Universidad de las Ciencias Informáticas

Facultad 7

Arquitectura para el sistema de identificación por perfiles de ADN

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

Autores: Rafael Fernández Peñalver

Alexander Rodríguez Bonet

Tutores: Msc. Eduardo Solís Céspedes

Ing. Alejandro Arias Naranjo

Ciudad de La Habana, febrero de 2009

"Año del 50 Aniversario del Triunfo de la Revolución"

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmamos la presente a los 27 días del mes de febrero del año 2009.

Autores:

Rafael Fernández Peñalver	Alexander Rodríguez Bonet

Tutores:

Msc. Eduardo Solís Céspedes	Ing. Alejandro Arias Naranjo

DATOS DE CONTACTO

Tutores:

Msc. Eduardo Solís Céspedes (esolis@uci.cu)

Profesor graduado de Licenciatura en Radioquímica y especialista en Bioinformática. Ha impartido las asignaturas de Física I, Física II y Práctica Profesional. Es profesor de la Facultad 7, actualmente se desempeña como Vice-decano de producción de la facultad.

Ing. Alejandro Arias Naranjo (aariasn@uci.cu)

Profesor graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas. Imparte la asignatura de Introducción a la Programación. Es Jefe de Proyecto dentro del Grupo de Procesamiento de Imágenes.

Dedicatoria

– De Rafael y Alexander

A Martí, Maceo, Mella, Camilo, Che y todos los que entregaron sus vidas defendiendo el honor y los derechos de este pueblo.

A los trabajadores cubanos, que continúan esforzándose con la esperanza y la certeza de que el desarrollo y la victoria nos corresponderán, sin ceder en nuestros principios.

– De Rafael

A Diana, mi princesita, por ser la luz que ilumina mi vida.

A mi madre, quien a pesar de estar lejos, no se ha separado de mi lado ni un segundo de su vida. Mi razón de existir.

A Eriet, mi padre, paradigma de hombre cuyo ejemplo guía mis pasos; quien ha sido más que tío, más que amigo, más que hermano.

– De Alexander

A mi madre, quien no pudo ver la concreción de parte de nuestros sueños, pero que siempre supo que se cumplirían.

A mi padre, que es y será siempre un ejemplo para mí; y que además critica lo mal hecho y estimula mis buenos actos.

A mi hermana, por todo su amor. Eres mi mejor amiga, mi refugio y mi hombro derecho.

Agradecimientos

– De Rafael y Alexander

A nuestros profesores, por abrir tantas puertas.

A nuestros compañeros de aula, de los que aprendimos mucho.

A GPI, por enseñarnos el valor de la consagración, la voluntad y el trabajo.

A la FEU y la UJC por brindarnos incalculables experiencias y permitirnos soñar y hacer.

A Fidel, por su genial idea de convertir el talento de los jóvenes en fuente de desarrollo para el país.

– De Rafael

A Lili, mi segunda madre: amiga, consejera, confidente. Gracias por existir.

A Yasmany, sin tu apoyo no hubiera sido posible este sueño.

A Tete, por sembrar en mí, desde muy pequeño, esa sed insaciable de conocimiento que me condujo hasta aquí.

A mi abuelo, a mis hermanos, a mis primos, a mis tíos y todos aquellos familiares que estuvieron siempre a mi lado dándome aliento para continuar.

A la "tropa Peque", a Robert, Anibal, Jacque y Moniquita por ser los hermanos que la vida me permitió escoger. Gracias por su apoyo y comprensión en todo momento.

– De Alexander

A mi familia, por ser el entorno ideal para mi crecimiento y desarrollo satisfactorios.

A mis amigos: por compartir todo sin previa solicitud, apoyarme, enseñarme, exigirme y comprenderme.

A mis vecinos de Alamar, por confiar tanto en mí, casi sin verme en los últimos años.

RESUMEN

El ADN está presente en las células del ser humano y es diferente en cada persona, por lo que es utilizado como método de identificación. Esta investigación propone la arquitectura para un software que almacene los perfiles de ADN, brinde a los especialistas los servicios de búsqueda y comparación entre perfiles, así como la generación de informes y reportes. Para identificar sospechosos de crímenes a través de su huella genética se enviarán solicitudes a la sede central de la policía nacional a través de mensajes de correo electrónico con una estructura definida.

La arquitectura propuesta es independiente a la lógica de comparación por lo que podrá ser utilizada en otros sistemas de identificación biométrica. Se desarrollará para Sistema Operativo GNU Linux, utilizando Java como lenguaje de programación y PostgreSQL como sistema gestor de bases de datos por las necesidades de independencia tecnológica, seguridad y adaptación.

Como resultado de este trabajo de diploma se realiza la propuesta de una arquitectura de sistema que posibilitará la implementación de una solución que automatice algunos procesos que incrementen la eficiencia del trabajo policial.

Palabras claves: Arquitectura, ADN, identificación, software.

TABLA DE CONTENIDO

INTRODUCCIÓN

Como salidas de una película de espías o de ciencia-ficción, las nuevas tecnologías de identificación por medio de sistemas biométricos se perfilan como la llave que abrirá todas las puertas en el futuro. Esta llave será el propio cuerpo, con sus características físicas únicas y distintas de las de cualquier otro. Serán la identificación por huellas dactilares, la geografía de la mano, el reconocimiento facial y del ADN, así como del iris o la voz, las nuevas claves de entrada a múltiples sistemas como: el acceso a cuentas bancarias, vehículos, áreas laborales y archivos informáticos.

El presente trabajo se enfoca en el empleo de la informática para mejorar los procesos de identificación a través del ADN.

Ante todo, se debe establecer que el ácido desoxirribonucleico, más conocido por las siglas ADN; contiene las instrucciones genéticas para el desarrollo y el funcionamiento de todos los organismos vivos. Su función principal es ser el portador y transmisor, entre generaciones, de la información genética.

El análisis del ADN humano tiene múltiples aplicaciones en el campo de las biociencias. Además, ha cobrado significativa importancia su utilización en dos áreas fundamentales: la salud y el sistema judicial. En la salud, el análisis se centra en el diagnóstico de enfermedades hereditarias, cromosomas defectuosos y cáncer; y en la otra, en la identificación de sospechosos en casos criminales –fundamentalmente en asesinatos, violaciones y otros actos violentos– y análisis de relaciones parentales en disputas sobre paternidad y casos de inmigración. Se ha demostrado su utilidad en accidentes marítimos y aéreos cuando las víctimas están en un estado donde sea imposible un reconocimiento visual. También se ha utilizado para las víctimas del terrorismo y la guerra, por ejemplo, en la guerra del Golfo en 1990 (Dorte Hammelev, 2008).

Estos análisis son posibles porque en los seres humanos, el núcleo de cada *célula somática* contiene 6.4 mil millones de pares de bases de ADN. El 99.9% del ADN humano es idéntico entre todos los individuos. Solamente el restante 0.1% puede variar de diversas formas de un individuo a otro. Es por esto que dicha porción o fragmento del ADN es denominado polimórfico (Roth, 1996).

Uno de los métodos de identificación biométrica más antiguo emplea las huellas dactilares. Con el desarrollo de las tecnologías actuales, se hizo posible la utilización del ADN con este objetivo debido a

que el mismo es extremadamente polimórfico entre los individuos y se encuentra en cada célula del organismo.

El proceso de separar los fragmentos polimórficos y únicos del ADN constituye una técnica conocida como **Huella Genética**. Este se puede realizar a partir de muestras de ADN presentes en la sangre, el semen, la piel, la saliva o el pelo.

En la actualidad, los periódicos publican diariamente historias de asesinatos y violaciones; para resolver estos casos se utiliza cada vez más la técnica de Huella Genética. Los perfiles de ADN extraídos de la muestra biológica encontrada en la escena del crimen son comparados con los de posibles sospechosos. Por lo que una de las principales problemáticas es que, al obtener el perfil de ADN de un sospechoso, resulta casi imposible compararlo con todas las muestras archivadas de los casos no esclarecidos existentes en el departamento de criminalística de la policía científica. Esto se debe a que dicha información no está presente de forma digital y el acceso a ella es lento, por lo que es muy engorroso analizar manualmente grandes cantidades de muestras. Además aumentan la posibilidad de la introducción de errores humanos y el tiempo empleado.

La situación planteada anteriormente conlleva a la presentación del siguiente ***Problema Científico:*** ¿Cómo garantizar la búsqueda y comparación de un perfil de ADN en una base de datos de casos no esclarecidos?

Para poder dar solución a dicho problema es necesario definir como ***Objeto de Estudio*** la búsqueda y comparación de perfiles de ADN, enfocando el ***Campo de Acción*** en la arquitectura para la búsqueda y comparación de perfiles de ADN.

El ***Objetivo General*** del presente trabajo de diploma es diseñar una arquitectura de software que garantice la búsqueda y comparación de un perfil de ADN en una base de datos de casos no esclarecidos.

Para dar cumplimiento a este objetivo se definieron las siguientes ***Tareas de la Investigación***:

- Determinar la estructura de la información que almacenan las principales bases de datos en sistemas de identificación por perfiles de ADN.
- Caracterizar los sistemas informáticos de identificación por perfiles de ADN.
- Investigar sobre los servicios de correos electrónicos para utilizarlos en la realización de comparaciones desde otros sitios a través de protocolos IMAP, POP y SMTP.

- Realizar una propuesta de tecnologías y metodologías a utilizar en el desarrollo de la aplicación.
- Definir los estilos arquitectónicos y patrones de diseño para el desarrollo de la solución informática.
- Describir las características y restricciones del sistema propuesto.
- Realizar el análisis y diseño de las clases relacionadas con la línea base de la arquitectura.
- Describir la propuesta de arquitectura del sistema.

En el desarrollo de la investigación, se analizaron los principales sistemas que automatizan el proceso de búsqueda y comparación de perfiles de ADN, de acuerdo a su prestigio, sus potencialidades y novedad. Para este trabajo se empleó el método **Analítico – sintético** para organizar la información de varias fuentes diferentes. El método **Inductivo – deductivo** para relacionar de lo general a lo particular en esta materia y generalizar hallazgos. El **Análisis histórico – lógico** para estudiar la evolución de los sistemas de este tipo, así como el método **Empírico** de la observación para comprender todas las características, ventajas y desventajas de los mismos. De esta misma forma, se seleccionaron las tecnologías y herramientas que se proponen para implementar el sistema.

Como resultado de este trabajo de diploma se realiza la propuesta de una arquitectura de sistema que posibilitará la implementación de una solución que automatice algunos procesos que incrementen la eficiencia del trabajo policial. Así mismo, deberá garantizar un acceso inmediato a información no disponible anteriormente. Estos aspectos incidirán en el aumento de la satisfacción de la sociedad y en un mayor apoyo a los tribunales para impartir la debida justicia.

En el desarrollo del trabajo de diploma, el contenido está estructurado de la siguiente manera:

El **Capítulo 1** aborda en detalle lo relacionado con la fundamentación teórica que sustenta el presente trabajo. Se detallan los aspectos relacionados con las diversas técnicas de obtención del perfil de ADN, caracterizando de manera general los sistemas de búsqueda y comparación de perfiles de ADN y se muestran ejemplos de sistemas de este tipo a nivel internacional con las facilidades que brindan. Se describen las principales tendencias actuales en el desarrollo de software y se exponen las principales técnicas, tecnologías y diversos estilos arquitectónicos más difundidos.

El **Capítulo 2** explica las características del sistema que se propone, se realiza un modelo conceptual del dominio, se identifican los actores y casos de uso del sistema, y se priorizan los casos de uso arquitectónicamente significativos.

El **Capítulo 3** aborda la arquitectura que se propone para el desarrollo del sistema. Se identifican los estilos y patrones a utilizar en la solución; se definen las herramientas, tecnologías y lenguaje de desarrollo; y se describen las vistas arquitectónicas del sistema.

En la sección de **Anexos** se encuentran los modelos auxiliares, tablas descriptivas de elementos del sistema y descripciones textuales de los casos de uso, entre otros artefactos.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

La arquitectura es una disciplina muy difícil en la ingeniería de software. Se requieren muchos años de experiencia y una constante actualización para realizar este trabajo con calidad. Es necesario conocer a fondo el negocio y los procesos del área que se va a informatizar. Luego se debe tener claridad de los avances existentes en diferentes regiones en este tipo de automatización y valorar la necesidad y factibilidad de implementar una solución.

Para poder definir las características y restricciones de la aplicación, se debe conocer a fondo todas las variantes de herramientas, tecnologías y tendencias existentes, que permitan elegir combinaciones adecuadas para el trabajo. Es por esto que antes de comenzar a definir y describir la arquitectura para el sistema informático, se ha realizado una exhaustiva búsqueda bibliográfica.

A continuación, en el desarrollo del capítulo, se evalúan las técnicas de obtención de perfiles de ADN. También se presentan los estándares a seguir para la creación de bases de datos que almacenen perfiles de ADN. Así como se enuncian las características y estado del arte de los sistemas que son líderes mundiales en las técnicas de identificación de personas, sobre la base de la comparación de estos perfiles. Además se dan a conocer las diversas herramientas, estilos arquitectónicos, lenguajes de programación y aplicaciones actuales que son utilizadas en el desarrollo de software.

1.1. Procesos de identificación por perfiles de ADN

El perfil de ADN fue descubierto “por accidente”, en el año 1984 por el profesor de Genética de la Universidad de Leicester[1], Sir Alec Jeffreys y sus colegas mientras rastreaban variaciones genéticas en una muestra de ADN. Así fue como se produjo la primera Huella Genética, la cual podría ser usada potencialmente para la identificación individual de cada ser humano.

No es hasta 1986 que se utilizó por primera vez en una causa penal británica. Usando esta técnica, Jeffreys comparó muestras de semen extraídas de dos personas asesinadas con la sangre del sospechoso, un joven de 17 años llamado Richard Buckland. Siendo este la primera persona

[1] La Universidad de Leicester es una universidad de investigación con sede en Leicester, Inglaterra, con aproximadamente 19.000 estudiantes matriculados -12.000 de ellos estudiantes a tiempo completo y 7.000 a tiempo parcial y de aprendizaje a distancia.

declarado de inocente, basado el veredicto en una técnica de identificación por perfil de ADN (Softcover, 2000).

1.1.1. *Técnicas de obtención de perfiles de ADN.*

Existen diversas técnicas para obtener un perfil de ADN a partir de una muestra que se posee. En la presente investigación se describen de manera general las técnicas más utilizadas a nivel mundial.

El proceso de determinación del perfil de ADN de un individuo está basado en reacciones químicas aplicadas a la muestra que contiene el ADN, en algunos casos auxiliados de dispositivos electrónicos para acelerar los procesos de reacción. Se utilizan varias técnicas o procesos para ello; entre las que se encuentran:

- *Restriction Fragment Length Polymorphism* (RFLP, por sus siglas en inglés).
- *Short Tandem Repeats (STRs*, por sus siglas en inglés*)*, combinada con *Polymerase Chain Reaction (PCR*, por sus siglas en inglés*).*

1.1.1.1. *Técnica RFLP*

La técnica RFLP puede ser resumida como: extraer, cortar, ordenar y fotografiar. La figura 1 muestra la secuencia de pasos necesarios para la aplicación de la técnica RFLP.

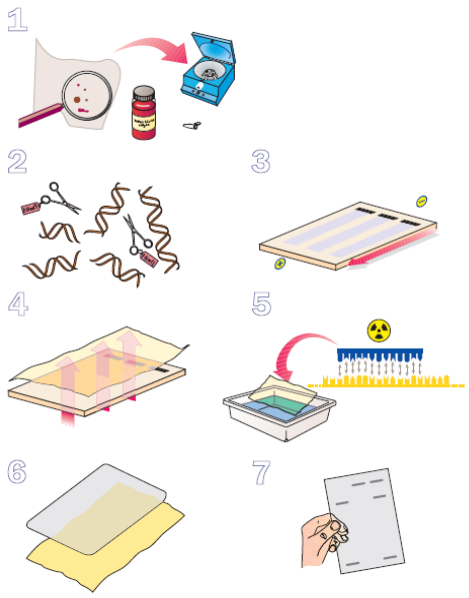


Figura 1. Pasos de la técnica RFLP.

Para este proceso se requiere una muestra de ADN relativamente grande –más de 25 muestras de cabello o una muestra de semen o sangre del tamaño de una moneda como mínimo.

Después de extraer la muestra de ADN, se mezcla con una endonucleasa de restricción –que es una enzima que corta el ADN en donde tenga una secuencia– separándose así en más de un millón de fragmentos distintos. El ADN resultante se ordena según su tamaño usando una técnica denominada *electroforesis en gel de agarrosa*.

Al finalizar este paso ya se contará con el perfil de ADN asociado a la muestra en cuestión; contenida dentro del bloque de gel donde se aplicó la técnica de electroforesis, pero todavía no es visible al ojo humano. A continuación se realizan otros procesos químicos, de los cuales se obtiene como resultado una foto denominada autoradiografía, la cual contiene el código de barras correspondiente al patrón del perfil de ADN.

La técnica RFLP es un proceso lento y engorroso. Fue la primera que se desarrolló y era lo suficientemente económica para expandir su aplicación. El análisis de las variaciones detectadas por dicho proceso fue una herramienta importante en la creación del mapa de genoma humano, en la localización de genes portadores de enfermedades genéticas, identificación de personas a partir de la huella genética en casos criminales y pruebas de paternidad.

1.1.1.2. *Técnica PCR*

La técnica PCR es ampliamente utilizada en la biología molecular. Su nombre es derivado de uno de sus componentes claves: la enzima ADN polimeraza, usada para amplificar una pieza de ADN a través de replicación enzimática *in vitro*.

Durante el progreso de la técnica PCR, el ADN que va siendo generado se usa él mismo como plantilla para la replicación. Así se establece una reacción en cadena, donde el ADN usado como plantilla se amplifica de manera exponencial. Con el uso de esta técnica es posible amplificar una única o pocas copias de una muestra de ADN generando millones de copias de la muestra en cuestión. La figura 2 muestra el proceso de replicación del ADN basándose en la técnica PCR.

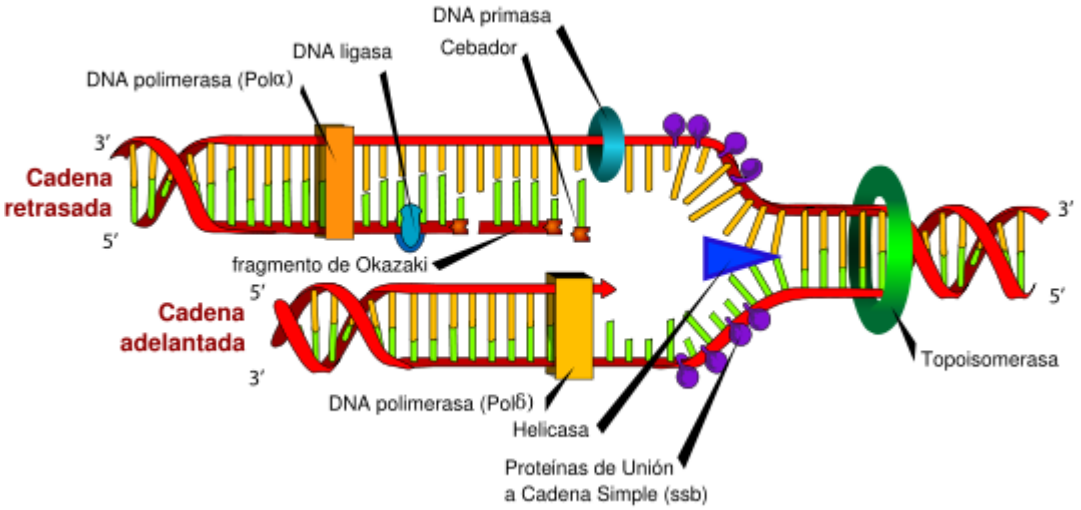


Figura 2. Replicación de ADN.

Esta técnica fue desarrollada en el año 1984 por el científico Kary Mullis[2]. En la actualidad es comúnmente utilizada en investigaciones médicas y biológicas de laboratorio; encontrándose aplicación en la clonación de ADN, análisis funcional de los genes, diagnóstico de enfermedades hereditarias, detección de enfermedades infecciosas y la obtención de perfiles de ADN –usado en las ciencias forenses y pruebas de paternidad.

1.1.1.3. *Técnica STR*

El método más actual usado hoy en día para la obtención de un perfil de ADN está basado en la técnica PCR, combinándolo con otra técnica denominada STR. Este método utiliza regiones del ADN elevadamente polimórficas que tienen secuencias cortas repetidas, las más comunes son las de 4 bases repetidas –aunque también se usan otras longitudes, incluyendo de 3 a 5 bases.

El análisis se realiza extrayendo el ADN del núcleo de las células de la muestra en cuestión, luego se amplifican las regiones polimórficas específicas del ADN extraído usando la técnica PCR. Una vez sea amplificada dicha secuencia, se le aplica a la muestra una *electroforesis en gel* o *electroforesis capilar*; lo cual le posibilita al analista determinar cuantas repeticiones de la secuencia STR existen.

En la figura 3 se muestra la gráfica –obtenida a partir de la técnica STR– con los valores de un perfil de ADN.

[2] Kary Banks Mullis, (Carolina del Norte, 28 de diciembre de 1944), bioquímico estadounidense, conocido por haber permitido, a través de la invención de la técnica de la PCR, una revolución en la investigación biológica y médica, lo que le llevó a recibir el premio Nobel de Química de 1993.

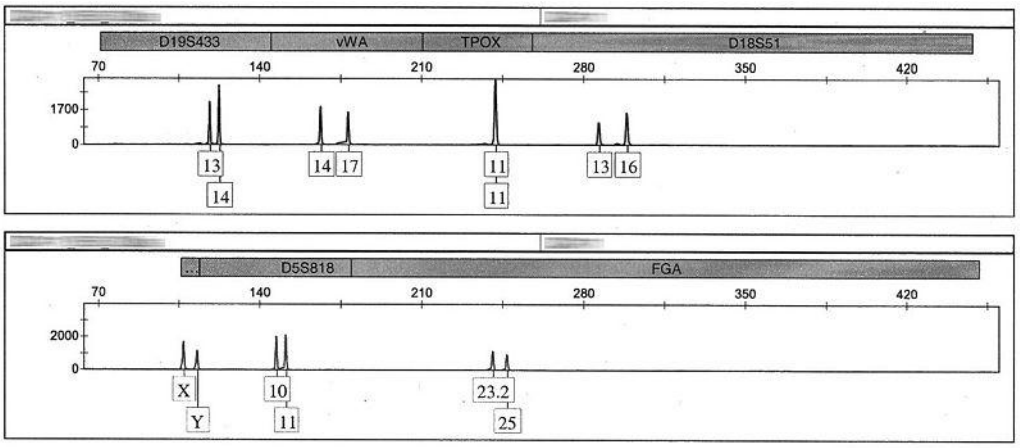


Figura 3. Perfil de ADN obtenido a partir de la técnica STR.

Los niveles de polimorfismo mostrados en cada región STR son bastante comunes entre ellas, sin embargo, la combinación única de estas regiones polimórficas enfocadas en múltiples *marcadores*, es lo que hace de este método de discriminar, una herramienta de identificación.

La técnica de análisis STR es una tecnología relativamente nueva en el campo de las ciencias forenses, se hizo popular a mediados de la década de 1990. Es usada para la obtención del perfil de ADN, ya que provee de un alto grado de datos libres de errores.

El verdadero poder del análisis basado en esta técnica se encuentra en su poder estadístico de discriminación. Con la obtención del perfil de ADN usando la técnica STR, teniendo en cuenta 13 marcadores STRs –conocidos también como regiones o locis– la probabilidad actual de que dos personas escogidas de manera aleatoria tengan la misma Huella Genética es 1 en mil millones.

A inicios de 1999, en Estados Unidos y Gran Bretaña se comenzó a adoptar esta técnica como Oficial para la obtención de los perfiles de ADN que almacenan en sus respectivas bases de datos. (Kimball, 2005)

1.1.2. *Marcadores más utilizados en la obtención de perfiles de ADN*

Al aplicarse cualquiera de las técnicas de obtención de perfiles de ADN, se obtienen un conjunto de marcadores. Estos son secuencias cortas y repetidas de ADN. El número de repeticiones en estos marcadores es altamente variable entre los individuos; haciéndolos efectivos para fines de identificación humana. Estos marcadores varían en dependencia de la técnica utilizada.

Usando la técnica *Restriction Fragment Length Polymorphism* se encuentran los marcadores RFLP (polimorfismo de la longitud de los fragmentos de restricción) y los VNTR (secuencias adyacentes que se repiten en número variable).

Aplicando la técnica PCR se encuentran los marcadores llamados RAPD (ADN polimórfico amplificado al azar), PCR iniciada con microsatélites (MP-PCR), AFLP (polimorfismo de longitud de fragmentos amplificados) y DAF (amplificación de huellas del ADN), entre otros.

Finalmente, dentro de las técnicas que combinan PCR o sus productos de ADN más la hibridación tipo Southern[3] –como la técnica STR– están los marcadores RAHM y RAMPO (amplificación aleatoria del polimorfismo de microsatélites).

Los marcadores obtenidos a partir de la técnica STR han demostrado tener varias ventajas que los hacen especialmente adecuados para la identificación de personas, entre las que se encuentran:

- Muy variable entre cada individuo.
- Tamaño pequeño del valor de los *alelos*.
- Posibilidad de amplificación en casos de muestras degradadas o mezcladas.
- Menor tasa de mutación, lo que hace que los datos sean más estables y previsibles.

Los marcadores reconocidos internacionalmente como estándares en la identificación de personas son los siguientes: CSF1PO, FGA, TH01, TPOX, VWA, D3S1358, D5S818, D7S820, D8S1179, D13S317, D16S539, D18S51 y D21S11. Estos además son los utilizados en la base de datos CODIS del FBI.

En la figura 4 se muestran los 13 marcadores utilizados en la base de datos CODIS, así como su posición cromosomal una vez obtenido el perfil de ADN.

[3] El método tipo ***Southern*** o ***Southern blot*** fue desarrollado por *E. M. Southern* para la detección de genes específicos en el **ADN** celular.

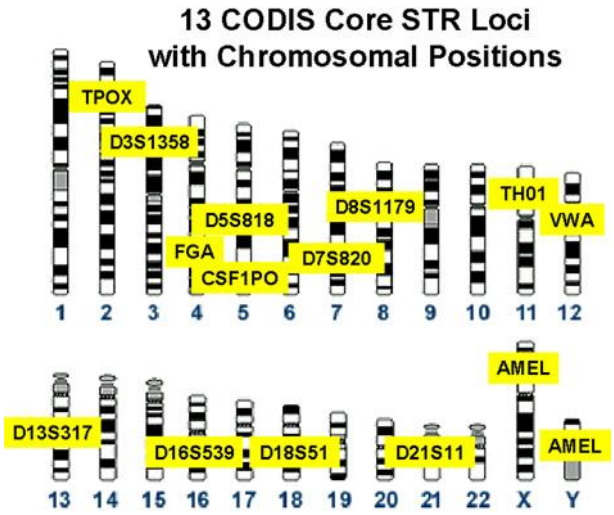


Figura 4. Muestra de los marcadores STRs usados por CODIS.

Teniendo en cuenta las características descritas anteriormente, los autores del presente trabajo de diploma proponen la utilización de los marcadores obtenidos a partir de la técnica STR, para la conformación del perfil de ADN que se almacenará en la base de datos. Esto posibilitará su posterior uso en la identificación de víctimas, victimarios, personas desconocidas y cualquier otro caso forense que requiera la identificación de un ser humano.

1.1.3. *Sistemas de búsqueda y comparación de perfiles de ADN*

Los sistemas de extracción y comparación de perfiles de ADN tienen como objetivo automatizar este proceso que consta de varios pasos fundamentales:

- Recolección de la muestra.
- Extracción del ADN.
- Amplificación en caso de que la muestra sea pequeña.
- Crear la huella genética.
- Almacenamiento de la huella genética en una base de datos.
- Comparar la huella genética de un sospechoso con los de la base de datos.

Generalmente son utilizadas grandes bases de datos y en casos como en el Reino Unido se ha propuesto almacenar el perfil genético de cada ciudadano. Se le brinda especial atención a la seguridad por la sensibilidad de la información que se maneja. (Jones, 2006)

Entidades como: *DNA Solutions* situada en California EEUU; *NEC Corporation* con sede central en Japón; *Home Office*, institución gubernamental del Reino Unido para temas de inmigración, enfrentamiento al terrorismo y policía; el Departamento de Ciencias Forenses de Alabama en EEUU y el Buró Federal de Investigaciones de ese propio país (FBI, por sus siglas en inglés) poseen tecnologías avanzadas con bases de datos de ADN y sistemas que se utilizan fundamentalmente para identificar sospechosos de crímenes a través de muestras de células obtenidas en la escena del crimen. Estos procedimientos han solucionado innumerables casos y dicha evidencia judicial es fundamental por la confiabilidad de la identificación con este método.

Desde el punto de vista médico investigativo se destaca la compañía *Applied Biosystems, Inc*. destacada en la ingeniería genética y creadora del núcleo principal de la máquina utilizada en el proyecto del Genoma Humano.

Algunas soluciones de este tipo son:

***Combined DNA Index System (CODIS)*:**

Es un sistema creado por el FBI en 1997 que almacena perfiles de ADN provenientes de laboratorios criminalísticos locales, estatales o federales de los EEUU con la habilidad para buscar en la base de datos e identificar sospechosos de crímenes.

Este sistema utiliza 13 *marcadores* STRs de ADN combinando las probabilidades de que cada una de estas regiones sea idéntica de un individuo a otro. Si dos muestras cualesquiera tienen genotipos coincidentes en los 13 marcadores del CODIS, existe una certeza virtual de que las dos muestras de ADN procedan del mismo individuo. Para que se tenga una idea, la Interpol solo utiliza 8 de estos 13 marcadores en su procedimiento estándar.

Hay muchas ventajas en el sistema STR de CODIS:

- El sistema CODIS ha sido adoptado por numerosos analistas de ADN forense.
- Los alelos STR se pueden determinar con rapidez empleando ensayos comerciales (*kits*).
- Los alelos STR son discretos –es decir, de número reducido– y se comportan según los principios establecidos de genética de poblaciones.
- Los datos son digitales y, por tanto, adecuados para las bases de datos informatizadas.
- Laboratorios de todo el mundo están contribuyendo al análisis de frecuencia de alelos STR en distintas poblaciones humanas.

- Los perfiles STR se pueden determinar a partir de cantidades muy pequeñas de ADN.

***The Alabama DNA Database System*:**

Es una colección de perfiles de ADN de personas recluidas o que hayan cometido actos violentos que es actualizada y mantenida por el Departamento de Ciencias Forenses de Alabama en los EEUU. Este sistema es compatible con CODIS.

***DS-Lociler*:**

Realiza un análisis con los mismos 13 marcadores del CODIS del FBI de forma muy competitiva y puede ser utilizado en cualquier equipo de análisis de ADN. El uso de las mismas regiones, proporciona una gran ventaja al compartir e intercambiar datos, ya que siempre se trabaja con la misma información base. La ventaja principal del *DS Lociler* viene dada por los productos específicos para el ADN, que han sido diseñados para que los fragmentos alélicos no se solapen. Así, son necesarios menos marcadores fluorescentes, reduciendo el coste y la necesidad de un mayor equipo de detección. Es un producto de *DNA Solutions*.

***Briefcase-sized DNA analysis system*:**

Sistema lanzado por *NEC Corporation* en el 2008 que permite a la policía forense realizar pruebas de ADN en tan solo 25 minutos en la misma escena del crimen. También realiza un análisis STR para crear el perfil genético de la persona.

La figura 5 muestra una imagen del sistema portátil desarrollado por *NEC Corporation.*

Figura 5. Sistema portátil lanzado por NEC.

DNA Based Identification and Tracking System:

Es un método y aparato para el seguimiento y la verificación de la identidad de personas y animales a través de una red distribuida de comunicaciones surgido en los EEUU. El servicio utiliza perfiles de

ADN para la identificación de la información específica a través de las distintas redes y bases de datos. El aparato consta de un transmisor que emite un único paquete de datos que contiene información de ADN del sujeto. El sistema utiliza las redes inalámbricas disponibles, los protocolos de internet, y bases de datos para poder localizar y ubicar al sujeto a cualquier distancia de forma instantánea.

Estos sistemas de identificación por perfiles de ADN, están estrechamente relacionados con las tecnologías de la información y comunicación (TICs).

1.2. Tendencias y tecnologías actuales

La gestión de la información mediante la utilización de hardware y software como un sistema informático, está englobada dentro de las *TICs*. Dichas tecnologías utilizan medios informáticos para la difusión, el procesamiento y almacenamiento de la información. Estos procesos tienen disímiles finalidades, entre ellas la formación pedagógica, la organización y gestión empresarial, la toma de decisiones, entre otras.

Las TICs se extienden también al campo de las ciencias forenses, sirviendo de herramienta potente para la investigación y esclarecimiento de casos judiciales.

En los sistemas informáticos actuales existen dos corrientes en el desarrollo de software, las aplicaciones de escritorio y las aplicaciones Web. Las aplicaciones de escritorio permiten hacer un gran uso de los recursos del sistema, potenciando el desarrollo de aplicaciones de una alta usabilidad al usuario a partir de las interfaces gráficas que estos poseen.

Las aplicaciones web presentan ciertos inconvenientes en cuanto a las altas prestaciones de su arquitectura física, las limitaciones que trae consigo el navegador al no ser posible realizar trabajos sin conexión a una red o de carga pesada, así como el nivel de seguridad que solicitan, por su extensibilidad y escalabilidad, por lo que no son idóneas para el desarrollo de sistemas que requieran de una gran capacidad de procesamiento y cálculo.

1.2.1. *Estilos arquitectónicos.*

El tópico más urgente y exitoso en arquitectura de software en los últimos años es, sin duda, el de los estilos y patrones. Esto permite reutilizar experiencias anteriores en la resolución de problemas bien definidos de manera que se estandariza el comportamiento a seguir.

A continuación se muestra una selección de algunos de los estilos y patrones arquitectónicos más utilizados en la actualidad.

1.2.1.1. *Estilos de Llamada y Retorno.*

1.2.1.1.1. Modelo-Vista-Controlador.

Este patrón se utiliza cuando es necesario modularizar la interfaz de usuario, las reglas de negocios y el control de eventos. En la figura 6 se muestra el esquema de sus relaciones.

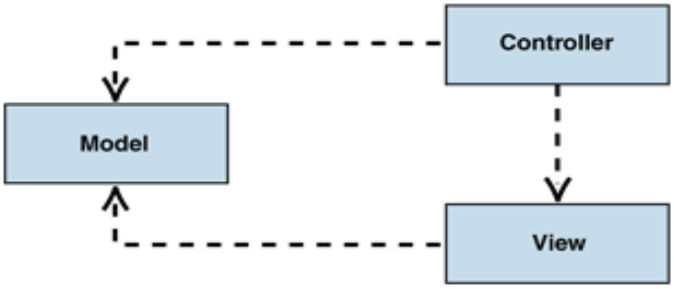


Figura 6. Modelo Vista Controlador

El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador). Mantiene el conocimiento del sistema. No depende de ninguna vista y de ningún controlador.

La vista maneja la visualización de la información mientras que el controlador interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado. (Reynoso, 2004)

1.2.1.1.2. Arquitectura en Capas

Garlan y Shaw la definen como una organización jerárquica donde cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. (Reynoso, y otros, 2004)

Los componentes de cada capa consisten en conjuntos de procedimientos; las interacciones entre las mismas, usualmente proceden por invocación de dichos procedimientos y por definición, los niveles de abajo no pueden usar funcionalidad ofrecida por los de arriba. (Reynoso, 2004)

1.2.1.1.3. Arquitectura Orientada a Objetos.

En este estilo los componentes son objetos y los conectores son procesos de invocación de procedimientos y funciones. La comunicación es implícitamente *stateful* (hablando a una instancia de

objeto creada previamente).Usualmente involucra protocolos, formatos y transportes específicos de tecnología.

Los componentes del estilo se basan en principios OO: encapsulamiento, herencia y polimorfismo. Son asimismo las unidades de modelado, diseño e implementación, y los objetos y sus interacciones son el centro de las incumbencias en el diseño de la arquitectura y en la estructura de la aplicación. Entre las cualidades de este estilo, la más básica concierne a que se puede modificar la implementación de un objeto sin afectar a sus clientes. Asimismo es posible descomponer problemas en colecciones de agentes en interacción. Además, por supuesto (y esa es la idea clave), un objeto es ante todo una entidad reutilizable en el entorno de desarrollo. (Reynoso, y otros, 2004)

1.2.1.2. *Estilos Peer-to-Peer.*

1.2.1.2.1. Arquitectura Orientada a Servicios (SOA)

IBM describe a SOA como “una arquitectura de aplicación en la cual todas las funciones se definen como servicios independientes con interfaces invocables bien definidas, que pueden ser llamadas en secuencias definidas para formar procesos de negocios”.

En este modelo existen tres actores principales: el proveedor del servicio, el registro del servicio y el solicitante del servicio. Un componente en esta arquitectura podría considerarse como un servicio que puede ser publicado, descubierto e invocado de forma dinámica. Esta variante permite la creación de sistemas altamente escalables que reflejan el negocio de la organización, a su vez brinda una forma estándar de exposición e invocación de servicios (comúnmente pero no exclusivamente servicios web), lo cual facilita la interacción entre diferentes sistemas propios o de terceros.

1.2.1.2.2. Arquitectura basada en eventos.

Las arquitecturas basadas en eventos se han llamado también de invocación implícita. Los conectores de estos sistemas incluyen procedimientos de llamada tradicionales y vínculos entre anuncios de eventos e invocación de procedimientos. La idea dominante en la invocación implícita es que, en lugar de invocar un procedimiento en forma directa (como se haría en un estilo orientado a objetos), un componente puede anunciar mediante difusión uno o más eventos. Un componente de un sistema puede anunciar su interés en un evento determinado asociando un procedimiento con la manifestación de dicho evento. Cuando el evento se anuncia, el sistema invoca todos los procedimientos que se han registrado para él. De este modo, el anuncio de un evento implícitamente ocasiona la invocación de determinados procedimientos en otros módulos.

Desde el punto de vista arquitectónico, los componentes de un estilo de invocación implícita son módulos cuyas interfaces proporcionan tanto una colección de procedimientos (igual que en el estilo de tipos de datos abstractos) como un conjunto de eventos. Los procedimientos se pueden invocar a la manera usual en modelos orientados a objeto, o mediante el sistema de suscripción que se ha descrito. (Reynoso, y otros, 2004)

1.2.1.3. *Estilos de Flujo de Datos.*

1.2.1.3.1. Tubería y Filtros

Esta popular arquitectura conecta componentes computacionales (filtros) a través de conectores (*pipes*), de modo que las computaciones se ejecutan a la manera de un flujo. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas. (Reynoso, 2004)

Tubería y filtros se utiliza cuando se tienen procesos que se pueden resolver a través de una serie de transformaciones. Además, un filtro debe ser totalmente independiente de los otros, cada uno recibe una entrada y entrega una salida.

1.2.1.4. *Estilos Centrados en Datos.*

1.2.1.4.1. Pizarra o Repositorio

Utiliza una estructura de datos para representar el estado actual y una colección de objetos independientes que operan sobre él como componentes principales. Estos sistemas se han usado en aplicaciones que requieren complejas interpretaciones de proceso de señales (reconocimiento de patrones, reconocimiento de habla, etc.), o en sistemas que involucran acceso compartido a datos con agentes débilmente acoplados. También se han implementado estilos de este tipo en procesos en lotes de base de datos y ambientes de programación organizados como colecciones de herramientas en torno a un repositorio común. (Reynoso, y otros, 2004)

En la figura 7 se muestra una imagen representativa de la arquitectura en pizarra.

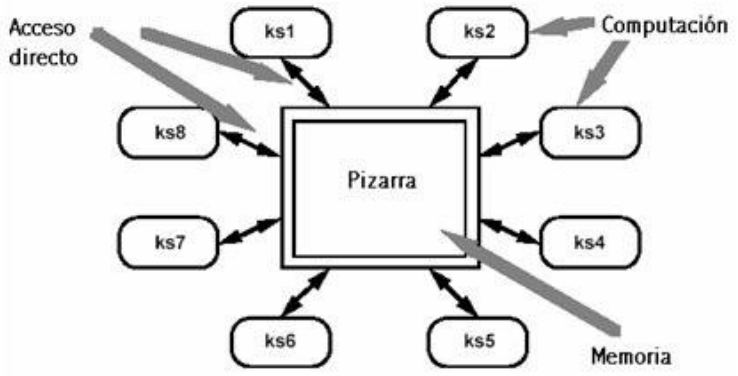


Figura 7. Arquitectura en Pizarra

1.2.1.5. *Estilos de Código Móvil.*

1.2.1.5.1. Arquitectura de Máquinas Virtuales.

La arquitectura de máquinas virtuales se ha llamado también intérpretes basados en tablas. Todo intérprete involucra una máquina virtual implementada en software. Se puede decir que un intérprete incluye un seudo-programa a interpretar y una máquina de interpretación. El seudo-programa a su vez incluye el programa mismo y el análogo que hace el intérprete de su estado de ejecución (o registro de activación). La máquina de interpretación incluye tanto la definición del intérprete como el estado actual de su ejecución. El estilo comprende básicamente dos formas o sub-estilos, que se han llamado intérpretes y sistemas basados en reglas. Ambas variedades abarcan, sin duda, un extenso espectro que va desde los llamados lenguajes de alto nivel hasta los paradigmas declarativos no secuenciales de programación. (Reynoso, y otros, 2004)

1.2.2. *Protocolos de comunicación*

Los protocolos de comunicación son reglas que permiten el flujo de información entre computadoras distintas que manejan lenguajes distintos. Para que dichas computadoras puedan conectarse entre sí, estas deben trabajar con el mismo protocolo.

1.2.2.1. *Protocolo de comunicación TCP/IP.*

El protocolo TCP/IP (Transmition Control Protocol/Internet Protocol) hace posible enlazar cualquier tipo de computadoras, sin importar el sistema operativo que usen o el fabricante. Actualmente, es posible tener una red mundial llamada Internet usando este protocolo. Este sistema de IP permite a las redes enviar correo electrónico (e-mail), transferencia de archivos (FTP) y tener una interacción con otras

computadoras (TELNET) no importando donde estén localizadas, tan solo que sean accesibles a través de Internet.

Algunos de los motivos de la popularidad de TCP/IP son:

- Independencia del fabricante.
- Soporta múltiples tecnologías.
- Puede funcionar en maquinas de cualquier tamaño.
- Estándar de EEUU desde 1983.

La arquitectura de un sistema en TCP/IP tiene una serie de metas:

- La independencia de la tecnología usada en la conexión a bajo nivel y la arquitectura del ordenador.
- Conectividad Universal a través de la red.
- Reconocimientos de extremo a extremo.
- Protocolos estandarizados.

1.2.2.2. *Protocolo de comunicación HTTP.*

El protocolo de transferencia de hipertexto (HTTP, por sus siglas en inglés) es utilizado en cada transacción de la Web. HTTP define la sintaxis y la semántica que utilizan los elementos software de la arquitectura web –clientes, servidores, proxies– para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP. Un proceso servidor escucha en un puerto de comunicaciones TCP –por defecto, el 80– y espera las solicitudes de conexión de los clientes Web. El cliente establece una conexión con el servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación) es conocido por su URL.

1.2.2.3. *Protocolo de comunicación SMTP.*

El protocolo Simple de Transferencia de Correo Electrónico (SMTP, por sus siglas en inglés) es un conjunto de reglas que rigen el formato y la transferencia de datos en un envío de correo electrónico. Es un protocolo de la capa de aplicación utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos –PDA's, teléfonos móviles, etc. SMTP se basa en el modelo cliente-servidor, donde un cliente envía un mensaje a uno o varios receptores. También dos servidores de correo que intentan transferir entre sí un mensaje utilizan SMTP para comunicarse, incluso si utilizan plataformas totalmente distintas.

1.2.2.4. *Protocolo de comunicación POP.*

El protocolo de oficina de correo (POP, por sus siglas en inglés) permite a los clientes de correo electrónico recuperar los mensajes de los servidores remotos y guardarlos en las máquinas locales. La mayoría de los clientes de correo que utilizan el protocolo POP se configuran automáticamente para eliminar el mensaje del servidor de correo después de transferirlo correctamente al sistema del cliente, aunque esto se puede cambiar. La mayoría de los usuarios que utilizan el protocolo POP sólo tienen un sistema y descargan los mensajes en sus máquinas para su almacenamiento. El protocolo POP también funciona adecuadamente si no se utiliza una conexión constante a Internet o a la red que contiene el servidor de correo. La versión más reciente de este protocolo es POP3.

1.2.2.5. *Protocolo de comunicación IMAP.*

Con el uso del protocolo de acceso a mensajes electrónicos almacenados en un servidor (IMAP, por sus siglas en inglés) se puede tener acceso al correo electrónico desde cualquier equipo que tenga una conexión a Internet. IMAP es utilizado frecuentemente en redes grandes; por ejemplo los sistemas de correo de una Universidad. IMAP permite a los usuarios acceder a los nuevos mensajes instantáneamente en sus computadoras, ya que el correo está almacenado en el servidor.

1.2.3. *Lenguajes de programación.*

En este epígrafe se especifican las características generales de los lenguajes de programación orientados a objetos comúnmente más utilizados en la actualidad (Java, C++, C#) para el desarrollo de aplicaciones de escritorio en la Universidad.

1.2.3.1. *Lenguaje de programación Java*

Java es un lenguaje de programación orientado a objetos, creado por *Sun Microsystems*. Tiene sus orígenes en el lenguaje "*Oak"* cuyo desarrollo comenzó a principios de los años 90. Su objetivo en aquel entonces era utilizarse para desarrollar sistemas incrustados en video grabadoras, tostadoras, asistentes personales de datos (*PDAs*, por las siglas en inglés*).* Sin embargo, en 1993, ante la explosión de las aplicaciones de Internet y el escaso mercado para los sistemas incrustados, se decidió enfocar el lenguaje "Oak" hacia el desarrollo de sistemas distribuidos y sustituir el nombre "Oak" por el actual. (JavaTech, 2004)

Entre sus características resaltan: poseer un mecanismo robusto de manejo eficiente de excepciones, ser fuertemente tipiado, la ausencia de los punteros –la memoria es manejada automáticamente–, ser multiplataforma, el soporte para la programación asincrónica y la utilización del enlace dinámico en tiempo de ejecución. (JavaTech, 2004)

Debido a que Java se ha desarrollado a la misma vez que Internet, es utilizado ampliamente en el desarrollo de sistemas distribuidos como aplicaciones web y además en el desarrollo de sistemas incrustados; dentro de los que sobresalen las aplicaciones para teléfonos celulares.

1.2.3.2. *Lenguaje de programación C++*

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por *Bjarne Stroustrup*. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. El nombre actual, C++, fue adoptado en el año 1983 con el objetivo de indicar que el lenguaje era un "*C mejorado*". (Stroustrup, 1998)

El lenguaje C++ soporta varios estilos de programación, por ejemplo: procedural, orientado a objetos; intenta ser tan eficiente y portable como lo es el lenguaje C. Además, permite la sobrecarga de los operadores, la inclusión de directivas del preprocesador y los tipos genéricos, entre otras funcionalidades. Soporta plenamente el polimorfismo y la herencia, destacándose la existencia de la herencia múltiple. (Stroustrup, 1998)

Este lenguaje ha sido utilizado para el desarrollo de disímiles aplicaciones, que van desde sistemas de gestión y videojuegos, hasta sistemas operativos. Entre ellos se encuentran los Sistemas Operativos Unix, Linux y Windows.

1.2.3.3. *Lenguaje de programación C#*

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. Está aprobado como un estándar por la ECMA[4] e ISO[5]. Su primera versión fue publicada en 2001; su desarrollo fue guiado por Anders Hejlsberg y está basado en los lenguajes Delphi, Visual Basic, C++ y Java, entre otros (Wikipedia, 2008).

El lenguaje C# ha sido diseñado para ser robusto, moderno y sencillo. Promueve las prácticas sanas de programación y las reglas de globalización. Además, funciona bajo un entorno de recolección automática de la memoria, lo que aumenta la productividad del desarrollador. En este lenguaje no existe el nivel de visibilidad global ni la herencia múltiple. Por otro lado, los punteros solo pueden ser utilizados en contextos no manejados y la memoria manejada no puede ser liberada explícitamente. Se destaca, además, la existencia de tipos genéricos, indexadores, delegados, eventos, clases parciales, estructuras e interfaces, entre otros rasgos. (Microsoft, 2008).

Con este lenguaje se han desarrollado sistemas de gran diversidad. Sobresalen entre ellos las aplicaciones web, las aplicaciones de escritorio y los componentes reutilizables.

1.2.4. *Bases de datos*

Una base de datos es una colección de elementos de datos interrelacionados que pueden procesarse por uno o más sistemas de aplicación (Hansen, y otros, 1997). Podría también considerarse como un objeto estructurado, compuesto por datos y metadatos. Los datos serían la información descriptiva que alberga la base de datos –por ejemplo: número del expediente criminal o valores de los marcadores. Los metadatos serían los encargados de definir la estructura que tomaría un grupo de datos en una tabla de la base de datos –por ejemplo: los campos que componen una tabla o el tipo de dato a usar para cada uno de ellos.

---

[4] **ECMA:** es una organización internacional basada en membresía de estándares para la comunicación y la información. Fue fundada en 1961 para estandarizar los sistemas computarizados en Europa. La membresía está abierta a las empresas que producen, comercializan o desarrollan sistemas computarizados o de comunicación en Europa.

[5] La **Organización Internacional para la Estandarización** (ISO, por sus siglas en inglés), fue creada el 23 de febrero de 1947). Es el organismo encargado de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación para todas las ramas industriales a excepción de la eléctrica y la electrónica. Su función principal es la de buscar la estandarización de normas de productos y seguridad para las empresas u organizaciones a nivel internacional.

Oracle, *Microsoft SQL Server*, *PostgreSQL* y *MySQL* son los gestores de bases de datos más populares en la actualidad. Cada uno de sus fabricantes comercializa su producto con sistemas para la gestión de las bases de datos y con los servicios para servidores.

1.2.5. *Sistemas gestores de bases de datos*

Un sistema gestor de bases de datos (SGBD) consiste en una aplicación capaz de examinar y manipular el contenido de una base de datos. Este software es ejecutado en un hardware servidor que provee servicios de bases de datos a computadores remotos. Un usuario de una base de datos cualquiera podrá acceder a su información con solo hacer una consulta a través de una petición al servidor, quien devolverá los datos solicitados. Sobre el servidor se pueden ejecutar también comandos que modifiquen alguna base de datos existente.

1.2.5.1. *Oracle*

Oracle es un sistema de gestión de base de datos relacional (RSGBD), elaborado por la compañía *Oracle Corporation.* Aparece en el mercado su primera versión en el año 1979. En la actualidad el sistema se encuentra en la versión 11g, liberada en el año 2007 bajo una licencia privativa.

El sistema es uno de los más completos en el mercado y se destaca por características como el soporte de transacciones, su estabilidad, su escalabilidad y la habilidad de ser multiplataforma. Oracle soporta diferentes lenguajes para el acceso y la manipulación de los datos, entre ellos, el lenguaje estructurado de consulta (SQL) avalado por el estándar ANSI, el lenguaje procedural y extensión de SQL (PL/SQL) perteneciente a Oracle y Java a partir de la versión 8i en la cual se agregó a la base de datos Oracle, la máquina virtual de Java. Las mayores críticas de la plataforma se centran en su seguridad, así como en las políticas de suministro de parches de seguridad en tiempo.

1.2.5.2. *Microsoft SQL Server*

Microsoft SQL Server es un RSGBD desarrollado por la empresa *Microsoft Corporation*. El sistema, con licencia Microsoft EULA, surge a mediados de los años 90 y su última versión es la correspondiente al año 2008.

El sistema está basado en el lenguaje Transact-SQL, aunque permite la escritura de procedimientos almacenados y *triggers* a través del uso de cualquier lenguaje de la plataforma .NET. Es capaz de sostener grandes niveles de concurrencia, soporta transacciones, es escalable y seguro. El RSGBD de Microsoft está orientado a diversos ambientes, para grandes, medianos o pequeños proyectos. Las mayores críticas radican en que no es portable a otros entornos fuera de la plataforma de Microsoft.

1.2.5.3. *MySQL*

MySQL es uno de los RSGBD perteneciente a la comunidad de software libre. Contrario a otros desarrollos de dicha comunidad, MySQL pertenece a una empresa privada llamada *Sun Microsystems*, la cual libera sus productos bajo dos licencias, una GNU GPL y otra para uso comercial. Actualmente se encuentra en su quinta versión, correspondiente al año 2008.

La arquitectura del gestor permite al usuario seleccionar el motor de búsqueda con el cual prefiere trabajar. MyISAM es uno de los motores más usados, debido a que su método no transaccional dota al sistema de una gran rapidez en la lectura. Estudios han revelado que el uso de MyISAM puede influir negativamente en la integridad de la base de datos en entornos de alta concurrencia en la modificación. MySQL, además de ser multiplataforma, soporta los procedimientos almacenados y triggers.

1.2.5.4. *PostgreSQL*

El RSGBD Orientado a Objetos PostgreSQL, comenzó su desarrollo sobre los años 80. En la actualidad cuenta con su versión 8.3.5 liberada en noviembre de 2008. El gestor se encuentra bajo una *licencia BSD* y es desarrollado por una comunidad mundial conocida como Grupo Mundial de Desarrollo de PostgreSQL (PGDG, por sus siglas en inglés).

Entre sus principales características se encuentra su destreza para manejar altos niveles de concurrencia mediante un sistema llamado Acceso Concurrente Multiversión (MVCC, por sus siglas en inglés). Posee una amplia variedad de tipos nativos y permite al usuario crear sus propios tipos de datos. Es multiplataforma y además soporta procedimientos almacenados y *triggers*, los cuales pueden ser escritos usando lenguajes como PL/PgSQL, C++, Java PL/Java web, PL/Python y plPHP.

Como resultado de la investigación, en este capítulo se identificó la presencia en el mundo de bases de datos y sistemas para la identificación a través del ADN. Estos, son desarrollados en países del primer mundo, tienen un carácter gubernamental y no poseen un fin comercial. Algunos de ellos brindan servicios externos bien definidos. Aunque hay cierta variedad en la utilización de marcadores en la conformación del perfil de ADN, se ha tratado de generalizar el empleo de los 13 locis del CODIS para facilitar el trabajo colaborativo desde diferentes regiones.

Además, la información recopilada sobre estilos arquitectónicos, patrones de diseño, protocolos de comunicación, lenguajes de programación y sistemas gestores de bases de datos, constituye la base para la toma de decisiones respecto a la manera de estructurar la solución.

Estos aspectos se tendrán en cuenta para la propuesta realizada en los próximos capítulos.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

Para comenzar a diseñar un sistema informático es necesario aclarar todos los conceptos y sus relaciones utilizando un lenguaje que sea comprendido por todos los involucrados. Se identifican los requisitos de la aplicación realizando diferentes actividades, los cuales son el apoyo para las estimaciones y la planificación de todo el proceso de desarrollo. Se define la visión general del proyecto y su alcance en iteraciones.

A continuación se hace una descripción de los procesos esenciales del negocio y las principales características del sistema propuesto; teniendo en cuenta la relación existente entre los conceptos presentes en el negocio. Se especifican los requerimientos funcionales y no funcionales que regirán el desarrollo del sistema y la correspondiente trazabilidad de estos con los casos de uso que poseerá el sistema.

2.1. Descripción de los procesos del negocio

Debido a que el presente trabajo de diploma se centra en la propuesta de arquitectura de un sistema que actualmente no tiene un cliente específico, los procesos del negocio no están debidamente definidos. Es por esto que los procesos descritos a continuación tienen una base genérica en las investigaciones previas desarrolladas por los autores.

El **proceso de identificación por perfiles de ADN** consta de varios pasos. Cuando la muestra de ADN es recibida en el laboratorio por el Especialista, esta pasa a ser procesada a través de un tratamiento químico con el objetivo de obtener el perfil de ADN correspondiente. Luego se procede a la comparación del perfil de ADN obtenido contra el archivo de perfiles existentes en el laboratorio.

El proceso que se pretende automatizar no cubre todos los pasos del proceso de identificación, sino está centrado en la gestión de la información de los casos criminalísticos y los perfiles de ADN asociados.

2.2. Propuesta de Sistema

Teniendo en cuenta los resultados arrojados de investigaciones desarrolladas sobre los diversos sistemas de identificación biométrica –específicamente los de identificación por perfiles de ADN– se determinaron las principales funcionalidades y características que poseerá el sistema.

El sistema de identificación por perfiles de ADN tendrá una arquitectura basada en componentes independientes –cada uno con responsabilidades debidamente definidas– que se relacionarán entre sí. El mismo será capaz de gestionar toda la información referente a un caso criminalístico y perfiles de ADN asociados y soportará dos formas de entrada de datos:

- Una interfaz de comunicación capaz de interpretar los datos numéricos resultantes del análisis de una muestra de ADN, arrojado por el dispositivo electrónico que se encarga de procesar dicha muestra.
- Una interfaz visual con los campos comprendidos en la definición del perfil de ADN por parte del administrador del sistema.

La conformación de los marcadores correspondientes al perfil de ADN son totalmente configurables; así como el umbral de similitud que se utiliza en la comparación de los marcadores del perfil.

La comunicación entre todos los componentes del sistema será cifrada y con confirmación de entrega, contribuyendo esto a la integridad de los datos. Los componentes involucrados en la comparación de perfiles de ADN poseerán un registro de bitácora de todas las operaciones de comunicación que realicen; teniéndose siempre el estado de las solicitudes enviadas y recibidas.

El sistema será capaz de realizar comparaciones entre perfiles de ADN de los tipos: uno a uno, y de uno a muchos local y remoto. La lógica de comparación de perfiles de ADN será transparente a la arquitectura del sistema. Esto constituye una ventaja que posibilita que la arquitectura del sistema que se propone, sea utilizada en el desarrollo de otros sistemas de identificación biométricos, modificando o sustituyendo solamente el módulo de comparación.

2.2.1. *Metodología de desarrollo.*

En un proyecto de desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo. Una metodología es un proceso. No existe una metodología de software universal. Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable.

El sistema se desarrollará utilizando el Proceso Unificado de Desarrollo (RUP, por sus siglas en inglés) como metodología. RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del Lenguaje Unificado de Modelado (UML, por sus siglas en inglés), y trabajo de muchas metodologías utilizadas por los clientes.

El ciclo de vida de RUP se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y ser iterativo e incremental. Estos puntos garantizan que se satisfagan las necesidades del cliente pues se abordan las cuestiones de alto riesgo en las primeras iteraciones. Además, maneja eficientemente las solicitudes de cambio y la gestión de la configuración.

2.3. Modelo de Dominio

Como quedó expuesto en el epígrafe 2.1 los procesos de negocio no están bien estructurados; por lo que se determina realizar un Modelo de Dominio donde se presenta un marco conceptual y la relación existente entre las definiciones.

2.3.1. *Conceptos fundamentales*

Para lograr una mayor comprensión del diagrama de Modelo de Dominio que se expondrá a continuación, es necesario realizar previamente la definición de los conceptos que están involucrados en dicho Modelo.

LPC: Es el Laboratorio de la Policía Científica donde se realiza la gestión de los perfiles de ADN encontrados en las distintas escenas del crimen.

Trabajador: Es una entidad virtual que representa la generalización de los roles que puede cumplir algún actor en una situación determinada.

Especialista: Trabajador encargado de realizar la obtención y comparación entre perfiles de ADN. Es el responsable de proveer el resultado.

Supervisor: Trabajador encargado de controlar el acceso al archivo de perfiles de ADN.

Muestra de ADN: Representa la abstracción de una muestra de ADN que es introducida al Laboratorio para su procesamiento.

Equipo de Adquisición: Dispositivo electrónico encargado de realizar y controlar los procesos químicos necesarios para la obtención de los datos del perfil de ADN.

Perfil de ADN: Conjunto de información obtenida a partir del procesamiento de una muestra de ADN.

2.3.2. *Diagrama del Modelo de Dominio*

El Modelo de Dominio o Modelo Conceptual, permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del sistema en desarrollo. Un Modelo del Dominio es una representación de las clases conceptuales del mundo real, no de componentes software.

La figura 8 muestra el diagrama del Modelo de Dominio referente a los procesos de identificación por perfiles de ADN.

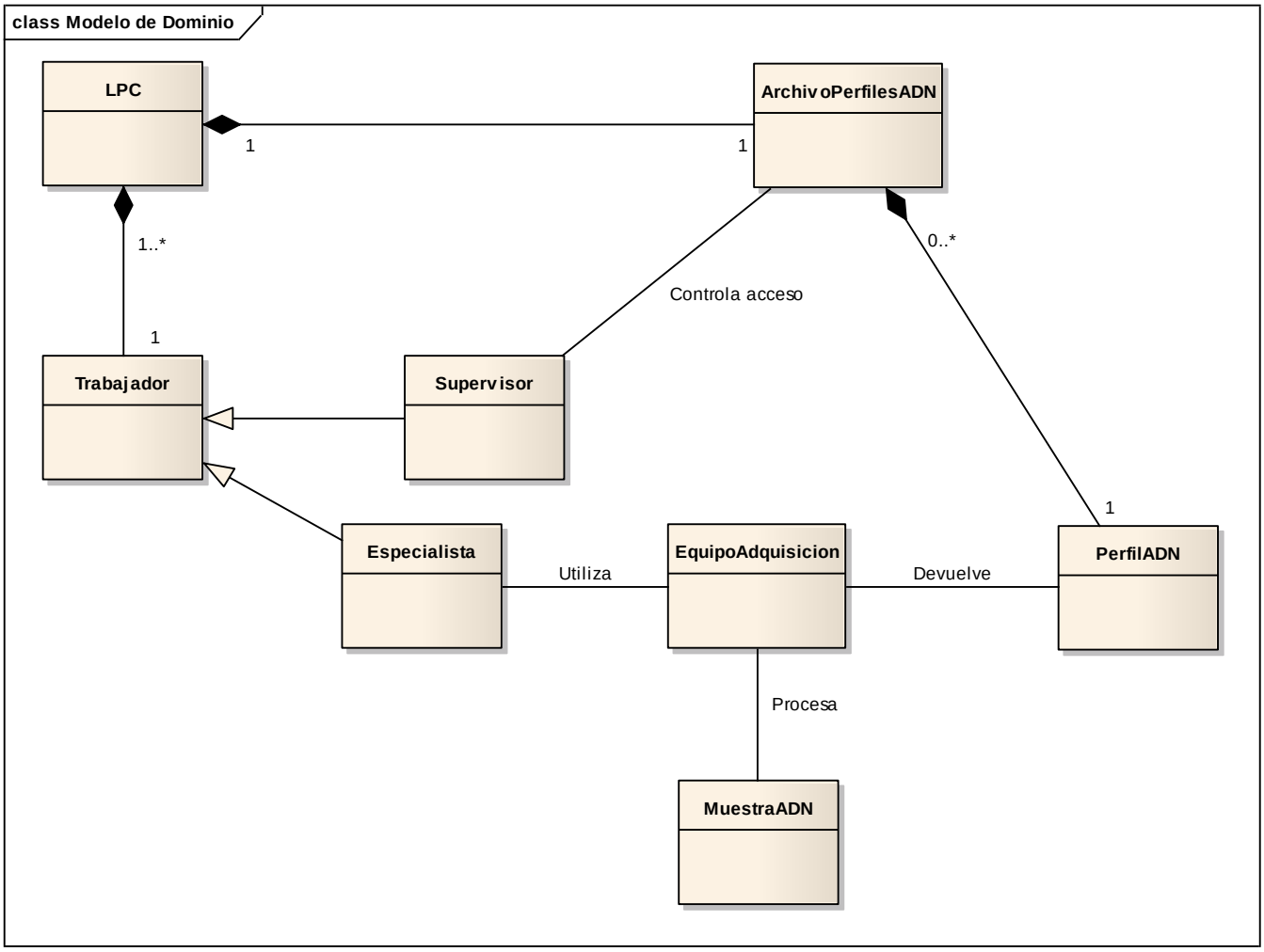


Figura 8. Modelo de Dominio.

2.4. Especificación de los requisitos de software

Un requisito de software es una condición que debe cumplir un sistema para satisfacer un contrato, estándar, especificación, u otra documentación formalmente impuesta. Estos se clasifican en dos grupos fundamentalmente: los requisitos funcionales –comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica– y los requisitos no funcionales –propiedades o cualidades que el producto debe tener.

2.4.1. *Requerimientos Funcionales*

RF1. Gestionar caso criminalístico.

- RF1.1. Almacenar información referente a un caso criminalístico.
- RF1.2. Recuperar información referente a un caso criminalístico.
- RF1.3. Eliminar un caso.
- RF1.4. Modificar datos de un caso.

RF2. Comparar perfiles de ADN.

- RF2.1. Investigar si un perfil de ADN es correspondiente con otro existente.
- RF2.2. Recuperar todos los casos cuyos perfiles de ADN coinciden con una solicitud que se realiza.
- RF2.3. Realizar las acciones anteriores utilizando bases de datos ubicadas en cualquier localización geográfica.

RF3. Sincronizar bases de datos locales con una central.

RF4. Gestionar usuario.

- RF4.1. Insertar un nuevo usuario.
- RF4.2. Eliminar un usuario existente.
- RF4.3. Modificar datos de un usuario.

RF5. Generar reportes estadísticos del uso del sistema.

RF6. Generar un informe preestablecido con la información que será utilizada como prueba.

RF7. Configurar sistema.

RF7.1. Configurar marcadores del sistema.

RF7.2. Configurar permisos de transmisión entre módulos.

2.4.2. *Requerimientos No Funcionales*

- Apariencia o interfaz externa.
  - Se debe tener en cuenta la experiencia de usuario en cuanto al diseño del menú, accesos directos por teclado, etc.
  - Las ventanas contendrán los datos de forma clara y bien estructurada.
  - El diseño de la interfaz de usuario del sistema buscará la ejecución de acciones de una manera rápida, minimizando los pasos a dar en cada proceso.
  - Los campos obligatorios de cada ventana deben estar debidamente identificados.
- Rendimiento.
  - El sistema debe soportar el manejo de un volumen grande de información en la memoria RAM.
  - El proceso de comparación entre perfiles de ADN debe realizarse en el menor tiempo posible.
- Seguridad.
  - Poseer seguridad de acceso y administración de usuarios: creación de usuarios; otorgamiento de privilegios y roles.
  - Poseer activación de permisos de comunicación entre las estaciones cliente – servidor por direcciones IP.
  - Los campos deben ser validados en el mismo momento de introducción de datos por parte de los usuarios y en el negocio del sistema.
  - La información intercambiada entre los componentes del sistema deberá estar cifrada.
  - El intercambio de cualquier tipo de información involucrará una confirmación de entrega y recepción de la misma del módulo receptor al emisor.

- Se registrarán las trazas de las operaciones realizadas por los usuarios del sistema.

- Políticos-culturales.
  - Los textos de los mensajes e interfaces visuales del sistema estarán escritos en idioma español internacional.

- Software.
  - Servidor central de procesamiento:
    - Sistema Operativo Unix.
    - *Framework* JavaMail y JavaBeans Activation Framework (JAF).
    - Java Virtual Machine.
  - PC Cliente:
    - Sistema Operativo Debian (etch).
    - *Framework* JavaMail y JavaBeans Activation Framework (JAF).
    - Java Virtual Machine.
  - Servidor central de bases de datos
    - Sistema Operativo Debian (etch).
    - Gestor de Base de Datos PostgreSQL 8.3.
  - Servidor regional de bases de datos
    - Sistema Operativo Debian (etch).
    - Gestor de Base de Datos PostgreSQL 8.3.
  - Servidor de intercambio de mensajería sede regional
    - Sistema Operativo Debian (etch).
    - Servidor de correo SendMail.
  - Servidor de intercambio de mensajería sede central
    - Sistema Operativo Debian (etch).
    - Servidor de correo SendMail.

- Hardware.

Todas las computadoras deberán tener una tarjeta de red con una velocidad de transmisión de 100Mbps. Los servidores de intercambio de mensajería deberán tener dos tarjetas de red, una para comunicarse con la subred local y la otra para la conexión a internet.

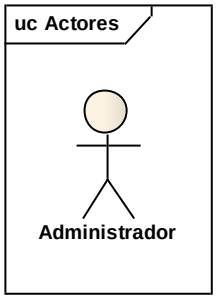
  - Servidor central de procesamiento:
    - Memoria RAM 4 GB DDR 2 (óptimo 8 GB).
    - Procesador *Intel® Itanium® 9000 sequence.*
    - 100 GB de capacidad de almacenamiento mínimo (óptimo 200 GB).
  - PC Cliente:
    - Memoria RAM 1 GB DDR 2 (óptimo 2 GB).
    - Procesador *Intel® Core 2 Duo®.*
    - 100 GB de capacidad de almacenamiento mínimo (óptimo 200 GB).
  - Servidor central de bases de datos:
    - Memoria RAM 4 GB DDR 2 (óptimo 8 GB).
    - 500 GB de capacidad de almacenamiento mínimo (óptimo 1 TB).
    - Procesador *Intel® Xeon®* serie 5000 (o superior).
  - Servidor regional de bases de datos
    - Memoria RAM 2 GB DDR 2 (óptimo 4 GB).
    - 200 GB de capacidad de almacenamiento mínimo (óptimo 400 GB).
    - Procesador *Intel® Xeon®* serie 5000 (o superior).
  - Servidor de intercambio de mensajería sede regional
    - Memoria RAM 1 GB DDR 2 (óptimo 2 GB).
    - Procesador *Intel® Core 2 Duo®.*
    - 100 GB de capacidad de almacenamiento mínimo (óptimo 200 GB).
  - Servidor de intercambio de mensajería sede central
    - Memoria RAM 1 GB DDR 2 (óptimo 2 GB).

- Procesador *Intel® Core 2 Duo®.*
- 100 GB de capacidad de almacenamiento mínimo (óptimo 200 GB).

2.5. Definición de los casos de uso

La arquitectura del sistema está condicionada por los casos de uso –los cuales representan la concreción de los requisitos funcionales a la hora de implementar una solución software. Estos proporcionan un medio para que los desarrolladores, usuarios finales y trabajadores lleguen a una comprensión común del sistema propuesto. Son empleados además para la validación de la arquitectura a lo largo del desarrollo.

2.5.1. *Definición de los Actores*

Actores del sistema	Justificación
	Es el administrador del sistema. Encargado de la configuración y gestión de los usuarios del sistema.
	Usuario del sistema encargado de la inserción y solicitud de comparación de los perfiles asociados a los casos en el sistema. Puede además generar reportes estadísticos e informes.
	Usuario del sistema encargado de la sincronización entre las bases de datos –central y local– y de la modificación y eliminación de los casos.

2.5.2. *Listado de los casos de uso*

CU-1	Insertar Caso
Actor	Especialista
Descripción	El Especialista introduce en el sistema los datos relacionados al caso criminalístico. Los datos del perfil de ADN asociado pueden ser introducidos de manera manual o por la interfaz de comunicación con el equipo que los genera. Los datos son almacenados en la base de datos local.
Referencia	RF1, RF1.1

CU-2	Comparar Caso
Actor	Especialista
Descripción	El Especialista selecciona el tipo de comparación que desea realizar. Introduce los datos a comparar y el sistema devuelve una respuesta acorde a dicha comparación.
Referencia	RF1.2, RF2, RF2.1, RF2.2, RF2.3

CU-3	Generar Informes
Actor	Especialista
Descripción	El especialista solicita la generación del informe oficial de la comparación realizada. El sistema genera el informe listo para imprimir.
Referencia	RF6

CU-4	Generar Reportes
Actor	Especialista
Descripción	El especialista solicita la generación de un reporte estadístico detallado con las operaciones realizadas en el sistema. El sistema genera el reporte listo para imprimir.

Referencia	RF5

CU-5	Modificar Caso
Actor	Comisario
Descripción	El comisario realiza una búsqueda de un caso y modifica sus datos. El sistema almacena los datos modificados.
Referencia	RF1, RF1.1, RF1.2, RF1.4

CU-6	Eliminar Caso
Actor	Comisario
Descripción	El comisario realiza una búsqueda de un caso y elimina sus datos. El sistema procede a la eliminación de dicha información de la base de datos.
Referencia	RF1, RF1.2, RF1.3

CU-7	Sincronizar bases de datos
Actor	Comisario
Descripción	El comisario realiza la sincronización de los datos locales con los almacenados en el servidor central. El sistema se conecta con la base de datos remota y realiza la sincronización. Devuelve un mensaje con el estado de la operación solicitada.
Referencia	RF3

CU-8	Configurar Sistema
Actor	Administrador
Descripción	El administrador del sistema configura los permisos de comunicación entre los distintos módulos del sistema y los marcadores que se usarán en la conformación de los perfiles de ADN.

Referencia	RF7, RF7.1, RF7.2

CU-9	Gestionar Usuarios
Actor	Administrador
Descripción	El administrador del sistema gestiona la información referente a los usuarios del sistema, puede crear un nuevo usuario, modificar los datos de uno existente o eliminarlos.
Referencia	RF4, RF4.1, RF4.2, RF4.3

CU-10	Buscar Caso
Actor	Especialista
Descripción	El especialista provee al sistema el identificador de un caso y el sistema devuelve la información referente a dicho caso.
Referencia	RF1, RF1.2

2.5.3. *Diagrama de casos de uso*

A continuación la figura 9 muestra el diagrama de casos de uso del sistema propuesto.

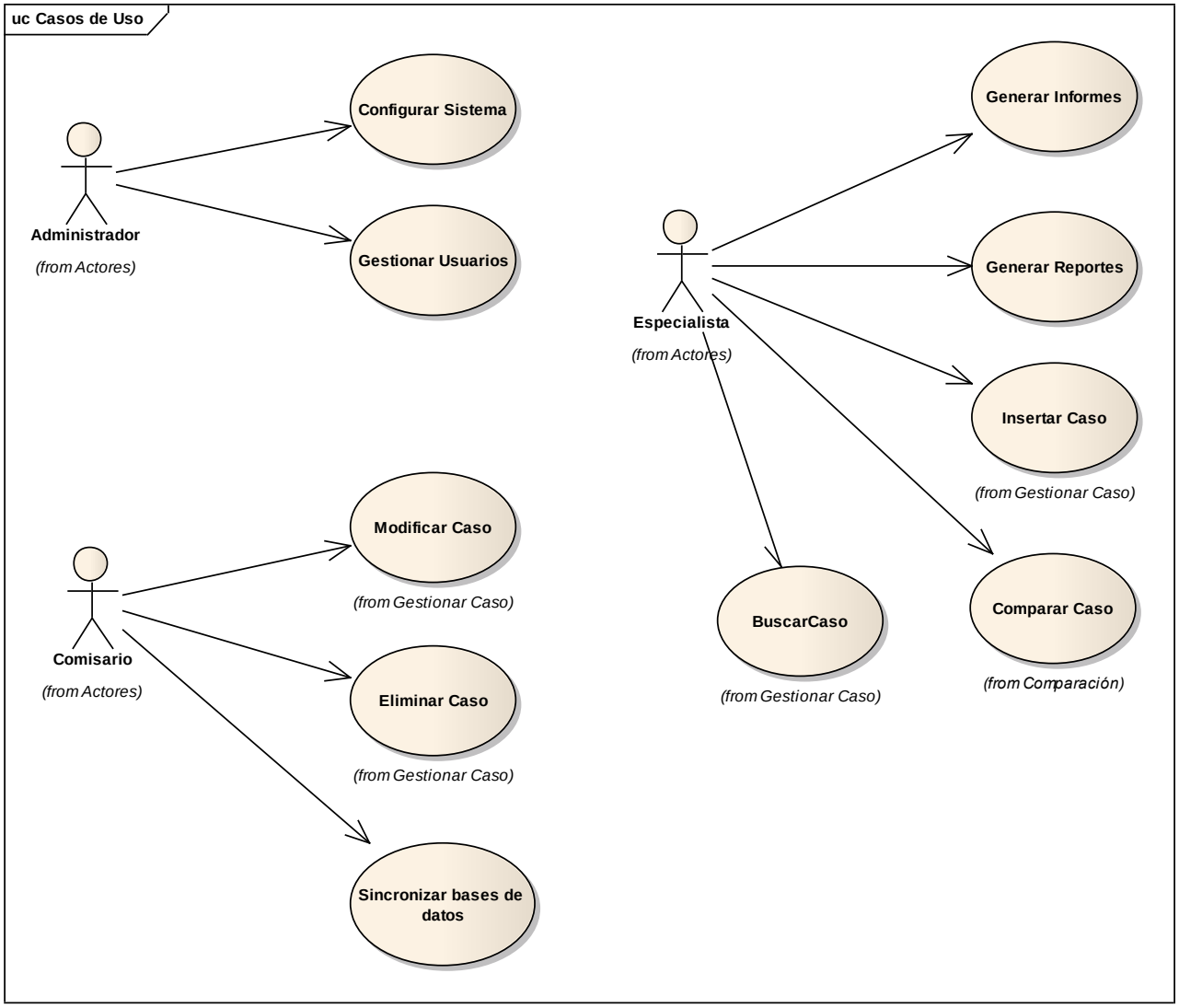


Figura 9. Diagrama de Casos de Uso del Sistema.

En este capítulo comenzó la traducción de lenguaje natural a lenguaje técnico para la implementación de la solución. Se determinaron los requisitos funcionales que poseerá el sistema. Estos marcaron una especie de contrato que regirá todo el proceso de desarrollo. Con la valoración de una serie de aspectos, se establecieron varias restricciones importantes a través de los requisitos no funcionales.

También se obtuvo el diagrama de casos de uso del sistema, el cual se corresponde con uno de los hitos fundamentales de la fase de inicio, lo que permitió evidenciar dónde se deben concentrar los esfuerzos del trabajo. Estos casos de uso, son los responsables de guiar la arquitectura, que será descrita en el próximo capítulo.

CAPITULO 3. PROPUESTA DE ARQUITECTURA

La arquitectura de un software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación ente ellos. Describe los aspectos estáticos y dinámicos de un sistema, dando lugar a un diseño de implementación y a un diagrama de flujo. Toda arquitectura debe ser implementable en nodos físicos, se determina qué computadora tendrá asignada cada tarea. Establece los fundamentos para que analistas, diseñadores y programadores trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.

No siempre es necesario definir una nueva arquitectura. En muchos casos es común seleccionar alguna ya establecida o probada en otros sistemas de información, la cual se ajuste a las necesidades del nuevo software.

Entre las principales tareas del arquitecto de software están: definir explícitamente las interfaces de comunicación entre componentes y subsistemas, encontrar elementos de reutilización para ser utilizados por los desarrolladores abaratando el costo, priorizar las funcionalidades del software que son significativas para la arquitectura y facilitar una estructura capaz de evolucionar que se adapte a los cambios futuros.

A continuación se brinda una comprensión arquitectónica global del sistema que se propone. Se utilizará el modelo 4 + 1 Vistas, que muestra las diferentes características del sistema. Además, se busca capacitar a los desarrolladores, directivos, clientes y otros usuarios en la compresión al detalle del sistema propuesto, para facilitar su participación en el mismo.

3.1. Estilos y patrones arquitectónicos propuestos

3.1.1. *Arquitectura en Capas*

Este patrón simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores. Brinda modularidad a las aplicaciones, ayuda a identificar qué puede reutilizarse y proporciona una estructura que posibilita tomar decisiones sobre qué partes comprar y qué partes construir.

Se propone una arquitectura en tres capas lógicas distintas, cada una de ellas con un grupo de interfaces perfectamente definido. La primera capa se denomina capa de presentación y consiste en las interfaces gráficas de usuario. La capa intermedia, o capa de negocio, consiste en la aplicación o lógica de negocio que encapsula las principales funcionalidades, y la tercera capa, la capa de acceso a datos, contiene las operaciones necesarias para el acceso y la manipulación de la información que se almacena. Este modelo brinda la ventaja de que el desarrollo se pueda llevar a cabo en varios niveles de manera simultánea y, en caso de que se detecte algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado.

3.1.2. *Arquitectura Orientada a Servicios (SOA)*

La característica más importante de este modelo es el bajo grado de acoplamiento entre componentes junto a una mayor flexibilidad ante cambios futuros. SOA ha surgido como la mejor manera de afrontar el desafío de hacer más con menos recursos. Promete hacer la reutilización y la integración mucho más fáciles, ayudando a reducir el tiempo de desarrollo y aumentando la agilidad organizacional. SOA es el estilo de arquitectura más importante del momento, en desarrollo simultáneo en la academia y la industria.

La comparación entre perfiles de ADN se propone a través de un servicio que utilizará mensajes de correo electrónico con un formato definido, estandarizando la manera de consumir el servicio y la respuesta que se obtiene del mismo. Dicho servicio estará disponible y documentado para cualquier necesidad posterior del cliente.

3.2. Patrones de diseño

Un sistema diseñado con tecnología orientada a objetos está compuesto de objetos que envían mensajes a otros objetos para que se lleven a cabo determinadas operaciones. La calidad del diseño de dichas interacciones, así como la responsabilidad que cada quien debe cumplir, es un tema importante a tener en cuenta durante la definición arquitectónica del sistema.

Las decisiones poco acertadas, realizadas en esta etapa, dan origen a sistemas y componentes frágiles y difíciles de mantener, entender, reutilizar o extender. Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, por lo que su utilización es la clave para un diseño exitoso.

A continuación se mostrarán las principales características de los patrones de asignación de responsabilidades que se utilizan en el diseño de este sistema.

3.2.1. *Experto.*

Problema: ¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos?

Solución: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Explicación: Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Da origen a diseños donde el objeto de software realiza las operaciones que normalmente se aplican a la cosa real que representa, por lo que ofrece una analogía con el mundo real.

Beneficios: Con la utilización de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y más cohesivas que son más fáciles de comprender y mantener.

3.2.2. *Creador.*

Problema: ¿Quién debería ser responsable de crear una nueva instancia de alguna clase?

Solución: Asignarle a la clase B la responsabilidad de crear una instancia de la clase A en uno de los siguientes casos:

B agrega los objetos A.

B contiene los objetos A.

B registra las instancias de los objetos A.

B utiliza específicamente los objetos A.

B tiene los datos de inicialización que serán trasmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A).

B es un creador de los objetos A.

Si existe más de una opción, prefiera la clase B que agregue o contenga la clase A.

Explicación: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento.

Beneficios: Brinda un soporte a un *bajo acoplamiento* –patrón que será descrito más adelante– lo que supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización.

3.2.3. *Bajo Acoplamiento.*

Problema: ¿Cómo dar soporte a una dependencia escasa y aun aumento de la reutilización?

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras clases.

Una clase con alto (o fuerte) acoplamiento recurre a muchas otras. Este tipo de clases no es conveniente, ya que presentan los siguientes problemas:

- Los cambios de las clases afines ocasionan cambios locales.
- Son más difíciles de entender cuando están aisladas.
- Son más difíciles de reutilizar porque se requiere la presencia de otras clases de las que dependen.

Solución: Asignar una responsabilidad para mantener bajo acoplamiento.

Explicación: El bajo acoplamiento es un principio que se debe tener siempre en cuenta durante las decisiones de diseño. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Este patrón estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecienten la oportunidad de una mayor productividad.

No puede considerarse en forma independiente de otros patrones como Experto o Alta cohesión, sino que más bien ha de incluirse como uno de los principios del diseño que influyen en la decisión de asignar responsabilidades.

Beneficios: Con el uso de este patrón los componentes no se afectan por cambios de otros componentes, son fáciles de entender por separado y fáciles de reutilizar.

3.2.4. *Alta Cohesión.*

Problema: ¿Cómo mantener la complejidad dentro de límites manejables?

En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. No conviene este tipo de clases pues presentan los siguientes problemas:

- Son difíciles de comprender.
- Son difíciles de reutilizar.
- Son difíciles de conservar.
- Son delicadas: las afectan constantemente los cambios.

Las clases con baja cohesión a menudo representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otros objetos.

Solución: Asignar una responsabilidad de modo que la cohesión siga siendo alta.

Explicación: El patrón Alta Cohesión es la meta principal que ha de tenerse en cuenta en cada momento en todas las decisiones de diseño. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño. Una clase de alta cohesión posee un número relativamente pequeño, con una importante funcionalidad relacionada y poco trabajo que hacer. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande.

Beneficios: Con el uso de este patrón mejoran la claridad y la facilidad con que se entiende el diseño. Se simplifican el mantenimiento y las mejoras en funcionalidad. A menudo se genera un bajo acoplamiento. La ventaja de una gran funcionalidad soporta una mayor capacidad de reutilización, porque una clase muy cohesiva puede destinarse a un propósito muy específico.

3.2.5. *Controlador.*

Problema: ¿Quién debería encargarse de atender un evento del sistema?

Un evento del sistema es un evento de alto nivel generado por un actor externo; es un evento de entrada externa. Se asocia a operaciones del sistema: las que emite en respuesta a los eventos del sistema.

Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación.

Solución: Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones:

- El "sistema" global (controlador de fachada).
- La empresa u organización global (controlador de fachada).
- Algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (controlador de tareas).
- Un manejador artificial de todos los eventos del sistema de un caso de uso (controlador de casos de uso).

Utilice la misma clase de controlador con todos los eventos del sistema en el mismo caso de uso.

Explicación: La mayor parte de los sistemas reciben eventos de entrada externa, los cuales generalmente incluyen una interfaz gráfica para el usuario operado por una persona. Otros medios de entrada son los mensajes externos o las señales procedentes de sensores como sucede en los sistemas de control de procesos.

En todos los casos, si se recurre a un diseño orientado a objetos, hay que elegir los controladores que manejen esos eventos de entrada. Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan.

La misma clase controlador debería utilizarse con todos los eventos sistémicos de un caso de uso, de modo que se pueda conservar la información referente al estado del caso.

Beneficios: Garantiza que la empresa o los procesos de dominio sean manejados por la capa de los objetos del dominio y no por la de la interfaz. Al delegar a un controlador la responsabilidad de la operación de un sistema entre las clases del dominio favorece la reutilización de la lógica para manejar los procesos afines del negocio en aplicaciones futuras.

3.3. Nomenclatura de identificadores

Para estandarizar la codificación de la solución se definió el uso de la notación Camello. Esta notación consiste en escribir los identificadores con la primera letra de cada palabra en mayúsculas y el resto en

minúscula. Se llama notación Camello porque los identificadores recuerdan las jorobas de un camello. Existen dos variantes:

- *UpperCamelCase*, *CamelCase* o *PascalCase*: en esta variante la primera letra también es mayúscula.
- *lowerCamelCase*, *camelCase* o *dromedaryCase*: la primera letra es minúscula.

Esta notación ha sido utilizada en la creación de varios frameworks de distintos lenguajes de programación (como Java y C#) y está muy difundida a nivel internacional.

Para la declaración de variables y atributos, se propone el uso de la notación camello en su variante "*dromedaryCase*". Para la declaración de clases, métodos, interfaces y enumerativos se propone el uso de la notación camello en su variante "*PascalCase*", también conocida simplemente como Notación Pascal.

Los identificadores de los campos y tablas de la base de datos se nombraran con minúscula y si el nombre es compuesto las palabras se separarán con un guión bajo. El identificador de las tablas será un sustantivo en singular. Para las funciones, vistas, triggers y esquemas se utilizara el mismo principio antes expuesto, adicionándole un prefijo de la manera que queda expuesto en la siguiente tabla:

Elemento	Prefijo	Ejemplo
Funciones	fn	fn_insertar_perfil
Vistas	vw	vw_perfil_adn
Triggers	tr	tr_actualizar_perfil_asociado
Esquemas	sch	sch_publico

Como restricción, las palabras reservadas no podrán ser utilizadas como nombre de identificadores en ninguno de los casos. Todos deberán estar escritos en idioma español tradicional, aunque no se podrá utilizar ningún carácter especial ni los siguientes caracteres: á, é, í, ó, ú, ñ, Ñ.

3.4. Herramientas de desarrollo.

El Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) a utilizar será *Eclipse* en su versión 3.4, el cual provee al desarrollador de múltiples herramientas y componentes que agilizan el trabajo. El código fuente de la solución así como la documentación generada en los ciclos de desarrollo, será controlada con el uso de *Subversion*, un sistema de control de versiones que facilita el trabajo

colaborativo sobre los documentos y artefactos generados, además de posibilitar entre otras cosas, conocer qué desarrollador ha hecho un cambio en algún componente y cuando ha sido.

Para el modelado del proceso ingenieril se utilizará *Enterprise Architect* en su versión 7.0, que constituye una herramienta de diseño guiado por computadora (CASE, por sus siglas en inglés), debido a que se encuentra integrados perfectamente con el IDE de desarrollo propuesto, a través del plugin "*MDG Link for Eclipse*". Este plugin ofrece dentro del IDE una navegación y edición intuitiva de los modelos gracias al Project Explorer de EA. Con un click de ratón, le permite realizar ingeniería inversa y directa de código Java a partir de los artefactos UML mediante el Framework de Ingeniería de Código dirigido por plantillas de EA. Cubre, además, la generación de informes y documentos, de alta calidad tanto en formatos convencionales como web. Proporciona la funcionalidad que se necesita para una plataforma completa de modelado dentro de Eclipse.

Como herramienta visual para la administración del sistema gestor de bases de datos se propone el uso del *EMS PostgreSQL Manager*; el cual permitirá conectarse a múltiples bases de datos y programar procedimientos almacenados y vistas. Así como crear y editar nuevas tablas en el modelo relacional.

3.5. Lenguaje, tecnologías y herramientas de apoyo al desarrollo

Teniendo en cuenta lo expuesto en el acápite 1.2, los autores proponen seleccionar herramientas libres, fundamentalmente, para el desarrollo del Sistema de identificación por perfiles de ADN.

Como lenguaje de desarrollo se propone el uso de **JAVA**, puesto que es independiente de la plataforma –significa que programas escritos en este lenguaje pueden ejecutarse igualmente en cualquier tipo de hardware. Es un lenguaje orientado a objetos, incluye por defecto soporte para trabajo en red y además está diseñado para ejecutar código en sistemas remotos de forma segura. Además es fácil de usar y toma lo mejor de otros lenguajes orientados a objetos como C++ y hace un manejo automático de la memoria –no existen los punteros.

Con el uso de este lenguaje se podrá emplear el framework **JavaMail** que brinda funcionalidades para la creación de aplicaciones que necesiten intercambio de mensajes por protocolos de internet. A través de su utilización se puede enviar y leer correos electrónicos. Estas APIs (Application Programming Interfaces) requieren proveedores de servicios que implementen protocolos específicos como:

- SMTP: Protocolo de transporte para enviar correos electrónicos.

- POP3: Protocolo estándar para recibir correos electrónicos.
- IMAP: Protocolo alternativo a POP3.

Además del proveedor de servicios, JavaMail requiere **JAF** (JavaBeans Activation Framework) para gestionar el contenido de los mensajes que sean diferentes de texto plano. Esto incluye MIME (Multipurpose Internet Mail Extensions), páginas URL (Uniform Resource Locator), así como archivos adjuntos.

En la figura 10 se muestran los componentes que interactúan con JavaMail.

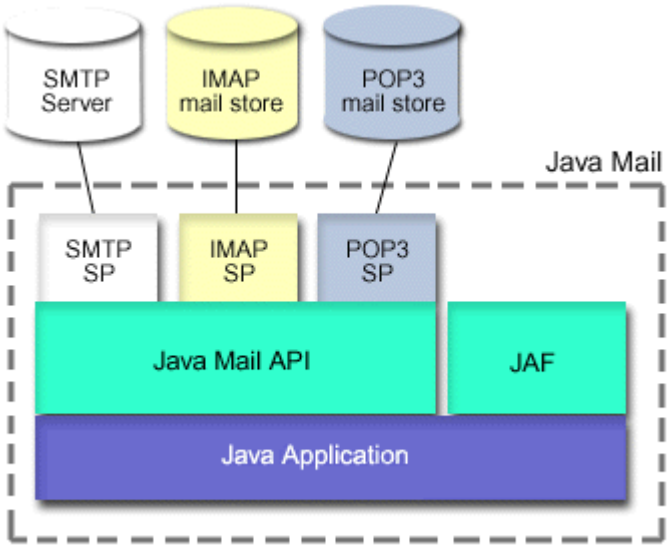


Figura 10. Descripción de las APIs de JavaMail

JavaBeans Activation Framework (JAF) es una extensión estándar de la plataforma Java que brinda ventajas en servicios como: determinar el tipo de una parte arbitraria de los datos, encapsular el acceso a la misma, identificar las operaciones disponibles en ella e instanciar lo que se necesita para ejecutar las operaciones.

Los autores proponen seleccionar la combinación entre **SMTP** y **POP3** como propuesta para la implementación de la comunicación entre los módulos del sistema que intervienen en la comparación de perfiles de ADN. La elección se debe a las bondades de SMTP para el envío de mensajes entre computadoras y otros dispositivos independientemente del fabricante o sistema operativo que utilicen. Con POP3 se pueden recuperar los mensajes de servidores remotos sin necesidad de estar conectado constantemente a la red. De esta manera SMTP será el encargado de enviar los mensajes y POP3 de

obtenerlos para los clientes. Las funcionalidades de estos protocolos brindan agilidad en este tipo de intercambio de mensajes.

Como servidor de correo electrónico se propone el uso de **SendMail**, ya que es un popular "agente de transporte de correo" (MTA, por sus siglas en Inglés) en Internet, cuya tarea consiste en "encaminar" los mensajes de correo de forma que estos lleguen a su destino. Se afirma que es el más popular MTA, corriendo sobre sistemas Unix y el responsable de la mayoría de envío de correos en internet.

La versión actual de SendMail, al igual que otros MTAs modernos, incorpora una serie de mejoras de seguridad y funciones opcionales que se pueden configurar para ayudar a prevenir problemas. Con el mismo se puede aprovechar también el ordenamiento de la cola de mensajes a enviar. Para esto se configuran parámetros para ordenar la cola por la prioridad, por el host de destino, por el estado de la conexión con el host de destino, o combinaciones entre estos.

Se propone añadir además una capa **SSL** -Protocolo de Capa de Conexión Segura-, es decir, un método de cifrado que puedan implementar tanto el servidor como el cliente para garantizar la seguridad durante el tráfico de los mensajes por la red.

El sistema gestor de bases de datos a usar será *PostgreSQL*, en su versión 8.3.1, el cual brinda una solución segura y eficiente, posibilita conexiones múltiples y posee herramientas para el trabajo directo sobre memoria RAM; además de no requerir el pago de licencias por su uso, al ser un gestor de tipo libre.

3.6. Definición de interfaces de comunicación

El mensaje de solicitud de comparación con la base de datos central, debe tener las siguientes características:

- En el campo asunto se escribirá el tipo de solicitud, seguido por su identificador.
- En el campo destino estará la dirección correspondiente con la sede central.
- En el campo remitente estará la dirección correspondiente con la sede que realiza la solicitud.
- En el cuerpo del correo viajará el perfil de ADN.

Los mensajes de confirmación tendrán la siguiente estructura:

- En el campo asunto irá el estado de la solicitud con su identificador.

- En el cuerpo del correo estará el identificador del nodo que envía la confirmación y la hora seguida de la palabra *success*.

El mensaje de respuesta deberá elaborarse de la siguiente manera:

- En el campo asunto se escribirá el tipo de solicitud, seguido por su identificador.
- En el campo destino estará la dirección correspondiente con la sede que realizó la solicitud.
- En el campo remitente estará la dirección correspondiente con la sede central.
- En el cuerpo del correo viajará la lista de identificadores de los casos cuyos perfiles de ADN se corresponden con el de la solicitud.

3.7. Representación arquitectónica.

En muchos libros y artículos se han intentado capturar todos los detalles de la arquitectura de un sistema usando un único diagrama. Pero si se observan detenidamente estos diagramas, se puede ver que en ellos se ha intentado representar más de un plano de lo que realmente podría expresar la notación. Generalmente los nodos representan varias cosas como programas en ejecución, partes de código fuente, computadores físicos o agrupaciones de funcionalidad y las líneas representan dependencias de compilación o flujos de control.

La arquitectura no requiere un estilo único. A veces esta tiene secuelas de un diseño donde se particionó prematuramente el software o se hizo un énfasis excesivo en la ingeniería de los datos, la eficiencia en tiempo de ejecución, o estrategias de desarrollo y organización de equipos. A menudo la arquitectura tampoco aborda los intereses de todos sus “clientes”.

El modelo de 4+1 vistas fue desarrollado para remediar este problema. Describe la arquitectura del software usando cinco vistas concurrentes, donde cada vista se refiere a un conjunto de intereses de diferentes *stakeholders* del sistema.

3.8. Modelo 4+1 vistas

El modelo 4 +1 vistas describe la arquitectura del software usando cinco vistas concurrentes, cada una de las cuales trata una serie de aspectos. Este modelo permite que diversas partes involucradas puedan encontrar lo que necesitan en la arquitectura de software. Los ingenieros de sistemas pueden

abordar en primer lugar la vista física y, a continuación, ver el proceso; los usuarios finales, clientes y especialistas, pueden aproximarse a los datos de la vista lógica, y los directores de proyectos y miembros del equipo de configuración del software, pueden abordar desde la visión de desarrollo. Estas cuatro vistas están guiadas por la vista de casos de uso que describe las funcionalidades del sistema que más inciden sobre su arquitectura.

Figura 11. Modelo 4+1 vistas

3.8.1. *Vista de casos de uso*

Esta vista representa un subconjunto del artefacto Modelo de casos de uso y lista los casos de usos o escenarios más significativos, con las funcionalidades centrales del sistema. Si el sistema se hace extenso entonces se debería organizar en paquetes, lo cual facilitaría la comprensión de la vista de casos de uso.

En la figura 12 se muestra el diagrama de casos de uso arquitectónicamente significativos.

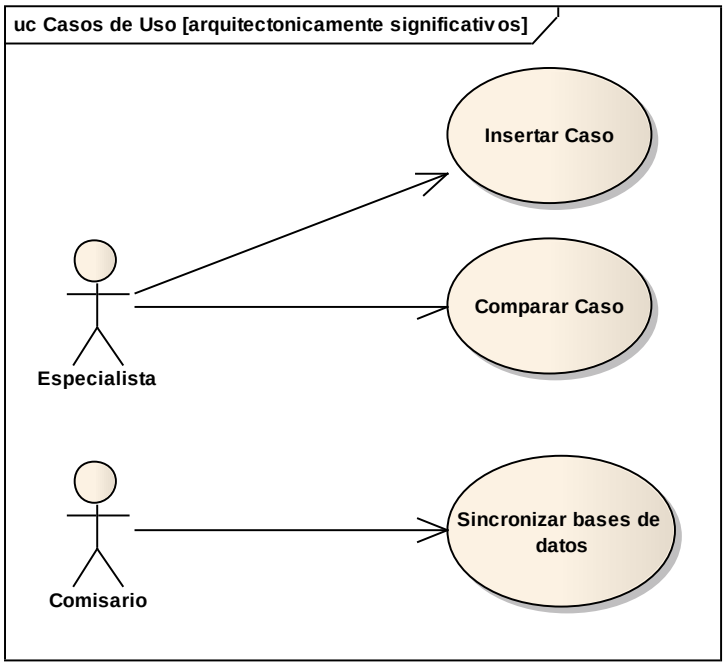


Figura 12. Diagrama de casos de usos arquitectónicamente significativos.

Estos casos de uso aparecen descritos en el Anexo 1 y encapsulan las funcionalidades esenciales que debe tener el sistema, correspondientes con los requisitos funcionales: RF1, RF1.1, RF1.2, RF2, RF2.1, RF2.2, RF2.3 y RF3.

En la segunda iteración se implementarán aquellos casos de uso que tienen menor prioridad dentro del sistema, pero que su funcionalidad es necesaria para completar la gestión eficiente de los perfiles de ADN asociados a los casos criminales.

Cód.	Nombre del Caso de Uso	Paquete
CU-3	Generar Informes	General
CU-4	Generar Reportes	General
CU-5	Modificar Caso	Gestionar Caso
CU-6	Eliminar Caso	Gestionar Caso
CU-8	Configurar Sistema	Administración
CU-9	Gestionar Usuarios	Administración
CU-10	Buscar Caso	Gestionar Caso

3.8.2. *Vista Lógica*

Esta vista representa un subconjunto del artefacto Modelo de diseño, el cual representa los elementos de diseño más importantes para la arquitectura del sistema. Describe las clases más importantes, su organización en paquetes y subsistemas, y estos a su vez en capas. También describe las realizaciones de casos de uso más importantes como por ejemplo las que describen aspectos dinámicos del sistema.

Los diagramas de clases de análisis y diseño correspondientes a los casos de uso arquitectónicamente significativos aparecen en los Anexos 2 y 3 respectivamente y los diagramas de interacción que describen la realización de dichos casos de uso se encuentran en el Anexo 4. A continuación se muestran en la figura 13, los paquetes y subsistemas identificados, distribuidos en las capas que se proponen para la aplicación.

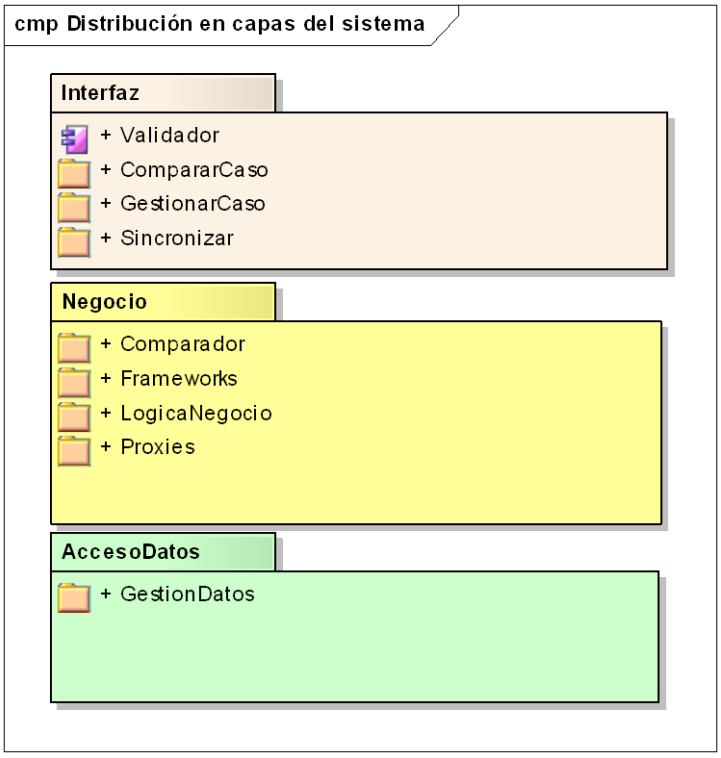


Figura 13. Distribución en capas.

3.8.3. *Vista de Despliegue.*

Esta vista suministra una base para la comprensión de la distribución física de un sistema a través de nodos. Suele utilizarse cuando el sistema está distribuido.

A continuación se presenta en la figura 14 el Diagrama de despliegue para los nodos del sistema que se propone.

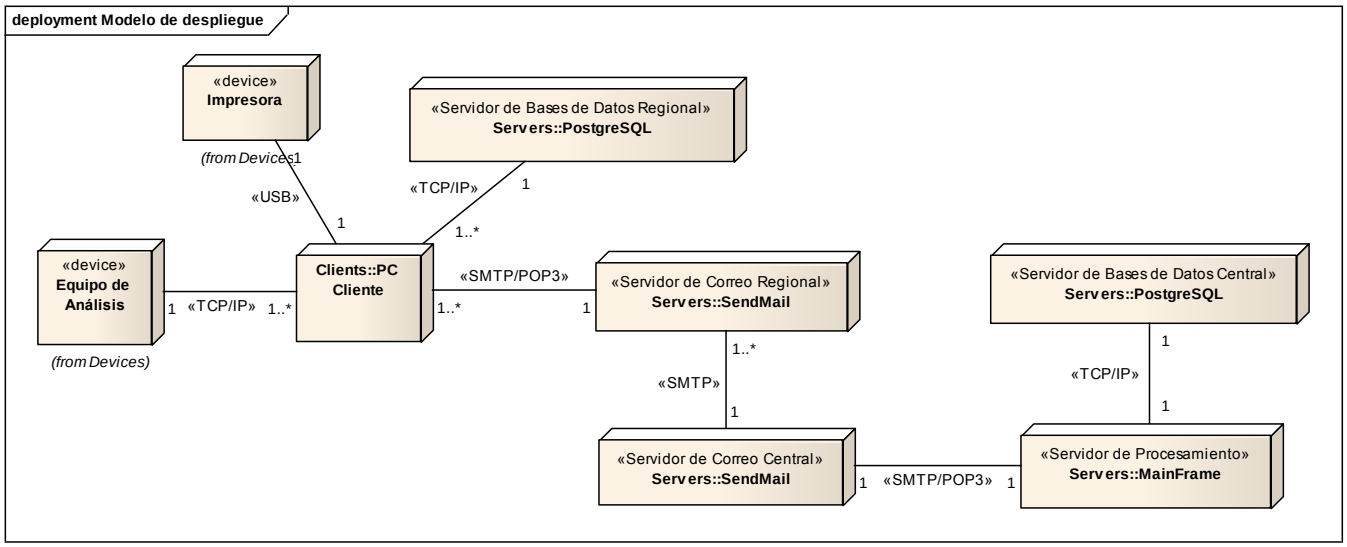


Figura 14. Diagrama de despliegue.

Nodo PC Cliente: En este nodo se desplegará la aplicación de escritorio con todas las funcionalidades que posibiliten la transmisión y recuperación de la información existente en la base de datos. Se comunicará con el servidor de intercambio de mensajería a través de protocolos estándares de correo electrónico para el envío de solicitud de comparaciones remotas.

Nodo Equipo de Análisis: Este nodo representa el equipo electrónico que procesa la muestra de ADN y le brinda al sistema el perfil de ADN correspondiente.

Nodo Impresora: Este nodo posibilita la impresión de los informes y reportes generados en el sistema.

Nodo Servidor de Bases de Datos Regional: Este nodo representa el servidor de base de datos de la sede regional, donde está instalado el SGBD PostgreSQL 8.3.

Nodo Servidor de Correo Regional: Posee instalado el servidor de correo SendMail de la sede regional.

Nodo Servidor de Correo Central: Posee instalado el servidor de correo SendMail de la sede central.

Nodo Servidor de Procesamiento: En este nodo se desplegará la aplicación de escritorio con todas las funcionalidades que posibiliten la transmisión y recuperación de la información existente en la base de datos. Responsable de dar respuesta a las solicitudes de comparaciones de las sedes regionales.

Se comunicará con el servidor de intercambio de mensajería a través de protocolos estándares de correo electrónico para el envío de resultados de solicitud de comparaciones.

Nodo Servidor de Bases de Datos Central: Este nodo representa el servidor de base de datos de la sede central, donde está instalado el SGBD PostgreSQL 8.3.

3.8.4. *Vista de Implementación.*

Esta vista describe la descomposición del software en capas y subsistemas de implementación. También provee una vista de la trazabilidad de los elementos de diseño de la vista lógica ahora para la implementación.

Esta contiene:

- Una enumeración de los subsistemas.
- Diagramas de componentes que ilustran la organización en capas y jerarquías de los subsistemas.
- Dependencia entre subsistemas.

En la figura 15 se muestra la estructuración de subsistemas en capas.

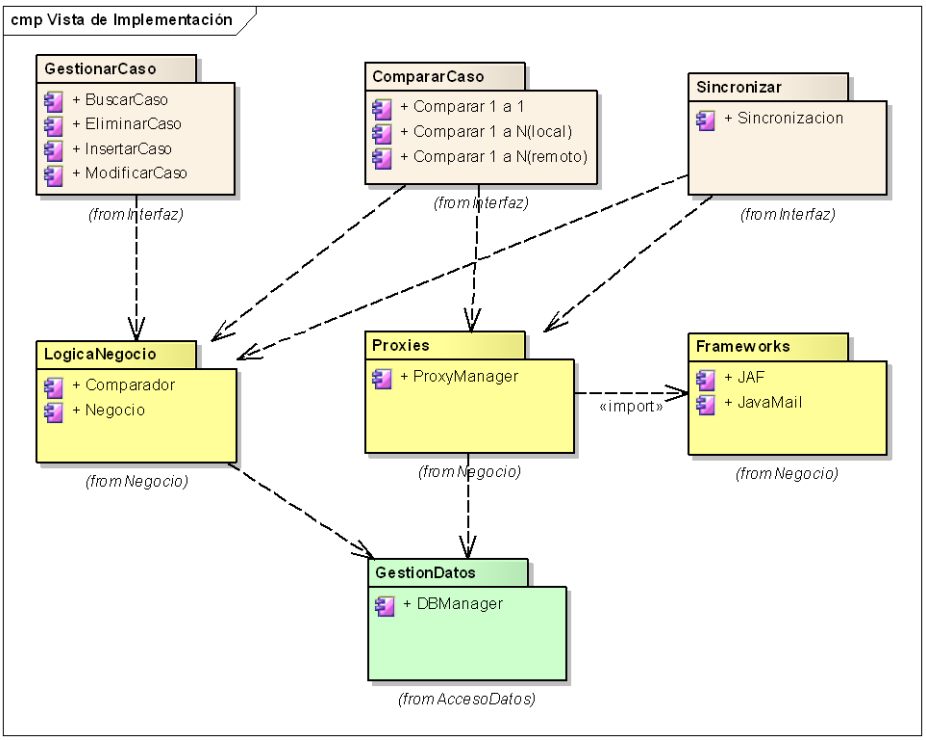


Figura 15. Modelo de implementación por capas.

Subsistema Gestionar Caso: Aquí se engloban las interfaces de usuario que intercambian información referida a los casos criminalísticos con los usuarios del sistema.

Subsistema Comparar Caso: Esta compuesto por las interfaces de usuario encargadas de guiar a los especialistas en la comparación de perfiles de ADN.

Subsistema Sincronizar: Agrupa las interfaces que sirven al actor correspondiente para solicitar la sincronización de los datos locales con los centrales.

Subsistema LógicaNegocio: Posee las clases necesarias para dar respuesta a los eventos ocurridos en las interfaces de usuario utilizando los datos de la capa inferior.

Subsistema Proxies: Agrupa el componente encargado de la mensajería entre las sedes regionales y centrales. Funciona como cliente de correo que construyen e interpretan estos mensajes.

Subsistema Frameworks: Son los componentes ya implementados que se reutilizarán en la implementación de las funcionalidades de los proxies.

Subsistema Comparador: Brinda el servicio de comparar un perfil de ADN con otro dando como resultado su similitud.

Subsistema GestiónDatos: Engloba las operaciones de manipulación de datos brindando servicio a las capas superiores.

Los principales componentes propuestos en los distintos subsistemas de implementación tienen las siguientes responsabilidades:

No	Componente	Descripción
1.	ProxyManager	Componente para servir de proxy a las diferentes peticiones de los clientes. Maneja las operaciones de los clientes: Comparar, Almacenar, Enviar a Comparar, Recibir Resultado, Buscar. Guarda el estado de las peticiones.
2.	DBManager	Maneja las operaciones a ejecutar relacionadas con la gestión de los datos. Utiliza el componente Comparador.
3.	Comparador	Componente que ejecuta las comparaciones.

4.	Sincronizador	Componente para sincronizar las BD de los sistemas. Es configurable y mantiene el estado de la última actualización, enviando alertas de ser necesario. Está integrado al componente DBManager.
5.	ADN_Client ADN_Server	Interfaz de usuario para interactuar con el sistema.

En este capítulo se logró una arquitectura independiente del módulo de comparación entre perfiles de ADN, permitiendo que pueda ser reutilizada en otros sistemas. Sus componentes son independientes y se relacionan entre sí. Estos logros estuvieron condicionados por el empleo de buenas prácticas definidas en patrones y estilos que dan solución a ciertos problemas.

Se describió la arquitectura del sistema propuesto en términos de componentes, subsistemas y clases que serán implementados en cada nodo físico; qué responsabilidades tendrá cada uno y como se comunicarán entre ellos para lograr el objetivo del sistema. Según las restricciones de la aplicación y la política tecnológica de la Universidad, se definieron las herramientas, lenguaje de programación y tecnologías a utilizar en la implementación.

CONCLUSIONES

El desarrollo de este trabajo permitió elaborar una propuesta de arquitectura para un sistema de identificación por perfiles de ADN, para ello se cumplió con los objetivos y tareas propuestas:

- En la mayoría de las regiones altamente automatizadas existen bases de datos y aplicaciones para el manejo electrónico de la información referente a la identificación a través del ADN. En Cuba no existe una solución de este tipo con alcance nacional.
- Aunque existe variedad en la utilización de marcadores para conformar el perfil de ADN, trece de ellos son los más reconocidos de manera internacional.
- Para la mensajería entre los módulos que se proponen, se identificó la combinación de SMTP/POP3 como protocolos a utilizar, por el negocio que tienen implementado y la posibilidad de manejo y configuración.
- El sistema informático se desarrollará para Sistema Operativo GNU Linux, distribución Debian; utilizando Java como lenguaje de programación y PostgreSQL como sistema gestor de bases de datos por las necesidades de independencia tecnológica, seguridad, adaptación y la posibilidad de la utilización de frameworks que se poseen.
- Tiene una arquitectura basada en componentes independientes que se relacionan entre sí. Los principales parámetros para su funcionamiento son altamente configurables y se prestó especial atención a la seguridad y confiabilidad en el manejo de los datos.
- Permite gestionar toda la información referente a los casos criminalísticos y realizar comparaciones en diferentes escenarios.
- La lógica de comparación de perfiles de ADN es transparente a la arquitectura, posibilitando que la misma pueda ser utilizada en el desarrollo de otros sistemas de identificación biométricos.
- Posee un diseño orientado a objetos cumpliendo con los patrones de asignación de responsabilidades. La arquitectura está basada en estilos arquitectónicos como la arquitectura en capas para simplificar la comprensión y la organización del desarrollo y reducir las dependencias, y la arquitectura orientada a servicios para la comunicación entre los módulos.

RECOMENDACIONES

Después de haber concluido la investigación los autores recomiendan:

- Continuar refinando la arquitectura en las diferentes iteraciones de desarrollo del sistema.
- Evaluar la arquitectura en aras de mejorar la visión de los procesos críticos y validar las decisiones de diseño que se tomaron.
- Evaluar el costo de la tecnología, fundamentalmente en cuanto a requerimientos de hardware para buscar variantes que reduzcan los mismos.
- Estudiar la arquitectura propuesta para que sea utilizada en otros sistemas de identificación biométrica a desarrollarse en el área temática de Procesamiento Digital de Imágenes de la Facultad 7.

REFERENCIAS BIBLIOGRÁFICAS

(Dorte Hammelev, 2008) **Dorte Hammelev, Dean Madden, Jill Turner. 2008.** DNA Profiling. European Initiative for Biotechnology Education. [En línea] Enero de 2008. http://www.eibe.reading.ac.uk/.

(Hansen, y otros, 1997) **Hansen, Gary W. y Hansen, James V. 1997.** *Diseño y Administración de Bases de Datos.* Madrid : Pretince Hall, 1997. 8483220024.

(JavaTech, 2004) **JavaTech. 2004.** An Introduction to Scientific and Technical Computing with Java. *JavaTech.* [En línea] 2004. [Citado el: 21 de Noviembre de 2008.] http://www.particle.kth.se/~lindsey/JavaCourse/Book/courseMap.html.

(Jones, 2006) **Jones, George. 2006.** DNA database' should include all. *Telegraph.co.uk.* [En línea] Telegraph Media Group Limited, 24 de Octubre de 2006. [Citado el: 20 de Noviembre de 2008.] http://www.telegraph.co.uk/news/uknews/1532210/DNA-database-%27should-include-all%27.html.

(Kimball, 2005) **Kimball, John W. 2005.** Restriction Fragment Length Polymorphisms (RFLPs). *Kimball's Biology Pages.* [En línea] 12 de Septiembre de 2005. [Citado el: 14 de Noviembre de 2008.] http://users.rcn.com/jkimball.ma.ultranet/BiologyPages/R/RFLPs.html.

(Microsoft, 2008) **Microsoft. 2008.** *MSDN.* [En línea] 2008. http://msdn.microsoft.com/es-ar/default.aspx.

(Reynoso, 2004) Reynoso, Billy. 2004. Profundizando en Estilos de Arquitectura de Software. 2004.

(Reynoso y otros, 2004) Reynoso, Carlos y Kiccillof, Nicolás. 2004. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. Buenos Aires: s.n., 2004.

(Roth, 1996) **Roth, W. Barry. 1996.** DNA Profile Analysis. *Access Excellence.* [En línea] 1996. [Citado el: 01 de Noviembre de 2008.] http://www.accessexcellence.org/AE/AEC/AEF/1996/roth_dna.php.

(Softcover, 2000) **Softcover, John Sanders. 2000.** *Forensic Casebook of Crime.* London : True Crime Library/Forum Press, 2000. 0-330-42106-9.

(Stroustrup, 1998) **Stroustrup, Bjarne. 1998.** *El lenguaje de programación C++.* Madrid : Addison-Wesley Pub Co, 1998. ISBN 84-7829-019-2.

(Wikipedia, 2008) **Wikipedia. 2008.** C Sharp. *Wikipedia. La Enciclopedia Libre.* [En línea] 2008. [Citado el: 21 de Noviembre de 2008.] http://es.wikipedia.org/wiki/C_Sharp.

BIBLIOGRAFÍA

1. **Ana Isabel Morales, Bernal Morera, Gerardo Jiménez-Arce. 2003.** La implementación forense de la tecnología del ADN en Costa Rica: Un análisis retrospectivo. *Revista de Biología Tropical.* [En línea] 20 de Octubre de 2003. [Citado el: 20 de Noviembre de 2008.] http://www.scielo.sa.cr/scielo.php?pid=S0034-77442004000300030&script=sci_arttext#rod96. ISSN 0034-7744.

2. **Aneiro Rodríguez, Lazaro Orlando. 2001.** *Elementos de arquitectura y seguridad informática.* s.l. : Editorial Pueblo y Educación, 2001. 959-13-0819-1.

3. **Baden, Dr. Michael. 2008.** DNA Profiling. *The Official Web Site of Dr. Kathy Reichs.* [En línea] Julio de 2008. [Citado el: 13 de Noviembre de 2008.] http://www.kathyreichs.com/dnaprofiling.htm.

4. **Bravo Estrada, Diego.** Tutorial de SendMail. [En línea] [Citado el: 24 de Enero de 2009.] http://es.tldp.org/Tutoriales/doc-guia-sendmail/doc-guia-sendmail-html/#AEN12.

5. Capítulo 16. Correo electrónico. *Red Hat Linux 7.3: Manual oficial de referencia de Red Hat Linux.* [En línea] [Citado el: 16 de Enero de 2009.] http://www.tu-chemnitz.de/docs/lindocs/RH73/RH-DOCS/rhl-rg-es-7.3/ch-email.html.

6. **Cerami, Ethan. 2002.** *Web Services Essentials. Distributed Applications with XML-RPC, SOAP, UDDI & WSDL.* s.l. : O'Reilly, 2002. 0-596-00224-6.

7. **Clements, Paul. 1996.** *Coming attractions in Software Architecture.* 1996.

8. **2008.** CODIS markers. *DNA Testing Systems.* [En línea] 2008. [Citado el: 19 de Noviembre de 2008.] http://www.dnaconsultants.com/Detailed/335.html.

9. **DDC. 2008.** Short Tandem Repeats (STRs). *DDC Forensic Department. DNA Diagnostics Center.* [En línea] 2008. [Citado el: 20 de Noviembre de 2008.] http://www.forensicdnacenter.com/dna-str.html.

10. **DENNIS, ALAN L. 2003.** *.NET Multithreading.* s.l. : Manning Publications Co., 2003.

11. **Dorte Hammelev, Dean Madden, Jill Turner. 2008.** DNA Profiling. *European Initiative for Biotechnology Education.* [En línea] Enero de 2008. http://www.eibe.reading.ac.uk/.

12. **Echarte, P. 2005.** *Introducción a la plataforma .NET y Mono.* 2005.

13. El protocolo HTTP. *Herramientas Web para la enseñanza de protocolos de comunicación.* [En línea] [Citado el: 16 de Enero de 2009.] http://neo.lcc.uma.es/evirtual/cdd/tutorial/aplicacion/http.html.

14. **Fajardo Martin, Jesmar Ernesto y Cardero Alvarez, Rafael Leodan. 2008.** *Localización y segmentación de matrículas de vehículos en imágenes digitales.* La Habana : s.n., 2008. Tesis de Grado.

15. **FBI. 2007.** National DNA Index System. *Federal Bureau of Investigation.* [En línea] US Department of Justice, Octubre de 2007. [Citado el: 19 de Noviembre de 2008.] http://www.fbi.gov/hq/lab/codis/national.htm.

16. **Fonseca, R. A. 2002.** *Programación de funciones en PL/pgSQL para PostgreSQL.* 2002.

17. **Fowler, M. 2002.** *Patterns of Enterprise Application Architecture.* Boston : Addison-Wesley, 2002.

18. **Franco Navarro, J. A. 2005.** *UML en acción. Modelando Aplicaciones Web.* 2005.

19. **Fuentes Suarez, Mainoldis y Ramos Medina, Alain.** Propuesta de arquitectura para sistemas de gestión hospitalaria. *Capítulo 3.* La Habana : s.n.

20. **Group, PostgreSQL Global Development. 2008.** PostgreSQL. *PostgreSQL.* [En línea] PostgreSQL Global Development Group, 2008. [Citado el: 22 de Noviembre de 2008.] http://www.postgresql.org/.

21. **Hansen, Gary W. y Hansen, James V. 1997.** *Diseño y Administración de Bases de Datos.* Madrid : Pretince Hall, 1997. 8483220024.

22. **Hicks, John W. 1995.** The Alabama DNA Database System. *Promega.* [Online] Noviembre 1995. [Cited: Noviembre 20, 2008.] http://www.promega.com/geneticidproc/ussymp6proc/hicks.htm.

23. **2005.** http://www.beta-paternidad.com/markers.html. *Beta Paternidad.* [En línea] El Primer Servicio en el Mundo en Pruebas de Paternidad realizadas en Casa, 2005. [Citado el: 20 de Noviembre de 2008.] http://www.beta-paternidad.com/markers.html.

24. **IBM.** Overview of JavaMail APIs. *i5/OS Information Center, Version 5 Release 4.* [Online] [Cited: Enero 24, 2009.] http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp?topic=/rzamy/50/program/jmluse.htm..

25. **J.M. Butlera, Corresponding Author Contact Information, E-mail The Corresponding Author, M.D. Coblea, A.E. Deckera, D.L. Duewerb, C.R. Hilla, M.C. Klinea, J.W. Redmana, P.M. Vallonea. 2006.**

Setting standards and developing technology to aid the human identity testing community. *ScienceDirect.* [En línea] Abril de 2006. [Citado el: 18 de Noviembre de 2008.] http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B7581-4JSBXBX-7K&_user=2342189&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000056883&_version=1&_urlVersion=0&_userid=2342189&md5=aa286bc2afaf2c57c2d406ced5ec729d.

26. **Jacobson, I., Booch, G., and Rumbaugh, J. 1999.** *The Unified Software Development Process.* 1999.

27. **Jacobson, Ivar, Rumbaugh, James y Booch, Grady. 2000.** *El lenguaje unificado de Modelado. Manual de Referencia.* 2000.

28. JavaBeans Activation Framework (JAF). *Sun Developer Network. .* [En línea] Sun Microsystems. [Citado el: 24 de Enero de 2009.] http://java.sun.com/javase/technologies/desktop/javabeans/jaf/.

29. **JavaTech. 2004.** An Introduction to Scientific and Technical Computing with Java. *JavaTech.* [En línea] 2004. [Citado el: 21 de Noviembre de 2008.] http://www.particle.kth.se/~lindsey/JavaCourse/Book/courseMap.html.

30. **John M. S. Bartlett, David Stirling. 2003.** *PCR Protocols.* 2003. págs. 3-6. 1-59259-384-4.

31. **Jones, George. 2006.** DNA database' should include all. *Telegraph.co.uk.* [En línea] Telegraph Media Group Limited, 24 de Octubre de 2006. [Citado el: 20 de Noviembre de 2008.] http://www.telegraph.co.uk/news/uknews/1532210/DNA-database-%27should-include-all%27.html.

32. **Kimball, John W. 2005.** Restriction Fragment Length Polymorphisms (RFLPs). *Kimball's Biology Pages.* [En línea] 12 de Septiembre de 2005. [Citado el: 14 de Noviembre de 2008.] http://users.rcn.com/jkimball.ma.ultranet/BiologyPages/R/RFLPs.html.

33. **Kruchten, Philippe.** Planos Arquitectónicos: El Modelo de "4+1" Vistas de la Arquitectura del Software. [En línea] [Citado el: 24 de Enero de 2009.]

34. **—.** The 4+1 View Model of Architecture. *CS DIGITAL LIBRARY. .* [En línea] [Citado el: 26 de Enero de 2009.] http://www2.computer.org/portal/web/csdl/doi/10.1109/52.469759.

35. **Larman, C. 2002.** *UML y Patrones.* Segunda Edición. s.l. : Addison Wesley Professional, 2002.

36. **—. 1999.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 1999.

37. **Larman, Craig. 2004.** *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development.* s.l. : Addison Wesley Professional, 2004. 0-13-148906-2.

38. **Liberty, Jesse. 2005.** *Visual C# 2005: A Developer's Notebook.* s.l. : O'Reilly, 2005.

39. **Microsoft. 2008.** *MSDN.* [En línea] 2008. http://msdn.microsoft.com/es-ar/default.aspx.

40. **Msezane, James Temba. 2006.** DNA Based Identification and Tracking System. *Free Patents Online.* [Online] FreePatentsOnline.com, Febrero 2006. [Cited: Noviembre 21, 2008.] http://www.freepatentsonline.com/y2006/0285685.html.

41. **2007.** NEC puts DNA lab in a briefcase . *Pink Tentacle.* [En línea] 25 de Septiembre de 2007. [Citado el: 20 de Noviembre de 2008.] http://www.pinktentacle.com/2007/09/nec-puts-dna-lab-in-a-briefcase-for-the-man/.

42. **Novell. 2008.** Mono 2.0.1 Release Notes. *Mono Project.* [En línea] 23 de Octubre de 2008. [Citado el: 21 de Noviembre de 2008.] http://www.mono-project.com/Release_Notes_Mono_2.0.1.

43. **Pressman, Roger S. 2002.** *Ingeniería de Software. Un enfoque práctico.* 5ta edición. s.l. : McGraw-Hill, 2002.

44. Protocolos TCP / IP. [En línea] [Citado el: 10 de Enero de 2009.] http://www4.uji.es/~al019803/tcpip/index.htm.

45. **Ramos Solis, Laura Yesenia. 2005.** ¿Qué son los marcadores moleculares? *La Ciencia y el Hombre. Revista de divulgación Científiica y Tecnológica de la Universidad Veracruzana.* [En línea] Volumen XVIII. Número 1, Abril de 2005. [Citado el: 20 de Noviembre de 2008.] http://www.uv.mx/cienciahombre/revistae/vol18num1/articulos/moleculares/index.htm.

46. **Reynoso, Billy. 2004.** *Profundizando en Estilos de Arquitectura de Software.* 2004.

47. **Reynoso, Carlos y Kiccillof, Nicolás. 2004.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Buenos Aires : s.n., 2004.

48. **Roth, W. Barry. 1996.** DNA Profile Analysis. *Access Excellence.* [En línea] 1996. [Citado el: 01 de Noviembre de 2008.] http://www.accessexcellence.org/AE/AEC/AEF/1996/roth_dna.php.

49. **Softcover, John Sanders. 2000.** *Forensic Casebook of Crime.* London : True Crime Library/Forum Press, 2000. 0-330-42106-9.

50. **Solutions, DNA.** DS - Lociler. *DNA Solutions.* [En línea] DNA Solutions. http://www.dnasolutions.es/productos.html.

51. **Stroustrup, Bjarne. 1998.** *El lenguaje de programación C++.* Madrid : Addison-Wesley Pub Co, 1998. ISBN 84-7829-019-2.

52. Usabilidad y arquitectura del software. *desarrolloweb.com.* [En línea] [Citado el: 24 de Enero de 2009.] http://www.desarrolloweb.com/articulos/1622.php.

53. **Vegas, J. 2006.** *Desarrollo de aplicaciones Web.* 2006.

54. **Wikipedia. 2008.** C Sharp. *Wikipedia. La Enciclopedia Libre.* [En línea] 2008. [Citado el: 21 de Noviembre de 2008.] http://es.wikipedia.org/wiki/C_Sharp.

ANEXOS

Anexo 1. Casos de Uso expandidos

Caso de uso	
CU-1	Insertar Caso
Propósito	Almacenar un caso criminalístico y perfil de ADN asociado en la base de datos.
Actores: Especialista (inicia)	
Resumen: El Especialista introduce en el sistema los datos relacionados al caso criminalístico. Los datos del perfil de ADN asociado pueden ser introducidos de manera manual o por la interfaz de comunicación con el equipo que los genera. Los datos son almacenados en la base de datos local.	
Referencias	RF1, RF1.1

Acción del actor	**Respuesta del sistema**
1. Selecciona la opción Insertar Caso.	2. Muestra interfaz con los campos relacionados con un Caso.
3. Introduce los datos referentes al Caso.	4. Valida datos del Caso.
4. Solicita insertar datos del perfil de ADN	5. Almacena en memoria datos del Caso.
	6. Muestra interfaz con los campos relacionados con el perfil de ADN.
7. Introduce los datos referentes al perfil de ADN.	8. Almacena los datos en la base de datos local y finaliza el caso de uso.

Caso de uso	
CU-2	Comparar Caso
Propósito	Comparar un perfil de ADN con otro(s).

Actores: Especialista (inicia)	
Resumen: El Especialista selecciona el tipo de comparación que desea realizar. Introduce los datos a comparar y el sistema devuelve una respuesta acorde a dicha comparación.	
Referencias	RF1.2, RF2, RF2.1, RF2.2, RF2.3
Flujo normal de los eventos	
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción de comparación que desea realizar.	2. En caso de que la selección sea Comparar 1:n local, ir a la sección Comparar 1:n local; si es 1:1 local, ir a la sección Comparar 1:1 local y si es 1:n remota, ir a la sección Comparar 1:n remota.
Sección Comparar 1:n local	
Acción del actor	**Respuesta del sistema**
3. Introduce datos del perfil de ADN y pulsa el botón comparar.	4. Busca los perfiles que coincidan con este en la Base de Datos.
	5. Muestra el código y nombre del caso correspondientes a cada perfil similar encontrado en la BD.
6. Acepta y se termina el caso de uso.	
Flujo alternativo: Sin perfiles similares	
Acción del actor	**Respuesta del sistema**
	5. El sistema muestra un mensaje informando que en la DB no existe ningún perfil similar al proveído por el actor.
6. Acepta y se termina el caso de uso.	
Sección Comparar 1:1 local	
Acción del actor	**Respuesta del sistema**
3. Introduce datos del primer perfil de ADN y pulsa Siguiente.	4. Almacena en memoria dichos datos y muestra la siguiente interfaz.

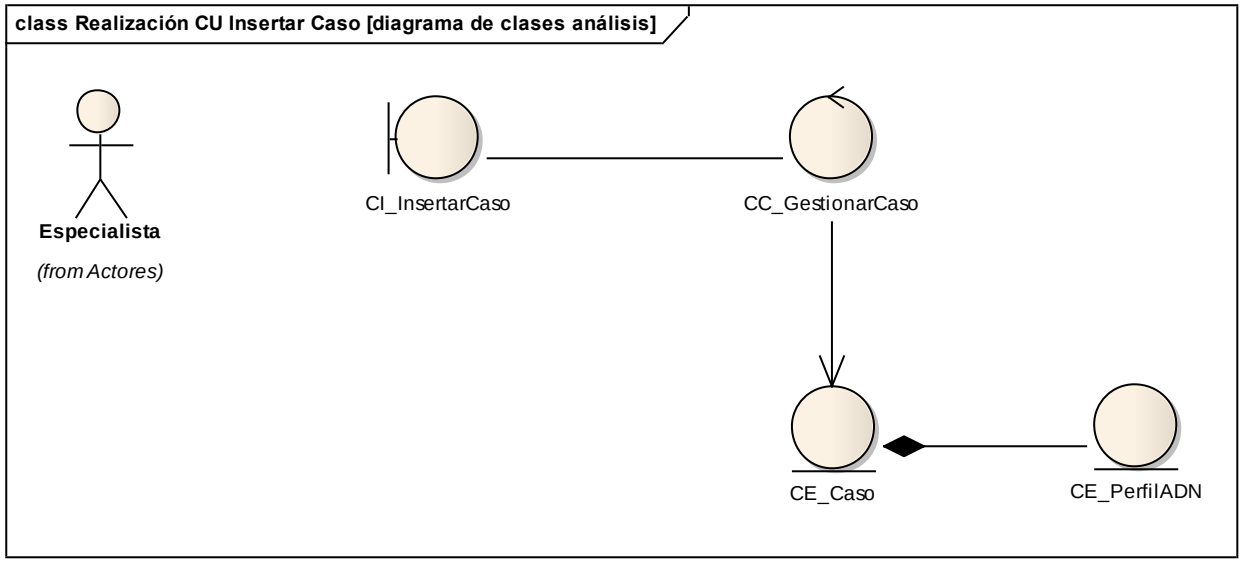
5. Introduce los datos del otro perfil de ADN y pulsa Comparar.	6. Realiza la comparación.
	7. Muestra mensaje con el resultado de la comparación.
8. Acepta y se termina el caso de uso.	
Sección Comparar 1:n remota	
Acción del actor	**Respuesta del sistema**
3. Selecciona la Base de Datos en la que desea comparar y pulsa Siguiente.	4. Establece conexión con dicho servidor remoto.
	5. Muestra interfaz con los campos relacionados con el perfil de ADN.
6. Introduce los datos del perfil de ADN y pulsa comparar.	7. Busca los perfiles que coincidan con este en la Base de Datos especificada.
	8. Muestra el código y nombre del caso correspondientes a cada perfil similar encontrado en la BD.
9. Acepta y se termina el CU.	
Flujo alternativo: Sin conexión con servidor remoto.	
Acción del actor	**Respuesta del sistema**
	5. Muestra mensaje anunciando que el servidor remoto no está disponible y no se podrá realizar la operación en este momento.
	6. Almacena la información temporalmente para su envío cuando haya conexión.
6. Acepta y se termina el CU.	
Flujo alternativo: Sin perfil similar	
Acción del actor	**Respuesta del sistema**
	8. El sistema muestra un mensaje informando que en la Base de Datos no existe ningún perfil similar al introducido por el actor.

9. Acepta y se termina el caso de uso.	

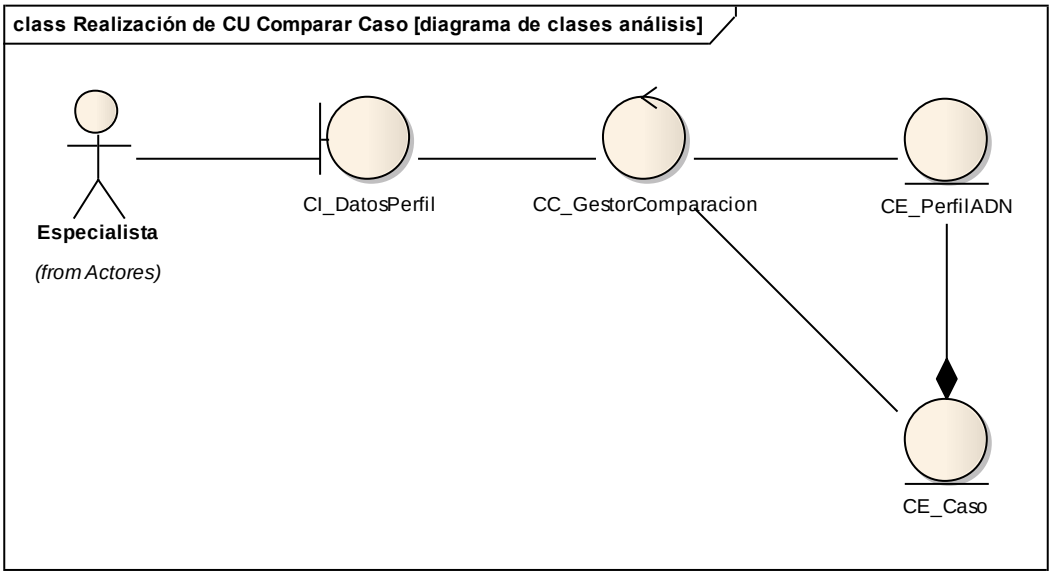
Caso de uso		
CU-7	Sincronizar bases de datos	
Propósito	Sincronizar los datos locales con los del servidor central	
Actores: Comisario(inicia)		
Resumen: El comisario realiza la sincronización de los datos locales con los almacenados en el servidor central.		
Referencias	RF3.	
Acción del actor		**Respuesta del sistema**
1. Selecciona la opción sincronizar bases de datos.		2. Establece comunicación con el servidor central.
		3. Realiza sincronización.
		4. Muestra mensaje de sincronización realizada satisfactoriamente.
5. Acepta y termina el caso de uso.		
Flujo alternativo: Sin conexión con servidor remoto.		
Acción del actor		**Respuesta del sistema**
		3. Muestra mensaje de error de sincronización.
4. Acepta y termina el caso de uso.		

Anexo 2. Diagrama de Clases del Análisis

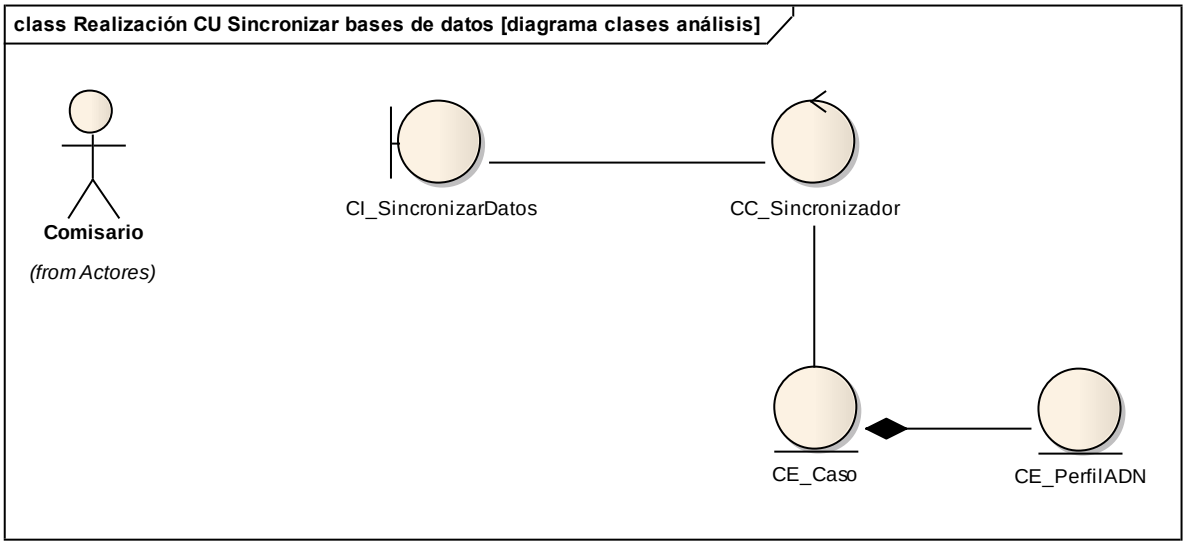
CU-1 Insertar Caso



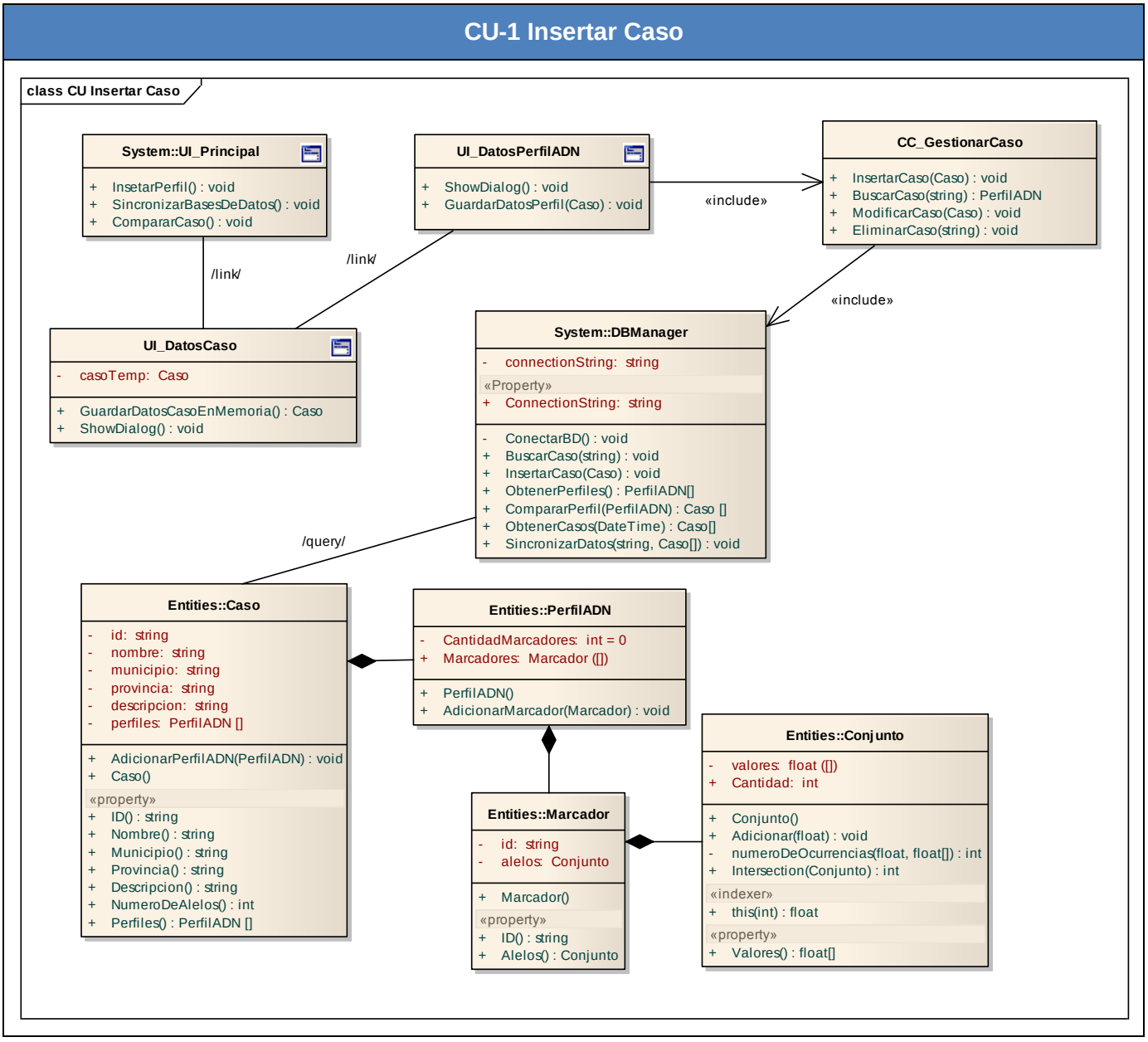
CU-2 Comparar Caso



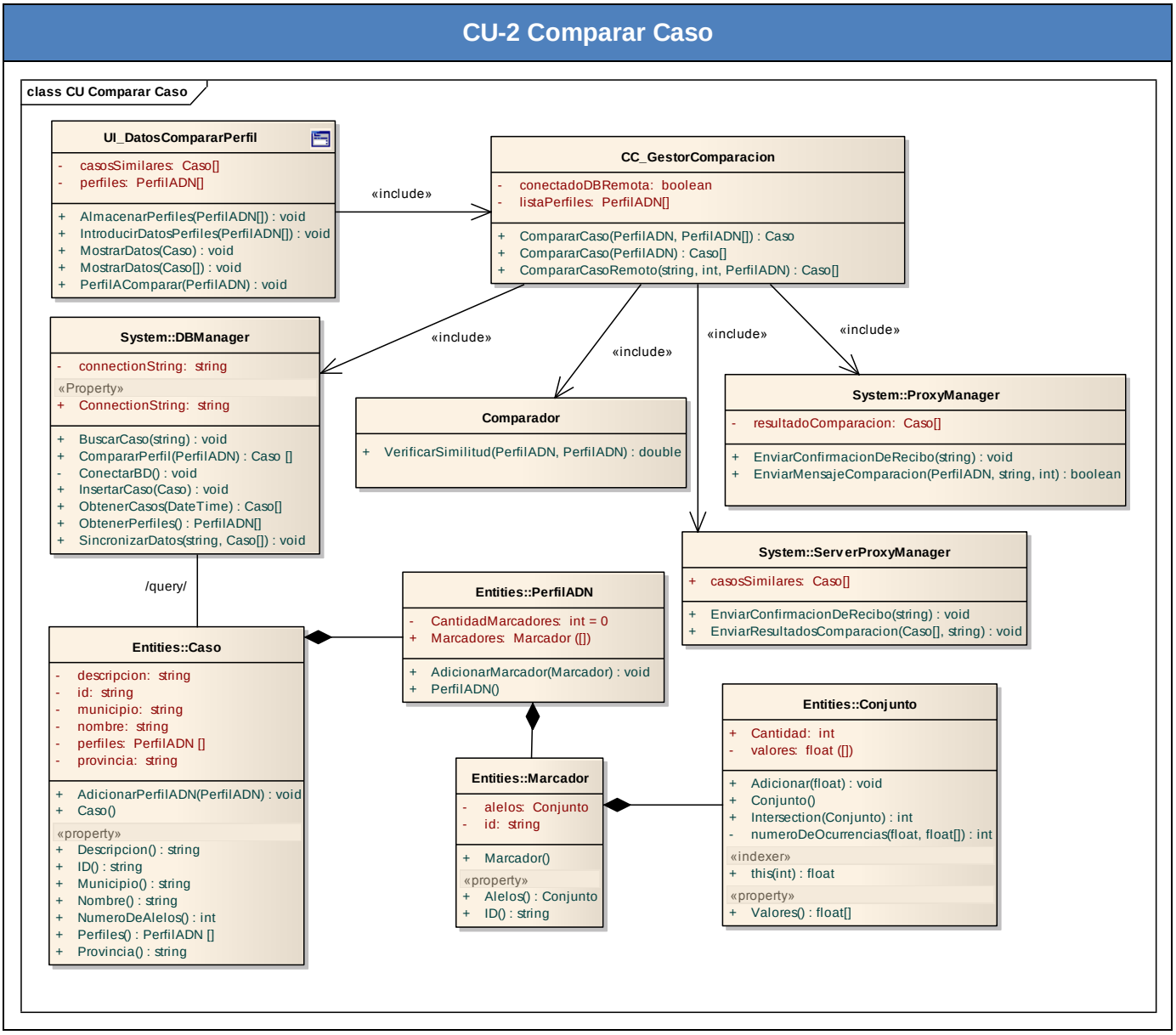
CU-7 Sincronizar bases de datos



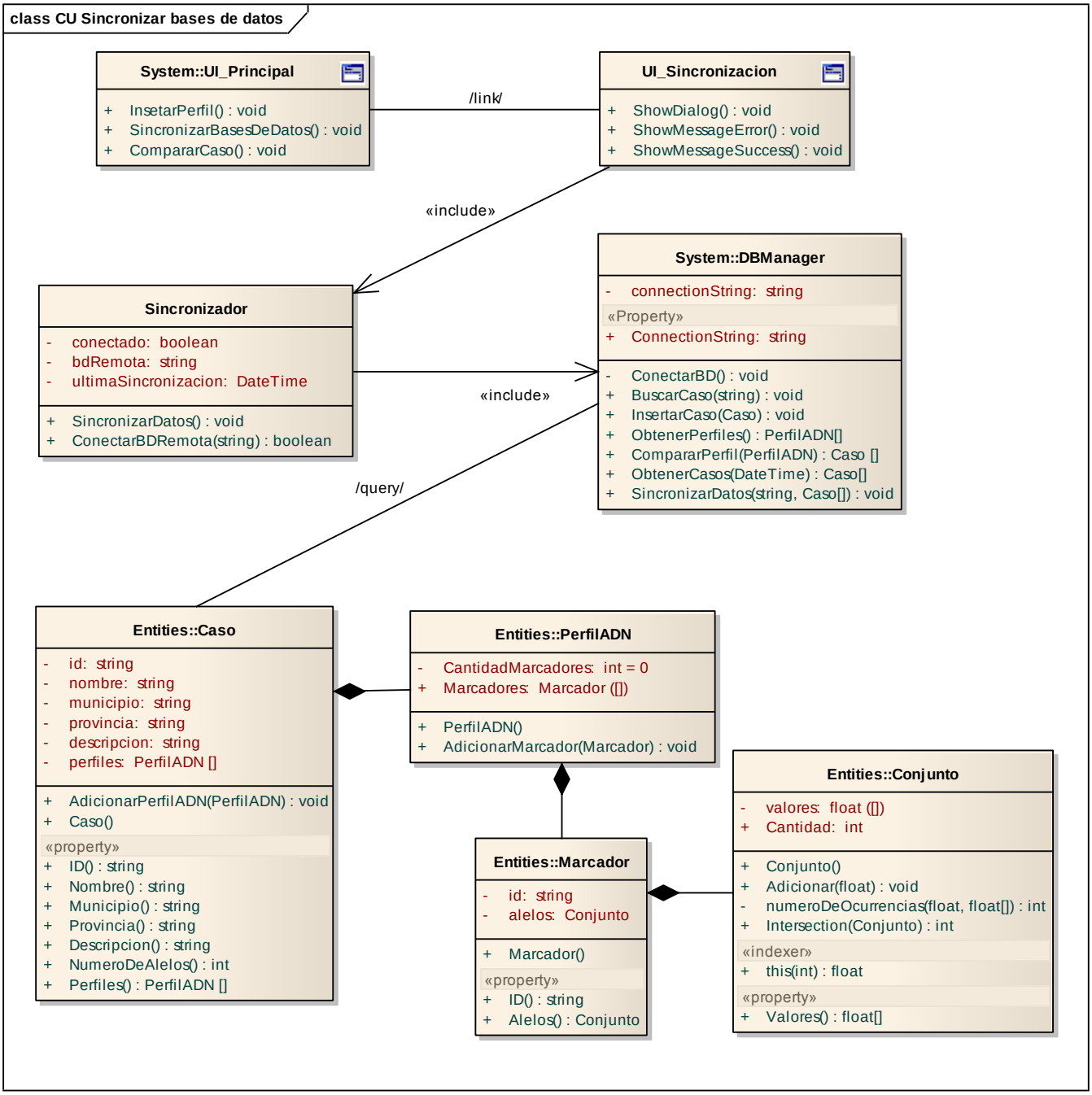
Anexo 3. Diagrama de clases del diseño



CU-2 Comparar Caso

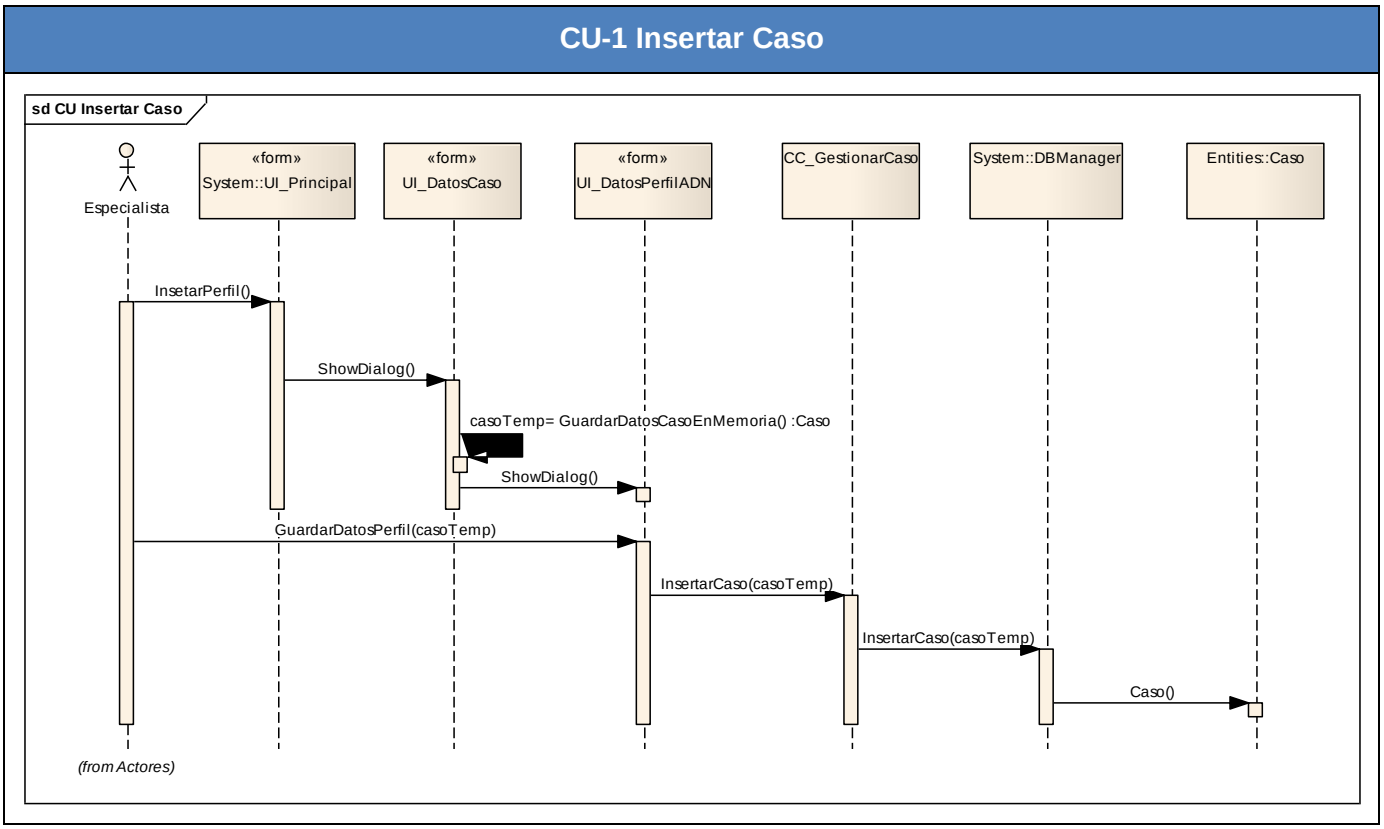


CU-7 Sincronizar bases de datos

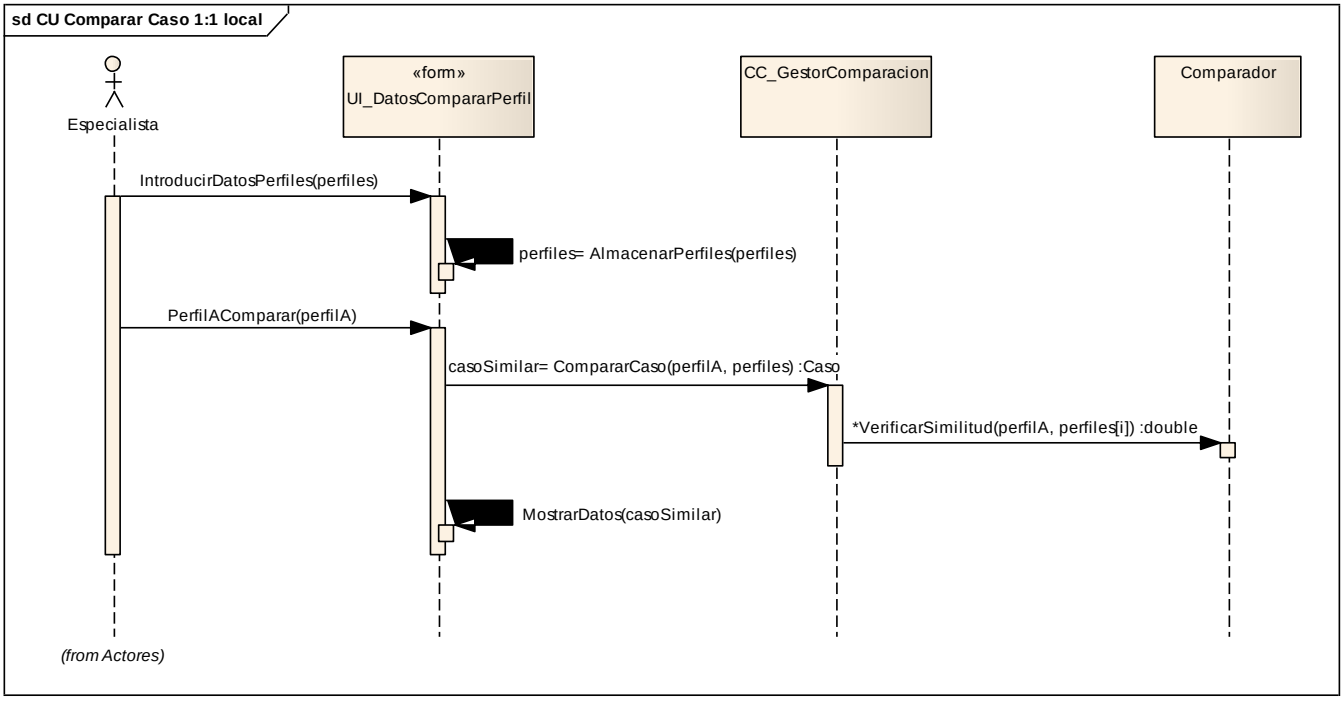


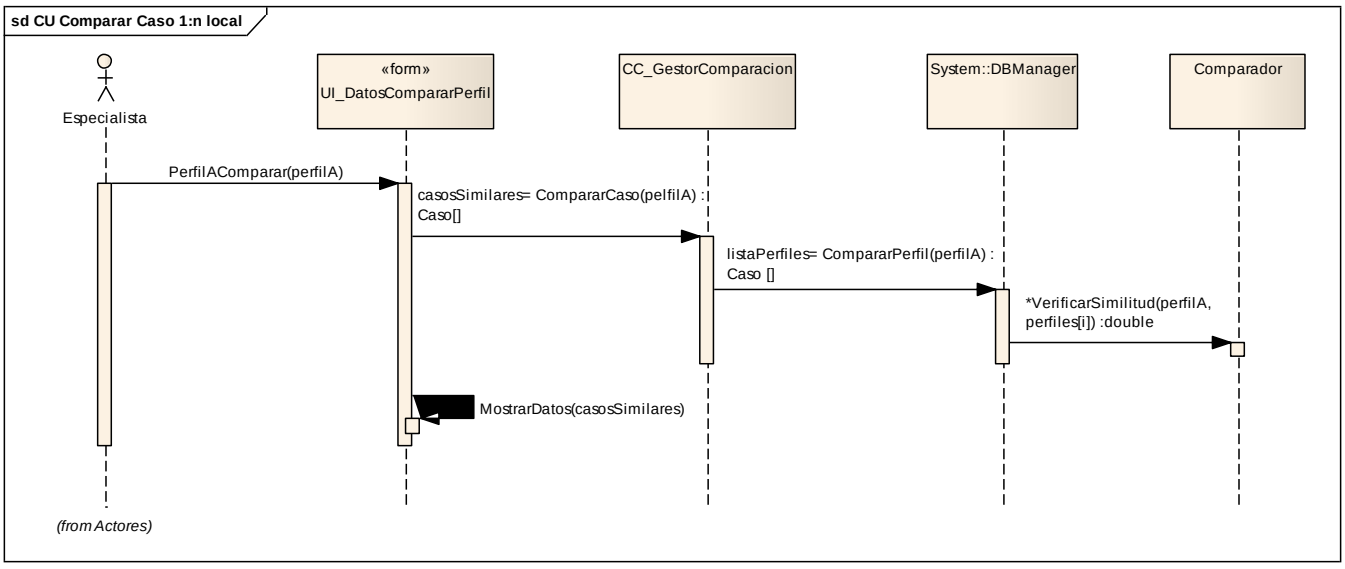
Anexo 4. Diagramas de interacción

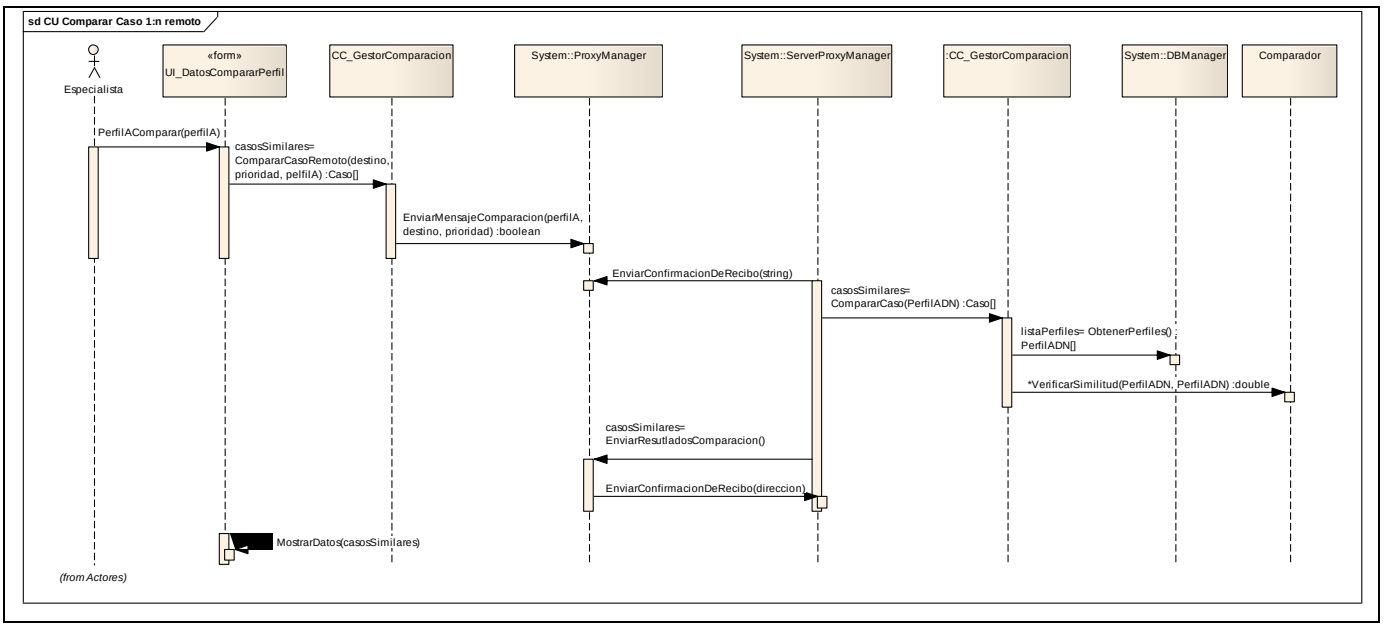
CU-1 Insertar Caso



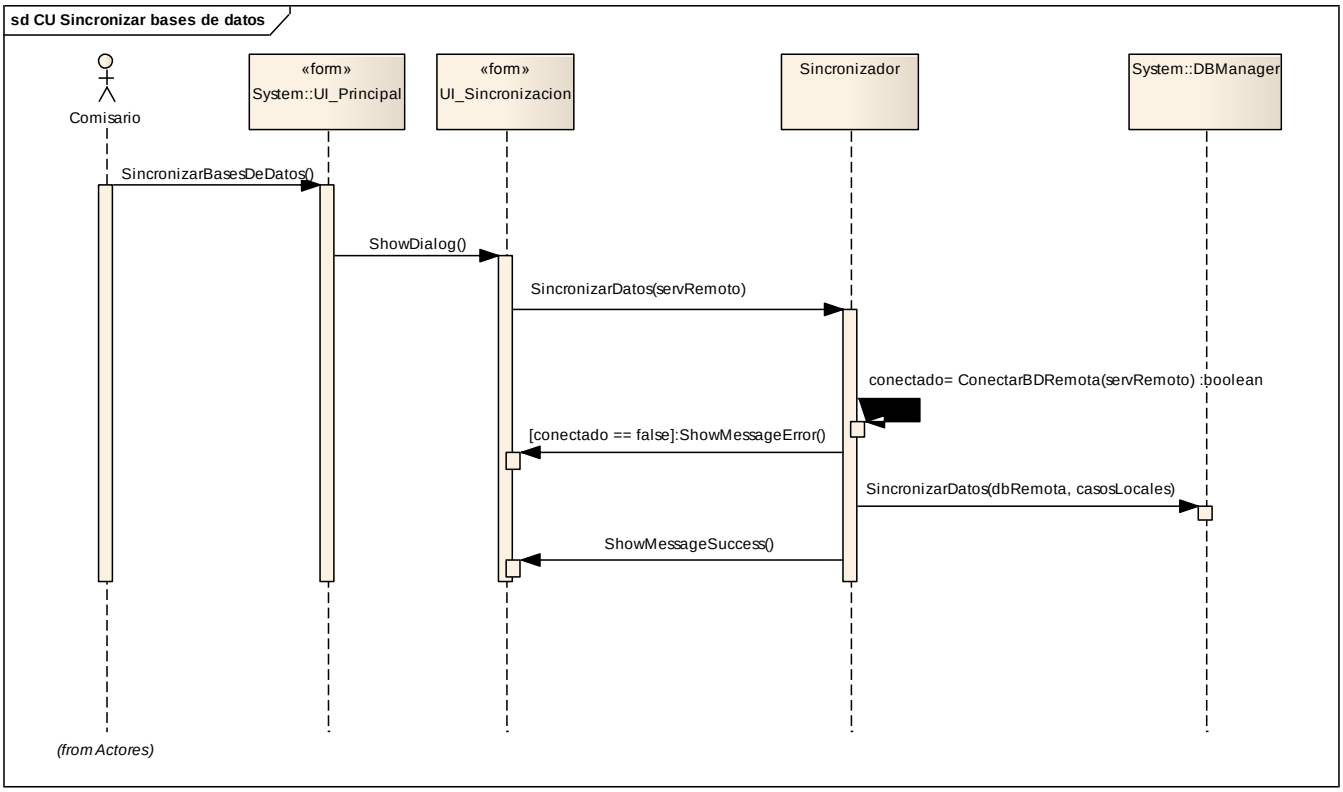
CU-2 Comparar Caso







CU-7 Sincronizar bases de datos



Anexo 5. Descripción de las clases de diseño

Nombre: Conjunto	
Tipo de clase: Entidad	
Atributo	**Tipo**
valores	float []
Cantidad	int
Para cada responsabilidad:	
Nombre:	Conjunto()
Descripción:	Constructor de la Clase.
Nombre:	Adicionar(float v)
Descripción:	Adicionar un elemento al arreglo de valores.
Nombre:	numeroDeOcurrencias(float n, float list)
Descripción:	Devuelve la cantidad de ocurrencias de ‘n’ en ‘list’.
Nombre:	Intersection(Conjunto s)
Descripción:	Devuelve la intersección entre un conjunto dado y los datos de la clase.

Nombre: Marcador	
Tipo de clase: Entidad	
Atributo	**Tipo**
id	string
Alelos	Conjunto
Para cada responsabilidad:	
Nombre:	Marcador()
Descripción:	Constructor de la clase.
Nombre:	ID()

Descripción:	Método de acceso para obtener el valor del atributo ‘id’.
Nombre:	Alelos()
Descripción:	Método de acceso para obtener el Conjunto de alelos.

Nombre: PerfilADN	
Tipo de clase: Entidad	
Atributo	**Tipo**
cantidadMarcadores	int
marcadores	Marcadores []
Para cada responsabilidad:	
Nombre:	PerfilADN()
Descripción:	Constructor de la clase.
Nombre:	AdicionarMarcador(Marcador m)
Descripción:	Método para adicionar un marcador al arreglo.

Nombre: Caso	
Tipo de clase: Entidad	
Atributo	**Tipo**
id	string
nombre	string
municipio	string
provincia	string
descripción	string
perfiles	PerfilADN[]
Para cada responsabilidad:	
Nombre:	Caso()

Descripción:	Constructor de la clase.
Nombre:	AdicionarPerfilADN(PerfilADN perfil)
Descripción:	Método para adicionar un perfil de ADN al arreglo.
Nombre:	Descripción()
Descripción:	Propiedad de acceso y modificación del atributo ‘descripción’.
Nombre:	ID()
Descripción:	Propiedad de acceso y modificación del atributo ‘id’.
Nombre:	Municipio()
Descripción:	Propiedad de acceso y modificación del atributo ‘municipio’.
Nombre:	Nombre()
Descripción:	Propiedad de acceso y modificación del atributo ‘nombre’.
Nombre:	NumeroDeAlelos()
Descripción:	Propiedad de acceso que devuelve la cantidad de alelos.
Nombre:	Perfiles()
Descripción:	Propiedad de acceso y modificación del atributo ‘perfiles’.
Nombre:	Provincia()
Descripción:	Propiedad de acceso y modificación del atributo ‘provincia’.

Nombre: DBManager	
Tipo de clase: Controladora	
Atributo	**Tipo**
connectionString	string
Para cada responsabilidad:	
Nombre:	ConnecionString
Descripción:	Propiedad de acceso y modificación del atributo ‘connectionString’.
Nombre:	ConectarBD()

Descripción:	Establece la conexión con la base de datos local.
Nombre:	BuscarCaso(string id)
Descripción:	Devuelve los datos de un caso criminalístico dado su identificador.
Nombre:	InsertarCaso(Caso c)
Descripción:	Adiciona en la base de datos la información de un nuevo caso.
Nombre:	ObtenerPerfiles()
Descripción:	Devuelve un listado con los perfiles de ADN existentes en la base de datos.
Nombre:	ObtenerCasos(DateTime ultSinc)
Descripción:	Devuelve los casos comprendidos en el rango de la fecha pasada por parámetro y el día que se realiza la operación.
Nombre:	SincronizarDatos(string dbRemota, Caso[] casos)
Descripción:	Realiza la sincronización de los datos locales con los de la base de datos remota.

Nombre: CC_GestionarCaso	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	InsertarCaso(Caso caso)
Descripción:	Inserta un caso en la base de datos.
Nombre:	BuscarCaso(string idcaso)
Descripción:	Devuelve los datos de un caso dado su identificador.
Nombre:	ModificarCaso(Caso caso)
Descripción:	Modifica en la base de datos los datos del caso pasado por parámetros.
Nombre:	EliminarCaso(string idCaso)
Descripción:	Elimina los datos de un caso dado su identificador.

Nombre: Sincronizador	
Tipo de clase: Controladora	
Atributo	**Tipo**
conectado	boolean
bdRemota	string
ultimaSincronizacion	DateTime
Para cada responsabilidad:	
Nombre:	SincronizarDatos()
Descripción:	Método para sincronizar los datos locales con los de un servidor remoto.
Nombre:	ConectarBDRemota(string bdRemota)
Descripción:	Devuelve true si se realiza la conexión con la base de datos remota.

Nombre: UI_Principal	
Tipo de clase: Interfaz	
Para cada responsabilidad:	
Nombre:	InsertarPerfil()
Descripción:	Opción que redirecciona a la interfaz UI_DatosCaso.
Nombre:	SincronizarBasesDeDatos()
Descripción:	Opción que muestra la interfaz UI_Sincronizacion.
Nombre:	CompararCaso()
Descripción:	Opción que muestra la interfaz UI_DatosCompararPerfil.

Nombre: UI_DatosCaso	
Tipo de clase: Interfaz	
Atributo	**Tipo**

casoTemp	Caso
Para cada responsabilidad:	
Nombre:	GuardarDatosCasoEnMemoria()
Descripción:	Almacena temporalmente los datos del caso a insertar en la variable casoTemp.
Nombre:	ShowDialog()
Descripción:	Muestra la interfaz de usuario UI_DatosCaso.

Nombre: UI_DatosPerfilADN	
Tipo de clase: Interfaz	
Para cada responsabilidad:	
Nombre:	GuardarDatosPerfil(Caso caso)
Descripción:	Invoca al método InsertarCaso(Caso caso) de la clase CC_GestionarCaso.
Nombre:	ShowDialog()
Descripción:	Muestra la interfaz de usuario UI_DatosPerfilADN.

Nombre: UI_Sincronizacion	
Tipo de clase: Interfaz	
Para cada responsabilidad:	
Nombre:	ShowDialog()
Descripción:	Muestra la interfaz de usuario UI_Sincronizacion.
Nombre:	ShowMessageError()
Descripción:	Muestra mensaje de error en caso de no realizarse la sincronización.
Nombre:	ShowMessageSuccess()
Descripción:	Muestra mensaje de sincronización realizada de manera satisfactoria.

Nombre: CC_GestorComparacion	
Tipo de clase: Controladora	
Atributo	**Tipo**
listaPerfiles	PerfilADN[]
conectadoDBRemota	boolean
Para cada responsabilidad:	
Nombre:	CompararCaso(PerfilADN perfilA, PerfilADN[] perfilesB)
Descripción:	Método para la comparación local de un perfil de ADN con una lista de posibles casos coincidentes.
Nombre:	CompararCaso(PerfilADN pelfilA)
Descripción:	Método para la comparación de un perfil de ADN contra los perfiles existentes en la base de datos local del sistema.
Nombre:	CompararCasoRemoto(string dbRemota, int prioridad, PerfilADN pelfilA)
Descripción:	Método para la comparación de un perfil de ADN contra los perfiles existentes en una base de datos remota con determinado nivel de prioridad.

Nombre: Comparador	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	VerificarSimilitud(PerfilADN perfilA, PerfilADN perfilB)
Descripción:	Devuelve el porciento de similitud entre dos perfiles de ADN.

Nombre: UI_DatosCompararPerfil	
Tipo de clase: Interfaz	
Atributo	**Tipo**
perfiles	PerfilADN[]

casosSimilares	Caso[]
Para cada responsabilidad:	
Nombre:	AlmacenarPerfiles(PerfilADN perfiles)
Descripción:	Método que guarda temporalmente los perfiles a comparar en la variable 'perfiles'.
Nombre:	IntroducirDatosPerfiles(PerfilADN[] perfil)
Descripción:	Método para introducir en el sistema los datos de los perfiles a comparar en el escenario Comparación 1:1.
Nombre:	PerfilAComparar(PerfilADN perfilA)
Descripción:	Método para introducir el perfil a comparar contra los restantes perfiles existentes en la base de datos (escenarios Comparación 1: n local y remota) o introducidos previamente (escenario Comparación 1:1).
Nombre:	MostrarDatos(Caso casoA)
Descripción:	Método para mostrar los datos del caso coincidente luego de realizada la comparación. Utilizado en el escenario Comparación 1:1.
Nombre:	MostrarDatos(Caso[] casos)
Descripción:	Método para mostrar los datos de los casos coincidentes luego de realizada la comparación. Utilizado en los escenarios Comparación 1: n local y remota.

Nombre: ProxyManager	
Tipo de clase: Controladora	
Atributo	**Tipo**
resultadoComparacion	Caso[]
Para cada responsabilidad:	
Nombre:	EnviarMensajeComparacion(PerfilADN perfilA, string destino, int prioridad)
Descripción:	Método que envía el perfil de ADN, con una prioridad definida, al servidor remoto donde se realizará la comparación.
Nombre:	EnviarConfirmacionDeRecibo(string direccion)

Descripción:	Método que envía la confirmación de recibo de los resultados al servidor remoto donde se realizó la comparación.

Nombre: ServerProxyManager	
Tipo de clase: Controladora	
Atributo	**Tipo**
casosSimilares	Caso[]
Para cada responsabilidad:	
Nombre:	EnviarResultadosComparacion(Caso[] casosSimilares, string direccion)
Descripción:	Método que envía los resultados de la comparación realizada al cliente que solicitó dicha comparación.
Nombre:	EnviarConfirmacionDeRecibo(string direccion)
Descripción:	Método que envía la confirmación de recibo del perfil de ADN a comparar.

GLOSARIO DE TÉRMINOS

Término	Definición
Alelos	Genes con la misma función pero diferentes formas de secuencia o repetición, por lo que tienen diferentes efectos.
ANSI	American National Standards Institute. Organización americana que establece las normas de las industrias de América mediante la coordinación de las normas internacionales y los requisitos de las solicitudes para el establecimiento de los estándares.
ARN	Acido Ribonucléico. Es un ácido nucleico formado por una larga cadena de nucleótidos. Se ubica en las células de tipo procariota y las de tipo eucariota. El ARN se define también como un material genético de ciertos virus y en los organismos celulares, molécula que dirige las etapas intermedias de la síntesis proteica.
Célula somática	Las células somáticas son aquellas que forman el conjunto de tejidos y órganos de un ser vivo, procedentes de células madre originadas durante el desarrollo embrionario.
Cromosoma	Cada una de las partes auto reproducibles que se hallan en el núcleo de la célula en número constante para cada especie. En las células de los seres animados el número de cromosomas es par (Diploide). En la célula humana hay 23 pares de cromosomas (total 46), de ellos, los dos que forman el par 23 se denominan cromosomas sexuales y se representan por las letras X e Y. Las células femeninas tienen dos cromosomas X (par XX) en tanto que las masculinas tienen uno X y otro Y (par XY). Los cromosomas están constituidos por el ADN y constan de GENES portadores de los factores determinantes de la herencia de caracteres. Cada cromosoma contiene miles de genes.
Electroforesis capilar	La electroforesis capilar es una técnica utilizada en bioquímica para separar las diferentes moléculas presentes en una disolución de acuerdo a la relación carga/tamaño de las mismas

Electroforesis en gel	La electroforesis en gel es un grupo de técnicas empleadas para separar moléculas basándose en propiedades como el tamaño, la forma o el punto isoeléctrico. Se utiliza generalmente con propósitos analíticos, pero puede ser una técnica preparativa para purificar moléculas parcialmente antes de aplicar espectrometría de masas, PCR, clonación o secuenciación de ADN.
EULA	En inglés End User License Agreement, es una licencia por la cual el uso de un producto sólo está permitido para un único usuario (el comprador).
Framework	En el desarrollo de software es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.
Gen	Parte de los cromosomas que contienen los factores hereditarios. Constituyen las diferentes partes de la hélice del ADN
HTTP	Hypertext Transfer Protocol. Protocolo de transmisión del hipertexto que sirve para incursionar en los sitios de WWW en Internet.
In vitro	(Latín: dentro del vidrio) se refiere a una técnica para realizar un determinado experimento en un tubo de ensayo, o generalmente en un ambiente controlado fuera un organismo vivo.
Licencia BSD	La licencia BSD es la licencia original de una distribución de Software: Berkeley Software Distribution, que acabó convirtiéndose en un derivativo de UNIX realizado por la conocida Universidad de California, Berkeley. La licencia BSD ha tenido dos formas principalmente: la clásica (con la cláusula de publicidad) y la actual (sin esa cláusula, desde Julio del 99). La cláusula de publicidad obligaba a trabajos que incluyeran el código licenciado BSD a publicar una nota tipo
Marcadores	Genes que sirven para establecer la individualización, en base al número de veces que se repiten. Pueden denominarse microsatélites o mini satélites según que estén formados por pocos (de 1 a 5) o por muchos (de 6 a 60) nucleótidos.

PDAs	Del inglés Personal Digital Assistant (Asistente Digital Personal), es un computador de mano originalmente diseñado como agenda electrónica (calendario, lista de contactos, notas y recordatorios) con un sistema de reconocimiento de escritura.
Stakeholders	Son todas aquellas personas u organizaciones que afectan o son afectadas por el proyecto, ya sea de forma positiva o negativa. Una buena planificación de proyectos debe involucrar la identificación y clasificación de los interesados, así como el estudio y la determinación de sus necesidades y expectativas.
TICs	Las tecnologías de la información y la comunicación son un conjunto de servicios, redes, software y dispositivos que tienen como fin la mejora de la calidad de vida de las personas dentro de un entorno, y que se integran a un sistema de información interconectado y complementario.
Triggers	Un trigger o un disparador en una base de datos es un evento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).