Componentes

Pablo Eduardo Ojeda Vasco

8 de septiembre de 2011

Capítulo 1

Diseño

1.1. Diseño a nivel de componentes

1.1.1. Introducción

El diseño a nivel de componentes define las estructuras de los datos, los algoritmos, las características de la interfaz y los mecanismos de comunicación asignados a cada componente. Son muy importantes puesto que representan un nivel en el que es posible revisar los detalles y evaluar la calidad. Se utilizan diversos tipos de diagramas UML para representarlos.

Desde la perspectiva del diseño orientado a objetos, podemos definir un componente como un bloque que contiene un conjunto de clases que colaboran entre sí, en el cuál las clases tienen todos los atributos y operaciones relevantes para su implementación.

El proceso que seguiremos acabo para diseñar el nivel de componentes será:

- A partir del modelo del análisis se elaborarán las clases de análisis y las de infraestructura, con sus principales atributos y métodos.
- Se diseñará el detalle algorítmico para la implementación de la lógica de las operaciones así como los mecanismos de las interfaces para la descripción de todos los mensajes que se requieran, lo cuál se hará mediante los diagramas de secuencia.

1.1.2. Diagrama de clases

Un **diagrama de clases** es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

Existen una serie de conceptos que tenemos que tener bastante claros. En primer lugar las **propiedades también llamados atributos o características**, son valores que corresponden a un objeto, como color, material, cantidad, ubicación. Generalmente se conoce como la información detallada del objeto. Suponiendo que el objeto es una puerta, sus propiedades serían: la marca, tamaño, color y peso. El siguiente concepto son las **operaciones comúnmente llamados métodos**, son aquellas actividades o verbos que se pueden realizar con/para este objeto, como por ejemplo abrir, cerrar, buscar, cancelar, acreditar, cargar. De la misma manera que el nombre de un atributo, el nombre de una operación se escribe con minúsculas si consta de una sola palabra. Si el nombre contiene más de una palabra, cada palabra será unida a la anterior y comenzará con una letra mayúscula, a excepción de la primera palabra que comenzará en minúscula. Por ejemplo: abrirPuerta, cerrarPuerta, buscarPuerta, etc. Otro concepto es el de **interfaz**, que es un conjunto de operaciones que permiten a un objeto comportarse de cierta manera, por lo que define los requerimientos mínimos del objeto. Hace referencia a polimorfismo. Y por último el concepto de **herencia**, que se define como la reutilización de un objeto padre ya definido para poder extender la funcionalidad en un objeto hijo. Los objetos hijos heredan todas las operaciones y/o propiedades de un objeto padre. Anotar que los atributos y los métodos pueden ser privados, públicos o protegidos, permitiendo así ser accedidos o no por objetos de su misma o de otras clases.

Debemos tener conocimiento, además, de que a la hora de diseñar se suelen utilizar **patrones de diseño**, los cuales brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Debemos tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios).

El proyecto que nos atañe se basa en el *framework rails*, que corre en el lenguaje de programación *ruby*. *Ruby on Rails* se basa en el patrón de diseño**Modelo Vista Controlador (MVC)**, que es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón de llamada y retorno MVC (según CMU), se ve frecuentemente en aplicaciones web como la que nos ocupa, donde **la vista** es la página HTML y el código que provee de datos dinámicos a la página. **El modelo** es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y **el controlador** es el responsable de recibir los eventos de entrada desde la vista.

Muchos de los sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos: en líneas generales del MVC corresponde al modelo. La unión entre capa de presentación y capa de negocio conocido en el paradigma de la Programación por capas representaría la integración entre Vista y su correspondiente Controlador de eventos y acceso a datos, MVC no pretende discriminar entre capa de negocio y capa de presentación pero si pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue *rails* es el siguiente:

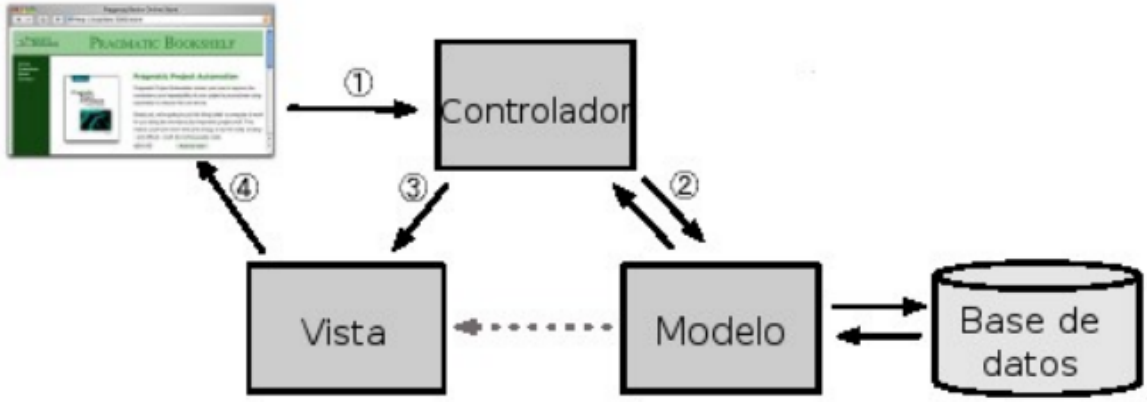

Figura 1.1: Flujo Patrón Modelo Vista Controlador

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.). Entonces, el navegador envía una petición.
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega. Entonces, el controlador accede al modelo (interactúa con el modelo), actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza los datos de un médico). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
3. Los controladores invocan a las vistas.
4. La vista *dibuja* la siguiente página para el navegador.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

A continuación el diagrama de clases (Figura 1.2).

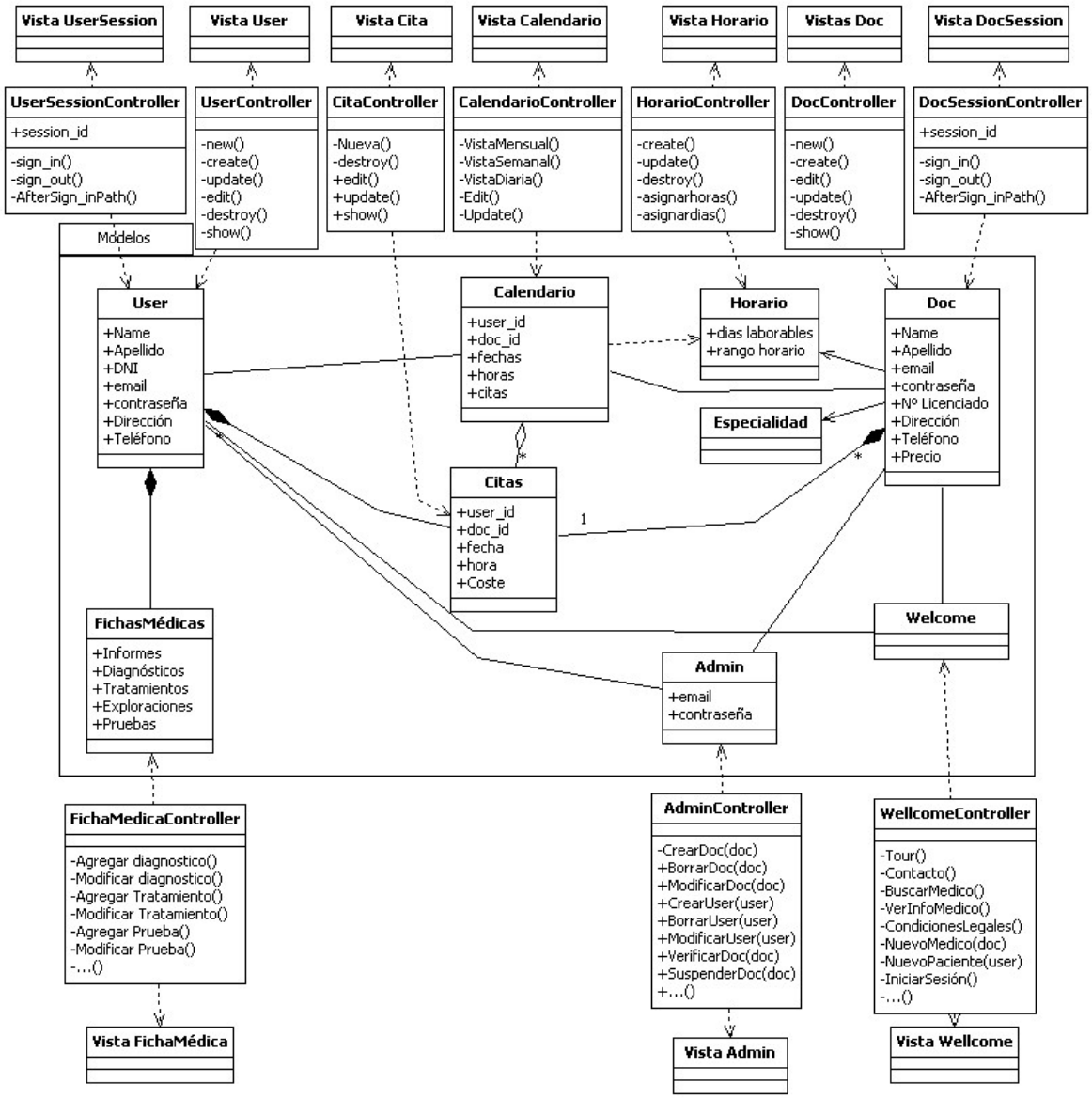

Figura 1.2: Diagrama de clases

1.1.3. Diagramas de secuencia

Cómo fue explicado en capítulos anteriores, los diagramas de interacción son diagramas que describen cómo grupos de objetos colaboran para conseguir algún fin. Estos diagramas muestran objetos, así como los mensajes que se pasan entre ellos dentro del caso de uso, es decir, capturan el comportamiento de los casos de uso.

Hay dos tipos de diagrama de interacción, ambos basados en la misma información, pero cada uno enfatizando un aspecto particular: **Diagramas de Secuencia** y Diagramas de Colaboración.

> Un *diagrama de Secuencia* muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo. Es decir, muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso.

En cuanto a la **representación**, el eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba abajo. Se pueden colocar etiquetas (como restricciones de tiempo, descripciones de acciones, etc.) bien en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren.

En esta parte del diseño, al igual que en el resto del documento, vamos a dividir los diagramas de secuencia en cinco grandes grupos. Así, se detallan dos diagramas de secuencia hacen referencia a:

- Acciones generales.
- Médicos.
- Pacientes.
- Administrador.
- Fichas médicas.

Diagramas de secuencia de las Actividades Generales.

Acceso y Autentificación Gestionado por el controlador y las vistas de *sessions* y haciendo uso del modelo *user*. Si todo va correctamente, tal y como se muestra en la Figura 1.3, un usuario introducirá su email y su contraseña en la vista. El sistema (control) creará una nueva sesión si los datos son correctos. Además, en el diagrama se aprecia que se ha accedido con el rol de médico, y por tanto el controlador de sesión redirige al usuario al panel correspondiente.

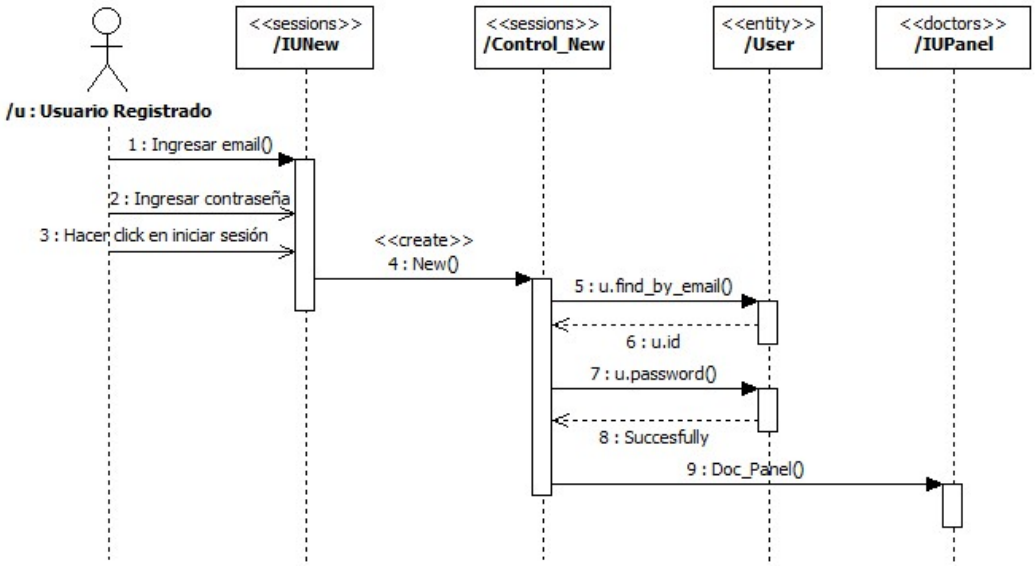

Figura 1.3: D. de secuencia del acceso y autentificación.

En el caso de que el email o la contraseña introducidos no sean correctos, o de que el usuario no exista (Figura 1.4), no se creará una nueva sesión, se renderizará de nuevo la vista de iniciar nueva sesión y se informará de que los datos introducidos no son correctos. Concretamente, en el diagrama se observa que el email introducido no existe.

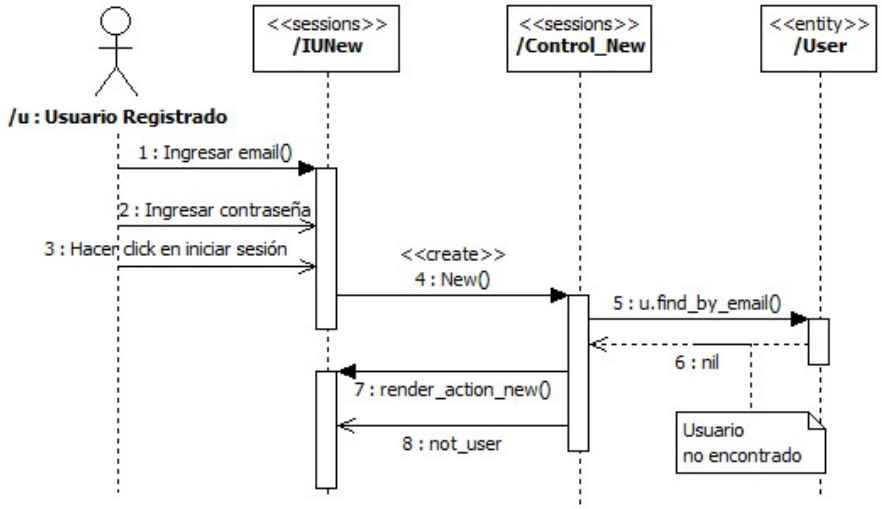

Figura 1.4: D. de secuencia de error en el acceso y autentificación.

Por último, podemos hacer que nuestro inicio de sesión sea recordado, para no tener que introducir los datos cada vez que accedemos a la aplicación desde el navegador. Para ello, tal y como se muestra en la Figura 1.5, se utilizan las *cookies* haciendo uso de un *helper de rails*.

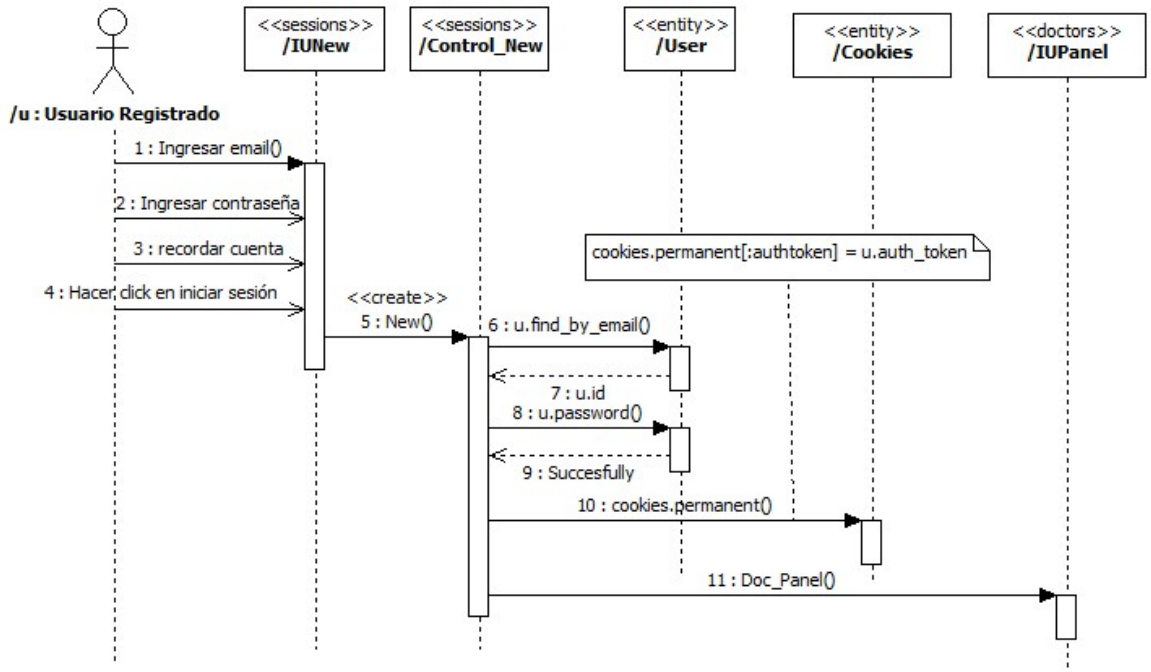

Figura 1.5: D. de secuencia de recordar acceso y autentificación.

Registro Se gestiona desde el controlador y las vistas *users* y hace uso del modelo *user*. Es el mismo tanto para médicos como para pacientes, excepto porque el valor *rol* difiere en cada caso (*doc* o *patient*). Como vemos en el diagrama de la Figura 1.6, el usuario debe introducir el email, la contraseña y la confirmación de la contraseña. Todo será validado por el sistema. Si todo es correcto, se creará una nueva instancia de usuario con el rol que le corresponda. Además, se generará un *authtoken*, que es un código único asociado a cada usuario para ser identificado en el caso de querer que se recuerde la sesión.

Si se produjese un error, bien porque el formato de email es incorrecto, las contraseñas no coinciden o el usuario ya existe, simplemente se renderizará de nuevo la vista de nuevo usuario y se informará del error que se ha producido (Figura 1.7)

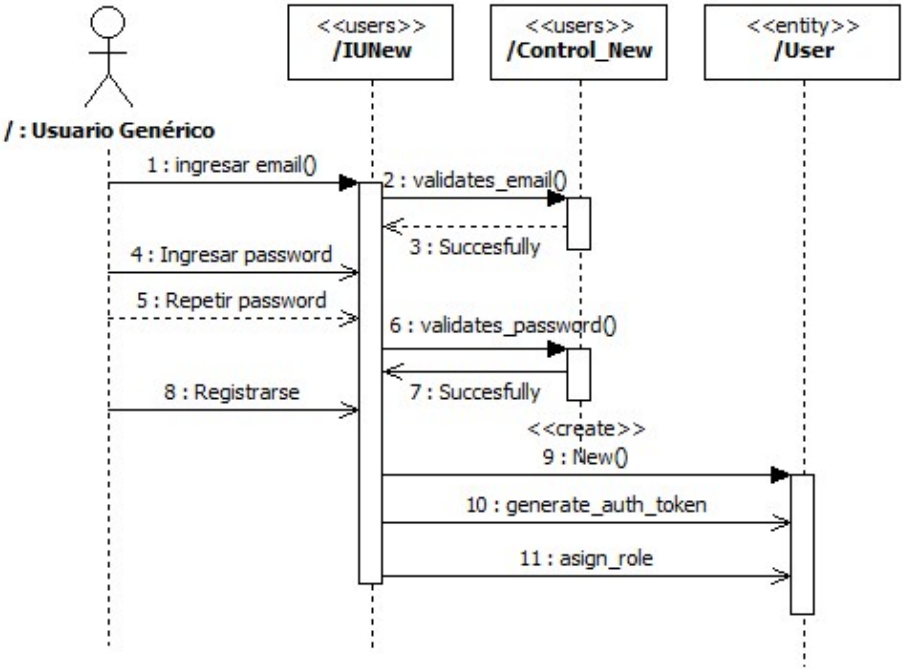

Figura 1.6: D. de secuencia de nuevo usuario.

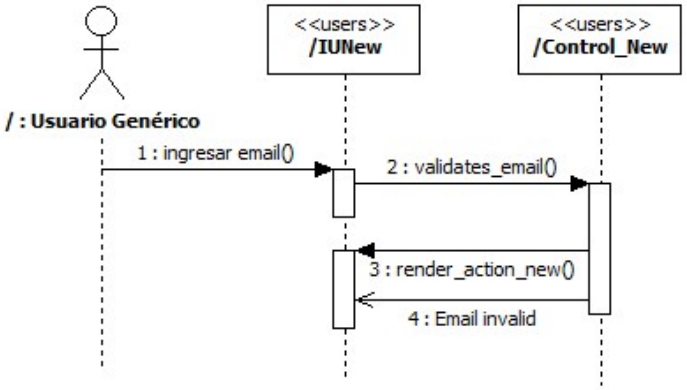

Figura 1.7: D. de secuencia de error en nuevo usuario.

Buscar Médico Utiliza el controlador y las vistas del *welcome* y el modelo *user*. Realiza una búsqueda en la base de datos en función del filtro seleccionado y los campos que se desean buscar. Finalmente se muestra una vista con la lista de los resultados al usuario (Figura 1.8). En el caso de no encontrar resultados, simplemente se mostrará la vista informando de que no existen elementos para la búsqueda realizada.

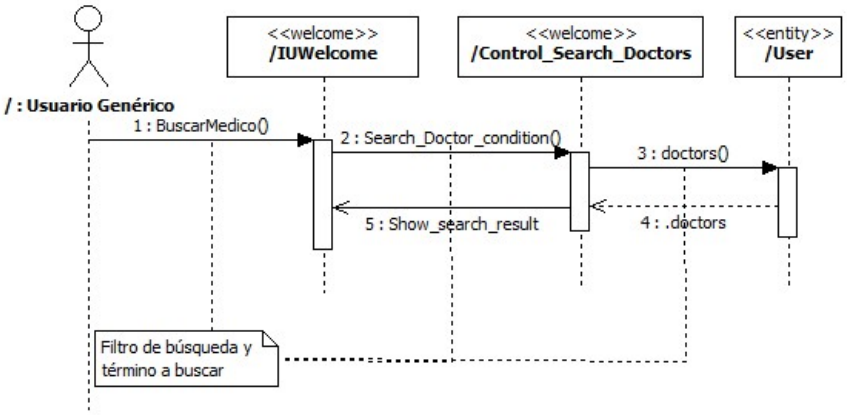

Figura 1.8: D. de secuencia de buscar médico.

Tour y resto de actividades Harán uso del controlador y vistas del *welcome*, y de las entidades del *tour, datos de contacto, términos legales, condiciones de uso*. En el diagrama de la Figura 1.9 se puede observar como el usuario selecciona la acción de *ver tour*, el controlador busca en la entidad *tour* la información y renderiza la vista correspondiente con dichos datos. El resto de acciones son exactamente igual, simplemente que se accede a la entidad correspondiente y se muestra la vista adecuada.

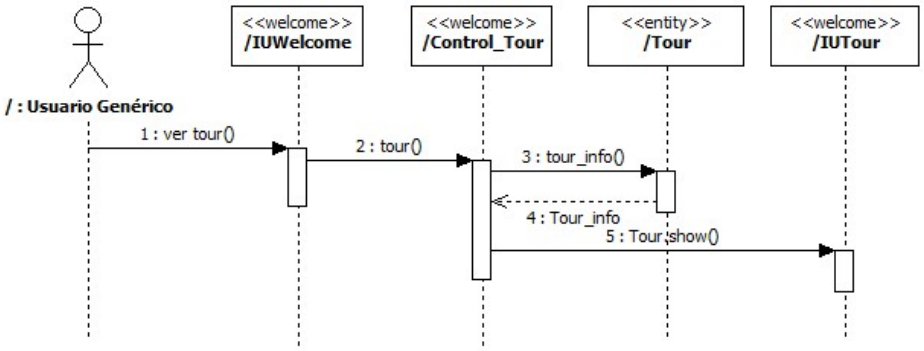

Figura 1.9: D. de secuencia ver tour.

Diagramas de secuencia de los Médicos.

Configuración del médico Utilizará las vistas y el controlador *doctors*, y el modelo *user, schedule y notification.*

En primer lugar vemos la configuración de los datos personales (Figura 1.10). En ella, el usuario introduce los datos personales, de contacto y de localización en las vistas. El controlador verificará la información, y si todo es correcto, se actualizarán los atributos del médico correspondiente y se informará de que todo ha ido correctamente. En caso de que se produzca algún error, se notificará al usuario y se renderizará de nuevo la vista de la edición de datos.

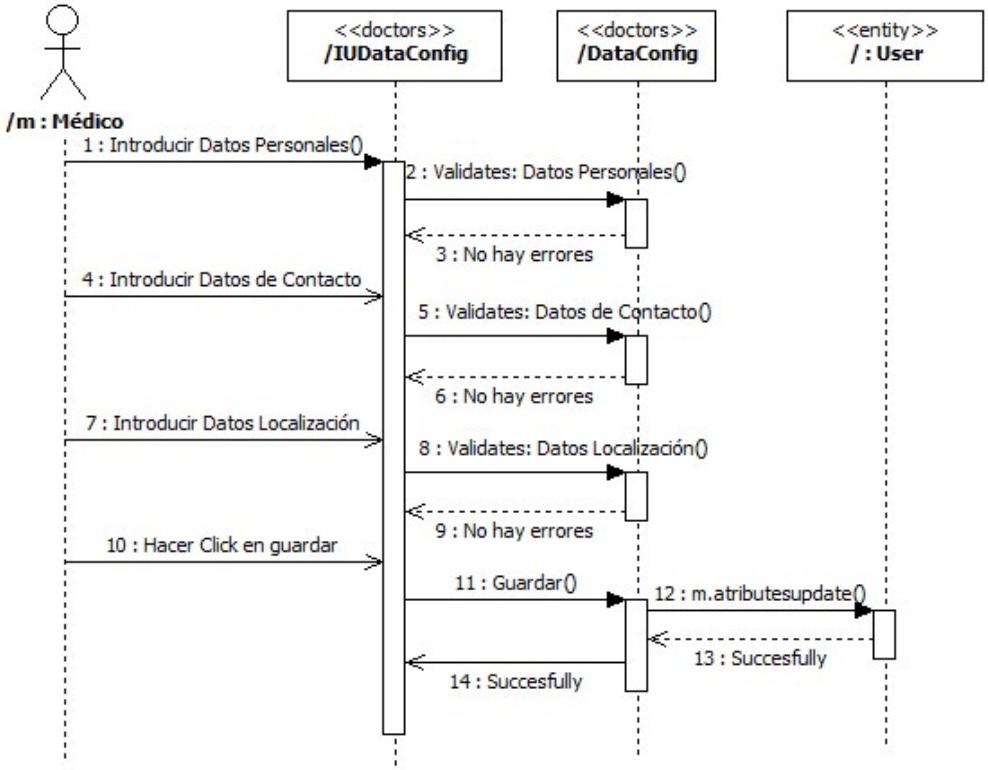

Figura 1.10: D. de secuencia de configuración de datos del médico.

La configuración de la cuenta es muy similar. En este, el diagrama de la Figura 1.11 muestra como se modifica la contraseña. Modificar el email o el idioma es igual, simplemente que no es necesario repetirlo, como ocurre con la contraseña. Si no existen errores, se actualizan los atributos del usuario y se informa. En caso contrario, se notificará y se volverá a la vista de edición de cuenta.

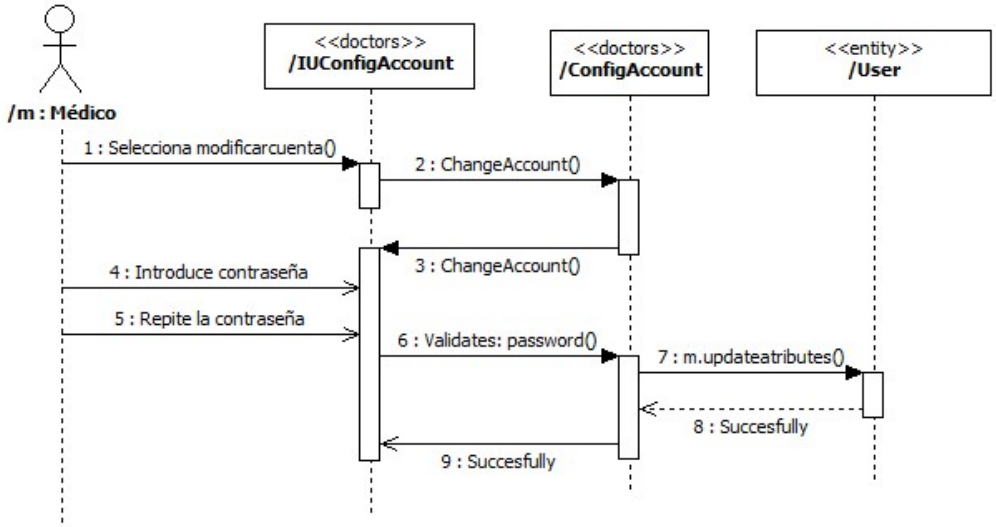

Figura 1.11: D. de secuencia de configuración de cuenta del médico.

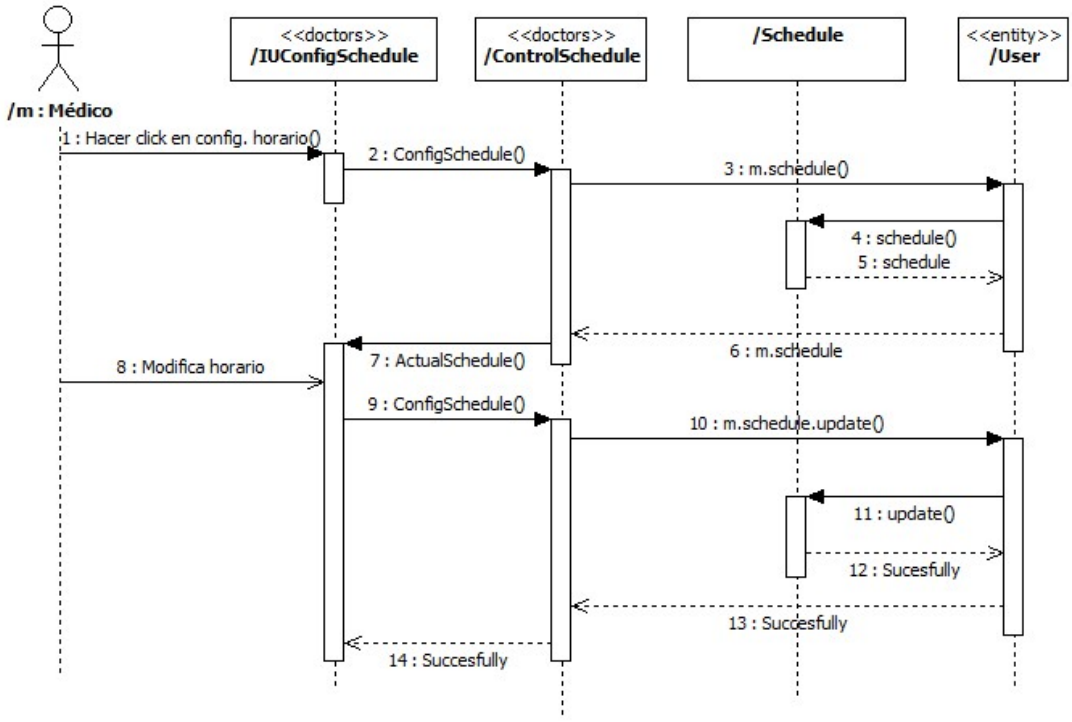

Figura 1.12: D. de secuencia de configuración de horario del médico.

En la configuración del horario (Figura 1.12), el controlador *doctors* busca en primer lugar en la tabla *users* la configuración actual, que debe buscarla a su vez en la tabla *schedules*. Una vez se tienen los datos necesarios, se muestra al usuario el formulario con la información actual para que este la modifique a su gusto. En la mayoría de los casos todo será correcto, se actualizará el médico (m) y su horario (schedule). En caso de existir cualquier error, simplemente se notificará y se renderizará la vista de modificar horario.

La configuración de las notificaciones es igual que la del horario, simplemente que la información se buscará y se actualizará en las notificaciones del médico correspondiente (m).

Calendario del médico En lo referente a los calendarios del médico, estaremos siempre trabajando con el controlador y las vistas de *doctors* y con los modelos *user, schedule y event.*

Principalmente podemos ver tres tipos de vista: diaria, semanal y mensual. En el diagrama de la Figura 1.13 vemos la vista semanal. Para mostrarla, el controlador busca en el usuario su calendario mediante su horario (Schedule) y sus citas (Event), para mostrar una vista dividida en franjas horarias en función de su horario de trabajo y que aparezcan las citas que ya tiene concertadas. Si todo ha ido correctamente, el médico verá su agenda semanal. Lo mismo ocurre con las otras funciones (diaria y mensual), cambiando la vista que se presenta al usuario.

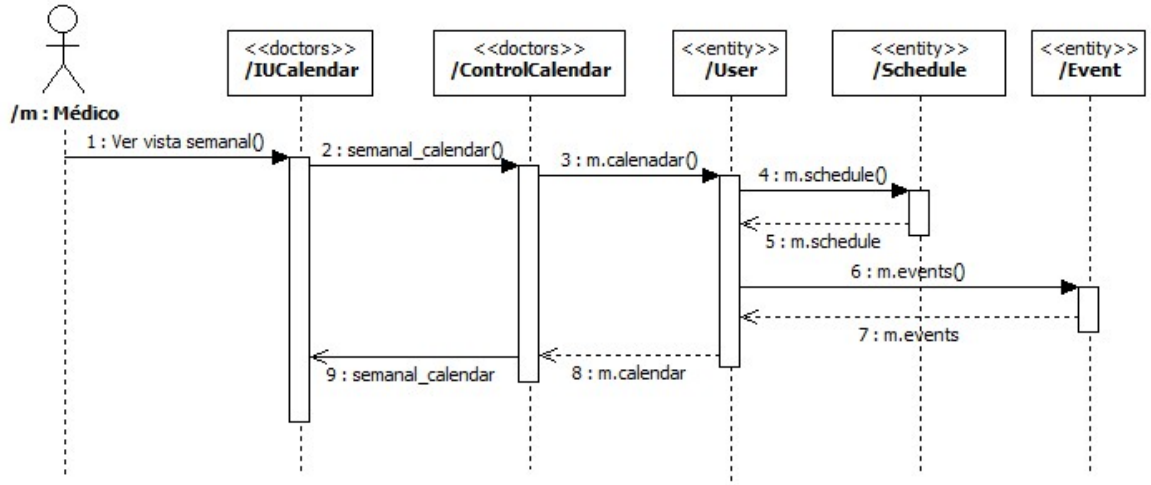

Figura 1.13: D. de secuencia de vista semanal del médico.

Otra acción importante es la de anular una cita (Figura 1.14), que se puede realizar bien desde la vista diaria, bien desde la semanal. Para ello, una vez seleccionada la cita a eliminar, se realiza un *delete* de los eventos del usuario y de la tabla de eventos.

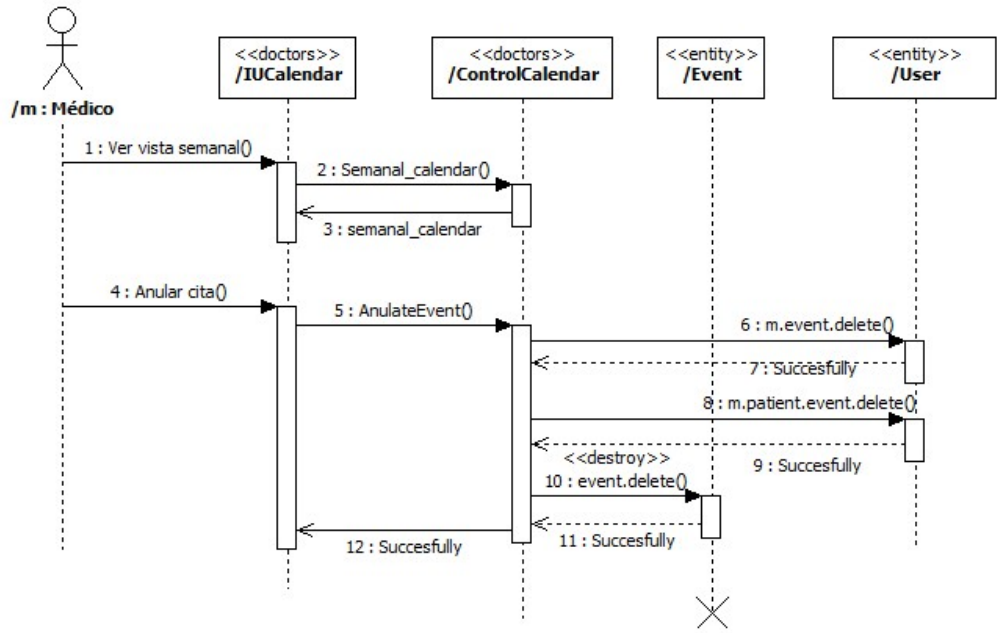

Figura 1.14: D. de secuencia de anular una cita.

Por su parte, anular un día entero es un bucle de anular las citas existentes en un día.

Pacientes del médico Hace uso de las vistas y del controlador *doctors* y de los modelos *user, event y fichamedica.*

Para ver todos los pacientes (Figura 1.15), simplemente tenemos que buscar los pacientes (*users* con rol *patient*) del médico *m*. Si quisiéramos ver los próximos o los últimos pacientes, el procedimiento sería cargar la lista de citas del médico, y mostrar las próximas o las siguientes en función del momento actual. Es decir, sería igual que el diagrama de la Figura 1.15, pero buscando en la tabla *events*.

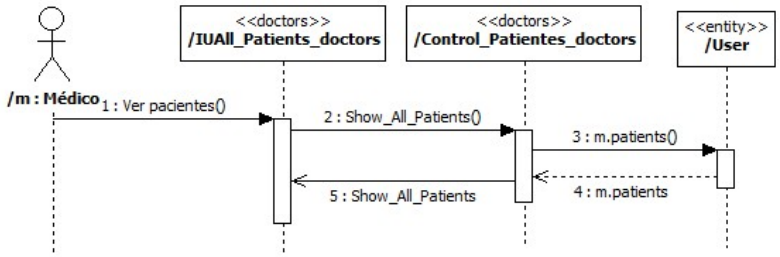

Figura 1.15: D. de secuencia de ver todos los pacientes del médico.

Para buscar un paciente (Figura 1.16), debemos aplicar un filtro de búsqueda (nombre, diagnóstico, enfermedad, etcétera) y buscar en los pacientes del médico en función de los filtros y campos introducidos. Se mostrará la vista con los resultados. En caso de no existir ninguno, se notificará al usuario en la vista.

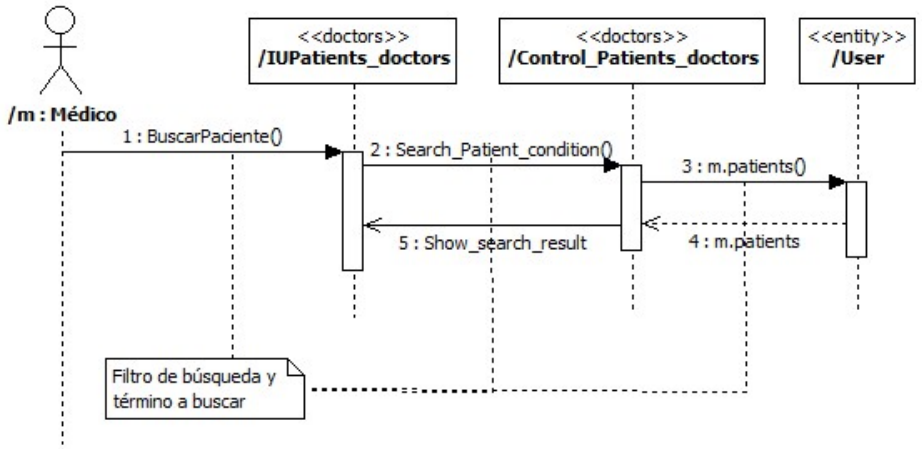

Figura 1.16: D. de secuencia de buscar pacientes del médico.

Ver ficha médica Para ver la ficha médica de un paciente, podemos hacerlo bien desde el calendario, bien desde la lista de pacientes del médico. En el diagrama de la Figura 1.17 lo hacemos después de buscar un paciente, y haciendo uso del controlador y las vistas *doctors* y los modelos *user y fichamedico*.

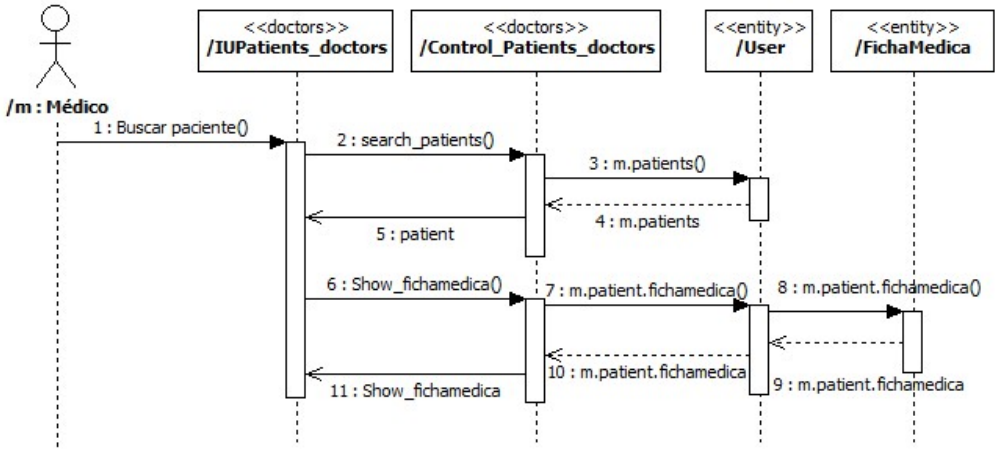

Figura 1.17: D. de secuencia de ver ficha médica.

Diagramas de secuencia de los Pacientes.

Configuración del paciente Utilizará las vistas y el controlador *patients*, y el modelo *user y notification.*

En primer lugar vemos la configuración de los datos personales (Figura 1.18). En ella, el usuario introduce los datos personales y los de contacto en las vistas. El controlador verificará la información, y si todo es correcto, se actualizarán los atributos del paciente correspondiente y se informará de que todo ha ido correctamente. En caso de que se produzca algún error, se notificará al usuario y se renderizará de nuevo la vista de la edición de datos.

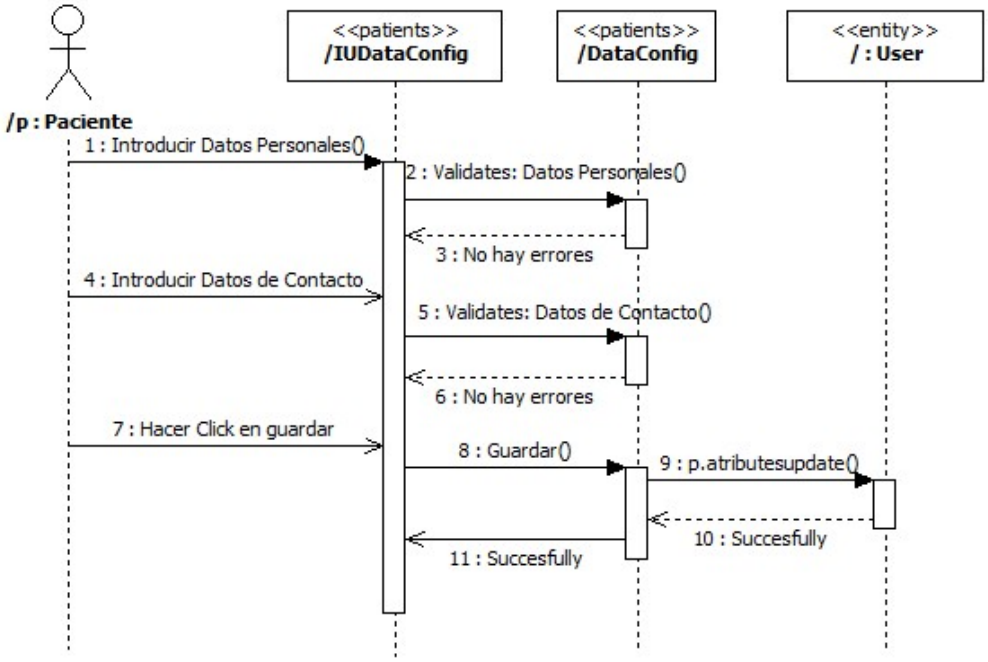

Figura 1.18: D. de secuencia de configuración de datos del paciente.

La configuración de la cuenta es muy similar. En este, el diagrama de la Figura 1.19 muestra como se modifica el idioma. Modificar el email o la contraseña es igual, simplemente que la contraseña es necesario repetirla. Si no existen errores, se actualizan los atributos del usuario y se informa. En caso contrario, se notificará y se volverá a la vista de edición de cuenta.

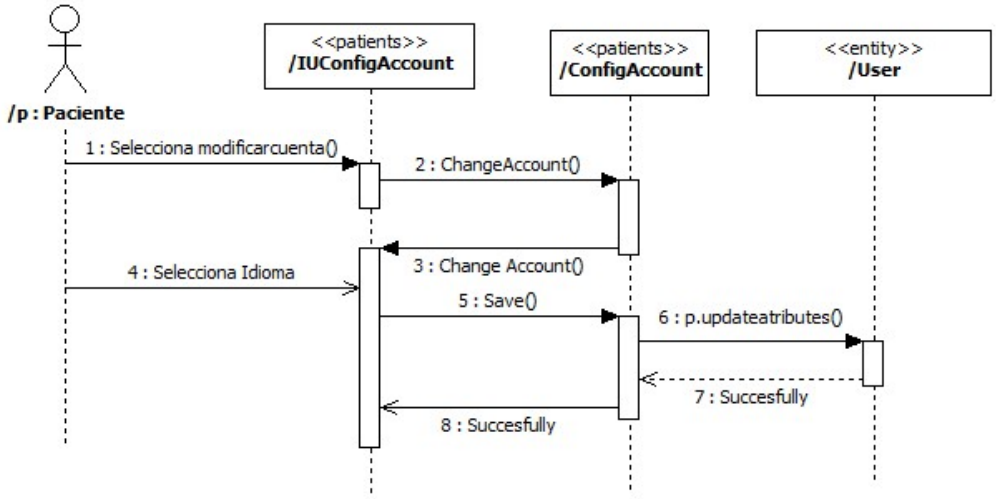

Figura 1.19: D. de secuencia de configuración de cuenta del paciente.

Calendario del paciente En lo referente a los calendarios del paciente, se hará uso del controlador y las vistas de *patients* y con los modelos *user y event*.

Principalmente podemos ver tres tipos de vista: diaria, semanal y mensual. En el diagrama de la Figura 1.20 vemos la vista semanal. Para mostrarla, el controlador busca en el usuario (User) y sus citas (Event), para mostrar una vista dividida en franjas horarias en la que aparezcan las citas que ya tiene concertadas. Si todo ha ido correctamente, el paciente verá su agenda semanal. Lo mismo ocurre con las otras funciones (diaria y mensual), cambiando la vista que se presenta al usuario.

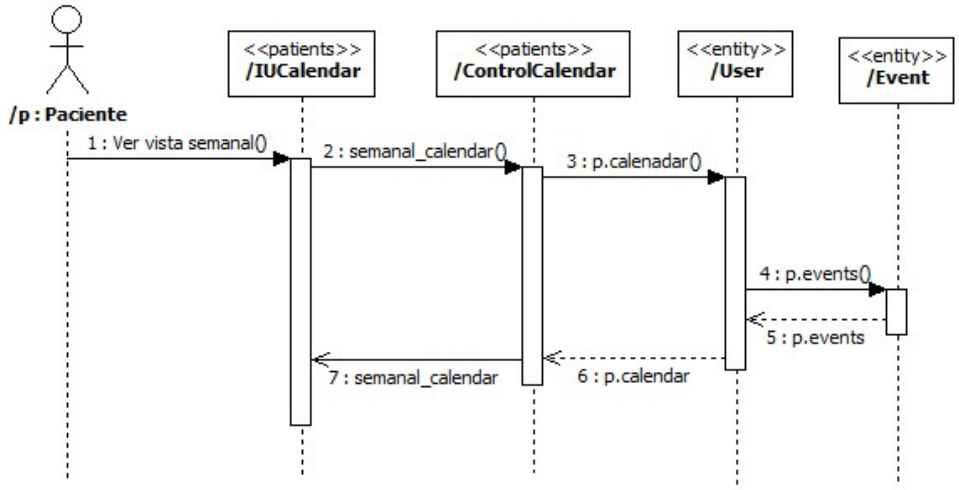

Figura 1.20: D. de secuencia de vista semanal del paciente.

Otra acción importante es la de anular una cita (Figura 1.21), que se puede realizar bien desde la vista diaria, bien desde la semanal. Para ello, una vez seleccionada la cita a eliminar, se realiza un *delete* de los eventos del usuario y de la tabla de eventos.

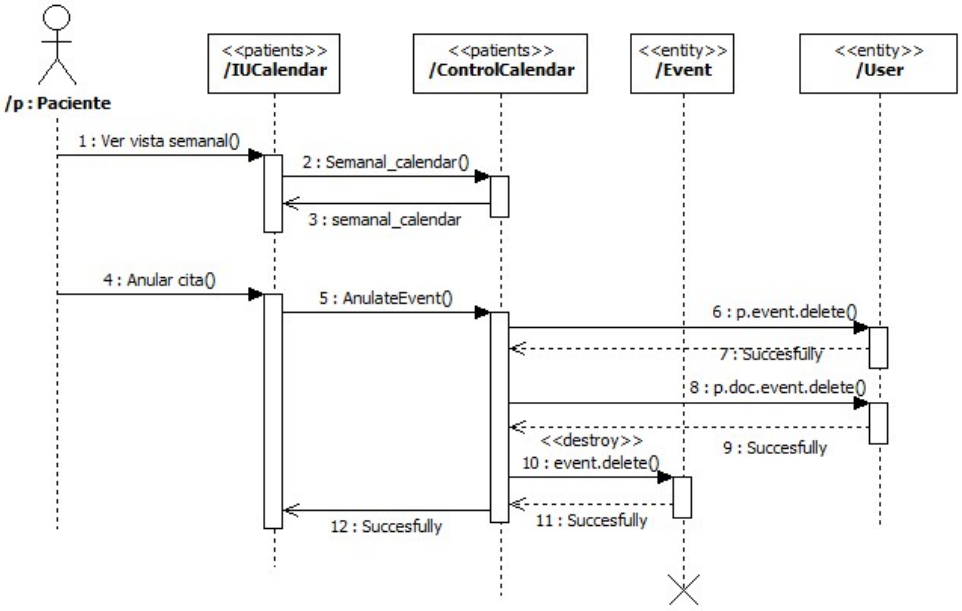

Figura 1.21: D. de secuencia de anular una cita.

Médicos del paciente Hace uso de las vistas y del controlador *patients* y de los modelos *user, event y fichamedica*.

Para ver todos los médicos (Figura 1.22), simplemente tenemos que buscar los medicos (*users* con rol *doc*) del paciente *p*. Si quisiéramos ver los próximos o los últimos médicos, el procedimiento sería cargar la lista de citas (Event) del paciente, y mostrar las próximas o las siguientes en función del momento actual. Es decir, sería igual que el diagrama de la Figura 1.22, pero buscando en la tabla *events*.

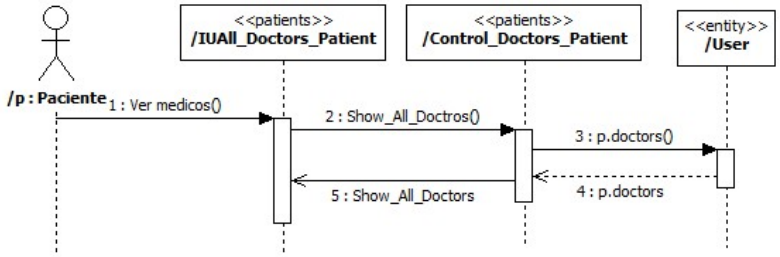

Figura 1.22: D. de secuencia de ver todos los médicos del paciente.

Para buscar un médico (Figura 1.23), debemos aplicar un filtro de búsqueda (nombre, especialidad, localidad, etcétera) y buscar en los médicos del paciente en función de los filtros y campos introducidos. Se mostrará la vista con los resultados. En caso de no existir ninguno, se notificará al usuario en la vista.

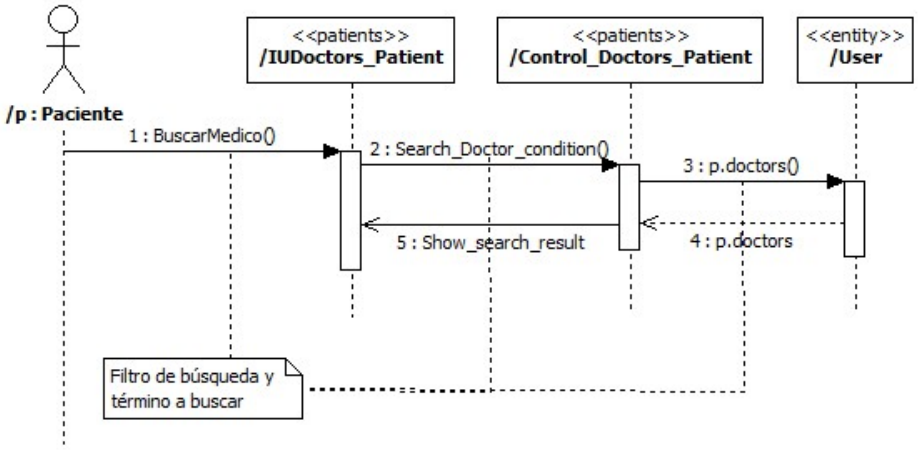

Figura 1.23: D. de secuencia de buscar médicos del paciente.

Ver ficha médica En el diagrama de la Figura 1.24 hacemos uso del controlador y las vistas *patients* y el modelo *fichamedica*.

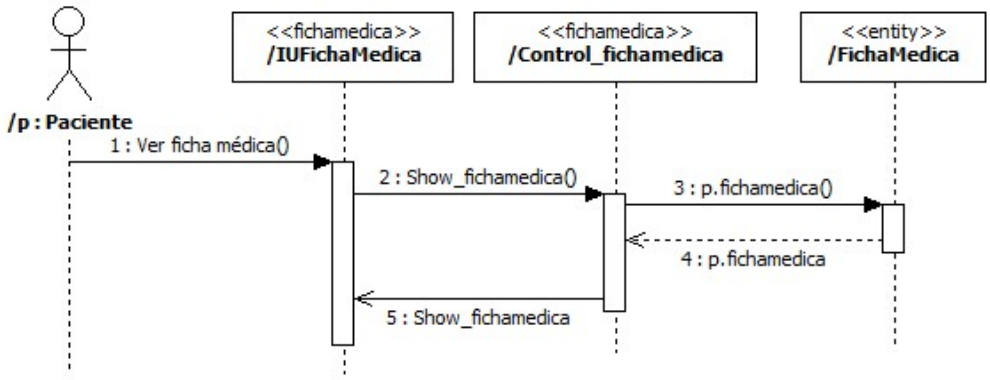

Figura 1.24: D. de secuencia de ver ficha médica.

Diagramas de secuencia del Administradores.

Respecto a los médicos Se usarán el controlador y las vistas de *admins* y los modelos *user y notification*. Todos los casos de uso (verificar, suspender, reactivar y eliminar) son iguales, excepto que en eliminar se realiza un *delete* del usuario (*User*). Se puede ver en el diagrama de la Figura 1.25 que desde la vista se indica que se verifique el médico, y el controlador pondrá el estado del campo *verificate* de dicho usuario (m) a activado. Luego se enviará la notificación correspondiente.

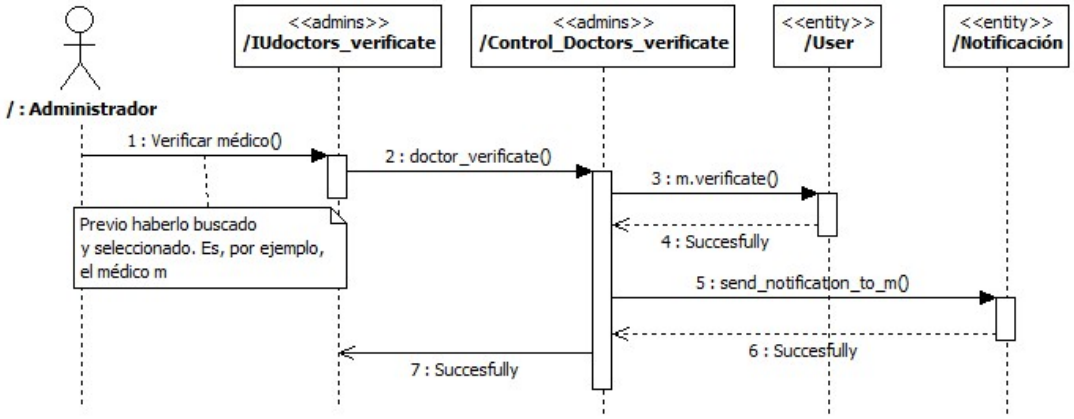

Figura 1.25: D. de secuencia de verificación de médico.

Respecto a la edición de información varia Se usarán el controlador y las vistas de *admins* y los modelos *user, conditions, terminoslegales, contacto, etcétera.*

Todos los casos de uso utilizan un diagrama de secuencia muy similar, modificando cada uno de ellos la tabla que le corresponda con la información redactada. En el diagrama de la Figura 1.26 podemos ver como se editan y actualizan las condiciones de uso.

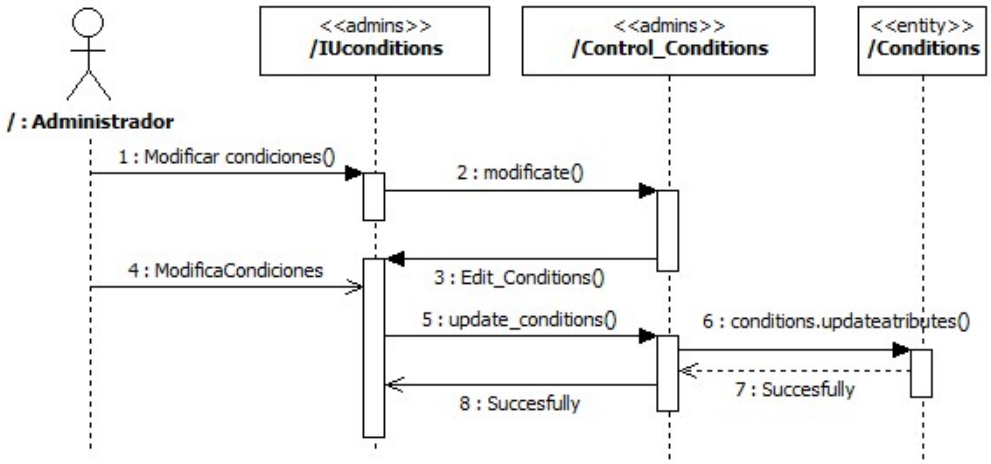

Figura 1.26: D. de secuencia de edición de las condiciones de uso.

Diagramas de secuencia de las Fichas médicas. Todos los diagramas de secuencia relacionados con las fichas médicas son muy similares. El motivo es que todos son casos C.R.U.D y sólo difieren en el tipo de objeto que se crea, se modifica, se visualiza o se elimina. Así, encontramos el controlador y las vistas *fichamedica* y los modelos *user, diagnosticos, antecedentes, tratamientos, informes, observaciones, pruebas, exploraciones y observaciones.*

Los diagramas de las Figuras 1.27, 1.28, 1.29 hacen referencia a los antecedentes, pero como se ha mencionado, serán los mismos para el resto de los apartados.

Ver un antecedente (Figura 1.27) es simplemente seleccionar en la vista el que nos interese, el controlador se encargará de buscar el antecedente del usuario y nos lo mostrará de nuevo en la vista.

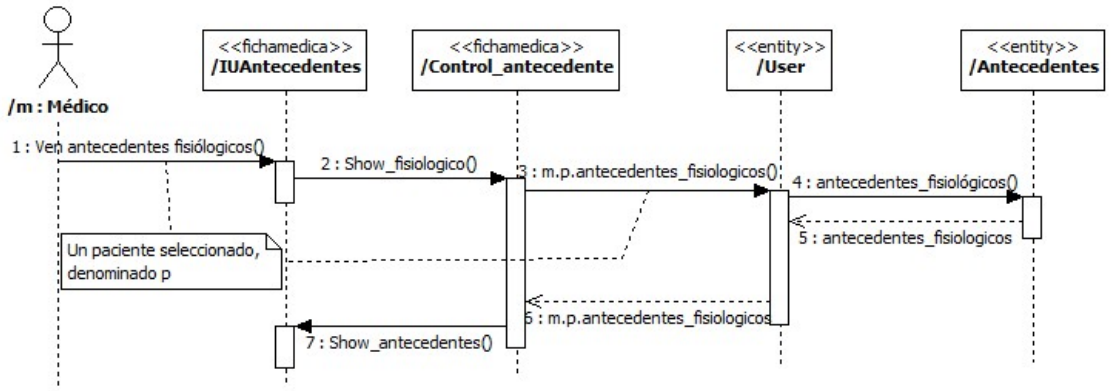

Figura 1.27: D. de secuencia de ver un antecedente.

Crear un nuevo antecedente es algo más complejo. En primer lugar se indica que se desea añadir un nuevo antecedente. A continuación, se muestra la vista correspondiente con el formulario. Una vez completo, el controlador hará un *create* del nuevo antecedente y se lo asignará al usuario actualizando su información.

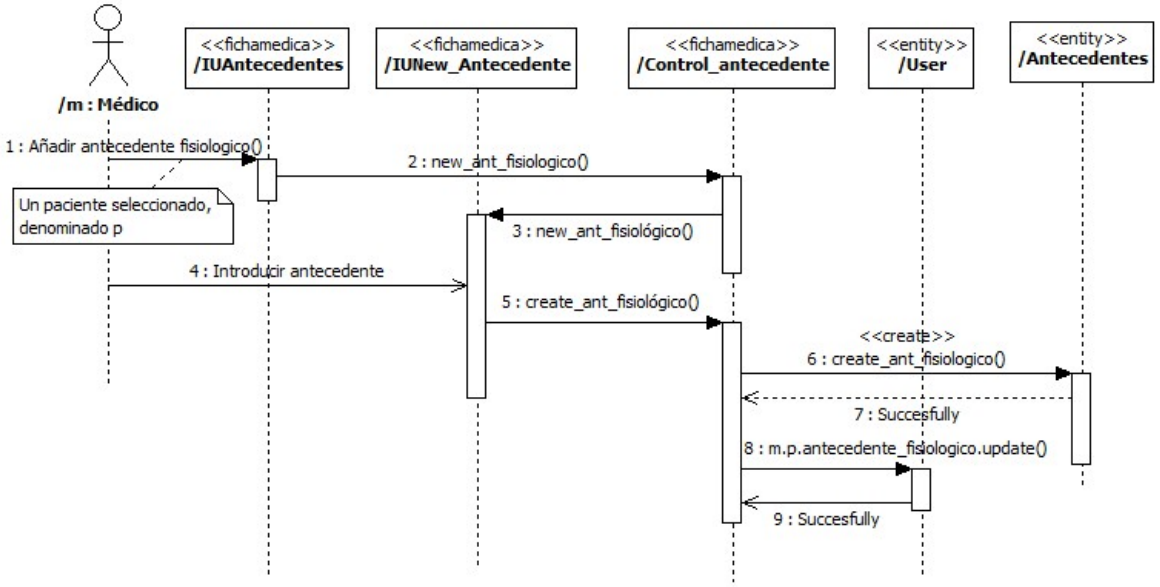

Figura 1.28: D. de secuencia para añadir un nuevo antecedente.

Por último vemos el diagrama de secuencia para modificar un antecedente. En primer lugar se selecciona el antecedente en la vista y se indica que vamos a editarlo. El controlador muestra la vista adecuada con el formulario para modificar la información. Por último, se actualizará el

antecedente y la información del usuario.

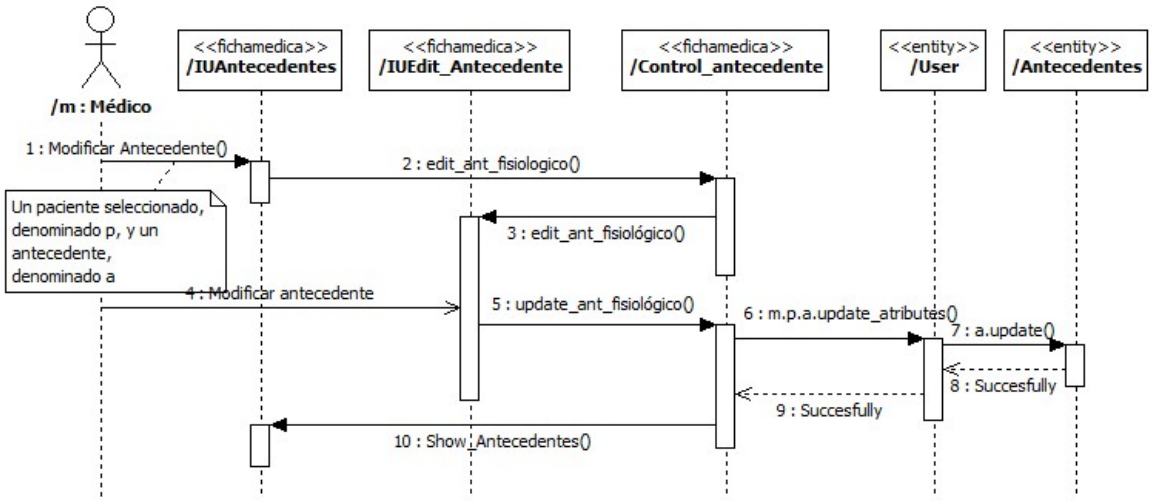

Figura 1.29: D. de secuencia para modificar un antecedente.