ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA DE LA SALUD

Sistema de Información para la Unidad de Podología de la UMA.

Information System for the Podiatry Unit of UMA.

Realizado por:
Maria José Muñoz González.

Tutorizado por:
Jose Manuel Jerez Aragonés.

Co-tutorizado por:
Julio Montes Torres.

Departamento
Lenguaje y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, Junio de 2018

Resumen

La recolección de datos de la Unidad Asistencial de Podología (UAP) de la Universidad de Málaga aún se hace a mano. Esto es un trabajo pesado y que incluye un gasto de tiempo, el cual podría emplearse en exploración del paciente. La aplicación web del proyecto "PodoApp" intenta sustituir esa manera actual de trabajo para poder tener una clínica digitalizada. De manera, que se recogen los datos y se almacenan directamente en una base de datos sin necesidad de archivar los historiales clínicos de los pacientes de manera física. PodoApp está implementado en ASP.NET y es parte de un gran proyecto. Este proyecto consistirá en dotar a la página web de total funcionalidad.
Aparte de la digitalización, lo que se busca es poder integrar a la podología en el campo de la investigación mediante técnicas de minería de datos, las cuales permitirán realizar estudios sobre ciertas variables de interés para los podólogos.

Palabras claves

UAP, PodoApp, ASP.NET, Digitalización, Minería de datos.

Abstract

The collection of data from the Podiatry Care Unit (UAP) of the University of Malaga is still done by hand. This is laborious work and involves spending a lot of time on the task, which could be focused on patient exploration. The web application of the "PodoApp" project tries to replace the current way of working in order to create a digitalized clinic. In this way, the data would be collected and stored directly in a database without the need to archive the clinical records of patients in a physical format. PodoApp is implemented in ASP.NET and is a part of a larger project. This project will provide the website with full functionality. Apart from digitalization, it is intended to integrate podiatry in the field of research using data mining techniques, which will allow studies of certain variables of interest to podiatrists.

Keywords

UAP, PodoApp, ASP.NET, Digitalization, Data mining.

Dedicado a mi familia y a todas las personas que me han ayudado y acompañado a lo largo de estos cuatro años. Solo puedo daros las gracias y deciros que de todos he aprendido algo.

Índice

1 Introducción

1.1 Motivación

Este proyecto surge ante la petición de la facultad de Ciencias de la Salud de la Universidad de Málaga de crear una Aplicación Web que les permitiese, entre otras cosas, la gestión de sus pacientes.

Cuando acudimos a la Unidad de Podología de la Universidad de Málaga observamos que los profesionales o estudiantes en prácticas recojen los datos de los pacientes a mano. Éstos rellenan una serie de informes a medida que se va desarrollando la consulta. Posteriormente, los informes son guardados en archivadores hasta la próxima consulta del paciente.

Lo que se pretende es sustituir todo el proceso manual de recolección de datos, rellenando así un formulario a través de Internet y pudiendo almacenar las respuestas. Por otro lado, cuando el paciente acudiese nuevamente a la consulta, bastaría con buscar los datos correspondientes para así poder acceder a todos los informes disponibles del mismo.

1.2 Estado de la cuestión

Encontramos diversas empresas que se encargan de producir software que está destinado a la gestión de clínicas, consultas y hospitales:

- Aquar Software [1]. Se trata de una aplicación de escritorio que permite el acceso desde distintos sistemas operativos (iOS, Windows). Está orientada a la gestión de datos de los pacientes mediante citas, agendas e historial clínico.
- DASI Clinic [2]. Se trata de una aplicación de escritorio dividida en tres grandes partes. Por un lado, la gestión de la agenda. Por otro lado, la gestión de pacientes y datos clínicos. Y, por último, gestión económica y administración. A diferencia de la anterior, ésta si permite la obtención de estadísticas a partir de los cuestionarios que rellenan los pacientes.
- Apclinic [3]. Se trata de una aplicación web muy orientada a la gestión de citas/agenda, la gestión de pacientes y la gestión administrativa. A diferencia de las anteriores ofrece la posibilidad de gestionar mutuas, pero nada acerca de obtención de estadísticas.
- NetClinicas [4]. Se trata de una aplicación de escritorio que, al igual que las mencionadas anteriormente, trata la gestión de pacientes, agenda y administración. Incluye la obtención de estadísticas pero se trata de un estudio básico.
- Qclinicas [5]. Se trata de una aplicación de escritorio que está centrada en la gestión de citas y en la gestión de pacientes.

La mayoría de estas aplicaciones son software genérico que se adapta a las posteriores peticiones de los clientes, en este caso, podólogos. Todas ellas tienen en común que están destinadas a la gestión de los datos de los pacientes, pero no están destinadas a la explotación de dichos datos. Es cierto que algunas presentan la opción de realizar estadísticas, pero se trata de estadísticas orientadas a los gastos de la consulta o de estadísticas básicas respecto a los pacientes.

Con la creación de nuestra aplicación se pretende poder realizar investigación. De modo que tendría doble utilidad. Por un lado, gestionar los pacientes y, por

otro lado, poder obtener estadísticas, cuyos datos posteriormente serán explotados, permitiendo así realizar investigación en el campo de la podología.

Cuando buscamos publicaciones acerca de estudios podológicos, encontramos poca información. La mayor parte de estos estudios están centrados en la diabetes y en los efectos que tiene en ciertas áreas de estudio de la podología. Como por ejemplo:

- *Identificación de nuevas asociaciones con el pie de Charcot en el contexto de modelo de patogénesis* [6].
- *Reconocimiento de la importancia de la atención podológica en la población diabética para evitar amputaciones* [7].

Por tanto, ¿qué es lo que realmente pretendemos con este proyecto? La respuesta es sencilla, pretendemos empezar a construir una aplicación web (*PodoApp*) que desde el principio atienda a las necesidades de los podólogos y les permita trabajar de manera óptima y práctica. De esta manera, en un futuro se podrán obtener estadísticas significativas y mediante la explotación de esos datos se podría llegar a conclusiones interesantes acerca del estudio que se realice.
La mayoría de las aplicaciones antes descritas se usan tanto como para una clínica de podología, como para una clínica dental. Simplemente el software se adapta. En cambio, *PodoApp* desde el principio está orientada a podólogos y se crea en base a los cuestionarios que habitualmente se realizan en la *Unidad Asistencial de Podología de la Universidad de Málaga.*

1.3 Metodología

A lo largo del proyecto se ha seguido una metodología de desarrollo ágil *Kanban* en un marco de desarrollo guiado por proyectos. Este método es una aproximación al proceso gradual, evolutivo y al cambio de sistemas para las organizaciones. Utiliza un sistema de extracción limitada del trabajo en curso como mecanismo básico para exponer los problemas de funcionamiento del sistema (o proceso) y estimular la colaboración para la mejora continua del sistema. Los principios son:

- Visualizar lo que se raliza: una visualización de todas las tareas realizadas, pendientes y en ejecución en una tabla contribuirá a estar al corriente del trabajo en un sólo vistazo. Para ello se ha usado *Trello* [8]. Con esto nos referimos a:

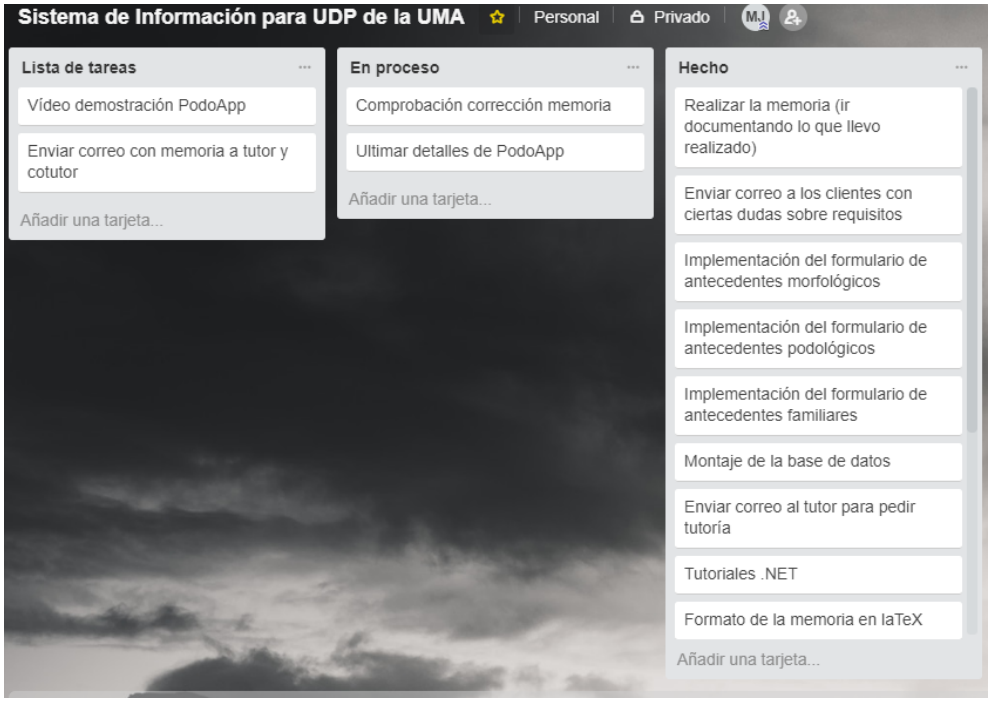

Figura 1. Tablero de Trello utilizado durante el proyecto.

- Limitar la cantidad de trabajo mediante el establecimiento de metas asequibles.
- Realizar un seguimiento del tiempo.
- Identificar los cuellos de botella y eliminar lo que resulta descartable.

Esa metodología ha sido aplicada a nuestro proyecto de la siguiente forma:

- Balance semanal para comprobar el estado del proyecto.
- Metas realistas para llevar a cabo el desarrollo del proyecto.
- Reuniones con los clientes para evaluar el proceso de desarrollo de la aplicación.

1.4 Objetivos

El principal objetivo de este proyecto es la sustitución de la toma de datos a papel, por la realización de un cuestionario a través de la web. Es decir, la *digitalización de la Unidad Asistencial de Podología de la Universidad de Málaga.*
Este objetivo principal conlleva otros como son:

- Disminución del tiempo que los profesionales dedican a recopilar datos médicos de pacientes durante las consultas, agilizando así el tiempo de espera de los pacientes.
- Agilización de los trámites o consultas del profesional, ya que toda la información se encuentra recogida en un único sistema de información.
- Obtención de estadísticas mediante estudios de minería de datos. Esto permitirá la visualización de gráficas, entre otras cosas, como por ejemplo: gráficas sobre el rendimiento de ciertos medicamentos en los pacientes.

1.5 Estructura de la memoria

La memoria se va a estructurar en tres partes. La primera se va a basar en el *Análisis* de las tecnologías y los requisitos. La segunda se va a basar en el *Diseño e Implementación* de la base de datos y de la aplicación web. Y la tercera, y última, se va a basar en las *Pruebas* realizadas durante la implementación.

Esta memoria va a ir acompañada de un manual de usuario para facilitar así el manejo de la aplicación en caso de duda de los profesionales y/o estudiantes.

2 Análisis

Se trata de una aplicación web privada a la que sólo los miembros pertenecientes a la Unidad de Podología de la UMA van a tener acceso. Por tanto, se contemplan tres roles: Administrador, Podólogo y Estudiante en prácticas. Ante la ausencia de especificación del cliente de la diferencia entre los tres perfiles, los tres roles van a ser implementados como administrador para así en un futuro poder hacer los cambios pertinentes.
Por lo que todos los requisitos que se van a describir a continuación serían del rol de administrador.

2.1 Requisitos

Requisitos Funcionales

RF1. El usuario puede iniciar sesión.

RF2. El usuario puede cerrar sesión.

RF3. El usuario puede crear un nuevo paciente.

- RF3.1. El usuario puede añadir los datos personales del paciente.
- RF3.2. El usuario puede editar los datos personales del paciente.
- RF3.3. El usuario puede añadir los datos clínicos del paciente.
- RF3.4. El usuario puede editar los datos clínicos del paciente.

RF4. El usuario puede añadir los antecedentes del paciente.

RF5. El usuario puede editar los antecedentes del paciente.

RF6. El usuario puede editar la primera visita de un paciente.

RF7. El usuario puede añadir el diagnóstico de un paciente.

RF8. El usuario puede editar el diagnóstico de un paciente.

RF9. El usuario puede añadir el tratamiento de un paciente.

RF10. El usuario puede editar el tratamiento de un paciente.

RF11. El usuario puede crear una nueva consulta de un paciente.

RF12. El usuario puede editar una consulta de un paciente.

Requisitos No Funcionales

RNF1. El sistema deberá implementarse mediante el patrón Modelo-Vista-Controlador (MVC).

RNF2. El sistema se implementará en ASP.NET, usando también HTML, CSS y Bootstrap.

RNF3. El sistema sólo permitirá el acceso de los usuarios correctamente identificados.

RNF4. Las contraseñas de los usuarios serán encriptadas.

2.2 Casos de uso

Una vez expuestos los requisitos pasamos a describir los casos de uso [1]:

Caso de uso	
Título	RF1. El usuario puede iniciar sesión.
Descripción	El usuario de la aplicación web perteneciente a la Unidad Asistencial de Podología de la UMA puede iniciar sesión en la aplicación.
Pre-condición	
Post-condición	
Prioridad	Alta.
Autor	Maria José Muñoz González.
Control de cambios	
Escenario principal	1. El usuario accede al link de la página web. 2. El sistema muestra la página de inicio de sesión. 3. El usuario rellena los datos requeridos (nombre y contraseña). 4. El sistema valida los datos. 5. El usuario accede a la aplicación web.
Escenario alternativo	1. El usuario accede al link de la página web. 2. El sistema muestra la página de inicio de sesión. 3. El usuario rellena los datos requeridos (nombre y contraseña). 4. El sistema no valida los datos. 5. El sistema indica los datos que son incorrectos.
Clases de análisis	PagesController, EntrarModel y Login.

Caso de uso	
Título	RF2. El usuario puede cerrar sesión.
Descripción	El usuario de la aplicación web perteneciente a la Unidad Asistencial de Podología de la UMA puede cerrar sesión en la aplicación tras haber iniciado sesión.
Pre-condición	
Post-condición	
Prioridad	Alta.
Autor	Maria José Muñoz González.
Control de cambios	
Escenario principal	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de cerrar sesión. 3. El sistema cierra sesión. 4. El sistema muestra la pantalla de inicio de sesión.
Escenario alternativo	
Clases de análisis	PagesController, EntrarModel y Login.

[1] En todos los casos de uso el contenido de *Clases de análisis* está dispuesta de la siguiente forma: Controlador, Modelo y Vista.

Caso de uso	
Título	RF3.1. El usuario puede añadir los datos personales de un nuevo paciente. RF3.3. El usuario puede añadir los datos personales de un nuevo paciente.
Descripción	El usuario una vez que ha iniciado sesión puede crear un paciente nuevo y añadir los datos personales y clínicos.
Pre-condición	
Post-condición	
Prioridad	Alta.
Autor	Maria José Muñoz González.
Control de cambios	
Escenario principal	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. El usuario pulsa sobre el botón de *Crear paciente*. 4. El usuario rellena los campos de datos. 5. El usuario pulsa el botón de *Añadir paciente*. 6. El sistema valida los datos. 7. El paciente ha sido creado.
Escenario alternativo	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. El usuario pulsa sobre el botón de *Crear paciente*. 4. El usuario rellena los campos necesarios. 5. El usuario pulsa el botón de *Añadir paciente*. 6. El sistema valida los datos. 7. El sistema muestra los datos que no poseen el formato correcto o los datos que son obligatorios de añadir. 8. El usuario corrige los campos erróneos. 9. El usuario pulsa el botón de *Añadir paciente*. 10. El sistema valida los datos. 11. El paciente ha sido creado.
Clases de análisis	PacienteController, FormPaciente y Create.

Caso de uso	
Título	RF3.2. El usuario puede editar los datos personales de un paciente. RF3.4. El usuario puede editar los datos clínicos de un paciente.
Descripción	El usuario una vez que ha iniciado sesión puede editar los datos de un paciente.
Pre-condición	
Post-condición	
Prioridad	Alta.
Autor	Maria José Muñoz González.
Control de cambios	
Escenario principal	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. El usuario pulsa sobre el botón de *Editar datos personales*. 4. El usuario modifica los campos necesarios. 5. El usuario pulsa el botón de *Modificar*. 6. El sistema valida los datos. 7. Los datos del paciente han sido editados.
Escenario alternativo	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. El usuario pulsa sobre el botón de *Editar datos personales*. 4. El usuario modifica los campos necesarios. 5. El usuario pulsa el botón de *Modificar*. 6. El sistema valida los datos. 7. El sistema muestra los datos que no poseen el formato correcto o los datos que son obligatorios de añadir. 8. El usuario corrige los campos erróneos. 9. El usuario pulsa el botón de *Modificar*. 10. El sistema valida los datos. 11. Los datos del paciente han sido editados.
Clases de análisis	PacienteController, FormPaciente y Edit.

Caso de uso	
Título	RF4. El usuario puede añadir los antecedentes de un nuevo paciente.
Descripción	El usuario una vez que ha iniciado sesión puede añadir los antecedentes de un paciente si estos no estaban previamente añadidos.
Pre-condición	
Post-condición	
Prioridad	Alta.
Autor	Maria José Muñoz González.
Control de cambios	
Escenario principal	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. El usuario pulsa sobre el botón de *Añadir antecedentes*. 4. El usuario rellena los campos necesarios. 5. El usuario pulsa el botón de *Añadir*. 6. El sistema valida los datos. 7. Los antecedentes de dicho paciente han sido creados.
Escenario alternativo	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. El usuario pulsa sobre el botón de *Añadir antecedentes*. 4. El usuario rellena los campos necesarios. 5. El usuario pulsa el botón de *Añadir*. 6. El sistema valida los datos. 7. El sistema muestra los datos que no poseen el formato correcto o los datos que son obligatorios de añadir. 8. El usuario corrige los campos erróneos. 9. El usuario pulsa el botón de *Añadir*. 10. El sistema valida los datos. 11. Los antecedentes de dicho paciente han sido creados.
Clases de análisis	AntecedentesController, FormAntecedentes y Create.

Caso de uso	
Título	RF5. El usuario puede editar los antecedentes de un paciente.
Descripción	El usuario una vez que ha iniciado sesión puede editar los antecedentes de un paciente si este ya tenía antecedentes añadidos.
Pre-condición	
Post-condición	
Prioridad	Alta.
Autor	Maria José Muñoz González.
Control de cambios	
Escenario principal	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. El usuario pulsa sobre el botón de *Editar Antecedentes*. 4. El usuario modifica los campos necesarios. 5. El usuario pulsa el botón de *Modificar*. 6. El sistema valida los datos. 7. Los antecedentes han sido modificados.
Escenario alternativo	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. El usuario pulsa sobre el botón de *Editar antecedentes*. 4. El usuario modifica los campos necesarios. 5. El usuario pulsa el botón de *Modificar*. 6. El sistema valida los datos. 7. El sistema muestra los datos que no poseen el formato correcto o los datos que son obligatorios de añadir. 8. El usuario corrige los campos erróneos. 9. El usuario pulsa el botón de *Modificar*. 10. El sistema valida los datos. 11. Los antecedentes han sido modificados.
Clases de análisis	AntecedentesController, FormAntecedentes y Edit.

Caso de uso	
Título	RF6. El usuario puede editar la primera visita de un paciente.
Descripción	El usuario puede editar la primera visita de un paciente.
Pre-condición	
Post-condición	
Prioridad	Alta.
Autor	Maria José Muñoz González.
Control de cambios	
Escenario principal	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. El usuario pulsa sobre el botón de *Editar primera visita*. 4. El usuario modifica los campos necesarios. 5. El usuario pulsa el botón de *Modificar*. 6. El sistema valida los datos. 7. La primera visita ha sido creada.
Escenario alternativo	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. El usuario pulsa sobre el botón de *Editar primera visita*. 4. El usuario modifica los campos necesarios. 5. El usuario pulsa el botón de *Modificar*. 6. El sistema valida los datos. 7. El sistema muestra los datos que no poseen el formato correcto o los datos que son obligatorios de añadir. 8. El usuario corrige los campos erróneos. 9. El usuario pulsa el botón de *Modificar*. 10. El sistema valida los datos. 11. La primera visita ha sido creada.
Clases de análisis	PrimeraVisitaController, FormPrimeraVisita y Edit.

Caso de uso	
Título	RF7. El usuario puede añadir el diágnostico de un paciente. RF9. El usuario puede añadir el tratamiento de un paciente.
Descripción	El sistema ofrece la posibilidad de añadir un diagnóstico y un tratamiento al paciente que se le esté realizando la consulta.
Pre-condición	
Post-condición	
Prioridad	Alta.
Autor	Maria José Muñoz González.
Control de cambios	
Escenario principal	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. En el paciente desado el usuario pulsa sobre el botón de *Añadir diagnóstico y tratamiento*. 4. El usuario rellena los campos necesarios. 5. El usuario pulsa sobre el botón *Añadir*. 6. El diagnóstico y el tratamiento son añadidos en la base de datos.
Escenario alternativo	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. En el paciente desado el usuario pulsa sobre el botón de *Añadir diagnóstico y tratamiento*. 4. El usuario rellena algunos campos. 5. El usuario pulsa sobre el botón *Añadir*. 6. El sistema muestra los campos erróneos. 7. El usuario corrige los campos. 8. El usuario pulsa sobre el botón *Añadir*. 9. El diagnóstico y el tratamiento son añadidos en la base de datos.
Clases de análisis	DiagnosticoTratamientoController, FormDiagnosticoTratamiento y Create.

Caso de uso	
Título	RF8. El usuario puede editar el diágnostico de un paciente. RF10. El usuario puede editar el tratamiento de un paciente.
Descripción	El sistema ofrece la posibilidad de editar el diagnóstico y el tratamiento del paciente al que se le esté realizando la consulta.
Pre-condición	
Post-condición	
Prioridad	Alta.
Autor	Maria José Muñoz González.
Control de cambios	
Escenario principal	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. En el paciente desado el usuario pulsa sobre el botón de *Editar diagnóstico y tratamiento*. 4. El usuario modifica los campos necesarios. 5. El usuario pulsa sobre el botón *Modificar*. 6. El diagnóstico y el tratamiento son añadidos en la base de datos.
Escenario alternativo	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. En el paciente desado el usuario pulsa sobre el botón de *Editar diagnóstico y tratamiento*. 4. El usuario modifica algunos campos. 5. El usuario pulsa sobre el botón *Modificar*. 6. El sistema muestra los campos erróneos. 7. El usuario corrige los campos. 8. El usuario pulsa sobre el botón *Modificar*. 9. El diagnóstico y el tratamiento son modificados en la base de datos.
Clases de análisis	DiagnosticoTratamientoController, FormDiagnosticoTratamiento y Edit.

Caso de uso	
Título	RF11. El usuario puede crear una nueva consulta de un paciente.
Descripción	El sistema ofrece la posibilidad de crear una nueva consulta para un paciente.
Pre-condición	
Post-condición	
Prioridad	Alta.
Autor	Maria José Muñoz González.
Control de cambios	
Escenario principal	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. En el paciente desado pulsa sobre el botón de *Añadir* consulta. 4. El usuario añade los campos necesarios. 5. El usuario pulsa sobre el botón *Añadir*. 6. La consulta se añade en la base de datos.
Escenario alternativo	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. En el paciente desado el usuario pulsa sobre el botón de *Añadir* consulta. 4. El usuario modifica algunos campos. 5. El usuario pulsa sobre el botón *Añadir*. 6. El sistema muestra los campos erróneos. 7. El usuario corrige los campos. 8. El usuario pulsa sobre el botón *Añadir*. 9. La consulta se añade en la base de datos.
Clases de análisis	ConsultaController, FormConsulta y Create.

Caso de uso	
Título	RF12. El usuario puede editar una consulta de un paciente.
Descripción	El sistema ofrece la posibilidad de editar una consulta de un paciente.
Pre-condición	
Post-condición	
Prioridad	Alta.
Autor	Maria José Muñoz González.
Control de cambios	
Escenario principal	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. En el paciente desado pulsa sobre el botón de *Consultas*. 4. El sistema muestra una lista con las consultas previas, en caso de que existan. 5. El usuario pulsa sobre el botón de *Editar* en la consulta que necesite. 6. El usuario modifica los campos necesarios. 7. El usuario pulsa sobre el botón *Modificar*. 8. La consulta se modifica en la base de datos.
Escenario alternativo	1. El usuario ha iniciado sesión satisfactoriamente en la página web. 2. El usuario pulsa sobre el botón de *Lista de pacientes*. 3. En el paciente desado pulsa sobre el botón de *Consultas*. 4. El sistema muestra una lista con las consultas previas, en caso de que existan. 5. El usuario pulsa sobre el botón de *Editar* en la consulta que necesite. 6. El usuario modifica algunos campos. 7. El usuario pulsa sobre el botón *Modificar*. 8. El sistema muestra los campos erróneos. 9. El usuario corrige los campos. 10. El usuario pulsa sobre el botón *Modificar*. 11. La consulta se modifica en la base de datos.
Clases de análisis	ConsultaController, FormConsulta y Edit.

3 Diseño e Implementación

En esta sección nos centraremos en varios puntos importantes. Abordaremos las diferentes tecnologías utilizadas, el diseño de la base de datos y el diseño de la aplicación web.

3.1 Tecnologías usadas

El entorno de desarrollo utilizado es Visual Studio 2017, ya que nos permite escribir código en C#, entre otros varios lenguajes que permite este IDE. Para el desarrollo, nos vamos a ayudar de ASP.NET y vamos a utilizar el patrón de Modelo-Vista-Controlador (MVC).

Como sistema gestor de base de datos utilizaremos SQL Server 2012. Además, aunque principalmente esta aplicación va a ser utilizada en tablets, se ha decidido que para realizar la interfaz nos vamos a apoyar en Bootstrap. De esta manera, si en algún momento es necesario utilizar un ordenador o un dispositivo móvil, se adaptaría sin ningún problema.

3.1.1 ASP.NET

ASP.NET [9] es un entorno para aplicaciones web, desarrollado y comercializado por Microsoft. Se usa para construir sitios web dinámicos, aplicaciones web y servicios web XML. ASP.NET es el sucesor de ASP (Active Server Pages). Además, está construido sobre el Common Language Runtime (CLR) que es un entorno de ejecución para los códigos de los programas que corren sobre la plataforma Microsoft .NET. Estar construido sobre CLR permite escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework (algunos de estos son: C#, Visual Basic .NET, F#, etc.)

ASP.NET [10] ofrece tres marcos de trabajo para crear aplicaciones web: páginas Web de ASP.NET, formularios Web Forms y ASP.NET MVC. En este caso, vamos a usar ASP.NET MVC.

3.1.2 MVC

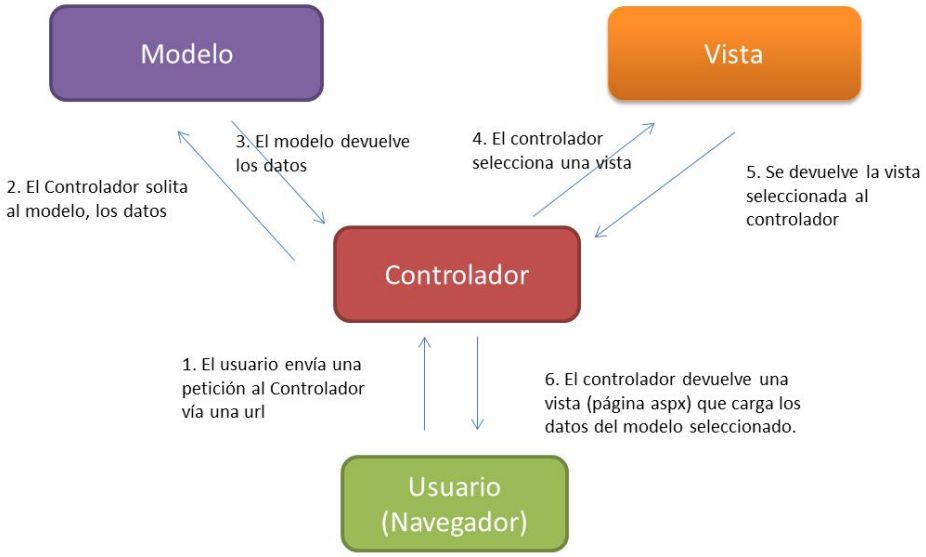

Figura 2. Esquema del patrón MVC.

Los componentes de MVC [11] se podrían definir tal que:

- Modelo. Contiene una representación de la información que maneja el sistema, su lógica de negocio y los mecanismos de persistencia. Es el que se encarga de acceder a la capa de almacenamiento de los datos y de definir la lógica de negocio (los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación). Por tanto, lo que hace es enviar a la Vista aquella parte de la información que en cada momento se le solicita para que sea mostrada al usuario.
- Vista, también llamada interfaz de usuario. Es la que presenta el Modelo, es decir, la información y la lógica de negocio en un formato adecuado para interactuar con el usuario. Por tanto, requiere de dicho Modelo la información que debe representar como salida.
- Controlador. Es el intermediario entre el Modelo y la Vista, de manera que responde a acciones del usuario (eventos en los que se hace alguna solicitud de información) mediante la invocación de peticiones al Modelo.

3.1.3 SQL Server

Se trata de un sistema de gestión de base de datos relacionales [12]. Esto significa que nos permite crear, actualizar y administrar una base de datos relacional. Siendo una base de datos relacional una colección de elementos de datos organizados en tablas.

3.2 Diseño

3.2.1 Diseño de la Base de datos

La creación de la base de datos fue el primer paso en el desarrollo de *PodoApp*. Se necesitan entidades para poder guardar los datos del paciente, que éste a su vez puede ser un alumno que esté en prácticas, un podólogo o una persona cualquiera.

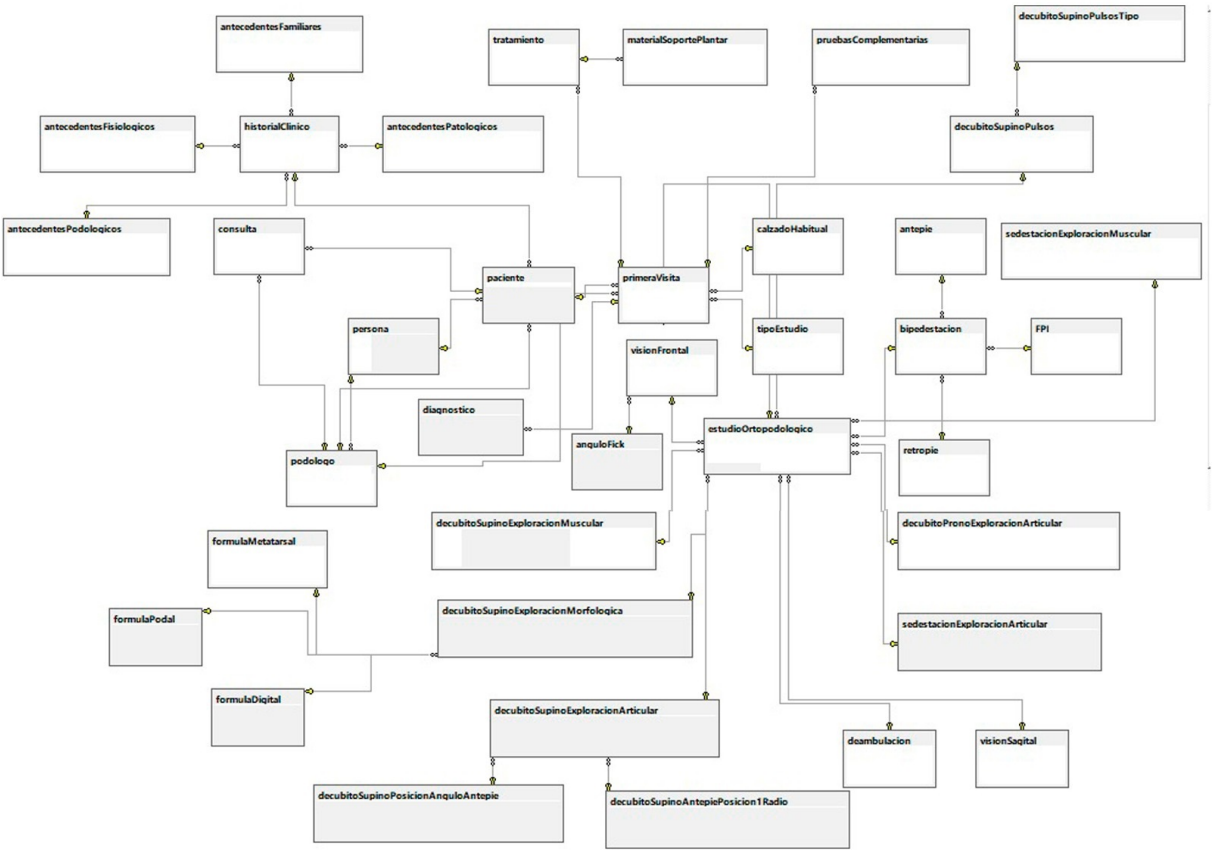

Figura 3. Vista general del esquema de la base de datos.

Como el diagrama de la base de datos es bastante grande iremos comentando las tablas por secciones.

En la siguiente imagen vemos que existen diferentes tipos de antecedentes y que éstos están relacionados con el historial clínico de un paciente. Los identificadores son de tipo *uniqueidentifier*, ya que se trata de una cadena hexadecimal con el formato xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx, donde cada x es un valor de 0-9 ó de a-f. Éste al ser aleatorio es muy improbable que se repita, ya que el para que se produzca una colisión se necesitan generar un elevadísimo número de GUIDs.

La mayoría de los campos son preguntas cuya respuesta es *Sí* o *No*, por lo que para ello se ha empleado el tipo *bit*. De esta manera en el posterior diseño de la web será más fácil la conversión a tipo bool.

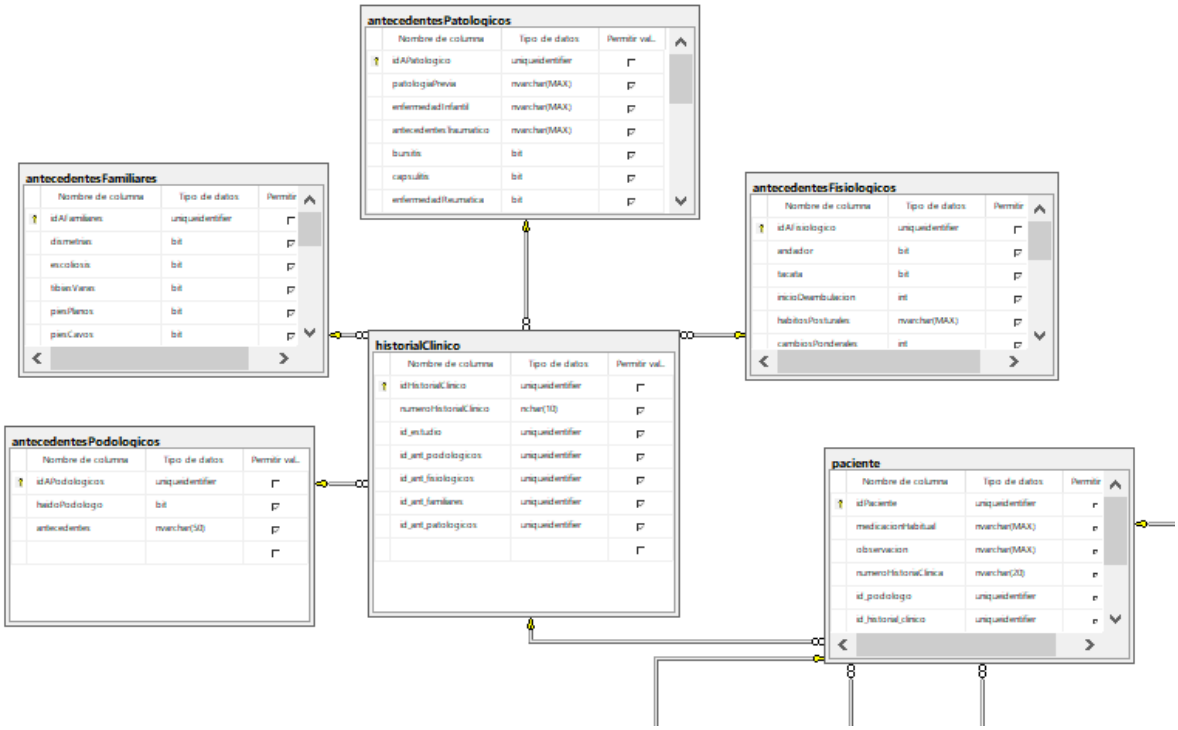

Figura 3.1. Vista de las diferentes tablas de *Antecedentes* y la relación con la tabla de *Paciente e Historial clínico*.

En la siguiente imagen vemos lo que se comentó anteriormente. Existe la entidad paciente donde se guardan los datos clínicos y, por otro lado, la entidad persona donde se guardan los datos personales. Un podólogo atiende a un paciente pero, también, un podólogo puede ser un paciente. Además, el paciente tiene una primera visita y el podólogo realiza las sucesivas consultas a un paciente. Los atributos que se traduzcan en un futuro campo de texto se añaden a la base de datos con el tipo *nvarchar(MAX)*.

Figura 3.2. Vista de las relaciones entre las tablas *Persona, Paciente, Podólogo, Consulta y Primera visita.*

En la siguiente imagen vemos las pruebas que se realizan en la primera visita. Cuando un paciente acude por primera vez a la consulta se le realiza uno de los tres tipos de estudio disponibles. También se le realiza un diagnóstico y como respuesta se le envía un tratamiento. A veces puede ser necesaria la realización de pruebas complementarias tales como radiografías, ecografías, etc.

Figura 3.3. Vista de las relaciones de varias tablas con la tabla *Primera visita*.

En la siguiente imagen se muestra una parte de los datos que se rellenan durante un estudio. Para que fuera una división más cómoda a la hora de crear la tablas, el estudio se ha dividido en diferentes entidades. Cada entidad corresponde a una parte de cómo se hace el estudio, es decir, tenemos la tabla *Bipedestación* porque el estudio al paciente se le realiza en bipedestación. En ella se almacenan todos los datos correspondientes a las pruebas realizadas en esa posición. Igual para el resto de tablas.
Esta imagen es una captura parcial, ya que hay bastantes tablas como se observa en la imagen principal [Figura 3.]. Pero la forma de razonamiento es la siguiente: se crea una tabla por cada posición de estudio, evitando así tener grandes tablas debido a la gran cantidad de atributos.
Por otro lado, en la tabla *Decubito Supino Exploración Muscular* hay dos identificadores y los atributos están repetidos, pero no son iguales. Esto se debe a que las pruebas en este momento se realizan dos veces, una para el pie derecho y otra para el pie izquierdo. Por tanto, es necesario guardar dos valores.

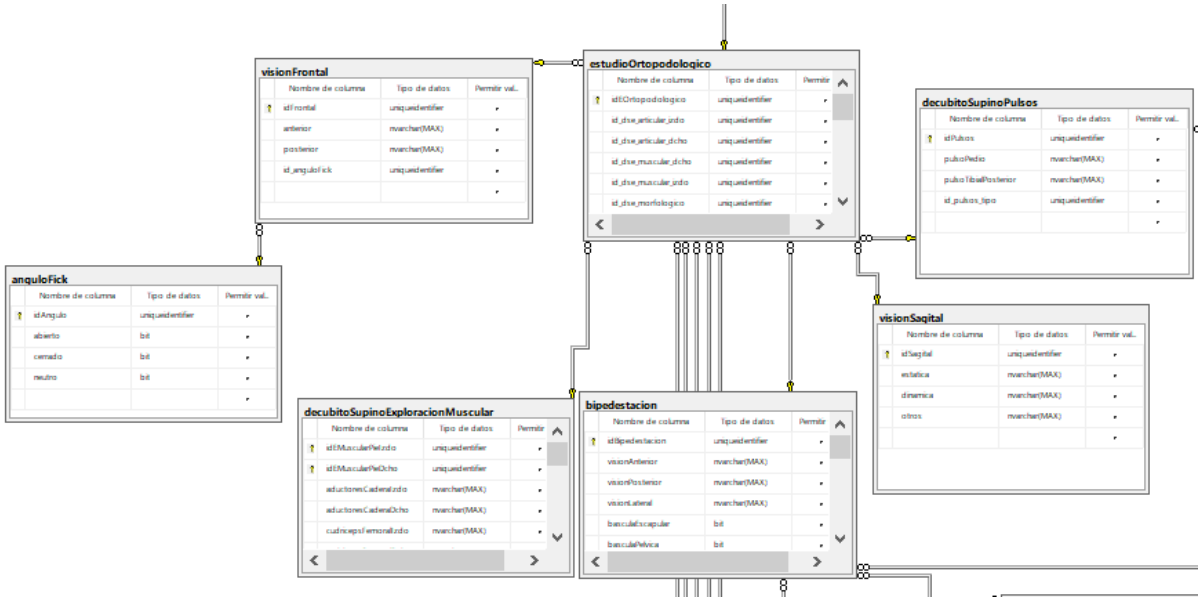

Figura 3.4. Vista de las relaciones entre algunas de las tablas con la tabla *Estudio ortopológico*.

3.2.2 Diseño de la Aplicación Web

Se va a dividir esta sección en tres apartados, ya que se ha utilizado el patrón MVC comentado anteriormente. Por tanto, vamos a centrar la explicación del diseño de la aplicación web en este patrón. Para ello, en cada apartado se va a hacer una descripción del proceso llevado a cabo.

Modelo

- EntrarModel.

  El primer modelo llevado a cabo ha sido EntrarModel. En él tenemos dos clases, una para hacer login (*EntrarModel*) y otra para registrarse (*RegistroModel*). Debemos declarar los campos que necesitamos. En *EntrarModel* declaramos Usuario y Contraseña. Y en *RegistroModel* como hereda de *EntrarModel*, sólo declaramos ConfirmarContraseña.

- FormPaciente.

  El segundo modelo llevado a cabo ha sido FormPaciente. Para todos los siguientes modelos se ha seguido el mismo formato. Este es:

  - Declaración de los campos de la base de datos.
  - Creación de la función *Rellenar()* que posteriormente será llamada en el controlador.
  - Creación de la función *InsertarEn()* que posteriormente será llamada en el controlador.
  - Creación de la función *GuardarEn()* que posteriormente será llamada en el controlador.

Entonces, el primer paso ha sido declarar todos los campos que se necesitaban de la base de datos. Este modelo va a requerir datos de varias tablas: principalmente de *Paciente* y *Persona*, pero también de *HistorialClinico* y *PrimeraVisita*.
Recordamos que los identificadores eran de tipo *uniqueidentifier*, las preguntas cuya respuestas eran Sí o No eran de tipo *bit*, los campos con respuesta de texto eran *nvarchar(MAX)* y la fecha era de tipo *DateTime*. Todos los campos no son necesarios, por ello en la base de datos se permitían valores nulos en ciertos datos. Esto en la clase FormPaciente se traduce de la siguiente manera:

- Los valores que permitan nulos se representan de la siguiente forma: tipo?. Por ejemplo:

  ```
  public int? Edad { get; set; }
  ```

  En este caso, el valor de edad puede ser nulo y se indica mediante int?.
- Los valores tipo nvarchar(MAX) se traducen a tipo string. Para este tipo **no** hay que indicar

  ```
  string?
  ```

  en caso de valor nulo.
- El tipo de la fecha se sigue manteniendo en DateTime.

Por otro lado, se han indicado ciertas restricciones. Por ejemplo la fecha queremos que siga el formato dd-MM-yyyy, para ello hay que usar:

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

Así de este modo estamos indicando el formato en el que queremos que se introduzca la fecha.

Además, podemos indicar que un campo sea requerido en el formulario indicando:

```
[Required()]
```

haciendo así, que si no se inserta el campo en la vista, se nos muestre un error.

Finalmente, podemos añadir un texto que nos puede servir para dos cosas:

1. Saber cuál es la propiedad a la que nos estamos refiriendo.
2. Poder llamar después ese texto desde la vista. De esta manera en la etiqueta se muestra el texto indicado dentro de la función *DisplayName()*.

```
[DisplayName("Fecha nacimiento")]
```

De este modo, obtendríamos:

```
[DisplayName("Fecha nacimiento")]
[DisplayFormat(DataFormatString = "{dd-MM-yyyy}")]
[Required()]
public DateTime? FechaNacimiento { get; set; }
```

Una vez realizado el proceso anterior para todos los campos de la base de datos que nos vayan a ser necesarios en el formulario, el siguiente paso es crear las distintas funciones que mencionamos anteriormente.

La idea de este modelo es que a través del identificador del paciente podamos recuperar los valores de las tablas Paciente y Persona. Por tanto, el método *Rellenar()* es el siguiente:

```
public static FormPaciente Rellenar(paciente paciente)
{
  persona persona = paciente.persona;
  return new FormPaciente
  {
    IdPaciente = paciente.idPaciente,
    MedicacionHabitual = paciente.medicacionHabitual,
    ...
    Nombre = persona.nombre,
    PrimerApellido = persona.apellido1,
    SegundoApellido = persona.apellido2,
    FechaNacimiento = persona.fechaNacimiento,
    ...
  };
}
```

Esta función recibe el identificador de un paciente. A partir de ese paciente nos creamos una persona, y luego pasamos a rellenar los campos declarados al principio con los valores que se encuentran en la base de datos. Finalmente, devolvemos el FormPaciente ya relleno.

Otro de los métodos creados es *InsertarEn()*. Esto lo hemos realizado de la siguiente forma:

```
public void InsertarEn(podologiaEntities podo)
{
  using (var tr = podo.Database.BeginTransaction())
        {
     try
      {
        var nuevoID = Guid.NewGuid();

                int ret = podo.Database.ExecuteSqlCommand(
          @"INSERT INTO persona(
          [idPersona],
          [nombre],
          [apellido1],
          [apellido2],
          [fechaNacimiento],
          ...
          )    VALUES(
          @p0, @p1,
          @p2, @p3,
```

```
            @p4,
            ...
            )",
            nuevoID,
            this.Nombre,
            this.PrimerApellido,
            this.SegundoApellido,
            this.FechaNacimiento,
            ...
            );
                    ...
      tr.Commit();
          }
          catch (Exception)
          {
       tr.Rollback();
       throw;
          }
  }
 }
```

A través de *podologiaEntities* accedemos a la base de datos.
Es necesario la creación de nuevos identificadores, para que así cada vez que introduzcamos un paciente nuevo, éste posea un identificador diferente.
Debemos hacer una sentencia *INSERT INTO* para cada tabla en la que queramos insertar datos. Aquí tenemos valores de la tabla Paciente, Persona, HistorialClinico y creamos el identificador de la PrimeraVisita.
La estructura del INSERT es la que se muestra arriba. Es importante que los campos escritos coincidan con los campos de las tablas de la base de datos creada. También es importante asegurarnos de que el campo *@px*, siendo x un número cualquiera, está haciendo referencia al campo correspondiente y se le asigna el valor deseado mediante *this.valor*, siendo *valor* el nombre de la propiedad.
Otro elemento importante, es hacer *Commit*, para guardar/confirmar los cambios realizados en la base de datos de forma permanente.

Por último, hemos creado el método *GuardarEn()*. Este lo que nos permite es poder realizar cambios en los elementos ya insertados en la base de datos. De este modo:

```
public void GuardarEn(podologiaEntities podo)
{
  paciente paciente = podo.paciente.Where(
  p => p.idPaciente == this.IdPaciente).FirstOrDefault();
  persona persona = paciente.persona;
  using (var tr = podo.Database.BeginTransaction())
    {
      try
      {
        Debug.Assert(
```

```
            this.IdPaciente == paciente.idPaciente);
            Debug.Assert(
            this.IdPodologo == paciente.id_podologo);

            int ret = podo.Database.ExecuteSqlCommand(
                @"UPDATE [paciente] SET
                [medicacionHabitual] = @p1,
                ...
                WHERE [idPaciente] = @p0
                ",
                paciente.idPaciente,
                this.MedicacionHabitual,
                ...
                );
            tr.Commit();
        }
        catch (Exception)
        {
            tr.Rollback();
            throw;
        }
            }
}
```

A través de *podologiaEntities* accedemos a la base de datos. A partir del identificador del paciente seleccionado, extraemos la información para las tablas paciente y persona. El siguiente paso sería comprobar que los identificadores se corresponden con los identificadores de las tablas que hemos recuperado. Y posteriormente, hacemos las sentencias UPDATE. Ahora, sólo se van a modificar las tablas Persona y Paciente, ni la tabla HistorialClinico, ni la tabla PrimeraVisita, ya que de estas dos últimas, dichos valores no podemos modificarlos. Por otro lado, debemos tener en cuenta las mismas cuestiones que para los INSERTS, es decir, hacer Commit, escribir adecuadamente los nombres de los campos de las tablas, etc.

- FormAntecedentes.

Para FormAntecedentes debemos declarar todos los campos de las diferentes tablas de antecedentes como son los antecedentes familiares, podológicos, patológicos y fisiológicos. En este caso, cuando hacemos el INSERT INTO a parte de insertar en las cuatro tablas de antecedentes debemos hacer un UPDATE de la tabla *HistorialClinico*. Hacemos UPDATE porque esa tabla ya se había creado anteriormente en *FormPaciente* y ahora solamente aprovechamos para añadir las claves ajenas (FK) de los distintos antecedentes.
Por otro lado, en el método *GuardarEn()* hacemos un UPDATE para las tablas de antecedentes y no lo hacemos para la tabla *HistorialClinico* pues estos valores no se modifican, ya que el número del historial clínico sólo se inserta una vez y no puede ser modificado, y las FK no se modifican.

- FormPrimeraVisita.

  En este Form tenemos información acerca de las tablas *PrimeraVisita* y *CalzadoHabitual*. Una peculiaridad es que la tabla *PrimeraVisita* ya fue creada cuando hicimos *FormPaciente*. Por ello, se ha decidido no hacer un método *InsertarEn()* y hacer un *GuardarEn()* directamente. De modo que hacemos un UPDATE para la tabla *PrimeraVisita* mientras que para la tabla *CalzadoHabitual* comprobamos si el calzado es nulo, es decir, no existe. En caso de que no exista hacemos un INSERT de dicha tabla, y en caso de que exista hacemos un UPDATE de dicha tabla.
  Lo mismo pasaría con el método *Rellenar()*. Hay que comprobar si la tabla *CalzadoHabitual* está rellena. En caso de que no sea nulo si que rellenamos los campos con los valores correspondientes.

- FormDiagnosticoTratamiento.

  En este Form abarcamos información de las tablas *Diagnostico*, *Tratamiento* y *MaterialSoportePlantar*. El principal detalle es la forma de hacer las sentencias de INSERT en la base de datos, ya que en *MaterialSoportePlantar* hay FK de tratamiento y en *Diagnostico* hay FK de tratamiento también. Por lo que primero insertamos *Tratamiento*, después hemos decidido insertar *MaterialSoportePlantar* y, por último, *Diagnostico*. En el caso del UPDATE no hay problema puesto que no necesitamos modificar las FK.

- FormConsulta.

  Para FormConsulta no hay ninguna peculiaridad, basta con declarar las propiedades y crear los métodos *Rellenar()*, *InsertarEn()* y *GuardarEn()*.

Controlador

- PagesController.

  Aquí debemos contemplar tres opciones: registrarse, hacer LogIn y hacer LogOut. En general, cada método tiene dos peticiones, una petición tipo GET y otra petición tipo POST. Con GET lo que hacemos es recuperar información del servidor, en este caso recuperamos los datos de nuestra base de datos. Mientras que con POST lo que hacemos es enviar información desde el lado del cliente para que sea procesada y actualizada o agregada en la base de datos.

- PacienteController.

  En el controlador disponemos de una serie de funciones como son *Create* y *Edit*. Como sus nombres indican una es para Crear el nuevo paciente y la otra es para Editar los datos de un paciente creado.

  - Create
    * GET. Simplemente devolvemos una vista vacía, ya que no hay nada que queramos recuperar de la base de datos, porque lo que pretendemos hacer es insertar un nuevo paciente.
    * POST. Aquí debemos llamar al método creado anteriormente en el modelo para así poder hacer las correspondientes insercciones en la base de datos.

  - Edit
    * GET. A partir de un identificador de un paciente lo que hacemos es recuperar y rellenar los campos del formulario. Esto se hace mediante el método Rellenar creado en el modelo.
    * POST. En esta petición lo que hacemos es llamar al método creado anteriormente en el modelo para así poder hacer las correspondientes actualizaciones en la base de datos.

- AntecedentesController.

  - Create
    * GET. Devolvemos una vista vacía, ya que no hay nada que queramos recuperar de la base de datos, porque lo que pretendemos hacer es insertar los antecedentes de un paciente.
    * POST. Debemos llamar al método creado anteriormente en el modelo para así poder hacer las correspondientes insercciones en la base de datos (*InsertarEn()*).

  - Edit
    * GET. A partir de un identificador de un paciente lo que hacemos es recuperar y rellenar los campos del formulario. Esto se hace mediante el método *Rellenar()* creado en el modelo.
    * POST. Llamamos al método creado anteriormente en el modelo para así poder hacer las correspondientes actualizaciones en la base de datos (*GuardarEn()*).

- PrimeraVisitaController.

  - Create
    * GET. Devolvemos una vista vacía, ya que no hay nada que queramos recuperar de la base de datos, porque lo que pretendemos hacer es insertar la primera visita de un paciente.
    * POST. Aquí debemos llamar al método creado anteriormente en el modelo para así poder hacer las correspondientes insercciones en la base de datos.

  - Edit
    * GET. A partir de un identificador de una visita lo que hacemos es recuperar y rellenar los campos del formulario. Esto se hace mediante el método *Rellenar()* creado en el modelo.
    * POST. En esta petición lo que hacemos es llamar al método creado anteriormente en el modelo para así poder hacer las correspondientes actualizaciones en la base de datos.

- DiagnosticoTratamientoController.

  - Create
    * GET. En esta petición devolvemos una vista vacía, ya que no hay nada que queramos recuperar de la base de datos, porque lo que pretendemos hacer es insertar el diágnostico y el tratamiento de un paciente.
    * POST. Aquí debemos llamar al método creado anteriormente en el modelo para así poder hacer las correspondientes insercciones en la base de datos.

  - Edit
    * GET. A partir de un identificador de un diagnóstico lo que hacemos es recuperar y rellenar los campos del formulario. Esto se hace mediante el método *Rellenar()* creado en el modelo.
    * POST. Llamamos al método creado en el modelo para así poder hacer las correspondientes actualizaciones en la base de datos.

- ConsultaController.

  - Create
    * GET. Devolvemos una vista vacía, ya que no hay nada que queramos recuperar de la base de datos, porque lo que pretendemos hacer es insertar una nueva consulta de un paciente.
    * POST. Llamamos al método creado en el modelo para así poder hacer las correspondientes insercciones en la base de datos.

  - Edit
    * GET. A partir de un identificador de un paciente lo que hacemos es recuperar y rellenar los campos del formulario. Esto se hace mediante el método *Rellenar()* creado en el modelo.
    * POST. En esta petición lo que hacemos es llamar al método creado anteriormente en el modelo para así poder hacer las correspondientes actualizaciones en la base de datos.

Vista[2]

Para el diseño de la interfaz la idea era la siguiente:

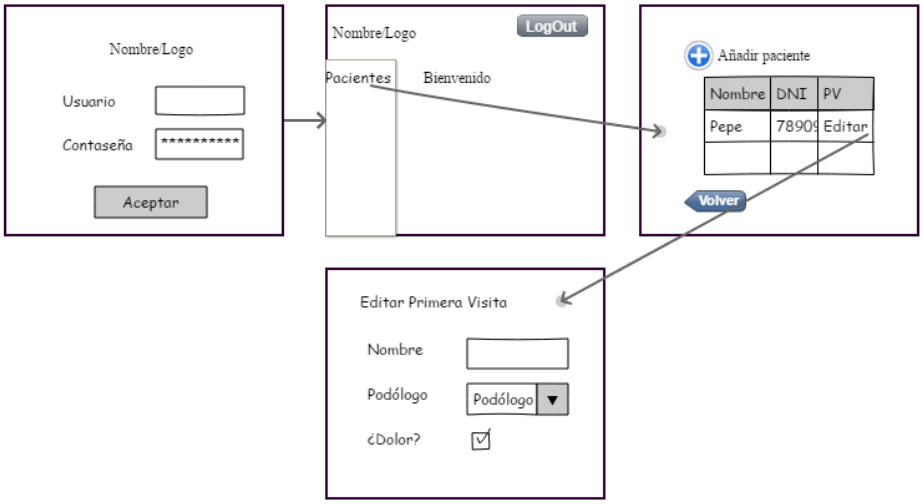

Figura 4. Primera versión del diseño de la aplicación web.

La idea era crear pantallas sencillas. De modo que en la primera pantalla se viese el logo y el nombre de la aplicación y estuviera el login. Una vez iniciada la sesión se muestra un menú lateral en el que se pudiera ver la lista de pacientes y en la barra superior tenemos un botón para cerrar la sesión. Si en el menú lateral hacemos click en el *Listado de pacientes*, se muestra una tabla y podemos añadir/editar la información que deseemos.

Para cada modelo y controlador hay diferentes vistas. Al menos una por cada conjunto de peticiones GET/POST.

A continuación se va a proceder a mostrar los resultados obtenidos para algunas vistas. En todo momento se ha pretendido que sea una interfaz sencilla y manejable por los usuarios. Para ello, se ha intentado que sea intuitiva y también se ha tenido en cuenta que probablemente se acceda a esta página web mediante un dispositivo móvil, como por ejemplo una tablet, ya que esto permite al podólogo un mayor acercamiento al paciente mientras se realiza la toma de datos.

[2]Para más detalle de las vistas consultar el Manual de Usuario.

- Registrar

  Vemos una vista en la que se nos pide introducir el nombre de usuario y la contraseña. Sólo el usuario administrador tiene acceso a esta vista, de manera que el administrador es el responsable de registrar a un nuevo usuario en la aplicación.

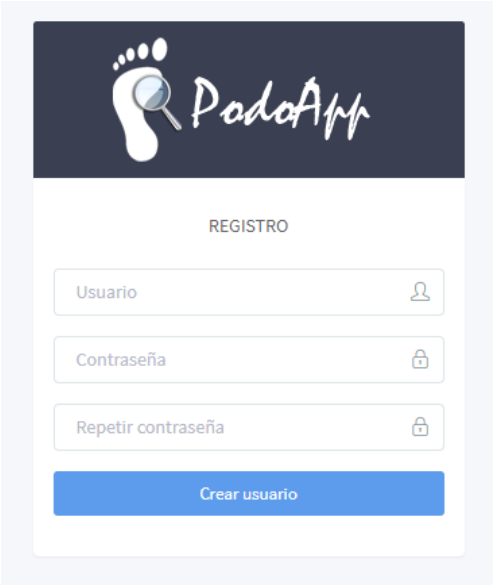

Figura 4.1. Interfaz de Registro de la aplicación.

- LogIn / LogOut

  Si el usuario está debidamente registrado, cuando acceda a la URL de la página web podrá rellenar esta vista tras la inserción de su nombre de usuario y su contraseña.

Figura 4.2. Interfaz del Inicio de Sesión.

Para cerrar sesión disponemos de un icono accesible desde cualquier punto de la aplicación. Tras pulsarlo volveremos a la vista de iniciar sesión.

Figura 4.3. Icono de cerrar sesión.

- Paciente.
  - Create View.

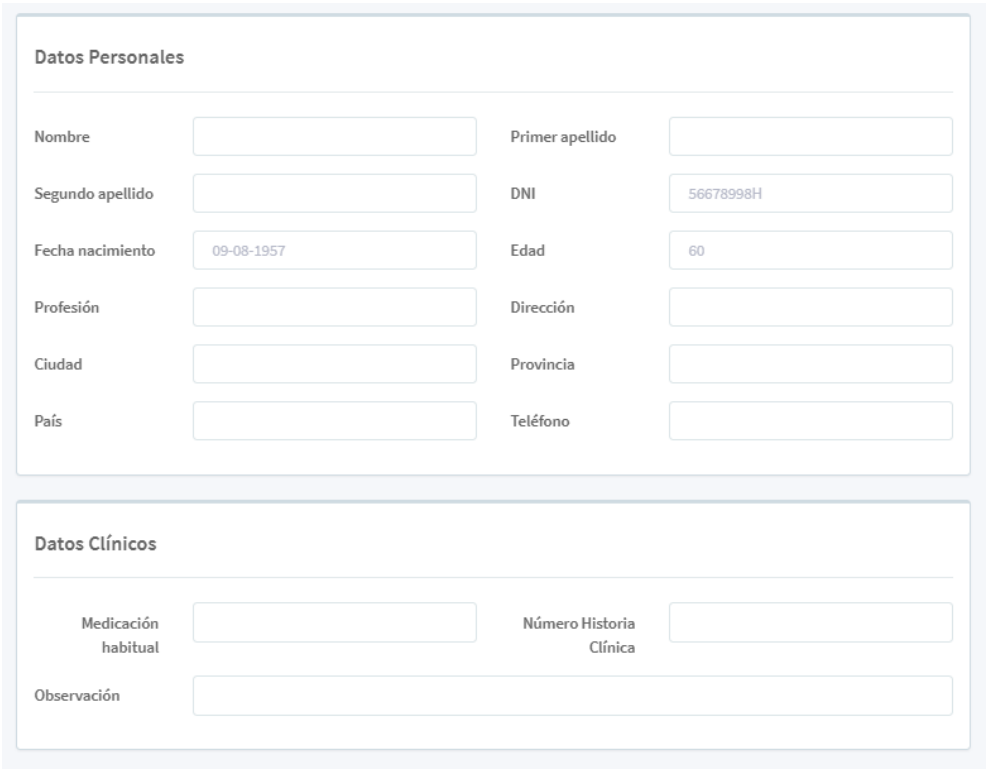

Figura 4.4. Interfaz de *Añadir los datos de un paciente*.

Vemos una vista sencilla en la que podemos rellenar los campos de *Datos del Paciente* y *Datos Clínicos*. En caso de que no se hayan introducido todos los datos obligatorios, o bien, en el formato adecuado, se muestra un mensaje bajo la caja de texto a la que se esté refiriendo.

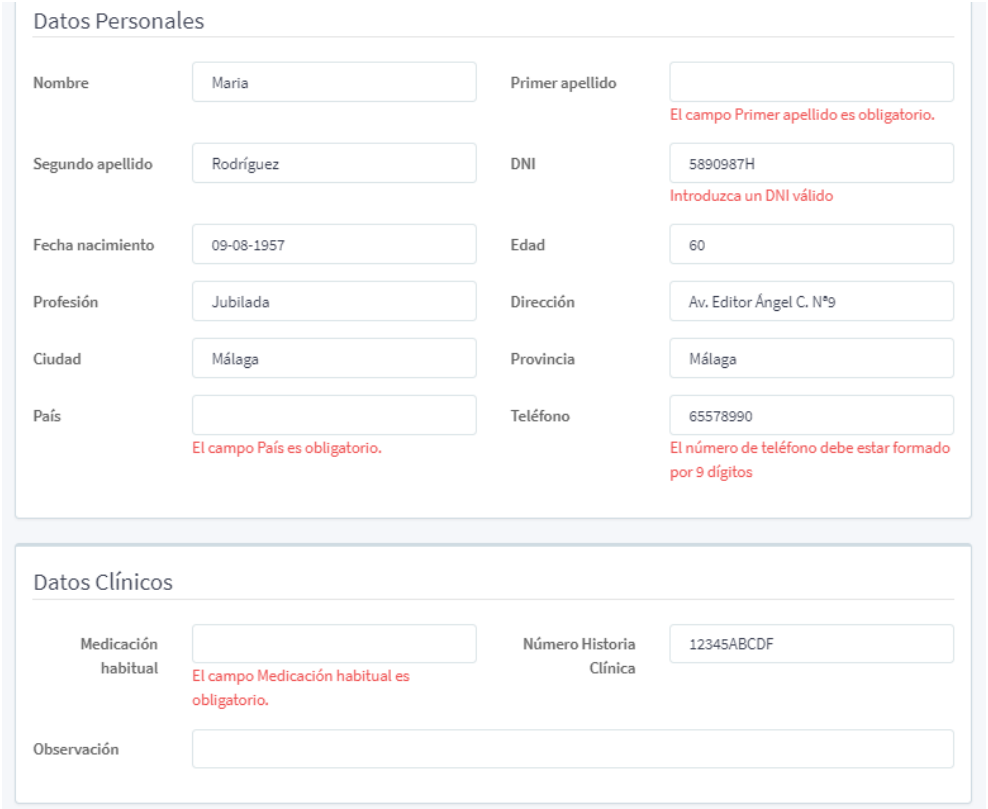

Figura 4.5. Interfaz de *Añadir los datos de un paciente* con errores.

Cuando todo está correcto y el paciente se ha insertado en la base de datos, volvemos a la tabla principal en la que se muestra la lista de los pacientes.

– Edit View.

También disponemos de una vista de edición del formulario anteriormente relleno. En caso de error o necesidad de modifcar algunos parámetros podemos ir al botón de *Editar* y podemos hacer los cambios necesarios. Por lo general, los formularios de edición tienen el mismo formato que el formulario de creación, para que así resulte más fácil a los usuarios de la aplicación.

- Listado de Pacientes.

Vemos una tabla con todos los pacientes que están registrados en el sistema. Si acabamos de resgitrar a un paciente y queremos añadir los datos, hacemos click sobre las distintas columnas de la tabla para llegar a los distintos cuestionarios y poder completarlos. En caso de querer editar los datos, hacemos el mismo proceso pero pulsando sobre editar.

Figura 4.6. Interfaz de la tabla con el listado de los pacientes.

- Listado de Consultas.

  Vemos una tabla con todos las consultas de un paciente concreto. Si queremos editar una visita en concreto pulsamos sobre el botón de *Editar Consulta*, de manera que nos lleva al formulario de editar la consulta y podemos hacer los cambios pertinentes.

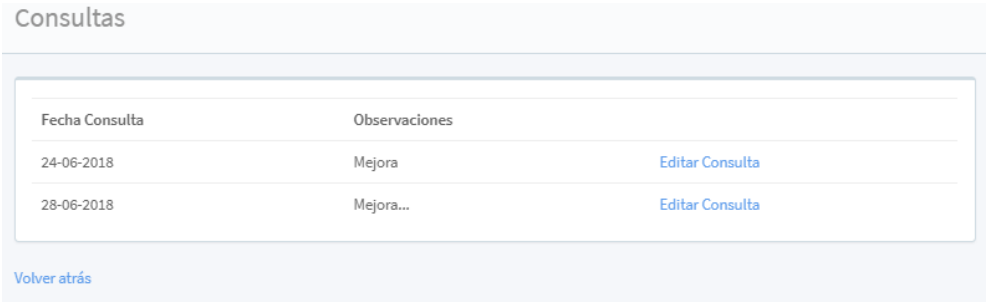

Figura 4.7. Interfaz de la tabla con el listado de consultas de un paciente.

4 Pruebas

Las pruebas se han ido realizando a medida que se hacía la implementación. Para ello, en primer lugar, se fueron probando los distintos módulos por separado, es decir, inserción de paciente, inserción de antecedentes, etc. En segundo lugar se procedió a la prueba conjunta de todos los módulos, de modo que si insertábamos un paciente y posteriormente los antecedentes, podíamos comprobar que los datos se insertaban y almacenaban correctamente mediante las relaciones de clave ajena. Para más detalle acerca de las pruebas realizadas consultar el *Anexo II*.

5 Conclusiones

Finalmente, hemos obtenido parte de una aplicación web que permite a los podólogos de la Unidad Asistencial de Podología de la Universidad de Málaga poder empezar a digitalizar dicha unidad. De modo, que van a poder, en cierto modo, ya que aún no está terminada, poder empezar a probar este nuevo sistema y ver si les resulta más útil y ahorran tiempo respecto a la anterior forma de recolección de datos.

Este TFG ha sido el comienzo de un gran proyecto con metas bastantes claras:

1. Digitalizar la Unidad de Podología.
2. Investigar en el campo de la Podología.

Es un proyecto con el que he podido aplicar todos mis conocimientos aprendidos a lo largo de la carrera sobre *Ingeniería del Software* y conceptos de *programación*. Además, he tenido que aprender a trabajar con ASP.NET, HTML y CSS, lo cual creo que me va a ser muy útil a lo largo de mi vida profesional.

A nivel personal me ha supuesto poder conocer de primera mano lo que es llevar a cabo un proyecto real, mediante reuniones con clientes, observar su forma de trabajo y sobre todo poder intentar ayudarlos a mejorar en su rutina de trabajo. Considero que cuando este proyecto se lleve a su fin, los miembros de la Unidad de Podología, tanto alumnos, como podólogos, lo van a agradecer, ya que se pretende facilitarles su labor y permitirles un acercamiento al paciente mientras rellenan los distintos cuestionarios. Además, de un fácil y rápido acceso a los datos de los pacientes que ya están registrados en el sistema.

5.1 Futuras líneas de trabajo

Existen varias líneas de trabajo como son:

- Crear el estudio ortopodológico.
- Crear el estudio quiropodológico.
- Crear el estudio quirúrgico.
- Aplicar técnicas de Minería de Datos para poder hacer investigación en el campo de la podología.

6 Bibliografía

[1] Software, A. URL: https://www.aquarsoftware.com/software-medico/

[2] S.L., D.I. URL: https://www.dasi.es/podologia/

[3] S.L., A.I. URL: https://www.apclinic.es

[4] NetClinicas. URL: https://www.netclinicas.com/software+clinicas+podologia.html

[5] S.L., Q. URL: http://www.qclinicas.com/servicios-para-clinicas/

[6] Michael E. Munson, James S. Wrobel, C.M.H., Hanauer, D.A.: Data mining for identifying novel associations and temporal relationships with charcot foot. **2014** (4 2014) 354–359

[7] Schmidt BM, Wrobel JS, M.M.R.G.H.C.: Podiatry impact on high-low amputation ratio characteristics: A 16-year retrospective study. **126** (4 2017) 272–277

[8] Trello. URL: https://trello.com/

[9] Gala, I.: Asp.net. URL: https://institutogala.com/codigo-m019-curso-asp-net-app-web-iosandroid-windows-24-h/

[10] colaborators, M.: Introducción a asp.net. URL: https://docs.microsoft.com/es-es/aspnet/overview

[11] colaborators, M.: Asp.net mvc 5. URL: https://docs.microsoft.com/es-es/aspnet/mvc/mvc5

[12] Rouse, M.: Sql server. URL: https://searchdatacenter.techtarget.com/es/definicion/SQLServer

ANEXO I

Manual de Usuario

Una vez hemos accedido a la URL en la que se aloja la aplicación nos encontramos con la siguiente pantalla:

Figura 1. Interfaz del Inicio de Sesión.

En ella, el usuario que previamente ha sido registrado puede iniciar sesión introduciendo su *Nombre de Usuario* y *Contraseña*.
Posteriormente se muestra la siguiente pantalla:

Figura 2. Interfaz de la Pantalla Principal.

En ella si movemos el ratón y pinchamos en *Listado de pacientes*, se nos muestra una tabla con los pacientes actuales registrados en el sistema [3].

[3] Los datos representados en la imagen son datos ficticios.

Figura 3. Interfaz del Listado de Pacientes.

Una vez aquí, si queremos crear un nuevo paciente, hacemos click sobre *Crear nuevo paciente*. A continuación, se nos mostrará un formulario para rellenar los datos personales y clínicos del paciente.

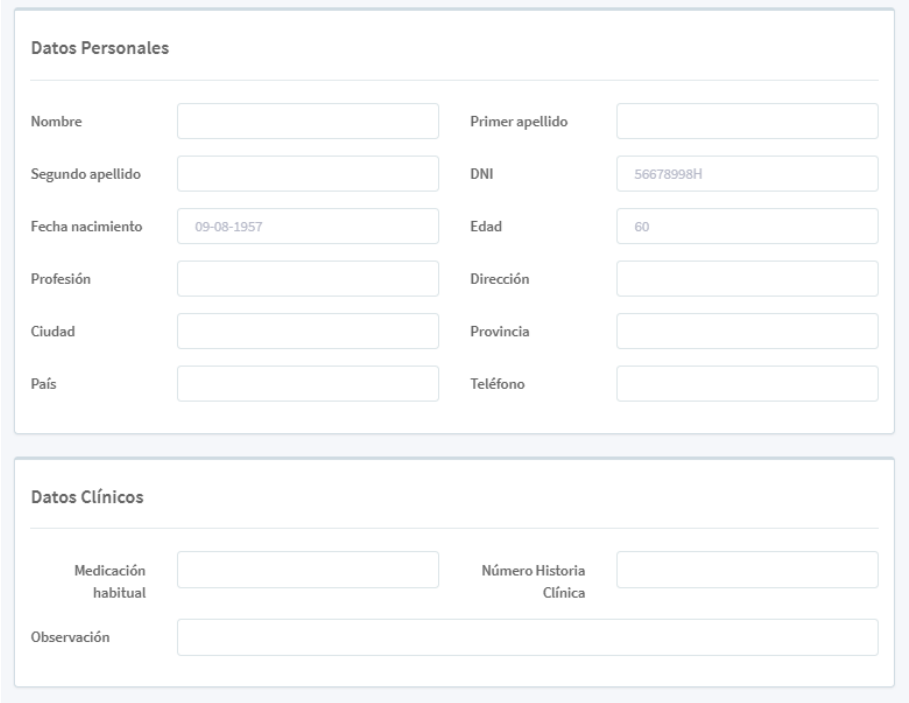

Figura 4. Interfaz de Crear un nuevo Paciente.

Una vez hemos introducido los datos, pulsamos sobre el botón *Crear paciente*. Si los datos han sido insertados en el formato adecuado, el paciente se insertará en la tabla y volveremos a la pantalla anterior en la que veíamos el listado de pacientes. En cambio, si algún campo del formulario es obligatorio y no se ha rellenado o algún valor no se ha introducido como se requiere se mostrará un mensaje de error. Tras

corregir los problemas que se nos han presentado, volvemos a pinchar sobre *Crear paciente* y volveremos a la pantalla con el listado de los pacientes.

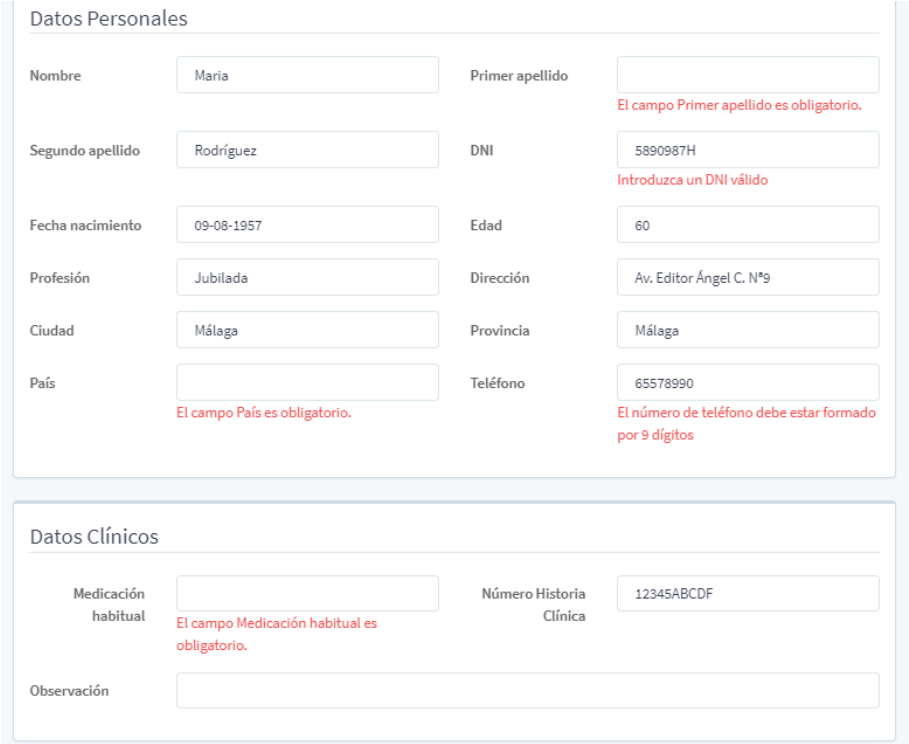

Figura 5. Interfaz de Error tras Crear un nuevo Paciente.

Todos los campos del formulario, que no son campos de texto libre, tienen una ayuda para ver el formato del texto que se debe introducir.

Desde el *Listado de Pacientes*, localizamos el paciente que acabamos de introducir y debemos rellenar los diferentes formularios en orden. En primer lugar rellenamos el formulario de antecedentes, posteriormente rellenamos el formulario de primera visita, en tercer lugar el formulario de diagnóstico y tratamiento, y, por último, en posteriores visitas, se rellenará el formulario de consultas.

Emepecemos por completar el formulario de antecedentes:

En este formulario debemos rellenar cuatro tipos de Antecedentes: patológicos, podológicos, fisiológicos y familiares. Para que sea lo más flexible posible, la mayoría de los campos no son obligatorios. Además, los cuestionarios se componen de preguntas de escritura y de selección.

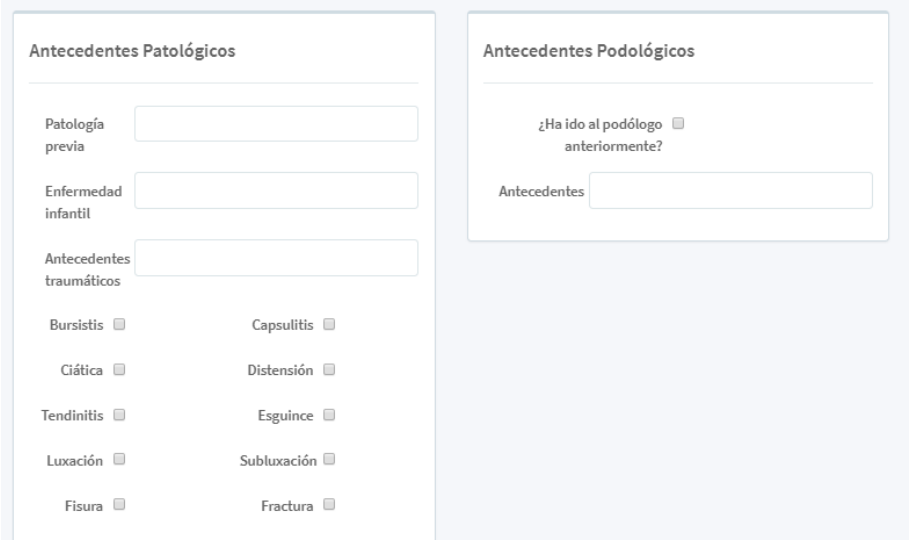

Figura 6. Interfaz de Añadir los Antecedentes Patológicos y Podológicos.

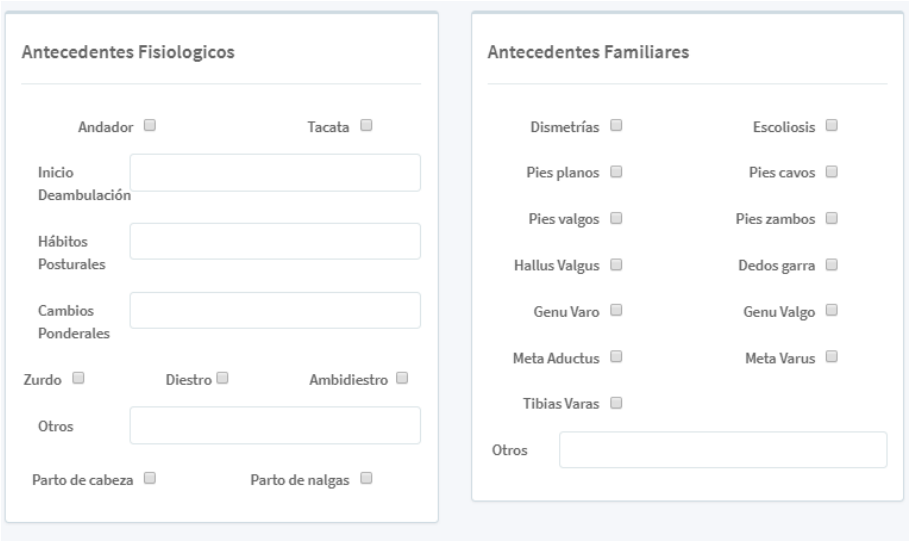

Figura 7. Interfaz de Añadir los Antecedentes Fisiológicos y Familiares.

Una vez rellenos todos los campos necesarios se pulsa sobre el botón *Añadir*, y volveremos al *Listado de Pacientes*. Si por casualidad nos hemos equivocado introduciendo algún valor del formulario, o bien, en consultas posteriores el paciente descubre que en sus antecedentes familiares había algún dato que anteriormente no se comentó, podemos pulsar sobre el botón *Editar* de Antecedentes y modificar los campos necesarios.

El siguiente paso es rellenar el formulario de Primera Visita. Éste sólo se puede rellenar una vez, puesto que el paciente sólo tiene una primera visita. Este formulario se compone de dos partes. En primer lugar, se realizan preguntas generales:

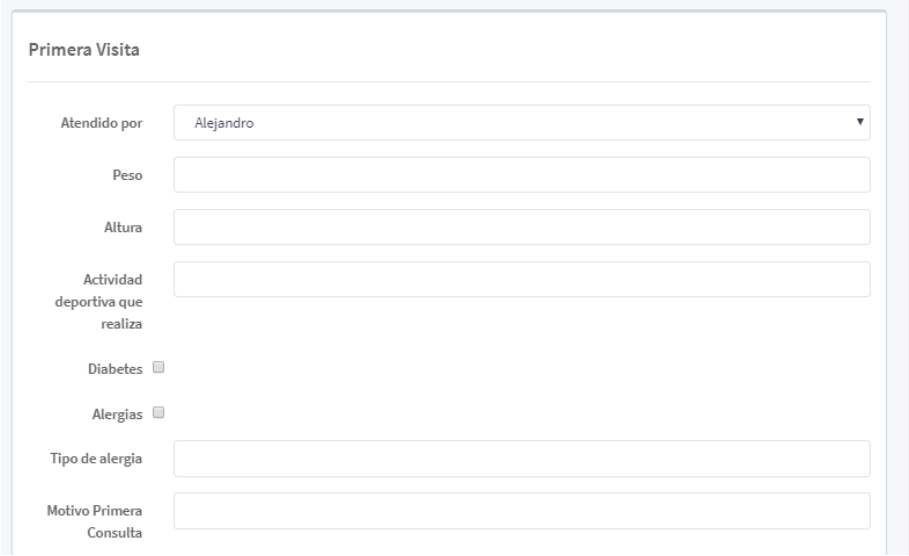

Figura 8. Interfaz de Añadir la Primera Visita (I).

Uno de los detalles de esta vista es que posee un desplegable para poder seleccionar el podólogo que realiza la consulta.

En segundo lugar, se pregunta acerca del calzado:

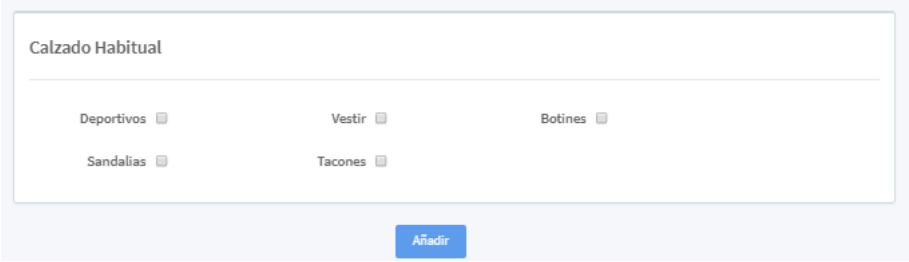

Figura 9. Interfaz de Añadir la Primera Visita (II).

Una vez rellenos los campos necesarios, pulsamos sobre el botón de *Añadir*. Si todo esta correcto volveremos a la página de *Listado de pacientes*.

El siguiente paso es crear el diagnóstico y el tratamiento del paciente. Para ello, en su columna pulsamos sobre el botón *Añadir*. A continuación se nos mostrará la siguiente pantalla:

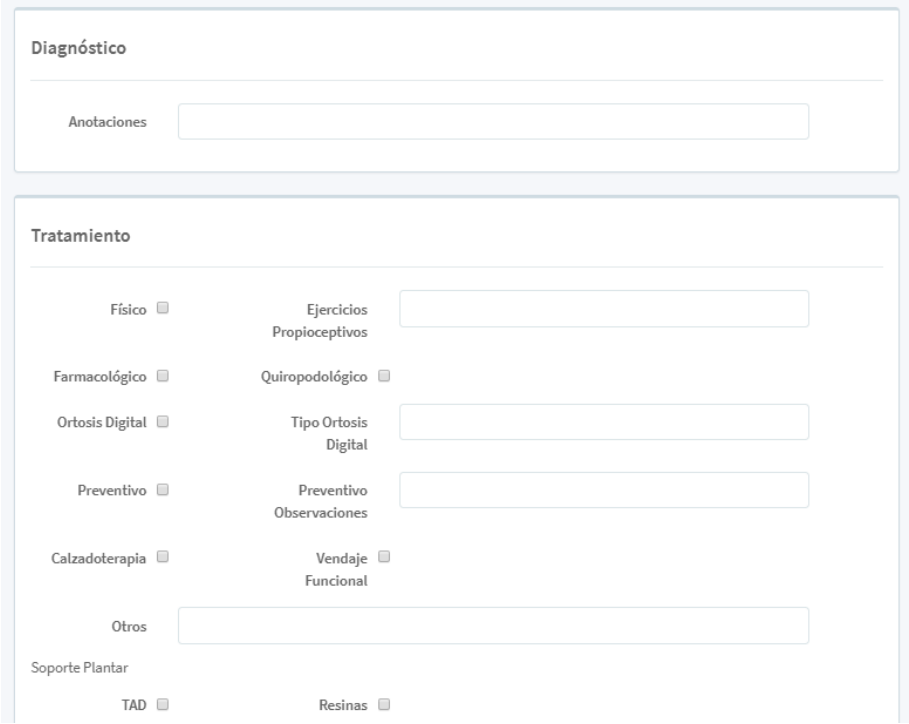

Figura 9. Interfaz de Añadir el Diagnóstico y el Tratamiento.

Una vez rellenos los datos pulsamos sobre *Añadir* y volvemos a la página del *Listado de Pacientes*. Si por algún motivo nos hemos equivocado escribiendo o se nos ha olvidado algún campo importante, podemos pulsar sobre *Editar*. Se nos volverá a mostrar el mismo formulario con las respuestas que habíamos puesto y podemos hacer los cambios pertinentes.

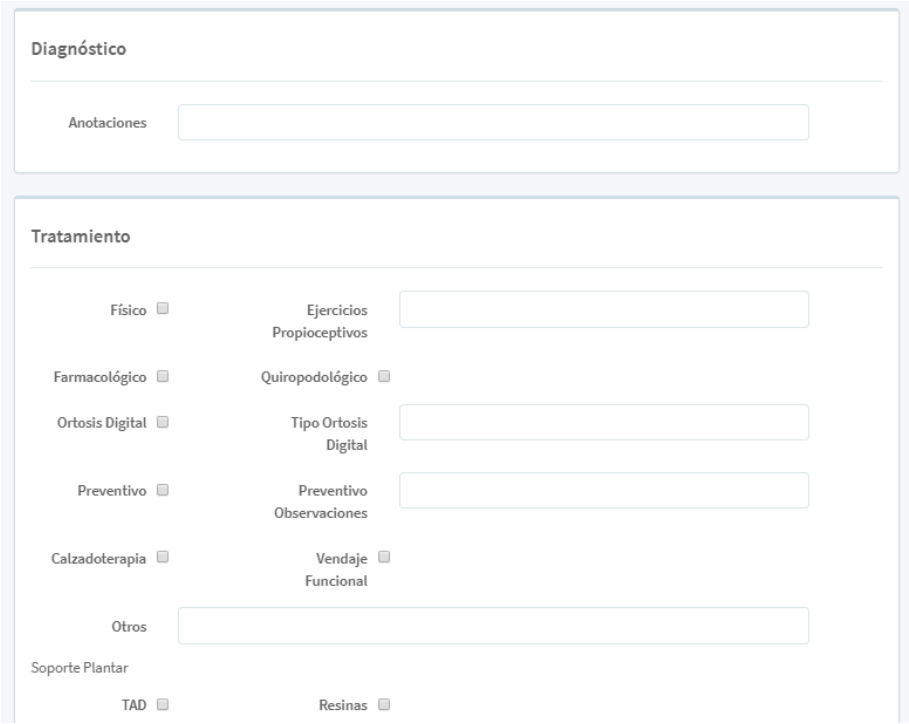

Figura 10. Interfaz de Editar el Diagnóstico y el Tratamiento.

Tras las modificaciones pulsamos *Modificar*. Para todos los formularios, si no queremos hacer la adicción o la edición podemos pulsar sobre *Volver atrás* en cualquier momento.

Por último, el paciente tendrá visitas posteiores. Para ello en el apartado *Consultas* disponemos de dos botones. En primer lugar, encontramos el botón de *Añadir* para añadir una nueva consulta.

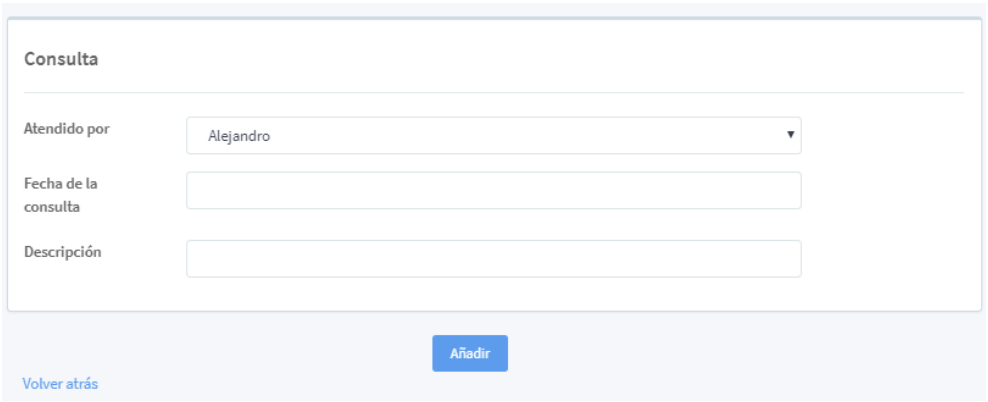

Figura 11. Interfaz de Añadir una nueva Consulta.

En segundo lugar, encontramos el botón de *Consultas*. Si pulsamos en él nos encontramos con un listado de todas las consultas que ha ido teniendo el paciente.

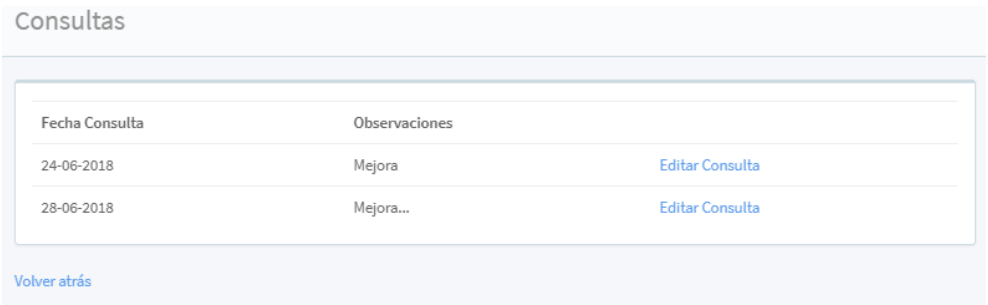

Figura 12. Interfaz del listado de consultas de un paciente.

En caso de querer editar algunas de las consultas anteriores bastaría con pulsar sobre el botón de *Editar* de la consulta que deseemos.

ANEXO II

Casos de Prueba

A continuación se detallan algunas de las funcionalidades de la web mediante los casos de prueba.

Título	CP01. Acceso a la web.	
Propósito	Comprobar que el sistema realiza el inicio de sesión	
Autor	Maria José Muñoz González	
Pasos	Resultados Esperados	Resultados Reales
1. Insertar usuario y contraseña		
2. Hacer click en Entrar	Redirección a la página principal	Redirección a la página principal

Título	CP02. Registrar un nuevo paciente.	
Propósito	Comprobar que se guarda el nuevo paciente registrado	
Autor	Maria José Muñoz González	
Pasos	Resultados Esperados	Resultados Reales
1. Hacer click en Añadir nuevo paciente	Redirección al formulario de crear paciente	Redirección al formulario de crear paciente
2. El usuario rellena los datos y hace click en Añadir	El sistema guarda los datos en la base de datos y se redirige al listado de pacientes	El sistema guarda los datos en la base de datos y se redirige al listado de pacientes

Título	CP03. Añadir los Antecedentes del paciente nuevo.	
Propósito	Comprobar que se guardan los antecedentes del nuevo paciente	
Autor	Maria José Muñoz González	
Pasos	Resultados Esperados	Resultados Reales
1. Hacer click en Añadir antecedentes	Redirección al formulario de crear antecedentes	Redirección al formulario de crear antecedentes
2. El usuario rellena los datos y hace click en Añadir	El sistema guarda los datos en la base de datos y se redirige al listado de pacientes	El sistema guarda los datos en la base de datos y se redirige al listado de pacientes
3. El usuario ve en listado de pacientes Editar en la columna Antecedentes	El sistema muestra Editar antecedentes porque ya están añadidos	El sistema muestra Editar antecedentes porque ya están añadidos

Título	CP04. Editar la Primera Visita del paciente nuevo.	
Propósito	Comprobar que se guardan los datos de la primera visita del nuevo paciente	
Autor	Maria José Muñoz González	
Pasos	Resultados Esperados	Resultados Reales
1. Hacer click en Editar Primera Visita	Redirección al formulario de crear primera visita	Redirección al formulario de crear primera visita
2. El usuario rellena los datos y hace click en Añadir	El sistema guarda los datos en la base de datos y se redirige al listado de pacientes	El sistema guarda los datos en la base de datos y se redirige al listado de pacientes

Título	CP05. Añadir el Diagnóstico y Tratamiento del paciente nuevo.	
Propósito	Comprobar que se guarda el Diagnóstico y Tratamiento del nuevo paciente	
Autor	Maria José Muñoz González	
Pasos	Resultados Esperados	Resultados Reales
1. Hacer click en Añadir Diágnostico y Tratamiento	Redirección al formulario de crear diagnóstico y tratamiento	Redirección al formulario de crear diagnóstico y tratamiento
2. El usuario rellena los datos y hace click en Añadir	El sistema guarda los datos en la base de datos y se redirige al listado de pacientes	El sistema guarda los datos en la base de datos y se redirige al listado de pacientes
3. El usuario ve en listado de pacientes Editar en la columna Disgnóstico y Tratamiento	El sistema muestra Editar diagnóstico y tratamiento porque ya están añadidos	El sistema muestra Editar diagnóstico y tratamiento porque ya están añadidos

Título	CP06. Añadir una consulta de un paciente.	
Propósito	Comprobar que se guarda la nueva consulta del paciente	
Autor	Maria José Muñoz González	
Pasos	Resultados Esperados	Resultados Reales
1. Hacer click en Añadir Consulta	Redirección al formulario de crear consulta	Redirección al formulario de crear consulta
2. El usuario rellena los datos y hace click en Añadir	El sistema guarda los datos en la base de datos y se redirige al listado de pacientes	El sistema guarda los datos en la base de datos y se redirige al listado de pacientes

Título	CP07. Consultar el listado de consultas de un paciente.	
Propósito	Comprobar que el sistema muestra correctamente el listado de consultas del paciente	
Autor	Maria José Muñoz González	
Pasos	Resultados Esperados	Resultados Reales
1. Hacer click en Consultas	Redirección a un listado de conusltas anteriores	Redirección a un listado de conusltas anteriores

Título	CP08. Editar un paciente.	
Propósito	Comprobar que se guarda el paciente cuyos datos han sido modificados	
Autor	Maria José Muñoz González	
Pasos	Resultados Esperados	Resultados Reales
1. Hacer click en Editar datos personales	Redirección al formulario de editar paciente	Redirección al formulario de editar paciente
2. El usuario modifica los datos y hace click en Editar	El sistema guarda los datos en la base de datos y se redirige al listado de pacientes	El sistema guarda los datos en la base de datos y se redirige al listado de pacientes

Título	CP09. Añadir los Antecedentes del paciente nuevo.	
Propósito	Comprobar que se editan los antecedentes del nuevo paciente	
Autor	Maria José Muñoz González	
Pasos	Resultados Esperados	Resultados Reales
1. Hacer click en Editar Antecedentes	Redirección al formulario de editar antecedentes	Redirección al formulario de editar antecedentes
2. El usuario rellena los datos y hace click en Modificar	El sistema guarda los datos en la base de datos y se redirige al listado de pacientes	El sistema guarda los datos en la base de datos y se redirige al listado de pacientes
3. El usuario ve en listado de pacientes Editar en la columna Antecedentes	El sistema muestra Editar antecedentes porque se pueden volver a editar	El sistema muestra Editar antecedentes porque se pueden volver a editar

Título	CP10. Editar el Diagnóstico y Tratamiento del paciente nuevo.	
Propósito	Comprobar que se guarda el Diagnóstico y Tratamiento del nuevo paciente	
Autor	Maria José Muñoz González	
Pasos	Resultados Esperados	Resultados Reales
1. Hacer click en Editar Diágnostico y Tratamiento	Redirección al formulario de editar diagnóstico y tratamiento	Redirección al formulario de editar diagnóstico y tratamiento
2. El usuario modifica los datos y hace click en Editar	El sistema actualiza los datos en la base de datos y se redirige al listado de pacientes	El sistema actualiza los datos en la base de datos y se redirige al listado de pacientes
3. El usuario ve en listado de pacientes Editar en la columna Disgnóstico y Tratamiento	El sistema muestra Editar diagnóstico y tratamiento porque se pueden volver a editar	El sistema muestra Editar diagnóstico y tratamiento porque se pueden volver a editar

Título	CP11. Editar la consulta de un paciente.	
Propósito	Comprobar que se puede editar las consultas de un paciente	
Autor	Maria José Muñoz González	
Pasos	Resultados Esperados	Resultados Reales
1. Hacer click en Consultas	Redirección a un listado de conusltas anteriores	Redirección a un listado de conusltas anteriores
2. Hacer click en Editar de la consulta deseada	Redirección al formulario de edición de dicha consulta	Redirección al formulario de edición de dicha consulta
3. El usuario modifica los campos necesarios y hace click en Modificar	El sistema actualiza los datos en la base de datos y se redirige al listado de pacientes	El sistema actualiza los datos en la base de datos y se redirige al listado de pacientes

Título	CP12. Desconectarse de la web.	
Propósito	Comprobar que el sistema realiza el cierre de sesión	
Autor	Maria José Muñoz González	
Pasos	Resultados Esperados	Resultados Reales
1. Hacer click sobre el icono de cerrar sesión	El sistema nos redirije a la pantalla de inicio de sesión	El sistema nos redirije a la pantalla de inicio de sesión