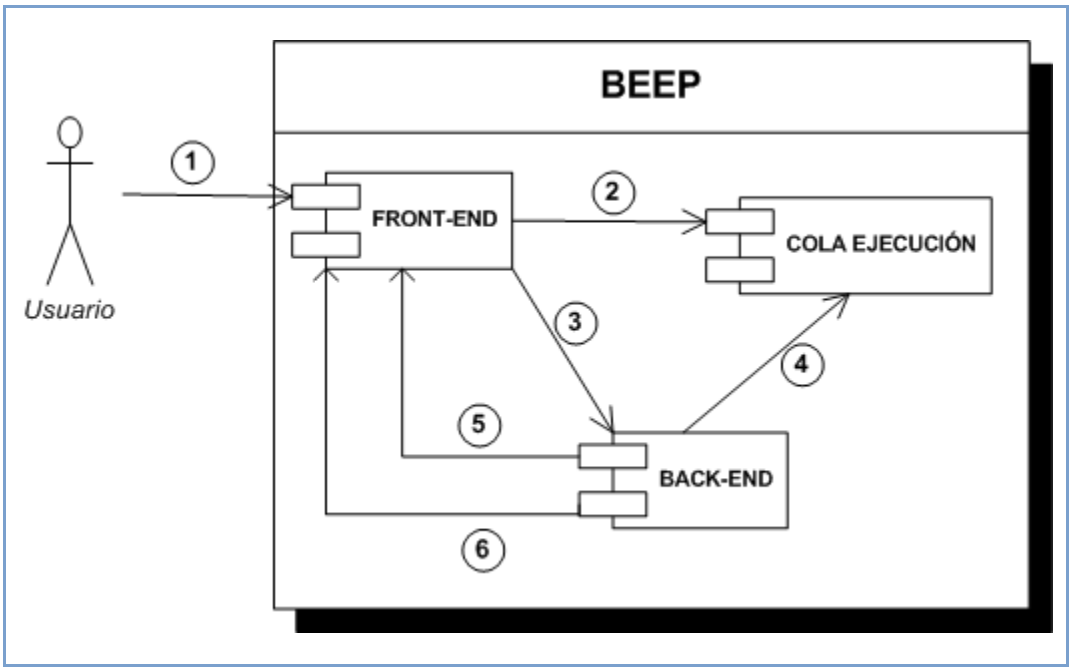


Figura 5.3. Mapa de procesos del Back-End.

Se han identificado en la figura anterior los procesos de comunicación entre el Front-End y el Back-End. A continuación se detallan cada uno de ellos:

1. El usuario de la aplicación envía una petición al interface para ejecutar un experimento. Si el usuario tiene los permisos suficientes se completa correctamente.

2. Si el experimento se encuentra libre, lo almacena en la cola para su posterior ejecución y le cambia el estado a Queued.

3. Envía una notificación al Back-End para indicarle que en la cola de ejecución se encuentra un experimento pendiente.

El Back-End recibe la notificación y se llevan a cabo los siguientes subprocesos claves:

4. Se dirige a la cola de ejecución para obtener la información que necesita sobre el siguiente experimento.

- Carga el modelo del experimento y lo valida antes de ejecutarlo, para comprobar la coherencia de los datos en cada uno de los nodos.

5. Si la validación ha sido correcta, y la definición del experimento contiene todos los datos necesarios para su ejecución, envía una notificación al front-end para indicarle el inicio.

- Ejecuta los nodos ordenadamente y guarda el resultado del experimento donde le ha indicado su nodo final.

6. Una vez finalizada la ejecución, vuelve a mandar una notificación al Front-End para indicarle su fin y éste a su vez, pueda avisar al usuario para que compruebe el resultado.

Los procesos clave interactúan entre sí según el siguiente diagrama:

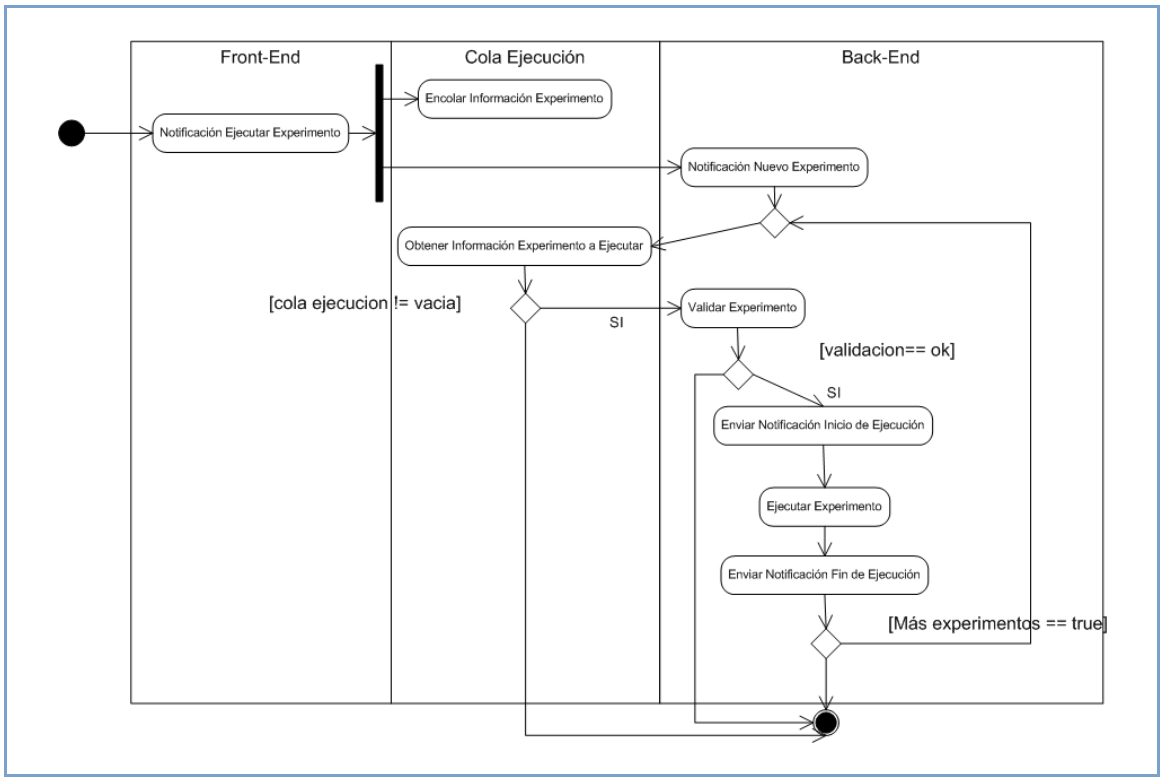


Figura 5.4. Diagrama del proceso ejecutar experimento.

5.8.1. Estados y transiciones de un experimento

En la figura se muestran los distintos estados por los que pasa la definición del experimento.

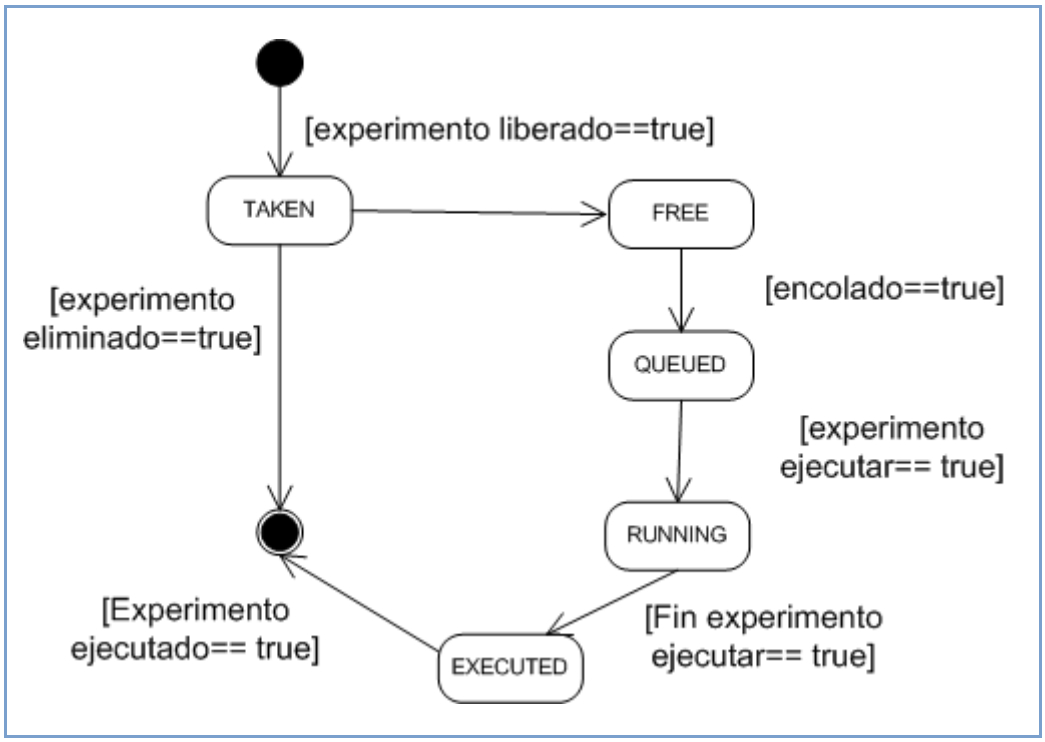


Figura 5. 5. Diagrama de estados y transiciones de un experimento.

La descripción de cada uno de los estados se detalla a continuación:

- En uso (taken): Una definición se encuentra en uso si es solicitada por un usuario para modificarla. En este estado, la definición queda bloqueada y no puede ser reutilizada hasta que no es liberada por el usuario que la solicitó.

- En cola (queued): Una definición se encuentra en el presente estado si ha sido enviada a la cola de ejecución y se encuentra a la espera de ser ejecutada por el back-end.

- En ejecución (running): Una definición se encuentra en este estado mientras el back-end se halla ejecutando el experimento descrito en la definición.

- Finalizada (executed): Un experimento se encuentra finalizado una vez ha terminado su ejecución por el back-end. Una definición llega a este estado independientemente de que la ejecución termine de forma correcta o no. Si ha alcanzado el estado executed no puede volver a ser puesta en cola y se hace preciso generar una nueva versión de la definición del experimento.

- Libre (free): Una definición se encuentra libre si no se encuandra en ninguno de los estados anteriores. Un usuario puede manipular una definición solamente si ésta se encuentra en este estado.

5.8.2. Subprocesos del Back-End.

Para la interpretación, la validación y la ejecución de un experimento se ha utilizado un modelo de estados y transiciones entre los mismos. Así, se parte de un estado inicial, transita por una serie de estados intermedios y finaliza en un estado terminal.

5.8.2.1. Obtener información del experimento a ejecutar.

Tras la notificación del Front-End de la existencia de un nuevo experimento en la cola de ejecución, se inicia este proceso.

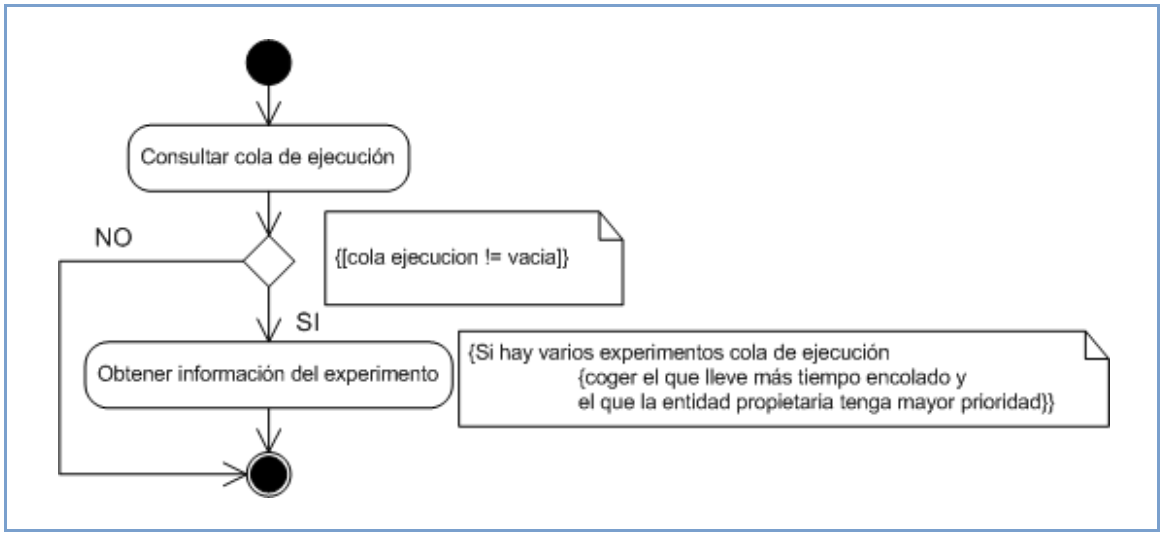


Figura 5. 6. Obtener Información del Experimento a ejecutar.

El Back-End obtiene de la cola de ejecución la información necesaria para su posterior tratamiento durante la ejecución.

5.8.2.2. Validar Experimento.

Tras obtener la información del experimento, se carga la definición del mismo. Si no se ha cargado correctamente, se pasa al estado final.

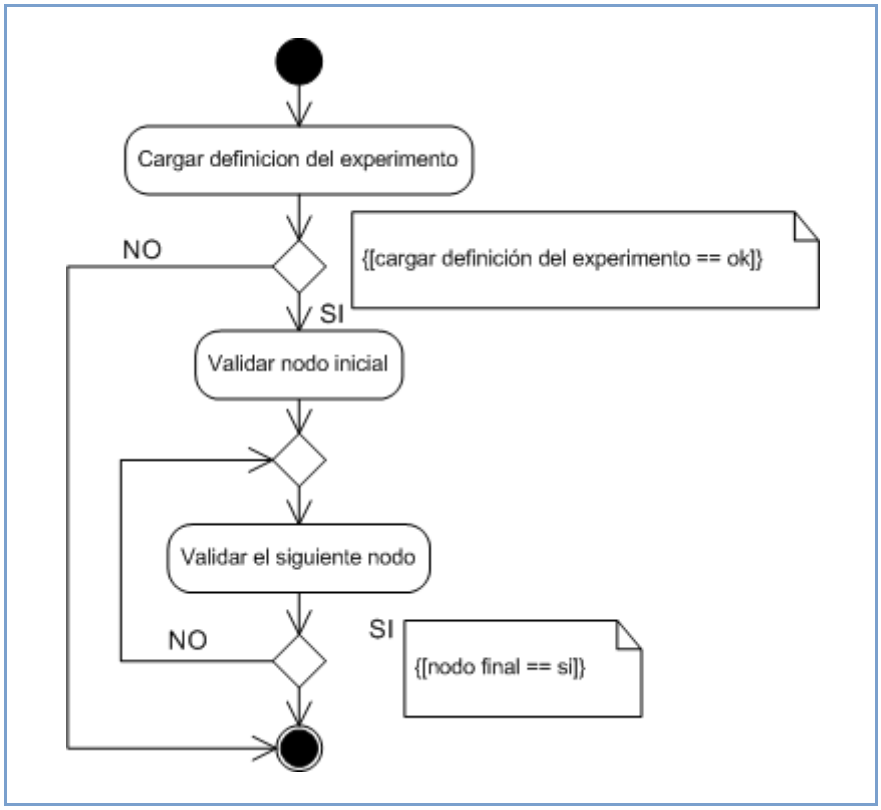


Figura 5. 7. Validar Experimento.

Por el contrario, si se ha cargado de manera satisfactoria, se valida el nodo inicial y se pasa al siguiente. Si es el nodo final, se ejecuta y se pasa al estado final de este proceso. En caso contrario, se repetiría tantas veces como nodos tenga la definición del experimento.

Dependiendo del tipo de nodo en el que se encuentre el proceso se realizan unas comprobaciones u otras. Algunas de ellas son:

Nodo inicial, o Source: se comprueba la existencia de los datos de origen.

Nodo final, o Sink: se comprueba que existe la base de datos en la que se inserta el resultado del experimento.

Join: puesto que puede contener varias entradas, como se expone en el apartado siguiente, se verifica la existencia de las mismas.

5.8.2.3. Ejecutar Experimento.

Si la validación se ha producido correctamente, se pasa al proceso de ejecución.

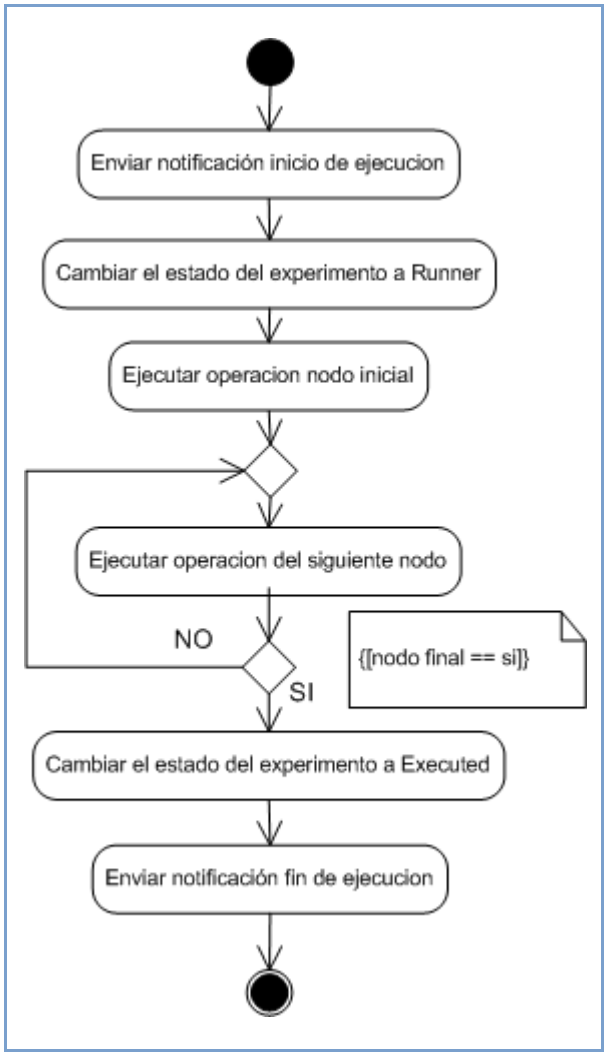


Figura 5. 8. Ejecutar Experimento.

Al comenzar, se notifica al Front-End que va a dar comienzo la ejecución del experimento, para que haga partícipe al usuario de dicha información. Se cambia de estado al experimento en la cola de ejecución, para evitar modificaciones en el mismo.

La ejecución del experimento comienza con la del nodo inicial y tras ésta, la del sucesivo. Si éste fuera el nodo final, se habría acabado la ejecución del experimento, si no se irán ejecutando las operaciones que representen los nodos, hasta completar todos los que contenga la definición del experimento.

5.8.3. Operaciones que representan cada nodo.

Los nodos que forman parte de la definición del experimento se muestran a continuación, junto con el análisis que el Back-End realiza en cada uno de

ellos. En primer lugar se encuentran los almacenes de datos.

ELEMENTO	DESCRIPCIÓN
 Source	El Source, obtiene la información necesaria para obtener los datos sobre los que se realizarán los experimentos. Éstos pueden ser: privados, propios de la entidad; o públicos para todos los usuarios que formen parte de la plataforma. Si los datos de entrada son públicos, se obtienen de la base de datos del sistema. Si por el contrario fueran privados, el sistema se conecta a la base de datos propia de la entidad y cargará los datos de la misma.
0 entradas	
1 o más salidas	
Posee el origen de los datos.	
 Sink	El Sink, obtiene la información de dónde se deben almacenar los datos resultantes del experimento. El sistema guarda los resultados de la ejecución del experimento en la base de datos de la entidad.
1 entrada	
0 salidas	
Obtiene la información de dónde se deben almacenar los datos evaluados en la ejecución del experimento.	
 Storage	Se denomina Storage a las tablas que se generan durante la ejecución del experimento. Puede que el usuario desee almacenar alguna de estas tablas intermedias y para ello marcará la opción de persistente y el sistema las guarda en su base de datos.
1 entrada	
1 o más salidas	
Pueden ser de almacenamiento temporal o persistente, tras la ejecución del experimento.	

Figura 5.9. Operaciones Almacenamiento.

A continuación se muestran el resto de las operaciones que se pueden realizar sobre los datos biométricos, según el metamodelo citado en el *CAPÍTULO 3:METAMODELO*.

Cada elemento lleva asociado una operación sobre los datos de entrada y cuyo resultado se ve reflejado en los datos de salida. Las operaciones intermedias se realizan en una base de datos del sistema, donde se almacenan las tablas generadas durante el experimento. Al finalizar todas, vuelca las tablas especificadas por el usuario en la base de datos de la entidad propietaria y el elimina el resto.

ELEMENTO	DESCRIPCIÓN
 Tap	El Tap recibe una tabla como entrada, le aplica un filtro y guarda el resultado en otra de salida.
1 entrada	
1 salida	
Filtra los datos de entrada a través de una condición.	
 Join	El Join recibe como mínimo dos entradas que el sistema debe asegurar que ya se encuentran cargadas en él, para poder llevar a cabo la operación. Les aplica un criterio de asociación y guarda el resultado en una tabla de salida.
2 o más entradas	
1 salida.	
Asocia los datos de entrada según un criterio.	

Figura 5. 10. Operaciones Tap y Join

ELEMENTO	DESCRIPCIÓN
 Process	El Process acepta una tabla de entrada donde se encuentra la información necesaria para cargar las librerías precisas para su ejecución. La salida de esta operación se almacena en otra tabla de salida.
1 entrada.	
1 salida.	
Procesa los datos de entrada mediante la aplicación de algoritmos y procesos.	
 Group	El Group recibe una tabla como entrada, agrupa sus datos según un criterio que se encuentra especificado en el nodo y vuelca los resultados obtenidos en una tabla de salida.
1 entrada.	
1 salida.	
Agrupa los datos según un determinado criterio.	

Figura 5.11. Operaciones Process y Group.

5.8.4. Control del Espacio y del Tiempo de la Ejecución.

El sistema debe controlar los límites de espacio y tiempo durante la ejecución de los experimentos. Estos valores se establecen en el fichero de configuración de la plataforma.

El Back-End carga por cada experimento el tiempo y el espacio que otorga a dichos parámetros.

5.8.5. Modelo de clases del dominio.

A continuación se muestra el diagrama de análisis de dominio a alto nivel.

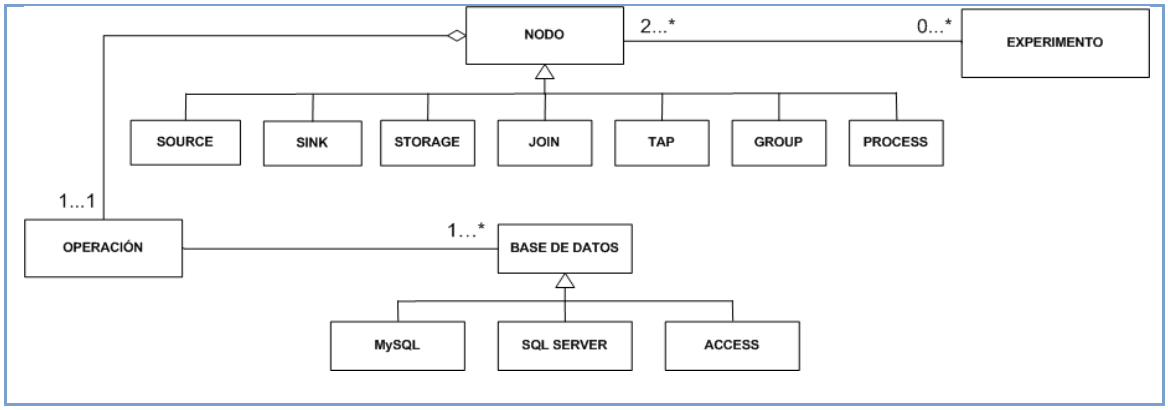


Figura 5. 12. Diagrama de clases del dominio.

En él se puede observar la generalización de nodo, en los distintos tipos existentes y de base de datos.

Operación obtiene el interface de cada una de las acciones que deben realizar los nodos según su tipología.

BACK-END. DISEÑO

En este capítulo se describe el diseño del sistema de experimentación biométrica back-end mediante la definición de la arquitectura del sistema, que comprende la organización en subsistemas software y la especificación del entorno tecnológico. Por su parte, la especificación detallada de los componentes queda representada por el diagrama de clases detallado y el diseño físico de los datos.

6.1. INTRODUCCIÓN

Un patrón describe un problema que ocurre en repetidas ocasiones en un entorno dado. El mismo patrón describe el núcleo de la solución al problema, de un modo que es posible utilizar dicha solución en sucesivas ocasiones sin repetir la forma dos veces [Gamma, 1998].

Los patrones ayudan a promover buenas prácticas de diseño y facilitan la construcción de arquitecturas de software complejas. Los patrones existentes cubren diferentes rangos de escala y abstracción. Por un lado, ayudan a estructurar el software en subsistemas y por otro, soportan el refinamiento de los componentes. Otros ayudan a implementar ciertos aspectos de diseño en un lenguaje de programación específico. A continuación se describen los patrones arquitectónicos y los de diseño que se han evaluado para la implementación de la aplicación.

6.2. PATRONES ARQUITECTÓNICOS

Expresan modelos para la organización estructural de los sistemas de software. Proporcionan un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos [Buschmann, 1996].

6.2.1. Modelo de Capas

Ofrece una estructuración de la aplicación en grupos de subtareas, de tal forma que cada grupo se encuentra a un nivel particular de abstracción [Buschmann, 1996]. Este modelo se suele aplicar en el diseño de sistemas cuya

característica dominante es una mezcla de operaciones de alto y bajo nivel, donde las tareas de alto nivel se apoyan sobre las de bajo nivel. La comunicación en estos sistemas consiste en peticiones que descienden desde los niveles superiores hacia los inferiores y respuestas a las peticiones, datos entrantes o notificaciones sobre eventos que viajan en dirección opuesta.

Mediante este patrón se estructura el sistema en un número apropiado de capas, cada una de las cuales está compuesta por elementos que trabajan al mismo nivel de abstracción. Los servicios de cada capa implementan una estrategia para combinar las operaciones de la capa inferior y proporcionar servicio a su capa superior.

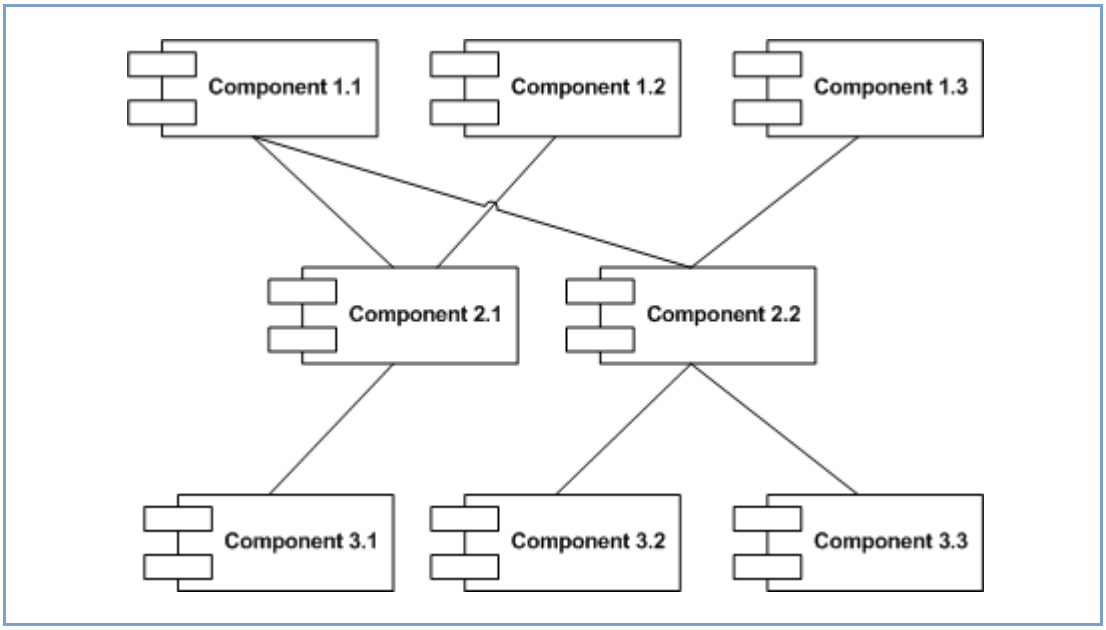


Figura 6. 1. Esquema del Patrón de Capas.

6.3. PATRONES DE DISEÑO

Expresan esquemas para definir estructuras de diseño con las que construir sistemas software. Estos patrones resuelven un problema de diseño general en un contexto particular [Buschmann, 1996]. A continuación se presentan los patrones de diseño utilizados en el proyecto.

6.3.1. Value Object

El patrón Value Object no es más que un objeto que empaqueta datos que permite de forma compacta y organizada la transferencia de datos entre capas. Dicho objeto contiene todos los datos necesarios procedentes de uno o varios objetos de negocio, accesibles mediante propiedades públicas o métodos de obtención o establecimiento.

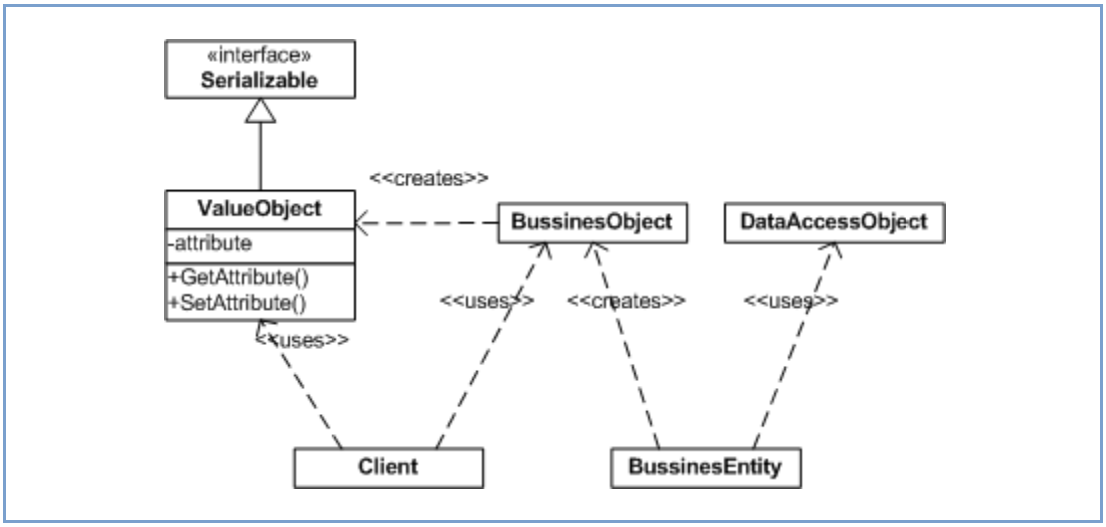


Figura 6. 2. Esquema del Patrón Value Object.

Aunque un Value Object se encuentra relacionado directamente con un objeto del dominio, no se trata de los mismos objetos. Mientras que un objeto del dominio puede contener lógica de negocio, un Value Object es un mero almacén de datos.

6.3.2. Factory Method

El patrón Factory Method permite escribir aplicaciones más flexibles respecto de los tipos a utilizar puesto que delega la creación de las instancias en el sistema a subclases que pueden ser extendidas a medida que evoluciona el sistema. Factory Method define un interfaz para crear objetos, pero son las subclases las que deciden las clases a instanciar.

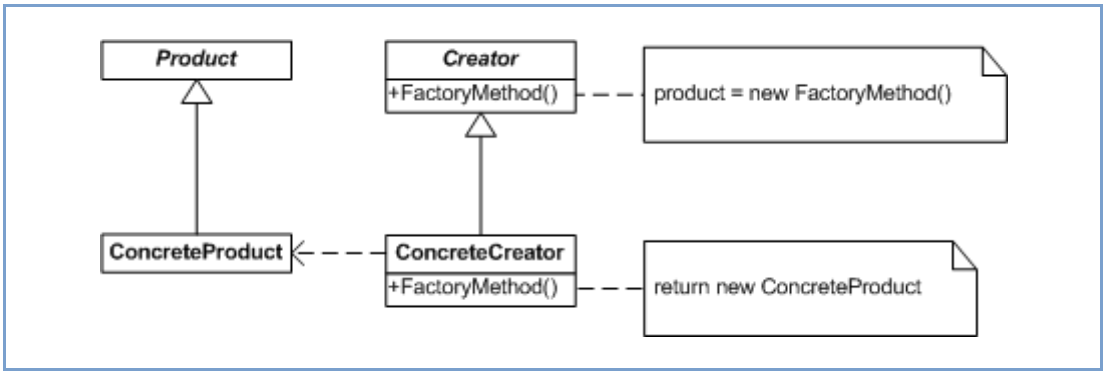


Figura 6. 3. Esquema del Patrón Factory Method.

Éste permite encapsular el conocimiento referente a la creación de objetos y hace más adaptable el diseño a costa de aumentar ligeramente la complejidad de la implementación.

6.3.3. Command

El patrón Command especifica una forma simple de separar la ejecución de

una acción del entorno generado por ella. Este patrón presenta una forma sencilla y versátil de implementar un sistema basado en acciones que permite su extensibilidad y mantenimiento.

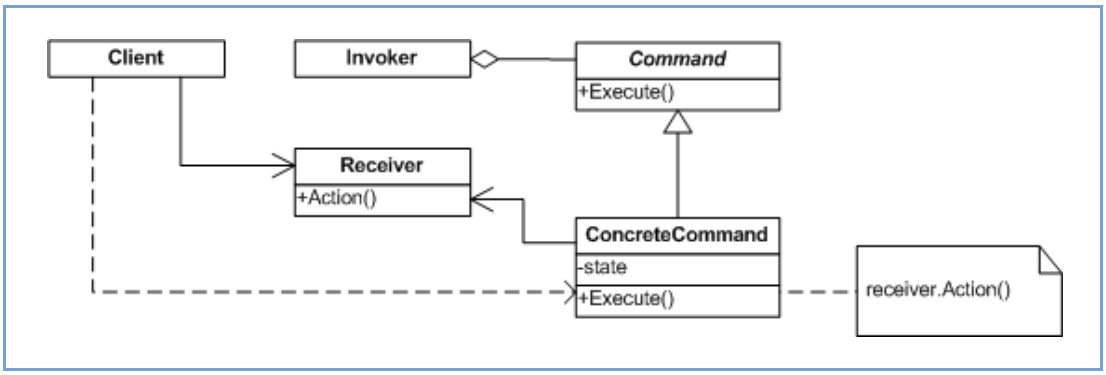


Figura 6.4. Esquema del Patrón Command.

Se suele aplicar cuando se quiere facilitar la parametrización de las acciones a realizar, independizar el momento de petición del de ejecución o desarrollar sistemas utilizando órdenes de alto nivel que se construyen con operaciones sencillas (primitivas).

6.4. DISEÑO DE LA ARQUITECTURA DEL SISTEMA

La definición de la arquitectura del Back-End comprende la partición física del sistema y su organización en subsistemas software.

6.4.1. Arquitectura Física

Tal y como se introdujo en el *apartado 4.3 ARQUITECTURA*, el Back-End que compone la plataforma BEEP, presenta una arquitectura cliente-servidor.

6.4.2. Arquitectura Lógica

Para la arquitectura lógica del Back-End Se aplica el estilo arquitectónico de capas, de modo que la distribución de los componentes se organiza en niveles bien definidos.

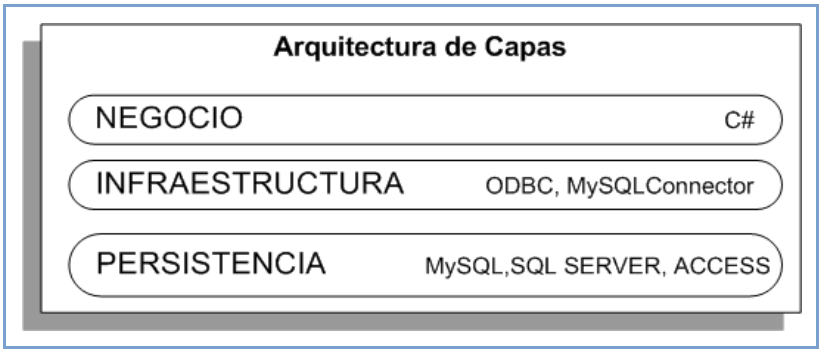


Figura 6. 5. Arquitectura en capas.

Los diagramas de componentes muestran los componentes software, (que constituyen una parte reusable) sus interfaces, y sus interrelaciones. En muchos aspectos se puede considerar que un diagrama de componentes es un diagrama de clases a gran escala. Un paquete en un diagrama de componentes representa una división física del sistema.

El diagrama de componentes se ofrece en la Figura 6. 5, presenta las siguientes capas:

- Negocio: Procesa las peticiones del interface. Es la encargada de ofrecer la lógica de negocio de la aplicación, los servicios de control de concurrencia y el acceso a la capa de persistencia a través de los subsistemas de infraestructura. Permite ejecutar los experimentos en el momento en el que la clase es instanciada. Esta capa se ha implementado a partir de clases C#.

- Infraestructura: Proporciona la comunicación con el subsistema de persistencia. Provee transacciones, seguridad y escalabilidad en el acceso a los datos. Esta capa está compuesta por MySQLConnector (acceso a los datos de la aplicación) y ODBC (acceso a las bases de datos biométricos).

- Persistencia: Tiene como finalidad albergar los datos de la aplicación.

A continuación, se muestra el particionamiento de las capas de la aplicación y las tecnologías elegidas para cada una de ellas. Para facilitar la comprensión de la disposición de las distintas capas, se muestra la Figura 6. 6 más detallada que hace más fácil el entendimiento de las mismas.

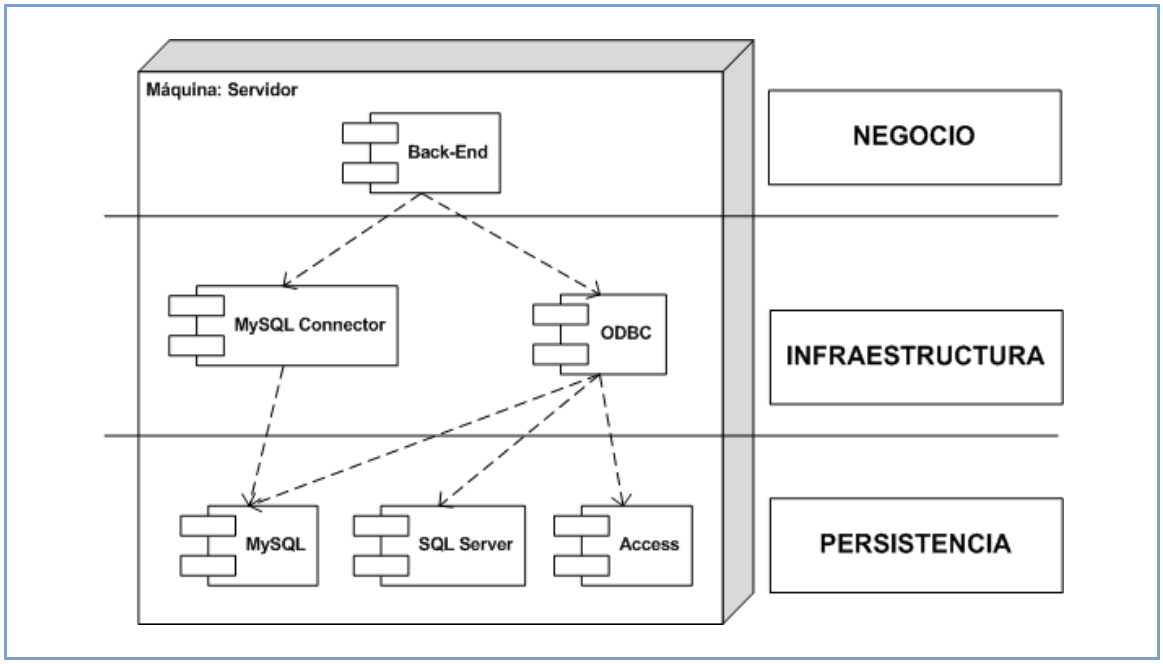


Figura 6. 6. Arquitectura en capas en la plataforma.

6.5. CAPA DE NEGOCIO

La capa de negocio alberga los procesos que ejecutan las operaciones necesarias para ofrecer las funcionalidades del sistema. Se comunica con el interface para recibir las notificaciones de nuevos experimentos en la cola de ejecución. Requiere de la capa de persistencia para almacenar y recuperar datos de ella.

La capa de negocio del Back-End se compone de paquetes propios y comunes a la plataforma, como se muestra en la Figura 6. 7 con la dependencia entre los mismos.

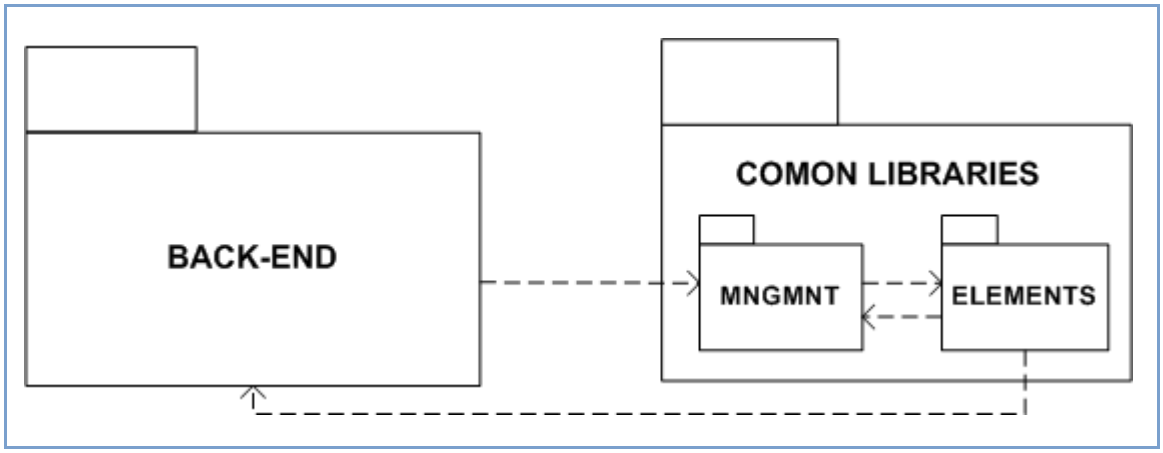


Figura 6. 7. Paquetes de la Capa de Negocio.

A continuación, se describe cada uno de los paquetes que forman parte de la lógica de negocio. Se muestra un esquema general con todos los paquetes que componen el subsistema.

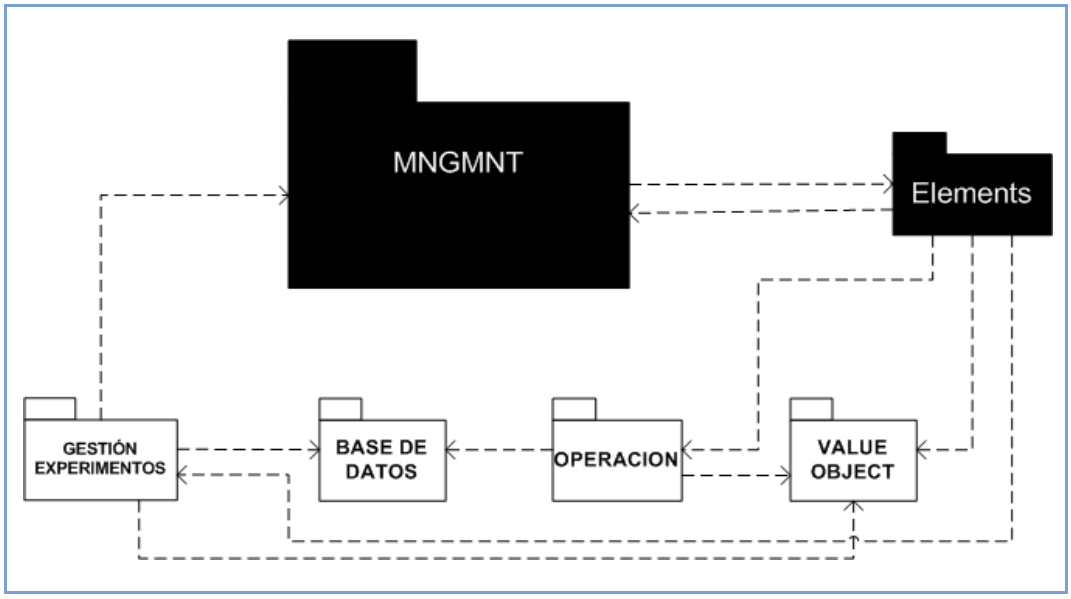


Figura 6. 8. Componentes del Subsistema.

Se han distinguido los paquetes propios del Back-End y los “Common Libreries” comunes de la plataforma. Para hacer más sencillo el entendimiento de las relaciones entre ambos, se han agrupado en el componente “MNGMNT” todos los paquetes de la librería común. A continuación se muestra el contenido del mismo en la siguiente figura:

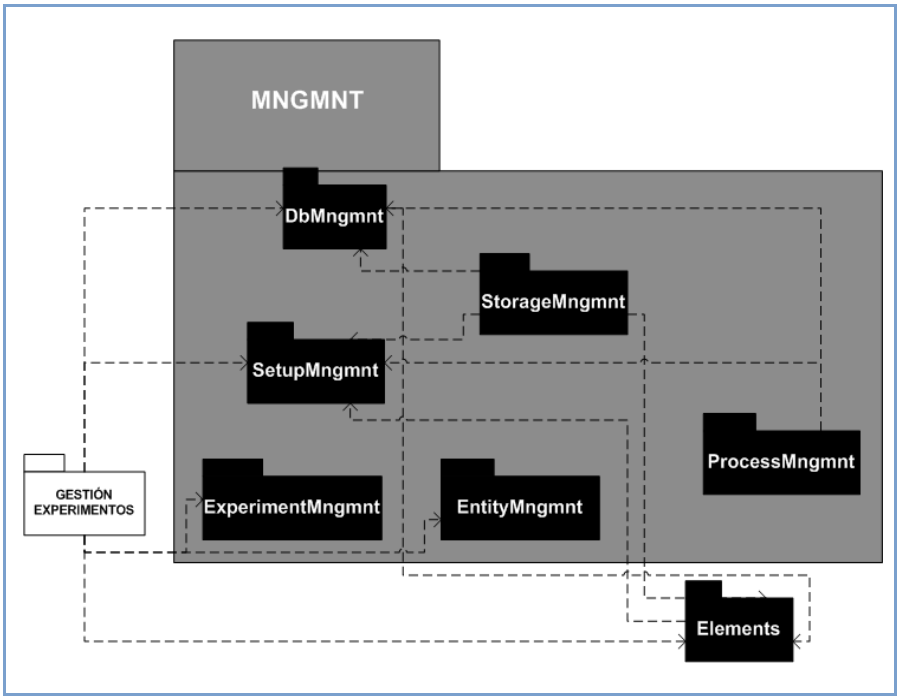


Figura 6. 9. Componente MNGMNT

6.6. FASE DE DISEÑO DETALLADO.

En este apartado se pretende hacer una descripción con mayor profundidad de cómo funciona el Back-End, con ayuda de distintos diagramas para facilitar su comprensión.

6.6.1. Restricciones Generales.

Se van a describir las limitaciones aplicables al desarrollo del Back-End. En esta sección no se impone ningún requisito específico, sino que se establecen las razones por las que existen ciertas restricciones.

El Back-End, como se ha descrito antes, permite la ejecución de una serie de operaciones sobre datos biométricos de distintas bases de datos. Para permitir a futuros desarrolladores incrementar el número de operaciones y de bases de datos, se debe aislar el código específico y trabajar con interfaces. Se utilizarán patrones de diseño para permitir esta escalabilidad.

Se ha seleccionado Visual Studio para desarrollar el Back-End, puesto que permite codificar con la técnica orientada a objetos, utilizar interfaces y tratar las colecciones de elementos utilizando sus propias clases e interfaces.

6.6.2. Diseño Detallado.

Se pretende poner un ejemplo, sobre los componentes del Back-End, para explicar detalladamente las clases que componen cada uno de ellos.

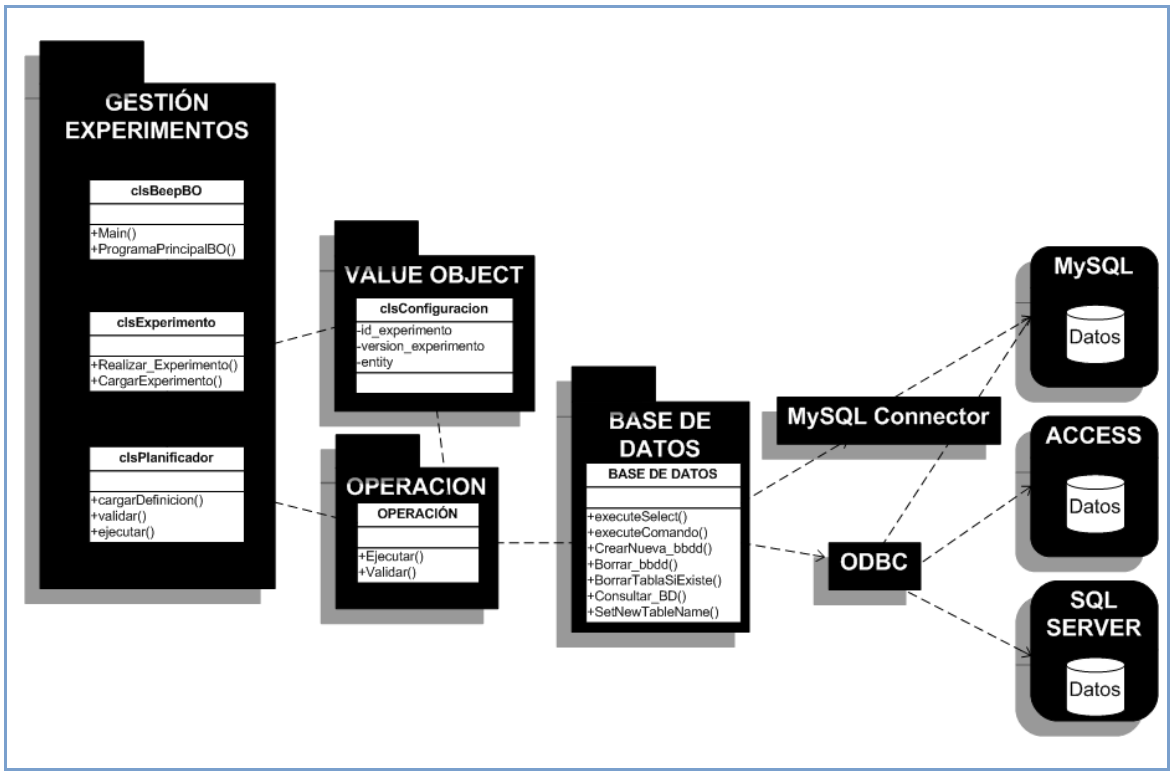


Figura 6. 10. Ejemplo general.

Para explicar detalladamente la funcionalidad de cada clase, tras esta figura general, se hace uso de los diagramas de interacción para hacer más sencillo su entendimiento.

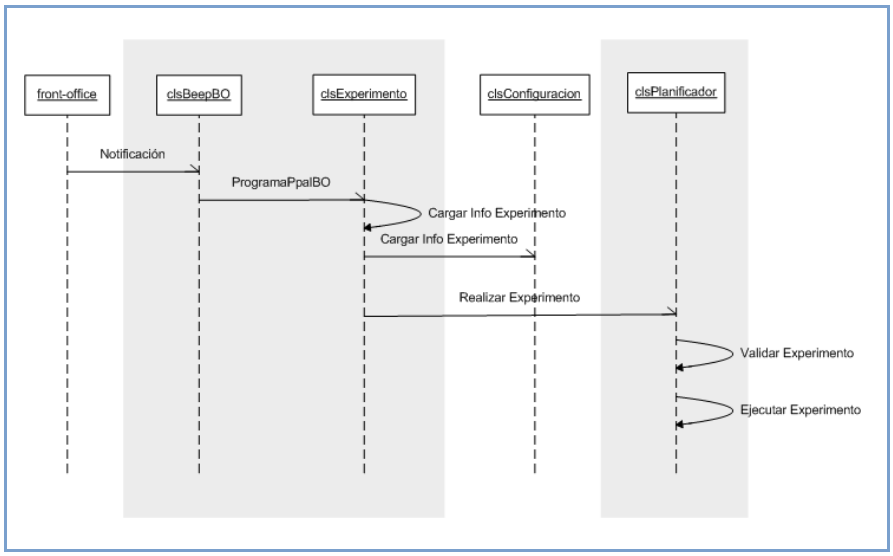


Figura 6. 11. Diagrama de Secuencia del Paquete Gestión de Experimentos.

A continuación se analizará cada uno de los paquetes expuestos anteriormente, describiendo las clases que lo componen y la funcionalidad que realizan.

PAQUETE GESTIÓN EXPERIMENTOS

Este paquete es el que se comunica con la interface de la plataforma. Cuando existe alguna notificación para ejecutar algún experimento, lo inserta en la cola de ejecución y envía una notificación al Back-End para indicarle que tiene un experimento pendiente en la cola de ejecución.

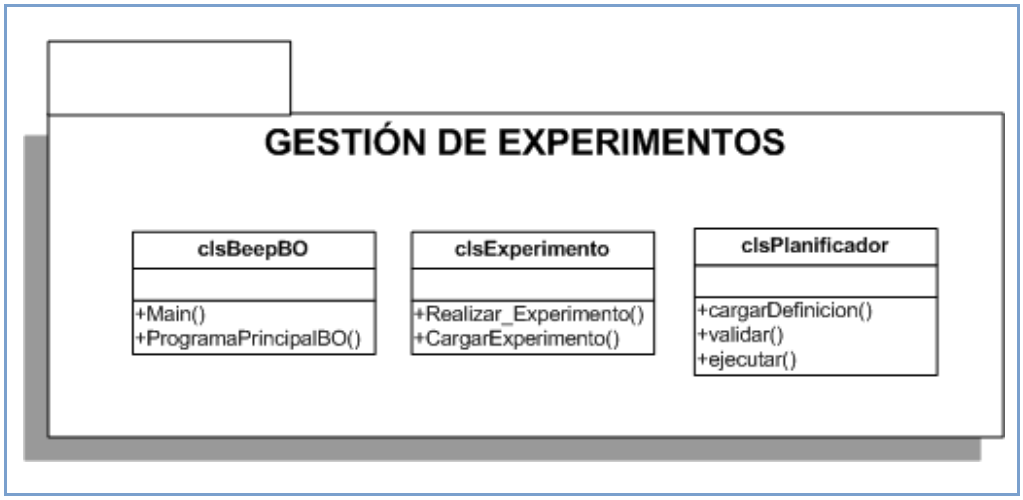


Figura 6. 12. Paquete Gestión de Experimentos.

La **clsExperimento** cargará la información del experimento a ejecutar. Si hay varios, se elegirá según la prioridad de la entidad propietaria del experimento y la fecha de puesta en cola. Una vez obtenidos los datos necesarios para poder ejecutar el experimento, llamará a la clase clsPlanificador.

La **clsPlanificador** carga la definición del experimento a ejecutar, y antes de hacerlo, valida cada nodo, por si hubiera algún dato asociado a la operación que no fuera correcto y así ahorrar, una ejecución que de antemano se sabe fallida. Por el contrario, si la validación es correcta, se llamará al método ejecutar.

Tanto el método **clsPlanificador**.**validar**, como el de **clsPlanificador**.**ejecutar**, siguen un algoritmo de planificación parecido y que merece especial mención, puesto que era una de las especificaciones a conseguir.

```
Para cada nodo de la definición del Experimento
{
  Si (nodo =nodo de inicio)
    {
      //Se valida/ejecuta el nodo
        clsOperacion Operacion=clsFactoryOperacion.getTipoOperacion(nodo)
        Operacion.validar() ò Operación.ejecutar()
     //Una vez que se haya validado/ejecutado se llama al nodo consecutivo
     }
 }
```

Figura 6. 13. Pseudocódigo planificar.

Ambos llaman al método de la clase clsFactoryOperacion, getTipoOperacion(), y devuelve en una variable de tipo clsOperacion, con el tipo de nodo concreto en el que nos encontramos. Llaman al método correspondiente en cada caso, validar o ejecutar, que se ha implementado mediante el patrón Command.

PAQUETE VALUE OBJECT

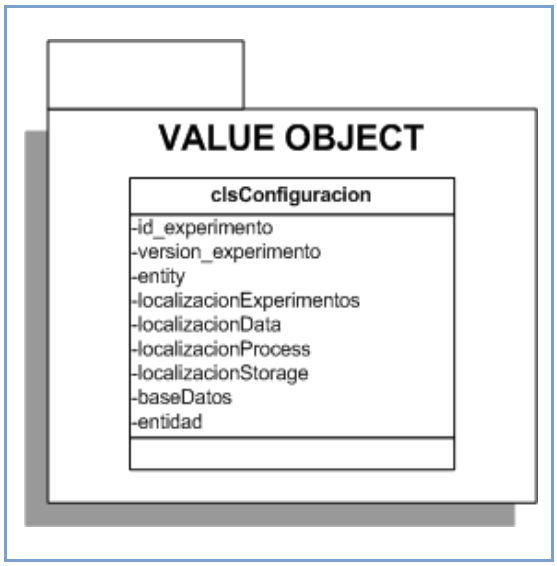


Figura 6. 14. Paquete Value Object.

La estrategia seguida en este caso ha sido definir una capa vertical de value objects patrón (explicado en el *Apartado 1.2: PATRONES DE DISEÑO*), representando el modelo de información de la aplicación. Ésta es compartida entre todas las capas que necesiten tomar esos valores, con el fin de que un cambio en cualquiera de las capas (por profundo que sea) no afecte al resto.

Al tratarse de objetos transversales a todas las capas, un cambio en la capa generadora de estos objetos no implica un cambio en las consumidoras y viceversa, consiguiendo así la ortogonalidad deseada.

Esta clase **clsConfiguración** obtendrá los valores cuando la clsExperimento llame al método cargarExperimento y serán utilizados por el resto de clases cuando necesite obtener alguno de los valores que contienen sus variables.

PAQUETE OPERACIÓN

Para el diseño de este paquete, se ha optado por la utilización del patrón Factory (explicado en el *Apartado 1.2.2:Factory Method*). Permite escribir aplicaciones que son más flexibles respecto de los tipos a utilizar, difiriendo la creación de las instancias en el sistema a subclases que pueden ser extendidas a medida que evoluciona el sistema. También, reduce las dependencias en el proceso de creación de las clases concretas que va a utilizar la clase cliente.

Permite encapsular el conocimiento referente a la creación de objetos, y hace también que el diseño sea más adaptable a cambio de sólo un poco más de complejidad. Esto se debe a que la clase Creator (FactoryOperación, en este caso) delega en sus subclases la creación de los objetos concretos. De esta forma cada subclase sabe qué tipo de objeto debe crear

Clase/Interface	Cometido
Operación	Define la interface de los objetos creados por el método FactoryOperacion.
Source,Sink,join,..	Implementa la Interface Operación.
FactoryOperacion	Declara el método getTipoOperacion(FactoryMethod), el cual devuelve un objeto que cumple la Interface Operación.
FactoryOperacion Creator	Implementa (redefine) el método factoría para devolver una instancia de una operación concreta.

Tabla 6.1. Patrón Factory Operaciones.

Las clases principales en este patrón son el FactoryOperacion y el Operación. El primero necesita crear instancias del segundo, pero el tipo concreto de Operación no debe ser forzado en las subclases del FactoryOperacion, porque entonces las posibles subclases del FactoryOperacion deben poder especificar subclases de la Operación a utilizar.

La solución para esto es hacer un método abstracto que se define en el FactoryOperacion. Se define para que devuelva una Operación. Las subclases del FactoryOperacion pueden sobrescribir este método para devolver subclases apropiadas de la Operación...

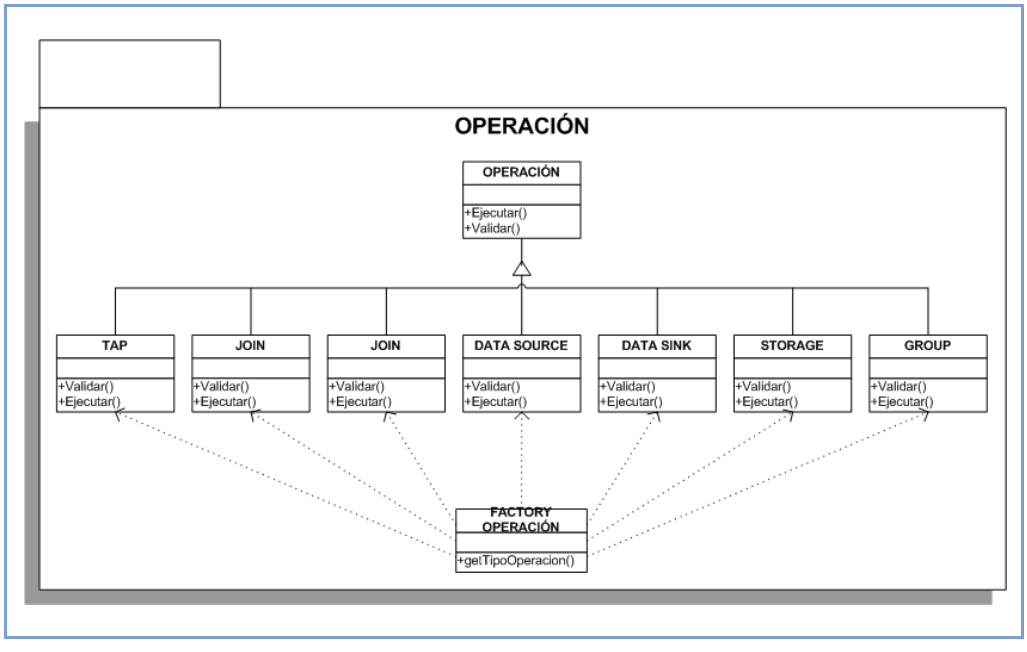


Figura 6. 15. Paquete Operación.

Cada uno de los nodos que representan las distintas operaciones que componen un experimento poseen los métodos validar y ejecutar.

El método validar() obtiene la información del nodo en el que estamos, analiza que todo esté correcto y pasa al siguiente nodo para repetir la misma operación hasta que llegue al final.

En el caso del método Ejecutar(), se encarga de ir ejecutando cada una de las operaciones que representa el nodo en el que se encuentra.

Ambas operaciones se han implementado mediante el patrón command, puesto que flexibiliza la adición de nuevas clases mediante el interface común.

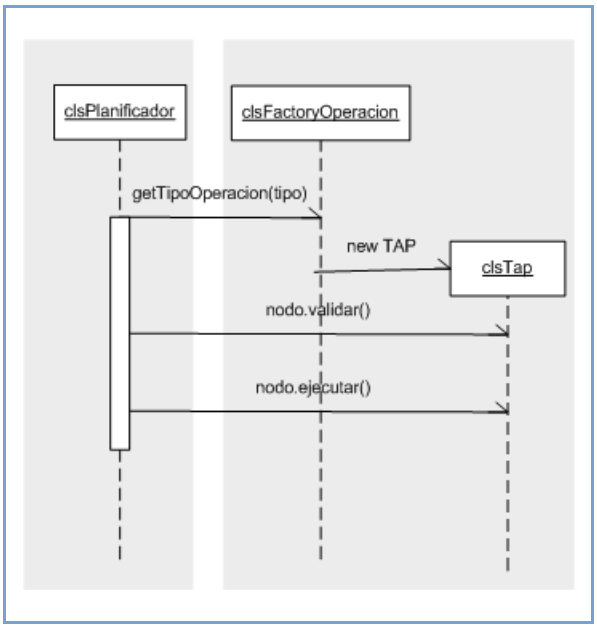


Figura 6. 16. Diagrama de Actividad paquete operación.

PAQUETE BASE DE DATOS

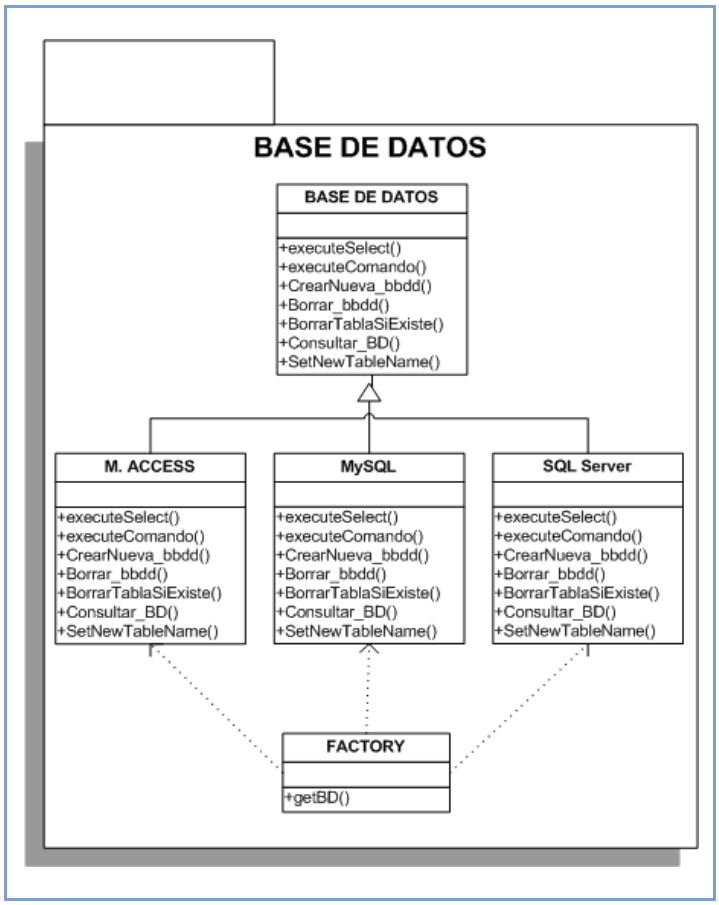


Figura 6. 17. Paquete Base de Datos.

En el caso de las Bases de Datos, se ha optado por la misma solución que en el paquete Operaciones, por las ventajas expuestas anteriormente.

Inicialmente se ha pensado en estos tres Gestores de Bases de datos, pero con el diseño actual, podría aumentar sin sufrir muchos cambios y dejar abierta la posibilidad de incluir algún gestor más en el futuro.

Clase/Interface	Cometido
Base de Datos	Define la interface de los objetos creados por el método Factory.
MySQL,Access,SQL Server	Implementa la Interface Base de datos.
Factory	Declara el método getBD(FactoryMethod), el cual devuelve un objeto que cumple la Interface Operación.

Factory Creator	Implementa (redefine) el método factoría para devolver una instancia de una operación concreta.

Tabla 6.2. Patrón Factory Base de Datos.

Cada nodo posee las operaciones necesarias para interactuar con las distintas Bases de datos, como son: Abrir(), Cerrar(), Consultar(), EjecutarComando(),…

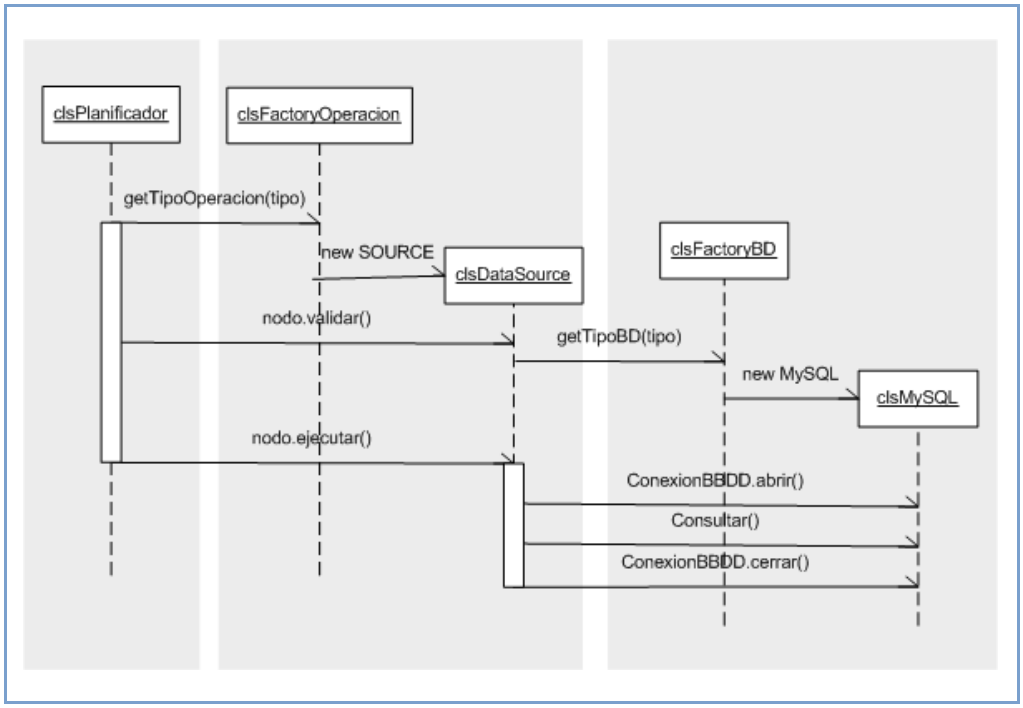


Figura 6. 18. Diagrama de Actividad Paquete Base de Datos.

6.6.3. Gestión del Espacio y Tiempo de Ejecución

Como se expuso en las especificaciones del proyecto, la ejecución de un experimento está limitado tanto en tiempo como en espacio, dependiendo de lo que la plataforma le haya asignado al experimento.

Para implementar dicha funcionalidad del sistema se ha hecho uso de hilos, para pausar la ejecución del experimento si fuera necesario, en caso de sobrepasar cualquiera de las restricciones.

En el caso de que la ejecución del experimento finalice correctamente, sin sobrepasar el tiempo y espacio predeterminado, el diagrama de secuencia sería el siguiente.

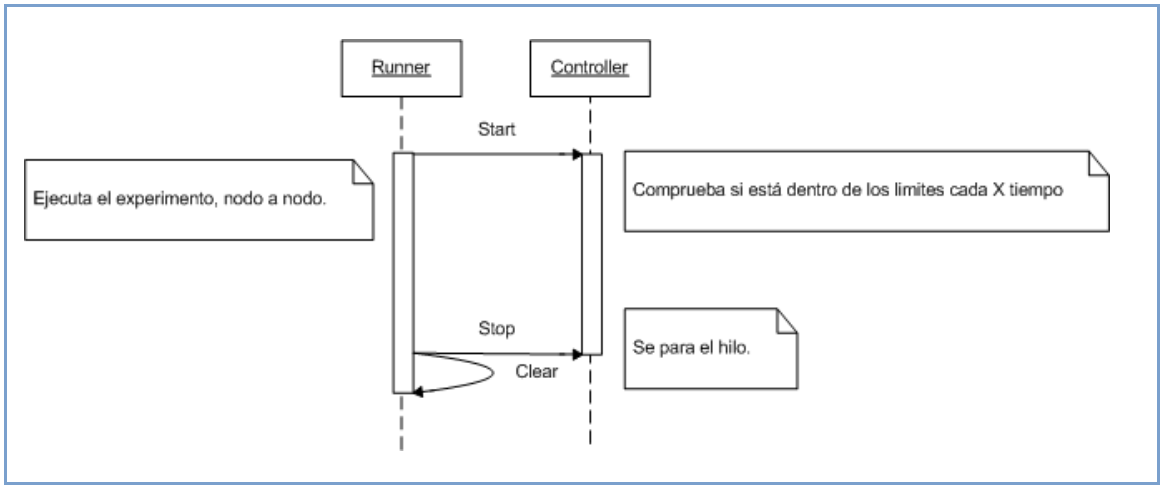


Figura 6. 19. Ejemplo Ejecución Correcta.

A continuación se observa en la Figura 6.20 como se ha tratado dicha especificación en el caso contrario, es decir, cuando la ejecución del experimento ha sobrepasado alguno de los límites.

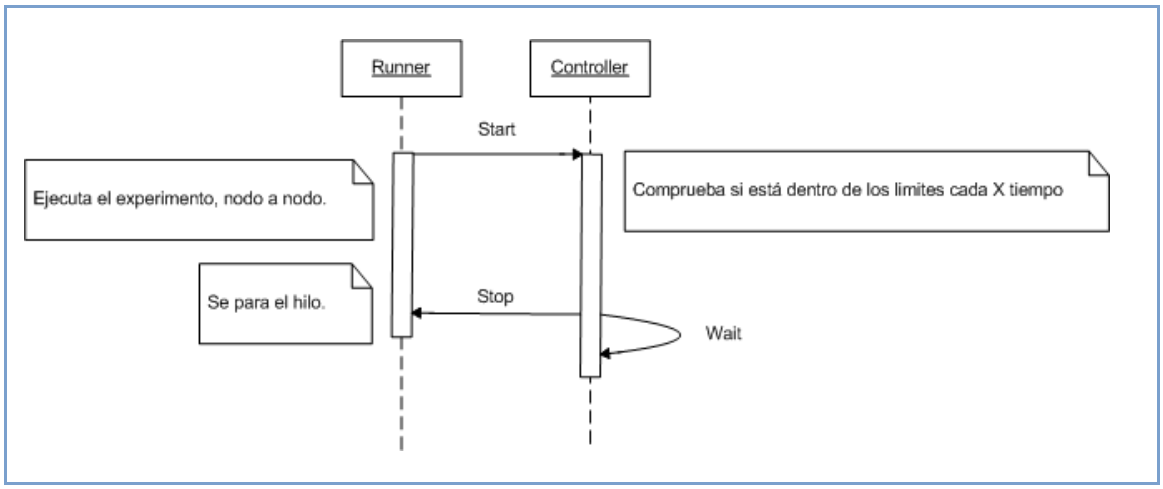


Figura 6. 20. Ejemplo de exceso de tiempo o espacio.

GESTIÓN DEL PROYECTO

En las siguientes páginas se detallan la planificación y el presupuesto para el proyecto de desarrollo del Back-End. Los objetivos del capítulo son por un lado, estimar los recursos económicos y humanos necesarios para la realización del proyecto y por otro, establecer un plan para definir la utilización de los recursos a lo largo de las diferentes etapas del mismo.

7.1. INTRODUCCIÓN

La planificación surge de la necesidad de estimar los costes económicos y temporales que requiere el desarrollo de un proyecto así como del control de los factores que pueden alterar su evolución [Humphrey, 1995]. En base al conjunto de funcionalidades propuestas, se hace imprescindible definir un calendario de trabajo con el fin de ordenar la ejecución de cada una de las tareas necesarias para su implementación.

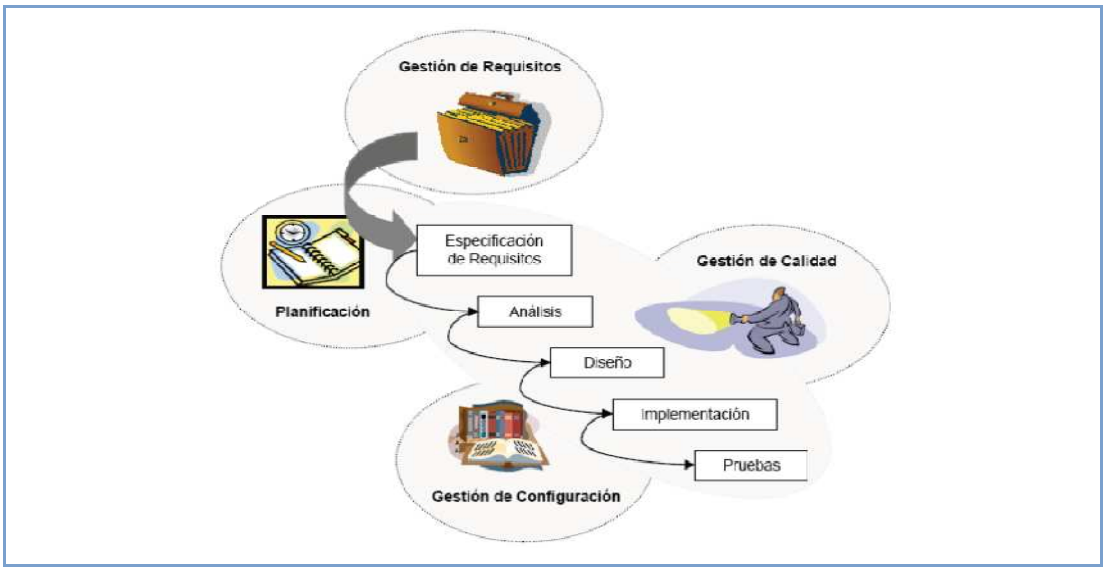


Figura 7.1. Representación del Proceso de Desarrollo Software.

El proceso software es la descripción de las etapas que se siguen durante la ejecución de un proyecto. Su correcta definición permite asegurar una adecuada asignación de recursos y conocer el estado del proyecto en cada momento. Establece el marco de trabajo, tanto técnico como de gestión, en la

aplicación de los métodos, las herramientas y las personas a las tareas de desarrollo de software [Cuevas, 2003]. Su definición identifica los roles y las tareas específicas y establece medidas para el control de la ejecución de cada paso.

Una metodología de trabajo definida permite que cada nuevo proyecto sea construido en base a la propia experiencia y a la de los predecesores. Su utilización permite identificar las causas de los problemas para corregirlos. En la Figura 7.1 se muestra el proceso donde se unen las actividades de desarrollo y gestión. Estas últimas se orientan a controlar el desarrollo del proceso software y a corregir las desviaciones con respecto a los parámetros de calidad establecidos. Los procesos de gestión se ejecutan de manera paralela a las operaciones de desarrollo del proyecto.

7.2. METODOLOGÍA

El ciclo de vida es el periodo que comienza cuando se concibe un producto software y termina cuando dicho producto deja de estar disponible. Se divide normalmente en fases que estructuran y organizan las etapas de concepción, desarrollo y mantenimiento del sistema software.

Debido a la naturaleza del proyecto se ha utilizado el ciclo de vida en espiral. En este modelo, el proyecto se ataca en una serie de ciclos de vida cortos, cada uno de los cuales finaliza con una revisión de software ejecutable.

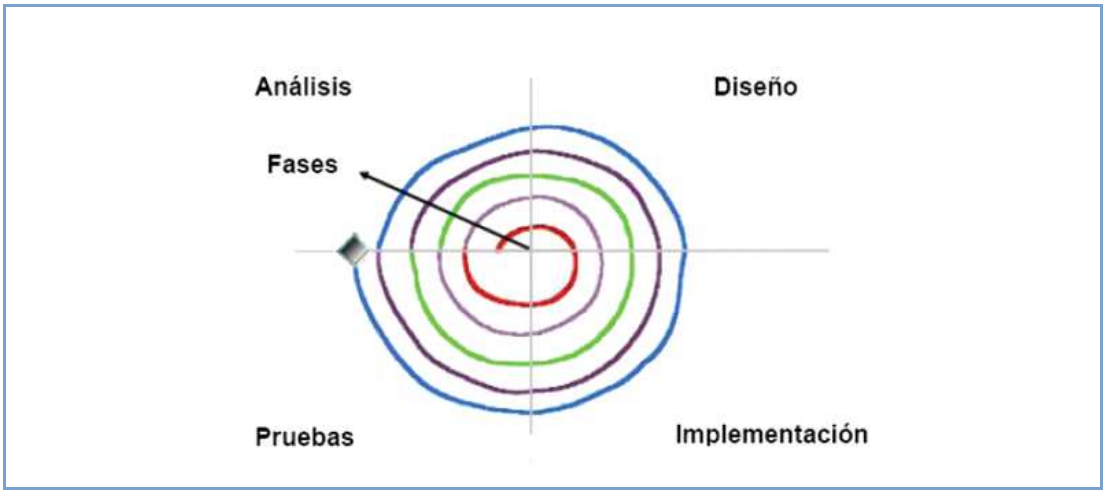


Figura 7. 2. Ciclo de Vida en Espiral.

En la Figura 7. 2 se muestra un ejemplo de desarrollo completado en cinco iteraciones del ciclo. Este ciclo de vida es especialmente recomendable cuando se planea hacer múltiples entregas del software. En cada una de éstas se desarrollan todas las fases del ciclo y se incorporan las experiencias de las entregas anteriores, con sucesivas etapas de especificación, construcción, explotación y revisión para obtener un refinamiento del producto de cara al siguiente ciclo.

Este ciclo de vida se usa en proyectos de gran tamaño o larga duración. Se emplea también en aquellos que necesitan tecnologías muy avanzadas o cuando se requiere la experiencia del usuario para refinar el diseño. Las principales motivaciones que han llevado a su elección son las siguientes:

- Complejidad del sistema: Debido a la complejidad del proyecto y a los recursos disponibles se ha abordado el desarrollo cada uno de los componentes de la plataforma por separado en varios ciclos del modelo. Esto ha permitido corregir anomalías y desviaciones.

- Riesgo tecnológico: El ciclo de vida adoptado permite asumir ciertos riesgos derivados del desconocimiento tecnológico.

- Dinamismo y adaptabilidad: El ciclo de vida acerca el sistema a la solución de forma progresiva al permitir el desarrollo en paralelo y la especialización del equipo de desarrollo en función del componente a implementar.

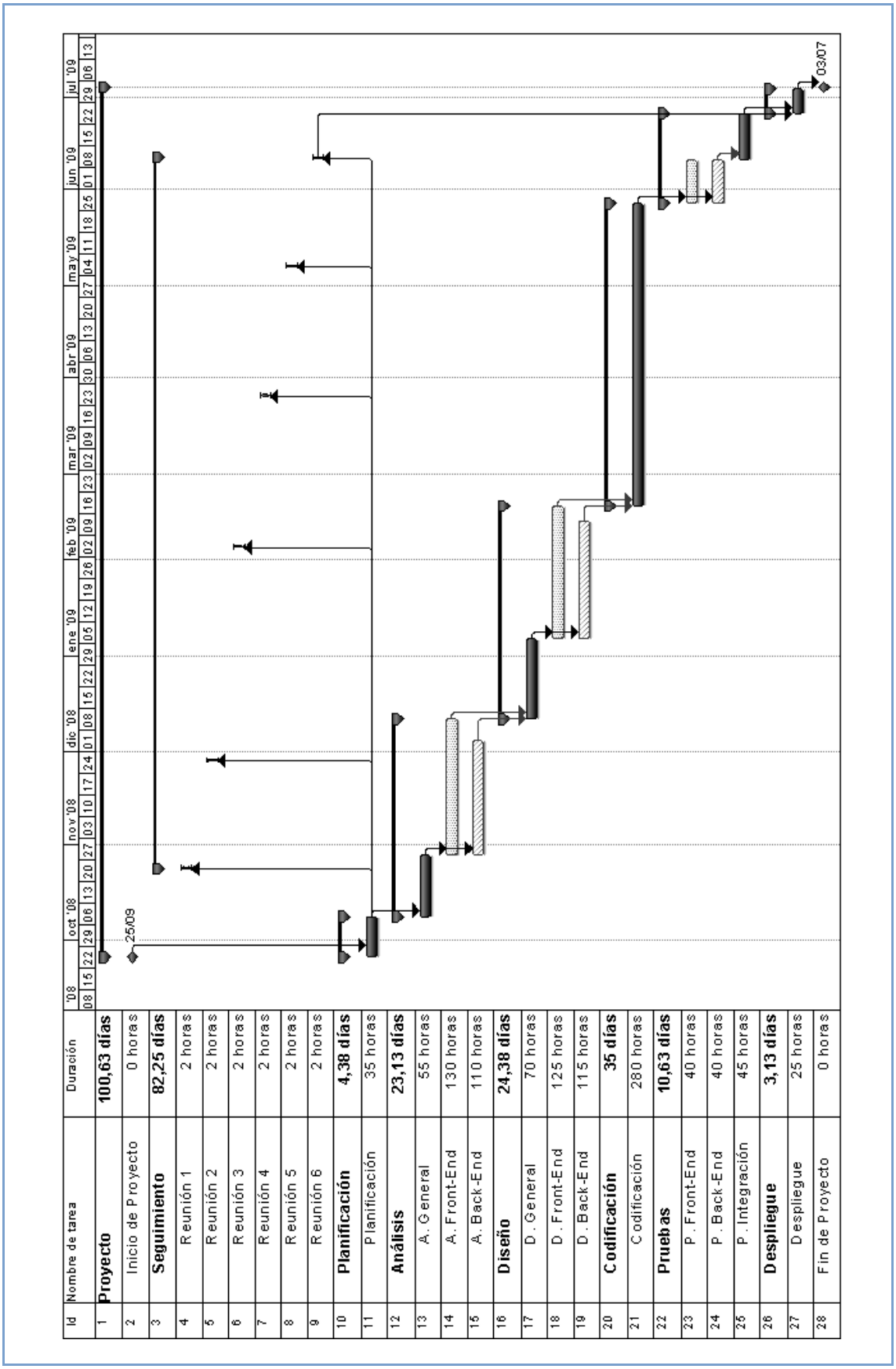
7.3. ESTIMACIÓN DE RECURSOS TEMPORALES

La fecha de inicio del proyecto se establece el día 25 de Septiembre de 2008 y la fecha de finalización en base a la actual planificación se emplaza el día 3 de Julio de 2009. Entre estas dos fechas se sitúa el desarrollo del proyecto con un total de 817 horas. El reparto de horas para cada una de las tareas del proyecto se presenta en la Tabla 7.1.

	TAREA	HORAS
1	Planificación	35
2	Análisis	165
3	Diseño	185
4	Codificación	280
5	Pruebas	85
6	Despliegue	25
7	Seguimiento	12
	TOTALES:	**817**

Tabla 7.1. Recursos Temporales por Fases del Proyecto

El diagrama de Gantt de la Figura 7.3 muestra la sucesión de cada una de las tareas que componen el desarrollo del proyecto.



Id	Nombre de tarea	Duración
1	**Proyecto**	**100,63 días**
2	Inicio de Proyecto	0 horas
3	**Seguimiento**	**82,25 días**
4	Reunión 1	2 horas
5	Reunión 2	2 horas
6	Reunión 3	2 horas
7	Reunión 4	2 horas
8	Reunión 5	2 horas
9	Reunión 6	2 horas
10	**Planificación**	**4,38 días**
11	Planificación	35 horas
12	**Análisis**	**23,13 días**
13	A. General	55 horas
14	A. Front-End	130 horas
15	A. Back-End	110 horas
16	**Diseño**	**24,38 días**
17	D. General	70 horas
18	D. Front-End	125 horas
19	D. Back-End	115 horas
20	**Codificación**	**35 días**
21	Codificación	280 horas
22	**Pruebas**	**10,63 días**
23	P. Front-End	40 horas
24	P. Back-End	40 horas
25	P. Integración	45 horas
26	**Despliegue**	**3,13 días**
27	Despliegue	25 horas
28	Fin de Proyecto	0 horas

Figura 7.3. Diagrama de Gantt.

7.4. ESTIMACIÓN DE RECURSOS ECONÓMICOS

En las siguientes secciones se determina el coste asociado a la ejecución del proyecto.

7.4.1. Recursos Materiales

La estimación del coste y de los periodos de amortización de los recursos materiales se han obtenido en base a lo descrito en [RD1777, 2004]. En este documento se establece un plazo de amortización de 8 años para los elementos de tratamiento de información y de 6 años para los programas informáticos. En base a esta información, el coste asociado a cada elemento viene determinado por la expresión de la siguiente figura.

Figura 7. 4. Expresión para el Cálculo de las Amortizaciones.

CONCEPTO	PRECIO	COEFICIENTE	COSTE
PC Portátil MSI	864 €	9,40 %	81,22 €
Impresora HP 5550	53,55 €	9,40 %	5,03 €
Conexión ADSL a 6 Mbps durante 12 meses a 39,90 €/mes.	359,10 €	100,00 %	359,10 €
Microsoft Office 2007	219,27 €	12,50 %	27,41 €
Microsoft Project 2007	149,00 €	12,50 %	18,63 €
Microsoft Visio 2007	221,00 €	12,50 %	27,63 €
Microsoft Visual Studio 2005	512,32 €	12,50 %	64,04 €
Macromeadia Fireworks 2004	404,84€	12,50 %	50,61 €
		TOTAL:	**633,65€**

Tabla 7.2. Costes de los Recursos Materiales para el Desarrollo del Proyecto.

7.4.2. Recursos Humanos

A partir de los recursos temporales presentados en la sección 7.3 y de los perfiles profesionales descritos a continuación, en el siguiente epígrafe se determina el coste asociado a los recursos humanos. En la Tabla 7.3 se muestran los perfiles de los profesionales requeridos para la ejecución de las diferentes tareas y los costes de cada uno de ellos.

DESCRIPCIÓN	€/HORA	HORAS	COSTE
Director de Proyecto	113,00 €	32	3.616 €
Jefe de Proyecto	75,00 €	64	4.800 €
Responsable de Calidad	63,00 €	60	3.780 €
Responsable de la Gestión de la configuración	63,00 €	60	3.780 €
Analista	63,00 €	169	10.647 €
Desarrollador	50,00 €	300	15.000 €
Probador de Software	25,00 €	102	2550 €
		TOTAL:	**44.173,00 €**

Tabla 7.3. Costes de los Recursos Humanos.

Los perfiles profesionales requeridos para desarrollar cada una de las tareas son los siguientes:

- Director de Proyecto: Profesional titulado en Ingeniería Informática con experiencia. Requiere habilidades como liderazgo, capacidad de comunicación e intermediación en conflictos. Los honorarios de este profesional se sitúan en 113 €/hora.

- Jefe de Proyecto: Profesional titulado en Ingeniería Informática con amplia formación en gestión de proyectos y recursos humanos. Los honorarios de este profesional se encuentran en 75 €/hora.

- Responsable de Calidad: titulado en Ingeniería Informática con prestigio, credibilidad, conocimientos y fácil llegada a los trabajadores. Los honorarios de este profesional se encuentran en 63 €/hora.

- Responsable de Gestión de la Configuración: titulado en Ingeniería Informática con capacidad para gestionar la planificación, identificación, control, seguimiento y auditoría de todos los elementos de configuración del sistema. Los honorarios de este profesional se encuentran en 63 €/hora.

- Analista: Profesional titulado en Ingeniería Informática con conocimientos de arquitectura de redes, protocolos de comunicaciones y entorno de desarrollo .NET. Así mismo, ha de poseer conocimientos en biometría. Los honorarios de este profesional se encuentra en 63 €/hora.

- Desarrollador: Profesional titulado en Ingeniería Informática con conocimientos de implementación en entorno .NET y otras tecnologías asociadas. Sus honorarios se encuentran en torno a los 50 €/hora.

- Probador de Software: Profesional titulado en Ciclo Formativo de Grado Superior en Informática. Su función es la de validar la aplicación a nivel de usuario y reportar los problemas. Sus honorarios se encuentran en 25 €/hora.

7.4.3. Costes Totales

El coste total asociado al desarrollo del proyecto asciende a 77.793,50 € (setenta y siete mil setecientos noventa y tres euros con cincuenta céntimos). De los cuales 74.668,62 €, corresponden a la implementación del Back-Office y 3.124,88 € a los costes soportados por el cliente. Las siguientes tablas recogen dichos valores:

CONCEPTO	VALOR
Recursos Materiales Necesarios para el Proyecto	633,65 €
Recursos Humanos	44.173,00 €
Gastos Generales (20% RRHH)	8.834,60 €
Subtotal	**53.641,25 €**
Beneficios Empresariales (20% Subtotal)	10.728.25 €
Base Imponible	**64.369,50 €**
I.V.A. (16%)	10.299,112 €
TOTAL:	**74.668,62 €**

Tabla 7.4. Costes del Proyecto.

CONCEPTO	PRECIO/ UNIDAD	UNIDADES	COSTE
Servidor HP Proliant 155	2.863,00 €	1	2.863,00 €
Solicitud de Registro de Dominio	26.88 €/año	1	26.88 €
Solicitud de Registro de Marca	235 €	1	235 €
		TOTAL (IVA incluido):	**3.124,88 €**

Tabla 7.5. Costes de los Recursos Materiales Necesarios para el Cliente.

7.5. PLAN DEL PROYECTO

A lo largo del siguiente apartado se describe el plan del proyecto. En primer lugar se encuentra el plan de fases del proyecto, donde se describe cada una de las tareas que componen la planificación. El plan de proyecto continúa con el plan de modificaciones, plan que describe el mecanismo de control de las modificaciones requeridas por el desarrollo del proyecto. Para finalizar se incluye el plan de revisiones e informes, donde se analiza el modo de informar sobre el estado del proyecto y se definen las revisiones formales asociadas al avance del mismo.

7.5.1. Plan de Fases del Proyecto

El desarrollo del presente proyecto se ha dividido en un total de seis etapas. A continuación se particularizan los objetivos y las tareas a desarrollar en cada una de las etapas.

7.5.1.1. Tarea 1. Planificación

La etapa de planificación pretende abordar los costes temporales y económicos de la ejecución del proyecto. Esta tarea se corresponde con la elaboración de la planificación contenida en el actual capítulo.

7.5.1.2. Tarea 2. Análisis

Se realiza una descripción completa de los requisitos funcionales del Back-End de la plataforma, es decir, analizar las funcionalidades que el Back-End ha de ofrecer a los usuarios de la aplicación. Se puede subdividir en varias subtareas:

- Captura de los requisitos: Obtención de los requisitos de usuario que ha de satisfacer el sistema.

- Definición general de la plataforma: Se presenta su arquitectura general y se muestran los diferentes componentes que forman parte de ella.

- Definición del álgebra del metamodelo: Durante esta fase se definen la interpretación y la semántica del lenguaje de modelado de los experimentos biométricos.

- Definición del modelo de datos: Esta etapa consiste en la definición de las entidades presentes en la aplicación y las relaciones existentes entre ellas.

- Definición de los perfiles de usuario: Se describen los diferentes perfiles de usuario presentes en la aplicación y los roles o funcionalidades asociadas a cada uno de ellos. Se completa con un diseño de casos de uso donde se proporcionan los escenarios de interacción del Back-End con los usuarios.

- Definición de los requisitos de software: La presente etapa finaliza con la

redacción de los requisitos de software que ha de cumplir el Back-End.

7.5.1.3. Tarea 3. Diseño

A lo largo de la tarea de diseño se efectúa la especificación de los componentes de la plataforma, la definición de los interfaces entre sistemas y la adopción de la tecnología para implementar la aplicación. La presente tarea se subdivide en varias fases.

- Adopción de la tecnología: En primer lugar se describe la adopción de las tecnologías para la implementación de la aplicación.
- Definición del patrón de diseño: Durante esta etapa se pretende definir el patrón de diseño de los componentes que forman los diferentes elementos de la plataforma.
- Definición del interface de comunicación entre el front y el Back-End: La presente fase pretende definir el interface y el protocolo de comunicación entre los dos principales subsistemas de la plataforma.
- Diseño del sistema: En la fase de diseño se acomete la elaboración de los elementos que compondrá el sistema para contemplar todos los requisitos que debe cumplir.
- Definición de requisitos no funcionales: La etapa de diseño concluye con la redacción de los requisitos no funcionales de la aplicación.

7.5.1.4. Tarea 4. Codificación

La tarea de codificación aborda la implementación de las funcionalidades propuestas en etapas anteriores. La codificación se ha subdividido en varias fases.

- Implementación del metamodelo: Hace referencia a la codificación de los sistemas necesarios para realizar operaciones de lectura, escritura y validación de modelos de experimentos biométricos.
- Implementación del interface de comunicación: Se desarrolla la implementación del sistema de comunicación entre el Front-End y el Back-End de la plataforma.
- Implementación del sistema: Se realizan las implementaciones de cada uno de los elementos que conforman el sistema.

7.5.1.5. Tarea 5. Pruebas

La fase de pruebas hace referencia a la evaluación del funcionamiento de la plataforma bajo entornos de ejecución genéricos. Dentro de la fase de pruebas se diferencias varias etapas:

- Pruebas unitarias de los componentes: Son una forma de probar el

correcto funcionamiento de cada uno de los módulos de código. Su objetivo es aislar cada parte del programa y mostrar que las partes individuales ofrecen un correcto funcionamiento.

- Pruebas de integración de componentes: Son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las unitarias. Éstas se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso.

- Pruebas de integración entre el front-end y el back-end: Este conjunto de pruebas pretende verificar que el front-end y el back-end funcionan correctamente en el servidor y cumplen todas las funcionalidades para los que fueron implementados.

- Pruebas de usabilidad y navegación: Hacen referencia al conjunto de pruebas realizadas para verificar que los elementos gráficos del front-end responden de forma correcta a la interacción del usuario.

7.5.1.6. Tarea 6. Despliegue

La fase de despliegue se corresponde con la puesta en marcha de la plataforma en el servidor una vez que las pruebas se han realizado de forma satisfactoria.

7.5.2. Plan de Control de Modificaciones

Para realizar la codificación se ha empleado la herramienta Visual Studio de Microsoft. Este sistema ofrece la herramienta Source Safe [MSDN, 2009a], cuya función principal consiste en realizar un seguimiento de los historiales de los archivos y proyectos. La base de datos de Source Safe incluye todas las versiones de un archivo o proyecto y permite ver el historial para recuperar cualquier versión de estos elementos.

Visual Studio también posee un sistema de control de versiones llamado Team Foundation Server [MSDN, 2009b], el cual ofrece colaboración en equipo, control de versiones, gestión de cambios, administración de la generación y elaboración de informes.

7.5.3. Plan de Seguimiento del Proyecto

A lo largo del proyecto se ha realizado un seguimiento quincenal de la evolución. Así mismo, se han propuesto reuniones mensuales para evaluar los objetivos alcanzados y el desarrollo conjunto del proyecto. En cada una de estas reuniones se ha planteado una revisión del trabajo realizado y los posibles cambios para mejorarlo. En el *ANEXO C: SEGUIMIENTO DEL PROYECTO* se recogen las principales reuniones de seguimiento del proyecto

7.6. HERRAMIENTAS

A continuación se presentan las herramientas utilizadas durante el desarrollo del proyecto tanto para la implementación de la aplicación como para la gestión del proyecto y la redacción de la memoria.

7.6.1. Microsoft Visual Studio 2005

Microsoft Visual Studio [12] es un entorno de desarrollo integrado para sistemas basados en Windows. Soporta lenguajes de programación como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET y actualmente se están desarrollado extensiones para muchos otros lenguajes. Visual Studio permite crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma. Así, se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

Visual Studio se ha utilizado para codificar las diferentes aplicaciones que forman parte de la plataforma y para definir el metamodelo de experimentación. Para realizar la documentación del código generado se ha empleado la herramienta Sandcaslte [13], un plugin para Visual Studio que permite la documentación de código.

7.6.2. Microsoft Office 2007

Microsoft Office 2007[14] es la versión más reciente de la suite ofimática de Microsoft. El paquete ofimático cuenta con herramientas para editar textos, realizar hojas de cálculo, presentaciones de diapositivas y otras muchas aplicaciones.

Para la elaboración de la presente memoria se ha empleado MS Word, para la realización de los diagramas UML se ha utilizado MS Visio, para realizar la planificación del proyecto se ha empleado MS Project y para calcular los costes del proyecto se ha empleado MS Excel.

7.6.3. Macromedia Fireworks 2004

Fireworks[15] es una aplicación en forma de estudio destinada al manejo híbrido de gráficos vectoriales y gráficos en mapa de bits. Permite la creación de prototipos de sitios web, interfaces de usuario e imágenes. Fireworks está enfocado en la creación y edición de gráficos para internet.

Fireworks se he empleado principalmente en la creación y edición de los

[12] http://www.microsoft.com/spanish/msdn/vs2005/default.mspx
[13] http://www.codeplex.com/Sandcastle
[14] http://office.microsoft.com/es-es/default.aspx
[15] http://www.adobe.com/products/fireworks/

elementos visuales de los interfaces gráficos de la aplicación web y del editor de experimentos.

7.6.4. Mozilla Firefox 3

Mozilla Firefox[16] es un navegador de Internet libre y de código abierto descendiente de Mozilla Application Suite, desarrollado por la Corporación Mozilla, la Fundación Mozilla y un gran número de voluntarios externos. Firefox es un navegador multiplataforma y está disponible en versiones para Microsoft Windows, Mac OS X, GNU/Linux y algunos sistemas basados en Unix.

Firefox se ha utilizado para evaluar el funcionamiento de la aplicación web de la plataforma.

7.6.5. Mendeley

Mendeley[17] es una aplicación que permite organizar referencias bibliográficas y generar citas en los formatos especificados de una manera elegante y estándar. Ofrece la integración con Microsoft Office Word para insertar citas desde una librería personal o directamente desde cualquier documento abierto. Así mismo, ofrece la posibilidad de acceder a las referencias de la librería desde el sitio web para compartirlas con otros usuarios.

Mendeley se ha utilizado en la gestión de las referencias bibliográficas de la presente memoria.

[16] http://www.mozilla-europe.org/es/firefox/
[17] http://www.mendeley.com/

CONCLUSIONES Y TRABAJO FUTURO

Por último, en este capítulo se comparan los resultados obtenidos con los objetivos planteados inicialmente. De esta manera se pueden extraer conclusiones sobre el trabajo realizado así como otras reflexiones personales, fruto del esfuerzo de tantos meses.

8.1. CONCLUSIONES

Las conclusiones presentadas a continuación resultan de la confrontación entre los objetivos planteados al inicio del proyecto y los conseguidos tras la finalización del mismo. Es posible que algunos pensamientos reflejen una realidad subjetiva discutible, por lo que deben entenderse como valoraciones personales.

Una de las tareas más difíciles a la hora de desarrollar una aplicación software es establecer las condiciones de satisfacción para que el trabajo se dé por concluido. Para establecer el grado de éxito del proyecto se recuerdan los objetivos generales de la plataforma y se muestra el grado de dificultad de la consecución de cada uno de ellos. Los objetivos que se han cubierto con el trabajo realizado son los siguientes:

- Se ha desarrollado un metamodelo capaz de describir las operaciones y los datos implicados en la ejecución de experimentos biométricos.
- Se ha conseguido desarrollar una plataforma que permite crear y modificar experimentos biométricos de acuerdo al metamodelo propuesto y con independencia de la ubicación del usuario.
- Se ha implementado un sistema capaz de generar descripciones de datos biométricos para bases de datos MySQL, SQL Server y Access.
- Se ha logrado desarrollar una aplicación que salvaguarda la información sensible de las muestras biométricas empleadas en la ejecución de los experimentos.
- Se ha implementado un sistema capaz de generar descripciones de algoritmos basados en librerías de código para el entorno de ejecución .NET.

- Se ha desarrollado un sistema capaz de interpretar las definiciones de los experimentos con el objetivo de ejecutar los procesos necesarios para evaluar los sistemas biométricos modelados.

- Se ha realizado un sistema capaz de generar y presentar informes de prestaciones a partir de los resultados obtenidos tras la ejecución de las definiciones de los experimentos.

- Por último, se ha desarrollado una arquitectura de comunicación para permitir la comunicación entre los sistemas de la plataforma, asegurando la independencia de los mismos.

El grado de cumplimiento de los objetivos planteados al inicio puede considerarse satisfactorio. La unión e integración exitosa de las diferentes aplicaciones que componen la plataforma va a permitir la realización de experimentos bajo un entorno de experimentación común.

Un punto clave en el desarrollo del proyecto ha sido el trabajo en equipo. Gran parte de los subsistemas de la plataforma han requerido un fuerte trabajo en equipo para conseguir su implementación. El trabajo en equipo ha sido clave en el desarrollo del interface de comunicación y en la definición del metamodelo.

8.2. TRABAJO FUTURO

En esta sección se enumeran algunas propuestas para trabajos futuros:

- Agregar nuevas operaciones primitivas al metamodelo con el fin de ampliar las operaciones sobre los datos biométricos.

- Optimizar el tiempo de ejecución de los experimentos mediante la paralelización de tareas.

- Agregar nuevos tipos de bases de datos.

- Eliminar la base de datos sobre la que se almacenan las tablas intermedias durante la ejecución del experimento y cambiarlas por tablas intermedias en memoria.

- Mejorar los mecanismos de seguridad de datos en la comunicación entre la aplicación web y el editor de experimentos.

- Ofrecer un mecanismo de intercambio de resultados entre las entidades de investigación con el objetivo de ofrecer un marco de comparación de resultados.

- Desarrollar un nuevo subsistema para permitir ejecutar algoritmos escritos en java.

REFERENCIAS

[Amber, 2004]

S. Amber: “The Object Primer. Agile Model Driven Development with UML 2”. 3rd Ed. Cambridge University Press. 2004.

[Atkinson et al, 2003]

C. Atkinson and T. Kühne: “Model-Driven Development. A meta-modeling foundation”. IEEE Software, May 2003, pp. 36-41. 2003.

[Atzeni et al, 1998]

P. Atzeni, G. Mecca and P. Merialdo: “Design and maintenance of data-intensive web sites”. Advances in Database Technology, EDBT '98. LNCS, vol. 1377, pp. 436-450. Springer. 1998.

[Blackburn, 2004]

D. Blackburn: “Biometrics 101”. Federal Bureau of Investigation. 2004.

[Boehm et al, 2007]

B.W. Boehm and R.W. Selby: “Software engineering: Barry W. Boehm's lifetime contributions to software development, management, and research”. Ed. Wiley-IEEE. 2007.

[Bolle et al, 2003]

R. M. Bolle, J. H. Connell, S. Pankanti, N. K. Ratha, and A. W. Senior: “Guide to Biometrics”. New-York: Springer-Verlag. 2003.

[Boyd et al, 2005]

J.E. Boyd and J.J. Little: “Biometric Gait Recognition”. LNCS, vol. 3161, pp.19-42. Springer. 2005.

[Buschmann, 1996]

F. Buschmann: “Pattern-Oriented Software Architecture. A System of Patterns”. Ed. John Wiley. 1996.

[Capelli et al, 2006]

R. Cappelli, D. Maio, D. Maltoni, J.L. Wayman and A.K. Jain.: “Performance evaluation of fingerprint verification systems”. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no.1, pp. 3-18. 2006.

[Chang et al, 1993]

H.D. Chang, J.F. Wang, and H.M. Suen: “Dynamic handwritten Chinese signature verification”. Proceedings of Second IEEE International Conference on Document Analysis and Recognition, pp. 258-261. 1993.

[Codd, 1970]

E. Codd: “A relational model of data for large shared data banks”. Communications of the ACM, vol. 13, no. 6, pp. 377-387. IBM Research Laboratory, San Jose, California. 1970.

[Cuevas, 2003]

A. Cuevas: “Gestión del proceso software”. Centro de Estudios Ramón Areces, S.A. 2003

[Dessimoz et al, 2005]

D. Dessimoz, J. Richiardi, C. Champod and A. Drygajlo: “Multimodal biometrics for identity documents”. University of Lausanne and EPFL (European Biometrics Portal). Technical Report PFS 341-08.05. 2005.

[Duta et al, 2002]

N. Duta, A.K. Jain and K.V. Mardia: “Matching of palm print”. Pattern Recognition Letters, vol. 23, pp. 477-485. 2002.

[ESA, 1991]

ESA: “Software Engineering Standards. Issue 2”. ESA Board for Software Standardization and Control. European Space Agency. 1991.

[Fernández, 2001]

A. Fernández: “Introducción a UML”. Universidad de Vigo. 2001.

Disponible en: http://www-gris.det.uvigo.es/~avilas/UML/UML.html

[Fogarty et al, 2005]

J. Fogarty, R. Baker and S. Hudson: “Case studies in the use of ROC curve analysis for sensor-based estimates in human computer interaction”. Proceedings of the ACM International Conference Series. Canadian Human-Computer Communications Society. 2005.

[Gamma, 1998]

E. Gamma: “Design Patterns. Elements of Reusable Object-Oriented Software”. Ed. Adisson Wesley. 1998.

[Jain et al, 2004]

A.K. Jain, A. Ross and S. Prabhakar: “An introduction to biometric recognition”. IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Image and Video Based Biometrics, vol. 14, no. 1, pp. 4-20. 2004.

[Jain et al, 2005]

A.K. Jain, K. Nandakumar and A. Ross: “Score normalization in multimodal biometric systems”. Pattern Recognition, vol. 38, no. 12, pp. 2270-2285. 2005.

[Justino et al, 2001]

E. Justino, F. Bortolozzi and R. Sabourin: “Off-line signature verification using HMM for random, simple and skilled forgeries”. Proceedings of 6th International Conference on Document Analysis and Recognition, pp. 1031-1034. 2001.

[Hong et al, 1999]

L. Hong, A.K. Jain and S. Pankanti: “Can multibiométricos improve performance?” Proceedings of AutoID ’99, pp. 59-64. 1999.

[Huang et al, 1997]

Kai Huang and Hong Yan: “Off-line signature verification based on geometric feature extraction and neural network classification”. Pattern Recognition, vol. 30, no. 1, pp. 9-17. 1997.

[Humphrey, 1995]

W.S. Humphrey: “A discipline for software engineering”. Ed. Adisson Wesley. 1995.

[IEEE, 1998]

IEEE Standard 830-1998: “IEEE Recommended Practice for Software Requirements Specifications”.

Disponible en: http://standards.ieee.org/reading/ieee/std/se/830-1998.pdf

[Kobrun, 2001]

C. Kobrun: “Introduction to UML: Structural and use case modeling”. Object Modeling with OMG UML Tutorial Series. 2001

[Larman, 2003]

C. Larman: “UML y patrones”. Ed. Pearson Educación. 2ª ed. 2003.

[LOPD, 1999]

Jefatura del Estado: “Ley Orgánica 15/1999, de 13 de Diciembre, de Protección de Datos de Carácter Personal”. Boletín Oficial del Estado. 1999

Disponible en: http://www.boe.es/g/es/bases_datos/doc.php.

[Maltoni et al, 2003]

D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar: “Handbook of Finger-print Recognition”. Springer-Verlag. 2003.

[Mansfield et al, 2002]

A.J. Mansfield and J.L. Wayman: “Best practices in testing and reporting performances of biometric devices”. Centre for Mathematics and Scientific Computing. National Physical Laboratory. 2002.

[Martin et al, 2004]

A. Martin, M. Przybocki, and J. Campbell: “The NIST speaker recognition evaluation program”. Biometric Systems Technology, Design and Performance Evaluation, J. Wayman, A. Jain, D. Maltoni, and D. Maio, Eds. Springer-Verlag. 2004.

[Masek, 2003]

L. Masek: “Recognition of human iris patterns for biometric identification”. Bachelor of Engineering degree, School of Computer Science and Software Engineering, University of Wester Australia, 2003.

[Monrose et al, 2000]

F. Monrose A.D. Rubin: “Keystroke dynamics as a biometric for authentication”. Future Generation Computer Systems, vol. 16, pp. 351-359. Elseiver. 2000.

[Morville, 2006]

P. Morville: “Information architecture for the World Wide Web. Designing Large-Scale Web Sites”. O’Reilly Media. 2006.

[MSDN, 2007]

MSDN: “Fundamentos de la normalización de bases de datos”. Microsoft Ayuda y Soporte. 2007.

Disponible en: http://support.microsoft.com/kb/283878

[MSDN, 2009a]

MSDN: “Documentación de Visual Studio 2005. Control de Versiones”. 2009.

Disponible en: http://msdn.microsoft.com/es-es/library/shyhfx67(vs.80).aspx

[MSDN, 2009b]

MSDN: “Documentación de Visual Studio Team System”. 2009.

Disponible en: http://msdn.microsoft.com/es-es/library/fda2bad5(VS.80).aspx

[MSDN, 2009c]

MSDN: “Interprocess Communications. Pipes”. 2009.

Disponible en: http://msdn.microsoft.com/en-us/library/aa365780(VS.85).aspx

[MSDN, 2009d]

MSDN: “Interprocess Communications. Named Pipes”. 2009.

Disponible en: http://msdn.microsoft.com/en-us/library/aa365590(VS.85).aspx

[Nielsen, 2000]

J. Nielsen:” Designing web usability”. New Riders. 2000.

[Nixon et al, 2006]

M.S. Nixon, T.N. Tan and R. Chellappa: “Human identification based on gait”. International Series on Biometrics, Springer. 2006.

[Oglesby et al, 1988]

J. Oglesby and J. Mason: “Speaker recognition with a neural classifier”. Speech 88: Proceedings of the 7th Federation of Acoustical Societies of Europe, pp. 1357-1363. 1988.

[OMG, 2009]

OMG:” Unified Modeling Language specifications”. 2009.

Disponible en: http://www.omg.org/spec/UML/2.2/

[Pandit et al, 1998]

M. Pandit and J. Kittler: “Feature selection for a DTW-based speaker verification system”. IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 1998, vol. 2, pp. 769-772. 1998.

[Phillips et al, 2000]

P.J Phillips, A. Martin, C.L. Wilson and M. Przybocky: “An introduction to evaluating biometric systems”. Computer, vol. 33, no. 2. 2000.

[Phillips et al, 2003]

P.J. Phillips, P. Grother, R.J. Michels, D.M. Blackburn, E. Tabassi and J.M. Bone: “Facial recognition vendor test 2002, evaluation report”. 2003.

Disponible en: http://www.frvt.org/FRVT2002

[Piattini et al, 2006]

M. Piattini, E. Marcos, B. Calero B. Vela: “Tecnología y diseño de bases de datos”. RA-MA. 2006.

[Poritz, 1982]

A. Poritz: “Linear predictive Hidden Markov Models and the speech signal”. IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 1982, vol. 7, pp. 1291-1294. 1982.

[Prabhakar et al, 2003]

S. Prabhakar, S. Pankanti, and A. K. Jain: “Biometric recognition: Security and privacy concerns”. IEEE Security and Privacy, vol. 1, no. 2, pp. 33-42. 2003.

[Puente, 2007]

L. Puente: “BEEP-MM. Propuesta de un metamodelo de experimentación biométrica”.

[Puente et al,2008]

L. Puente Rodríguez, A. García Crespo, M. J. Poza Lara and B. Ruiz Mezcua: “Study of different fusion techniques for multimodal biometric authentication”. Proceedings of the 2008 IEEE International Conference on Wireless & Mobile Computing, Networking & Communication. pp. 666-671. 2008.

[Raphael et al, 1974]

D. E. Raphael and J.R. Young: “Automated Personal Identification”. SRI International. 1974.

[RD1777, 2004]

Ministerio de Economía y Hacienda: “Real Decreto 1777/2004, de 30 de Julio, por el que se aprueba el Reglamento del impuesto de Sociedades. Anexo: Tablas de Coeficientes de Amortización”. 2004.

Disponible en: http://www.boe.es/g/es/bases_datos/doc.php

[Renesse, 2002]

R. L. van Renesse: “Implications of applying biometrics to travel documents”. Proceedings of SPIE, vol. 4677, pp. 290-298. 2002.

[Reynolds, 1995]

D. A. Reynolds: “Speaker identification and verification using gaussian mixture speaker models”. Speech Communication, vol. 17, pp. 91-108. 1995.

[Richiardi et al, 2003]

J. Richiardi and A. Drygajlo: “Gaussian mixture models for on-line signature verification”. Proceedings of International Multimedia Conference 2003, pp. 115-122. 2003.

[Ruíz,1998]

B. Ruíz “Modelado Estadístico y Conexionista para Reconocimiento de Locutores con Aprendizaje de la Variabilidad Temporal del Habla”. Tesis Doctoral. 1998

[Saake et al, 1993]

G. Saake, R. Jungclaus and T. Hartmann: “Application modeling in heterogeneous environments using an object specification language”. Proceedings of CoopIS, pp. 309-318. 1993.

[Sato et al, 1982]

Y. Sato and K. Kogure: “Online signature verification based on shape, motion, and writing pressure”. Proceedings of 6th International Conference on Pattern Recognition, pp. 823-826. 1982.

[Sánchez et al, 2000]

R. Sánchez Reillo, C. Sánchez Avila and A. González Marcos: “Biometric identification through hand geometry measurements”. IEEE Transactions on Pattern Analysis and Matching, vol. 22, no. 10, pp. 1168-1171. 2000.

[Sánchez, 2000]

R. Sánchez Reillo: “Hand geometry pattern recognition through Gaussian Mixture Modeling”. Proceedings of the 15th Int. Conf. Pattern Recognition, vol. 2, pp. 941-944. 2000.

[Schmidt et al, 1996]

M. Schmidt and H. Gish: “Speaker identification via support vector classifiers”. IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 1996, vol. 1, pp. 105-108. 1996.

[Seidewitz, 2003]

E. Seidewitz: “What models mean”. IEEE Software, September 2003, pp. 26-32. 2003.

[Silberschatz et al, 2006]

A. Silberschatz, P.B. Galvin and G. Gagne: “Fundamentos de sistemas operativos”. Mc Graw Hill. 2006.

[Sirovich et al, 1987]

L. Sirovich and M. Kirby: “Low-dimensional procedure for the characterization of human faces”. Journal of the Optical Society of America, vol. 4, no. 3, pp. 519-524. 1987.

[Tisee, 2003]

C.L. Tisse : "Contribution à la vérification biométrique de personnes par reconnaissance de l’iris". Ph. D, Université de Montpellier. 2003.

[Van Duyne et al, 2002]

D.K. Van Duyne, J.A. Landay and J.I. Hong: “The design of sites. Patterns, principles and processes for crafting a customer-centered web experience”. Addison Wesley. 2002.

[W3C, 2004]

W3C Recommendation: “XML Information Set”. 2nd Ed. 2004.

Disponible en: http://www.w3.org/TR/xml-infoset/

[Wildes, 2005]

R. Wildes: “Iris recognition”. Biometric Systems: Technology, Design and Performance Evaluation, J. L. Wayman, A. K. Jain, D. Maltoni, and D. Maio, Eds. London: Springer-Verlag, pp. 63-95. 2005.

[Wolff, 1993]

K. E. Wolff: “A first course in formal concept analysis”. Fachhochschule Darmstadt. Forschungsgruppe Begriffsanalyse der Technischen Hochschule Darmstadt. 1993.

[Yager et al, 2004]

N. Yager and A. Amin: “Fingerprint verification based on minutiae features: a review”. Pattern Analysis and Application, vol. 17, pp. 94-113. 2004.

[Zhao et al, 2003]

W. Zhao, R. Chellappa, J. Phillips, and A. Rosenfeld: “Face recognition: A literature survey”. ACM Computing Surveys, vol. 35, no. 4, pp. 399-458. 2003.

ANEXO A. GLOSARIO

A lo largo de las siguientes páginas se presentan las definiciones de los conceptos utilizados a lo largo de la memoria.

Almacén de Información Biométrica

Tabla desde la que se extraen o insertan datos durante la ejecución de un experimento.

Caducidad

Concepto que hace referencia al nivel de actividad de las entidades de investigación. De forma general, una entidad está no caducada puesto que se produce el acceso a la misma por parte de sus miembros de forma continua y en plazos cortos de tiempo. Una entidad pasa a estar caducada cuando no se produce ninguna actividad durante un determinado periodo de tiempo y ningún usuario accede a ella. La caducidad de las entidades se establece para controlar los recursos de la plataforma y mantener el servidor libre de datos inservibles, de tal forma que aquellas entidades que no demuestran actividad alguna son eliminadas junto a sus espacios de trabajo.

Conexión

Define la cadena de conexión a la base de datos que contiene los datos de un almacén.

Definición de Experimento

Archivo XML que contiene el modelo de un experimento biométrico.

Descriptor de Contenido

Archivo XML que contiene la descripción de los campos y tipos de datos asociados a un almacén de información biométrica.

Descriptor de Interfaces

Archivo XML que define los parámetros de entrada, salida y configuración de los procesos de tratamiento de información biométrica.

Entidad de Investigación

Organización con fines de investigación externa a la plataforma que utiliza los recursos proporcionados por ésta. Una entidad investigadora está formada por grupos de trabajo y por usuarios.

Espacio de Trabajo

El espacio de trabajo hace referencia a los recursos de almacenamiento que la plataforma ofrece a cada una de las entidades. En el espacio de trabajo, una entidad investigadora puede almacenar procesos, experimentos, datos biométricos y los informes de resultados de la ejecución de los experimentos. El espacio de trabajo se comparte por todos los grupos dependientes de la entidad.

Grupo de Trabajo

Agrupación de usuarios de una entidad de investigación que posee acceso común a una serie de experimentos e informes de resultados.

Librería

Archivo de tipo DLL que contiene clases para su utilización en procesos.

Proceso

Conjunto de clases, obtenidas de una o más librerías de código, que transforman los datos biométricos que reciben.

ANEXO B. ESPECIFICACIÓN DE REQUISITOS

El presente anexo recoge la especificación de requisitos funcionales del sistema. En primer lugar, se ofrecen los requisitos generales del sistema (RG), a continuación, los requisitos de usuario (RU) y por último se presentan los requisitos de software (RF), es decir, los requisitos funcionales y no funcionales (RNF).

ID	ESENCIAL	MODIFICABLE	PRIORIDAD
RG-001	Si	No	Alta
DESCRIPCIÓN: La aplicación es autónoma.			
RG-002	Si	No	Alta
DESCRIPCIÓN: El sistema está comunicado con el Front-End mediante un protocolo de comunicación.			
RG-003	Si	No	Alta
DESCRIPCIÓN: La aplicación envía notificaciones de fin de ejecución al Front-End de la plataforma.			
RG-004	Si	No	Alta
DESCRIPCIÓN: La aplicación valida modelos de experimentos conforme a la especificación del metamodelo.			

ID	ESENCIAL	MODIFICABLE	PRIORIDAD
RG-005	Si	No	Alta
DESCRIPCIÓN: La aplicación ejecuta modelos de experimentos conforme a la especificación del metamodelo.			
RG-006	Si	No	Alta
DESCRIPCIÓN: La aplicación evalúa modelos de experimentos conforme a la especificación del metamodelo.			
RG-007	Si	No	Alta
DESCRIPCIÓN: El sistema ofrece un mecanismo de gestión de la cola de ejecución para la seleccionar el siguiente experimento a ejecutar.			

ID	ESENCIAL	MODIFICABLE	PRIORIDAD
RU-001	Si	No	Alta
DESCRIPCIÓN: La aplicación permite ejecutar experimentos independientemente de si el Front-End se encuentra activo o no.			
RU-002	Si	No	Alta
DESCRIPCIÓN: La aplicación se comunica con el Front-End mediante un protocolo de comunicación asíncrono.			
RU-003	No	No	Media
DESCRIPCIÓN: El sistema permite ejecutar experimentos mientras haya en la cola de ejecución experimentos pendientes.			
RU-004	Si	No	Alta
DESCRIPCIÓN: El sistema ofrece un mecanismo de gestión de la cola de ejecución para seleccionar el siguiente experimento a ejecutar si existiera más de uno.			
RU-005	Si	No	Alta
DESCRIPCIÓN: La aplicación recibe notificaciones del Front-End cada vez que un usuario sube un experimento a la cola de ejecución.			
RU-006	Si	No	Alta
DESCRIPCIÓN: El sistema envía una notificación al Front-End cada vez que comienza la ejecución de un experimento.			
RU-007	Si	No	Alta
DESCRIPCIÓN: El sistema envía una notificación al Front-End cada vez que finaliza la ejecución de un experimento.			

ID	ESENCIAL	MODIFICABLE	PRIORIDAD
RU-008	Si	No	Alta
DESCRIPCIÓN: La aplicación valida los modelos de experimentos según la especificación del metamodelo.			
RU-009	Si	No	Alta
DESCRIPCIÓN: La aplicación distingue los tipos de nodos que componen el modelo de experimentos para validarlos según el tipo.			
RU-010	No	No	Media
DESCRIPCIÓN: La aplicación ejecuta los modelos de experimentos según la especificación del metamodelo.			
RU-011	Si	No	Alta
DESCRIPCIÓN: La aplicación distingue los tipos de nodos que componen el modelo de experimentos para ejecutarlos según el tipo.			
RU-012	Si	No	Alta
DESCRIPCIÓN: La aplicación evalúa modelos de experimentos conforme a la especificación del metamodelo.			
RU-013	Si	No	Alta
DESCRIPCIÓN: El sistema permite ejecutar experimentos dentro de los límites de tiempo y espacio asignados a la entidad propietaria del experimento.			

ID	ESENCIAL	MODIFICABLE	PRIORIDAD
RF-001	Si	No	Alta
DESCRIPCIÓN: La aplicación funciona de forma autónoma.			
RF-002	Si	No	Alta
DESCRIPCIÓN: La aplicación permite validar modelos de experimentos biométricos de acuerdo al metamodelo establecido.			
RF-003	No	No	Baja
DESCRIPCIÓN: La aplicación permite ejecutar modelos de experimentos biométricos de acuerdo al metamodelo establecido.			
RF-004	Si	No	Alta
DESCRIPCIÓN: La aplicación permite evaluar modelos de experimentos biométricos de acuerdo al metamodelo establecido.			
RF-005	Si	No	Alta
DESCRIPCIÓN: El sistema ejecuta y valida los modelos de experimentos ordenando los nodos que componen la definición.			
RF-006	Si	No	Alta
DESCRIPCIÓN: El sistema permite obtener de la cola de ejecución el siguiente experimento a ejecutar.			
RF-007	Si	No	Alta
DESCRIPCIÓN: El sistema ofrece un mecanismo de gestión de la cola de ejecución para seleccionar el siguiente experimento a ejecutar si existiera más de uno.			

ID	ESENCIAL	MODIFICABLE	PRIORIDAD
RF-008	No	No	Media
DESCRIPCIÓN: La aplicación notifica al Front-End el comienzo de la ejecución de un experimento.			
RF-009	Si	No	Alta
DESCRIPCIÓN: La aplicación notifica al Front-End el fin de la ejecución de un experimento.			
RF-010	Si	No	Alta
DESCRIPCIÓN: El sistema tiene un espacio reservado para realizar las operaciones intermedias necesarias durante la ejecución.			
RF-011	Si	No	Alta
DESCRIPCIÓN: El sistema al finalizar la ejecución guardará los resultados de la misma dónde indicara el nodo destino.			
RF-012	No	No	Media
DESCRIPCIÓN: El sistema guarda los almacenes persistentes.			
RF-013	No	No	Media
DESCRIPCIÓN: El sistema utiliza almacenes de información pública y privada, dependiendo de lo que índice cada nodo.			
RF-014	Si	No	Alta
DESCRIPCIÓN: El sistema utiliza orígenes de datos de información pública y privada, dependiendo de lo que indique cada nodo.			

ID	ESENCIAL	MODIFICABLE	PRIORIDAD
RF-015	Si	No	Alta
DESCRIPCIÓN: La aplicación gestiona librerías de código de tipo DLL, tanto de código gestionado como no gestionado.			
RF-016	Si	No	Alta
DESCRIPCIÓN: La aplicación permite utilizar procesos para el tratamiento de los datos biométricos a partir de las librerías de código disponibles.			
RF-017	Si	No	Media
DESCRIPCIÓN: El sistema es capaz de generar descripciones de almacenes de información a partir de tablas MySQL, SQL Server y Access.			
RF-018	Si	No	Alta
DESCRIPCIÓN: El sistema es capaz de generar las descripciones de los almacenes generados tras la ejecución de un experimento.			
RF-019	Si	Si	Alta
DESCRIPCIÓN: La aplicación permite crear y eliminar conexiones a bases de datos para obtener nuevos almacenes de información.			
RF-020	Si	No	Media
DESCRIPCIÓN: Un almacén de información solamente puede encontrarse en la localización establecida por su conexión.			
RF-021	Si	No	Alta
DESCRIPCIÓN: La aplicación permite consultar y modificar los parámetros de configuración de la plataforma.			

ID	ESENCIAL	MODIFICABLE	PRIORIDAD
RNF-001	No	No	Media
DESCRIPCIÓN: El sistema cumplirá en todo momento la Ley Orgánica de Protección de Datos Personales.			
RNF-002	Si	No	Alta
DESCRIPCIÓN: La plataforma se ejecutará en Windows Server 2003.			

ANEXO C. SEGUIMIENTO DEL PROYECTO

ACTA DE REUNIÓN 1 (24/10/2008)
ASISTENTES: • Director de Proyecto: Luis Puente Rodríguez • Jefe del proyecto Back-End: Elena Arribas González • Jefe del proyecto Front-End: Diego Carrero Figueroa
ORDEN DEL DÍA: 1. Captura de los requisitos funcionales de la plataforma. 2. Definición de los requisitos del metamodelo. 3. Definición del alcance del Back-End.
DESARROLLO DE LA SESIÓN: 1. Se definieron los objetivos de la plataforma y se establecieron las restricciones impuestas para su implementación. 2. Se recogió la propuesta del metamodelo específico para el dominio de experimentación. 3. Se plantearon los objetivos a conseguir con la elaboración del proyecto.

ACTA DE REUNIÓN 2 (28/11/2008)
ASISTENTES: • Director de Proyecto: Luis Puente Rodríguez • Jefe del proyecto Back-End: Elena Arribas González • Jefe del proyecto Front-End: Diego Carrero Figueroa
ORDEN DEL DÍA: 1. Entrega del documento del alcance de la aplicación. 2. Revisión de los requisitos funcionales de la plataforma y del metamodelo. 3. Elección del ciclo de vida del proyecto. 4. Análisis del modelo de datos del sistema. 5. Definición del interface de comunicación entre los subsistemas.
DESARROLLO DE LA SESIÓN: 1. Se hizo entrega del documento de Alcance del sistema quedando su aprobación pendiente para la próxima reunión. 2. Se aprobaron los requisitos funcionales propuestos para los subsistemas de la plataforma. 3. Se optó por el ciclo de vida iterativo e incremental, debido a la complejidad del proyecto. 4. Se presentó el análisis conceptual a alto nivel sobre la base de dato de la plataforma y se eligió utilizar bases de datos independientes para el sistema y las entidades. 5. Se definió el infertace de comunicación, con el objetivo de asegurar la independencia de los subsistemas de la plataforma.

ACTA DE REUNIÓN 3 (05/02/2009)
ASISTENTES: • Director de Proyecto: Luis Puente Rodríguez • Jefe del proyecto Back-End: Elena Arribas González • Jefe del proyecto Front-End: Diego Carrero Figueroa
ORDEN DEL DÍA: 1. Revisión del alcance del proyecto. 2. Entrega del documento de análisis. 3. Presentación de los documentos relativos a la gestión y planificación del proyecto.
DESARROLLO DE LA SESIÓN: 1. Se aprobó el alcance del proyecto del Back-End. 2. Se entregó el documento de Análisis del Back-Office. 3. Se hizo entrega de los documentos de gestión del proyecto.

ACTA DE REUNIÓN 4 (26/03/2009)
ASISTENTES: • Director de Proyecto: Luis Puente Rodríguez • Jefe del proyecto Back-End: Elena Arribas González • Jefe del proyecto Front-End: Diego Carrero Figueroa
ORDEN DEL DÍA: 1. Revisión del análisis del proyecto. 2. Presentación del diseño. 3. Definición del contenido de la memoria del proyecto. 4. Resolución de dudas.
DESARROLLO DE LA SESIÓN: 1. Se aprobó el documento de análisis. 2. Establecimiento de las pautas del diseño para el Back-Office, se hizo entrega del documento de la definición del metamodelo y se eligieron las tecnologías para la implementación. 3. Se realizó un esbozo del contenido de la memoria y de cómo debía ser tratado cada uno de los apartados. 4. Se evaluó el estado del proyecto, y debido a retrasos en la planificación, se decidió reajustar la misma.

ACTA DE REUNIÓN 5 (07/05/2009)
ASISTENTES: • Director de Proyecto: Luis Puente Rodríguez • Jefe del proyecto Back-End: Elena Arribas González • Jefe del proyecto Front-End: Diego Carrero Figueroa
ORDEN DEL DÍA: 1. Entrega del documento de diseño. 2. Definición de aspectos referentes a la implementación.
DESARROLLO DE LA SESIÓN: 1. Se hizo entrega del documento de Diseño del Back-End y su evaluación quedó pendiente para la siguiente reunión. 2. Se eligió la implementación del interface de comunicación como una tubería con nombre y se decidió el uso de librerías de código común para ambos subsistemas con el fin de reutilizar el código.

ACTA DE REUNIÓN 6 (11/06/2009)
ASISTENTES: • Director de Proyecto: Luis Puente Rodríguez • Jefe del proyecto Back-End: Elena Arribas González • Jefe del proyecto Front-End: Diego Carrero Figueroa
ORDEN DEL DÍA: 1. Revisión del documento de Diseño del Back-End. 2. Presentación de resultados sobre la integración de los dos sistemas de la plataforma.
DESARROLLO DE LA SESIÓN: 1. Se aprobó el documento de Diseño. 2. Se propuso solución a los distintos problemas que surgieron durante la integración de los subsistemas.