Universidad de las Ciencias Informáticas
Facultad 6

Título: "Componente visual para la representación de árboles genealógicos."

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autor(es): Omar Macias Llorente.

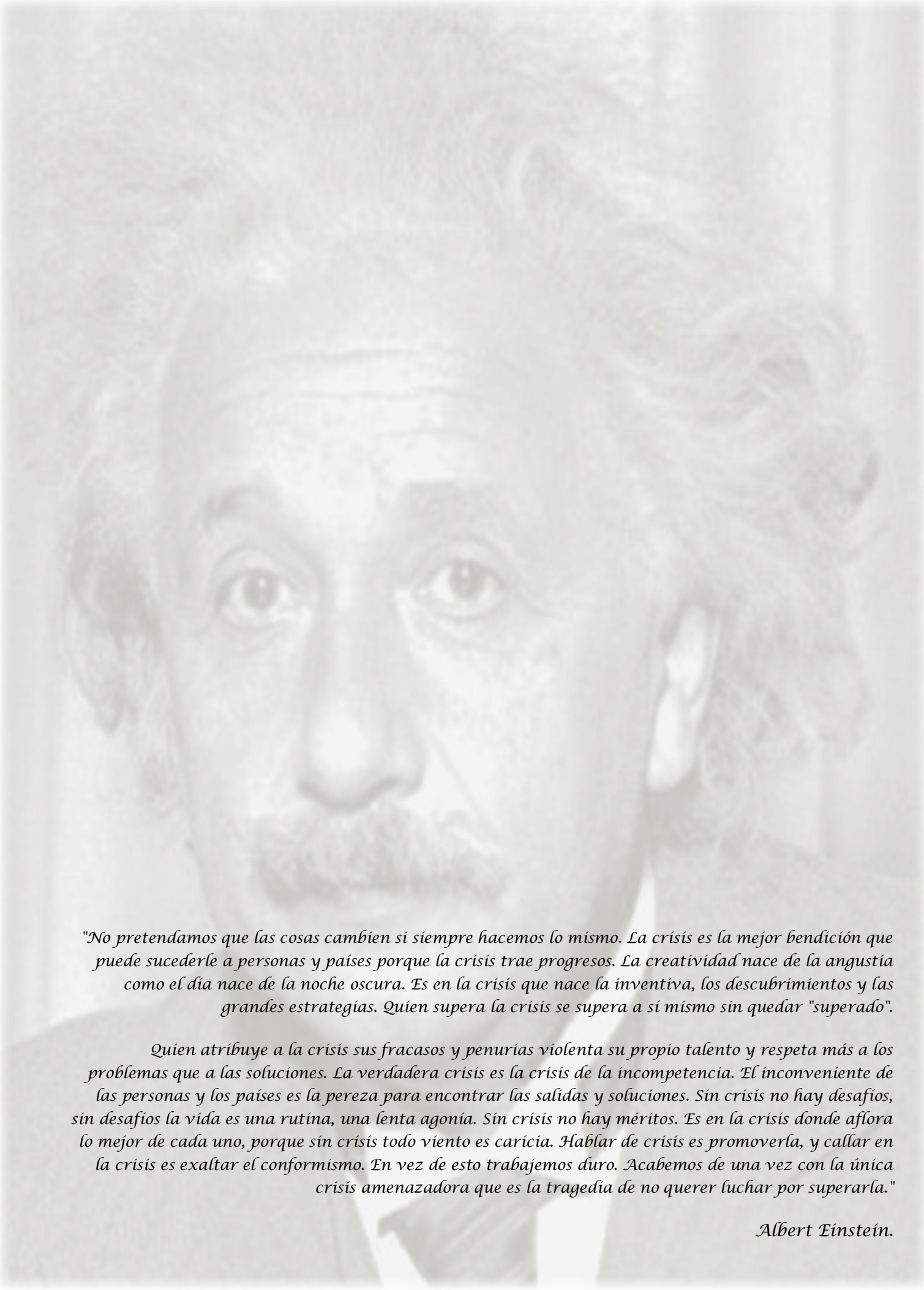
Tutor(es): Ing. Reynaldo Alvarez Luna.

Ing. Rayko Emilio Torres Cruz.

Co-Tutor(es): Ing. Yosvany Arrastia Machin

La Habana, Junio 2011

"Año 53 de la Revolución"



"No pretendamos que las cosas cambien si siempre hacemos lo mismo. La crisis es la mejor bendición que puede sucederle a personas y países porque la crisis trae progresos. La creatividad nace de la angustia como el día nace de la noche oscura. Es en la crisis que nace la inventiva, los descubrimientos y las grandes estrategias. Quien supera la crisis se supera a sí mismo sin quedar "superado".

Quien atribuye a la crisis sus fracasos y penurias violenta su propio talento y respeta más a los problemas que a las soluciones. La verdadera crisis es la crisis de la incompetencia. El inconveniente de las personas y los países es la pereza para encontrar las salidas y soluciones. Sin crisis no hay desafíos, sin desafíos la vida es una rutina, una lenta agonía. Sin crisis no hay méritos. Es en la crisis donde aflora lo mejor de cada uno, porque sin crisis todo viento es caricia. Hablar de crisis es promoverla, y callar en la crisis es exaltar el conformismo. En vez de esto trabajemos duro. Acabemos de una vez con la única crisis amenazadora que es la tragedia de no querer luchar por superarla."

Albert Einstein.

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Autor: _____

Omar Macias LLorente

Tutores: _____ _____

Ing. Reynaldo Alvarez Luna Ing. Rayko Emilio Torres Cruz

Co-Tutor: _____

Ing. Yosvany Arrastia Machin

DATOS DE CONTACTO

Autor: Omar Macias Llorente.
Universidad de las Ciencias Informáticas, Habana, Cuba.
ollorente@estudiantes.uci.cu

Tutor: Ing. Reynaldo Alvarez Luna
Universidad de las Ciencias Informáticas, Habana, Cuba.
rluna@uci.cu

Tutor: Ing. Rayko Emilio Torres Cruz
Universidad de las Ciencias Informáticas, Habana, Cuba.
retorres@uci.cu

Co-Tutor: Ing. Yosvany Arrastia Machin
Universidad de las Ciencias Informáticas, Habana, Cuba.
yarrastia@uci.cu

AGRADECIMIENTOS

Quiero agradecer a mis padres, mi hermano, mi esposa, tíos y tías y al resto de la familia por entregarme el amor más hermoso del mundo, por su preocupación, su confianza, su apoyo.

A Yanelis Benítez por ser otra madre para mí, por enseñarme tantas cosas en estos años, por ser excepcional, por su grandeza, su cariño y confianza.

A Yadrián Serrano por la ayuda incondicional, por su amistad.

Agradezco a mis tutores Reynaldo Alvarez Luna, Rayko Emilio Torres Cruz y Yosvany Arrastia Machín, por su ayuda, permitiéndome estar hoy aquí, contribuyendo a que sea posible este sueño.

A todos los amigos que me apoyaron tanto en el éxito de estos cinco años.

A los profesores que me forjaron como joven de estos tiempos, que contribuyeron a mi formación, mi educación.

En especial al siempre Comandante en Jefe Fidel, por darme la oportunidad de participar en este proyecto de crear una nueva sociedad, de haber sido parte de la UCI, de inculcarnos sentimientos tan humanos.

A todos los que me ayudaron a conquistar este título.

DEDICATORIA

Quiero dedicar este gigantesco sueño a mis padres Juan Macías Torres y Arminda Llorente Molina, que siempre confiaron en mí, me apoyaron y me dieron fuerzas para seguir; a ellos que son mi orgullo y lo más valioso en mi vida, el ejemplo a seguir cada día.

A mi hermano Oscar Macías Llorente por su entera confianza, por ser el mejor hermano del mundo, y porque este sueño es de los dos, hoy nos graduamos los dos.

A mi esposa Natacha Pérez Montano, lo mejor que me ha pasado en la vida, por su amor incondicional, por compartir conmigo tanta felicidad, por su cariño.

A mis tías y tíos que me han querido como otro hijo, por su apoyo, su ayuda, su cariño, su paciencia, su preocupación, porque han estado todo este tiempo conmigo.

A mis abuelos, que aunque hoy no viven, sé que estarían muy orgullosos de mí, por seguir el ejemplo de la familia, por ser un hombre de bien.

A mis primos y primas que nos queremos como si fuéramos hermanos.

A toda mi familia por ser una familia tan hermosa, donde el amor que nos tenemos "Es un amor para la Historia".

A mis compañeros de la vida, los que estuvieron ahí en todo momento, con su amistad, con la sonrisa agradable, con el abrazo y el beso de hermanos.

RESUMEN

La Universidad de las Ciencias Informáticas ha desarrollado una herramienta informática para los estudios genéticos de la sociedad cubana, llamada alasARBOGEN, la cual es usada por los genetistas en las consultas de nuestros centros de genética médica. Este software presenta un diseño característico de las herramientas médicas de representación de árboles genealógicos. Su interfaz gráfica es muy dinámica y sencilla permitiendo al usuario realizar disímiles acciones para crear un árbol genealógico, entiéndase adicionar algún familiar, eliminarlo, relacionarlo con otros individuos, etc. Gráficamente presenta problemas en la representación de los individuos por generaciones, provocando dificultades en el diagnóstico y retardando el proceso de toma de decisiones, a partir de incorrectas interpretaciones del diseño. En la actualidad, los estudios genéticos han determinado nuevas representaciones gráficas que permiten hacer análisis diferentes de los árboles familiares. El sistema alasARBOGEN no presenta este tipo de representaciones, de manera que no le aporta a los genetistas datos más específicos de un árbol familiar. El sistema no se encuentra bajo los estándares para la representación de árboles genealógicos. Estas limitaciones hacen al sistema alasARBOGEN, una herramienta con dificultades en la representación gráfica de los árboles genealógicos.

Para el desarrollo de esta investigación se realizó el estudio de los estándares y las principales herramientas de representación de árboles genealógicos. El mismo arrojó la necesidad de implementar la representación del árbol genealógico en su forma concéntrica cuadrada y la vista de descendientes horizontal, además de implementar un algoritmo para el ordenamiento del árbol familiar a partir de las características propias de un árbol genealógico. El desarrollo de este componente aportará mejoras gráficas al sistema, aportando a los genetistas representaciones gráficas de árboles familiares que le permitirán obtener datos específicos del árbol genealógico general, como la línea de descendencia y progenitores de un individuo determinado.

PALABRAS CLAVES

alasARBOGEN, componente de software, PSTF, visualización de árboles genealógicos.

TABLA DE CONTENIDOS

AGRADECIMIENTOS I

RESUMEN III

INTRODUCCIÓN 1

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS 5

1.1. Conceptos Fundamentales 5
1.1.1. Árbol Genealógico. 5
1.1.2. Componentes de software. 5
1.2. Elementos teóricos sobre los árboles genealógicos 7
1.2.1. Estándar para la representación gráfica de árboles genealógicos. 7
1.2.2. Criterios estéticos para representar gráficamente árboles genealógicos. 7
1.2.3. Clasificación según las relaciones que representa. 8
1.2.4. Clasificación según la forma del esquema. 9
1.3. Herramientas para la representación gráfica de árboles genealógicos 11
1.3.1. Herramientas Privativas. 12
1.3.2. Herramientas Libres. 13
1.4. Herramientas y tecnologías para el desarrollo 14
1.4.1. Lenguaje de programación. 14
1.4.2. Entorno Integrado de Desarrollo. 16
1.4.3. Herramienta de Modelado. 17
1.4.4. Lenguaje de modelado 20
1.4.5. Metodología de Desarrollo de Software. 21
1.5. Conclusiones parciales 22

CAPÍTULO 2: DISEÑO DEL COMPONENTE 23

2.1. Descripción del componente a desarrollar 23
2.2. Especificación de los Requisitos del Componente. 23
2.2.1. Requisitos Funcionales del Componente. 24
2.2.2. Requisitos No Funcionales del Componente. 24
2.3. Definición de los Casos de Uso del Sistema 25

2.3.1. Actor del Sistema. 25
2.3.2. Diagrama de Casos de Uso del Sistema. 26
2.3.3. Descripción de los Casos de Uso del Sistema. 26
2.4. Arquitectura Utilizada 30
2.4.1. Estilo Arquitectónico Utilizado 31
2.5. Patrones de Diseño Utilizados 33
2.6. Diagramas de Clases del Diseño 35
2.6.1. Diagrama de Clases del Diseño del CU Ordenar Árbol Genealógico. 35
2.6.2. Diagrama de Clases del Diseño del CU Representar Vistas. 36
2.6.3. Diagrama de Clases del Diseño del CU Visualizar Leyenda. 37
2.7. Diagramas de Secuencia 39
2.8. Conclusiones parciales 41

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS 42

3.1. Diagrama de Componentes 42
3.2. Resultados fundamentales de la implementación. 44
3.3. Pruebas 46
3.3.1. Diseño de casos de prueba. 47
3.4. Conclusiones parciales 53

CONCLUSIONES 54

REFERENCIAS BIBLIOGRÁFICAS 59

ANEXOS 61

GLOSARIO DE TÉRMINOS 68

INTRODUCCIÓN

Desde inicios de los 90 comenzaron los estudios del lenguaje genético para lograr una nomenclatura en el lenguaje de los genetistas. Después de cinco años de propuestas de estándares, símbolos y abreviaciones, es en 1995 que el "*Pedigree Standardization Task Force*" (PSTF) de un grupo de investigadores del Comité de la Sociedad Nacional de Consejeros Genéticos de los Estados Unidos, define el primer estándar para la representación gráfica de árboles genealógicos. Este estándar define la simbología médica a utilizar para la visualización de árboles genealógicos en las herramientas informáticas de estudios genéticos. Esta nomenclatura unifica los estándares propuestos por muchos especialistas, de manera que su uso adecuado permitirá el intercambio de dichos árboles entre diferentes especialistas que traten un caso determinado y puedan entender perfectamente la simbología utilizada.

Para la representación gráfica de los árboles genealógicos se proponen criterios estéticos, que garantizan una correcta visualización e interpretación de los datos. Un árbol construido bajo el cumplimiento de estos criterios garantizará la representación espacial correcta del mismo, eliminando posibles ambigüedades que pueden provocar falsas interpretaciones.

De igual manera existen diferentes visualizaciones para representar gráficamente árboles genealógicos, las cuales se fundamentan en las diferentes interpretaciones que le puedan aportar a los genetistas en sus estudios en dependencia del tipo de información que estas brindan. Nuevas representaciones gráficas mejoran la calidad de los estudios genéticos y facilitan la toma de decisiones de los genetistas.

Desde el triunfo revolucionario del primero de Enero de 1959, los servicios de salud pública se convirtieron en una tarea de primer orden para el país. El 5 de agosto de 2003 nuestro comandante en jefe, Fidel Castro Ruz, inaugura el Centro Nacional de Genética Médica (CNGM), con el propósito de ampliar las investigaciones científicas, mejorar los tratamientos y llevar a cabo un monitoreo extensivo de todas las enfermedades de origen genético.

Es de interés especial en el trabajo de los genetistas del CNGM la representación gráfica de árboles genealógicos de los pacientes que se presentan a la consulta, para lograr un estudio profundo de las

enfermedades genéticas en nuestra población. Se hace necesario para ello, la utilización de un sistema informático que manejara eficientemente este tipo de información, entiéndase almacenamiento, análisis y recuperación; con el fin de crear árboles genealógicos que reflejen las relaciones entre los individuos de determinada familia, y que además gestionara datos como: enfermedades padecidas, relaciones conyugales, entre otros. Una de las soluciones puestas en práctica en el CNGM para la representación de árboles genealógicos, fue el uso del *Cyrillic*. Este software es una herramienta propietaria para la representación de árboles genealógicos, cuyo costo por copia es inaccesible para nuestro país, pues la licencia por cada estación de trabajo en la que se vaya a utilizar valora sobre los $770.00 USD. Su adquisición constituye un gasto financiero considerable que el país no puede cubrir, por las condiciones económicas actuales.

A partir de las deficiencias encontradas en el CNGM, se decidió en conjunto con la Universidad de las Ciencias Informáticas (UCI), desarrollar una aplicación (alasARBOGEN) que cumpliera con los requisitos que se necesitaban para los estudios genéticos en el país. Actualmente esta herramienta se utiliza en nuestros centros de salud en la representación del árbol genealógico.

Una vez puesto en funcionamiento se detectó que alasARBOGEN no cumplía con todas las normas del estándar del PSTF. Esto podía provocar interpretaciones incorrectas de la información médica y genética de los pacientes y sus familiares, lo que incide en la veracidad de los resultados de los estudios genéticos. El PSTF propone el uso de leyendas para aclarar la simbología médica, lo cual facilitaría la comunicación entre investigadores involucrados en el estudio de la familia, cosa que el sistema implementado no contiene. Para la representación gráfica de los árboles genealógicos, alasARBOGEN sólo hace uso de la representación jerarquía clásica vertical, presentando problemas en la organización de los árboles y de los individuos por generaciones, por lo que no cumple con criterios estéticos fundamentales que han sido adoptados igualmente como convenio entre los especialistas. Este sistema no será completo mientras no permita la representación gráfica de los árboles genealógicos de manera correcta, siguiendo estándares para su representación, por lo que dificulta la comunicación y el intercambio científico del CNGM con otros centros del mundo. Además, no permite la interpretación de los datos desde otras visualizaciones o representaciones como: la concéntrica circular y la concéntrica cuadrada, las cuales podrían aportar otras interpretaciones de los datos. Estas necesidades existentes hacen del sistema alasARBOGEN, una aplicación que presenta carencias en la representación gráfica de los árboles genealógicos.

Teniendo en cuenta esta problemática se ha determinado como **problema de la investigación:** La representación gráfica de árboles genealógicos en alasARBOGEN presenta limitaciones que inciden negativamente en la interpretación de los datos genealógicos.

Se define como **objeto de estudio:** la representación gráfica de árboles genealógicos.

El **campo de acción:** La implementación de componentes de software para la representación de árboles genealógicos en alasARBOGEN.

Para darle cumplimiento al problema científico expuesto, se define como **objetivo general** de la investigación, desarrollar un componente de software para la estructuración visual de árboles genealógicos en el sistema alasARBOGEN, que favorezca la interpretación de los datos genealógicos desde diferentes perspectivas visuales. Para darle cumplimiento al objetivo general se plantean como **objetivos específicos:**

- Identificar las normas para la representación gráfica de árboles genealógicos.
- Diseñar las clases necesarias para la modelación del componente visual para la representación de árboles genealógicos.
- Implementar el componente visual para la representación de árboles genealógicos.

En función de los objetivos específicos enunciados, se definen como tareas de la investigación:

- Análisis del proceso de representación gráfica de árboles genealógicos.
- Estudio de las diferentes formas de organización jerárquica para árboles genealógicos.
- Definición de las diferentes formas de organización jerárquica para árboles genealógicos.
- Definición de los requerimientos del componente visual para la representación de árboles genealógicos.
- Identificación del formato para los datos genealógicos.
- Definición de la arquitectura del componente visual para la representación de árboles genealógicos.
- Realización de los diagramas de clases del diseño necesarios para la implementación del componente visual para la representación de árboles genealógicos.
- Realización de los diagramas de secuencia.
- Realización del diagrama de componentes.

- Implementación de los componentes definidos.
- Integración a alasARBOGEN para mejorar la capacidad de organización del sistema y sus componentes visuales.
- Realización de pruebas funcionales.

Aporte práctico esperado del trabajo: Componente visual para la representación de árboles genealógicos. Permitirá la integración a alasARBOGEN y aportará al sistema:

- Nuevas formas de visualización: concéntrica cuadrada y descendiente horizontal.
- Leyenda para la simbología utilizada en los árboles genealógicos.
- Mejoras a la representación jerárquica clásica vertical implementada actualmente en el sistema.

El presente trabajo estará estructurado por 3 capítulos. A continuación se describe de manera breve el contenido de cada uno de ellos.

Capítulo 1: Fundamentos Teóricos: En este capítulo se realizará un estudio de las características de los árboles genealógicos. Se abordará el estado actual de las aplicaciones para representar gráficamente árboles genealógicos en el mundo y en nuestro país. Se incluye además un estudio y selección de las metodologías, herramientas y tecnologías que se van a utilizar en el desarrollo del componente, atendiendo a las tendencias actuales en el desarrollo de software en el mundo.

Capítulo 2: Diseño del Componente: En el presente capítulo se definirán los requisitos del componente a implementar. Se explica la arquitectura utilizada así como los principales patrones de diseño empleados. Se definirán los Casos de Uso así como sus descripciones. Se realizan los diagramas de clases del diseño y los diagramas de secuencia, describiendo como implementar, enfocado a como el componente cumple los objetivos planteados en los requisitos funcionales y no funcionales.

Capítulo 3: Implementación y Prueba del Componente: En este capítulo se describe como fue implementado el componente. Se realiza el diagrama de Componentes. Se muestran los principales resultados de la implementación, dígase funcionalidades y clases importantes. Se describen además las pruebas funcionales realizadas al componente una vez integrado a alasARBOGEN.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS

El presente capítulo aborda la fundamentación teórica de la visualización de árboles genealógicos. Se identifican a los gráficos como su representación fundamental para el trabajo de genetistas. Para ello se da una descripción de los principales conceptos asociados al dominio del problema. Además, se hace un estudio del estado del conocimiento de las técnicas, tendencias, metodologías, herramientas informáticas y tecnologías, usadas en nuestro país y en el mundo para la representación gráfica de árboles genealógicos. También se seleccionan las herramientas y tecnologías para desarrollar el componente visual, así como la arquitectura base.

1.1. Conceptos Fundamentales

1.1.1. Árbol Genealógico.

El árbol genealógico es una representación gráfica que expone los datos genealógicos de un individuo en una forma organizada y sistemática, sea en forma de árbol o tabla. Puede ser ascendente, es decir, que expone los antepasados o ancestros de un sujeto, o puede ser descendente, que expone todos los descendientes del sujeto. Para realizar un árbol genealógico es necesario, primeramente, haber realizado una investigación genealógica o genealogía del individuo. (1)

Los árboles genealógicos son fundamentales en la historia clínica de un paciente. De igual manera establecen un patrón de herencia frente a una enfermedad genética dada. Brindan la posibilidad de identificar personas en riesgo e identificar desordenes genéticos en pacientes.

1.1.2. Componentes de software.

El desarrollo de un sistema basado en componentes de software permite su reutilización en otros entornos o aplicaciones. El objetivo fundamental del desarrollo de un componente de software es satisfacer las demandas de mantenimiento y soporte del sistema una vez implantado a través de la actualización, mejora o incluso nuevo acoplamiento de componentes en un tiempo aceptable.

Un componente de software es una unidad de composición con interfaces contractualmente especificadas y explícitas sólo con dependencias dentro de un contexto. Un componente de software puede ser desplegado independientemente y es sujeto a la composición de terceros. (2)

- **Características claves para que un elemento pueda ser catalogado como componente**: (2)

Identificable: Debe tener una identificación que permita acceder fácilmente a sus servicios y que permita su clasificación.

Auto contenido: Un componente no debe requerir de la utilización de otros para finiquitar la función para la cual fue diseñado.

Puede ser reemplazado por otro componente: Se puede reemplazar por nuevas versiones u otro componente que lo reemplace y mejore.

Con acceso solamente a través de su interfaz: Debe asegurar que estas no cambiarán a lo largo de su implementación.

Sus servicios no varían: Las funcionalidades ofrecidas en su interfaz no deben variar, pero su implementación sí.

Bien documentado: Un componente debe estar correctamente documentado para facilitar su búsqueda, actualización o integración con otros.

Es genérico: Sus servicios deben servir para varias aplicaciones.

Reutilizado dinámicamente: Puede ser cargado en tiempo de ejecución en una aplicación.

Independiente de la plataforma: Independiente del Hardware, Software y Sistema Operativo.

El sistema alasARBOGEN cuenta con una arquitectura que permite la integración de componentes al diseño de la aplicación. La región seleccionada en la figura del anexo 7, muestra el componente “Visor de Propiedades” que se encuentra integrado al sistema. El componente a desarrollar estaría integrado en las regiones determinadas en las figuras de los anexos 8 y 9.

1.2. Elementos teóricos sobre los árboles genealógicos

1.2.1. Estándar para la representación gráfica de árboles genealógicos.

En 1995 el equipo de trabajo para la estandarización de árboles genealógicos (PSTF siglas en inglés), actualmente llamado grupo de trabajo para la estandarización de árboles genealógicos (PSWG siglas en inglés), determinó un estándar para la representación gráfica de los árboles genealógicos.

Al estandarizar la nomenclatura de un árbol genealógico se lograba globalizar el lenguaje clínico y científico para lograr el entendimiento de los genetistas sobre los datos genéticos del paciente. Con este estándar se reducían las posibilidades de interpretaciones erradas en el diagnóstico.

Tras estudios realizados entre junio de 2002 y junio de 2007 el PSWG propone nuevos cambios al estándar propuesto en 1995. La nueva nomenclatura elimina algunas ambigüedades encontradas en la primera. La imagen del anexo 5, representa un árbol dibujado bajo este estándar establecido por el PSWG, como muestra de la forma de representación de las relaciones y la simbología de los individuos.

Como se puede apreciar en el anexo # 6, el estándar propone el uso de leyendas, lo que sería de mucha importancia para el entendimiento entre los diferentes médicos que estudien un caso determinado. Es de mucha importancia que los árboles diseñados por determinado genetista sean entendibles por otros colegas de trabajo.

1.2.2. Criterios estéticos para representar gráficamente árboles genealógicos.

De igual manera han sido definidos criterios estéticos para graficar árboles genealógicos, con el objetivo de guiar al diseñador para lograr que la estructura del árbol quede ordenada y visualizada correctamente, sin ambigüedades. Estos garantizan que no existan colisiones o solapamientos entre los elementos gráficos y que los mismos queden ubicados coherentemente según su nivel y sus relaciones.

Para representar correctamente árboles genealógicos se plantean los siguientes requisitos estéticos: (3)

- El diseño debe mostrar la estructura jerárquica del árbol, es decir, la coordenada ‘X’ de un nodo está dada por su nivel.

- Los bordes no se cruzan entre sí y los nodos en el mismo nivel deben guardar una distancia mínima horizontal.
- El dibujo de un sub-árbol no depende de su posición en el árbol, es decir, los sub-árboles isomorfos se dibujan de forma idéntica a la de su progenitor.

Si el árbol a graficar es ordenado, se deben seguir los siguientes requisitos estéticos: (3)

- El orden de los hijos de un nodo se muestra en el dibujo.
- El algoritmo de representación gráfica trabaja de forma simétrica, es decir, el plano de la reflexión de un árbol es el reflejo del dibujo del árbol original.

1.2.3. Clasificación según las relaciones que representa.

Un árbol genealógico puede representar relaciones familiares más allá de las genealógicas, es decir, puede mostrar además de los antepasados todos los otros familiares. O puede representar solo unas relaciones concretas:

- **Parentescos**: Muestra las relaciones familiares, tanto consanguíneas como adopciones y políticas directas.
- **Ascendente**: Muestra solo los parentescos ascendentes (padre/madre, tíos/as, abuelos/as, tíos-abuelos/as, bisabuelos/bisabuelas...)
- **Descendente**: Muestra solo los parentescos descendentes (hijo/a, yerno/nuera, nieto/nieta ...)

Estos árboles genealógicos pueden llegar a ser extremadamente extensos y complejos. Así cada uno de estos árboles genealógicos puede concretarse, el autor puede reducir sus ramas por grados de relación o siguiendo los estándares genealógicos.

- **Costados**: Muestra las relaciones directas (no muestra a tíos, tíos abuelos, etc.). Se expande de forma exponencial en cada generación, mostrando padre y madre, cuatro abuelos/as, ocho bisabuelos/as.
- **Agnaticia**: (línea masculina de sangre o linaje): Esquema que muestra una línea directa solo entre varones (en nuestro caso, normalmente, la que sigue al primer apellido) hasta en la actualidad que los padres pueden decidir modificar el orden de los apellidos.

- **Cognaticia**: (línea femenina o de ombligo): Esquema que muestra una línea directa solo entre hembras. Si eres mujer y quieres realizarte un estudio genealógico genético es la única línea de la que podrán informarte.
- **Primogenitura**: Como la Agnaticia pero siguiendo únicamente la línea de los primeros varones (común en los árboles genealógicos descendientes).

Un árbol genealógico de parentescos bien completo ayudará a tener una visión general y será útil para marcar los objetivos de la investigación pero no para realizar la propia investigación. Para iniciar la investigación genealógica es recomendable acotar la búsqueda de forma agnaticia o cognaticia.

1.2.4. Clasificación según la forma del esquema.

Un árbol genealógico no tiene por qué tener forma ni aspecto de árbol real, existen muchas formas de estructura y aspectos posibles. Algunas formas y estilos serán más aconsejables o solo posibles según el contenido que se desee mostrar.

Todos los **genogramas** muestra en desgloses los contenidos, nos muestran elementos que contienen más elementos y estos a su vez contienen otros tantos y así de forma continua mostrándolos en niveles.

Un elemento sería por ejemplo un padre y una madre que contienen 4 elementos que serían sus hijos y estos a su vez contienen otros elementos que serían los hijos de estos (nietos de los primeros). Esos estarían representados en 3 niveles que coinciden con el número de generaciones que representamos (abuelos, hijos y nietos). Aquí se muestra unos ejemplos de árboles genealógicos (o genogramas) clasificados en dos amplios grupos.

- **Árbol genealógico clásico.**

Adecuado para todo tipo de contenido, es el más utilizado para el de "Parentescos" y estudios genéticos. A partir del protagonista del árbol genealógico crece en una dirección. Este genograma puede mostrarse de forma Horizontal (las generaciones las tendríamos en el eje horizontal o "X") o de forma Vertical (las generaciones se muestran en el eje vertical o "Y"). A su vez estos pueden mostrarse en una tabla o en esquema (formas geométricas unidas por líneas).

Los horizontales suelen representarse mostrando el crecimiento de izquierda a derecha quedando los antepasados más alejados al protagonista del árbol genealógico en el lado derecho. Los verticales se representan con el crecimiento de la parte inferior a la superior quedando los antepasados más alejados en la parte más alta o a la inversa de superior a inferior quedando en la parte superior el protagonista del árbol genealógico o sus descendientes.

Ejemplo de árbol genealógico clásico horizontal en tabla de crecimiento de izquierda a derecha (solo de representación consanguínea), (Anexo 1).

Ejemplo de árbol genealógico clásico vertical en esquema de crecimiento de inferior a superior (Anexo 2).

- **Árboles genealógicos Concéntricos.**

Este tipo de árbol genealógico es adecuado para contenido descendente o ascendente. El protagonista del árbol genealógico tiene que ser el inicio del genograma. El protagonista del árbol genealógico "preside" el centro de la figura geométrica y a su alrededor van creciendo sus descendientes o ascendientes. La figura geométrica más habitual para este tipo de esquemas es la redonda pero también pueden ser utilizadas otras.

Figura cuadrada:

Adecuado solo para generaciones ascendentes y de costado, solo representa las líneas directas, no muestra a hermanos, tíos y otros. Se destacan por la facilidad con que se puede seguir la serie de antepasados de una persona por determinada línea, o los sucesores de cualquiera de los miembros del linaje. Las genealogías rectangulares, son utilísimas para recoger datos, por la gran cantidad de espacio que puede dedicarse a cada ascendiente. Ejemplo (Anexo 3).

Figura circular:

En este tipo de árbol genealógico las generaciones están contenidas en anillas del círculo. El dibujo puede realizarse solo con secciones de estas anillas o con líneas (ramas). Modernamente, se estima que tales esquemas genealógicos circulares son realmente otra simbolización del árbol natural, visto a través de un corte horizontal del tronco. En la superficie de ese corte se observa, por círculos, que de la médula van a la periferia, la edad y desarrollo del árbol, y el nacimiento sucesivo de sus ramas, de

análoga manera lo que sucede en nuestra vida, en que al padre, le rodean los hijos, y en círculos, cada vez más amplios, sus nietos, sus bisnietos, y demás generaciones de descendientes.

Árbol genealógico Divergente: Adecuado para cualquier representación de descendentes. Visualiza las generaciones por sectores, de inspiración zodiacal, y que es a modo de una rueda genealógica. Las generaciones en forma descendiente se desarrollan, cada una, dentro de un sector circular. Muy apto para comprender muchas de estas y hacer árboles de descendientes que muestran de una manera total los parentescos entre los individuos del linaje. El centro lo ocupa un antiguo ascendiente común, las generaciones están dispuestas sobre circunferencias concéntricas, y en cada línea radical que de ellas sale se describen, a uno y otro lado, el nombre del descendiente. Ejemplo (Anexo 4).

Árbol genealógico Convergente: Adecuado solo para generaciones ascendentes y de costado, solo representa las líneas directas, no muestra a hermanos, tíos. Por generaciones concéntricas, sirve para árboles de costados; al centro aparece el sujeto genealógico, y ocupando un semicírculo, los ascendientes paternos, y en otro los maternos, que en su desarrollo quedan completamente separados. Ejemplo (Anexo 5).

Para el dibujo del árbol genealógico los expertos y profesionales utilizan una gran variedad de tipos de líneas, símbolos y colores para mostrar el tipo de relaciones. Con estos símbolos con un solo vistazo podemos saber muchas cosas, como divorcios, adopciones e incluso relaciones conflictivas entre familiares.

1.3. Herramientas para la representación gráfica de árboles genealógicos

Como parte de la investigación científica fueron estudiados los sistemas más conocidos en la representación de árboles genealógicos existentes en la actualidad. La gran mayoría de estos sistemas están orientados a la construcción del árbol familiar, llegando a hacer un análisis y un seguimiento de las enfermedades genéticas de un individuo de la familia. Estos son software que en la mayoría de las ocasiones se difunden por internet y están siempre disponibles en la web. Están basados en la construcción de árboles familiares, brindando la posibilidad de añadirles fotos personales y una diversidad de datos, según las posibilidades de la aplicación. A continuación, se hará un estudio de algunos ejemplos de los sistemas existentes.

1.3.1. Herramientas Privativas.

- **GenoPro**

Es una aplicación para dibujar árboles familiares y genogramas. Es un software para graficar árboles genealógicos que ha tenido gran aceptación en el mundo y se encuentra disponible en 25 idiomas y ha sido adquirido por 170 países. Permite construir el árbol genealógico con un asistente familiar que brinda la posibilidad de agregar o eliminar hijos con gran sencillez. (4)

GenoPro permite la posibilidad de dibujar una familia sin importar cuán compleja. Cuenta con un rango completo de funcionalidades para hacer análisis y manejar escenarios complejos, y una vez terminado, brinda la posibilidad de hacer salvas en archivos que se pueden cargar en una gran variedad de aplicaciones de Windows y ayuda a crear un CD con el árbol familiar.

- **BitGen II**

BitGen v2.0 es otra de las aplicaciones que se encuentran disponibles en el mercado para la confección de Árboles Genealógicos de una familia, partiendo de datos previamente almacenados relativos a esta. Es una aplicación en español, que tiene como objetivo principal, facilitar la elaboración de árboles genealógicos en una computadora. (5)

La interfaz de BitGen II para la confección de un árbol genealógico presenta cuatro pergaminos de tamaño DIN-A4, bordeados por una pareja de reglas para medir adecuadamente la posición de cada elemento gráfico. Además, contiene una barra de herramientas y los controles de movimientos, desde donde se pueden indicar todas las funciones que se quieren realizar. (6)

BitGen II es la segunda versión de un software al que aún le falta dinamismo, rapidez, intuición y un funcionamiento gráfico más cercano al de aplicaciones destinadas a trabajar en entornos operativos Windows. (6)

- **Cyrillic**

Posee todas las características necesarias para dibujar líneas genéticas. Es el más completo, funcional y fácil de usar. Es un software diseñado para el manejo de datos genéticos en el sistema operativo Windows. Todas las herramientas están dispuestas de forma que usted pueda, rápida y fácilmente, manejar los datos de los que disponga. (7)

Sus interfaces están diseñadas de forma agradable y permiten moverse sin ningún esfuerzo de una fase del proceso a otra. La implantación de numerosas barras de herramientas especiales y cajas de diálogo hacen que su manejo sea muy intuitivo, pero si usted necesitara ayuda, todas las respuestas a sus cuestiones están disponibles instantáneamente a través de una completa ayuda en línea. (7)

Cyrillic puede almacenar un elevado rango de datos para cada individuo, incluyendo información personal, clínica y genética. Todos los datos son accesibles a través de una caja de diálogo y los cambios que hagamos en ellos son reflejados automáticamente en el dibujo del árbol genético. (7)

Como características importantes presenta:

- Actualización automática de los datos mientras se está dibujando.
- Perfeccionamiento del manejo para casos de gemelos y embarazos múltiples.
- Cálculo de riesgos para enfermedades familiares, permitiendo hacer el cálculo de riesgo de enfermedad para un individuo dentro de una familia donde exista una determinada enfermedad.
- Automatización del proceso de construcción de los árboles genealógicos, lo cual es necesario en el trabajo diario de los genetistas.

El estudio de algunas herramientas privativas arrojó que eran muy completas como software médico, pero no era posible reutilizar su código fuente ni sus componentes debido a que no son de código abierto. Sus valores monetarios por adquisición de una copia son inaccesibles para nuestro país, convirtiéndose en un gasto financiero considerable.

1.3.2. Herramientas Libres.

Se realizó el estudio de algunas herramientas para la representación gráfica de árboles genealógicos, entre ellas: *ScionPC*, *KinShip* 1.1.0, *HaploPainter* 1.043, *FamilyTreeBuilder* y *GenealogyJ*. Estas herramientas permiten graficar el árbol generacional de individuos. Reciben y gestionan datos de los mismos, a través de interfaces que importan y exportan en datos genéticos reconocidos. Son libres de uso, pero no de código abierto. Están desarrolladas en lenguaje C y C++, no permitiendo que se puedan integrar a alasARBOGEN. Algunas utilizan como interfaz la consola de Windows, lo que hace que el trabajo de los genetistas se haga más complejo al utilizar estas herramientas. En especial el GenealogyJ, presenta una interfaz gráfica adecuada para estudios médicos, con funcionalidades bien definidas y correctamente diseñadas.

1.4. Herramientas y tecnologías para el desarrollo

1.4.1. Lenguaje de programación.

Después de analizados lenguajes de programación como C++, C# y Java se llega a la conclusión que el más adecuado para el desarrollo del componente es Java, ya que es Orientado a Objetos, es extensible, modificable, personalizable. Posee herramientas de trabajo gráfico y librerías gráficas de interés para el desarrollo del componente a implementar. Es el propio lenguaje en el que está desarrollado alasARBOGEN, y es de interés para su integración que esté desarrollado en lenguaje Java. Se presentan a continuación las características del lenguaje que lo hicieron idóneo para su selección.

- **JAVA**

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. En sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. (8)

- **Filosofía**

La primera característica, orientado a objetos (“OO”), se refiere a un método de programación y al diseño del lenguaje. Aunque hay muchas interpretaciones para OO, una primera idea es diseñar el software de forma que los distintos tipos de datos que usen estén unidos a sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el “comportamiento” (el código) y el “estado” (datos). El principio es separar aquello que cambia de las cosas que permanecen inalterables. Esta separación en objetos coherentes e independientes ofrece una base más estable para el diseño de un sistema software.

La segunda característica, la independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Este es el significado de ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, “escribe una vez y ejecuta en cualquier lugar”.

❖ **En aplicaciones de escritorio**

Hoy en día existen multitud de aplicaciones gráficas de usuario basadas en Java. El entorno de ejecución Java (JRE), se ha convertido en un componente habitual en los PC de usuarios de los sistemas operativos más usados en el mundo. Además, muchas aplicaciones Java lo incluyen dentro del propio paquete de la aplicación de modo que se ejecuten en cualquier PC. En las primeras versiones de la plataforma Java, existían importantes limitaciones en las APIs de desarrollo gráfico (AWT). Desde la aparición de la biblioteca Swing la situación mejoró sustancialmente.

❖ **AWT**

La *Abstract Window Toolkit* (AWT, en español Kit de Herramientas de Ventana Abstracta) es un kit de herramientas de gráficos, interfaz de usuario, y sistema de ventanas independiente de la plataforma original de Java. AWT es ahora parte de las Java Foundation Classes (JFC) - la API estándar para suministrar una interfaz gráfica de usuario (GUI) para un programa Java. Esta biblioteca estaba concebida como una API estandarizada que permitía utilizar los componentes nativos de cada sistema operativo. Entonces una aplicación Java corriendo en Microsoft Windows usaría el botón estándar de Windows y una aplicación corriendo en UNIX usaría el botón estándar de Motif. En la práctica esta tecnología no funcionó.

❖ **Swing (biblioteca gráfica)**

Swing es una biblioteca gráfica para Java. Incluye widgets para interfaz gráfica de usuario tales como cajas de texto, botones, desplegables y tablas.

Arquitectónicamente es un framework MVC para desarrollar interfaces gráficas para Java con independencia de la plataforma. Sigue un simple modelo de programación por hilos, y posee las siguientes características principales:

- **Independencia de plataforma**.
- **Extensibilidad**: es una arquitectura altamente particionada: los usuarios pueden proveer sus propias implementaciones modificadas para sobrescribir las implementaciones por defecto. Se puede extender clases existentes proveyendo alternativas de implementación para elementos esenciales.
- **Personalizable**: dado el modelo de representación programático del framework de swing, el control permite representar diferentes estilos de apariencia "*look and feel*" (desde apariencia

MacOS hasta apariencia Windows XP pasando por apariencia GTK+, IBM UNIX o HP UX entre otros). Además, los usuarios pueden proveer su propia implementación de apariencia, que permitirá cambios uniformes en la apariencia existente en las aplicaciones Swing sin efectuar ningún cambio al código de aplicación.

1.4.2. Entorno Integrado de Desarrollo.

- **NetBeans 6.9**

Es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo que contiene clases de Java escritas para interactuar con las APIs de NetBeans y un archivo especial (*manifest file*) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

Esta versión del IDE introduce JavaFXComposer, una herramienta de diseño para la creación de aplicaciones gráficas JavaFX, parecido al constructor de aplicaciones gráficas Swing para aplicaciones Java SE. (9)

- **Eclipse**

Eclipse es un Entorno Integrado de Desarrollo (IDE), para todo tipo de aplicaciones libres. Es una herramienta para el programador, principalmente para el desarrollo de aplicaciones Java, facilitando al máximo la gestión de proyectos mediante el control de versiones CVS o con subversión (con subeclipse), es posible también exportar e importar proyectos.

Permite añadir nuevas funcionalidades al editor, a través de nuevos módulos ('plugins'), para programar en otros lenguajes de programación además de Java como C/C++, PHP, Python, Ruby, Cobol, etc. Es un IDE Multiplataforma (GNU/Linux, Solaris, Mac OSX, Windows), soportado para

distintas arquitecturas. Estructurado por plugins, hace sencillo añadir nuevas características y funcionalidades. Tiene resaltado de sintaxis, autocompletado, tabulador de un bloque de código seleccionado. Brinda asistentes (Wizards) para la creación, exportación e importación de proyectos y para generar esqueletos de códigos (templates). (10)

- **Selección del Entorno Integrado de Desarrollo a utilizar**

A partir del estudio de los dos principales IDES para aplicaciones Java, se determinó seleccionar para el desarrollo del componente el NetBeans 6.9, ya que presenta una interfaz más amigable para el diseño visual de aplicaciones, NetBeans ofrece programar interfaces de modo muy visual (mediante Swing). Permite realizar diagramas UML y generar el código correspondiente. Es multiplataforma. Posee una multitud de lenguajes, de plugins y de características que facilitan el trabajo del desarrollador, además de importar plugins y librerías de una manera muy sencilla, dándole más potencia y fortaleza al IDE. Son características en el IDE la capacidad de respuesta y el rendimiento.

1.4.3. Herramienta de Modelado.

- **Visual Paradigm**

Visual Paradigm para UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado ayuda a una construcción más rápida de aplicaciones de calidad, mejores y a un menor coste, además permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación, así como una serie de tutoriales, demostraciones interactivas y proyectos. (11) Es importante señalar que la herramienta de modelado Visual Paradigm no es gratuito, pero la compañía Visual Paradigm UML Comunity, tiene disponible distintas versiones y facilita licencias especiales para fines académicos, sin interés de lucro. (12)

Visual Paradigm para UML (VP-UML) es la alternativa fácil y económica junto a la Borland IBM Rational Rose. Su Smart Development Environment (SDE) está disponible en varias versiones, donde cada una de las cuales se integran con un entorno de desarrollo.

Las diez razones por la que una organización debe seleccionar VP-UML y la SDE sobre otras herramientas de modelado son: (13)

- Permite incorporar dibujos de Visio en cualquier diagrama UML. A diferencia de otras herramientas CASE-UML, VP-UML y SDE prolongan esta limitación permitiendo incorporar dibujos de Visio en cualquier diagrama de UML, pudiendo modelar un dominio específico de hardware, software, redes, componentes, etc.

- Integración perfecta con los principales IDEs: SDE está disponible en varios IDEs incluyendo Microsoft Visual Studio, BorlandJBuilder, Eclipse/IBM WSAD, NetBeans/Sun ONE, IntelliJ IDEA y JDeveloper, lo que puede proporcionar un entorno que integra todo el modelo de código desplegando el proceso de desarrollo de software.

- Completa integración con Microsoft Office: VP-UML y SDE soporta la copia de diagramas como objetos OLE, para poder pegar el diagrama a cualquier documento Word, Excel, Power Point. El contenido del diagrama se puede editar directamente, sin embargo, no hay que preocuparse de perder el diagrama original de la fuente.

- Herramientas efectivas y costeables: Son las más asequibles con las actuales funciones de las herramientas CASE-UML. Cuestan la octava parte del precio de otras herramientas de modelado con funcionalidades similares. Sus diversas ediciones se encuentran disponibles y se puede elegir de acuerdo con su presupuesto y necesidades.

- Herramientas UML más fáciles de usar: Ofrece una edición de líneas para diagramas UML. Se puede crear y especificar un modelo de elementos sin cuadro de diálogos. Los recursos centrados en la interfaz ayuda, en las tareas y botones de acceso directo se muestran cuando un elemento del diagrama está seleccionado. Se realizan operaciones como mostrar/ocultar compartimentos, ajuste de tamaño y creación de elementos con un sólo clic del ratón.

- Soporta múltiples plataformas: No importa el Sistema Operativo que esté en uso, VP-UML y SDE se puede ejecutar en equipo y están disponibles en muchas plataformas como Windows, Linux, Mac OS X y Java Desktop.

- Análisis textual para la identificación de clases candidatas: El análisis textual es una técnica útil y práctica para la captura de los requisitos del sistema y la identificación de las clases candidatas. Estas son las únicas herramientas CASE que apoyan el análisis textual. Sólo se

rellena la declaración del problema y las herramientas pueden ayudarle a identificar el sistema de clases con sólo arrastrar y soltar.

- Soporte para las tarjetas CRC: Las tarjetas CRC identifican clases, responsabilidades y colaboraciones en un sistema OO. Son las únicas herramientas de modelado que apoyan las tarjetas CRC.

- Conversión instantánea de código fuente, archivos ejecutables y binarios en modelos: Permite invertir programas de código fuente, archivos ejecutables y binarios y modelos UML de inmediato. Además de idiomas y formatos que incluyen XML, esquema XML, .NET dll o exe, Java de fuente/class/jar, origen de archivo C++ y archivos fuente CORBA IDL.

- Diseño automático del diagrama: Con su diseño automático se pueden ordenar los diagramas desordenados con tan sólo unos pocos clics con el ratón. Se puede elegir el trazado ortogonal, la disposición jerárquica o el árbol del diseño de acuerdo a la naturaleza del diagrama. Cada estilo del diseño puede afinarse con un conjunto de parámetros configurables.

Visual Paradigm es una herramienta de modelado multiplataforma que brinda una interfaz de usuario amigable, permite integrarse con varios IDEs de desarrollo. También posibilita la aplicación de Ingeniería inversa, permitiendo entregar al cliente un producto bien documentado. Además, tiene incluido un panel para el diseño de la interfaz de usuario lo cual permite no tener que integrarlo con Visio.

Como herramientas de modelado se estudiaron tres principales que constituyen líderes en el mundo del desarrollo de software actual. Ellas son: Rational Rose, ArgoUML y Visual Paradigm, seleccionándose la última como la más adecuada para el desarrollo del componente, ya que es una herramienta CASE para UML que es muy fácil de usar, soporta la última anotación UML 2.1, ingeniería inversa así como directa, generación de código, modelado de diagramas de clases, importación desde Rational Rose, exportación/importación XML, generador de informes, editor de figuras e integración con MS Visio, plugin, integración IDE con Eclipse, NetBeans y otros.

Es una herramienta que provee el modelado de negocio y un lenguaje de mapeo para lenguajes de programación de alto nivel como son Java y .NET.

Es un estándar que se encuentra ampliamente utilizado en el mundo por las empresas, para el modelado en el proceso de desarrollo de software. Una de las principales características que presenta es que su diseño se ve centrado en los casos de uso y enfocado en el desarrollo de un software con mayor calidad. Genera la documentación del proyecto en distintos formatos, destacándose el formato pdf y permite el control de versiones. Además de las facilidades que brinda para integrarse con el modelo UML, permitiendo un modelado de procesos de negocio capaz de plasmar en modelos, las actividades reales del Tutor Virtual. (14)

1.4.4. Lenguaje de modelado.

- **UML (Lenguaje Unificado de Modelado)**

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos.

Ventajas:

- Es un lenguaje conocido
- Estándar
- Fácil de aprender

Desventajas:

- No ha sido diseñado para modelar procesos de negocios
- Implica un enfoque orientado a objetos
- UML no tiene todavía una semántica formal.

Se estudiaron varios lenguajes de modelado centrándose la selección entre el BPMN y UML, donde se llegó a la conclusión que el más adecuado para el desarrollo del componente es UML pues permite modelar el negocio de manera clara y entendible para el equipo de desarrollo y para el cliente. Además, permite modelar todo el ciclo de desarrollo de software incluyendo esquemas de bases de datos y componentes reusables, los cuales ahorrarán tiempo de trabajo a los programadores.

1.4.5. Metodología de Desarrollo de Software.

Una metodología de desarrollo de software indica paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben tener. Detalla la información que se debe producir como resultado de una actividad y la necesaria para comenzarla.

- **OpenUP/Basic**

Es un FrameWork de procesos de desarrollo de software de código abierto. Presenta un modelo extensible, dirigido a la gestión y desarrollo de proyectos de software basados en desarrollo iterativo, es ágil e incremental; aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo. Es un proceso para pequeños equipos de desarrollo que valoran los beneficios de la colaboración y del trabajo mutuo.

OpenUP / Basic divide el proyecto en iteraciones: intervalos planificados, en caja de tiempo, por lo general se mide en semanas. Las iteraciones necesitan de la atención y responsabilidad del equipo en la entrega de valor incremental a los interesados de una manera predecible. El plan de iteración define lo que debe ser entregado dentro de la iteración, y el resultado es un artefacto o entregable. Permite auto-organizarse en torno a cómo lograr los objetivos de la iteración y compromete a entregar los resultados. (29)

Conserva las características principales del modelo de desarrollo RUP, incluye el desarrollo iterativo, permite identificar los requisitos operacionales del sistema, prever las interacciones con los usuarios y prevenir los posibles riesgos en el desarrollo del sistema. Es ligero y proporciona una comprensión detallada del proyecto, beneficiando a clientes y desarrolladores sobre productos a entregar y su formalidad.

Es una forma de desarrollo ágil y ligera, consiste en equipos a los cuales se les asigna una fase del desarrollo que tienen que complementarse entre sí para obtener un buen producto final, no puede ser una sola persona la que realice todo el trabajo pues esto podría ocasionar que se pierda de vista ciertas características importantes. Por ejemplo para un proyecto pequeño constituyen equipos de tres a seis personas e implican tres a seis meses de esfuerzo del desarrollo. Describe el conjunto mínimo de roles, y para cada uno un conjunto de tareas a realizar y artefactos que obtendrán en el desarrollo del software. (15)

Después del estudio y análisis de un conjunto de metodologías de desarrollo de software -ágiles y robustas- como: XP, RUP, SCRUM y OpenUp/Basic se determinó que la más adecuada para el desarrollo del componente es OpenUP / Basic debido a que es una forma ágil y ligera, que incluye el desarrollo iterativo, aplicable a pequeños proyectos. Es un proceso de desarrollo iterativo del software que es mínimo porque solo incluye el contenido del proceso fundamental; completo debido a que el mismo puede ser manifestado como proceso entero para construir un sistema; y extensible ya que puede ser utilizado como base para agregar o para adaptar más procesos. (16) OpenUp es la metodología utilizada por desarrolladores de alto nivel en casi todo el mundo por sus altas cualidades administrativas. Entre sus beneficios se encuentran: permitir disminuir las probabilidades de fracaso, incrementar las probabilidades de éxito y permitir detectar errores tempranos a través de un ciclo iterativo. (17)

1.5. Conclusiones parciales

En este capítulo se realizó un estudio de los principales conceptos del problema a resolver, entiéndase el estudio de los árboles genealógicos y de las características de un componente de software. Se realizó un análisis de los principales sistemas de representación gráfica de árboles genealógicos en el mundo y en el país. A partir de las características del sistema alasARBOGEN se evidenció la necesidad actual de realizar mejoras visuales al mismo. Para el desarrollo del componente se definieron las herramientas y tecnologías a utilizar, quedando propuestas como metodología de desarrollo OpenUp/Basic, Visual Paradigm como herramienta CASE, como lenguaje de programación JAVA y el entorno integrado de desarrollo NetBeans.

CAPÍTULO 2: DISEÑO DEL COMPONENTE

En el presente capítulo se definirán los requisitos del componente a implementar. Se explica la arquitectura utilizada así como los principales patrones de diseño empleados. Se definirán los Casos de uso así como sus descripciones. Se realizan los diagramas de clases del diseño y los diagramas de secuencia, describiendo como implementar, enfocado a como el componente cumple los objetivos planteados en los requisitos funcionales y no funcionales

2.1. Descripción del componente a desarrollar

El componente a desarrollar tiene como objetivo fundamental la visualización gráfica de los árboles genealógicos. Entre las principales opciones que el mismo brindará se destacan:

- Visualizar una leyenda con los nomencladores de los símbolos genéticos.
- Ordenar un árbol genealógico de la forma jerárquica clásica vertical.
- Visualizar el árbol genealógico de la forma concéntrica cuadrada
- Visualizar el árbol genealógico de la forma descendente horizontal.

2.2. Especificación de los Requisitos del Componente.

Un requisito es: (18)

- Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
- Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
- Una representación documentada de una condición o capacidad como en las dos anteriores.

Los requisitos se pueden clasificar en: funcionales y no funcionales. Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Los requisitos no funcionales son propiedades o cualidades que el producto debe tener, las características que hacen al producto atractivo, usable, rápido o confiable.

2.2.1. Requisitos Funcionales del Componente.

El sistema alasARBOGEN cuenta con requisitos funcionales que fueron cumplidos una vez liberado el proyecto. Para el desarrollo del componente a implementar se reutilizan algunas clases ya definidas en el sistema. Del sistema alasARBOGEN es propio el requisito: RF1, el cual será modificado aportando mejoras a la solución actual. Serán agregados los requisitos: RF2, RF3 y RF4. El componente una vez desarrollado debe cumplir con la totalidad de los requisitos propuestos.

RF1. Ordenar Árbol Genealógico.

RF2. Visualizar Árbol Concéntrico Cuadrado.

RF3. Visualizar Árbol de Descendencia Horizontal.

RF4. Visualizar Leyenda.

2.2.2. Requisitos No Funcionales del Componente.

- **Apariencia e interfaz externa**

Su interfaz de presentación será simple y más amigable, posibilitando que sea de fácil uso por el genetista. Tendrá un ambiente de trabajo similar, independiente del sistema operativo en que se trabaje. Predominará en el componente los colores azul y blanco. La interfaz externa debe estar diseñada para verse en cualquier resolución igual o superior a 1024x768. Se mostrará en pantalla un área de trabajo en la que se realizarán las distintas operaciones sobre el árbol genealógico. Se presentan opciones de visualización del árbol genealógico para las distintas interpretaciones que desee hacer el cliente.

- **Usabilidad**

El acceso al componente se realizará de forma directa y rápida. Este puede ser usado por cualquier persona que posea conocimientos básicos en el manejo de una computadora.

- **Portabilidad**

El componente debe permitir ejecutarse en cualquiera de los sistemas operativos más usados en la actualidad. Destacándose el sistema operativo GNU Linux.

- **Seguridad**

El componente a implementar se regirá por las políticas y restricciones de seguridad del sistema alasARBOGEN. El mismo no implementará mecanismos de seguridad propios, ya que funcionará como un componente íntegro del sistema alasARBOGEN.

- **Software**

Se requiere para el funcionamiento de la aplicación disponer de un sistema operativo que tenga la Máquina Virtual de Java 5.0 o versiones superiores.

- **Hardware**

Para el desarrollo y ejecución del componente se recomienda:

- Procesador Pentium 4 o superior.
- 512 MB RAM como mínimo.

- **Requisitos legales**

Las tecnologías y herramientas que se utilicen para desarrollar el sistema deben estar bajo la licencia de software libre.

2.3. Definición de los Casos de Uso del Sistema

2.3.1. Actor del Sistema.

Actor	Descripción
Genetista	El especialista en genética es el que interactúa con el componente y el único encargado de realizar las operaciones que brinda el mismo.

Tabla 1. Actor del Sistema

2.3.2. Diagrama de Casos de Uso del Sistema.

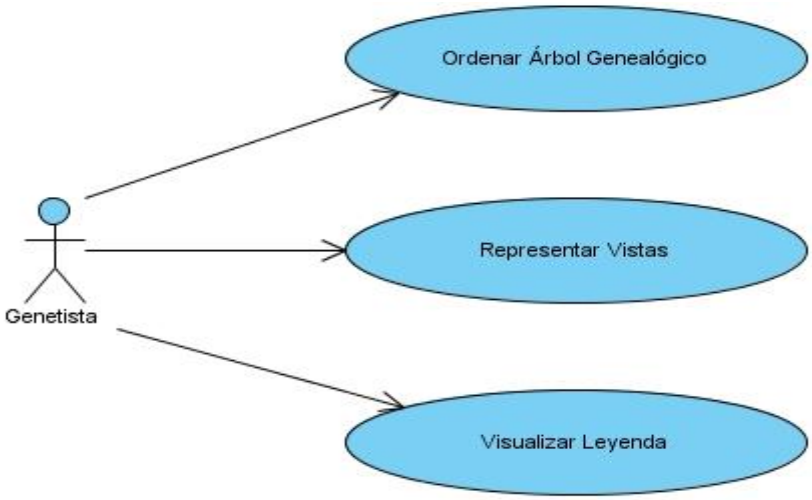


Figura 1. Diagrama de Casos de Uso del Sistema

2.3.3. Descripción de los Casos de Uso del Sistema.

- **CU Ordenar Árbol Genealógico.**

Caso de Uso:	Ordenar Árbol Genealógico.
Actores:	Genetista
Resumen:	El caso de uso se inicia una vez que el genetista selecciona la opción: "Ordenar Árbol". El caso de uso finaliza una vez que se muestra el árbol genealógico ordenado.
Precondiciones:	La aplicación alasARBOGEN debe estar ejecutándose. Debe estar visualizado el árbol genealógico.
Referencias:	RF1.
Prioridad:	Crítico.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El genetista da clic en la Opción "Ordenar Árbol" del Menú "Edición".	2. El sistema visualiza el árbol genealógico actual ordenado correctamente.

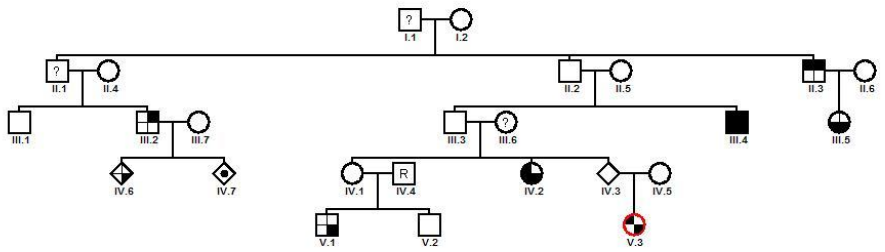
Prototipo de Interfaz 	
Poscondiciones	Se muestra un árbol genealógico clásico ordenado jerárquicamente.

Tabla 2. Ordenar Árbol Genealógico.

- **CU Representar Vistas.**

Caso de Uso:	Representar Vistas.
Actores:	Genetista
Resumen:	El caso de uso se inicia cuando el genetista da Clic en el submenú "Vistas" del menú "Edición". El caso de uso finaliza una vez que se visualizan las diferentes representaciones del árbol genealógico actual.
Precondiciones:	La aplicación alasARBOGEN debe estar ejecutándose. Debe estar visualizado el árbol genealógico.
Referencias:	RF2 y RF3
Prioridad:	Crítico.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El genetista da clic en el submenú "Vistas" del menú "Edición".	2. El sistema muestra dos Ítems : Ítem1: Árbol Concéntrico Cuadrado. Ítem2: Árbol Descendente Horizontal.
3. El genetista selecciona el tipo de representación que desea se visualice.	4. El sistema verifica que se haya seleccionado un individuo.

	5. Si ha sido seleccionado un individuo previamente, en dependencia de la selección realizada por el genetista, el sistema hace lo siguiente: • Si se selecciona el Ítem: Árbol Concéntrico Cuadrado ir a sección "Visualizar Árbol Concéntrico Cuadrado". • Si se selecciona el Ítem: Árbol Descendente Horizontal ir a sección "Visualizar Árbol Descendente Horizontal".
Sección "Visualizar Árbol Concéntrico Cuadrado"	
1. El genetista da clic en el Ítem: Árbol Concéntrico Cuadrado.	2. El sistema visualiza el árbol genealógico en su forma Concéntrica Cuadrada.
Sección "Visualizar Árbol Descendente Horizontal"	
1. El genetista da clic en el Ítem: Árbol Descendente Horizontal.	2. El sistema visualiza el árbol genealógico en su forma Descendente Horizontal.
Flujo Alterno General	
	5.1 Si no ha sido seleccionado algún individuo previamente, el sistema muestra el mensaje: "Seleccione un individuo".

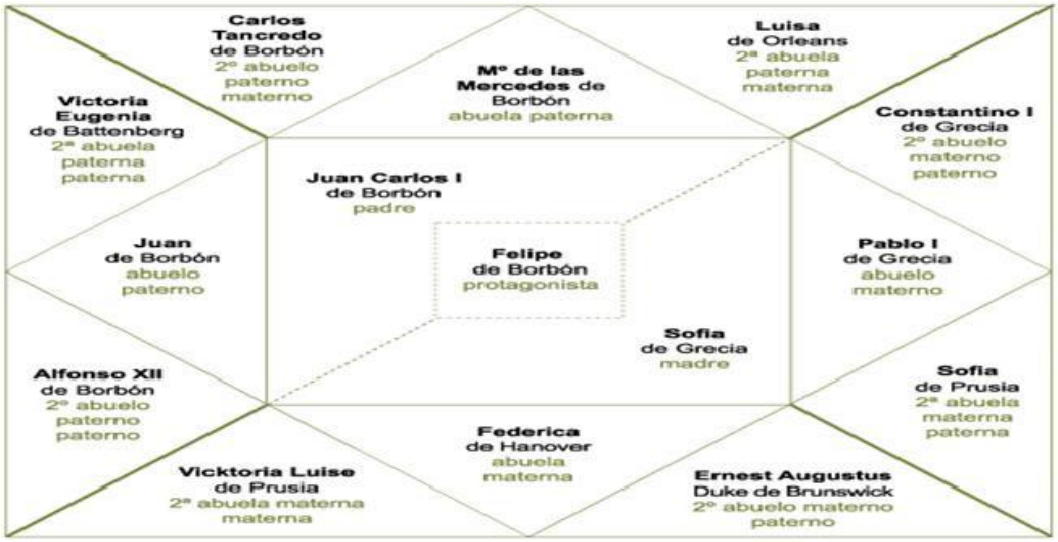
Prototipo de Interfaz 	
Poscondiciones	El árbol genealógico clásico queda visualizado en sus diferentes representaciones.

Tabla 3. CU Representar Vistas.

- **CU Visualizar Leyenda.**

Caso de Uso:	Visualizar Leyenda.
Actores:	Genetista
Resumen:	El caso de uso se inicia cuando el genetista da clic en la opción “Leyenda” del menú “Ver” y el sistema muestra en una nueva pestaña una leyenda en la que se identifican los significados de los nomencladores utilizados en la simbología médica. El mismo finaliza una vez que se cierra la aplicación.
Precondiciones:	La aplicación alasARBOGEN debe estar ejecutándose.
Referencias:	RF4.
Prioridad:	Crítico.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

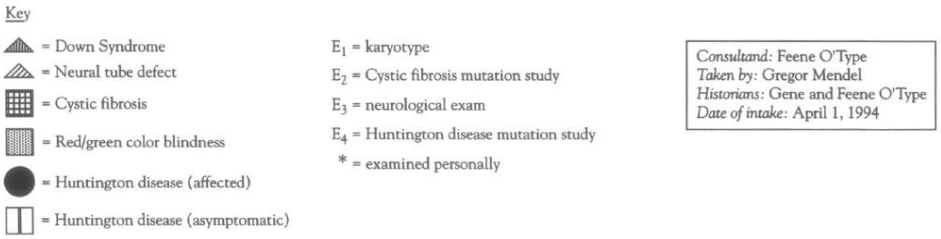
1. El genetista selecciona la opción "Leyenda" en el menú "Ver".	2. El sistema muestra en una nueva pestaña una leyenda con los significados de los nomencladores utilizados en la simbología médica así como los comentarios realizados por el genetista a los individuos de la familia en estudio. Se identifican de esta manera los significados genéticos de cada símbolo.
Prototipo de Interfaz 	
Poscondiciones	Queda mostrada la leyenda de la simbología utilizada.

Tabla 4. CU Visualizar Leyenda

2.4. Arquitectura Utilizada

Se utilizó la arquitectura Multi-capa, pues permite el aislamiento de la lógica de aplicaciones en componentes independientes susceptibles de reutilizarse después en otros sistemas. Hace una distribución de las capas en varios nodos físicos y en varios procesos. Cada capa puede ser modificada tanto como sea necesario sin provocar cambios en las demás. Una capa no tiene dependencias con la capa superior, cada capa depende solamente de la fachada que le permite comunicarse con la capa inmediata inferior. Esta dependencia entre capas es normalmente a través de fachadas, asegurando que el acoplamiento sea el más bajo posible y la abstracción del funcionamiento de la capa inferior, sea casi total. Todo ello mejora el desempeño, la coordinación y el compartir la información.

2.4.1. Estilo Arquitectónico Utilizado

Fueron estudiados los estilos arquitectónicos de llamada y retorno ya que enfatizan la modificación y la escalabilidad, además de ser idóneos para los sistemas orientados a objetos. De ellos fue usado para el desarrollo del componente el patrón arquitectónico Modelo Vista Controlador. Este patrón de diseño, se basa en la separación en tres capas del diseño de las aplicaciones: el modelo de datos, la presentación de los mismos y las acciones de los usuarios. El componente hace uso intensivo de interfaces visuales que requieren de una actualización constante de las propiedades que se modifican. Realiza un diseño que desacopla la vista del modelo, con la finalidad de mejorar la reusabilidad, de esta forma, las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos. A continuación se muestra este patrón de arquitectura

El patrón Modelo – Vista – Controlador separa el modelo del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes: (19)

- **Modelo:** Representa específicamente el dominio de la información sobre la cual funciona la aplicación. El modelo es otra forma de llamar a la capa de dominio. La lógica de dominio añade significado a los datos. El modelo encapsula los datos y las funcionalidades. Es posible apreciar en la figura 2 como las clases **GrafoNoPonderadoLA**, **PersonaVisual** y **MatrimonioVisual**, son las responsables de administrar el comportamiento y los datos del dominio de aplicación, quienes responden a instrucciones de cambiar el estado (desde el controlador).
- **Vista:** Presenta el modelo en un formato adecuado para interactuar, usualmente con un elemento de interfaz de usuario. Maneja la visualización de la información. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador. Se puede observar en la figura 2 como la capa Vista encapsula todos los componentes visuales que se muestran al usuario a través de eventos de salida, díganse **Main**, **Horizontal** y **Leyenda**, que solo se encargan de mostrar la información que les envía la clase controladora, actualizando sus vistas.
- **Controlador:** Responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Los eventos son traducidos a solicitudes de servicio (“services request”) para el modelo o la vista. En la figura 2 se puede observar la capa

Controlador, donde se encuentra la clase controladora **ArbolControler** como la encargada de responder ante eventos del usuario, haciendo cambios en las vistas y en el modelo.

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual.

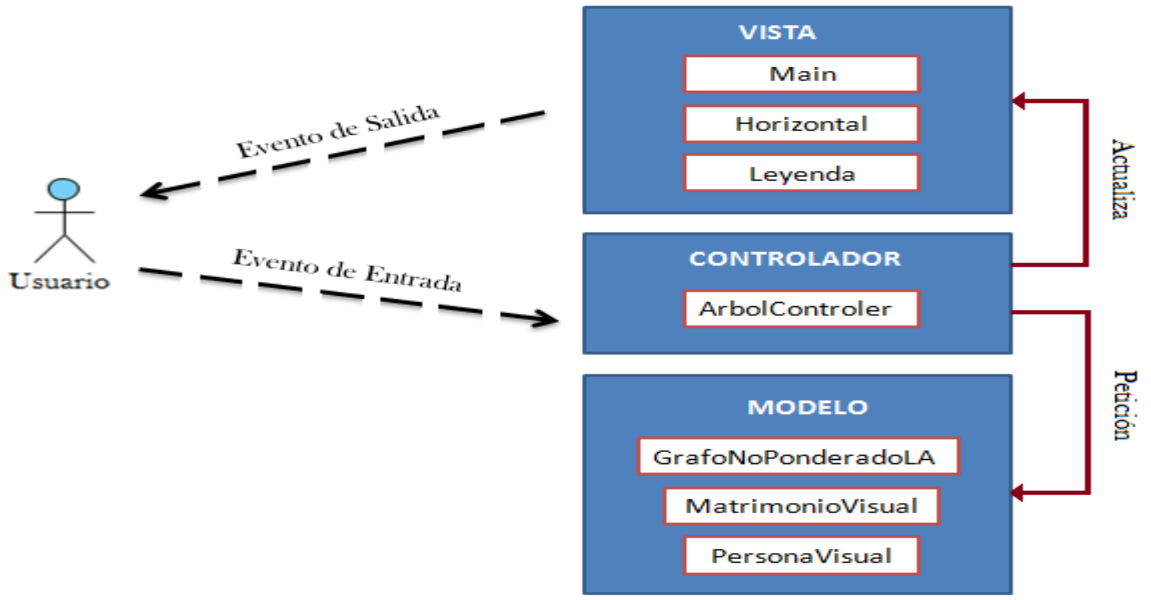


Figura 2. Patrón arquitectónico M-V-C.

Entre las principales **ventajas** que presenta este patrón arquitectónico se encuentran: (19)

- Soporte para múltiples vistas: puesto que la vista se separa del modelo y no hay ninguna dependencia directa entre vista y modelo, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos al mismo tiempo.
- Mayor soporte a los cambios: los requisitos de interfaz tienden a cambiar más rápidamente que las reglas de negocio. Puesto que el modelo no depende de las vistas, la adición de nuevos tipos de vista al sistema generalmente no afectan al modelo. Por tanto, el ámbito del cambio se limita a la vista.

En el desempeño de este trabajo se hace énfasis en la capa “VISTA” debido a que el resultado esperado es un componente visual. El mayor peso del aporte científico se realizará en el trabajo sobre la vista y el controlador.

2.5. Patrones de Diseño Utilizados

Un patrón es una solución recurrente para un problema típico y en un contexto determinado. La solución se refiere a la respuesta al problema, por lo que ayuda a resolver las dificultades de diseño en problemas similares. Los patrones deberían comunicar soluciones de diseño a los desarrolladores y arquitectos que los leen y los utilizan. **(20)**

En el diseño del componente se utilizaron los siguientes patrones GRASP para dar solución a las diferentes problemáticas.

- **Patrón Creador.**

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se conecta con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. El patrón Creador indica que la clase incluyente del contenedor o registro es idónea para asumir la responsabilidad de crear la instancia contenida o registrada.

En el diseño del componente se definió que la clase **ArbolControler** es la encargada de crear una nueva instancia de la clase **Main**. La clase controladora del componente a desarrollar es la encargada de crear un objeto visual “Main” para mostrarlo en una nueva vista.

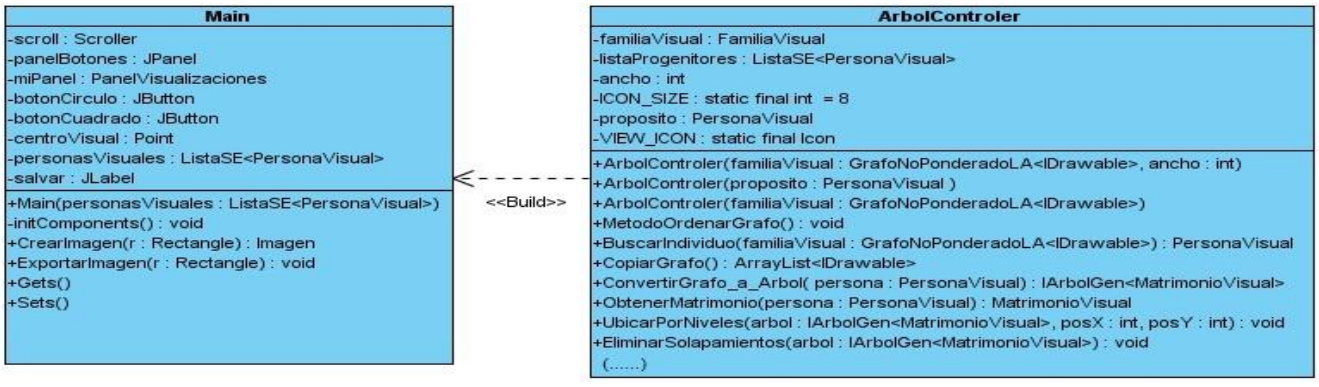


Figura 3. Ejemplo de utilización del patrón Creador.

- **Patrón Experto.**

El patrón experto asigna una responsabilidad al experto en información, es decir, la clase que cuenta con la información necesaria para cumplir la responsabilidad. Conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide.

La clase **Leyenda** contiene todos los atributos necesarios para mostrar en la vista, además de cumplir con la responsabilidad de crear una View para ser integrada. Encapsula las funcionalidades necesarias para la creación del componente visual.

Figura 4. Ejemplo de utilización del patrón Experto.

- **Patrón Polimorfismo.**

Este patrón se utiliza cuando las alternativas o comportamientos relacionados varían según el tipo (clase), asigna la responsabilidad para el comportamiento, utilizando operaciones polimórficas a los tipos para los que varía el comportamiento. Tiene como ventajas que se añaden fácilmente las extensiones necesarias para nuevas variaciones y las nuevas implementaciones se pueden introducir sin afectar a los clientes.

El uso de este patrón da respuesta a la problemática de manejar diferentes comportamientos en dependencia del tipo de objeto. En el diseño de la solución fue utilizado por la necesidad de expresar diferentes comportamientos de un determinado objeto. Es posible observar como en la figura 5 se muestra la clase ***Cuadrado*** que define la funcionalidad **paintComponent()**, la cual es redefinida por las clases ***CuadradoLineal*** y ***CuadradoMoved.***

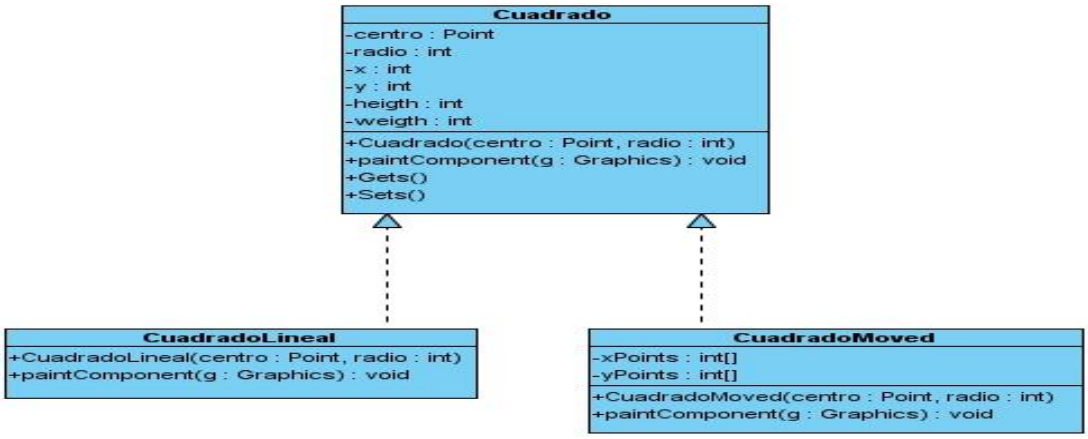


Figura 5. Ejemplo de utilización del patrón Polimorfismo.

- **Patrón Cadena de Responsabilidad**

La cadena de responsabilidad se encarga de evitar el acoplamiento del remitente de una petición a su receptor, dando a más de un objeto la posibilidad de manejar la petición. Es utilizado en la mayoría de los diagramas de clases siendo aplicado en el tratamiento de excepciones. Un ejemplo de su uso es cuando se produce un error al insertar un valor en una lista en una posición no válida, el cual es captado por las clases inferiores, reenviando la excepción hasta la clase controladora donde se traduce al lenguaje del usuario.

2.6. Diagramas de Clases del Diseño

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema; también muestra sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, cuando se crea el diseño conceptual de la información que se manejará en el sistema, conjuntamente con los componentes que se encargarán del funcionamiento y las relaciones entre unos y otros. A continuación se muestran los diagramas de clase propuestos:

2.6.1. Diagrama de Clases del Diseño del CU Ordenar Árbol Genealógico.

Este diagrama muestra la clase controladora **ArbolControler** que es la encargada de realizar la funcionalidad de ordenar el árbol genealógico. Para ello hace uso de las clases **ArbolGen** y **MatrimonioVisual**, que son utilizadas para la confección de la estructura capaz de convertir el grafo en un árbol, logrando la jerarquía en el grafo dado por parámetro. A continuación, el diagrama correspondiente:

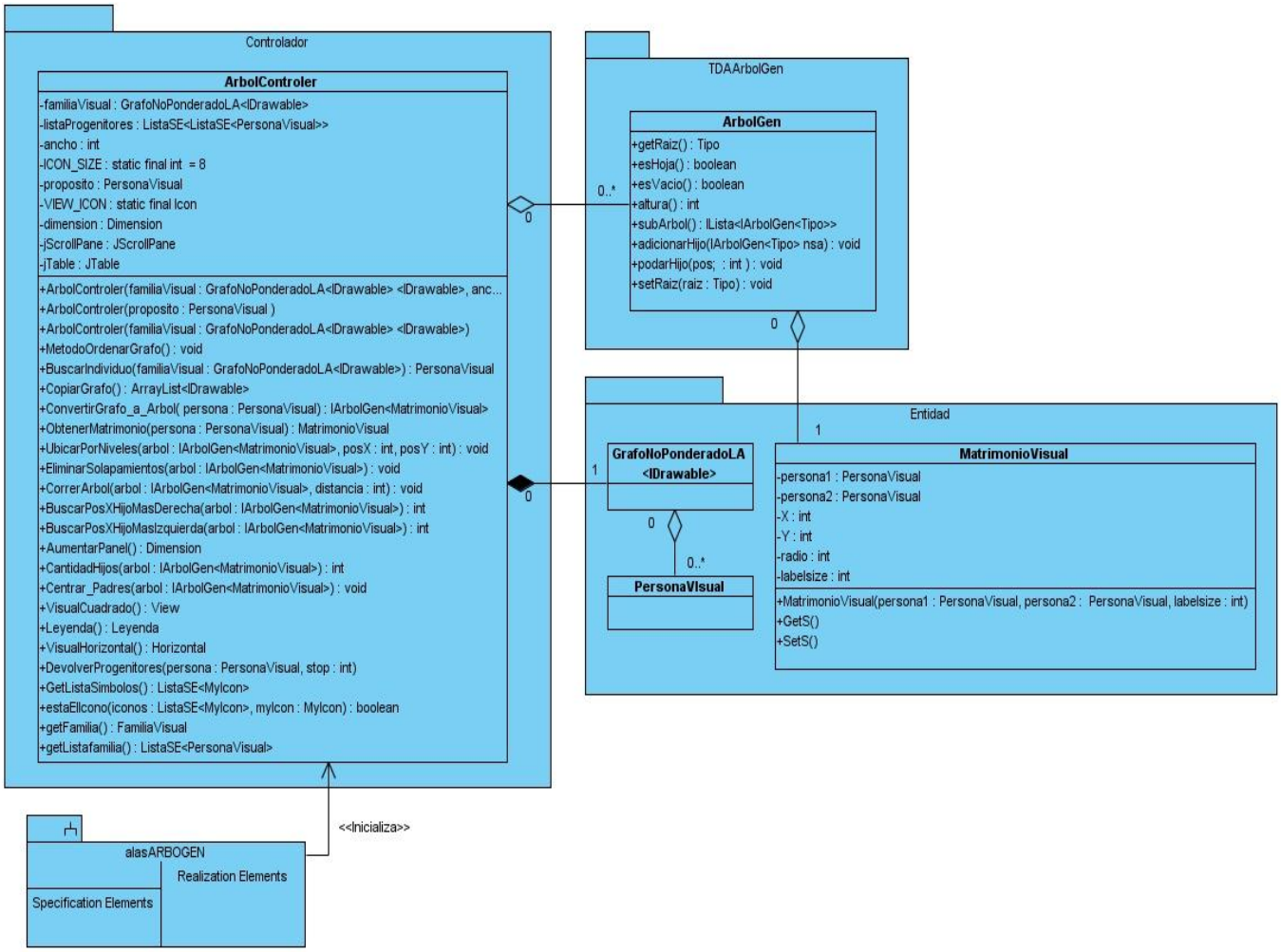


Figura 6. Diagrama de Clases del Diseño del CU Ordenar Árbol Genealógico.

2.6.2. Diagrama de Clases del Diseño del CU Representar Vistas.

Este diagrama evidencia las relaciones de composición y agregación entre las clases visuales. Se puede observar como la clase **Main** es una clase contenedora de las clases **PanelCuadrado** y **Scroller**. La clase **PanelCuadrado** muestra la vista de Árbol Concéntrico Cuadrado, mientras que la clase **Scroller** gestiona el posicionamiento del **PanelCuadrado** en el contenedor **Main**. A su vez, la clase **PanelCuadrado** está compuesta por diferentes clases que dan forma al componente visual a representar. Además, se observa la relación entre la clase **ArbolControler** y **Horizontal** siendo la primera la clase que crea la segunda, para mostrar la vista de Árbol Descendente Horizontal.

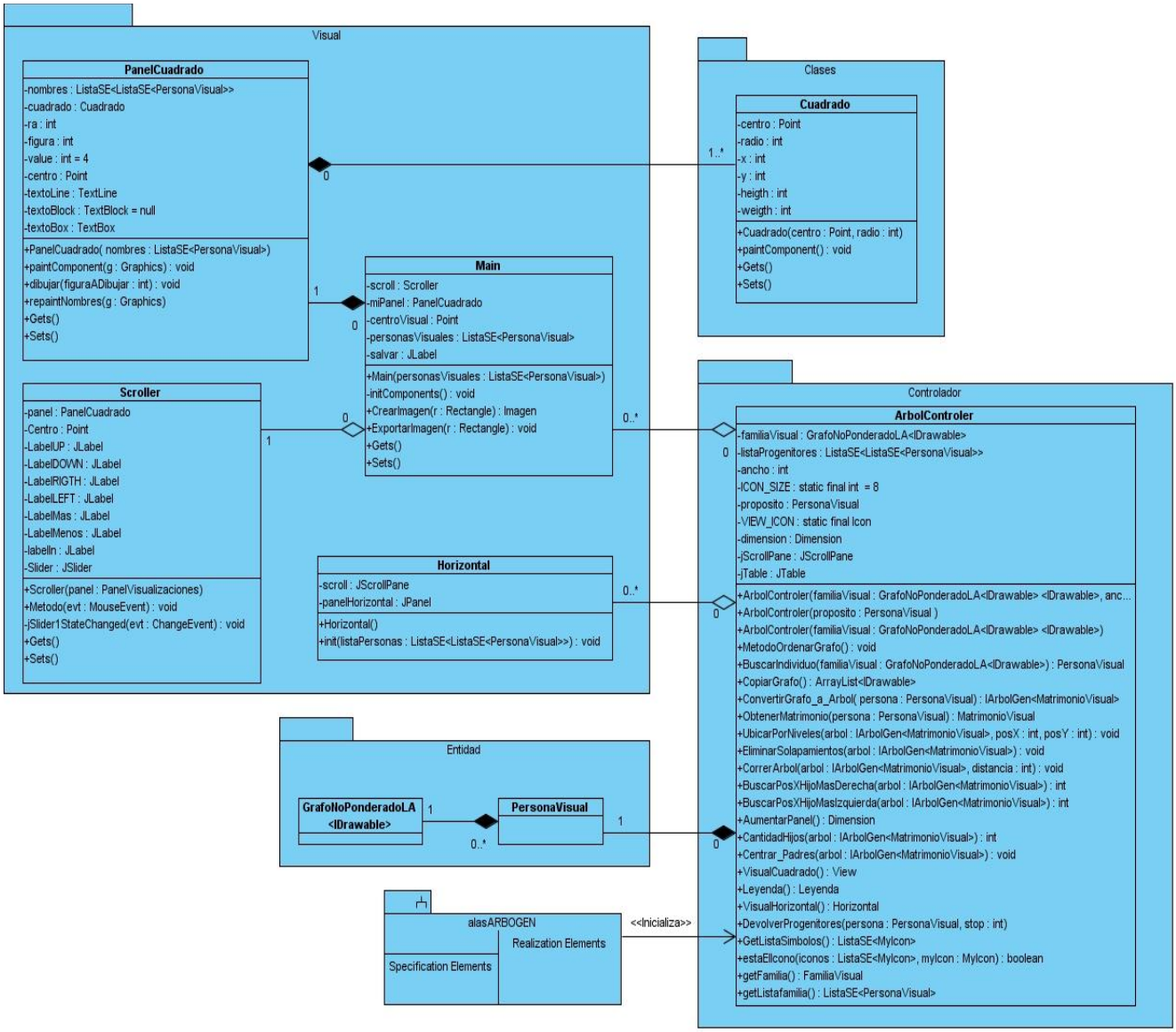


Figura 7. Diagrama de Clases del Diseño del CU Representar Vistas

2.6.3. Diagrama de Clases del Diseño del CU Visualizar Leyenda.

El diagrama muestra las relaciones entre las clases **ArbolControler** y **Leyenda**. La primera, gestiona la información a mostrar en la vista y la segunda la encargada de mostrar la información al cliente. Se recibe como entrada un grafo con la familia en estudio.

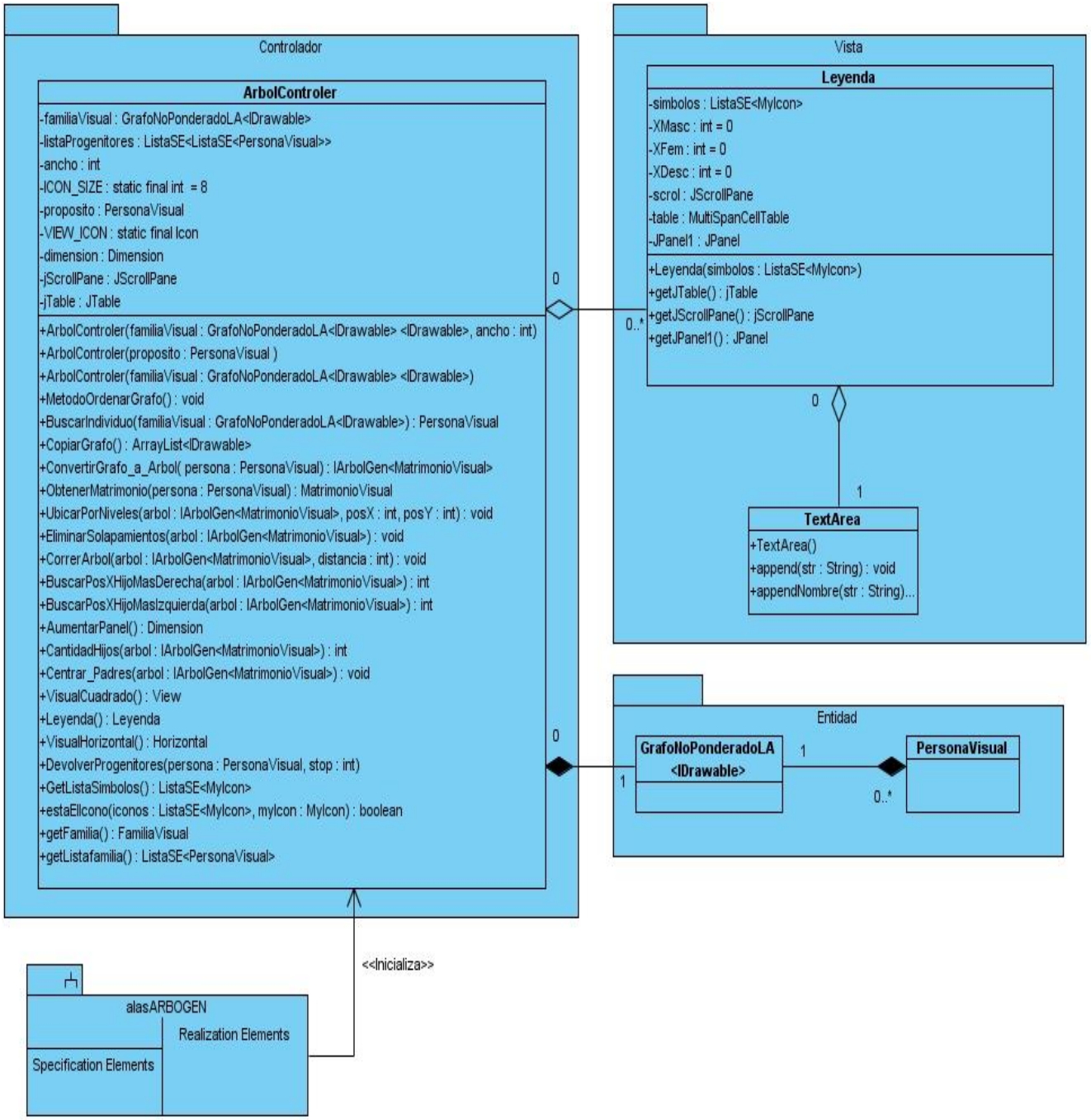


Figura 8. Diagrama de Clases del Diseño del CU Visualizar Leyenda.

2.7. Diagramas de Secuencia

Los diagramas de secuencia, representan con gran claridad, el flujo de acciones que el sistema debe realizar. Estos están compuestos por una serie de objetos que interactúan entre sí, a través de mensajes que estos se envían y reciben y que a su vez se evidencian en el diagrama. Los diagramas de secuencia guían al programador durante la fase de implementación. A continuación, se presentan los diagramas de secuencia correspondientes a cada caso de uso:

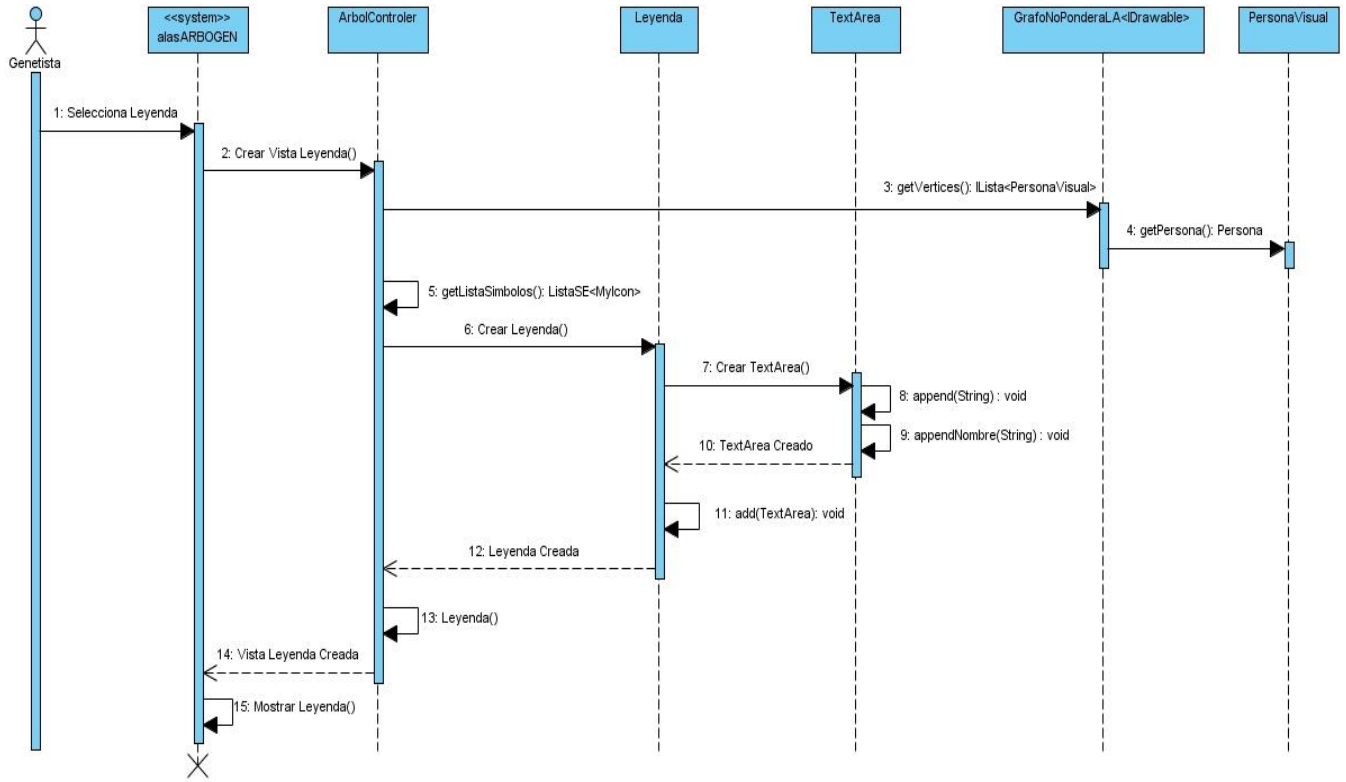


Figura 9. Diagrama de Secuencia del CU Visualizar Leyenda.

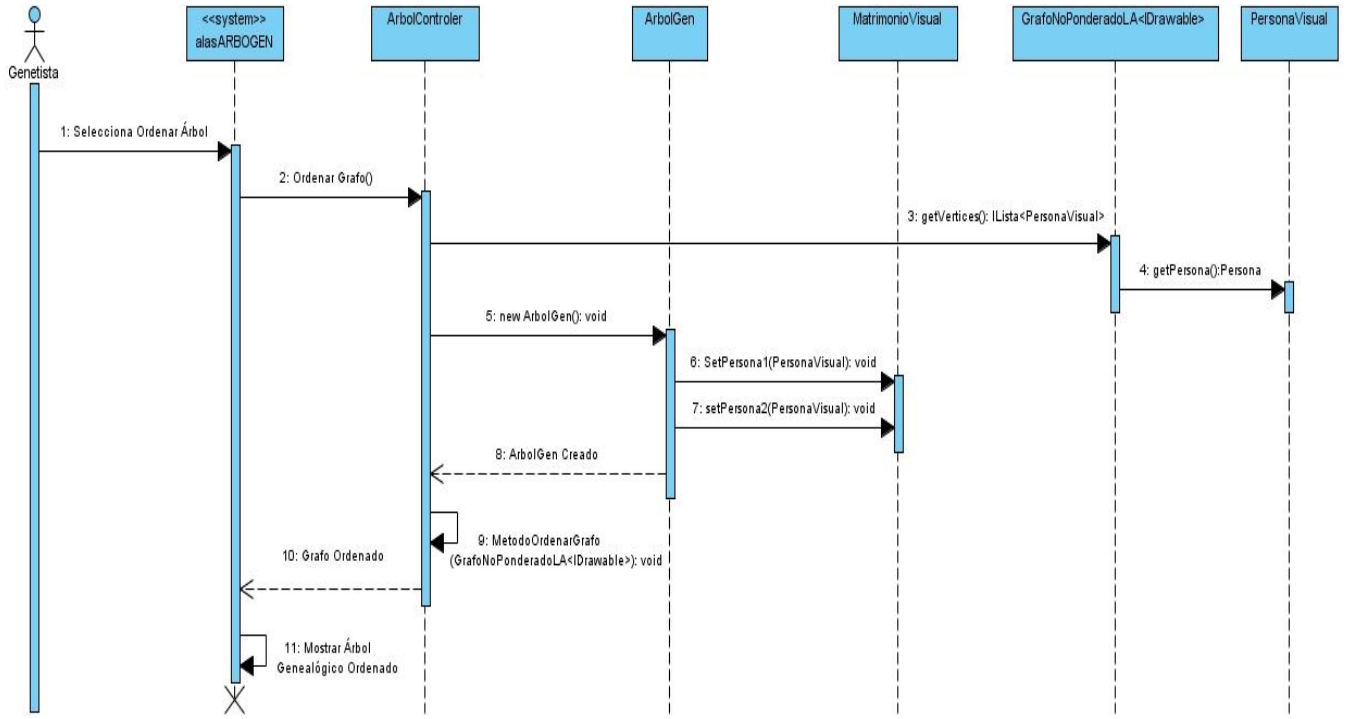


Figura 10. Diagrama de Secuencia del CU Ordenar Árbol Genealógico.

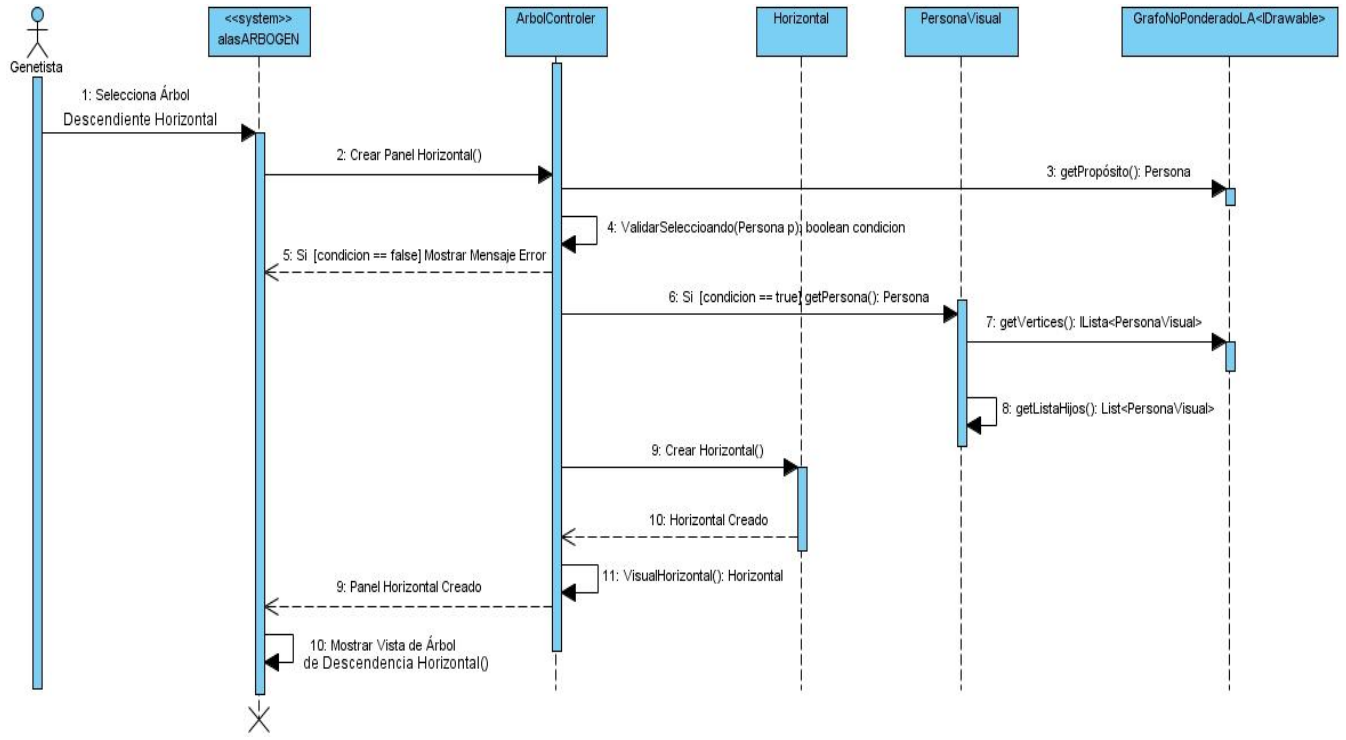


Figura 11. Diagrama de Secuencia del CU Representar Vistas. Escenario 2.

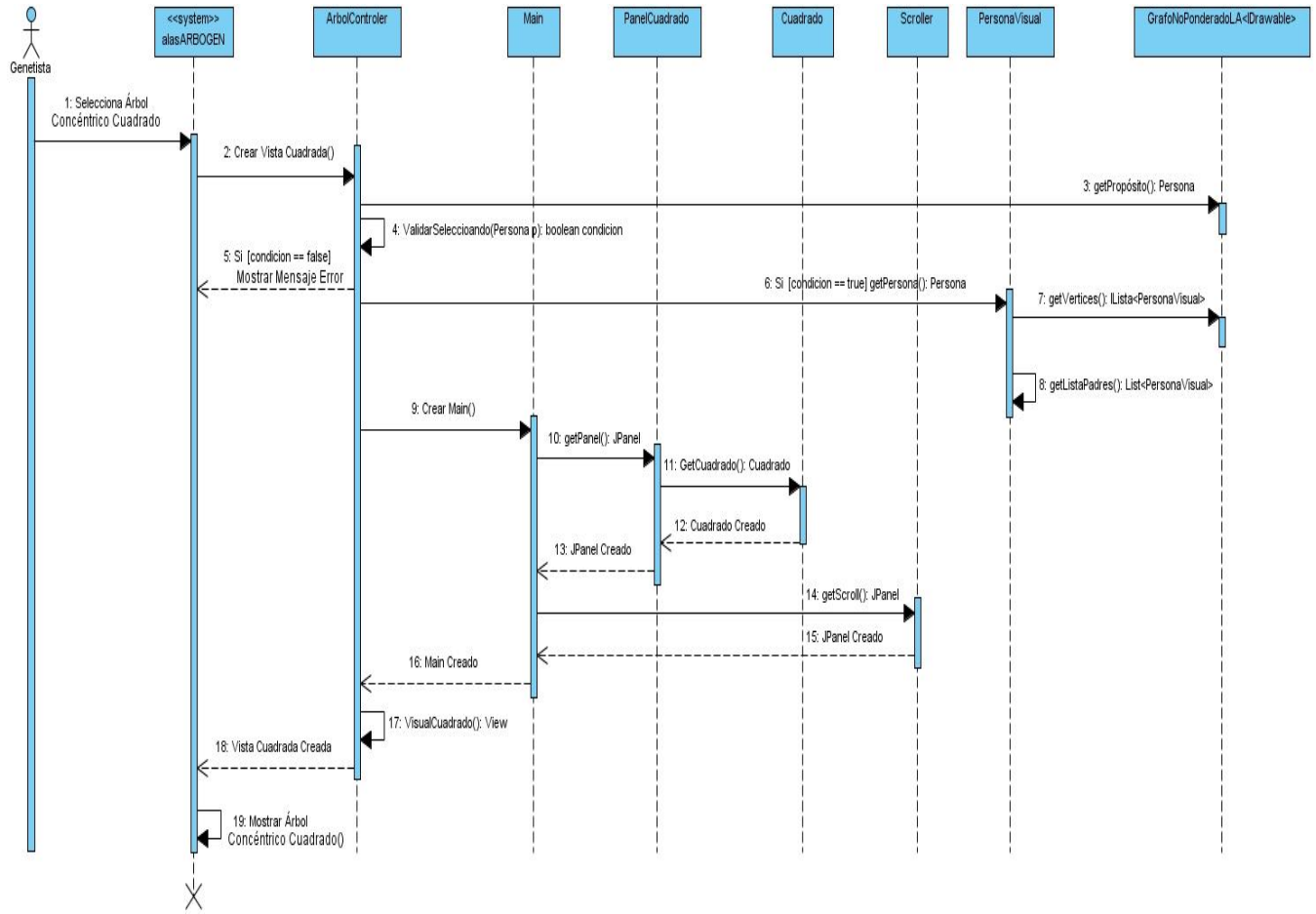


Figura 12. Diagrama de Secuencia del CU Representar Vistas. Escenario 1

2.8. Conclusiones parciales

En este capítulo se generaron los artefactos: requisitos funcionales y casos de uso. Fue definido el diseño del componente a desarrollar, realizándose los diagramas de clases del diseño de cada caso de uso y los diagramas de secuencia correspondientes. De igual manera quedó establecida la arquitectura y los patrones de diseño utilizados para el desarrollo del componente.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

En este capítulo se describe como fue implementado el componente. Se describen las principales funcionalidades implementadas y los resultados fundamentales de la implementación. Se realiza una descripción de los artefactos generados en esta etapa. Se describen las pruebas funcionales realizadas al componente una vez integrado a alasARBOGEN.

3.1. Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. (21) Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre sus elementos.

El diagrama se modeló en paquetes, concentrando los componentes según sus características. En el diagrama propuesto se definió el paquete ***TableEditable*** que concentra los componentes encargados de crear un **JTable** personalizado con celdas editables, con el fin de poder combinar, dividir e insertar componentes dentro de las propias celdas, debido a que el componente **JTable** de Java no está implementado con estos fines, que se convirtieron en necesidades de los componentes visuales a implementar. El paquete **Clases** contiene los componentes a dibujar en la representación concéntrica cuadrada. El paquete ***Vista*** contiene todos los componentes visuales que serán creados para mostrar la información al usuario. El paquete ***Entidad*** contiene los componentes definidos para la representación específica del dominio de los datos de entrada. Para la implementación de la lógica y las funcionalidades se definió el paquete ***Control*** encargado de encapsular el componente **ArbolControler**, definido para controlar las vistas definidas en el paquete ***Vista***, así como implementar las funcionalidades necesarias para el funcionamiento del componente a desarrollar. El paquete ***TDAArbolGen*** encapsula los componentes encargados de la implementación del tipo de dato abstracto **IArbolGen**, necesario para la implementación del algoritmo de ordenamiento del grafo entrado por parámetro.

A continuación, se muestra el Diagrama de Componentes:

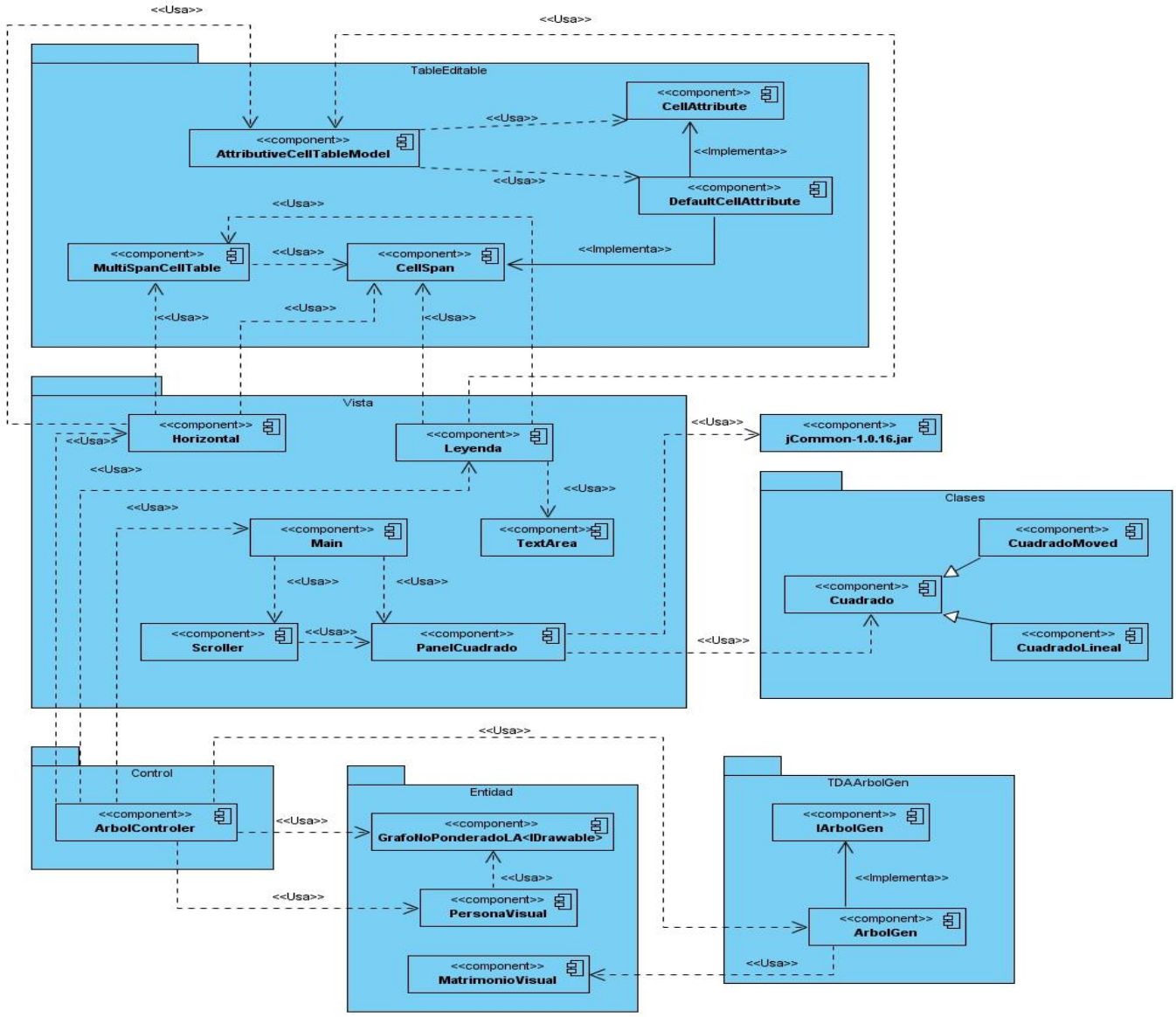


Figura 13. Diagrama de Componentes.

En el diagrama de componentes se puede comprobar el uso del estilo arquitectónico Modelo Vista Controlador. Como se muestra en la figura, la clase **ArbolControler** es la encargada de gestionar y controlar las funcionalidades del componente haciendo uso para ello de componentes visuales como **Leyenda** y **Main** que visualizan datos, a partir de entradas como un grafo no ponderado. Se puede evidenciar la utilización de la librería **Jcommon-1.0.16.jar**, que implementa funcionalidades para dibujar textos, necesario para la visualización de los datos en la representación concéntrica cuadrada.

3.2. Resultados fundamentales de la implementación.

- **Visualización de árboles genealógicos en sus formas Concéntrica Cuadrada y Descendente Horizontal.**

Se implementaron Vistas de Árboles Genealógicos utilizando las librerías gráficas de Java y en el caso particular de la representación concéntrica cuadrada se utilizó la librería Jcommon-1.0.16.jar para el trabajo con textos. El componente encargado de visualizar la representación concéntrica cuadrada es **PanelCuadrado** que tiene por entrada una lista de personas, de las cuales se mostrarán su nombre y apellidos. El componente que implementa la vista descendente Horizontal, es **Horizontal**, un JPanel, que tiene por entrada una **PersonaVisual**, a partir de la cual se muestra los nombres y apellidos de sus descendientes. Para obtener la lista de ancestros y las personas descendientes se realizó un recorrido recursivo sobre el grafo para seleccionar la familia de descendiente y ancestros. Las mismas tienen por entrada una persona o individuo que debe seleccionar el genetista para ejecutar sus vistas. De esta persona se localiza en la lista de adyacencia los individuos que tienen como relación "madre y padre" para el caso de los antepasados o relación de "hijo e hija" en el caso de los descendientes. Ejemplo de cómo estas vistas se integraron en alasARBOGEN lo constituyen las figuras de los anexos 10 y 11.

- **Visualización de una Leyenda estándar definida por el PSTF.**

La leyenda se diseñó a partir del estándar propuesto por el PSTF, mostrando los símbolos genéticos de los individuos de una familia determinada y plasma los comentarios realizados por el genetista sobre determinados individuos del árbol familiar. Esta funcionalidad recorre recursivamente el grafo visual y selecciona (sin duplicar), los símbolos usados en los individuos de la familia, creando una imagen de los mismos y mostrándolos en una tabla según el sexo. De igual manera recorre recursivamente el grafo familiar buscando los comentarios realizados a los individuos de la familia, mostrando al final de la leyenda dichos comentarios con el siguiente formato: número de la generación, número del individuo en dicha generación y nombre y apellidos del individuo. El anexo # 12 muestra el componente integrado en el sistema alasARBOGEN.

- **Algoritmo de ordenamiento de un árbol genealógico.**

Entre las principales metas del desarrollador se encuentra la implementación de un algoritmo para ordenar el árbol genealógico entrado por parámetros. Como característica fundamental del árbol

genealógico se encuentra que debido a los tipos de relaciones que se generan fue necesaria la utilización de una estructura de datos que lo soportara. Para ello se hace uso de un grafo no ponderado con lista de adyacencia. Como resultado de la investigación se creó una estructura de dato que fuera capaz de simular un árbol a partir de un grafo entrado. Para ello se diseñó la clase **MatrimonioVisual,** que modela la relación entre dos cónyuges, los cuales se encuentran en el mismo nivel de un árbol genealógico. Un **MatrimonioVisual** está compuesto por dos personas visuales (Hombre y Mujer). Un grafo que cumpla con los criterios estéticos de árboles genealógicos, no es más que un árbol de matrimonios, donde cada elemento hijo de dos padres puede o no, tener una esposa, y estos a su vez tener hijos. Un **MatrimonioVisual** puede ser solamente una persona, y en el caso de tener hijos serían la persona y su cónyuge. De esta manera, se concibió convertir la estructura de un grafo de **PersonaVisual** a un árbol de **MatrimonioVisual**.

El algoritmo para el ordenamiento se realizó basado en la propuesta del algoritmo de Stein y Benteler. (21) Se definen a continuación las consideraciones propuestas por los autores, enfocando las mismas a la solución particular del algoritmo a implementar.

- **Tamaño del área de dibujo**: el componente FamiliaVisual permite la representación de árboles de tamaños dinámicos a través del uso del **Scroll** horizontal y vertical. El tamaño será calculado por el eje de las ‘X’ calculando la distancia entre la ‘X’ del individuo más a la derecha y la ‘X’ del individuo más a la izquierda; y en el eje ‘Y’ según la cantidad de niveles.
- **Topología**: se ordenarán los individuos según su generación y bajo los criterios estéticos definidos en los fundamentos teóricos. Para ello se ubicarán en niveles con una separación mínima definida y se centrarán los padres para mantener la estética del árbol.
- **Estructuración del esquema**: se asignarán coordenadas a los diferentes subárboles de acuerdo con el tamaño establecido en el paso 1 y la topología determinada en el paso 2. Para esto se utilizará lo planteado por el algoritmo de Walker en su variante más eficiente planteada por Buchheim. (21) Para ello se transforma el grafo en un árbol con características idóneas para representar la jerarquía familiar.
- **Mapeo reverso de la topología**: se asignan valores de coordenadas a los objetos de tipo **PersonaVisual** pertenecientes a cada uno de los subárboles según el tamaño definido.

Entre otras consideraciones se tuvo en cuenta la eliminación de solapamientos de individuos, provocadas por la expansión del árbol a medida que se construye. Para ello se implementó un algoritmo que reposiciona los subárboles hermanos de un mismo nivel, excluyendo siempre al primer subárbol. Esta variante recursiva elimina los posibles solapamientos del dibujo. En el anexo # 13 se puede observar el ordenamiento del árbol genealógico después de integrado el componente a alasARBOGEN.

3.3. Pruebas

El desarrollo del software implica una serie de actividades de producción en las que las posibilidades de que aparezca la falibilidad humana son comunes. Los errores pueden empezar a darse desde el primer momento del proceso en el que los objetivos pueden estar especificados de forma errónea e imperfecta; así en los posteriores pasos del diseño y desarrollo. Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software ha de ir acompañado de una actividad que garantice la calidad.

Las pruebas son una actividad en la cual un sistema o componente, es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente.

La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación.

En esta disciplina el software es refinado a través de iteraciones en el ciclo de vida. El ciclo de vida de prueba se beneficia siguiendo un proceso iterativo equivalente. En cada iteración el equipo de desarrollo produce uno o más builds, cada builds es un candidato potencial para probar. Los objetivos del equipo de desarrollo difieren de una iteración a otra. El equipo de prueba estructura su prueba de acuerdo con los objetivos de la iteración. (23)

- **Nivel de Prueba**

La Prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se distinguen diferentes niveles de pruebas en dependencia de las características del sistema a

implementar. (23) Al componente propuesto se le realizaron las pruebas a nivel de desarrollador para detectar posibles errores en las funcionalidades.

- **Prueba de Desarrollador.**

Es la prueba diseñada e implementada por el equipo de desarrollo. Tradicionalmente estas pruebas han sido consideradas solo para la prueba de unidad, aunque en la actualidad en algunos casos pueden ejecutar pruebas de integración. Se recomienda que estas pruebas cubran más que las pruebas de unidad, debido a la importancia de encontrar posibles errores o respuestas inesperadas de las funcionalidades implementadas. El desarrollador tiene una labor indispensable en las mismas, proporcionando una validación a los componentes implementados. (23)

- **Tipo de prueba**

El tipo de prueba a realizar son las pruebas de funcionalidad o pruebas funcionales, pruebas que se realizan fijando su atención en la validación de las funciones, métodos, servicios, caso de uso. Estas permiten medir hasta donde fueron cumplidos los requerimientos, con las funcionalidades implementadas, al mismo tiempo, de validar las posibles respuestas del sistema ante situaciones determinadas, entre otras funciones.

- **Método de Prueba**

Como método de prueba se realizó el método de Caja Negra. La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

Se centran principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. (23)

3.3.1. Diseño de casos de prueba.

A partir del momento en que el equipo de desarrollo ha elaborado los casos de usos del sistema se pueden elaborar los casos de pruebas que no son más que un conjunto de entradas con datos de

prueba, unas condiciones de ejecución, y unos resultados esperados con el propósito de identificar y comunicar las condiciones que se llevarán a cabo en la prueba. Los casos de pruebas son necesarios para verificar la aplicación exitosa y aceptable de los requisitos del producto (que se especifican en los casos de uso).

A continuación se describen los casos de pruebas realizados al componente visual para la representación de árboles genealógicos.

- **CUS: "Ordenar Árbol Genealógico"**

- **Diseño del caso de prueba correspondiente al CUS: Ordenar Árbol Genealógico.**

Nombre de la Sección	Escenario	Descripción	Flujo donde empieza
SC 1: Ordenar Árbol Genealógico.	EC 1.1: El usuario selecciona con un clic la opción Ordenar Árbol del menú Edición.	El sistema responde mostrando el árbol genealógico ordenado.	Principal.

- **Descripción de las variables del CUS "Ordenar Árbol Genealógico"**

No.	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Ordenar Árbol	JMenuItem	No	Se visualiza el árbol genealógico ordenado.

- **Matriz de Datos. CUS "Ordenar Árbol Genealógico"**

ID del escenario	Escenario	Variable 1: JMenuItem	Respuesta del sistema	Resultado de la Prueba
1.1	El usuario selecciona con un clic la opción Ordenar Árbol del menú Edición.	N/A	El sistema muestra el árbol genealógico ordenado.	Satisfactoria.

➢ **CUS "Representar Vistas"**

- **Diseño del caso de prueba correspondiente al CUS: "Representar Vistas".**

Nombre de la Sección	Escenario	Descripción	Flujo donde empieza
SC 1: Visualizar Árbol Concéntrico Cuadrado.	1.1 El genetista selecciona un individuo y da clic en la opción Árbol Concéntrico Cuadrado.	El sistema responde mostrando el árbol genealógico concéntrico cuadrado.	Principal
	1.2 El genetista no selecciona un individuo y da clic en la opción Árbol Concéntrico Cuadrado.	El sistema responde mostrando el mensaje: "Seleccione un individuo".	Alterno
SC 2: Visualizar Árbol Descendiente Horizontal.	2.1 El genetista selecciona un individuo y da clic en la opción Árbol Descendente Horizontal.	El sistema responde mostrando el árbol genealógico descendente horizontal.	Principal
	2.2 El genetista no selecciona un individuo y da clic en la opción Árbol Descendente Horizontal.	El sistema responde mostrando el mensaje: "Seleccione un individuo".	Alterno

- **Descripción de las variables del CUS "Representar Vistas"**

No.	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Árbol Concéntrico Cuadrado.	JmenuItem.	No	Se visualiza árbol concéntrico cuadrado.
2	Árbol Descendente Horizontal.	JmenuItem.	No	Se visualiza árbol descendente horizontal.

- **Matriz de Datos. CUS "Representar Vistas". SC1: Visualizar Árbol Concéntrico Cuadrado.**

ID del escenario	Escenario	Variable 1: JMenuItem	Respuesta del sistema	Resultado de la Prueba
1.1	El genetista selecciona un individuo y da clic en la opción Árbol Concéntrico Cuadrado.	N/A	El sistema muestra el árbol genealógico concéntrico cuadrado con errores en los datos cargados.	Insatisfactoria.
1.2	El genetista no selecciona un individuo y da clic en la opción Árbol Concéntrico Cuadrado.	N/A	El sistema muestra el mensaje: "Seleccione un individuo".	Satisfactoria.

- **Matriz de Datos. CUS "Representar Vistas". SC2: Visualizar Árbol Descendente Horizontal.**

ID del escenario	Escenario	Variable 1: JMenuItem	Respuesta del sistema	Resultado de la Prueba
2.1	El genetista selecciona un individuo y da clic en la opción Árbol Descendente Horizontal	N/A	El sistema muestra el árbol genealógico descendente horizontal.	Satisfactoria.
2.2	El genetista no selecciona un individuo y da clic en la opción Árbol Descendente Horizontal.	N/A	El sistema no responde a la solicitud realizada, no muestra el mensaje: "Seleccione un individuo".	Insatisfactoria.

➢ **CUS: "Visualizar Leyenda"**

- **Diseño del caso de prueba correspondiente al CUS: Visualizar Leyenda.**

Nombre de la Sección	Escenario	Descripción	Flujo donde empieza
SC 1: Visualizar Leyenda	EC 1.1: El usuario selecciona con un clic la opción Leyenda del Menú "Ver".	El sistema responde mostrando en una nueva pestaña una leyenda con los significados de los nomencladores utilizados en la simbología médica, se identifican de esta manera los significados genéticos de cada símbolo.	Principal

- **Descripción de las variables del CUS "Visualizar Leyenda"**

No.	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Leyenda	JMenuItem	No	Se visualiza La leyenda de la familia activa.

- **Matriz de Datos. CUS "Visualizar Leyenda"**

ID del escenario	Escenario	Variable 1: JMenuItem	Respuesta del sistema	Resultado de la Prueba
1.1	El usuario selecciona con un clic la opción Leyenda del Menú "Ver".	N/A	El sistema muestra en una nueva pestaña una leyenda con los significados de los nomencladores utilizados en la simbología médica, se identifican de esta manera los significados genéticos de cada símbolo.	Satisfactoria.

La siguiente tabla muestra las No Conformidades detectadas durante la etapa de pruebas, las cuales fueron resueltas satisfactoriamente.

- **Registro de defectos y dificultades detectados.**

No.	No conformidad	Aspecto correspondiente	Etapa de Detección	Significativa	No Significativa	Estado NC	Respuesta
1	En el CUS "Representar Vistas" en la SC1, cuando el genetista no selecciona un individuo y da clic en la opción "Árbol Concéntrico Cuadrado" se muestra la vista pero con errores en los datos cargados.	Vista "Árbol Concéntrico Cuadrado".	Pruebas de funcionalidad.		x	PD 24/04/11 Resuelta 6/05/11	El sistema una vez seleccionado un individuo y ejecutada la opción "Árbol Concéntrico Cuadrado" visualiza la vista cuadrada correctamente, con los datos válidos.
2	En el CUS "Representar Vistas" en la SC2, cuando el genetista selecciona un individuo y da clic en la opción "Árbol Descendente Horizontal" el sistema no responde a la acción realizada cuando debería enviar un mensaje de error advirtiendo la necesidad de que sea seleccionado un individuo.	Vista "Árbol Descendente Horizontal ".	Pruebas de funcionalidad.		x	PD 29/04/11 Resuelta 2/05/11	El sistema una vez seleccionada la opción "Árbol Descendente Horizontal", sin haber sido seleccionado previamente un individuo, muestra el mensaje de error: "Seleccione un individuo".

3.4. Conclusiones parciales

En este capítulo se realizaron los artefactos Diagrama de Componentes evidenciando el uso del estilo arquitectónico Modelo-Vista-Controlador. Se realizó el análisis de los algoritmos fundamentales que arrojaron funcionalidades reales. Se definieron: el nivel de pruebas, el tipo de pruebas y el método de prueba para la validación de las funcionalidades propuestas. Además, se mostraron los casos de pruebas fundamentales, con los resultados obtenidos y las acciones realizadas frente a las no conformidades.

CONCLUSIONES

- El estudio de las normas y sistemas para la representación de árboles genealógicos existentes permitieron la definición de un conjunto de normas para la visualización correcta de árboles genealógicos, siguiéndose como estándar el propuesto por el PSTF.

- Se diseñaron las clases necesarias para la modelación del componente visual, utilizando el estilo arquitectónico modelo vista controlador, permitiendo futuros cambios en la lógica del negocio realizándose los mismos solamente sobre las clases implicadas, sin necesidad de cambiar toda la implementación.

- Se implementaron e integraron a alasARBOGEN los componentes necesarios para la representación de árboles genealógicos desde diferentes perspectivas.

- Las pruebas funcionales desarrolladas permiten afirmar que el componente desarrollado cumple con los requerimientos propuestos, siendo corregidas las no conformidades detectadas en el proceso.

RECOMENDACIONES

- Incluir la leyenda como parte de los reportes del árbol genealógico.
- Implementar funciones para unir árboles genealógicos que permitan asociar familias con relaciones.

BIBLIOGRAFÍA

1. **Babylon. Arbol Genealógico.** [En línea] [Citado el: 14 de 11 de 2010.] http://www.babylon.com/definition/%C3%A1rbol_geneal%C3%B3gico/spanish.

2. **Benítez Morales, Erick Alexander.** Herramienta case. "ArgoUML". [En línea] 2006. [Citado el: 4 de Noviembre de 2010.] http://erickbenitez.iespana.es/Herramientas%20case.pdf.

3. **BitGen II.** Árboles genealógicos y heráldica desde el PC. [En línea] [Citado el: 12 de 10 de 2010.] http://www.idg.es/pcworld/BitGen-II.-%C3%81rboles-genealogicos-y-heraldica-desde-/art131220.htm.

4. **BitGen II.** Heráldica y Árboles Genealógicos. [En línea] [Citado el: 12 de 10 de 2010.] http://www.quierosoft.com/index.html?target=p_67.html&lang=es.

5. **Blog de WordPress.com.** [En línea] 14 de Febrero de 2008. [Citado el: 5 de Noviembre de 2010.] http://arturoweb.wordpress.com/2008/02/14/scrum-metodologia-agil-de-desarrollo/.

6. **BOC Business Objectives Consulting Ibérica, S.L.U.** [En línea] [Citado el: 4 de Noviembre de 2010.] http://www.boc-group.com/documents/products/BPMN_en_ADONIS_Flyer.pdf.

7. **Boost Productivity with Innovative and Intuitive Technologies.** [En línea] [Citado el: 4 de Noviembre de 2010.] http://www.visual-paradigm.com/aboutus/10reasons.jsp.

8. **CEDS e Learning.** [En línea] [Citado el: 4 de Noviembre de 2010.] http://ceds.nauta.es/informes/case04.htm.

9. **Congreso Iberoamericano de Seguridad Informática.** Hacia la definición de Procesos de Negocios Seguros basados en una Arquitectura Dirigida por Modelos. [En línea] Noviembre de 2005. http://cibsi05.inf.utfsm.cl/presentaciones/sesion11/Hacia_una_definicion_de_procesos_de_negocios_seguros.pdf.

10. **Cyrillic.** Para dibujo de árboles genéticos y manejo de datos. [En línea] [Citado el: 12 de 11 de 2010.] http://www.softwarecientifico.com/paginas/cyrillic.htm.

11. **Eclipse .** [En línea] [Citado el: 5 de Noviembre de 2010.] http://amap.cantabria.es/confluence/display/DEV/Eclipse.

12. **EVA.** [En línea] [Citado el: 23 de Febrero de 2011.] http://eva.uci.cu/file.php/259/Curso_2010-2011/Semana_2/Conferencia_2/Materiales_Basicos/Estilos_y_Patrones_en_la_Estrategia_de_Arquitectura_de_Microsoft.pdf.

13. **EVA.** [En línea] [Citado el: 20 de Marzo de 2011.] http://eva.uci.cu/file.php/259/Curso_2010-2011/Semana_9/Conferencia_7/Materiales_Basicos/Sobre_la_disciplina_de_Prueba.pdf.

14. **GenoPro.** Programa de Árboles Familiares. [En línea] [Citado el: 15 de 10 de 2010.] http://www.taringa.net/posts/downloads/1517489/GenoPro-2_0_1_4---Programa-de-%C3%A1rboles-familiares.html.

15. **Gestión de Proyectos.** OpenUP como alternativa metodológica para proyectos pequeños de software. [En línea] [Citado el: 02 de 12 de 2010.] http://kasyles.blogspot.com/2008/09/openup-como-alternativa-metodolgica.html.

16. **González Barbone, Víctor A.** XP: Extreme Programming (Programación Extrema. [En línea] [Citado el: 5 de Noviembre de 2010.] http://iie.fing.edu.uy/~nacho/blandos/seminario/XProg1.html.

17. **Grupo Soluciones GSINNOVA.** [En línea] [Citado el: 4 de Noviembre de 2010.] http://www.rational.com.ar/herramientas/roseenterprise.html.

18. **IEEE.** IEEE Standard Glossary of Software Engineering Terminology.

19. **Improving Walker´s Algorithm to Run in Linear Time.** Buchheim, Cristoph, Junger, Michael y Leipert, Sebastian.

20. **Ingeniería del Software.** Tema 12: Modelo de implementación. [En línea] [Citado el: 5 de 12 de 2010.] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema.pdf. Módulo 2.

21. **Introducción Ingeniería de Software.** Análisis de la clase (RUP parte 2 y OPEN/UP). [En línea] http://eproano334.blogspot.es/tags/Vida/.

22. **Lago Torres, Manuel.** Introducción al diseño con patrones. [En línea] [Citado el: 26 de Noviembre de 2010.] http://www.elrincondelprogramador.com/default.asp?id=29&pag=articulos/leer.asp.

23. **M, Itzcoalt Álvarez.** Desarrollo Ágil con SCRUM. [En línea] [Citado el: 4 de Noviembre de 2010.] http://www.sg.com.mx/sg07/presentaciones/Mejora%20de%20procesos/SG07.P02.Scrum.pdf.

24. **Mora, Beatriz, Ruiz, Francisco y García, Félix y Piattini Mario.** Experiencia en transformación de modelos de procesos de negocios desde BPMN a XPDL. [En línea] [Citado el: 5 de Noviembre de 2010.] http://kuainasi.ciens.ucv.ve/ideas07/documentos/articulos_ideas/Articulo35.pdf.

25. **Net Beans.** Información general sobre la versión. [En línea] [Citado el: 15 de 10 de 2010.] http://netbeans.org/community/releases/69/index_es.html.

26. **Obeso Agüera, Ivan.** Estudio de herramientas CASE de análisis y diseño orientado a objetos. [En línea] [Citado el: 4 de Noviembre de 2010.] http://petra.euitio.uniovi.es/~i2133798/hd/archivos/disenio/estudioHerramientasDisenio.pdf.

27. **Open UP.** [En línea] [Citado el: 5 de Noviembre de 2010.] http://epf.eclipse.org/wikis/openup/.

28. **Programación Extrema.** [En línea] [Citado el: 5 de Noviembre de 2010.] http://kmels.net/files/2009/uvg/cc2003/Resources/Contenidos/XP/xp.pdf.

29. **Rueda Chacón, Julio César.** Aplicación de la Metodología RUP para el Desarrollo Rápido de Aplicaciones Basado en el Estándar J2EE. [En línea] Marzo de 2006. [Citado el: 4 de Noviembre de 2010.] Tesis (Ingeniero en Ciencias y Sistemas). . http://biblioteca.usac.edu.gt/tesis/08/08_7691.pdf.

30. **Sitio de Descargas de Software.** [En línea] [Citado el: 4 de Noviembre de 2010.] http://www.freedownload manager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

31. **Szyperski, Clemens.** Component Software: Beyond Object-Oriented Programming. Segunda. 2002. págs. 20-22.

32. **Tan Nozawa, Jaime M.** . PHP experto. [En línea] 9 de Diciembre de 2008. [Citado el: 4 de Noviembre de 2010.] http://phpexperto.blogspot.com/2008/12/comendo-con-uml-y-php.html.

33. **Visual Paradigm for UML.** [En línea] [Citado el: 02 de 11 de 2010.] http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

34. **Wesley, Ed. Addison.** Extreme Programming Explained.Embrace Change . Notas sobre Metodologías Ágiles. [En línea] 2 de Abril. http://migueljaque.com/index.php/metodologias/xp/29-xp/63-programacionextrema.

35. **Wikipedia.** Wikipedia.ORG. [En línea] [Citado el: 9 de 11 de 2010.] http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29.

REFERENCIAS BIBLIOGRÁFICAS

1. **Babylon.** Arbol Genealógico. [En línea] [Citado el: 14 de 11 de 2010.] http://www.babylon.com/definition/%C3%A1rbol_geneal%C3%B3gico/spanish.

2. **Szyperski, Clemens.** *Component Software: Beyond Object-Oriented Programming.* Segunda. 2002. págs. 20-22.

3. **Improving Walker´s Algorithm to Run in Linear Time**. **Buchheim, Cristoph, Junger, Michael y Leipert, Sebastian.**

4. **GenoPro**. *Programa de Árboles Familiares.* [En línea] [Citado el: 15 de 10 de 2010.] http://www.taringa.net/posts/downloads/1517489/GenoPro-2_0_1_4---Programa-de-%C3%A1rboles-familiares.html.

5. **BitGen II.** *Heráldica y Árboles Genealógicos.* [En línea] [Citado el: 12 de 10 de 2010.] http://www.quierosoft.com/index.html?target=p_67.html&lang=es.

6. **BitGen II.** *Árboles genealógicos y heráldica desde el PC.* [En línea] [Citado el: 12 de 10 de 2010.] http://www.idg.es/pcworld/BitGen-II.-%C3%81rboles-genealogicos-y-heraldica-desde-/art131220.htm.

7. **Cyrillic.** *Para dibujo de árboles genéticos y manejo de datos.* [En línea] [Citado el: 12 de 11 de 2010.] http://www.softwarecientifico.com/paginas/cyrillic.htm.

8. **Wikipedia.** Wikipedia.ORG. [En línea] [Citado el: 9 de 11 de 2010.] http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29.

9. **Net Beans.** *Información general sobre la versión.* [En línea] [Citado el: 15 de 10 de 2010.] http://netbeans.org/community/releases/69/index_es.html.

10. **Eclipse** . [En línea] [Citado el: 5 de Noviembre de 2010.] http://amap.cantabria.es/confluence/display/DEV/Eclipse.

11. **Sitio de Descargas de Software**. [En línea] [Citado el: 4 de Noviembre de 2010.] http://www.freedownload manager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

12. **CEDS e Learning**. [En línea] [Citado el: 4 de Noviembre de 2010.] http://ceds.nauta.es/informes/case04.htm.

13. **Boost Productivity with Innovative and Intuitive Technologies**. [En línea] [Citado el: 4 de Noviembre de 2010.] http://www.visual-paradigm.com/aboutus/10reasons.jsp.

14. **Visual Paradigm for UML**. [En línea] [Citado el: 02 de 11 de 2010.] http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

15. **Open UP**. [En línea] [Citado el: 5 de Noviembre de 2010.] http://epf.eclipse.org/wikis/openup/.

16. **Introducción Ingeniería de Software**. *Análisis de la clase (RUP parte 2 y OPEN/UP).* [En línea] http://eproano334.blogspot.es/tags/Vida/.

17. **Gestión de Proyectos**. *OpenUP como alternativa metodológica para proyectos pequeños de software.* [En línea] [Citado el: 02 de 12 de 2010.] http://kasyles.blogspot.com/2008/09/openup-como-alternativa-metodolgica.html.

18. **IEEE.** IEEE Standard Glossary of Software Engineering Terminology.

19. **EVA**. [En línea] [Citado el: 23 de Febrero de 2011.] http://eva.uci.cu/file.php/259/Curso_2010-2011/Semana_2/Conferencia_2/Materiales_Basicos/Estilos_y_Patrones_en_la_Estrategia_de_Arquitectura_de_Microsoft.pdf.

20. **Lago Torres, Manuel.** Introducción al diseño con patrones. [En línea] [Citado el: 26 de Noviembre de 2010.] http://www.elrincondelprogramador.com/default.asp?id=29&pag=articulos/leer.asp.

21. **Ingeniería del Software**. *Tema 12: Modelo de implementación.* [En línea] [Citado el: 5 de 12 de 2010.] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema.pdf. Módulo 2.

22. **Buchheim, Chritoph, Jûnger, Michael y Leipert, Sebastian.** CiteSeer*beta. [En línea] [Citado el: 12 de Enero de 2011.] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.8757.

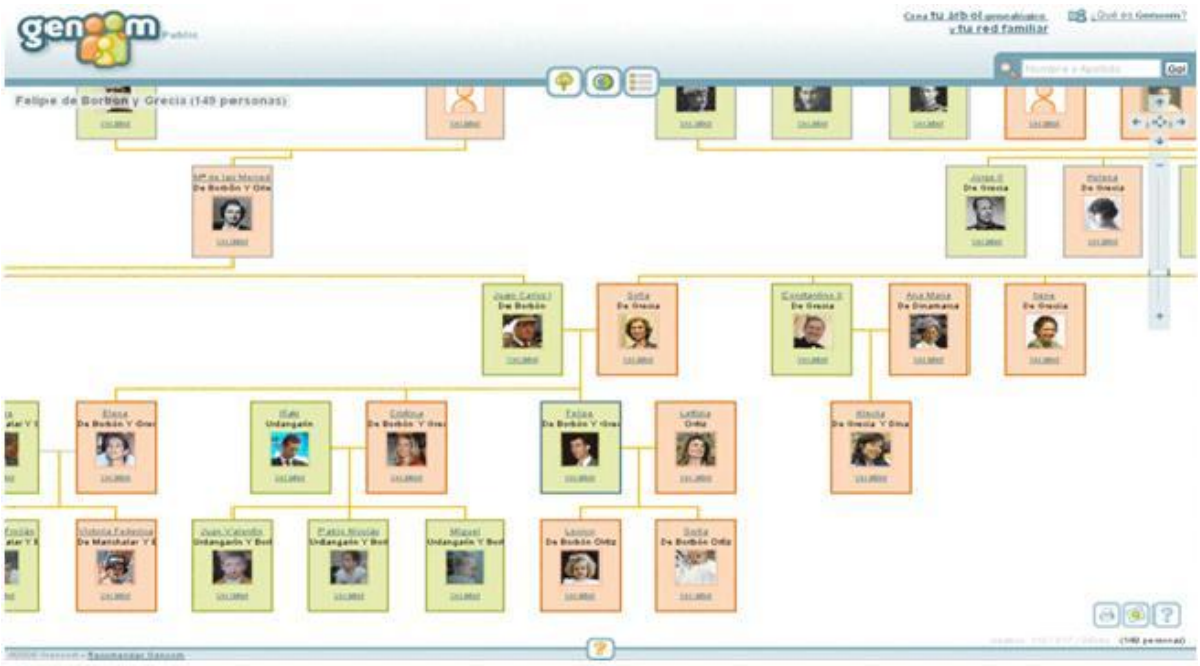
23. **EVA.** [En línea] [Citado el: 20 de Marzo de 2011.] http://eva.uci.cu/file.php/259/Curso_2010-2011/Semana_9/Conferencia_7/Materiales_Basicos/Sobre_la_disciplina_de_Prueba.pdf.

ANEXOS

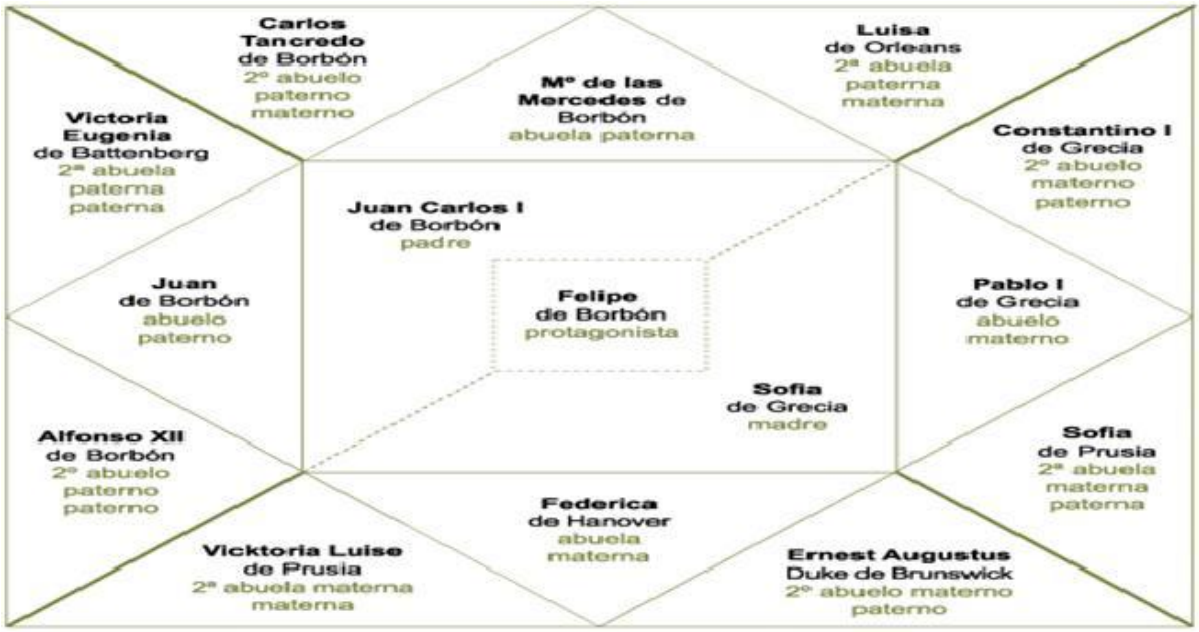
Anexo # 1. Vista de árbol genealógico clásico horizontal en tabla de crecimiento de izquierda a derecha (solo de representación consanguínea).

1° generación	2° generación	3° generación	4° generación
Juan de Borbón	Juan Carlos I de Borbón	Elena de Borbón y Grecia	Felipe J. Froilán De Marichalar Y Borbón
			Victoria Federica De Marichalar Y Borbón
		Cristina de Borbón y Grecia	Juan Valentín Urdangarín Y Borbón
			Pablo Nicolás Urdangarín Y Borbón
			Miguel Urdangarín Y Borbón
		Felipe de Borbón y Grecia	Leonor De Borbón Ortiz
			Sofía De Borbón Ortiz
	Alfonso de Borbón		
	Margarita de Borbón	Alfonso Zurita de Borbón	
		María Zurita de Borbón	
	Pilar de Borbón	Simoneta - Luisa Gomez-Acebo de Borbón	Luis Juan Fernández Gomez-Acebo
			Mª de las Mercedes Fernández Gomez-Acebo
			Pablo Fernández Gomez-Acebo
		Juan Filiberto Nicolás Gomez-Acebo de Borbón	
		Bruno Alejandro Gomez-Acebo de Borbón	Alejandro Juan Gomez-Acebo Cano
			Guillermo Gomez-Acebo Cano

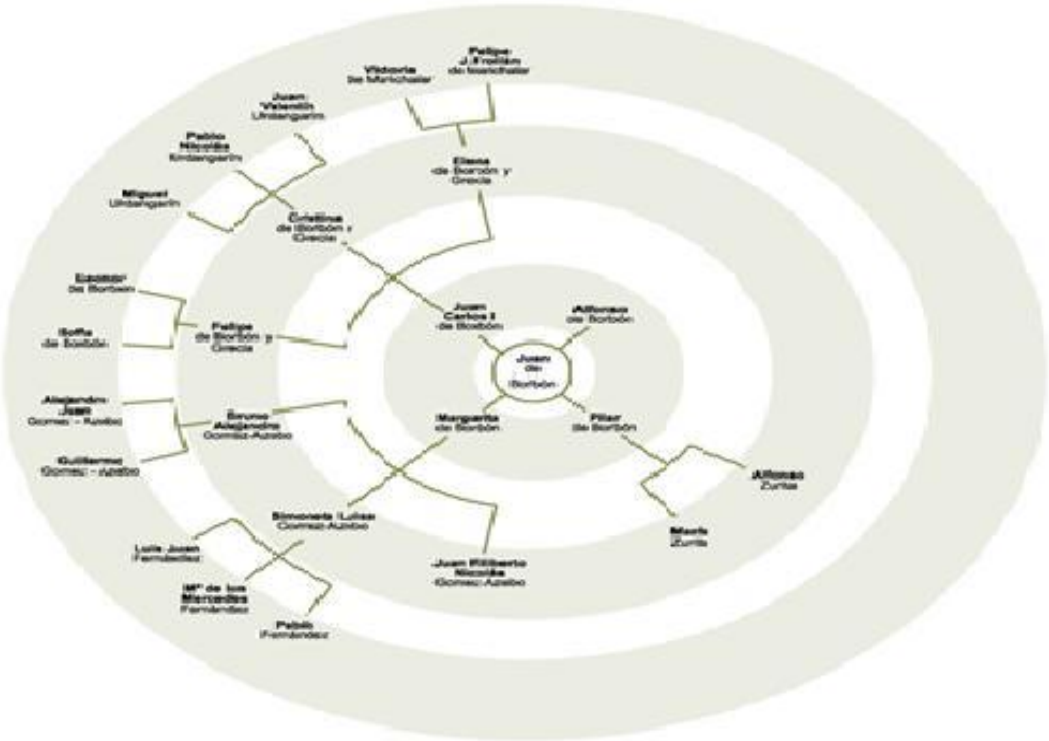
Anexo # 2. Vista de árbol genealógico clásico vertical en esquema de crecimiento de inferior a superior.



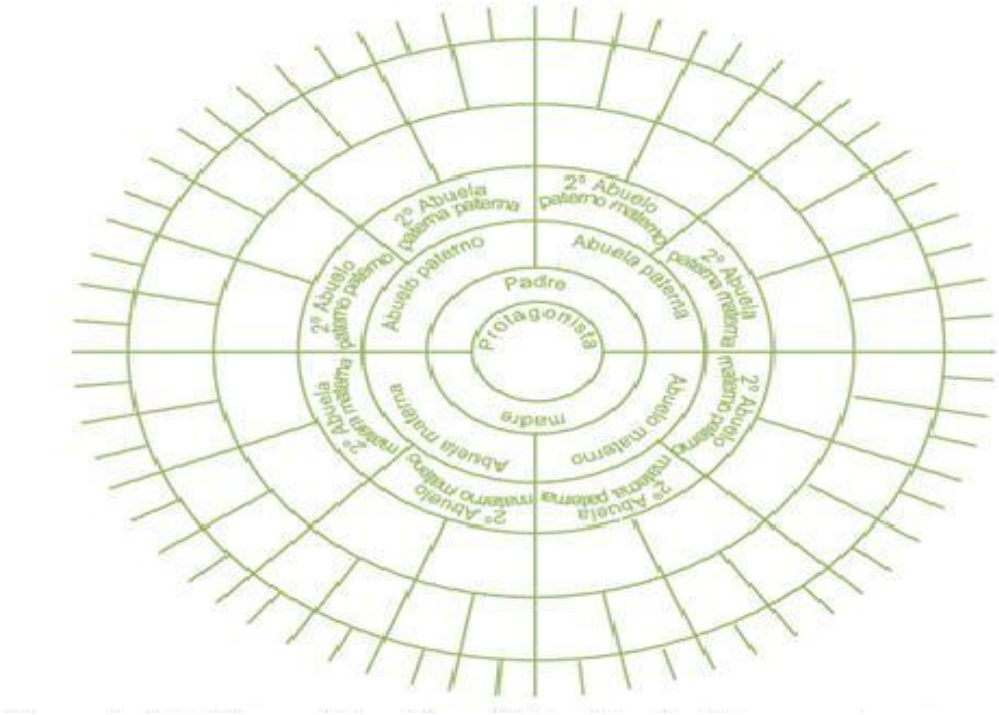
Anexo # 3. Vista de la representación concéntrica cuadrada.



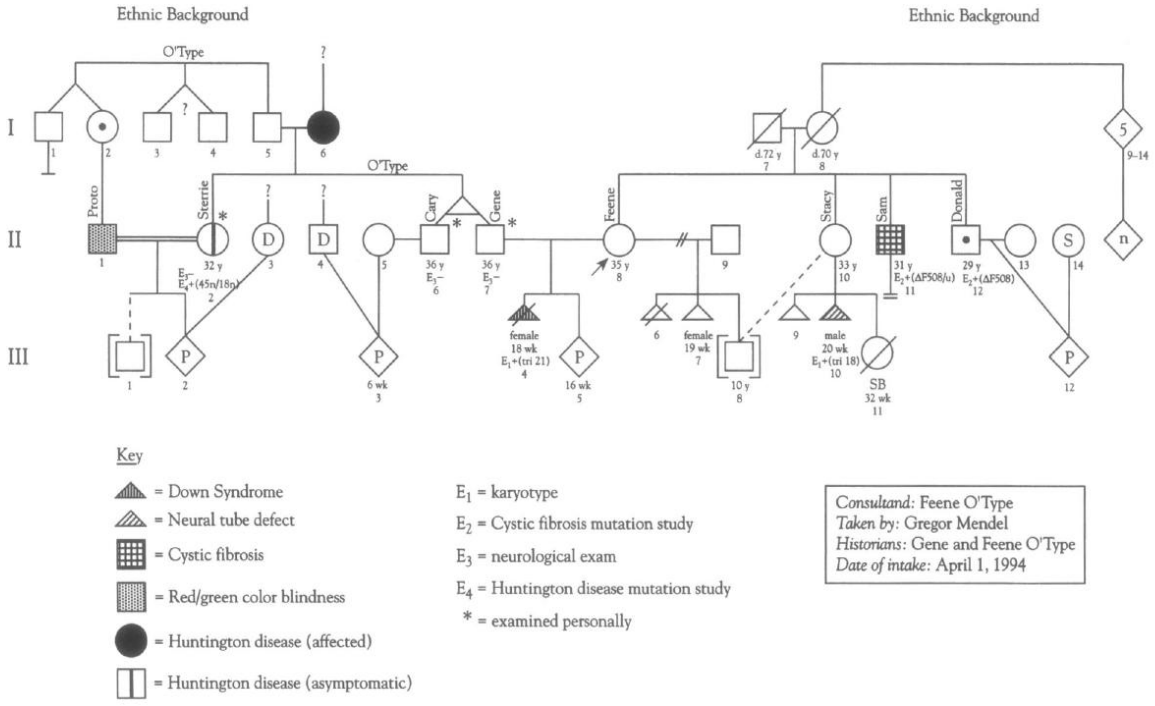
Anexo # 4. Vista de la representación concéntrica circular divergente.



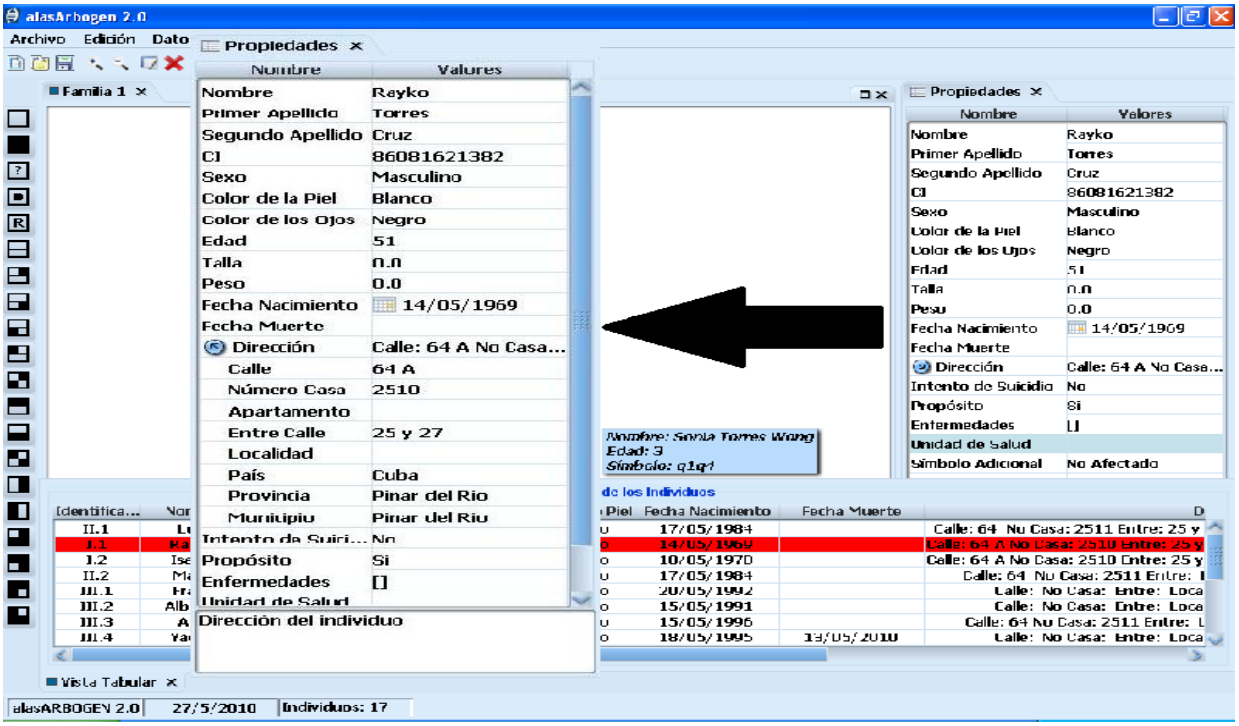
Anexo # 5. Vista de la representación concéntrica circular convergente.



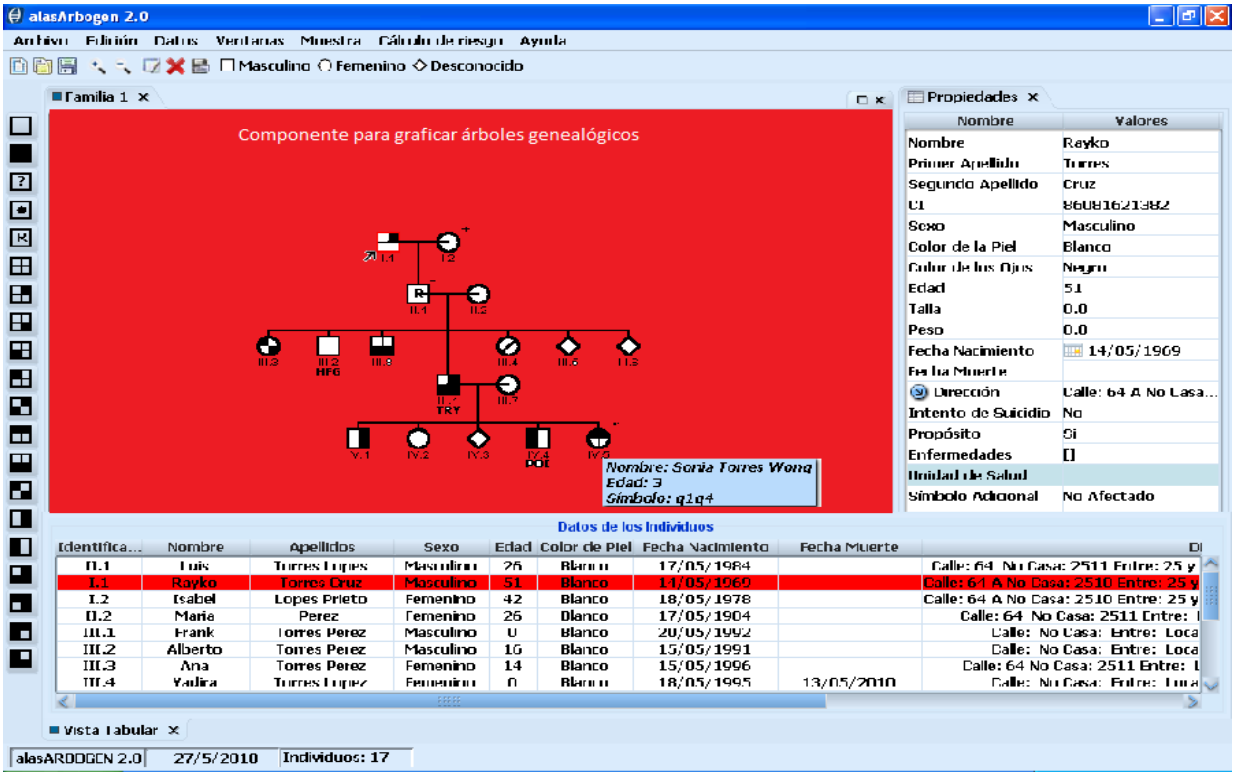
Anexo # 6. Propuesta del estándar implementado por el PSWG.



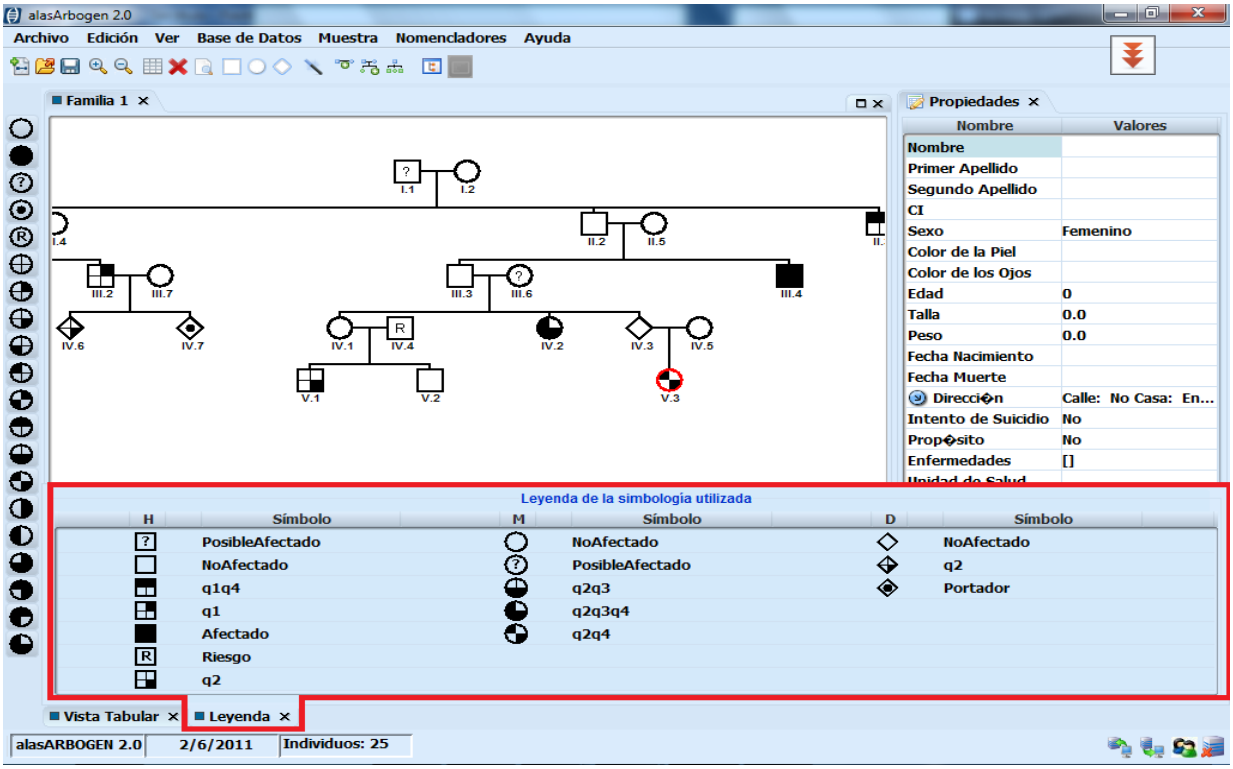
Anexo # 7. Localización del componente Visor de Propiedades.



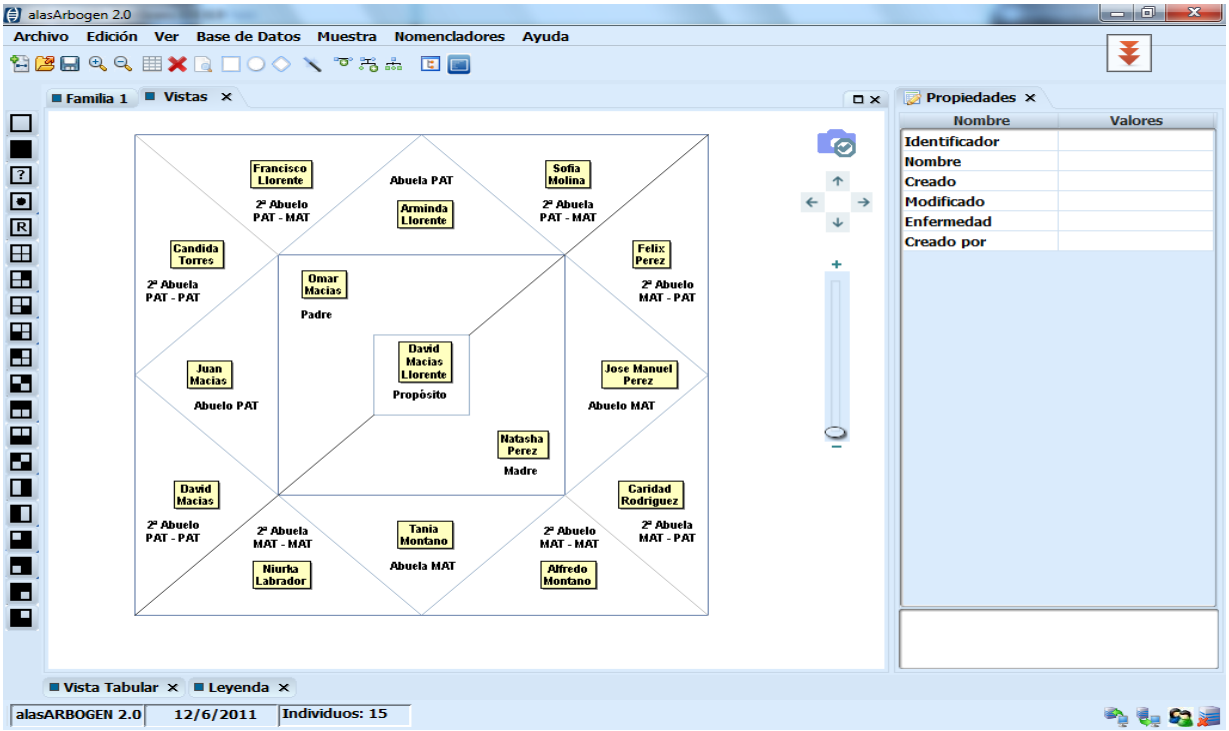
Anexo # 8. Lugar en la interfaz de alasARBOGEN donde se ve la integración del componente.



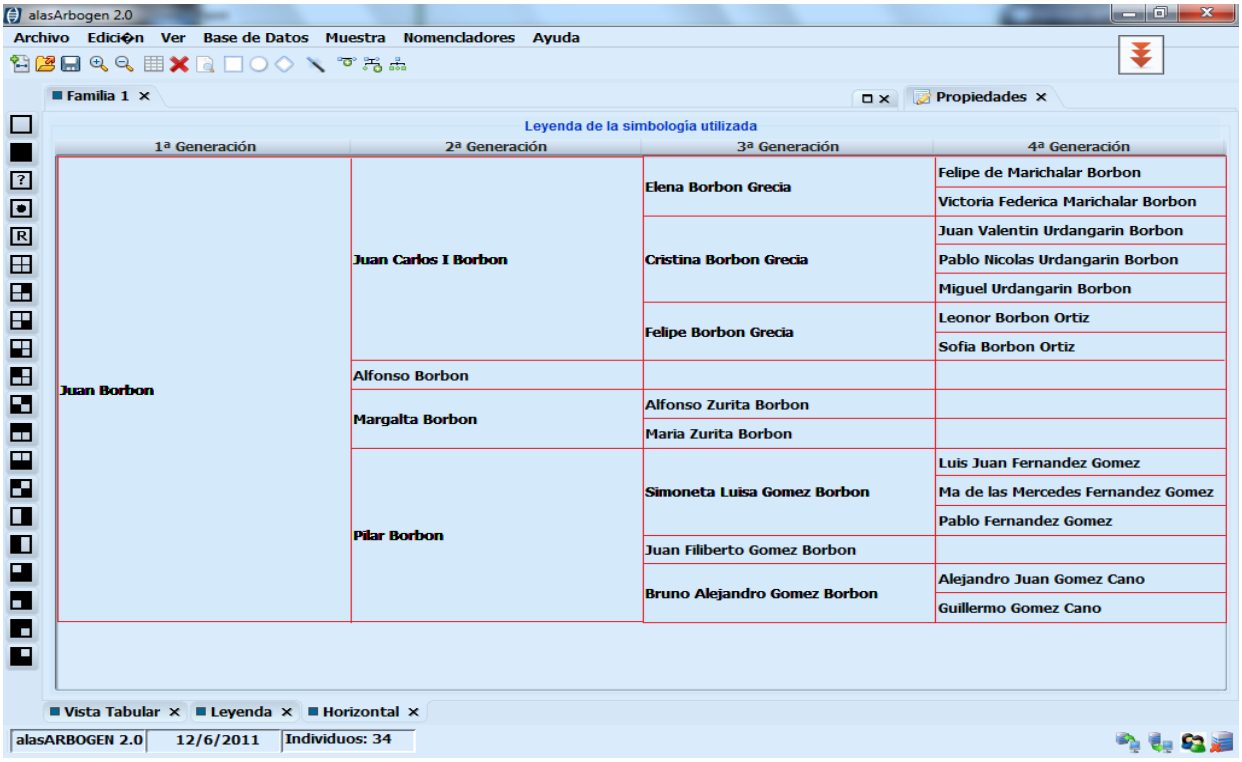
Anexo # 9. Lugar en la interfaz de alasARBOGEN donde se integra el componente “Leyenda”.



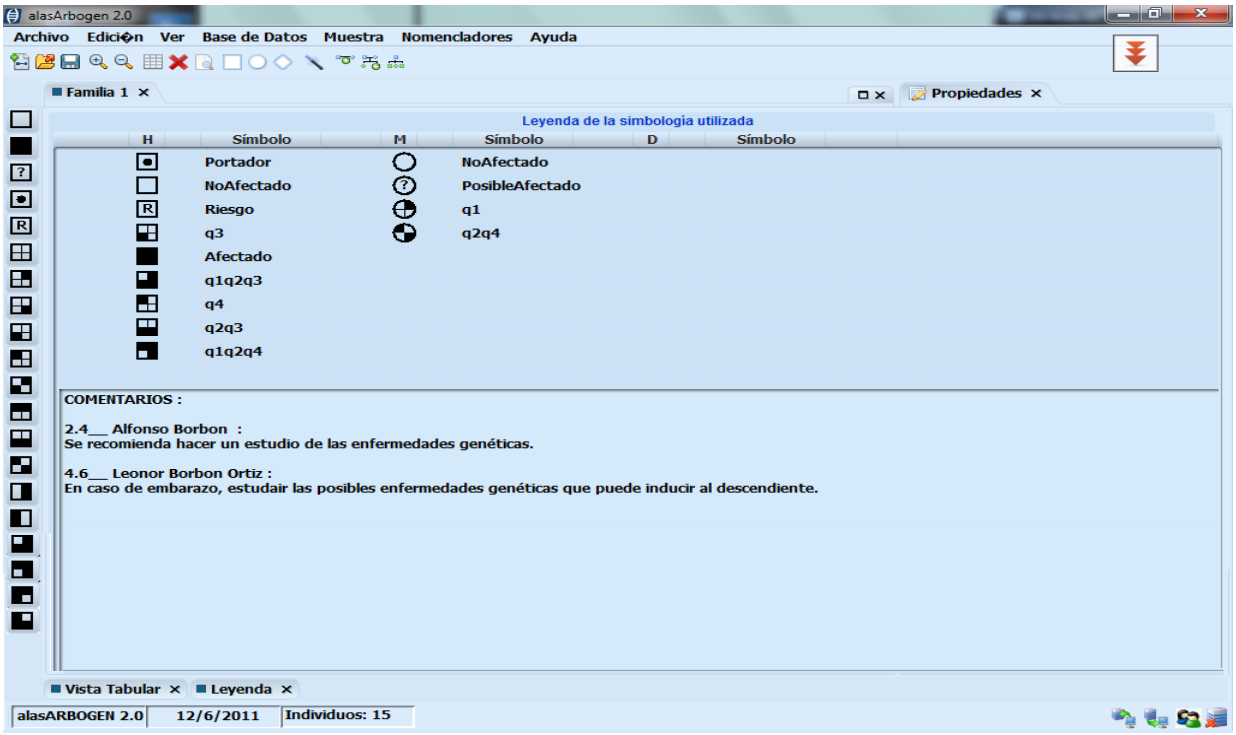
Anexo # 10. Representación de la Vista del Árbol Genealógico Concéntrico Cuadrado.



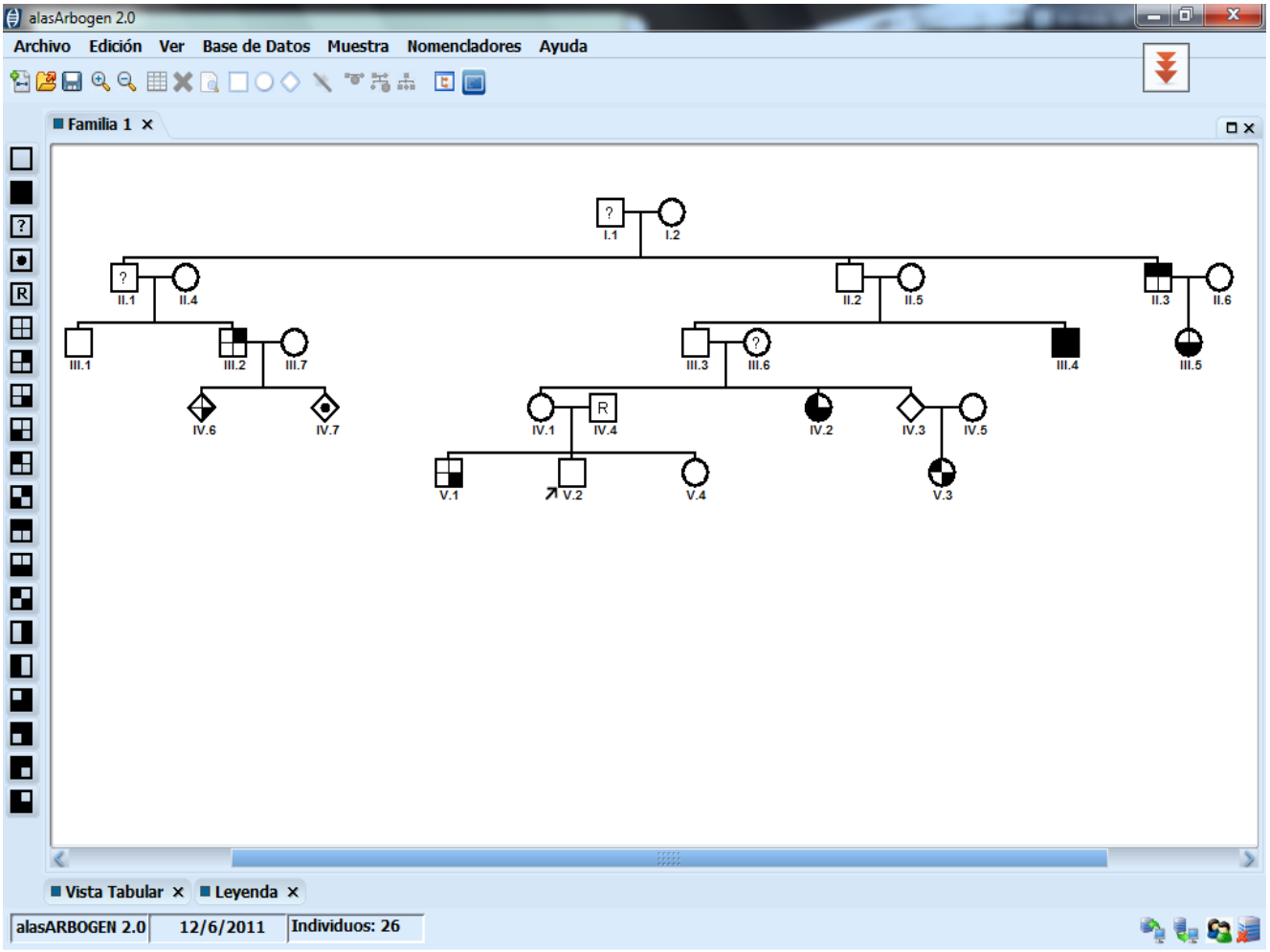
Anexo # 11. Representación de la Vista del Árbol Genealógico Descendente Horizontal.



Anexo # 12. Representación de la Leyenda



Anexo # 13. Ejemplo de un árbol genealógico ordenado después de integrado el componente.



GLOSARIO DE TÉRMINOS

Genograma: es una representación gráfica que registra información sobre los miembros de una familia y sus relaciones. Su estructura en forma de árbol proporciona una rápida mirada de las relaciones familiares y es una rica fuente de hipótesis sobre cómo un problema puede estar relacionado con el contexto familiar y su evolución a través del tiempo.

CNGM: Centro Nacional de Genética Médica.

Metodología de desarrollo de software: Es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo de software.

alasARBOGEN: Sistema para la representación de árboles genealógicos.

PSTF (Pedigree Standardization Task Force): Equipo de trabajo para la estandarización de árboles genealógicos.