UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
[GRADO EN INGENIERÍA DEL SOFTWARE]

Diseño de módulos de gestión de enfermería y consejo genético en una unidad de oncología médica.

Design of nursing management modules and genetic counseling in a medical oncology unit.

Realizado por
Marco Bullones Subirats

Tutorizado por
Jose Manuel Jerez Aragonés

Co-tutorizado por
Jose Luis Subirats Contreras

Departamento
Lenguaje y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2019

Fecha defensa:

Fdo. El/la Secretario/a del Tribunal

Resumen

Galén es un sistema desarrollado en ASP.NET bajo el patrón de diseño MVC, para cubrir las necesidades asistenciales y de información de un servicio de oncología. Dispone de los componentes necesarios para realizar una adecuada explotación estadística de la información a través de una serie de algoritmos para el cálculo de curvas de supervivencia, regresiones logísticas, así como, extracción de información oculta a través de técnicas de minería de datos y modelos predictivos relativa a casi 50.000 pacientes con neoplasias de los hospitales públicos de la provincia de Málaga. Tal es la relevancia de esta aplicación en el ámbito clínico, que actualmente está siendo desarrollado en colaboración con la Junta de Andalucía para poder integrarse en su versión 2.0 con la aplicación de gestión clínica DIRAYA. El objetivo de este trabajo se centrará en el desarrollo de los módulos de enfermería y de consejo genético, integrándolos en el proyecto, así como su despliegue en producción para su uso efectivo. Para ello, se han de realizar diferentes actualizaciones de la base de datos, ya que esta ha sido sustancialmente modificada, asi como una migración de los datos existentes previamente en la base de datos de Galén en su versión inicial (1.0). Esta ha sido una tarea crítica ya que la base de datos en producción está en continuo funcionamiento por el servicios de oncología de los hospitales de la provincia de Málaga, no debiéndose cometer errores en esta, ya que pararían el servicio.

Palabras clave:

Galén, ASP.NET, MVC, Minería de datos.

Abstract

Galén is a system developed in ASP.NET under the MVC design pattern, to cover the assistance and information needs of an oncology service. It has the necessary components to carry out an adequate statistical evaluation of the information through a series of algorithms for the calculation of survival curves, logistic regressions, as well as, extraction of hidden information through data mining techniques and predictive models. relative to almost 50,000 patients with neoplasms of public hospitals in the province of Malaga. Such is the relevance of this application in the clinical field, which is currently being developed in collaboration with the Andalusian government to be able to integrate in its version 2.0 with the DIRAYA clinical management application. The objective of this work will focus on the development of the nursing and genetic counseling modules, integrating them into the project, as well as their deployment in production for their effective use. For this, different database updates have been made, since this has been modified, as well as a migration of previously affected data in the Galén database in its initial version (1.0). This has been a critical task since the database in production is in continuous operation by the oncology services of the hospitals of the province of Malaga, not having to make mistakes in this, since the service is stopped.

Keywords:

Galén, ASP.NET, MVC, Data mining technique

Índice

Resumen **1**
Abstract **1**
Índice **1**
Introducción **3**
1.1 Motivación **3**
1.2 Objetivos **4**
1.3 Estructura de la memoria **5**
Análisis **7**
2.1 Modulo de Enfermería **7**
2.1.1 Roles 7
2.1.2 Requisitos 7
2.2 Modulo de Consejo Genético **8**
2.2.1 Roles 8
2.2.2 Requisitos 8
Implementación **11**
3.1 Tecnologías usadas **11**
3.1.1 ASP.NET 11
3.1.2 MVC 12
3.1.3 Bootstrap 12
3.1.4 JavaScript 13
3.1.5 JQuery 13
3.1.6 Razor 13
3.1.7 SQL Server 14
3.1.8 SourceTree 14
3.2 Diseño de la base de datos **14**
3.2.1 Módulo de Enfermería 17
3.2.1 Módulo de Consejo Genético 18
3.3 Diseño de la aplicación web **18**

3.3.1 Modelo 18
3.3.2 Controlador 29
3.3.3 Vista 38
3.4 Problemas encontrados durante el desarrollo 59
Conclusiones 63
4.1 Futuras líneas de trabajo 64
Bibliografía 65

1

Introducción

1.1 Motivación

Este proyecto nace a raíz de la necesidad de los servicios de oncología de la provincia de Málaga de tener una aplicación que se adapte a sus necesidades, agilizando todo el proceso de atención al paciente, así como la extracción de datos para su posterior análisis utilizando técnicas de inteligencia computacional. Además, esta herramienta deberá generar en tiempo real distintas gráficas ampliamente usadas por los facultativos oncólogos, tales como curvas de supervivencia , filtros, histogramas, graficas, etc. Las cuáles serán de utilidad en el servicio de oncología y en los estudios realizados por los bioinformáticos de los hospitales públicos de Málaga.

En esta nueva versión del software ya introducido en 2005 se mejoran tanto los componentes gráficos de la interfaz, dando un enfoque centrado en el paciente, como la revisión de la base de datos actualizándola al marco jurídico de protección de datos vigente, introduciendo metadatos en cada una de las tablas permitiéndonos saber en todo momento quien hizo alguna modificación en la base de datos, así como toda la información del registro desde que se creó, hasta su eliminación, pudiendo volver rápido y fácilmente a versiones anteriores de este.

Mi motivación personal para desarrollar este proyecto es el afán de aprender nuevas tecnologías de desarrollo web, así como estar produciendo un software que está en uso actualmente.

1.2 Objetivos

Las funcionalidades asistenciales de Galén constituyen una historia clínica electrónica completa, y refleja fielmente la metodología de trabajo utilizada sobre el formato papel. Actualmente, la implantación de Galén en el servicio es prácticamente completa gracias a la simplicidad de su uso y a la similitud de su interfaz con la historia clínica en papel, así como al esfuerzo realizado por todos los facultativos para adaptarse a esta nueva manera de realizar la tarea asistencial diaria.

Actualmente se encuentra en desarrollo una nueva versión de Galén (2.0) en la cual, debido al carácter critico de la información almacenada, se ha mejorado sustancialmente la interfaz y la base de datos facilitando un control de versiones sobre cualquier modificación de la base de datos además de una mayor rapidez a la hora de buscar cualquier información. Con esta idea se implementó una base de datos temporal donde nunca se modifica o elimina la información, sino que se almacena en nuevos registros enlazados todos a través de un complejo sistema de metadatos. En esta nueva versión se han aprovechado las tecnologías de HTML5 en las cuales el navegador se adapta a la pantalla en la cual se está observando la aplicación.

El trabajo a desarrollar en este proyecto consistirá en la implementación de dos módulos utilizando ASP.NET y el patrón de diseño MVC desarrollados previamente en Galén 1.0 en los cuales se han realizado diferencias sustanciales de diseño: módulo de enfermería y consejo genético.

El módulo de enfermería será utilizado por los enfermeros del hospital para tratar todos los análisis de los pacientes que están siendo tratados de una neoplasia. Aquí se tratan diferentes variables asociadas a distintas tablas de la base de datos, tales como, acciones realizadas, enfermeros, médicos involucrados, así como información relativa al tratamiento realizado.

El módulo de consejo genético se orienta más a la investigación donde se analiza la familia de un paciente con una neoplasia asociada para predecir la probabilidad de padecer esa neoplasia en dichos familiares. Este módulo es de gran complejidad ya que está involucrada mucha información de la base de datos y se conecta con diferentes aplicaciones del hospital para mostrar los resultados de los árboles genealógicos de los pacientes involucrados obtenidos en el estudio. Ambos módulos generan informes en formato PDF y Word de la información almacenada.

El desarrollo de estos dos módulos implica la modificación de todas las tablas de la base de datos así como la migración de la base de datos original (1.0) a la nueva base de datos (2.0). Estos scripts son de capital importancia ya que se trabaja con datos reales de un sistema

ya en funcionamiento y un error en este podría parar el servicio de oncología.

Cada unidad funcional dentro de la aplicación está desarrollada en un módulo diferente facilitando así la portabilidad, legibilidad y la escalabilidad del proyecto. La intención del desarrollo de esta aplicación es que gestione todos los procedimientos y secciones que existen dentro de una unidad de oncología médica.

1.3 Estructura de la memoria

Al ser el objetivo de este trabajo el desarrollo de dos módulos totalmente independientes, todas las secciones (requisitos, actores, implementación y los scripts para la migración correspondientes) se dividirán para focalizar mejor cada uno de los módulos.

2

Análisis

En esta sección analizaremos cada uno de los dos módulos a desarrollar analizando para cada uno de ellos los roles implicados y los requisitos tanto funcionales como no funcionales.

2.1 Módulo de Enfermería

2.1.1 Roles

- Administrador
- Enfermera
- Enfermera EECC

2.1.2 Requisitos

Requisitos Funcionales

RF1. Añadir un nuevo módulo dentro del panel de gestión de un paciente asociado a las consultas de enfermería.
RF1.1. Listar todas las consultas de enfermería en una tabla visualizando los campos (Fecha, Hora, Tipo de Consulta y Facultativo), permitiendo además ordenar/filtrar a través de cualquiera de sus campos.
RF1.2. Añadir/editar/eliminar una consulta de enfermería.
RF1.3. Descargar el informe de la consulta de enfermería en formato Word/PDF.
RF1.4. Añadir al módulo de control de versiones cualquier cambio realizado sobre una consulta de enfermería (edición o borrado).
RF1.5. Añadir las valoraciones de enfermería en el informe de actividad asistencial (PDF) que se genera de forma automática y será siniestrado a la aplicación DIRAYA en un futuro.

Requisitos No Funcionales

RNF1. El sistema sólo permitirá la visualización de este módulo a los usuarios con los roles previamente mencionados.

RNF2. El módulo debe de ser implementado siguiendo el mismo esquema de desarrollo del resto de módulos ya implementados, puesto que esas cuestiones fueron discutidas previamente a la realización de este trabajo.

RNF3. El sistema deberá implementarse mediante el patrón Modelo-Vista-Controlador(MVC).

RNF4. Se deben generar la debidas pruebas unitarias para comprobar el buen funcionamiento del módulo ya que va ser implementado en un proyecto real que está siendo usado en todos los hospitales públicos de la provincia de Málaga.

2.2 Modulo de Consejo Genético

2.2.1 Roles

- Administrador (visualización)
- Medico (visualización)
- Editor de consejo Genético (visualización y edición)

2.2.2 Requisitos

Requisitos Funcionales

RF1. Añadir una nueva vista accesible desde cualquier punto de la aplicación para visualizar/gestionar las familias pertenecientes al estudio de consejo genético.

RF1.1. Listar todas las familias de consejo genético en una tabla, "Gestión de familias", visualizando los campos (N.º familia, año inclusión, criterio, iniciales y dos accesos rápidos a descargar el árbol genealógico de la familia en formato GNO y/o PDF), permitiendo además ordenar/filtrar a través de cualquiera de sus campos.

RF1.1.1. Crear/editar una familia de consejo genético.

RF1.2. Listar todos los pacientes de consejo genético pertenecientes al programa de consejo genético en una tabla, "Individuos de la Familia", visualizando los campos (N.º historia, nombre, apellidos, teléfono, probando/familiar, NUHSA y un acceso rápido al panel del paciente), permitiendo además ordenar/filtrar a través de cualquiera de sus campos.

RF1.3. Dentro de la tabla "Gestión de familias", al hacer *click* sobre la fila de una determinada familia se filtraran en la tabla "Individuos de la familia" todos aquellos pacientes que pertenezcan a dicha familia. Además de visualizar los datos de dicha familia.

RF1.4. Dentro de la tabla "Gestión de individuos", al hacer *click* sobre la fila de un determinado paciente se mostraran los datos relativos a su familia, árboles genealógicos en formato GNO/PDF y se mostraran los integrantes de su familia de consejo genético.
RF2. Añadir modulo dentro del panel de gestión de un paciente para mostrar toda la información relativa a consejo genético de ese paciente.
RF2.1. Incorporar a el paciente a una familia de consejo genético.
RF2.2. Añadir/editar datos relativos al estudio de consejo genético.
RF2.3. Añadir/editar la primera visita.
RF2.4. Añadir/editar/eliminar consultas sucesivas.
RF2.5. Añadir/editar/eliminar anotaciones.
RF3.1 Generación de informe en formato PDF de: primera visita, consultas sucesivas y anotaciones.

Requisitos No Funcionales

RNF1. El sistema sólo permitirá la visualización de este módulo a los usuarios con los roles previamente mencionados.
RNF2. El módulo debe de ser implementado siguiendo el mismo esquema de desarrollo del resto de módulos ya implementados, puesto que esas cuestiones fueron discutidas previamente a la realización de este trabajo.
RNF3. El sistema deberá implementarse mediante el patrón Modelo-Vista-Controlador(MVC).
RNF4. Las dos tablas se diseñaran utilizando el esquema Maestro-Detalle, posicionando la tabla "Gestión de familias" en la mitad izquierda de la pantalla, y la tabla "Individuos de la familia" en la mitad derecha.
RNF5. Se utilizara el color (#6c7e9b) para marcar una fila seleccionada en cualquiera de los dos tablas.

3

Implementación

En esta sección abordaremos las distintas tecnologías utilizadas, el diseño de la base de datos, diseño de la aplicación web y los scripts de migración.

3.1 Tecnologías usadas

El entorno de desarrollo utilizado es *Visual Studio 2017* utilizando el framework de desarrollo *ASP.NET MVC*. Como sistema gestor de base de datos utilizaremos *SQL Server 2012*, ayudándonos de la herramienta *SQL Server Management Studio* (SSMS) para facilitar el manejo de esta. Al trabajar todo un equipo en más líneas de trabajo dentro de este proyecto, como control de versiones para el proyecto de visual studio y para los scripts relativos a la base de datos se ha utilizado la herramienta *SourceTree*. Para el desarrollo de las vistas utilizaremos *HTML*, *CSS* y *Bootstrap* para un diseño "responsive". Se utilizara tanto *JavaScript*, como *JQuery* para el manejo de eventos. Y la herramienta *Razor* para la programación de las vistas.

3.1.1 ASP.NET

ASP.NET es un entorno para aplicaciones web desarrollado y comercializado por Microsoft, es usado por programadores y diseñadores para construir sitios web dinámicos, aplicaciones web y servicios web XML. Está construido sobre el Common Language Runtime, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework.

Actualmente, ASP.NET soporta tres modelos de programación: ASP.NET Web Forms, ASP.NET MVC y ASP.NET Web Pages. Aunque los tres modelos de programación se ejecutan

sobre la misma base de ASP.NET, cada uno de ellos estructura la aplicación de maneras completamente distintas.

Para este trabajo se utilizara el modelo ASP.NET MVC el cual es un framework de aplicaciones web que implementa el patrón de diseño modelo-vista-controlador (MVC).

3.1.2 MVC

MVC es un patrón de arquitectura que ayuda a crear una separación lógica entre el modelo (información y lógica de negocio), la vista (la lógica de presentación) y el controlador (intermediario entre la vista y el modelo).

- Modelo: Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones.
- Controlador: Gestiona las peticiones de los usuarios. Es responsable de responder la información solicitada con la ayuda tanto del modelo como de la vista.
- Vista: Es responsable del uso de la información de la cual dispone para producir cualquier interfaz de presentación de cualquier petición que se presente.

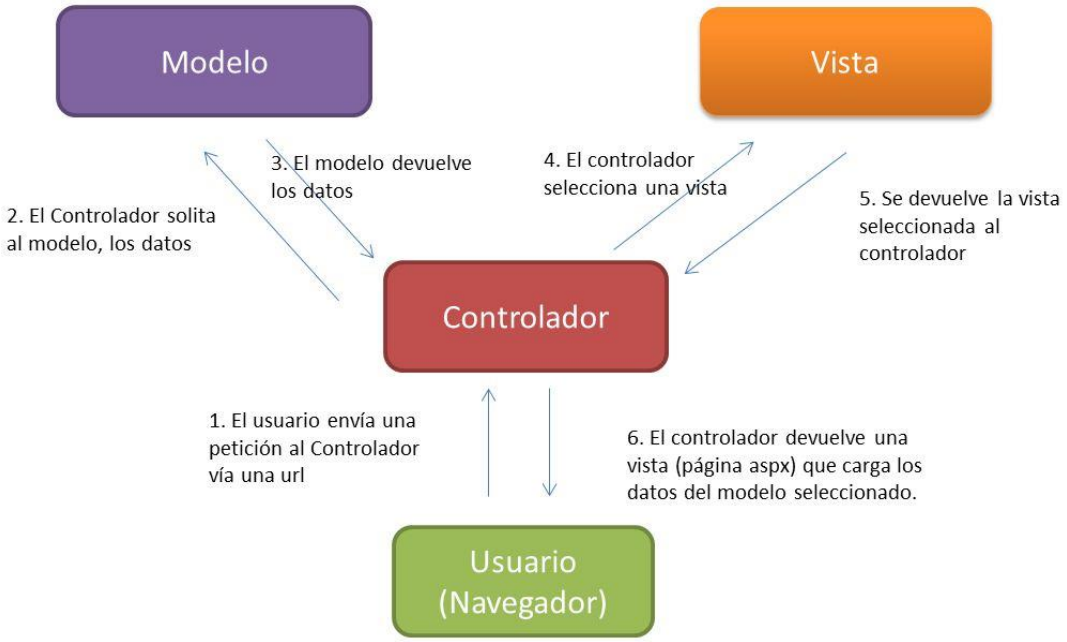

Figura 1: Esquema del patrón MVC.

3.1.3 Bootstrap

Boostrap es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web que facilita la maquetación de sitios web ofreciéndonos herramientas para que nuestro sitio web se vea bien en toda clase de dispositivos, ahorrándonos así el trabajo de tener que rediseñar un sitio web.

Además contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. Solo se ocupa del desarrollo front-end.

3.1.4 JavaScript

JavaScript nace como un lenguaje sencillo destinado a añadir algunas características interactivas con el usuario a las páginas web. No requiere de compilación ya que el lenguaje funciona en el lado del cliente y los navegadores son los encargados de interpretar estos códigos. Algunas de sus principales características son:

- Multiplataforma, ya que se puede utilizar en Windows, Linux o Mac o en el navegador de tu preferencia.
- Es imperativo y estructurado, mediante un conjunto de instrucciones indica al computador qué tarea debe realizar.
- Prototipado, debido a que usa prototipos en vez de clases para el uso de herencia.
- Orientado a objetos y eventos.
- Es interpretado, no se compila para poder ejecutarse.

3.1.5 JQuery

JQuery es una librería de JavaScript de código abierto que simplifica la tarea de programar en JavaScript y permite agregar interactividad a un sitio web sin tener grandes conocimientos del lenguaje.

Entre sus grandes ventajas se encuentran:

- Facilita en gran manera el uso de Ajax. Éste funciona de forma asíncrona del resto del código, lo cual significa que el código escrito con Ajax puede comunicarse con el servidor y actualizar su contenido sin necesidad de volver a cargar la página.
- Te permite insertar y/o eliminar elementos DOM (Modelo de Objetos del Documento) en una página HTML, así como agrupar líneas más fácilmente.
- Acceder y desplazarse por los documentos HTML, así como la realización de efectos y el manejo de eventos, también son mejorados con jQuery.

3.1.6 Razor

En una sintaxis basada en C# (aunque se puede programar en Visual Basic) que funciona como un motor de vistas donde todas las variables que mostremos con '@' son parseadas con HTML Encode. Lo que hace es remplazar símbolos <, > o & por sus correspondientes códigos proporcionando una sintaxis optimizada para la generación de HTML utilizando un enfoque

de plantillas centrado en el código y con una transición mínima entre HTML y código.

De esta manera evitamos que nos introduzcan código mal intencionado que pueda alterar el comportamiento de nuestro programa.

Sus principales características son:

- Compacto, expresivo y fluido: buscan reducir la cantidad de código que necesitamos para crear las vistas, evitando que tengamos que denotar de una forma especial cada línea de código procedural.
- Fácil de aprender.
- Testeable: podremos crear tests unitarios de las vistas.

3.1.7 SQL Server

SQL Server es un sistema de gestión de bases de datos relacionales (RDBMS) de Microsoft que está diseñado para el entorno empresarial.

SQL (Structured Query Language) es un lenguaje estándar e interactivo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Nos permiten almacenar y gestionar gran cantidad de datos los cuales se almacenan en diferentes tablas y las relaciones se establecen usando claves primarias u otras llaves conocidas como claves externas o foráneas.

3.1.8 SourceTree

SourceTree es quizás uno de los mejores clientes GUI para manejar repositorios git y mercurial que existen en la actualidad. Entre sus principales características se encuentran:

- Crear y clonar repositorios de cualquier sitio, tanto Git como Mercurial. Además de integrarse perfectamente con Bitbucket o Github.
- Commit, push, pull y merge de nuestros archivos
- Detectar y resolver conflictos
- Consultar el historial de cambios de nuestros repositorios.

3.2 Diseño de la base de datos

Todas las tablas que posean información personal debido a la criticidad de la información y siguiendo el Reglamento Europeo de Protección de Datos (RPGD) y la Ley Orgánica de Protección de datos (LOPD), nunca se realiza ningún cambio (edición/borrado) directamente sobre un registro, sino que se hará una copia del registro, conservando su estado antes de la modificación y después modificando el registro original donde están enganchadas todas las claves foráneas.

El esquema utilizado para el control de versiones ya se discutió previamente a la realización de este trabajo, por lo que solo se expondrán el significado de cada uno de los

campos implicados. Debido a la longitud de los identificadores, para ilustrar el ejemplo se utilizaran enteros como identificadores.

- IdCV: Este valor se mantiene en todos las versiones del registro desde su creación en la operación de INSERT, coincidiendo con la clave primaria, hasta su marcado como borrado en la operación de DELETE. Este valor, por lo tanto, nos agrupa todos los registros para poder analizar ciclo de vida.
- ActualCV: El valor 1 representara el registro actual, mientras que el valor 0 indica que es un registro pasado.
- BorradaCV: El valor 1 representara que ese registro está marcado como borrado, 0 en caso contrario.
- IdUsuarioCV: El identificador del usuario que genero ese registro.
- FCreacionCV: La fecha cuando se generó ese registro.

Supongamos a un primer usuario con el id (10), y otro usuario que va a modificar el valor identificado como "dato" y tiene como identificador el valor 20. Este sería el cambio.

ID	DATO	IDCV	ACTUALCV	BORRADACV	IDUSUARIOCV	FCREACION
1	A	1	1	0	10	1/1/2019

Tabla 1: Estado del registro antes de la modificación.

ID	DATO	IDCV	ACTUALCV	BORRADACV	IDUSUARIOCV	FCREACION
1	B	1	1	0	20	2/1/2019
2	A	1	0	0	10	1/1/2019

Tabla 2: Estado del registro después de la modificación.

Estas tablas al tener gran cantidad de registros puesto que almacenan todas las modificaciones realizadas sobre un registro, se han creado vistas[1] para su manipulación y mostrar la lista de datos, puesto que solo nos interesaran los registros marcados como actuales y que no estén borrados.

A la hora de insertar, modificar o borrar datos sobre una de las tablas que siguen el esquema de control de versiones anteriormente mencionado (ej. insertar una familia de consejo genético) se programaran una serie de triggers[2]

[1] Una vista actúa como filtro de las tablas subyacentes a las que se hace referencia en ella. La consulta que define la vista puede provenir de una o de varias tablas, o bien de otras vistas de la base de datos actual u otras bases de datos.

[2] Un desencadenador (o Trigger) es una clase especial de procedimiento almacenado que se ejecuta automáticamente cuando se produce un evento en el servidor de bases de datos. Los desencadenadores DML se ejecutan cuando un usuario intenta modificar datos mediante un evento de lenguaje de manipulación de datos (DML). Los eventos DML son instrucciones INSERT, UPDATE o DELETE de una tabla o vista.

para ejecutar este comportamiento, detallados en la ilustración 1, 2 y 3.

Debido a la cantidad de vistas a abordar se ilustrara el ejemplo de la vista de "FamiliaCG" puesto que el resto se realizó de forma análoga.

```
USE [Galen_V2_0]
GO
/****** Object:  Trigger [dbo].[v_FamiliaCG_on_insert]    Script Date: 02/07/2019 13:23:35 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER TRIGGER [dbo].[v_FamiliaCG_on_insert] on [dbo].[v_FamiliaCG]
INSTEAD OF INSERT
AS
BEGIN
SET NOCOUNT ON

DECLARE @nuevoid uniqueidentifier
SELECT @nuevoid = NEWID()

DECLARE @fecha datetime
SELECT @fecha = GETDATE()

declare @nuevoCod int
select @nuevoCod = (isnull(max(cod),0)+1) from FamiliaCG where year(FechaInclusion) = year(@fecha)

INSERT INTO FamiliaCG(IdFamiliaCG, FechaInclusion, IdCriterioSeleccionRiesgo, IdArbolGenealogicoGNO, IdArbolGenealogicoPDF, IdArbolGenealogicoGNOMaterno, IdArbolGenealogicoPDFMaterno,
cod, idCV, actualCV, borradaCV, idUsuarioCV, fCreacionCV)
 SELECT isnull(IdFamiliaCG, @nuevoid), FechaInclusion, IdCriterioSeleccionRiesgo, IdArbolGenealogicoGNO, IdArbolGenealogicoPDF, IdArbolGenealogicoGNOMaterno, IdArbolGenealogicoPDFMaterno,
  @nuevoCod, isnull(IdFamiliaCG, @nuevoid) AS idCV, 1 AS actualCV, 0 AS borradaCV, idUsuarioCV, @fecha AS fCreacionCV  FROM inserted

INSERT INTO historico(idHistorico, idUsuario, nombreTabla, idFila, operacion, fecha)
 SELECT @nuevoid, idUsuarioCV, 'FamiliaCG' AS nombreTabla, @nuevoid AS idFila, 'insert' AS operacion, @fecha AS fecha  FROM inserted
END
```

Ilustración 1: Trigger "on insert" de la vista v FamiliaCG

```
USE [Galen_V2_0]
GO
/****** Object:  Trigger [dbo].[v_FamiliaCG_on_update]    Script Date: 02/07/2019 13:36:34 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


ALTER TRIGGER [dbo].[v_FamiliaCG_on_update] on [dbo].[v_FamiliaCG]
INSTEAD OF UPDATE
AS
BEGIN
SET NOCOUNT ON

DECLARE @nuevoid uniqueidentifier
SELECT @nuevoid = NEWID()

DECLARE @fecha datetime
SELECT @fecha = GETDATE()

INSERT INTO FamiliaCG(IdFamiliaCG, FechaInclusion, IdCriterioSeleccionRiesgo, IdArbolGenealogicoGNO, IdArbolGenealogicoPDF, IdArbolGenealogicoGNOMaterno, IdArbolGenealogicoPDFMaterno,
cod, idCV, actualCV, borradaCV, idUsuarioCV, fCreacionCV)
 SELECT @nuevoid, FechaInclusion, IdCriterioSeleccionRiesgo, IdArbolGenealogicoGNO, IdArbolGenealogicoPDF, IdArbolGenealogicoGNOMaterno, IdArbolGenealogicoPDFMaterno, cod,
 idCV, 0 AS actualCV, borradaCV, idUsuarioCV, fCreacionCV FROM FamiliaCG
 WHERE IdFamiliaCG in (SELECT ins.IdFamiliaCG FROM inserted ins)

UPDATE FamiliaCG SET
IdFamiliaCG = u.IdFamiliaCG, FechaInclusion = u.FechaInclusion, IdCriterioSeleccionRiesgo = u.IdCriterioSeleccionRiesgo, IdArbolGenealogicoGNO = u.IdArbolGenealogicoGNO,
IdArbolGenealogicoPDF = u.IdArbolGenealogicoPDF, IdArbolGenealogicoGNOMaterno = u.IdArbolGenealogicoGNOMaterno, IdArbolGenealogicoPDFMaterno = u.IdArbolGenealogicoPDFMaterno,
idUsuarioCV = u.idUsuarioCV, borradaCV = 0, fCreacionCV = @fecha
FROM FamiliaCG t, inserted u WHERE t.IdFamiliaCG = u.IdFamiliaCG

INSERT INTO historico(idHistorico, idUsuario, nombreTabla, idFila, operacion, fecha)
 SELECT @nuevoid, idUsuarioCV, 'FamiliaCG' AS nombreTabla, idCV AS idFila, 'update' AS operacion, @fecha AS fecha  FROM FamiliaCG
 WHERE IdFamiliaCG in (SELECT ins.IdFamiliaCG FROM inserted ins)


END
```

Ilustración 2: Trigger "on update" de la vista v FamiliaCG

```
USE [Galen_V2_0]
GO
/****** Object:  Trigger [dbo].[v_FamiliaCG_on_delete]    Script Date: 02/07/2019 13:38:12 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


ALTER TRIGGER [dbo].[v_FamiliaCG_on_delete] on [dbo].[v_FamiliaCG]
INSTEAD OF DELETE
AS
BEGIN
SET NOCOUNT ON


END
```

Ilustración 3: Trigger "on delete" de la vista v FamiliaCG

La creación de las tablas en la base de datos en base a los requerimientos de información solicitados por el hospital generaron los diagramas relaciones expuestos en las figuras 2 y 3.

Todos los identificadores son de tipo uniqueidentifier (GUID), una cadena hexadecimal con el formato (xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx).

3.2.1 Módulo de Enfermería

La tabla llamada "enfermeria Valoracion" representara una visita de un paciente a la unidad de enfermería, la cual recoge toda la información solicitada por el hospital y relaciona la visita, paciente, enfermera y centro de salud donde se le atendió. Además de todas las tablas ubicadas a la derecha de estas que representan todas las posibilidades para un determinado atributo, evitando así repetir información.

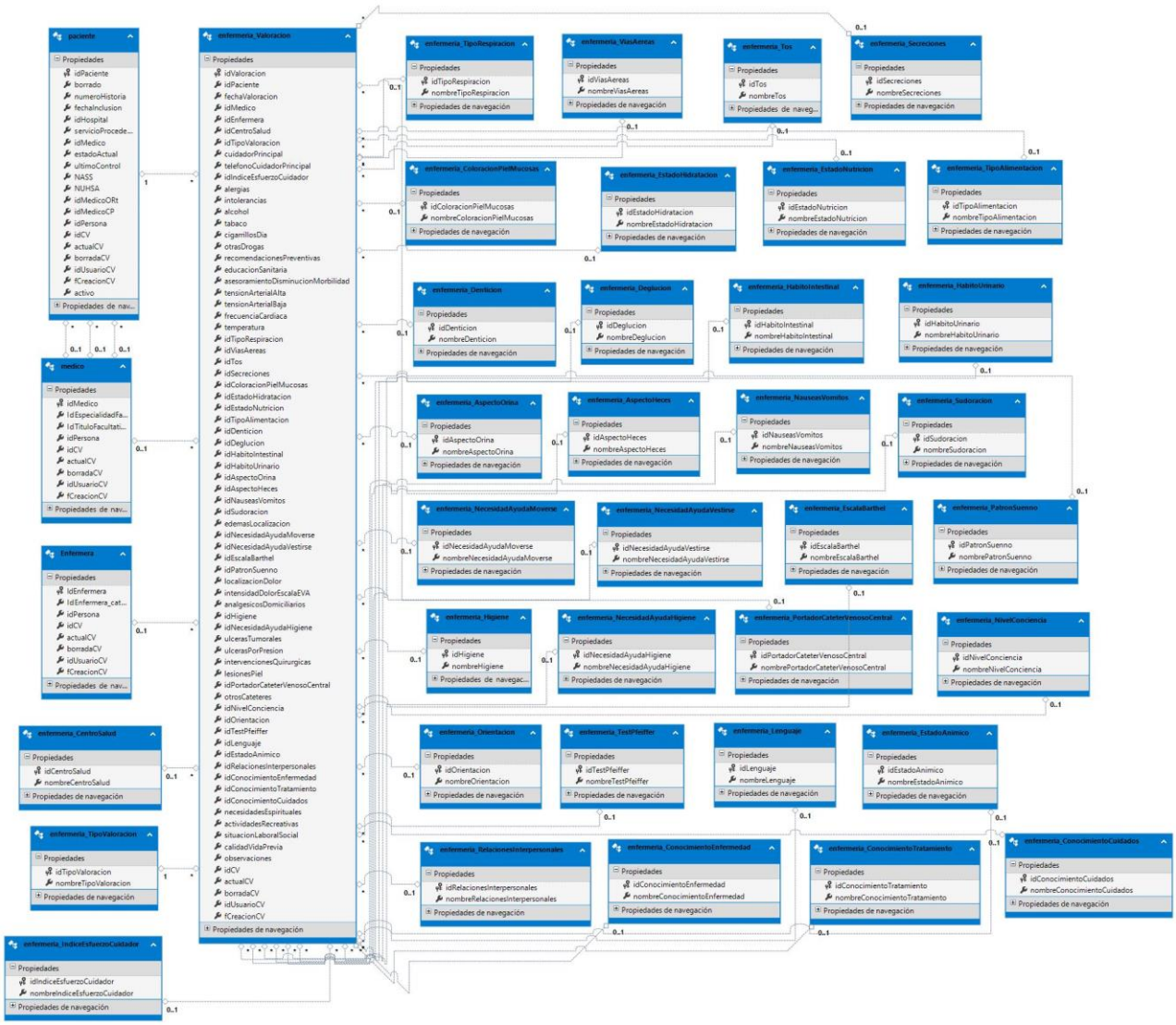

Figura 2: Vista general de las tablas del módulo de enfermería

3.2.1 Módulo de Consejo Genético

En este módulo encontraremos dos tablas principales:

- FamiliaCG: representa una familia de estudio del programa de consejo genético.
- PacienteCG: representa la relación de un paciente con la familia de consejo genético a la que está asociada, se relaciona mediante esta tabla puesto que en la fase de requisitos se solicitó que un mismo paciente pudiese formar parte de varias familias de consejo genético.

El resto de tablas recoge información relacionada con el estudio de consejo genético que se le realiza a dicho paciente.

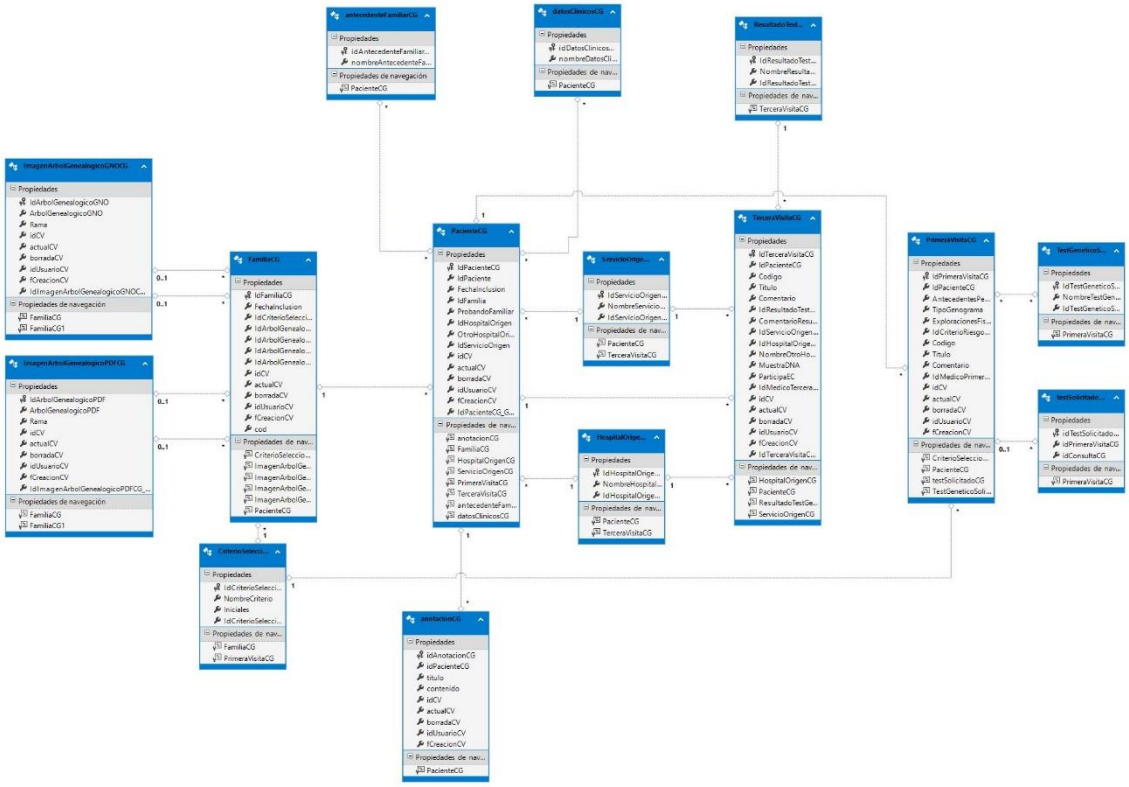

Figura 3: Vista general de las tablas del módulo de consejo genético

3.3 Diseño de la aplicación web

Debido a la cantidad de información relativa tanto al módulo de enfermería, como al de consejo genético se expondrán los principales ejemplos, además de algunos problemas encontrados durante su desarrollo.

Se dividirá en primera instancia en modelo, vista y controlador siguiendo el esquema MVC, y posteriormente en los dos módulos a desarrollar.

3.3.1 Modelo

Debido a que como mencionamos en el apartado 3.2, trabajamos sobre vistas en lugar de sobre las propias tablas,

resulto que no podíamos trabajar directamente sobre el entity framework[3] generado automáticamente, puesto que este no nos permitía trabajar sobre las vistas. Por tanto se tuvieron que crear modelos parciales que será con lo que se trabajara a partir de ahora. Para el desarrollo de ambos modelos se ha seguido el siguiente formato:

- Declaración de los atributos asociados a la tabla en la base de datos (campos).
- Creación de la función Rellenar() que devolverá una instancia del modelo habiendo rellenado los campos a partir del modelo de entity framework.
- Creación de la función Nueva() que devuelva una instancia del modelo habiendo inicializado los identificadores necesarios, así como la fecha.
- Creación de la función InsertarEn() que intentara insertar en las vistas con los datos recogidos lanzando el trigger "on insert".
- Creación de la función GuardarEn() que intentara guardar algún cambio dentro de un registro a través de la vista, lanzando así al trigger "on update".

3.3.1.1 Módulo de Enfermería (FormEnfermeria)

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.ComponentModel.DataAnnotations;
5   using System.Diagnostics;
6   using System.Linq;
7   using System.Threading.Tasks;
8   using System.Web;
9   using System.Web.Mvc;
10
11  namespace Galen.Models
12  {
13      public class FormEnfermeria
14      {
15          campos
289
290         public static FormEnfermeria Rellenar(ModeloPaciente mPaciente)...
303
304         public static FormEnfermeria Rellenar(enfermeria_Valoracion enfermeria_Valoracion)...
385
386         public static FormEnfermeria Nueva(paciente pac)...
405
406         public void GuardarEn(Galen_V2_0Entities ctx, enfermeria_Valoracion valoracion, string userName)...
595
596         public void InsertarEn(Galen_V2_0Entities ctx, string userName)...
797     }
798 }
```

Ilustración 4: Estructura general del modelo FormEnfermeria

A continuación se analizaran una a una cada una de las secciones que aparecen en la ilustración 4.

[3] Entity Framework permite a los programadores trabajar con datos en forma de objetos y propiedades específicos del dominio, por ejemplo, con clientes y direcciones de clientes, sin tener que pensar en las tablas de las bases de datos subyacentes y en las columnas en las que se almacenan estos datos.

- Dentro de la región llamada como "campos" estarán todos los atributos relevantes para nuestro modelo, donde detallaremos algunos ejemplos.

```
public Guid idValoracion { get; set; }
public Guid idPaciente { get; set; }

[DisplayName("Fecha")]
[DataType(DataType.Date)]
[DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
[Required()]
public DateTime fechaEnfermeria { get; set; }

[DisplayName("Hora")]
[Range(0, 23)]
[Required()]
public int horaEnfermeria { get; set; }

[DisplayName("Minutos")]
[Range(0, 59)]
[Required()]
public int minutosEnfermeria { get; set; }

[DisplayName("Médico responsable:")]
public string medicoResponsable { get; set; }

[DisplayName("D.E que realiza la valoración:")]
public Guid? idEnfermeraRealizaValoracion { get; set; }

[DisplayName("Centro de Salud:")]
public Guid? idCentroSalud { get; set; }

[DisplayName("Tipo de valoración:")]
[Required()]
public Guid? idTipoValoracion { get; set; }

[DisplayName("Cuidador principal:")]
public string cuidadorPrincipal { get; set; }

[DisplayName("Teléfono del cuidador principal:")]
public string telefonoCuidadorPrincipal { get; set; }
```

Ilustración 5: Algunos ejemplos de propiedades de FormEnfermeria

A la hora de representar los atributos existen una serie de propiedades que se detallaran a continuación.

- Cadena que utilizaremos en la vista para mostrar el nombre de la propiedad a cumplimentar.

```
[DisplayName("Médico responsable:")]
```

- Indicar que un campo sea requerido en el formulario indicando.

```
[Required()]
```

- Permite valores nulos en un atributo mediante el carácter '?' después del tipo de dato.

```
[DisplayName("Centro de Salud:")]
public Guid? idCentroSalud { get; set; }
```

- Indicar ciertas restricciones, por ejemplo que la fecha siga el formato 'dd-MM-yyyy'.

```
[DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}")]
```

- O que tengan cierto rango de valores posibles.

```
[Range(0, 999)]
```

- En nuestra función Rellenar, asignaremos a nuestros atributos los del entity framework.

```
public static FormEnfermeria Rellenar(enfermeria_Valoracion enfermeria_Valoracion)
      {
          Galen_V2_0Entities db = new Galen_V2_0Entities();
          return new FormEnfermeria
          {
              actividadesRecreativas = enfermeria_Valoracion.actividadesRecreativas, alcohol = enfermeria_Valoracion.alcohol,
              alergias = enfermeria_Valoracion.alergias, analgesicosDomiciliarios = enfermeria_Valoracion.analgesicosDomiciliarios,
              ...,
              ulcerasTumorales = enfermeria_Valoracion.ulcerasTumorales
          };
      }
```

Ilustración 6: Función Rellenar del modelo FormEnfermeria

- En la función Nueva, se creara una instancia de nuestro modelo con los atributos iniciales necesarios.

```
public static FormEnfermeria Nueva(paciente pac)
      {
          var ahora = DateTime.Now;


          return new FormEnfermeria
          {
              idValoracion = Guid.NewGuid(),
              idPaciente = pac.idPaciente,
              fechaEnfermeria = ahora.Date,
              horaEnfermeria = ahora.Hour,
              minutosEnfermeria = ahora.Minute,
              medicoResponsable = paciente.cadenaMedicoResponsable(pac)
          };
      }
```

Ilustración 7: Función Nueva del modelo FormEnfermeria

- En la función GuardarEn, se intentara guardar una modificación de un registro en la base de datos.

```
public void GuardarEn(Galen_V2_0Entities ctx, enfermeria_Valoracion valoracion, string userName)
        {
            var fecha = this.fechaEnfermeria.Date
                .AddHours(this.horaEnfermeria)
                .AddMinutes(this.minutosEnfermeria);

            var user = ctx.AspNetUsers
                .Where(x => x.UserName == userName)
                .Single();

            using (var tr = ctx.Database.BeginTransaction())
            {
                try
                {
                    Debug.Assert(this.idValoracion == valoracion.idValoracion);
                    Debug.Assert(this.idPaciente == valoracion.idPaciente);

                    int ret = ctx.Database.ExecuteSqlCommand(
                        @"UPDATE [v_enfermeria_Valoracion] SET
                            [idPaciente] = @p1, [fechaValoracion] = @p2, [idMedico] = @p3, [idEnfermera] = @p4, [idCentroSalud] = @p5, [idTipoValoracion] = @p6
                           ...
                           ,[calidadVidaPrevia] = @p67, [observaciones] = @p68, [idUsuarioCV] = @p69
                        WHERE [idValoracion] = @p0
                        ",
                        this.idValoracion, this.idPaciente, fecha, this.medicoResponsable, this.idEnfermeraRealizaValoracion, this.idCentroSalud, this.idTipoValoracion,
                        ...
                        this.calidadVidaPrevia, this.observaciones, user.Id
                    );

                    tr.Commit();
                }
                catch (Exception)
                {
                    tr.Rollback();
                    throw;
                }
            }
        }
```

Ilustración 8: Función GuardarEn del modelo FormEnfermeria

- En la función InsertarEn, se intentara insertar un nuevo registro en la base de datos si su GUID no existía previamente, sino se llamara a la función GuardarEn.

```
public void InsertarEn(Galen_V2_0Entities ctx, string userName)
        {
            enfermeria_Valoracion enfVal = enfermeria_Valoracion.Buscar(ctx, idValoracion);
            if (enfVal != null) GuardarEn(ctx, enfVal, userName);
            else
            {
                var fecha = this.fechaEnfermeria.Date
                .AddHours(this.horaEnfermeria)
                .AddMinutes(this.minutosEnfermeria);

                var user = ctx.AspNetUsers
                    .Where(x => x.UserName == userName)
                    .Single();

                using (var tr = ctx.Database.BeginTransaction())
                {
                    try
                    {

                        int ret = ctx.Database.ExecuteSqlCommand(
                            @"INSERT INTO v_enfermeria_Valoracion(
                           [idValoracion]
                          ,[idPaciente]
                          ,[fechaValoracion]
                          ,[idMedico]
                          ...
                          ,[idUsuarioCV]
                    ) VALUES(
                           @p0,@p1,@p2,@p3,@p4,@p5,@p6,@p7,@p8,@p9
                          ...
                    )",
                            this.idValoracion,
                            this.idPaciente,
                            fecha,
                            this.medicoResponsable,
                            ...
                            user.Id
                        );

                        tr.Commit();
                    }
                    catch (Exception e)
                    {

                        tr.Rollback();
                        throw;
                    }
                }
            }
        }
```

Ilustración 9: Función InsertarEn del modelo FormEnfermeria

3.3.1.2 Módulo de Consejo Genético (FormFamiliaCG)

Al seguir el mismo esquema se detallan las principales diferencias entre ambos.

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.ComponentModel.DataAnnotations;
5   using System.Diagnostics;
6   using System.Linq;
7   using System.Threading.Tasks;
8   using System.Web;
9   using System.Web.Mvc;
10
11  namespace Galen.Models
12  {
13      public class FormFamiliaCG
14      {
15          campos
44
45          public static string generaCodigoGalen_1(FamiliaCG familia)...
49
50          public static FormFamiliaCG Nueva()...
61
62          public static FormFamiliaCG Rellenar(FamiliaCG familiaCG)...
75
76          public void InsertarEn(Galen_V2_0Entities ctx, string userName)...
160
161         public void GuardarEn(Galen_V2_0Entities ctx, FamiliaCG familiaCG, string userName)...
281
282     }
283 }
```

Ilustración 10: Estructura general del modelo FormFamiliaCG

Uno de los cambios que se ha realizado con respecto a la primera versión de esta aplicación (Galen 1) fue modificar el identificador de la familia que era un entero auto-incremental (cod), por un valor GUID. Pero los oncólogos usaban ese identificador como parte de un código que se utiliza en los informes (Fecha de inclusión - Iniciales del criterio de selección de riesgo - entero auto-incremental identificativo, ej. 2010-LF-000001).

- Generamos el código usado por los oncólogos como una función estática.

```
public static string generaCodigoGalen_1(FamiliaCG familia)
        {
            return familia.FechaInclusion.Year.ToString() + "-" + familia.CriterioSeleccionRiesgoCG.Iniciales + "-" + familia.cod;
        }
```

- Se creará una instancia de nuestro modelo con los atributos iniciales necesarios.

```
public static FormFamiliaCG Nueva()
        {
            var ahora = DateTime.Now;


            return new FormFamiliaCG
            {
                idFamiliaCG = Guid.NewGuid(),
                fechaInclusion = ahora
            };
        }
```

- Rellenaremos nuestro modelo con los atributos del entity framework.

```
public static FormFamiliaCG Rellenar(FamiliaCG familiaCG)
        {
            return new FormFamiliaCG
            {
                idFamiliaCG = familiaCG.IdFamiliaCG,
                fechaInclusion = familiaCG.FechaInclusion,
                idCriterioSeleccionRiesgo = familiaCG.IdCriterioSeleccionRiesgo,
                cod = familiaCG.cod,
                codigo = generaCodigoGalen_1(familiaCG),
                idArbolGenealogicoGNO = familiaCG.IdArbolGenealogicoGNO,
                idArbolGenealogicoPDF = familiaCG.IdArbolGenealogicoPDF
            };
        }
```

- A la hora de insertar una nueva familia hay que tener en cuenta que atacaremos a tres tablas de la base de datos, puesto que las imágenes de los árboles genéticos (GNO/PDF) deben conservar todas las diferentes subidas a una misma familia se hicieron en tablas separadas. Se seguirá el mismo esquema mostrado en la ilustración 9, pero se harán 3 inserciones separadas.

```
public void InsertarEn(Galen_V2_0Entities ctx, string userName)
        {
            FamiliaCG familiaCG = FamiliaCG.Buscar(ctx, idFamiliaCG);

            if (familiaCG != null) GuardarEn(ctx, familiaCG, userName);
            else
            {

                var user = ctx.AspNetUsers
                    .Where(x => x.UserName == userName)
                    .Single();

                using (var tr = ctx.Database.BeginTransaction())
                {
                    try
                    {
                        if(idArbolGenealogicoGNO != null)
                        {
                            int ret1 = ctx.Database.ExecuteSqlCommand(
                            @"INSERT INTO v_ImagenArbolGenealogicoGNOCG(
                            [IdArbolGenealogicoGNO]
                            ,[ArbolGenealogicoGNO]
                            ,[Rama]
                            ,[idUsuarioCV]
                            ) VALUES(
                                  @p0,@p1,@p2,@p3
                            )",
                                   this.idArbolGenealogicoGNO,
                                   this.arbolGenealogicoGNO,
                                   "Paterna",
                                   user.Id
                            );
                        }
```

```
        if(idArbolGenealogicoPDF != null)
        {
            int ret1 = ctx.Database.ExecuteSqlCommand(
            @"INSERT INTO v_ImagenArbolGenealogicoPDFCG(
           [IdArbolGenealogicoPDF]
          ,[ArbolGenealogicoPDF]
          ,[Rama]
          ,[idUsuarioCV]
    ) VALUES(
           @p0,@p1,@p2,@p3
    )",
            this.idArbolGenealogicoPDF,
            this.arbolGenealogicoPDF,
            "Paterna",
            user.Id
        );
```

```
                int ret = ctx.Database.ExecuteSqlCommand(
                    @"INSERT INTO v_FamiliaCG(
                  [IdFamiliaCG]
                  ,[FechaInclusion]
                  ,[IdCriterioSeleccionRiesgo]
                  ,[IdArbolGenealogicoGNO]
                  ,[IdArbolGenealogicoPDF]
                  ,[idUsuarioCV]
            ) VALUES(
                   @p0,@p1,@p2,@p3,@p4,@p5
            )",
                    this.idFamiliaCG,
                    this.fechaInclusion,
                    this.idCriterioSeleccionRiesgo,
                    this.idArbolGenealogicoGNO,
                    this.idArbolGenealogicoPDF,
                    user.Id
                );

                tr.Commit();
            }

            catch (Exception e)
            {

                tr.Rollback();
                throw;
            }
        }


    }
```

- A la hora de guardar hay que tener en cuenta si al actualizar dicha familia tenía previamente ya un árbol genealógico asociado y lo ha modificado, porque esto definirá si se tendrá que realizar un insert/update sobre dichas tablas.

```
public void GuardarEn(Galen_V2_0Entities ctx, FamiliaCG familiaCG, string userName)
        {
            var user = ctx.AspNetUsers
                .Where(x => x.UserName == userName)
                .Single();

            this.idArbolGenealogicoGNO = familiaCG.IdArbolGenealogicoGNO;
            this.idArbolGenealogicoPDF = familiaCG.IdArbolGenealogicoPDF;

            using (var tr = ctx.Database.BeginTransaction())
            {
                try
                {
                    Debug.Assert(this.idFamiliaCG == familiaCG.IdFamiliaCG);
```

Dependiendo de si tenía previamente generado ya un árbol genealógico dicha familia se realizara un insert/update para lanzar el trigger correspondiente.

```
if (this.arbolGenealogicoGNO != null && this.arbolGenealogicoGNO.Length != 0)
{
    if(this.idArbolGenealogicoGNO == null)
    {
        this.idArbolGenealogicoGNO = Guid.NewGuid();
        int ret1 = ctx.Database.ExecuteSqlCommand(
            @"INSERT INTO v_ImagenArbolGenealogicoGNOCG(
           [IdArbolGenealogicoGNO]
          ,[ArbolGenealogicoGNO]
          ,[Rama]
          ,[idUsuarioCV]
        ) VALUES(
                @p0,@p1,@p2,@p3
        )",
                this.idArbolGenealogicoGNO,
                this.arbolGenealogicoGNO,
                "Paterna",
                user.Id
            );
    }
    else
    {
        int ret1 = ctx.Database.ExecuteSqlCommand(
        @"UPDATE [v_ImagenArbolGenealogicoGNOCG] SET
            [ArbolGenealogicoGNO] = @p1
            ,[Rama] = @p2
            ,[idUsuarioCV] = @p3
        WHERE [IdArbolGenealogicoGNO] = @p0
        ",
            familiaCG.IdArbolGenealogicoGNO,
            this.arbolGenealogicoGNO,
            "Paterna",
            user.Id
        );
    }
}
```

```
        if (this.arbolGenealogicoPDF != null && this.arbolGenealogicoPDF.Length != 0)
        {
            if (this.idArbolGenealogicoPDF == null)
            {
                this.idArbolGenealogicoPDF = Guid.NewGuid();
                int ret1 = ctx.Database.ExecuteSqlCommand(
                    @"INSERT INTO v_ImagenArbolGenealogicoPDFCG(
                      [IdArbolGenealogicoPDF]
                     ,[ArbolGenealogicoPDF]
                     ,[Rama]
                     ,[idUsuarioCV]
                ) VALUES(
                        @p0,@p1,@p2,@p3
                )",
                        this.idArbolGenealogicoPDF,
                        this.arbolGenealogicoPDF,
                        "Paterna",
                        user.Id
                    );
            }
            else
            {
                int ret1 = ctx.Database.ExecuteSqlCommand(
                @"UPDATE [v_ImagenArbolGenealogicoPDFCG] SET
                    [ArbolGenealogicoPDF] = @p1
                    ,[Rama] = @p2
                    ,[idUsuarioCV] = @p3
                WHERE [IdArbolGenealogicoPDF] = @p0
                ",
                    familiaCG.IdArbolGenealogicoPDF,
                    this.arbolGenealogicoPDF,
                    "Paterna",
                    user.Id
                );
            }
        }
```

```
            int ret = ctx.Database.ExecuteSqlCommand(
                @"UPDATE [v_FamiliaCG] SET
                    [fechaInclusion] = @p1
                    ,[idCriterioSeleccionRiesgo] = @p2
                    ,[idArbolGenealogicoGNO] = @p3
                    ,[idArbolGenealogicoPDF] = @p4
                    ,[idUsuarioCV] = @p5
                WHERE [idFamiliaCG] = @p0
                ",
                    this.idFamiliaCG,
                    this.fechaInclusion,
                    this.idCriterioSeleccionRiesgo,
                    this.idArbolGenealogicoGNO,
                    this.idArbolGenealogicoPDF,
                    user.Id
                );

                tr.Commit();
            }
            catch (Exception e)
            {
                tr.Rollback();
                throw;
            }
        }


}
```

3.3.2 Controlador

Para los controladores utilizaremos principalmente dos funciones Create y Edit. Cada método tiene dos peticiones: GET y POST. La petición GET es obtener datos del servidor y POST es enviar información desde el cliente para que sea actualice o agregue información al servidor.

Además se añadirán dos métodos más llamados Diferencia y Versión usados por el módulo de control de versiones para mostrar los datos modificados de una versión a otra.

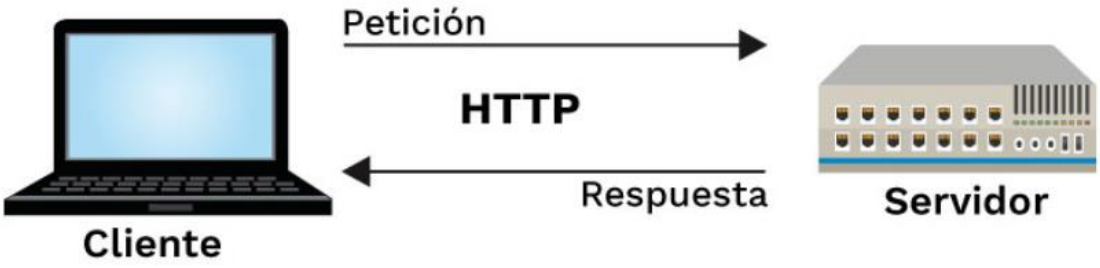

Ilustración 11: Diferencia entre GET y POST

- GET: Para mandar datos a la vista utilizaremos por un lado una serie de ViewBag[4] que se utilizaran en los diferentes SelectBox para mostrar las distintas posibilidades a seleccionar, y por otro lado, se utilizara el modelo que se definió en el apartado (3.3.1.1, formEnfermeria). Y finalmente devolveremos la vista con toda esa información.
- POST: Analizaremos el modelo cumplimentado a través de la vista y llamaremos a los métodos del modelo previamente descrito InsertarEn/GuardarEn.

[4] El modelo MVC (Model-View-Controller) estrictamente hablando puede pasar un único objeto a la vista, que sería el Modelo. Pero a veces la vista necesita más información. Ahí entra el ViewBag. Usando el ViewBag uno puede pasar cualquier dato/objeto adicional que se necesite en la vista, como un título, una lista de valores, etc.

3.3.2.1 Módulo de Enfermería (EnfermeriaController)

1. Create

- GET:

```
public ActionResult Create(Guid? idPaciente)
{
    paciente pac = paciente.Buscar(db, idPaciente);
    if (pac == null)
    {
        return new HttpNotFoundResult("Paciente no encontrado");
    }

    //General
    ViewBag.indiceEsfuerzoCuidador = db.enfermeria_IndiceEsfuerzoCuidador.OrderBy(x => x.nombreIndiceEsfuerzoCuidador).ToList();
    ViewBag.tipoValoracion = db.enfermeria_TipoValoracion.OrderBy(x => x.nombreTipoValoracion).ToList();
    ViewBag.centroSalud = db.enfermeria_CentroSalud.OrderBy(x => x.nombreCentroSalud).ToList();

    //Con/Res/Nut/Hid
    ViewBag.tipoRespiracion = db.enfermeria_TipoRespiracion.OrderBy(x => x.nombreTipoRespiracion).ToList();
    ViewBag.viasAereas = db.enfermeria_ViasAereas.OrderBy(x => x.nombreViasAereas).ToList();
    ViewBag.tos = db.enfermeria_Tos.OrderBy(x => x.nombreTos).ToList();
    ViewBag.secreciones = db.enfermeria_Secreciones.OrderBy(x => x.nombreSecreciones).ToList();
    ViewBag.coloracionPielMucosas = db.enfermeria_ColoracionPielMucosas.OrderBy(x => x.nombreColoracionPielMucosas).ToList();
    ViewBag.estadoHidratacion = db.enfermeria_EstadoHidratacion.OrderBy(x => x.nombreEstadoHidratacion).ToList();
    ViewBag.estadoNutricion = db.enfermeria_EstadoNutricion.OrderBy(x => x.nombreEstadoNutricion).ToList();
    ViewBag.tipoAlimentacion = db.enfermeria_TipoAlimentacion.OrderBy(x => x.nombreTipoAlimentacion).ToList();
    ViewBag.Denticion = db.enfermeria_Denticion.OrderBy(x => x.nombreDenticion).ToList();
    ViewBag.Deglucion = db.enfermeria_Deglucion.OrderBy(x => x.nombreDeglucion).ToList();

    ......

    var form = FormEnfermeria.Nueva(pac);
    ViewBag.CadenaPaciente = pac.NUHSA + " - " + pac.persona.NombreCompleto;

    if (Request.IsAjaxRequest())
    {
        return PartialView("_FormCreate", form);
    }
    return View(form);
}
```

- POST:

```
[HttpPost]
[Authorize(Roles = "Administrador,Médico, Enfermera, Enfermera EECC")]
[ValidateAntiForgeryToken]
public ActionResult Create(FormEnfermeria formEnfermeria)
{
    var idPaciente = formEnfermeria.idPaciente;

    paciente model = paciente.Buscar(db, idPaciente);
    if (model == null)
    {
        return new HttpNotFoundResult("Paciente no encontrado");
    }
    if (ModelState.IsValid)
    {
        formEnfermeria.InsertarEn(db, User.Identity.Name);
        if (Request.IsAjaxRequest())
        {
            return new HttpStatusCodeResult(HttpStatusCode.Accepted);
        }
    }
    else
    {
        Response.StatusCode = 422;
    }

    ViewBag.CadenaPaciente = model.NUHSA + " - " + model.persona.NombreCompleto;

    return PartialView("_FormEdit", formEnfermeria);
}
```

2. Edit

- GET:

```
public ActionResult Edit(Guid? id)
{
    if (!id.HasValue)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    enfermeria_Valoracion model = enfermeria_Valoracion.Buscar(db, id);
    if (model == null)
    {
        return HttpNotFound();
    }

    //ViewBags para la vista
    ViewBag.CadenaPaciente = model.paciente.NUHSA + " - " + model.paciente.persona.NombreCompleto;

    //General
    ViewBag.indiceEsfuerzoCuidador = db.enfermeria_IndiceEsfuerzoCuidador.OrderBy(x => x.nombreIndiceEsfuerzoCuidador).ToList();
    ViewBag.tipoValoracion = db.enfermeria_TipoValoracion.OrderBy(x => x.nombreTipoValoracion).ToList();
    ViewBag.centroSalud = db.enfermeria_CentroSalud.OrderBy(x => x.nombreCentroSalud).ToList();

    ......

    //Modelo con los atributos
    var form = FormEnfermeria.Rellenar(model);
    if (Request.IsAjaxRequest())
    {
        return PartialView("_FormEdit", form);
    }
    return View(FormEnfermeria.Rellenar(model));
}
```

- POST:

```
[HttpPost]
[Authorize(Roles = "Administrador,Médico, Enfermera, Enfermera EECC")]
[ValidateAntiForgeryToken]
public ActionResult Edit(FormEnfermeria formEnfermeria)
{
    var idValoracion = formEnfermeria.idValoracion;

    enfermeria_Valoracion model = enfermeria_Valoracion.Buscar(db, idValoracion);
    if (model == null)
    {
        return new HttpNotFoundResult("Valoración no encontrada");
    }
    if (ModelState.IsValid)
    {
        formEnfermeria.GuardarEn(db, model, User.Identity.Name);
        if (Request.IsAjaxRequest())
        {
            return new HttpStatusCodeResult(HttpStatusCode.Accepted);
        }
    }
    else
    {
        Response.StatusCode = 422;
    }

    ViewBag.CadenaPaciente = model.paciente.NUHSA + " - " + model.paciente.persona.NombreCompleto;

    return PartialView("_FormEdit", formEnfermeria);
}
```

3. Versión

```
[Authorize(Roles = "Administrador")]
public async Task<ActionResult> Version(Guid id)
{
    var fila = await db.enfermeria_Valoracion.FindAsync(id);
    if (fila == null)
    {
        return HttpNotFound();
    }

    var version = FilaCV.CrearVersion(db, fila);

    return View("Version", model: version);
}
```

4. Diferencia

```
[Authorize(Roles = "Administrador")]
public async Task<ActionResult> Diferencia(Guid id)
{
    var fila = await db.enfermeria_Valoracion.FindAsync(id);
    if (fila == null)
    {
        return HttpNotFound();
    }

    var comparador = ComparadorCV.Crear(db, fila);

    return View("Diferencia", model: comparador);
}
```

3.3.2.2 Módulo de Consejo Genético (GestionFamiliasCG)

1. Create

- GET:

```
public ActionResult Create()
      {
          ViewBag.criterioSeleccionRiesgo = db.CriterioSeleccionRiesgoCG.OrderBy(x => x.NombreCriterio).ToList();

          var form = FormFamiliaCG.Nueva();
          if (Request.IsAjaxRequest())
          {
              return PartialView("_FormCreate", form);
          }
          return View(form);
      }
```

- POST: En este apartado se capturara el archivo subido desde la vista y se codificara como un array de Bytes para su inserción en la base de datos.

```
[HttpPost]
[Authorize(Roles = "Administrador,Médico, Enfermera, Enfermera EECC")]
[ValidateAntiForgeryToken]
public ActionResult Create(FormFamiliaCG formFamiliaCG)
{
    var imagenArbolGenealogicoGNO = Request.Files["imagenArbolGenealogicoGNO"];
    var imagenArbolGenealogicoPDF = Request.Files["imagenArbolGenealogicoPDF"];
    if ( imagenArbolGenealogicoGNO.ContentLength != 0)
    {
        //Determining file name. You can format it as you wish.
        string FileName = imagenArbolGenealogicoGNO.FileName;
        //Determining file size.
        int FileSize = imagenArbolGenealogicoGNO.ContentLength;
        //Creating a byte array corresponding to file size.
        byte[] FileByteArray = new byte[FileSize];
        //Posted file is being pushed into byte array.
        imagenArbolGenealogicoGNO.InputStream.Read(FileByteArray, 0, FileSize);
        //Uploading properly formatted file to server.
        formFamiliaCG.arbolGenealogicoGNO = FileByteArray;
        formFamiliaCG.idArbolGenealogicoGNO = Guid.NewGuid();
    }
    if (imagenArbolGenealogicoPDF.ContentLength != 0)
    {
        //Determining file name. You can format it as you wish.
        string FileName = imagenArbolGenealogicoPDF.FileName;
        //Determining file size.
        int FileSize = imagenArbolGenealogicoPDF.ContentLength;
        //Creating a byte array corresponding to file size.
        byte[] FileByteArray = new byte[FileSize];
        //Posted file is being pushed into byte array.
        imagenArbolGenealogicoPDF.InputStream.Read(FileByteArray, 0, FileSize);
        //Uploading properly formatted file to server.
        formFamiliaCG.arbolGenealogicoPDF = FileByteArray;
        formFamiliaCG.idArbolGenealogicoPDF = Guid.NewGuid();
    }

    if (ModelState.IsValid)
    {
        formFamiliaCG.InsertarEn(db, User.Identity.Name);
        if (Request.IsAjaxRequest())
        {
            return new HttpStatusCodeResult(HttpStatusCode.Accepted);
        }
    }
    else
    {
        Response.StatusCode = 422;
    }
    return PartialView("_FormEdit", formFamiliaCG);
}
```

2. Edit

- GET:

```
public ActionResult Edit(Guid? id)
{
    if (!id.HasValue)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    FamiliaCG model = FamiliaCG.Buscar(db, id);
    if (model == null)
    {
        return HttpNotFound();
    }

    ViewBag.criterioSeleccionRiesgo = db.CriterioSeleccionRiesgoCG.OrderBy(x => x.NombreCriterio).ToList();

    //Modelo con los atributos
    var form = FormFamiliaCG.Rellenar(model);
    if (Request.IsAjaxRequest())
    {
        return PartialView("_FormEdit", form);
    }
    return View(form);
}
```

- POST: Al igual que en el método POST de *Create* anteriormente mencionado, hay que capturar el archivo subido y codificarlo como un array de Bytes, que posteriormente se harán los correspondientes *insert/update* desde el modelo *formFamiliaCG.*

```
[HttpPost]
[Authorize(Roles = "Administrador,Médico, Enfermera, Enfermera EECC")]
[ValidateAntiForgeryToken]
public ActionResult Edit(FormFamiliaCG formFamiliaCG)
{
    var idFamilia = formFamiliaCG.idFamiliaCG;
    var imagenArbolGenealogicoGNO = Request.Files["imagenArbolGenealogicoGNO"];
    var imagenArbolGenealogicoPDF = Request.Files["imagenArbolGenealogicoPDF"];
    if (imagenArbolGenealogicoGNO.ContentLength != 0)
    {

        //Determining file name. You can format it as you wish.
        string FileName = imagenArbolGenealogicoGNO.FileName;
        //Determining file size.
        int FileSize = imagenArbolGenealogicoGNO.ContentLength;
        //Creating a byte array corresponding to file size.
        byte[] FileByteArray = new byte[FileSize];
        //Posted file is being pushed into byte array.
        imagenArbolGenealogicoGNO.InputStream.Read(FileByteArray, 0, FileSize);
        //Uploading properly formatted file to server.
        formFamiliaCG.arbolGenealogicoGNO = FileByteArray;
    }
    if (imagenArbolGenealogicoPDF.ContentLength != 0)
    {

        //Determining file name. You can format it as you wish.
        string FileName = imagenArbolGenealogicoPDF.FileName;
        //Determining file size.
        int FileSize = imagenArbolGenealogicoPDF.ContentLength;
        //Creating a byte array corresponding to file size.
        byte[] FileByteArray = new byte[FileSize];
        //Posted file is being pushed into byte array.
        imagenArbolGenealogicoPDF.InputStream.Read(FileByteArray, 0, FileSize);
        //Uploading properly formatted file to server.
        formFamiliaCG.arbolGenealogicoPDF = FileByteArray;
    }
    FamiliaCG model = FamiliaCG.Buscar(db, idFamilia);
    if (model == null)
    {
        return new HttpNotFoundResult("Familia no encontrada");
    }
    if (ModelState.IsValid)
    {
        formFamiliaCG.GuardarEn(db, model, User.Identity.Name);
        if (Request.IsAjaxRequest())
        {
            return new HttpStatusCodeResult(HttpStatusCode.Accepted);
        }
    }
    else
    {
        Response.StatusCode = 422;
    }
    return PartialView("_FormEdit", formFamiliaCG);
```

3. Versión:

```
[Authorize(Roles = "Administrador")]
public async Task<ActionResult> Version(Guid id)
{
    var fila = await db.FamiliaCG.FindAsync(id);
    if (fila == null)
    {
        return HttpNotFound();
    }

    var version = FilaCV.CrearVersion(db, fila);

    return View("Version", model: version);
}
```

4. Diferencia:

```
[Authorize(Roles = "Administrador")]
public async Task<ActionResult> Diferencia(Guid id)
{
    var fila = await db.FamiliaCG.FindAsync(id);
    if (fila == null)
    {
        return HttpNotFound();
    }

    var comparador = ComparadorCV.Crear(db, fila);

    return View("Diferencia", model: comparador);
}
```

5. Consultas JSON: Añadiremos una serie de consultas JSON debido al problema mencionado en la sección (3.4.1).

```
public ActionResult json_list_Pacientes()
{
    var datos = PacienteCG.QActuales(db).Select(f => new {
        Numerofamilia = f.FamiliaCG.cod.ToString(),
        NumeroHistoria = f.paciente.numeroHistoria,
        Nombre = f.paciente.persona.nombre,
        Apellido1 = f.paciente.persona.apellido1,
        Apellido2 = f.paciente.persona.apellido2,
        Telefono = f.paciente.persona.telefono1 != null ? f.paciente.persona.telefono1 : f.paciente.persona.telefono2,
        PF = f.ProbandoFamiliar ? "P" : "F",
        Nuhsa = f.paciente.NUHSA,
        linkAPaciente = f.IdPaciente
    }).ToList();

    return Json(new { data = datos }, JsonRequestBehavior.AllowGet);
}

public ActionResult json_list_Familias()
{
    var datos = FamiliaCG.QActuales(db).Select(f => new {
        Numerofamilia = f.cod.ToString(),
        AnoInclusión = f.FechaInclusion.Year,
        Criterio = f.CriterioSeleccionRiesgoCG.NombreCriterio,
        Iniciales = f.CriterioSeleccionRiesgoCG.Iniciales,
        Edit = f.IdFamiliaCG,
        //Borrar = f.IdFamiliaCG,
        ArbolGNO = f.IdArbolGenealogicoGNO,
        ArbolPDF = f.IdArbolGenealogicoPDF
    }).ToList();

    return Json(new { data = datos }, JsonRequestBehavior.AllowGet);
}
```

6. Descarga de archivos GNO/PDF: Para identificar los archivos a la hora de descargarlos usaremos el patrón (código de la familia "Arbol GNO/PDF" .gno.

```
public async Task<ActionResult> VerArchivoGNO(Guid? idArbolGenealogicoGNO)
{
    if (!idArbolGenealogicoGNO.HasValue)
    {
        return new HttpNotFoundResult();
    }

    var file = await db.ImagenArbolGenealogicoGNOCG.FindAsync(idArbolGenealogicoGNO);
    if (file == null)
    {
        return new HttpNotFoundResult();
    }

    FamiliaCG familia = FamiliaCG.QActuales(db).Where( x=> x.IdArbolGenealogicoGNO == idArbolGenealogicoGNO).FirstOrDefault();

    string nombreCompleto = string.Format("{0}.{1}", FormFamiliaCG.generaCodigoGalen_1(familia) + "_Arbol_GNO", "gno");

    var mime = MimeMapping.GetMimeMapping(nombreCompleto);

    return File(
        file.ArbolGenealogicoGNO,
        mime,
        nombreCompleto
    );
}

public async Task<ActionResult> VerArchivoPDF(Guid? idArbolGenealogicoPDF)
{
    if (!idArbolGenealogicoPDF.HasValue)
    {
        return new HttpNotFoundResult();
    }

    var file = await db.ImagenArbolGenealogicoPDFCG.FindAsync(idArbolGenealogicoPDF);
    if (file == null)
    {
        return new HttpNotFoundResult();
    }


    FamiliaCG familia = FamiliaCG.QActuales(db).Where(x => x.IdArbolGenealogicoPDF == idArbolGenealogicoPDF).FirstOrDefault();

    string nombreCompleto = string.Format("{0}.{1}", FormFamiliaCG.generaCodigoGalen_1(familia) + "_Arbol_PDF", "pdf");

    var mime = MimeMapping.GetMimeMapping(nombreCompleto);

    return File(
        file.ArbolGenealogicoPDF,
        mime,
        nombreCompleto
    );
}
```

3.3.3 Vista

A la hora de estudiar ambos módulos, primero se mostrara una captura general del módulo correspondiente, seguido de pequeñas capturas por cada una de las acciones que puede realizar el usuario.

Todas las inserciones/ediciones se harán sobre un modal inyectado mediante JavaScript que se lanzara sobre la vista actual, al realizarse en varias ocasiones con distintos módulos se creó un asistente para cualquier módulo que aparece en las ilustraciones 12 y 13.

Se han ocultado los datos personales de los pacientes o usado pacientes ficticios debido a la criticidad de los datos.

```
function asistenteNuevoModal(selectorModal, selectorForm, claseBotonNueva, ferror) {
    var asistenteAux = function () {
        var $form = $(selectorForm);
        asistente($form);
        activarDatepicker($form.find("input[data-tipo='Calendario']"));
        renderCombo($form)
        renderComboMultiple($form)
    }
    asistenteAux();
    // Cuando se pulsa sobre un enlace de consulta se inyecta en el modal y se muestra
    $("body").on("click", claseBotonNueva, function (ev) {
        var $modal = $(selectorModal);
        var url = $modal.data("url");
        if ($(this).data("url") != undefined) url += $(this).data("url");
        $.ajax({
            type: "GET",
            url: url,
        })
            .done(function (data, textStatus, xhr) {
                var $newForm = $(selectorModal + " .modal-content");
                $newForm.html(xhr.responseText);
                // al reemplazar en el dom hay que religar los componentes asociados al formulario
                asistenteAux();
                $modal.modal("show");
            });

        ev.preventDefault();
        return false;
    });
    var continuar = false;
    $(selectorModal)
        .on("click", ".btn-guardar", function (ev) {
            continuar = $(this).is(".btn-continuar");
            $(selectorForm).steps("finish");
        })
        .on("submit", "form", function (ev) {
            var $form = $(this);
            var error = ferror($form);
            if (error != "") {
                swal("Existen errores", error);
            } else {
                $.ajax({
                    type: "POST",
                    url: $form.attr("action"),
                    data: $form.serialize()
                })
                    .done(function (data, textStatus, xhr) {
                        var $selector = $(selectorModal).one("hidden.bs.modal", function () {
                            location.reload();
                        });
                        if (!continuar) {
                            $selector.modal("hide")
                        }
                    })
            .fail(function (xhr, textStatus, errorText) {
                var $newForm = $(selectorModal + " .modal-content");
                if (xhr.status === 422) {
                    $newForm.html(xhr.responseText);
                    // al reemplazar en el dom hay que religar los componentes asociados al formulario
                    asistenteAux();
                } else {
                    $newForm.html(xhr.responseText);
                }
            });
    }
    ev.preventDefault();
    return false;
});
```

Ilustración 12: Función asistente de JavaScript para inyectar un modal de creación

```
function asistenteModal(selectorModal, selectorForm, claseEnlaceEdicion, dataModal, ferror ) {
    var asistenteAux = function () {
        var $form = $(selectorForm);
        asistente($form);
        activarDatepicker($form.find("input[data-tipo='Calendario']"));
        renderCombo($form)
        renderComboMultiple($form)
    }
    asistenteAux();
    // Cuando se pulsa sobre un enlace de consulta se inyecta en el modal y se muestra
    $("body").on("click", claseEnlaceEdicion, function (ev) {
        var $modal = $(selectorModal);
        var href = $(this).attr("href");
        $modal.attr(dataModal, href);
        $.ajax({
            type: "GET",
            url: href,
        })
            .done(function (data, textStatus, xhr) {
                var $newForm = $(selectorModal + " .modal-content");
                $newForm.html(xhr.responseText);
                // al reemplazar en el dom hay que religar los
                // componentes asociados al formulario
                asistenteAux();

                $modal.modal("show");
            });
        ev.preventDefault();
        return false;
    });
    var continuar = false;
    $(selectorModal)
        .on("click", ".btn-guardar", function (ev) {
            continuar = $(this).is(".btn-continuar");
            $(selectorForm).steps("finish");
        })
        .on("submit", "form", function (ev) {
            var $form = $(this);
            var error = ferror($form);
            if (error != "") {
                swal("Existen errores", error);
            } else {
                $.ajax({
                    type: "POST",
                    url: $form.attr("action"),
                    data: $form.serialize()
                })
                    .done(function (data, textStatus, xhr) {
                        var $selector = $(selectorModal).one("hidden.bs.modal", function () {
                            location.reload();
                        });
                        if (!continuar) {
                            $selector.modal("hide")
                        }
                    })
                    .fail(function (xhr, textStatus, errorText) {
                        var $newForm = $(selectorModal + " .modal-content");
                        if (xhr.status === 422) {
                            $newForm.html(xhr.responseText);
                            // al reemplazar en el dom hay que religar los
                            // componentes asociados al formulario
                            asistenteConsulta();
                        } else {
                            $newForm.html(xhr.responseText);
                        }
                    });
            }
            ev.preventDefault();
            return false;
        });
}
```

Ilustración 13: Función asistente de JavaScript para inyectar un modal de edición

3.3.3.1 Módulo de Enfermería

En el panel general de un paciente en concreto dentro del módulo de consultas, urgencias y anotaciones, se añade la posibilidad de realizar informes por parte del módulo de enfermería del centro correspondiente. En la ilustración 14 se muestra la apariencia que tendría dicho módulo, permitiéndonos crear, editar, eliminar u obtener el informe en formato PDF/WORD de alguna de las citas realizadas.

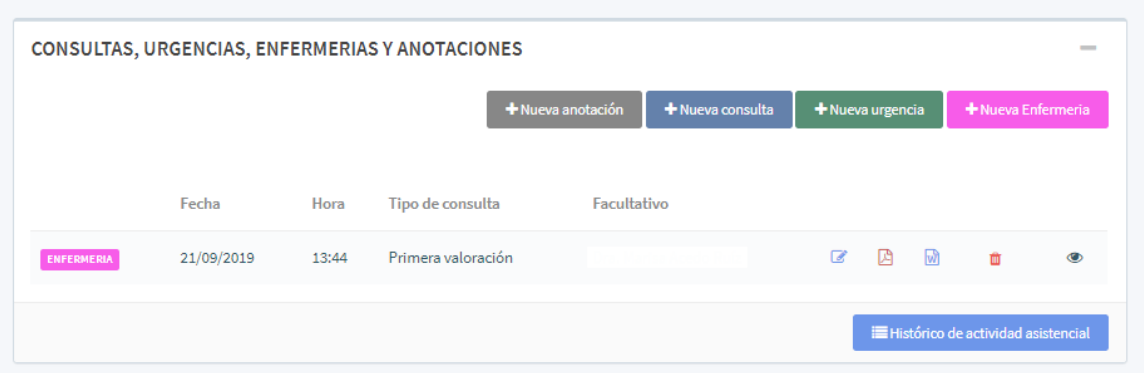

Ilustración 14: Panel general del módulo de enfermería.

En las ilustraciones 15-20 se muestra el modal que contiene el informe a cumplimentar por parte de los facultativos a la hora de crear una nueva enfermería.

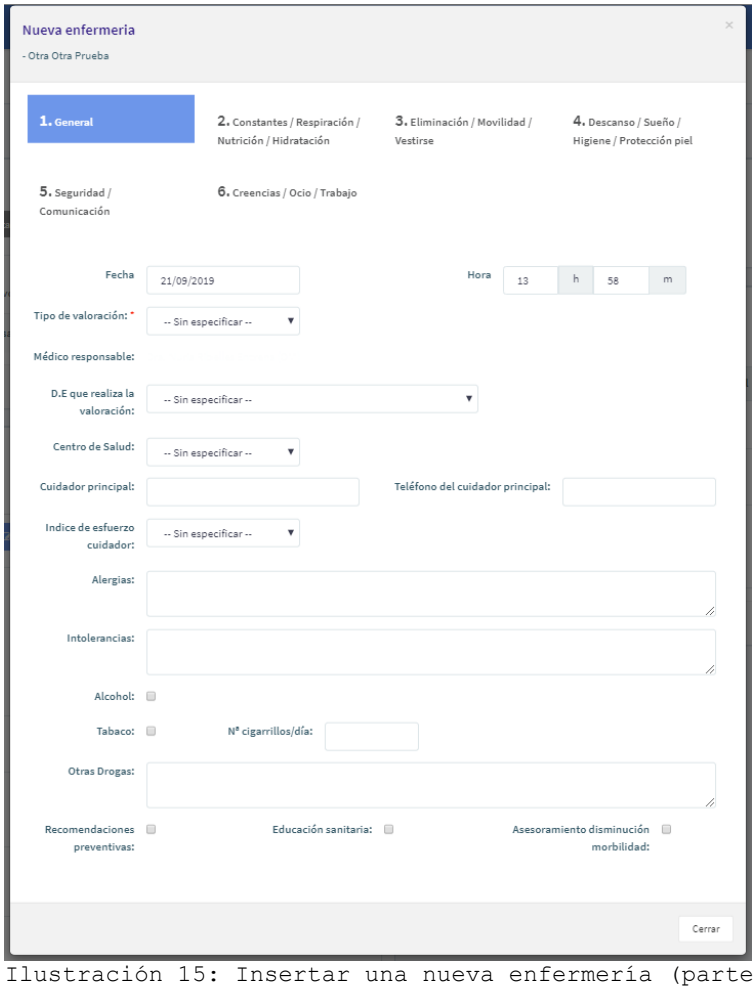

Ilustración 15: Insertar una nueva enfermería (parte 1)

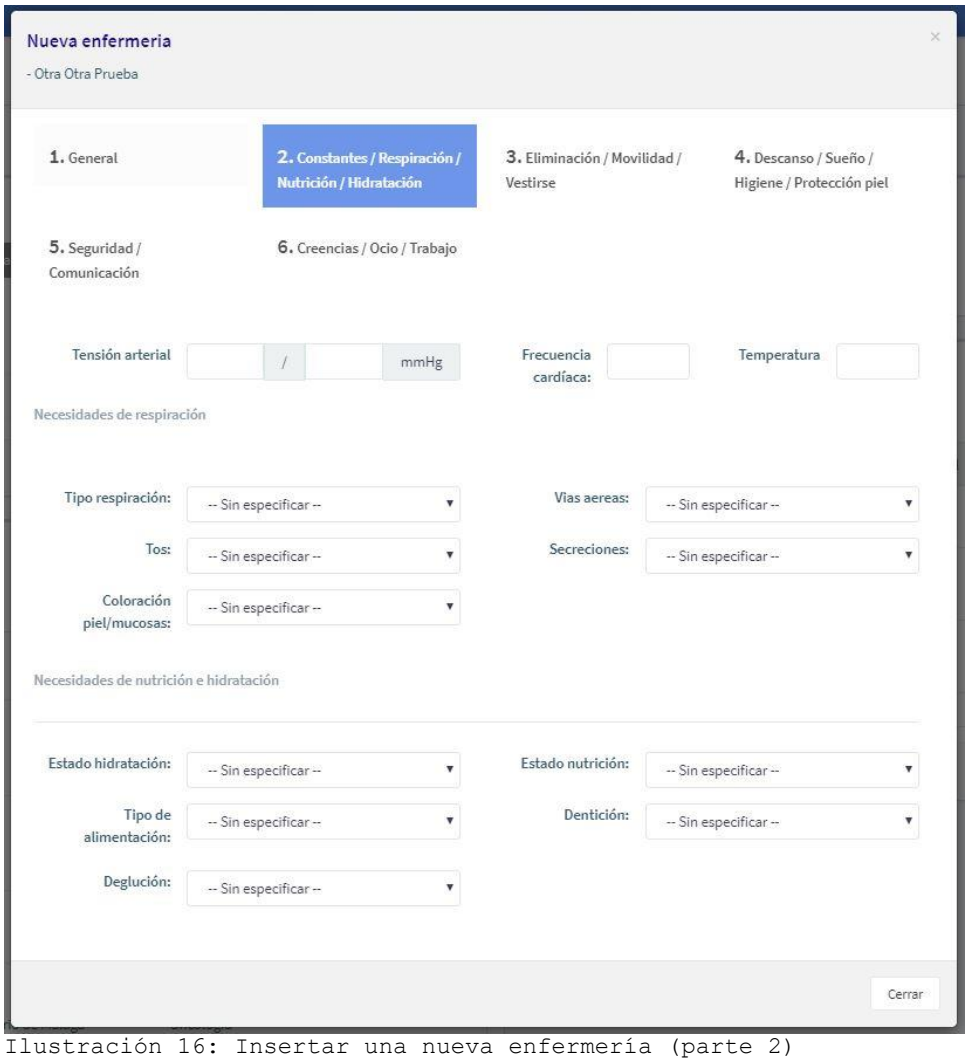

Ilustración 16: Insertar una nueva enfermería (parte 2)

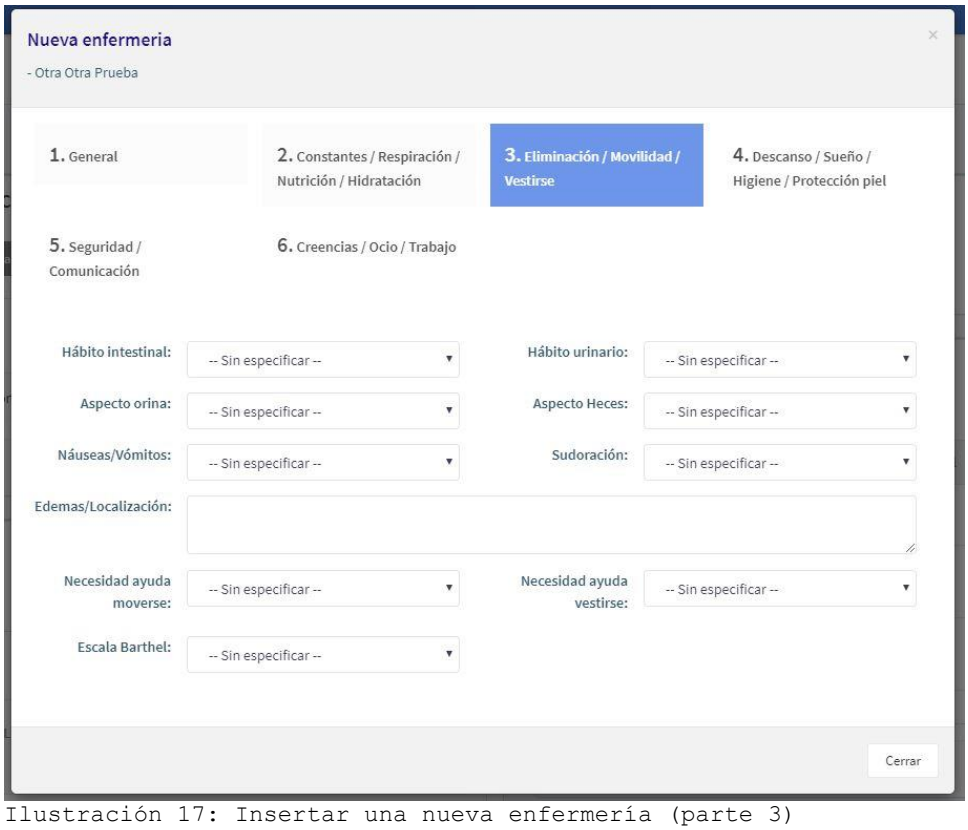

Ilustración 17: Insertar una nueva enfermería (parte 3)

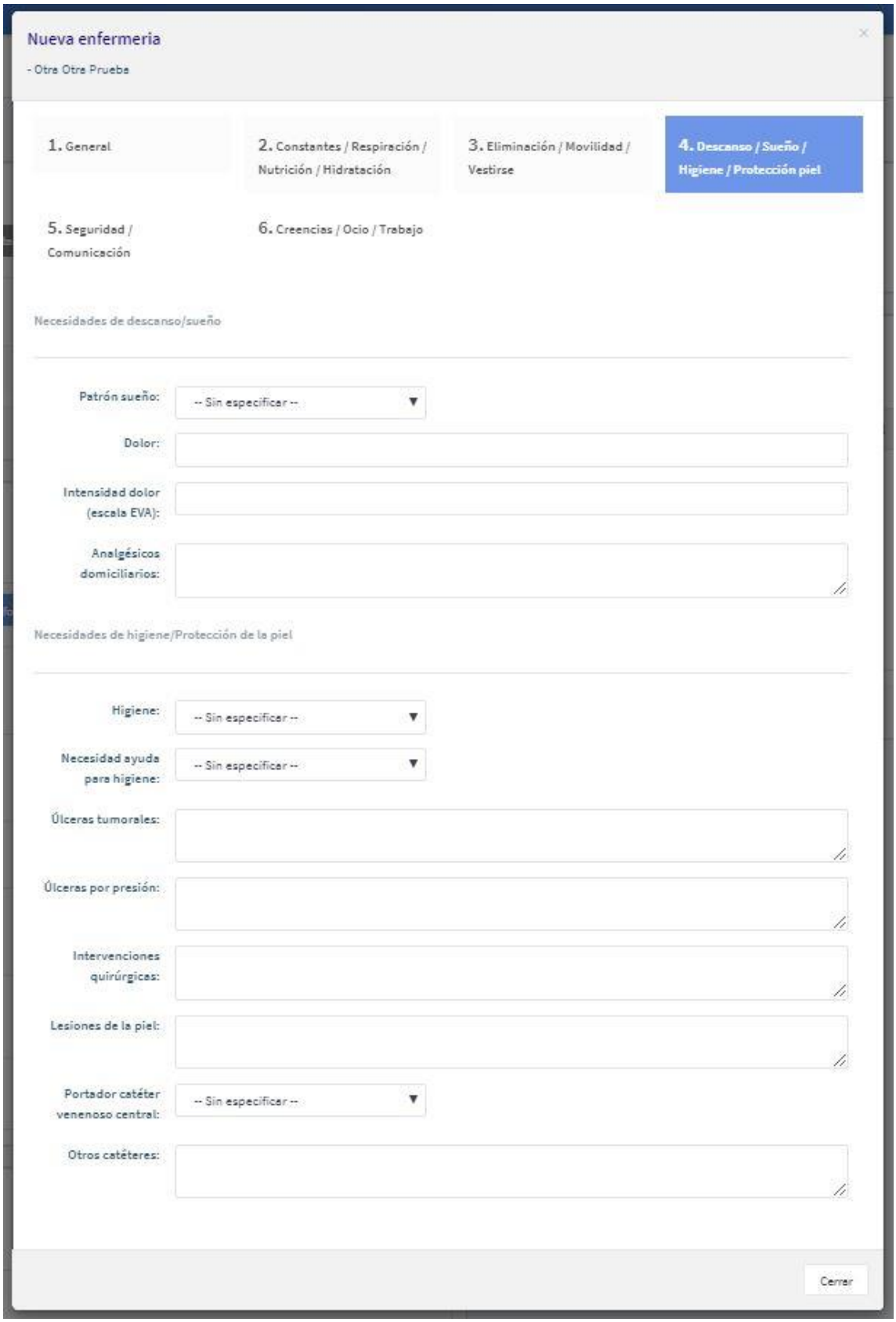

Ilustración 18: Insertar una nueva enfermería (parte 4)

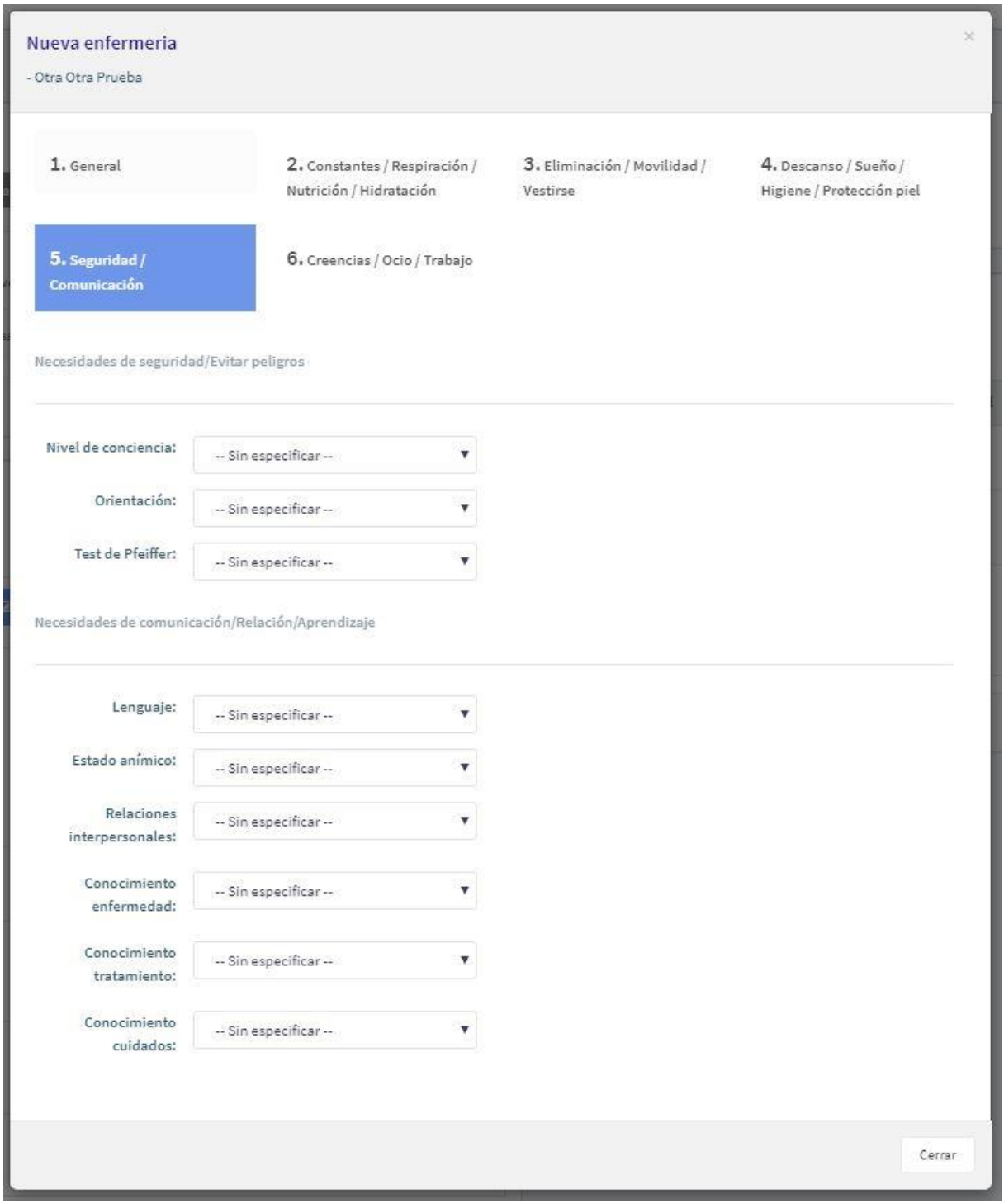

Ilustración 19: Insertar una nueva enfermería (parte 5)

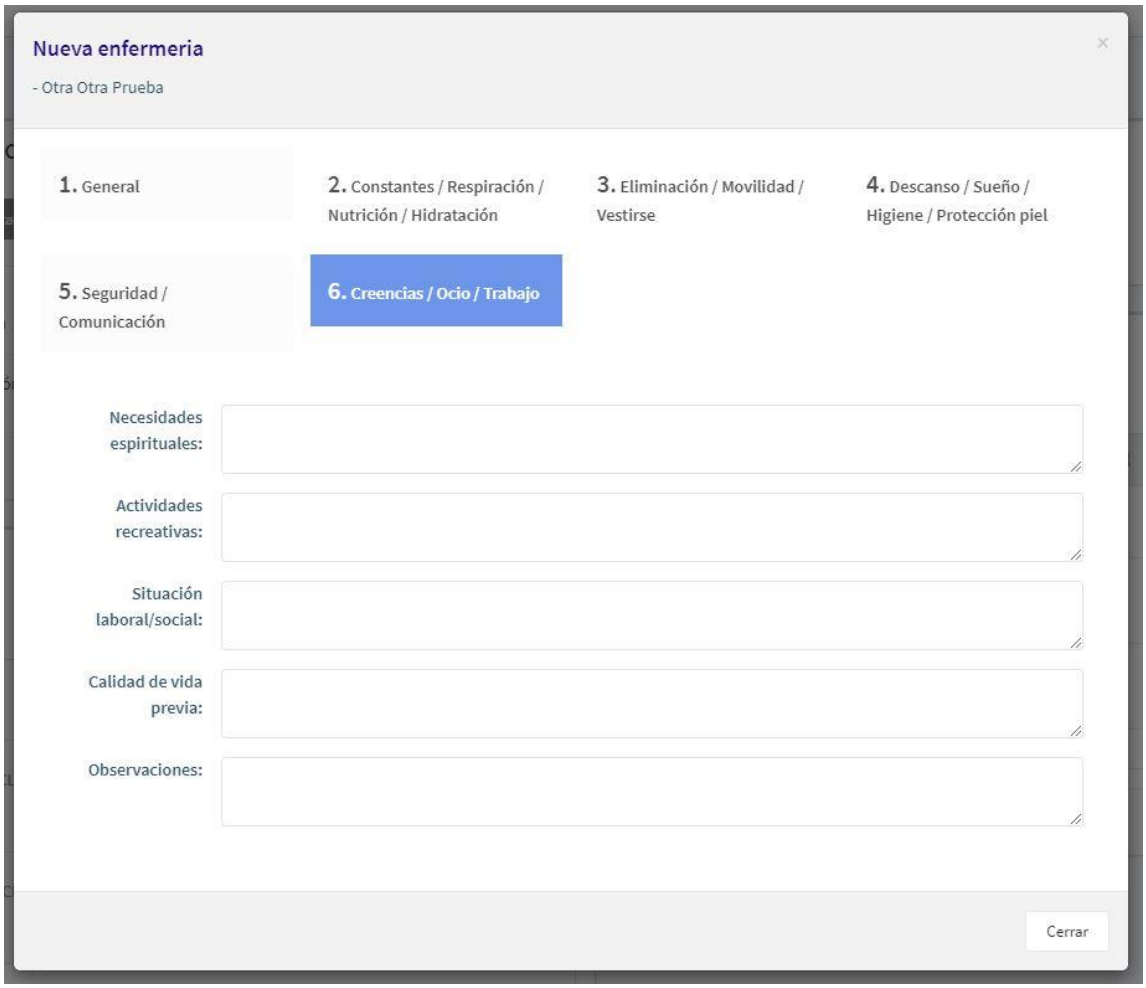

Ilustración 20: Insertar una nueva enfermería (parte 6)

A la hora de borrar una enfermería se mostrara el modal de la ilustración 21 para que el facultativo confirme el borrado.

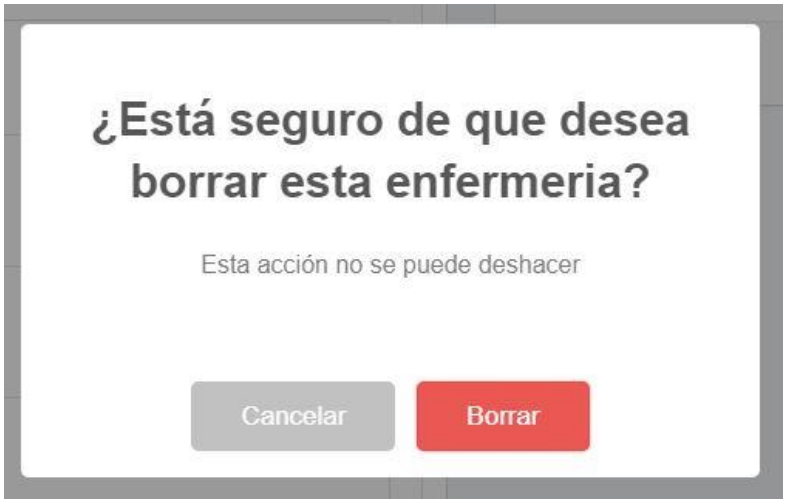

Ilustración 21: Borrar una enfermería

En lo referente al control de cambios, se ha añadido que recoja todas las modificaciones, creaciones o borrados sobre las enfermerías. En la ilustración 22 se muestra cual sería el resultado, en este caso al crear una enfermería con fecha '21/09/2019' se actualiza también un campo perteneciente a la tabla paciente donde aparece la última revisión que se le realizo, por eso aparecen ambos cambios.

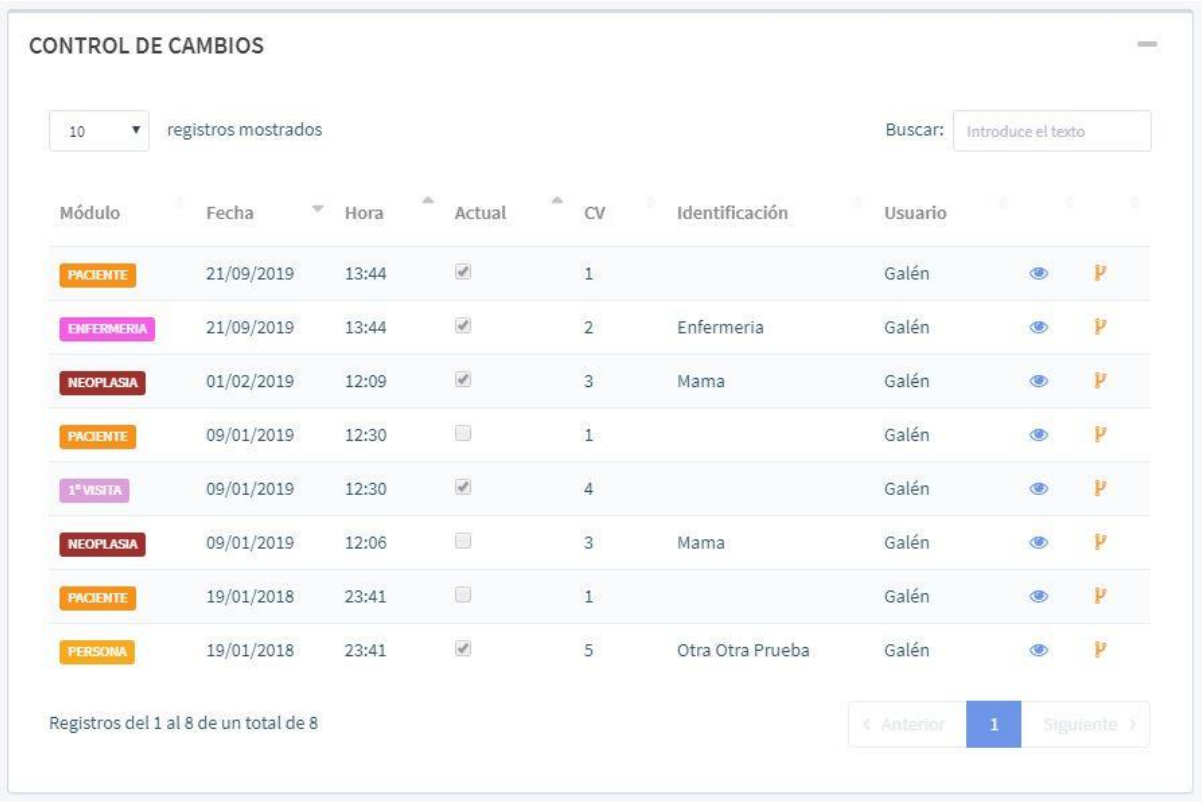

Ilustración 22: Control de cambios asociado a un paciente.

3.3.3.2 Módulo de Consejo Genético

En la ilustración 23 se muestra como seria la visión general del módulo de consejo genético. Permitiendo la búsqueda por cualquiera de sus campos de ambas tablas,

crear/editar una familia de consejo genético, así como un link desde cada paciente para acceder a su panel general.

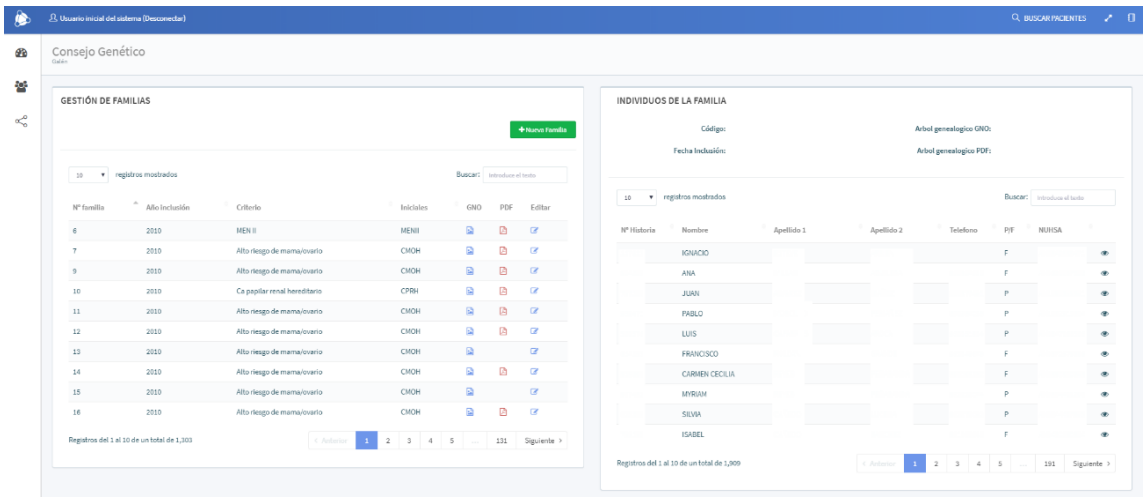

Ilustración 23: Visión general del módulo de consejo genético.

Al hacer '*click*' en una determinada familia deberán de filtrarse los pacientes para que solo aparezcan los que pertenecen a esa familia, además de aparecer información sobre la familia en el cuadro superior, permitiendo así ver rápidamente toda la información de la familia, descargar sus árboles genealógicos si fuese necesario y ver que pacientes pertenecen a esa familia.

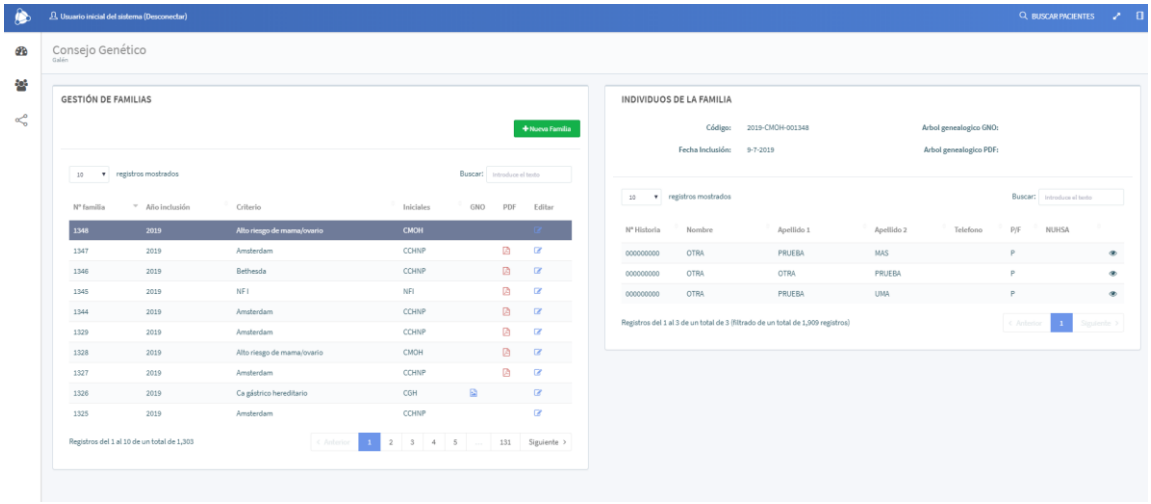

Ilustración 24: Filtrado de pacientes al hacer click en una familia.

Al hacer 'click' en un determinado paciente se deberán de filtrar los pacientes para que aparezcan solo los que pertenezcan a esa familia, además de mostrar en el cuadro superior la información de su familia.

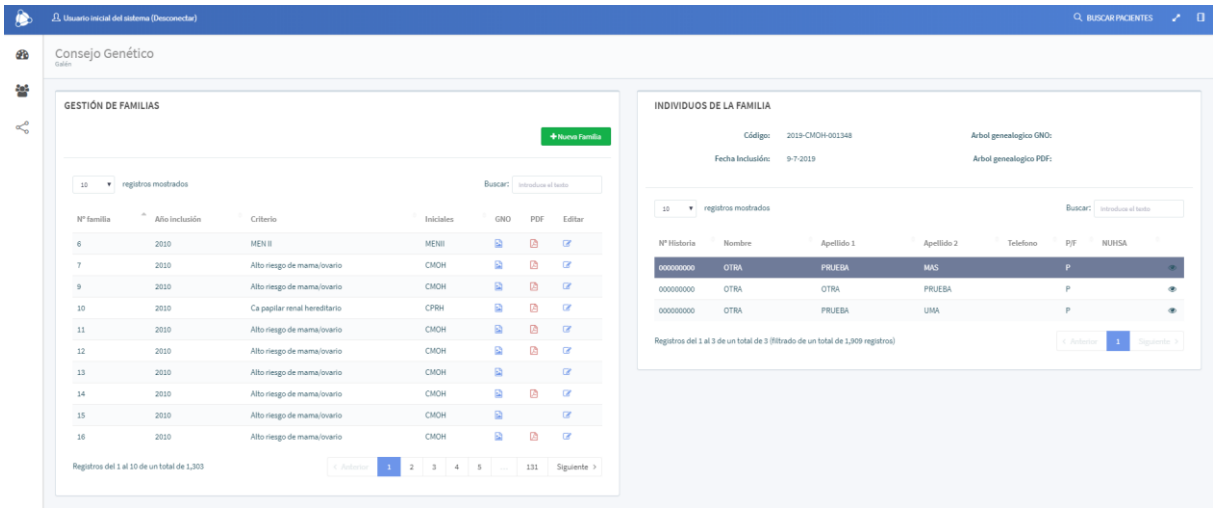

Ilustración 25: Filtrado de pacientes al hacer click en un paciente.

Al hacer '*click*' sobre el botón '*Nueva Familia*' situado en la sección de *Gestión de Familias* se mostrara el siguiente modal para cumplimentar sus datos. A petición del cliente la inserción de los pacientes en una determinada familia se realizara desde el panel del paciente en concreto.

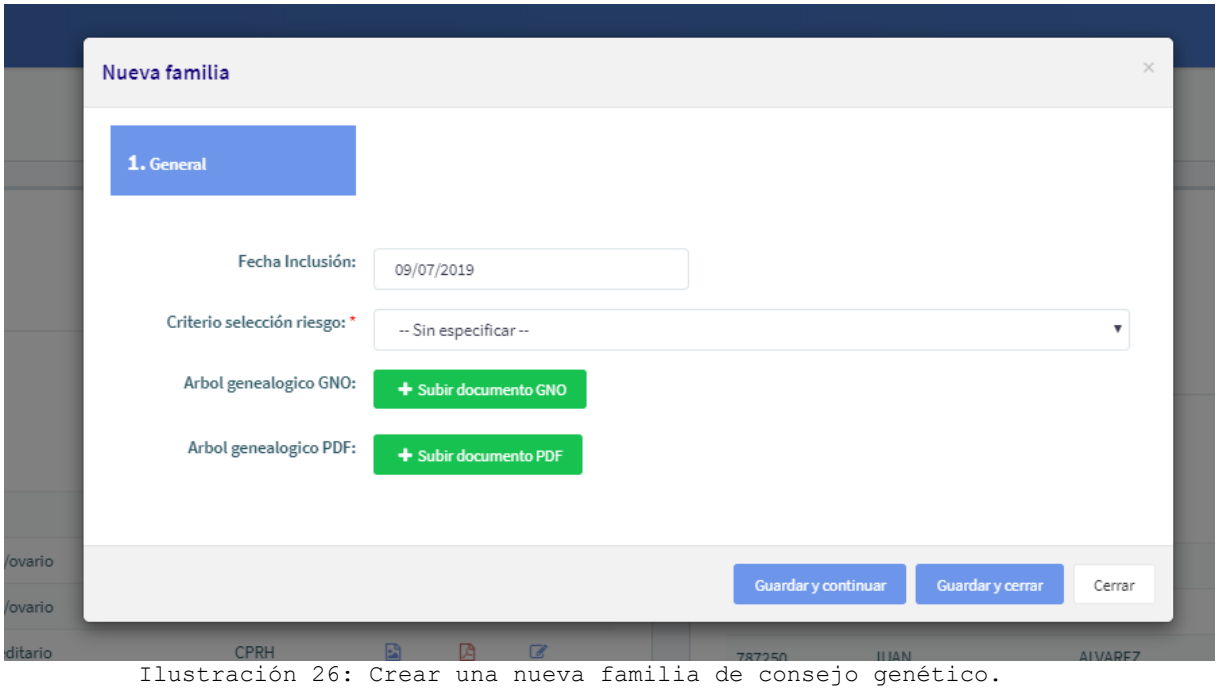

Ilustración 26: Crear una nueva familia de consejo genético.

En la edición aparecerán los campos rellenos además de permitir descargar los árboles genealógicos.

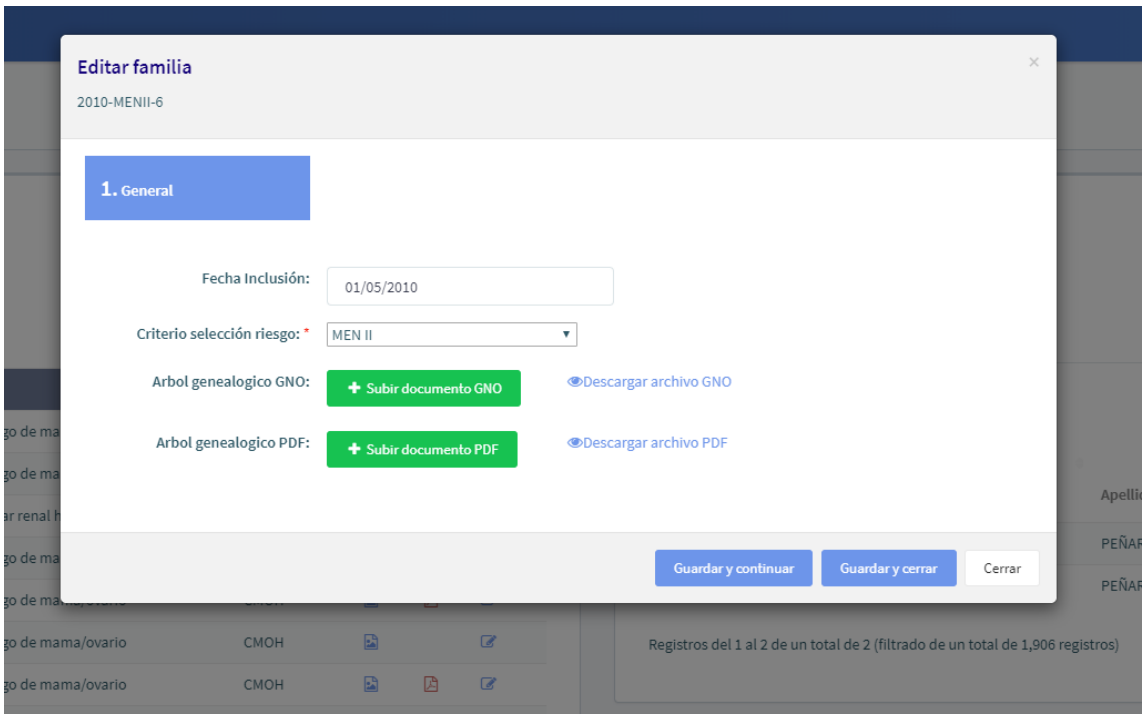

Ilustración 27: Editar una familia de consejo genético.

A la hora de incluir a un paciente en una familia de estudio de consejo genético, si este no pertenecía a ninguna familia de consejo genético, se mostrara dentro de su panel general el módulo de consejo genético con este diseño (ilustración 28).

Ilustración 28: Paciente sin familia de consejo genético asociada.

Cuando pulsamos sobre el botón "Incluir como paciente de estudio en consejo genético" se muestra el modal que aparece en la ilustración 29, donde es seleccionada la familia a la que se quiere adherir, además de cierta información necesaria para el estudio y las curvas de supervivencia pertinentes.

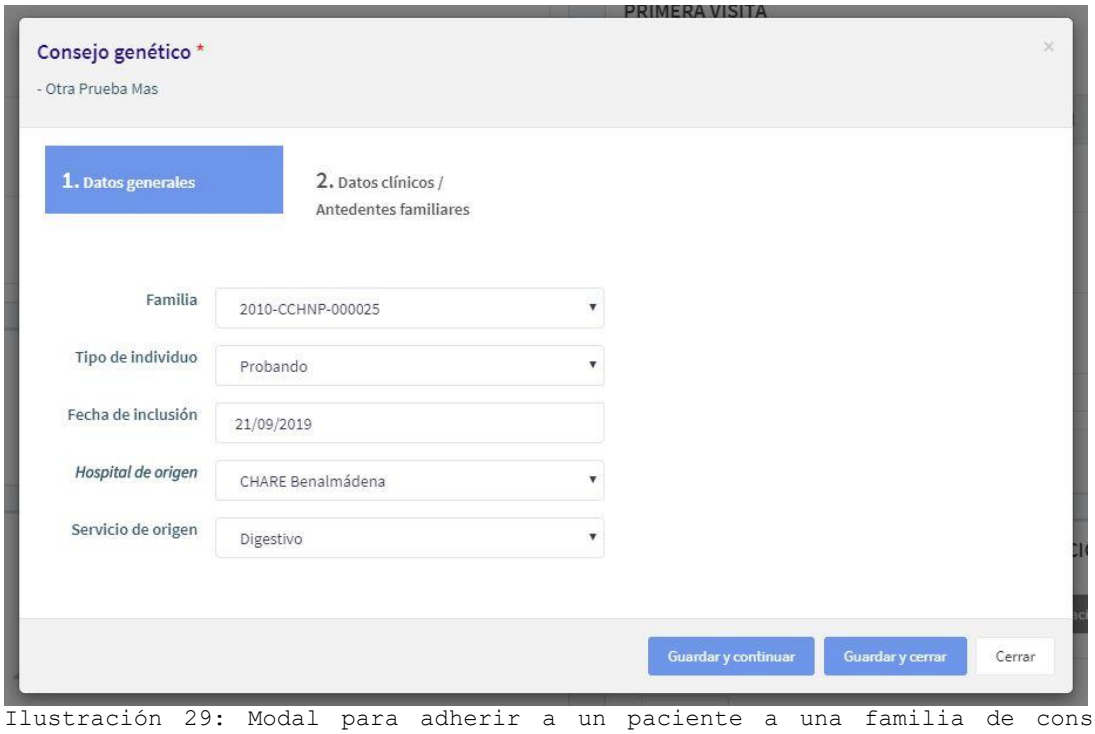

Ilustración 29: Modal para adherir a un paciente a una familia de consejo genético (parte 1).

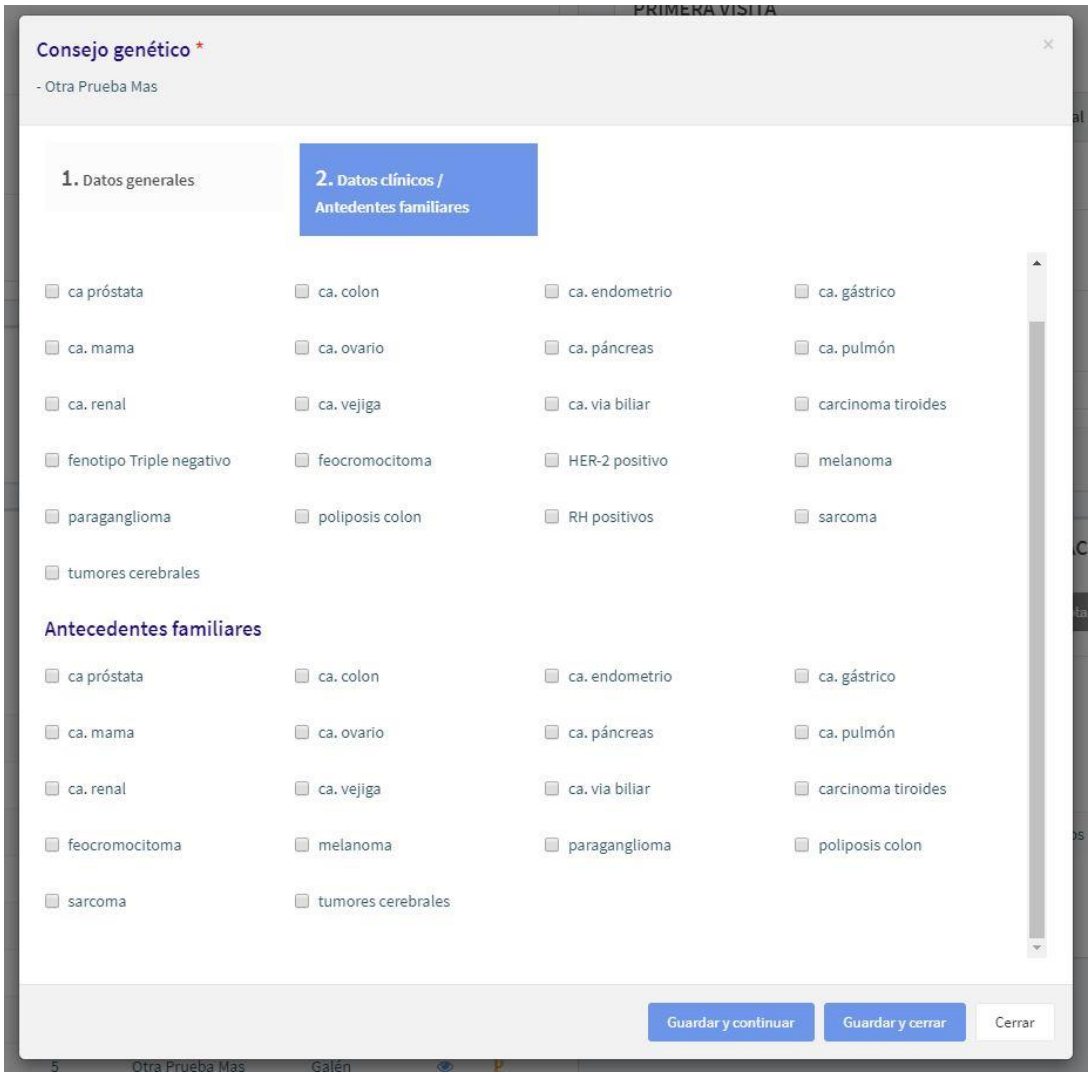

Ilustración 30: Modal para adherir a un paciente a una familia de consejo genético (parte 2).

Una vez incluido el paciente en la familia correspondiente (ej. 2010-CCHNP-000025), el módulo tendrá el aspecto detallado en la ilustración 31, donde el primer nivel de pestañas seleccionara la familia, y la segunda todos los datos que se pueden recoger del paciente asociado a dicha familia de consejo genético.

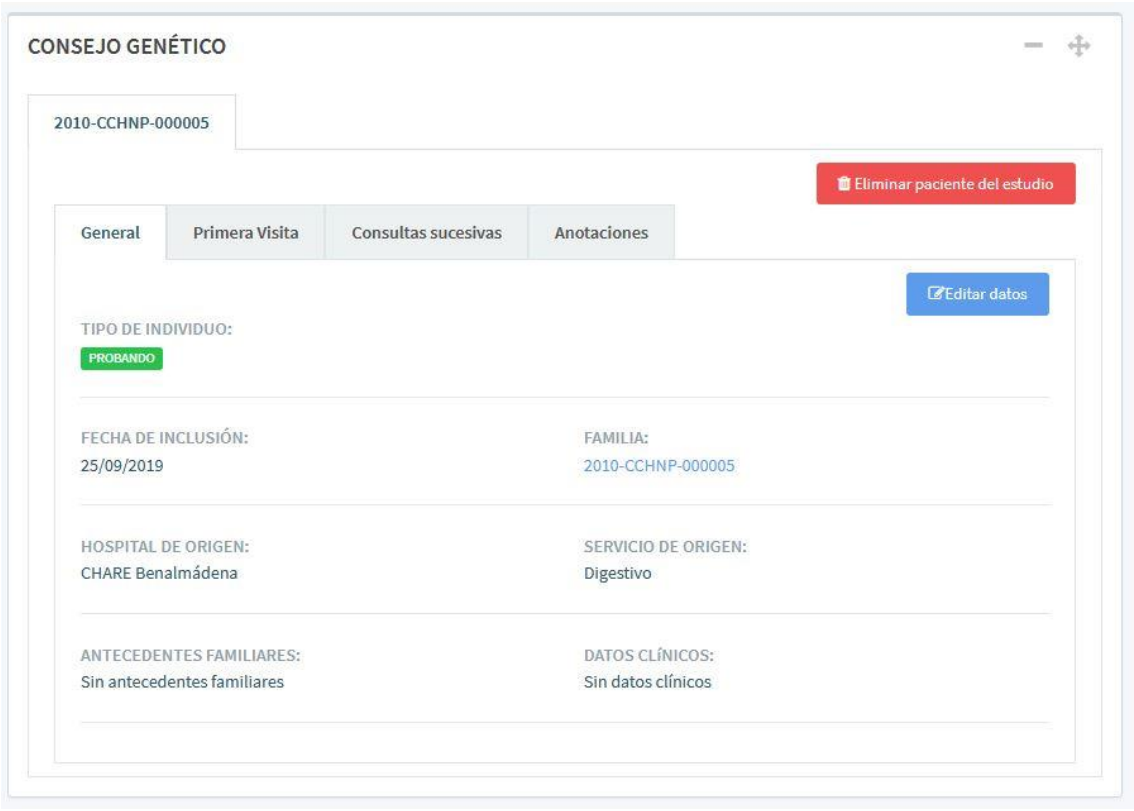

Ilustración 31: Vista del módulo de consejo genético con una familia asociada (General).

El enlace a la vista general de la familia nos direcciona a una simple vista similar a la de la ilustración 25, detallada en la ilustración 32.

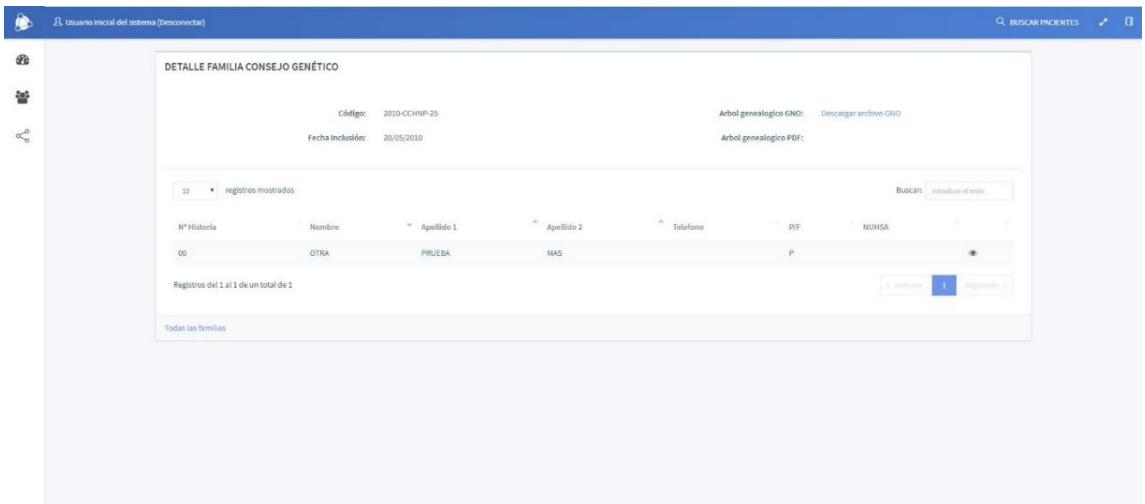

Ilustración 32: Vista general de una familia determinada.

En la pestaña "Primera Visita", solo se podrá guardar un único informe referente a la primera vez que acudió al médico dicho paciente.

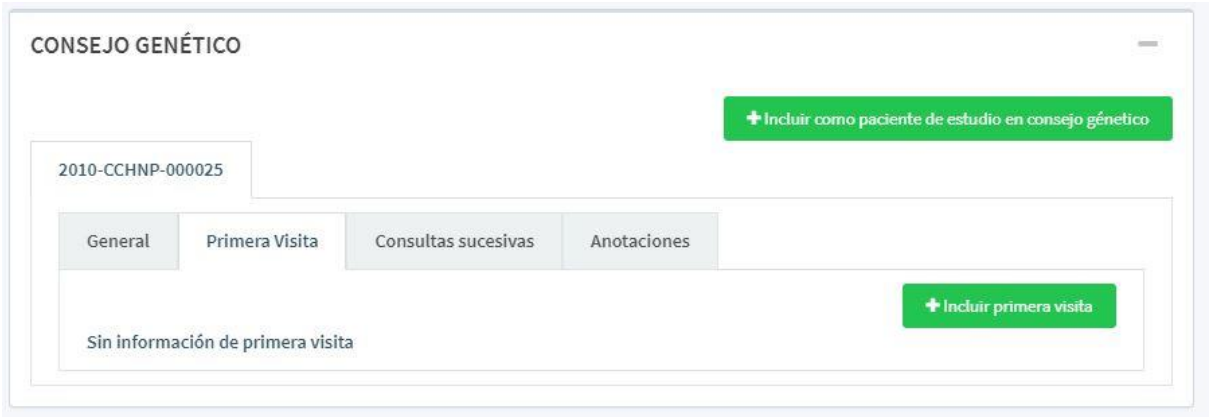

Ilustración 33: Vista del módulo de consejo genético con una familia asociada (Primera Visita), sin cumplimentar.

Al pulsar sobre el botón "Incluir primera visita" se mostrara el siguiente modal.

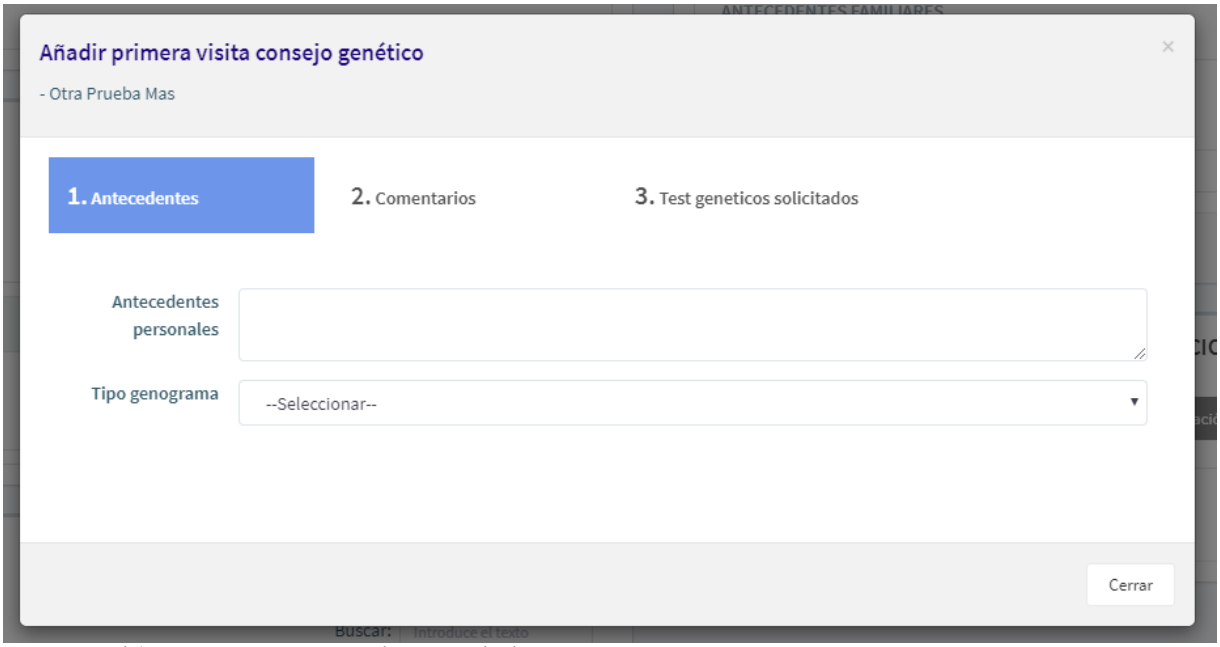

Ilustración 34: Creando Primera Visita (parte 1).

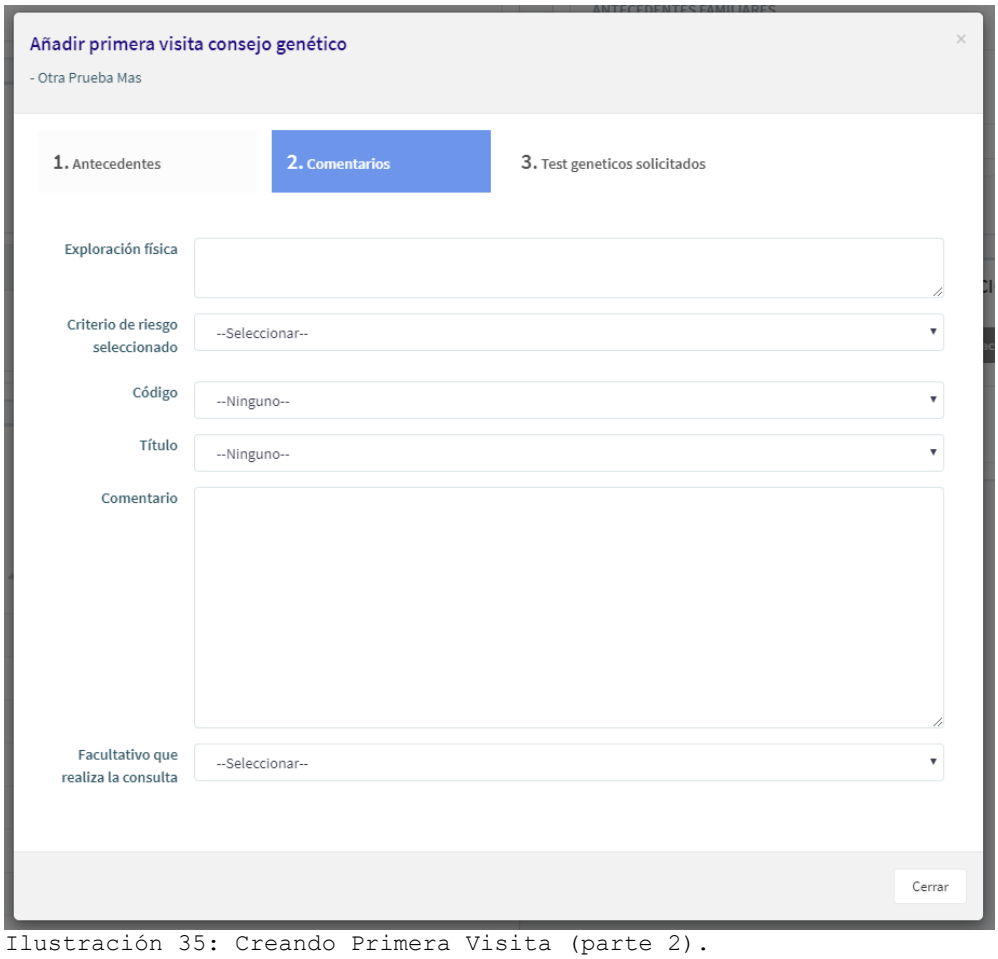

Ilustración 35: Creando Primera Visita (parte 2).

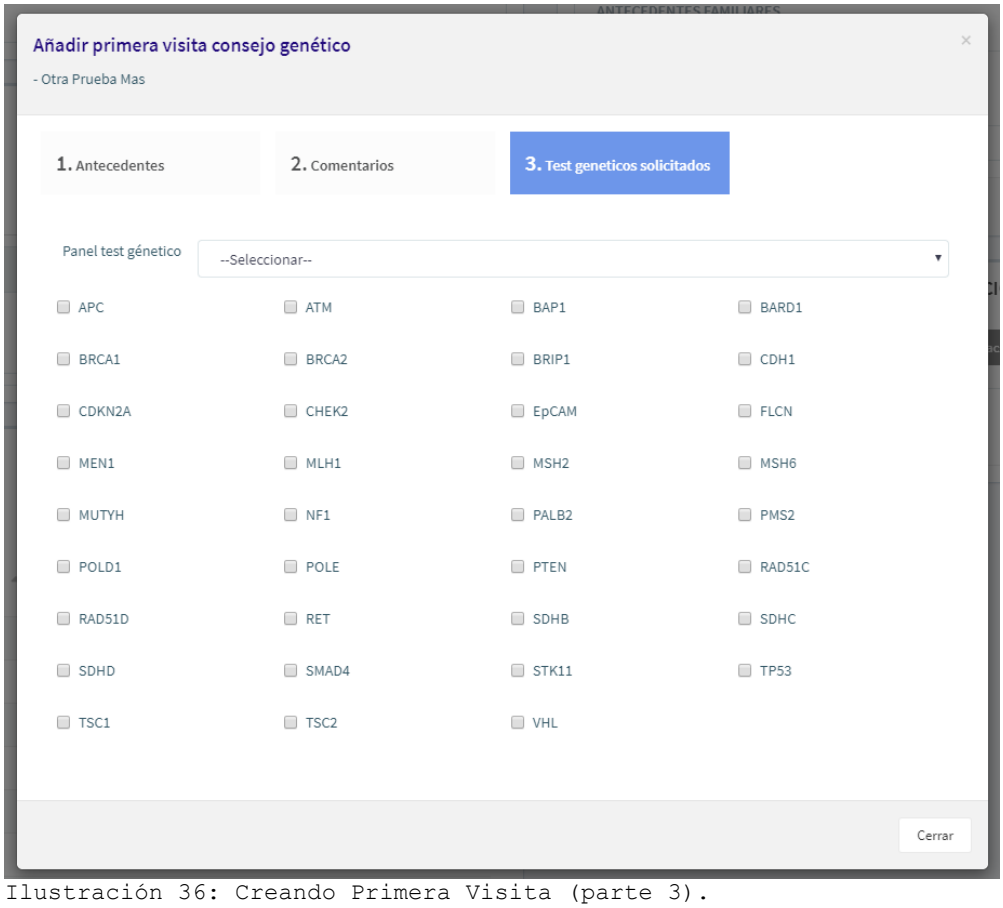

Ilustración 36: Creando Primera Visita (parte 3).

Respecto a la ilustración 36, se marcaran los test genéticos solicitados usando los checkBoxs mostrados pero se facilita un selectBox con algunas de las situaciones más encontradas la cual marcara los checkBoxs necesarios automáticamente ahorrando así una gran cantidad de tiempo a lo largo de la jornada laboral de los facultativos. Se muestra un ejemplo de uso en la ilustración 37.

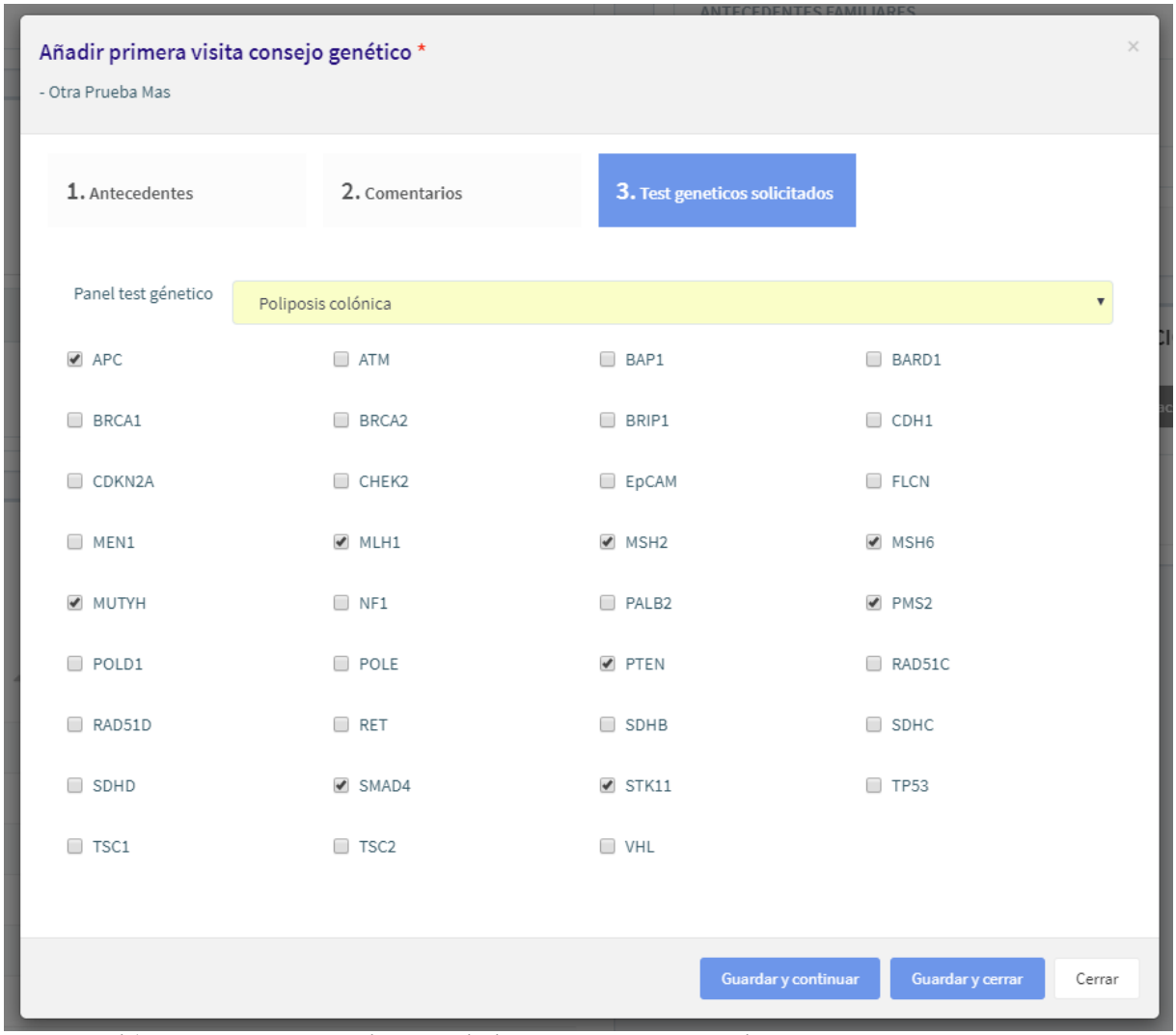

Ilustración 37: Creando Primera Visita (parte 3), haciendo uso de la ayuda.

Una vez rellenado el informe de la primera visita, el módulo de consejo genético tendrá el aspecto de la ilustración 38, permitiéndonos en todo momento editarla u obtener el informe en formato PDF/WORD.

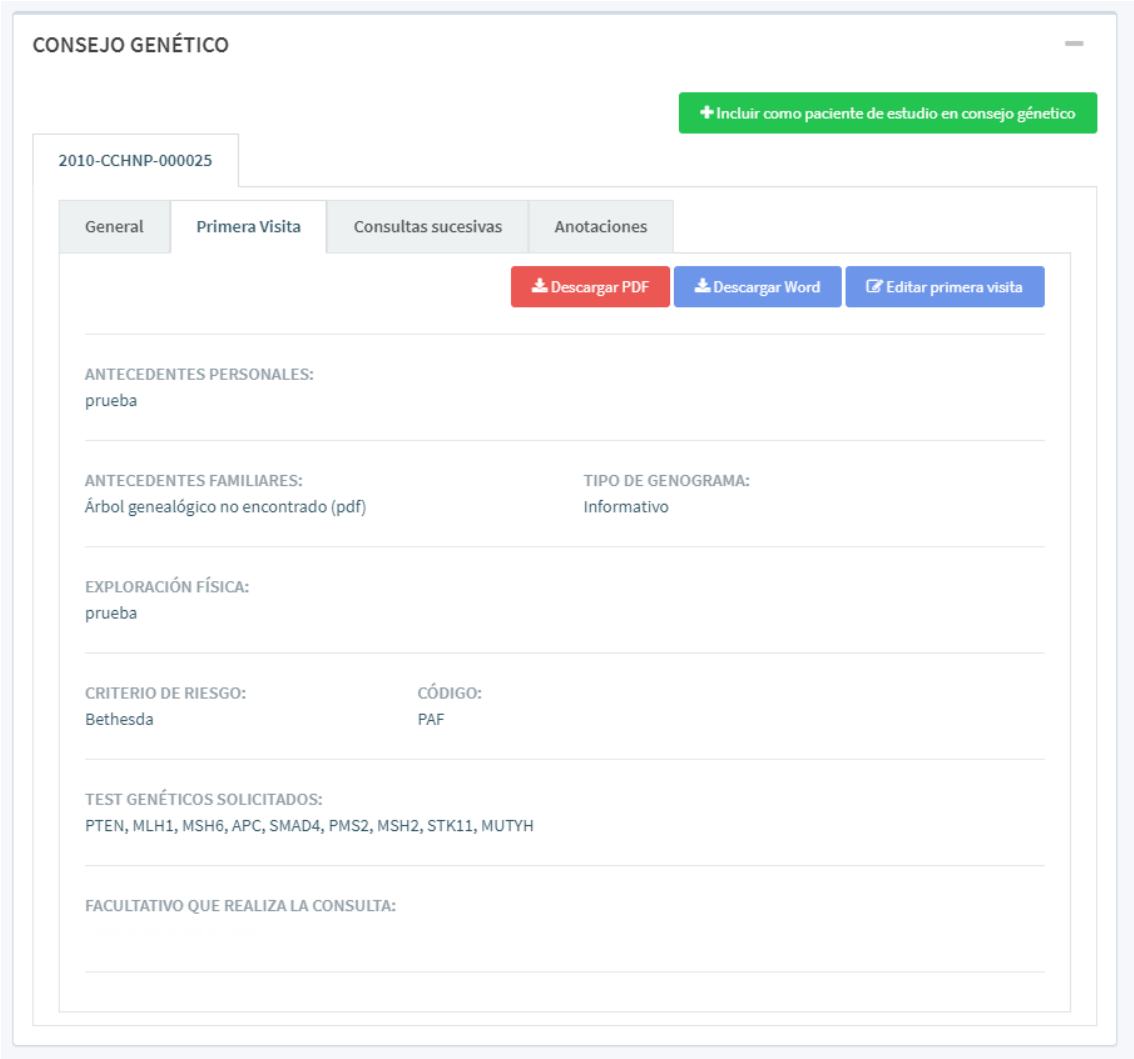

Ilustración 38: Vista del módulo de consejo genético con una familia asociada (Primera Visita).

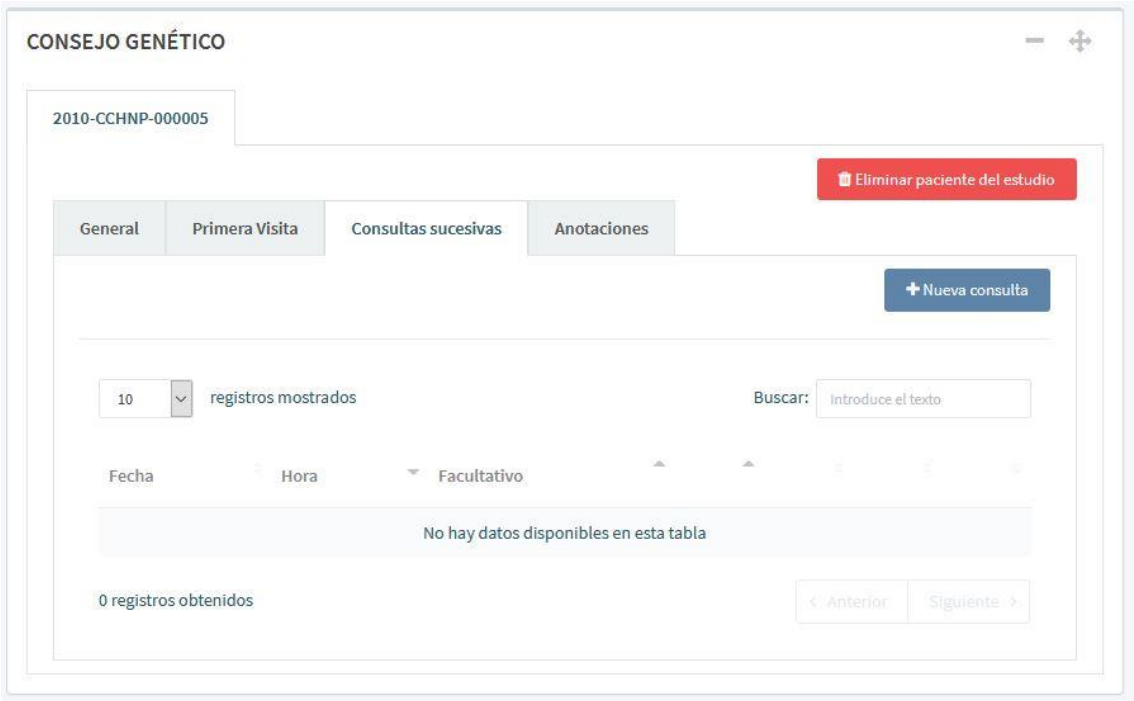

Ilustración 39: Vista del módulo de consejo genético con una familia asociada (Consultas sucesivas).

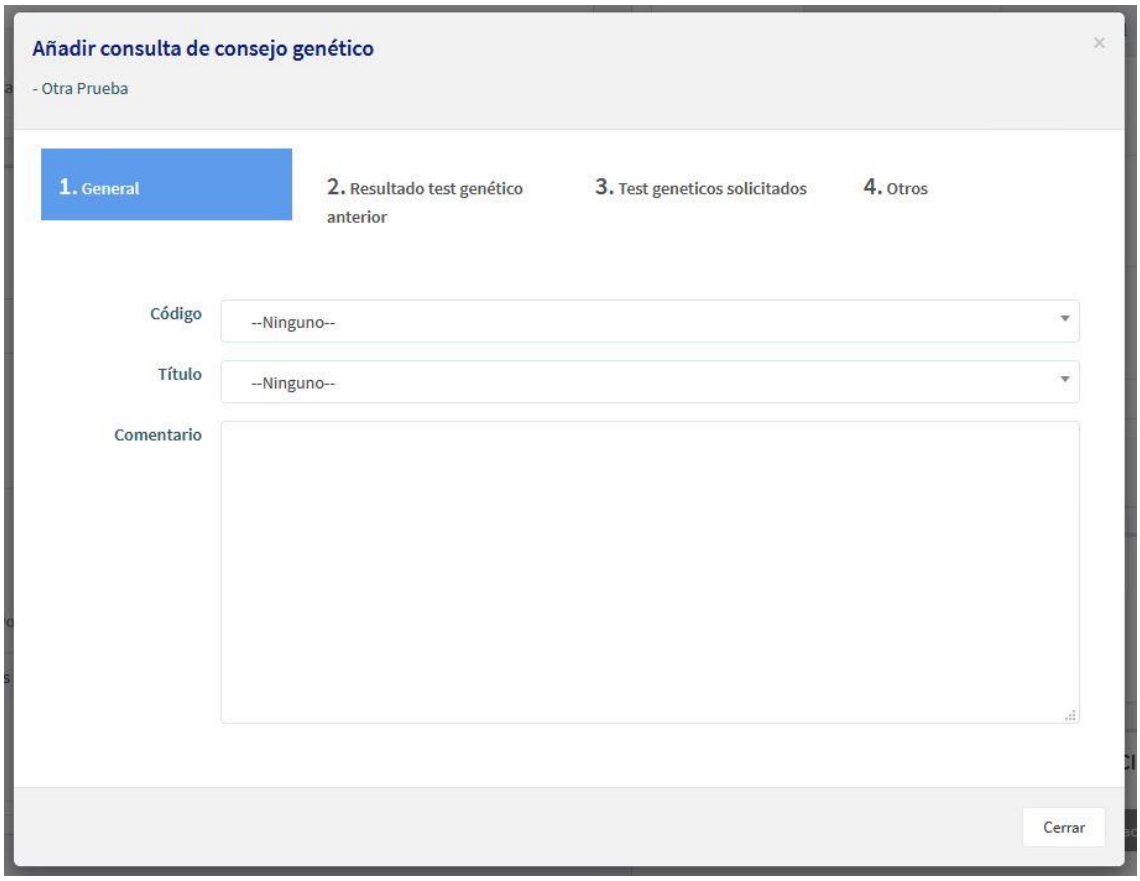

Ilustración 40: Añadir consulta de consejo genético (parte 1).

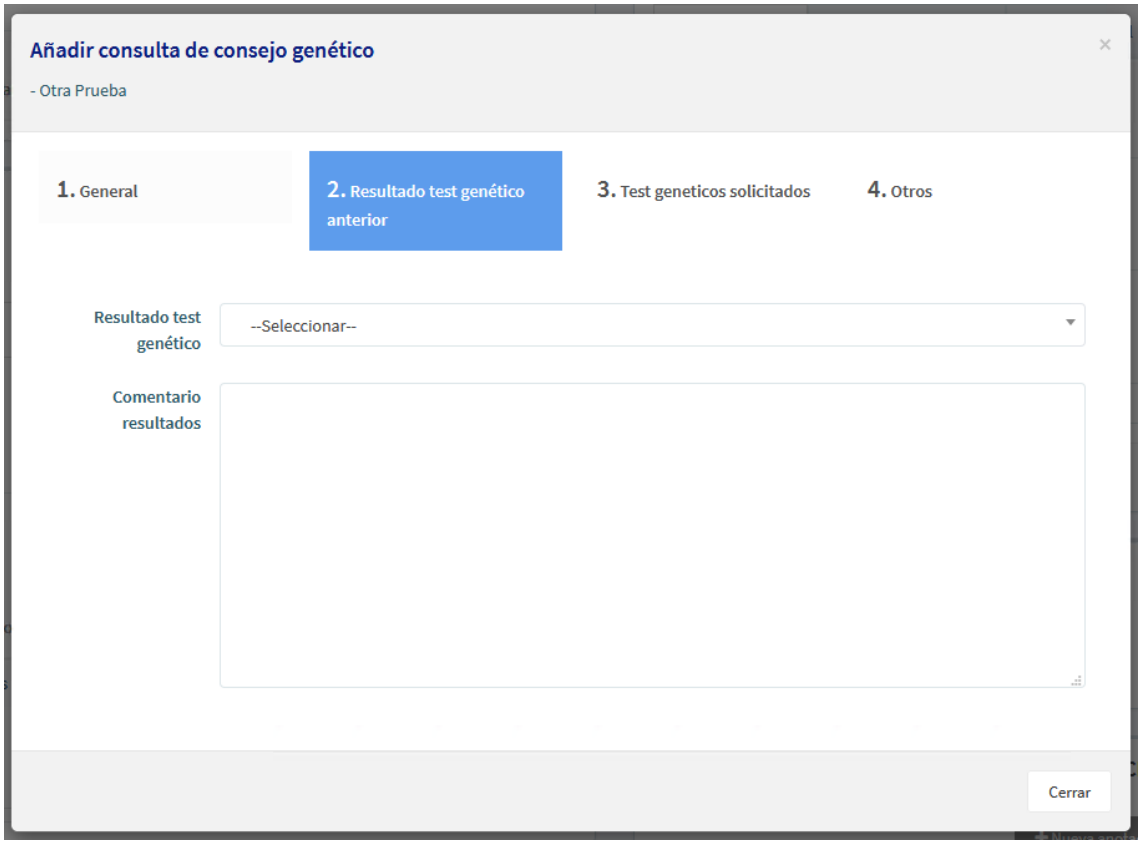

Ilustración 41: Añadir consulta de consejo genético (parte 2).

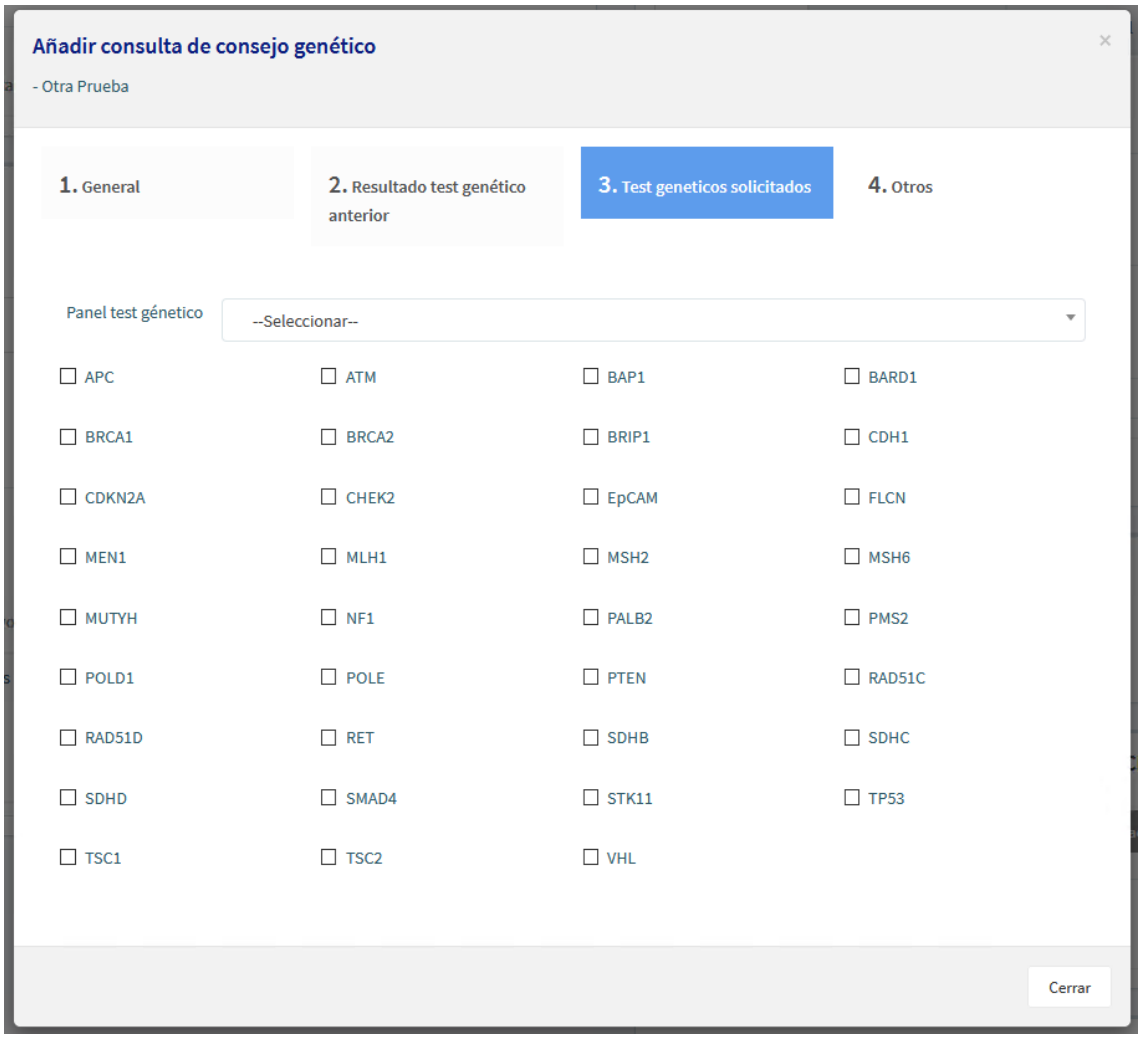

Ilustración 42: Añadir consulta de consejo genético (parte 3).

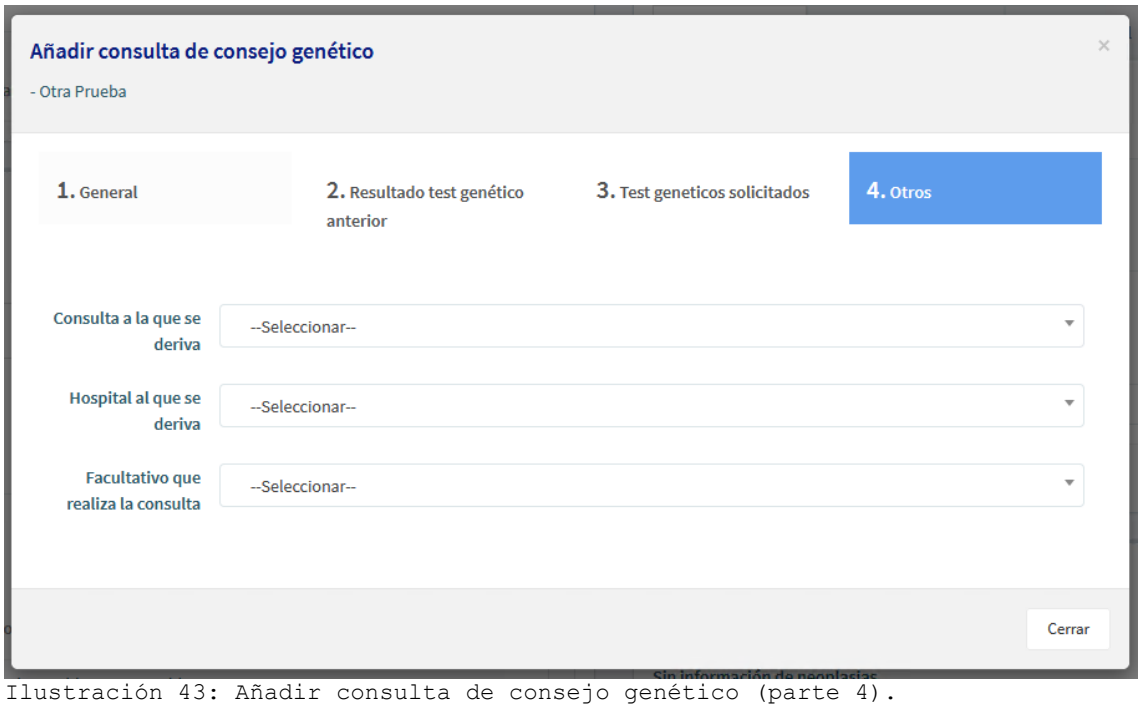

Ilustración 43: Añadir consulta de consejo genético (parte 4).

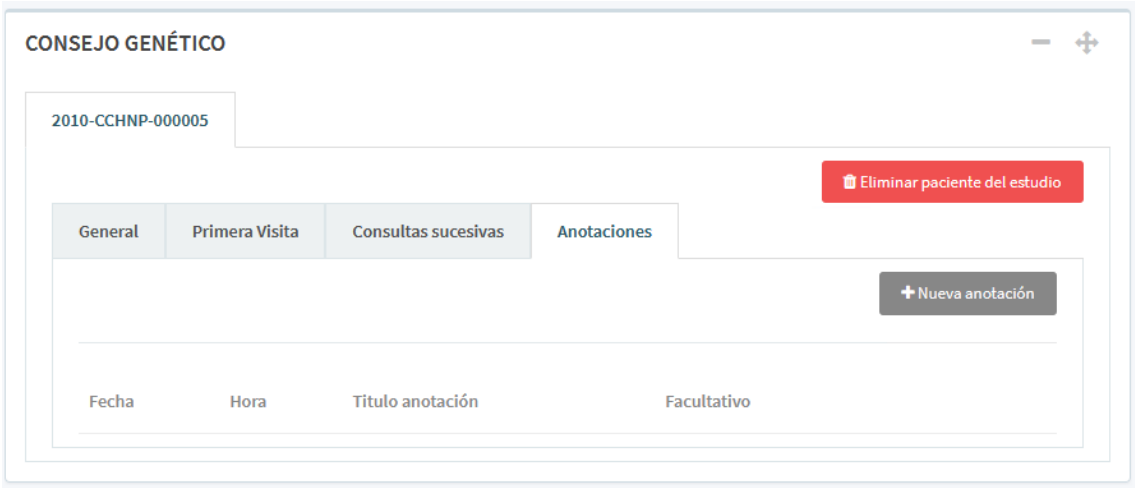

Ilustración 44: Vista del módulo de consejo genético con una familia asociada (Anotaciones).

Ilustración 45: Añadir anotación de consejo genético.

3.4 Problemas encontrados durante el desarrollo

3.4.1 Problema a la hora de indexar las tablas para la extensión de JQuery llamada DataTable5.

En un principio se intentó realizar como se indica en el manual que aparece en la bibliografía (GALLOWAY J., WILSON B., SCOTT ALLEN K., MATSON D., Professional ASP.NET MVC 5, Ed. Wrox, 2012), a través de nuestro modelo (*ModeloGestionFamiliasCG*) que posee una función estática llamada *QFamiliasCG* devolvería la lista de familias actuales y no borradas, la cual mediante la herramienta Razor se genera todo el HTML con los datos para la tabla para después con JavaScript indexar todos esos datos e instanciar dicha tabla con un DataTable, proporcionando todas las funcionalidades que nos solicitaban en los requisitos sobre las tablas (búsqueda, ordenación y paginación). Pero resultaba que debido a la cantidad de datos (en torno a 10.000 registros) la carga de la página se ralentizaba demasiado.

```
@using Galen.Models
@model ModeloGestionFamiliasCG
<table id="datatableC" class="table table-striped table-hover">
    <thead>
        <tr>
            <th scope="col">Nº familia</th>
            <th scope="col" data-orderSequence="desc">Año inclusión</th>
            <th scope="col">Criterio</th>
            <th scope="col">Iniciales</th>
            <th class="sort-alpha sorting" tabindex="0" aria-controls="datatable2" rowspan="1" colspan="1"></th>
            <th class="sort-alpha sorting" tabindex="0" aria-controls="datatable2" rowspan="1" colspan="1"></th>
            <th class="sort-alpha sorting" tabindex="0" aria-controls="datatable2" rowspan="1" colspan="1"></th>
            <th class="sort-alpha sorting" tabindex="0" aria-controls="datatable2" rowspan="1" colspan="1"></th>
            @if (puedeEscribir)
            {
                <th class="sort-alpha sorting" tabindex="0" aria-controls="datatable2" rowspan="1" colspan="1"></th>
            }
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model.QFamiliaCG)
        {
            string fecha = item.FechaInclusion.Year.ToString();
            if (item != null)
            {
        <tr id="@item.IdFamiliaCG" class="gradeA odd tr-familiasCG">
            <td class="centro"><span class="text-uppercase label bg-familiasCG">@item.cod</span></td>
            <td class="centro">@fecha</td>
            <td class="izqda">@item.CriterioSeleccionRiesgoCG.NombreCriterio</td>
            <td class="izqda">@item.CriterioSeleccionRiesgoCG.Iniciales</td>
            <td><a class="enlace-familiasCG" href="@Url.Action("Edit", "GestionFamiliasCG", new { id = item.IdFamiliaCG })"
                data-toggle="tooltip" title="" data-original-title="editar"><i class="fa fa-edit"></i></a></td>
            <td><a href="@Url.Action("GetPdfReportFamiliaCG", "Informes", new { idFamiliaCG = item.IdFamiliaCG })"
                data-toggle="tooltip" class="text-pdf" data-original-title="descargar pdf"><i class="fa fa-file-pdf-o"></i></a></td>
            <td><a href="@Url.Action("GetWordReportFamiliaCG", "Informes", new { idFamiliaCG = item.IdFamiliaCG })"
                data-toggle="tooltip" data-original-title="descargar word"><i class="fa fa-file-word-o"></i></a></td>
            @if (puedeEscribir)
            {
                <td>
                    @using (Html.BeginForm("Delete", "GestionFamiliasCG", FormMethod.Post, htmlAttributes: new { @class = "borrar-elemento" }))
                    {
                        @Html.AntiForgeryToken()
                        <input type="hidden" name="idFamiliaCG" value="@item.IdFamiliaCG" />
                        <button type="submit" class="btn btn-link" data-toggle="tooltip" title="" data-original-title="borrar">
                            <span class="text-danger"><i class="fa fa-trash"></i></span>
                        </button>
                    }
                </td>
            }
            <td><i class="fa fa-eye" data-toggle="tooltip" title="" data-original-title="ver" onclick="seleccionaFamiliaCG('@item.IdFamiliaCG')"></i></td>
        </tr>
            }
        }
    </tbody>
```

Ilustración 46: Vista antes de la modificación

[5] Datatable es una extensión de jQuery que nos permite pintar tablas con paginado, búsqueda, ordenar por columnas, etc.

```
    $('#datatableC').dataTable({
    'paging': true,  // Table pagination
    'ordering': true,  // Column ordering
    'order': [[ 1, 'desc' ], [2, 'asc'], [3, 'asc']],
    'info': true,  // Bottom left status text
    // Text translation options
    // Note the required keywords between underscores (e.g _MENU_)
    language: { url: '../Scripts/datatables-consultas.json' }
});
```

Ilustración 47: Script para instanciar Datatable antes de la modificación

Debido a dicho retardo se tuvo que modificar la manera en la que se indexaban toda esa cantidad de datos, por lo que para agilizar la carga se optó por en vez de mediante Razor generar el HTML de la tabla y después instanciar la tabla mediante JavaScript como un DataTable, lo que se hizo fue dejar la tabla que aparece en la vista solo con las cabeceras y cuando se instanciase la tabla como un DataTable generar los datos por consultas JSON a través del controlador correspondiente (sección: 3.3.2.2) y modificar el JavaScript para pasarle los datos a la hora de instanciar la tabla, mostrado en la ilustración 22.

```
// #region CARGA DE TABLAS
var tablaFamilias = $('#datatableFamilias').DataTable({
language: { url: '../Scripts/datatables-consultas.json' },
"ajax": {
    "url": "/GestionFamiliasCG/json_list_Familias",
    "type": "GET",
    "datatype": "json",
    "select": "single"
},
"columns": [
    { "data": "Numerofamilia", "autoWidth": true },
    { "data": "AnoInclusión", "autoWidth": true },
    { "data": "Criterio", "autoWidth": true },
    { "data": "Iniciales", "autoWidth": true },
    {
        "data": "ArbolGNO",
        "orderable": false,
        "autoWidth": true,
        "render": function (data, type, row, meta) {
            if (type === 'display' && data != null) {
                data = '<a href=GestionFamiliasCG/VerArchivoGNO?idArbolGenealogicoGNO='
                + data + ' data-toggle="tooltip" data-original-title="editar" ><i class="fa fa-file-image-o"></i></a>'
            }
            return data;
        }
    },
    {
        "data": "ArbolPDF",
        "orderable": false,
        "autoWidth": true,
        "render": function (data, type, row, meta) {
            if (type === 'display' && data != null) {
                data = '<a class="text-pdf" href=GestionFamiliasCG/VerArchivoPDF?idArbolGenealogicoPDF='
                + data + ' data-toggle="tooltip" data-original-title="editar" ><i class="fa fa-file-pdf-o"></i></a>'
            }
            return data;
        }
    },
    {
        "data": "Edit",
        "orderable": false,
        "autoWidth": true,
        "render": function (data, type, row, meta) {
            if (type === 'display') {
                data = '<a class="enlace-familiasCG" href=GestionFamiliasCG/Edit?id='
                + data + ' data-toggle="tooltip" title data-original-title="editar"><i class="fa fa-edit"></i></a>'
            }

            return data;
        }
    },
]
});
```

Ilustración 48: Script para instanciar Datatable después de la modificación

3.4.2 Problema de usabilidad para los SelectBox.

En una de las reuniones mantenidas con el hospital se nos notificó que los SelectBox con demasiados datos eran poco usables debido a la dificultad de los oncólogos para encontrar un dato concreto (una enfermera, un doctor, un criterio concreto,…). Dicho problema se solucionó usando una extensión de JQuery llamada Select2[6], a continuación se mostraran las diferencias.

[6] Select2 ofrece un cuadro de selección personalizable con soporte para búsqueda, etiquetado, conjuntos de datos remotos, desplazamiento infinito y muchas otras opciones.

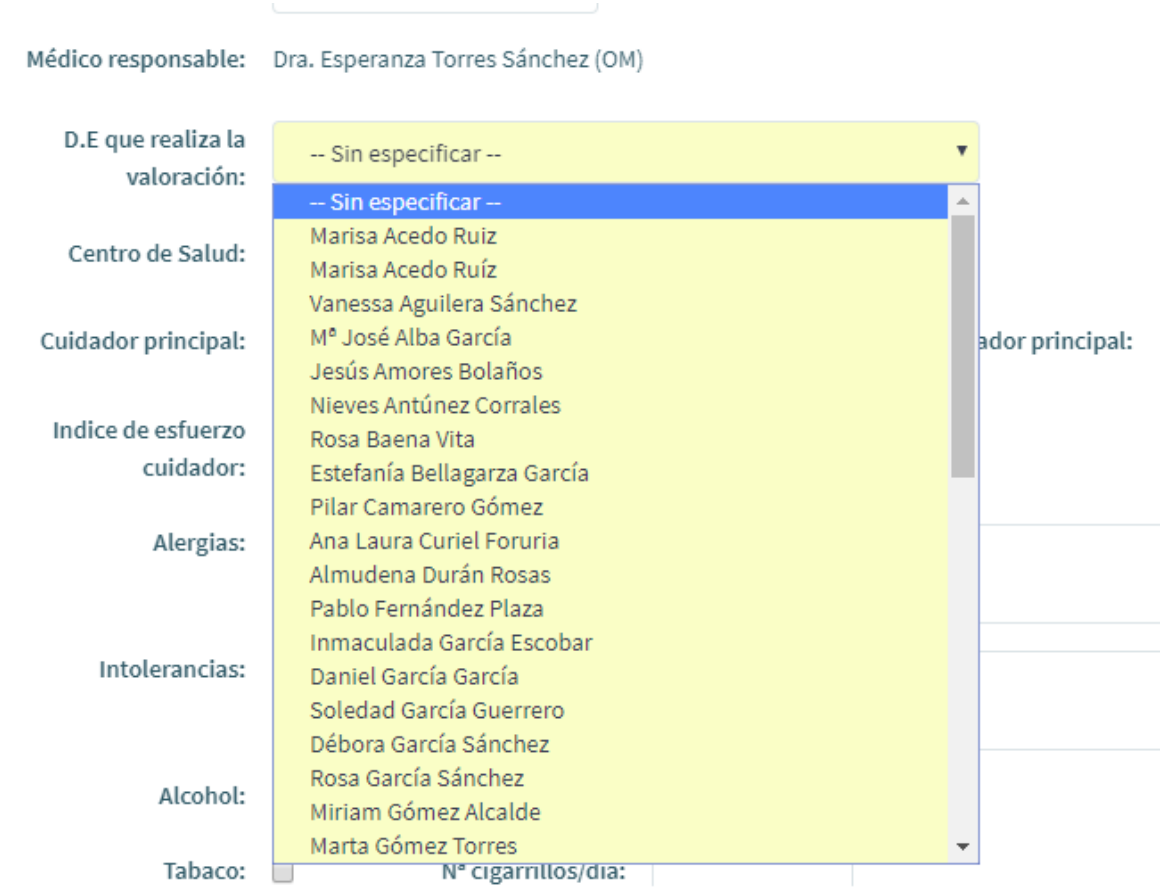

Ilustración 49: SelectBox sin usar la extensión Select2 de JQuery

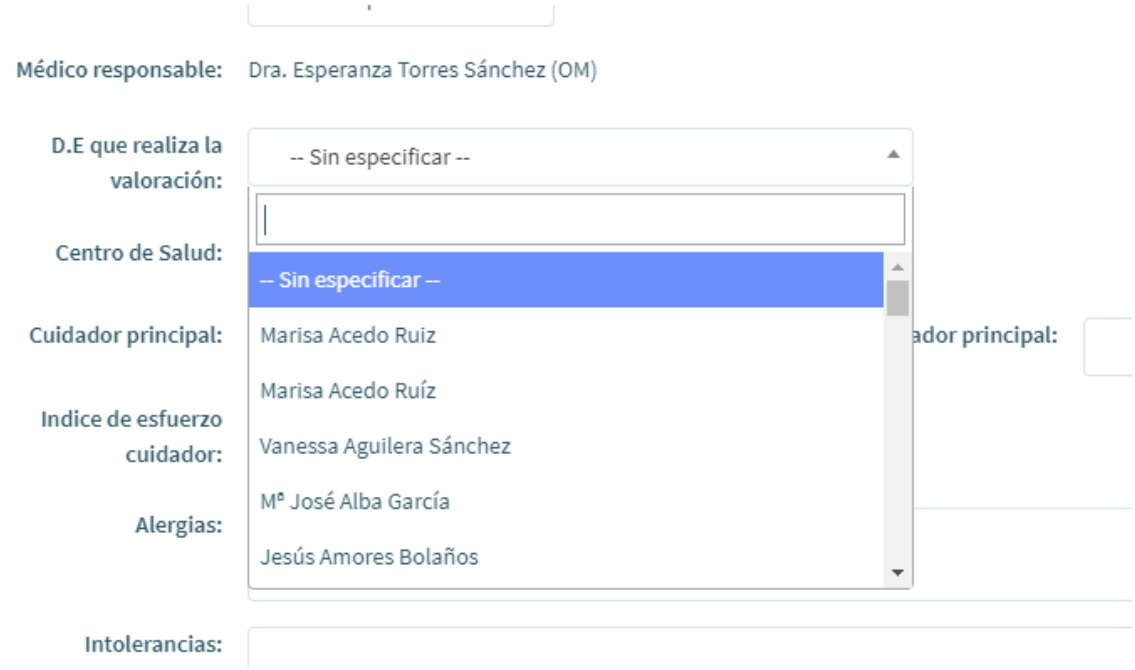

Ilustración 50: SelectBox usando la extensión Select2 de JQuery

Como se puede observar mejora bastante la usabilidad para los oncólogos permitiéndoles seleccionar rápidamente una determinada opción, agilizando así la cumplimentación del informe.

4

Conclusiones

Galén ha sido desde su creación un compromiso entre la exhaustividad, en la cual la información relevante para todo el servicio de oncología ha sido almacenada, y la simplicidad, donde diseñar una estructura que por su complejidad inherente, haga difícil la recogida de los datos a través de sus usuarios debido a una compleja usabilidad.

A pesar de ello, el grupo de investigación ICB (Inteligencia Computacional en Biomedicina) ha seguido trabajando en mejorar lo obtenido en una nueva versión 2.0 que además de tener una nueva interfaz de usuario responsiva basada en la plantilla 'Bootstrap', la base de datos se ha cambiado sustancialmente para poder afrontar los requisitos de seguridad en los cuales se pudiera auditar toda la información pasada de cualquier registro o dato del sistema, así como su recuperación inmediata.

Otra de las fortalezas de un sistema de información como Galén son las oportunidades de desarrollo que ofrece. Es posible establecer conexiones con otros Servicios Centrales del Hospital para acceder a los resultados de las diferentes pruebas diagnósticas realizadas a los pacientes e incorporarlos al formato de las Consultas. También permite integrar cualquier otra funcionalidad que nos planteemos desarrollar para cubrir las necesidades de otros aspectos asistenciales del Servicio de Oncología, como el Área de Hospitalización, la Consulta de Enfermería, la Programación de trabajo diario en Hospital de Día, consejo genético, ensayos clínicos, peticiones al laboratorio o el Control de los pacientes que acuden diariamente al Área de Consultas, etc.

En lo que se refiere a la explotación estadística de la información, se están ya incorporando las herramientas necesarias para realizar análisis multivariado y

predicciones sobre probabilidades de recidiva o supervivencia de los pacientes utilizando modelos predictivos basados en técnicas de Inteligencia Artificial y desarrollados a partir de los datos de nuestros propios pacientes.

En este trabajo fin de grado hemos desarrollado los módulos de enfermería y consejo genético en la versión de Galén 2.0. Si bien estos módulos ya existían en la versión anterior, han sufrido muchos cambios en su implementación y diseño. En concreto, el módulo de consejo genético ha sido completamente rediseñado ya que la versión 1.0 no se adaptaba a los requerimientos necesarios y no se estaba usando debidamente en la anterior versión.

Estas modificaciones han sido críticas ya que una vez implementado cada uno de los módulos, se han tenido que migrar los datos de la versión anterior a la nueva versión sin existir una coincidencia real de los registros ni de las tablas nuevas.

Los dos módulos, tras su despliegue, están siendo ya usados por los servicios de oncología de los hospitales de la provincia de Málaga y su uso está modificando sustancialmente la eficiencia de estos servicios.

A nivel personal, gracias a este proyecto he podido aplicar los conocimientos adquiridos durante el grado de *Ingeniera del Software* durante todo el desarrollo de una aplicación real y ya en funcionamiento, además de aprender nuevas tecnologías como C#.NET, HTML, XML, JSON, JavaScript, JQuery, SQL server, triggers, vistas, etc.

4.1 Futuras líneas de trabajo

El proyecto Galén está en continua expansión debido a su buen funcionamiento desde que se inició en 2006. Tal es su crecimiento que ya han existido diferentes reuniones en las cuales se habla de una expansión a nivel autonómico. Para ello se siguen implementando en estos momentos más módulos que están siendo solicitados por otros aspectos asistenciales de los servicios de oncología, así como trabajos más ambiciosos en los cuales a través de técnicas de minería de datos e inteligencia computacional se quiere extraer información oculta en los campos abiertos de los informes generados por los usuarios de la aplicación. Actualmente, existen varias líneas de trabajo como son:

- Módulo del laboratorio de biología molecular del cáncer encargado de gestionar las peticiones al laboratorio de manera rápida y automática.
- Módulo de ensayos clínicos.
- Modulo estadístico.
- Extracción de información en campos abiertos para la imputación de datos perdidos, predicción de asistencia en urgencias, predicción de recidiva de alguna neoplasia.

5

Bibliografía

1. GALLOWAY J., WILSON B., SCOTT ALLEN K., MATSON D., Professional ASP.NET MVC 5, Ed. Wrox, 2012
2. Ribelles N, Jerez JM, Urda D, Subirats JL, Márquez A, Quero C. Galén: Sistema de Información para la gestión y coordinación de procesos en un servicio de Oncología. FeSalud [Internet] 2010 Disponible en: http://www.revistaesalud.com/index.php/revistaesalud/article/view/371/723
3. https://es.wikipedia.org/wiki/ASP.NET MVC Framework
4. https://es.wikipedia.org/wiki/ASP.NET
5. https://book.cakephp.org/2.0/es/cakephp-overview/understanding-model-view-controller.html
6. https://devcode.la/blog/que-es-bootstrap/
7. https://www.hostinger.es/tutoriales/que-es-jquery/
8. https://desarrolloweb.com/articulos/la-sintaxis-razor.html
9. https://searchdatacenter.techtarget.com/es/definicion/SQL-Server
10. https://devcode.la/blog/que-es-sql/
11. https://www.genbeta.com/desarrollo/sourcetree-cliente-gui-para-manejar-repositorios-git-o-mercurial-llegara-en-breve-a-windows