

Índice de contenidos

1. Especificaciones del sistema.....	2
1.1 Introducción.....	2
1.2 Descripción del sistema.....	2
1.2.1 Descripción del problema.....	2
1.2.1 Descripción de la solución.....	3
2. Análisis de requisitos.....	3
2.1 Introducción.....	3
2.2 Requisitos funcionales.....	3
2.3 Requisitos no funcionales.....	3
2.4 Historias de usuario.....	4
2.5 Actores.....	4
3. Especificación y validación de los casos de uso.....	4
3.1 Introducción.....	4
3.2 Especificación.....	4
3.3 Validación.....	6
4. Análisis estructural.....	15
4.1 Introducción.....	15
4.2 Representación.....	15
4.3 Definición.....	16
4.4 Especificaciones.....	17
5. Interacciones del sistema.....	17
5.1 Introducción.....	17
5.2 Diagramas.....	18
6. Metodología, implementación, refinamiento y pruebas finales.....	23
6.1 Introducción.....	23
6.2 Metodología.....	23
6.3 Implementación.....	24
6.4 Refinamiento.....	25
6.5 Pruebas finales.....	26

1. ESPECIFICACIONES DEL SISTEMA

1.1 Introducción

Actualmente la necesidad de software de gestión para cualquier tipo de empresa se ha vuelto una necesidad imperativa que no se puede evitar y cada vez se invierte más dinero en los mismos, bien para conseguir funcionalidades exclusivas que permitan la diferenciación con otras empresas, o bien para que la propia empresa trabaje de una forma más rápida y eficiente que la competencia.

En un principio estos programas eran considerados sencillos y sin mucho valor, pero debido a que la competencia es increíblemente grande cada vez se fuerza más a los desarrolladores a ser originales y crear software que valga la pena tanto en recursos personales como económicos. Es por eso que los softwares de gestión punteros requieren personal muy dedicado que esté constantemente renovándose y creando nuevas ideas que les permitan seguir en la cima.

Por otra parte, están las pequeñas empresas que simplemente no pueden permitirse un equipo de desarrollo que les cree un software puntero. Es precisamente por esto, por lo que existe otro tipo de mercado que es totalmente lo opuesto, satisfacer las necesidades básicas de la empresa con el menor desembolso posible. De esta forma, la persona o equipos al mando de la economía de la empresa pueden decidir qué opción tomar para satisfacer sus necesidades.

1.2 Descripción del sistema

Como la mayoría de las empresas la clínica que nos contrató quería un sistema barato y funcional que les permitiera cumplir las funciones básicas para gestionar sus datos de forma ordenada y poder organizarse de una manera un poco más eficiente.

Por su simplicidad y su precio extremadamente asequible, incluso para cualquier autónomo que acaba de empezar su negocio, este software de gestión servirá como iniciación para cualquier persona/empresa que quiera empezar en el mundo de la gestión automatizada. Únicamente deberán hacer un par de “clicks” y ya tendrán el programa listo para su funcionamiento.

El objetivo de este software es satisfacer las necesidades requeridas por la empresa y luego comercializarlo de forma que cualquiera pueda acceder a un sistema sencillo y barato para darle alas su negocio y poder despegar en el mundillo empresarial.

1.2.1 Descripción del problema

La clínica en cuestión nos pidió un sistema que le permitiera gestionar sus pacientes, tratamientos, citas y llevar un histórico de lo que va sucediendo con relación a los objetos nombrados anteriormente.

También se requirieron posibles interrelaciones entre los diferentes tipos de objetos, pero sin dejar de lado la posible independencia unos de otros en ciertas situaciones.

1.2.2 Descripción de la solución.

Una vez que analizamos los datos propuestos por el cliente decidimos dar una solución adecuada a las peticiones.

Crearemos un software sin entorno gráfico que funcione mediante la terminal propia del sistema, utilizando para su instalación la compilación de dichos archivos y el almacenamiento de los datos se hará mediante ficheros de texto o binarios modificables.

Este programa solo será utilizable en Linux por nuestro cliente, pero se podría extender a Windows a la hora de su comercialización en caso de que se diera.

2. ANÁLISIS DE REQUISITOS

2.1 Introducción

Para continuar con el desarrollo del programa vamos a proceder a dar un análisis más complejo y técnico de la situación que tenemos y las soluciones para satisfacer las necesidades anteriores.

2.2 Requisitos funcionales

En estos requisitos vamos a describir funciones utilizables en nuestro sistema para gestionar la entrada/salida de la información según desee el usuario.

- El usuario deberá ser capaz de añadir o buscar pacientes según lo necesite desde el menú principal
- Se podrán modificar, mostrar o eliminar los datos de un paciente una vez accedido al mismo.
- Las citas se gestionarán mediante una agenda en la que se podrán añadir, eliminar, modificar o consultar todas las citas registradas.
- Para asignar un tratamiento deberá ser necesario que el usuario acceda primero a un paciente en cuestión.
- Para modificar o cancelar un tratamiento deberá procederse de la forma anterior también.
- Tanto tratamientos como las citas se guardarán en un historial al que se podrá acceder desde el menú principal.

2.3 Requisitos no funcionales

- Requerimiento de sistema operativo Linux para poder utilizar el programa.
- Toda la codificación se utilizará en el lenguaje de programación C++.
- No podrán existir dos pacientes con los mismos datos, es decir no será posible la duplicación.
- Las citas no tendrán posibilidad de solaparse unas con otras
- Se requerirá confirmación antes de cada acción de añadir, borrar, eliminar o modificar en cualquiera de los campos.
- No existirá la posibilidad de borrar o modificar tratamientos o historiales a no ser que sean recientes.

2.4 Historias de usuario (requisitos)

Aquí se analizan todos los requisitos pedidos por el cliente para su posterior validación.

- El actor quiere poder buscar pacientes para consultar sus datos y operar con ellos.
- El actor quiere poder añadir pacientes para guardar sus datos e historial médico.
- El actor quiere poder modificar los datos de los pacientes para tener todo correcto y sin errores.
- El actor quiere poder dar de baja a sus pacientes eliminando todos los datos de los mismos.
- El actor quiere añadir citas al sistema y poder consultarlas cuando desee.
- El actor quiere modificar las citas (fecha/hora) y poder añadir comentarios.
- El actor quiere cancelar citas eliminando todos sus datos del sistema.
- El actor quiere añadir tratamientos asociados a sus pacientes.
- El actor quiere poder modificar el tratamiento de algún paciente en todo momento y si es necesario eliminarlo.
- El actor quiere poder consultar los tratamientos asociados a un paciente.
- El actor quiere añadir anotaciones asociadas a sus pacientes para una atención más personalizada.
- El actor quiere poder consultar y modificar el historial de notas.

2.5 Actores

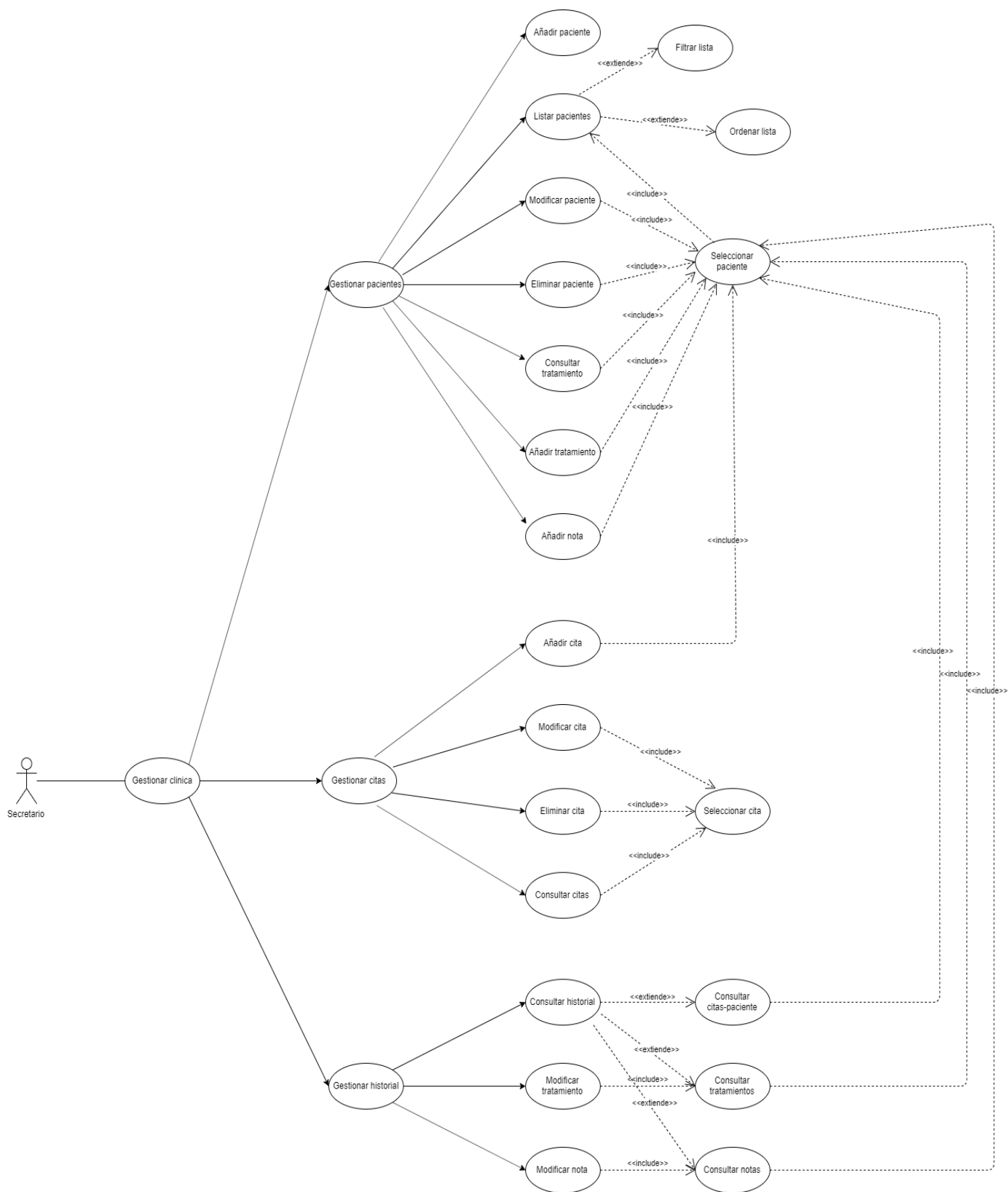
En nuestro sistema tendremos un actor principal que no requerirá de muchos conocimientos para poder manejar el menú de forma eficiente. Dicho actor será el secretario de la clínica y será el única que interactuará con el sistema. Por otra parte, habrá un actor secundario que será el administrador encargado de solucionar los problemas y actualizar el sistema, pero en ningún momento interactúa de forma directa en el sistema, por ello no se tendrá en cuenta en las representaciones.

3. ESPECIFICACIÓN Y VALIDACIÓN DE LOS CASOS DE USO

3.1 Especificación

Nombre	Sistema "Clínica médica"
ID	CU-0
Descripción	Breve explicación sobre el funcionamiento del sistema
Actores	Secretario
Casos de uso	-Buscar paciente -Añadir paciente -Modificar paciente -Eliminar paciente -Añadir cita -Modificar cita -Consultar cita -Eliminar cita -Añadir tratamiento -Modificar tratamiento -Consultar tratamiento -Añadir nota -Modificar historial -Consultar historial

Para un mejor entendimiento del funcionamiento del sistema se puede consultar el siguiente diagrama de casos de uso:



3.2 Validaciones

Nombre	Buscar paciente	
ID	CU-1	
Descripción	Se busca un paciente de la forma que elija el usuario y se muestra, dependiendo del método que se elija se muestra una lista con los pacientes.	
Actores	Secretario	
Precondición	Tener un registro con paciente	
Secuencia normal	Paso	Acción
	CU-1.1 CU-1.2 CU-1.3	-Seleccionar la opción de búsqueda. -Introducción del criterio y datos de búsqueda. -Seleccionar un paciente de la lista.
Postcondición	Se mostrará al paciente y sus opciones	
Secuencia alternativa	Paso	Acción
	CU-1.2A CU-1.2B	-Si no existen pacientes con los parámetros introducidos se muestra un mensaje de error. - Si existe más de un usuario con los parámetros introducidos se mostrará una lista con los pacientes que los cumplan ordenados por una opción por defecto.

Nombre	Añadir paciente	
ID	CU-2	
Descripción	Se agrega un paciente a la lista de pacientes de la clínica, introduciendo sus datos.	
Actores	Secretario	
Precondición		
Secuencia normal	Paso	Acción
	CU-2.1 CU-2.2 CU-2.3	-Seleccionar la opción de añadir al paciente. -Introducción y confirmación de los datos del paciente -El sistema guarda los datos del paciente.
Postcondición	Se mostrarán las opciones para el paciente	
Secuencia alternativa	Paso	Acción
	CU-2.2A CU-2.3A	-Si los datos son erróneos o no son confirmados se pedirá su reintroducción. -Si el usuario agregado ya existe, se preguntará si se desea modificar o no.

Nombre	Modificar paciente	
ID	CU-3	
Descripción	Permite modificar uno o varios datos de un paciente	
Actores	Secretario	
Precondición	Haber seleccionado un paciente	
Secuencia normal	Paso	Acción
	CU-3.1 CU-3.2 CU-3.3 CU-3.4	-Seleccionar la opción de modificar al paciente. -Seleccionar los datos a modificar. -El secretario modifica y confirma los datos. -El sistema guarda los cambios.
Postcondición	Se mostrará un mensaje de confirmación	
Secuencia alternativa	Paso	Acción
	CU-3.3A CU-3.3B	-Si los datos son erróneos o no son confirmados se pedirá su reintroducción. -Si no hay confirmación de los datos se volverán a pedir.

Nombre	Eliminar paciente	
ID	CU-4	
Descripción	Se eliminan los datos de un paciente	
Actores	Secretario	
Precondición	Haber seleccionado un paciente	
Secuencia normal	Paso	Acción
	CU-4.1 CU-4.2 CU-4.3	-Seleccionar la opción de eliminar al paciente. -El sistema pide confirmación sobre la eliminación. -El sistema elimina al paciente.
Postcondición	Se mostrará un mensaje de confirmación	
Secuencia alternativa	Paso	Acción
	CU-4.2A	-Si no se confirma no se eliminará al paciente.

Nombre	Añadir cita	
ID	CU-5	
Descripción	Se añade una cita al sistema	
Actores	Secretario	
Precondición	Haber seleccionado un paciente	
Secuencia normal	Paso	Acción
	CU-5.1 CU-5.2 CU-5.3	-Seleccionar la opción de añadir cita. - El secretario introduce la fecha de la cita, el paciente y tiene la opción de añadir un comentario a la cita. Después se pide confirmación. -El sistema guarda la cita.
Postcondición	Se mostrará un mensaje de confirmación	
Secuencia alternativa	Paso	Acción
	CU-5.2A CU-5.2B CU-5.3A CU-5.3B	-Si algún dato es inválido se pedirá su reintroducción -Si no se confirma se cancela la acción. - Se comprobará que la cita no se solape con otras ya existentes, y si lo hace se mostrará un mensaje de error y mostrará por pantalla las citas con las que se solapa -Si la cita se solapa se le dará al secretario la opción de cambiar la fecha y hora de la cita o cancelar e ir directamente al menú de citas.

Nombre	Modificar cita	
ID	CU-6	
Descripción	Modifica los datos de una cita	
Actores	Secretario	
Precondición	Existencia de citas	
Secuencia normal	Paso	Acción
	CU-6.1 CU-6.2 CU-6.3 CU-6.4	-Seleccionar la opción de modificar cita. -Seleccionar la cita. -El secretario modifica y pide confirmación de los datos. -El sistema guarda los cambios.
Postcondición	Se mostrará un mensaje de confirmación	
Secuencia alternativa	Paso	Acción
	CU-6.3A CU-6.3B	-Si los datos son erróneos o no son confirmados se pedirá su reintroducción. -Si no hay confirmación la cita no se modificará.

Nombre	Consultar citas	
ID	CU-7	
Descripción	Permite visualizar las próximas citas en rangos de 1,7,14 o 30 días o la totalidad de las citas	
Actores	Secretario	
Precondición	Existencia de citas	
Secuencia normal	Paso	Acción
	CU-7.1 CU-7.2 CU-7.3	-Seleccionar la opción de consultar citas. -El secretario selecciona el intervalo de citas que desea visualizar. -El sistema muestra por pantalla las citas.
Postcondición		
Secuencia alternativa	Paso	Acción
	CU-7.3A	-Si no hay citas en el rango seleccionado, se mostrará un mensaje de error.

Nombre	Eliminar cita	
ID	CU-8	
Descripción	Se eliminan los datos de una cita	
Actores	Secretario	
Precondición	Existencia de citas	
Secuencia normal	Paso	Acción
	CU-8.1 CU-8.2 CU-8.3	-Seleccionar la opción de eliminar cita. -El secretario selecciona la cita y se pide confirmación de eliminación. -El sistema elimina los datos de la cita.
Postcondición	Se mostrará un mensaje de confirmación	
Secuencia alternativa	Paso	Acción
	CU-8.2A	-Si no se confirma no se eliminará la cita.

Nombre	Añadir tratamiento	
ID	CU-9	
Descripción	Añade un tratamiento a un paciente	
Actores	Secretario	
Precondición	Haber seleccionado un paciente	
Secuencia normal	Paso	Acción
	CU-9.1 CU-9.2 CU-9.3	-Seleccionar la opción de añadir tratamiento. -El secretario introduce los datos del tratamiento y se pide confirmación. -El sistema guarda el tratamiento
Postcondición	Se mostrará un mensaje de confirmación	
Secuencia alternativa	Paso	Acción
	CU-9.2A CU-9.2B	-Si algún dato es inválido se pedirá su reintroducción -Si no se confirma se cancela la acción.

Nombre	Modificar tratamiento	
ID	CU-10	
Descripción	Modifica los datos de un tratamiento	
Actores	Secretario	
Precondición	Haber seleccionado un paciente. El tratamiento debe ser reciente.	
Secuencia normal	Paso	Acción
	CU-10.1 CU-10.2 CU-10.3	-Seleccionar la opción de modificar o eliminar tratamiento. -El secretario modifica y pide confirmación de los datos. -El sistema guarda los cambios.
Postcondición	Se mostrará un mensaje de confirmación	
Secuencia alternativa	Paso	Acción
	CU-10.1A CU-10.2A	-Si se elige la opción de eliminar tratamiento se pasará a la confirmación directamente. -Si no hay confirmación el tratamiento no se modificará.

Nombre	Consultar tratamiento	
ID	CU-11	
Descripción	Accede a los datos del tratamiento de un paciente.	
Actores	Secretario	
Precondición	Haber seleccionado un paciente	
Secuencia normal	Paso	Acción
	CU-10.1 CU-10.2	-Seleccionar la opción de consultar tratamiento. -El sistema muestra los tratamientos del paciente.
Postcondición	Se abren las opciones para los tratamientos	
Secuencia alternativa	Paso	Acción
	CU-10.1A CU-10.2A	-Si el paciente no está siguiendo ningún tratamiento se informará y finalizará la acción. -Si el paciente no tiene tratamientos activos no se mostrará nada.

Nombre	Añadir nota	
ID	CU-12	
Descripción	Se introduce una nota y se añade al historial	
Actores	Secretario	
Precondición	Haber seleccionado un paciente	
Secuencia normal	Paso	Acción
	CU-12.1 CU-12.2 CU-12.3	-Seleccionar la opción de añadir nota. -El secretario introduce los datos de la nota y se pide confirmación. -Se guarda la nota en el sistema
Postcondición	Se mostrará un mensaje de confirmación	
Secuencia alternativa	Paso	Acción
	CU-12.2A	-Si no se confirma se cancela la acción.

Nombre	Modificar historial	
ID	CU-13	
Descripción	Modifica los datos de un tratamiento	
Actores	Secretario	
Precondición	Haber seleccionado una nota. La nota debe ser reciente.	
Secuencia normal	Paso	Acción
	CU-13.1 CU-13.2 CU-13.3 CU-13.4	-Seleccionar la opción de modificar historial. -El secretario selecciona la nota que desea modificar. -El secretario modifica y pide confirmación de los datos. -El sistema actualiza el historial.
Postcondición	Se mostrará un mensaje de confirmación	
Secuencia alternativa	Paso	Acción
	CU-13.2A CU-13.3A	-Si no se introducen datos se advertirá que la nota se va a eliminar. -Si no hay confirmación el historial no se modificará.

Nombre	Consultar historial	
ID	CU-14	
Descripción	Accede a los datos del tratamiento de un paciente.	
Actores	Secretario	
Precondición	Haber seleccionado un paciente. Existencia de historial en ese paciente.	
Secuencia normal	Paso	Acción
	CU-14.1 CU-14.2	-Seleccionar la opción de consultar historial. -Se pide al secretario que diferencia entre tratamientos o notas. -Se muestra el historial pedido por el secretario.
Postcondición	Se abren las opciones para el historial	
Secuencia alternativa	Paso	Acción
	CU-14.2A	-Si el historial seleccionado no existe se informará al secretario y finalizará la acción.

Como se puede observar en las tablas anteriores todos los requisitos funcionales quedan perfectamente cubiertos con los siguientes casos de uso.

4. Análisis estructural

4.1 Introducción

Continuando con la documentación ahora se analizarán las clases del sistema, su representación, definición y especificación. De esta forma demostraremos la validación de los casos de uso con dichas clases.

4.2 Representación

Nombre	Paciente
Atributos	-ID_ -Nombre_ -Apellidos_ -Direccion_ -Fechanacimiento_ -Telefono_ -Codpostal_ -tipo_

Nombre	Registro
Atributos	-Fecha_ -Hora_ -Paciente_

Nombre	Cita
Atributos	-Comentario_

Nombre	Tratamiento
Atributos	-Medicamento_ -Concentración_ -Regularidad_ -Inicio_ -Final_ -Estado_ -Comentario_

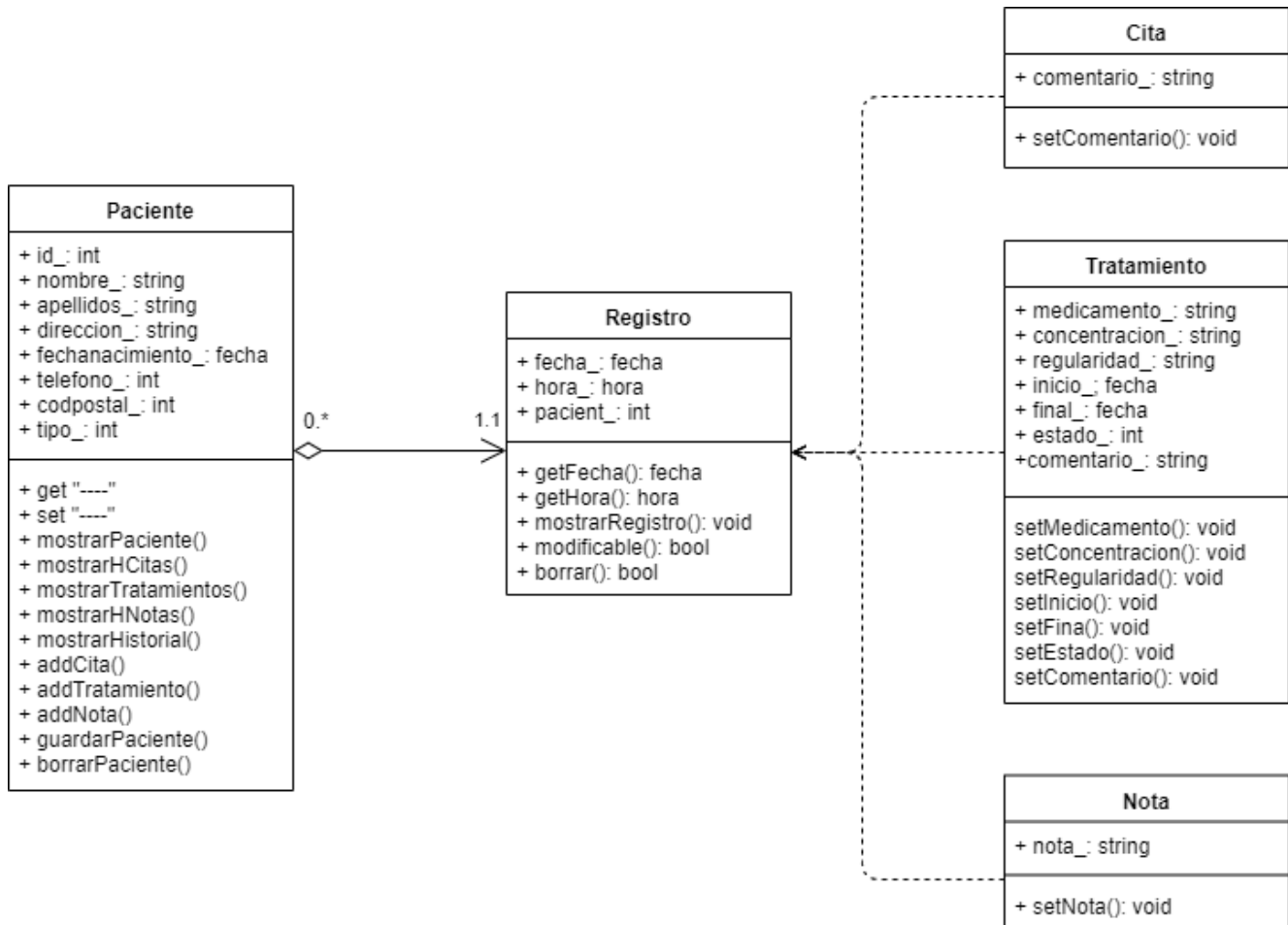
Nombre	Nota
Atributos	-Nota_

4.3 Definición

- **Paciente:** Esta clase es la columna vertebral del programa, todas las demás necesitan de su existencia. Aquí se reunirán todos los datos fundamentales de cada objeto que vaya añadiendo al programa. A partir del uso de “paciente” se podrá acceder a una gran parte de las funcionalidades del programa, por eso actuará como eje para resto de clases.
- **Registro:** La clase registro actuará como pivote para las siguientes teniendo en cuenta que “Tratamiento” y “Cita” van a heredar código de la misma, por ello aquí se guardarán variables como la fecha y la hora de las citas y algunos métodos que serán útiles en cita también. En este objeto pretendemos guardar información que sirva de interrelación entre paciente y el resto de clases como veremos en el diagrama más adelante.
- **Cita:** Esta clase hereda casi todo su código desde “registro” por lo cual se encargará de gestionar las citas que tienen los pacientes y guardar el concepto de las citas en el atributo “comentario_”.
- **Tratamiento:** En esta clase se gestionan los medicamentos, duraciones, concentraciones y demás aspectos importantes de un tratamiento, hereda código de “Registro” para poder asignar una variable que indique si el tratamiento es reciente o no, ya que anteriormente concluimos que los tratamientos solo serían modificables si eran reciente.
- **Nota:** Esta clase es la más sencilla del programa simplemente se encargará de apuntar los comentarios que el doctor quiera hacer sobre algún paciente, quedando asignada al mismo para su posterior consulta en el historial.

4.4 Especificación

A continuación, se muestra el diagrama de las clases para ver sus métodos e interrelaciones.



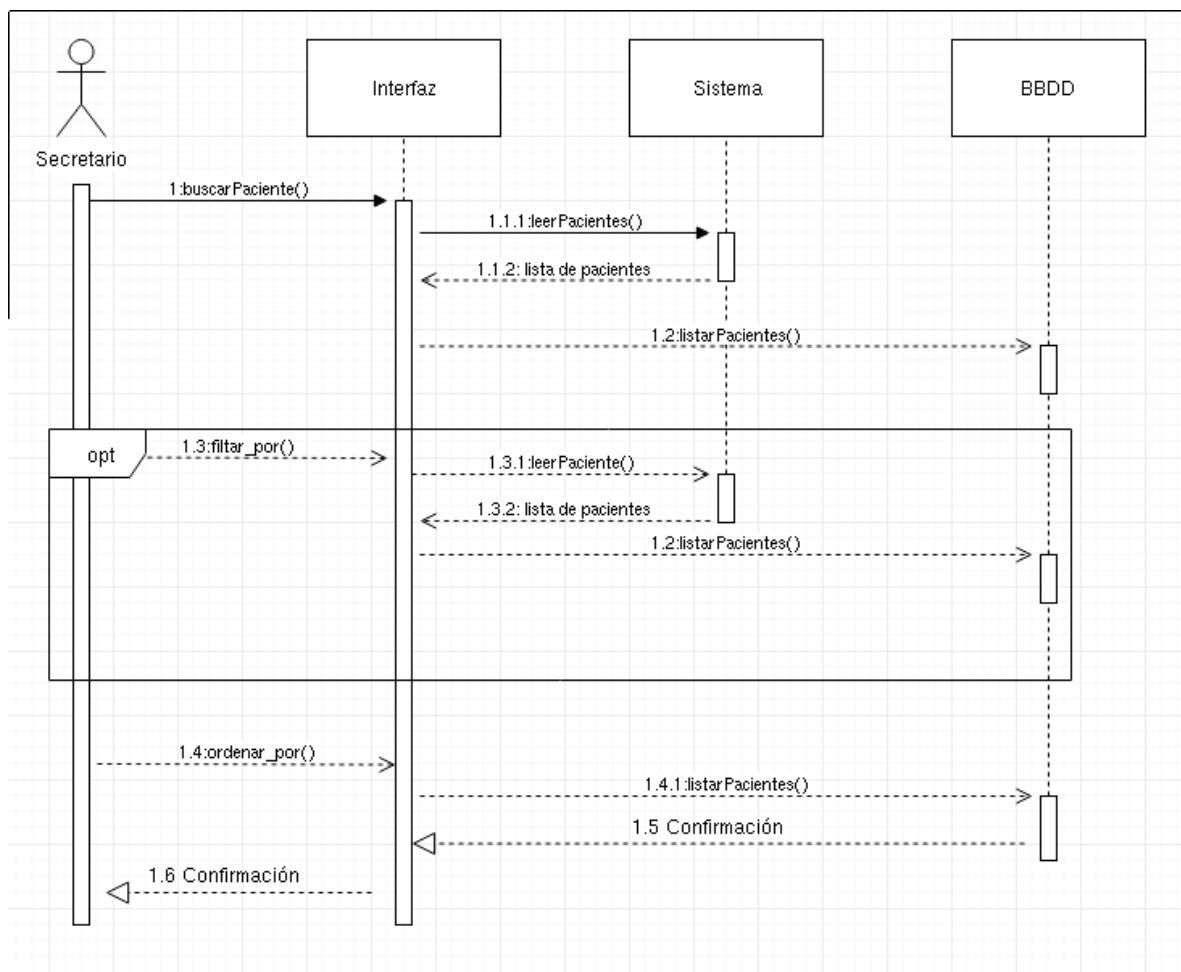
A partir de los datos mostrados anteriormente podemos satisfacer sin problemas los casos de uso en su totalidad, más adelante se mostrarán las interacciones en los casos de uso para que todo quede claro.

5. INTERACCIONES DEL SISTEMA

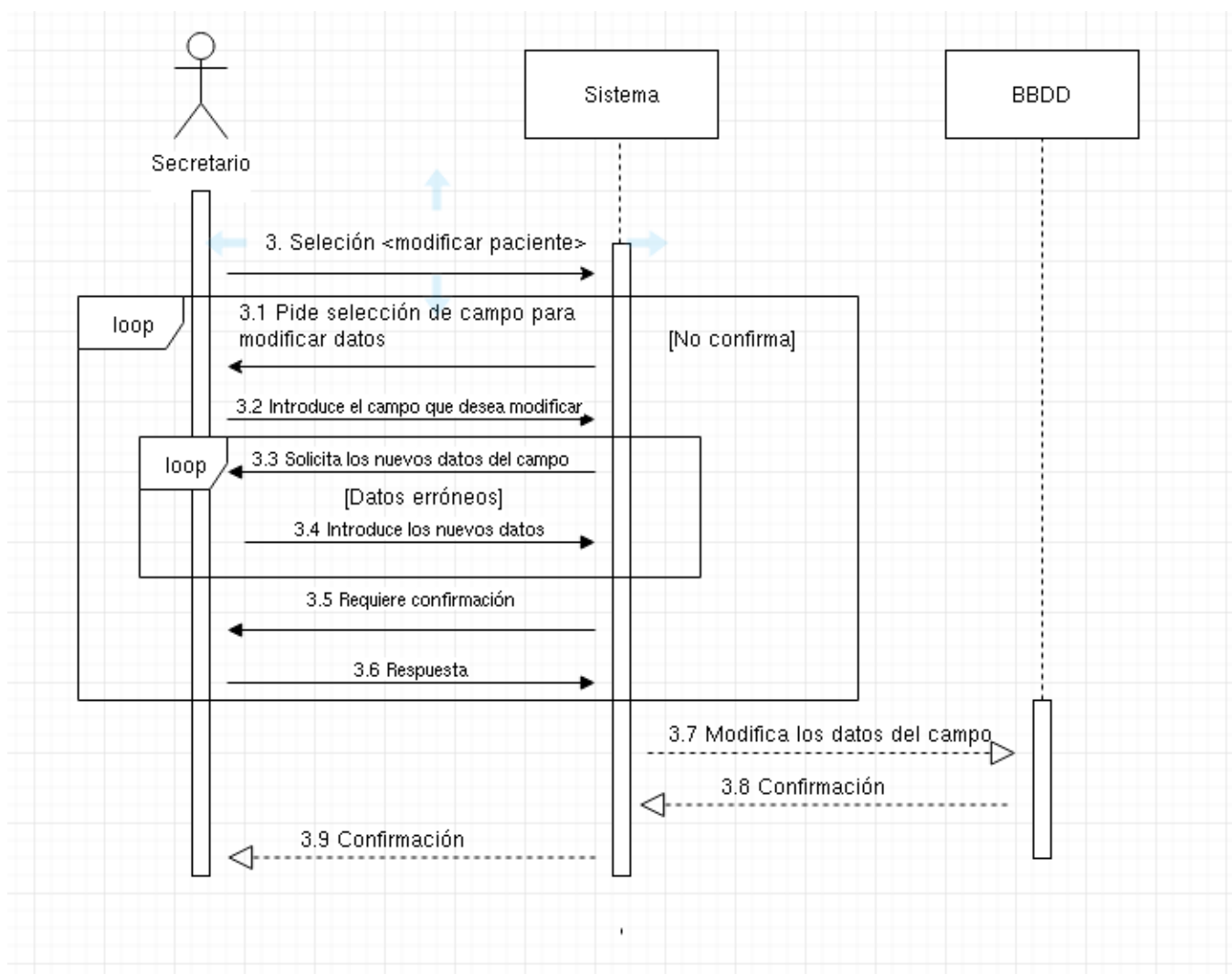
5.1 Introducción

En este apartado se expondrán los distintos escenarios de interacción que se produzcan en el sistema para llevar a cabo las funcionalidades descritas en los casos de uso. Para ello usaremos diagramas de secuencia que mostrarán el flujo del programa en los casos más complejos.

5.2 Diagramas

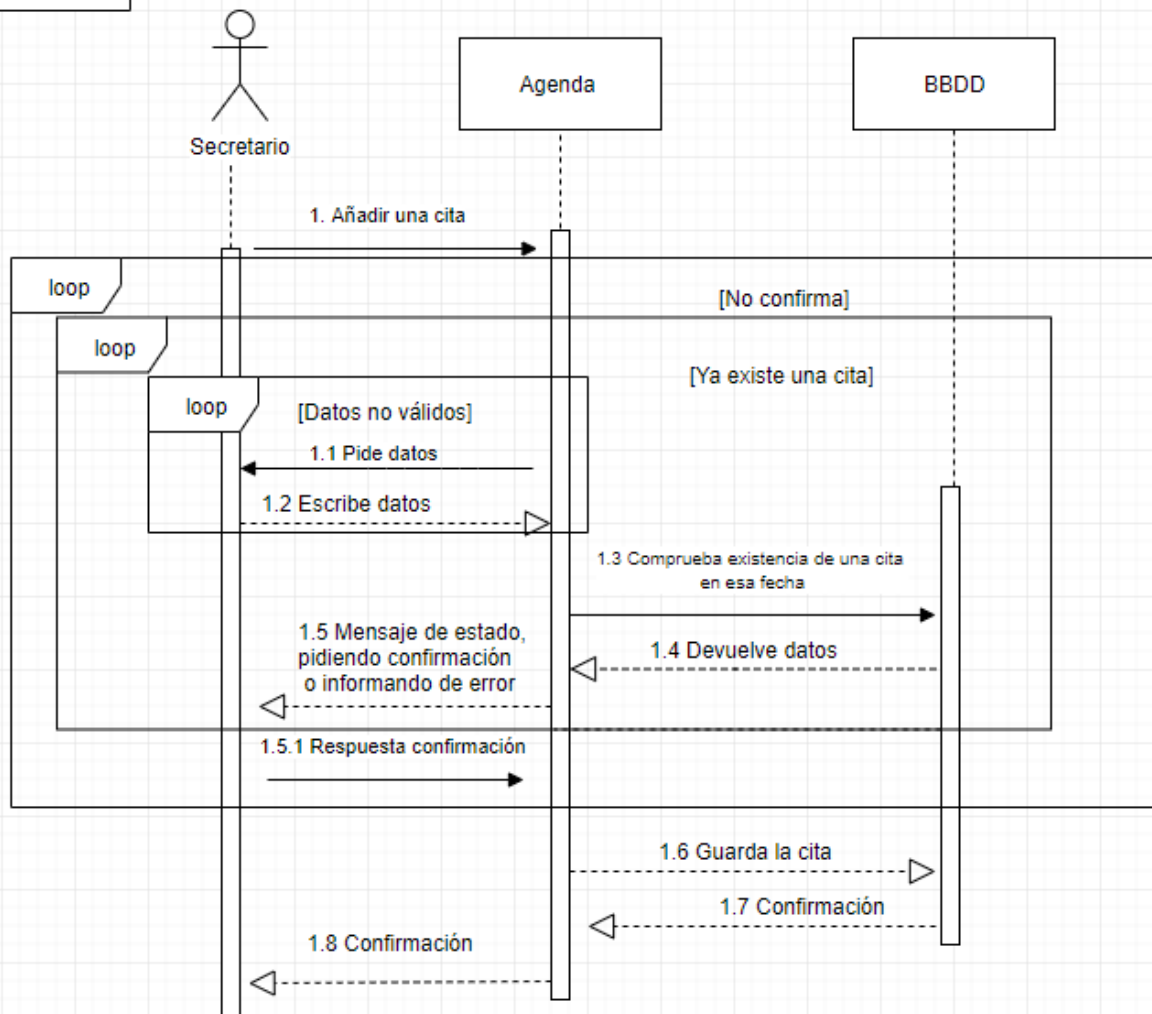


-Buscar paciente

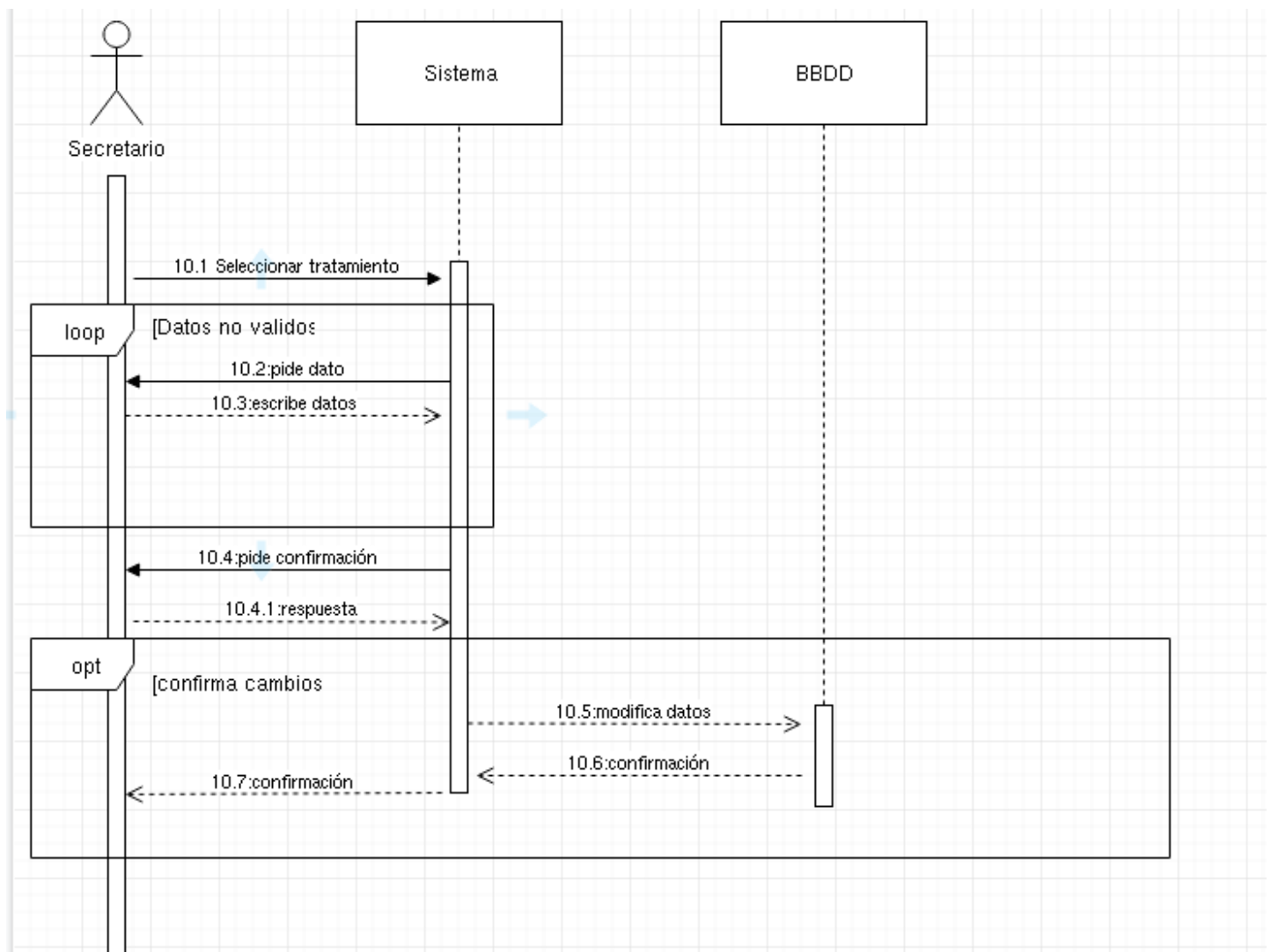


-Modificar paciente

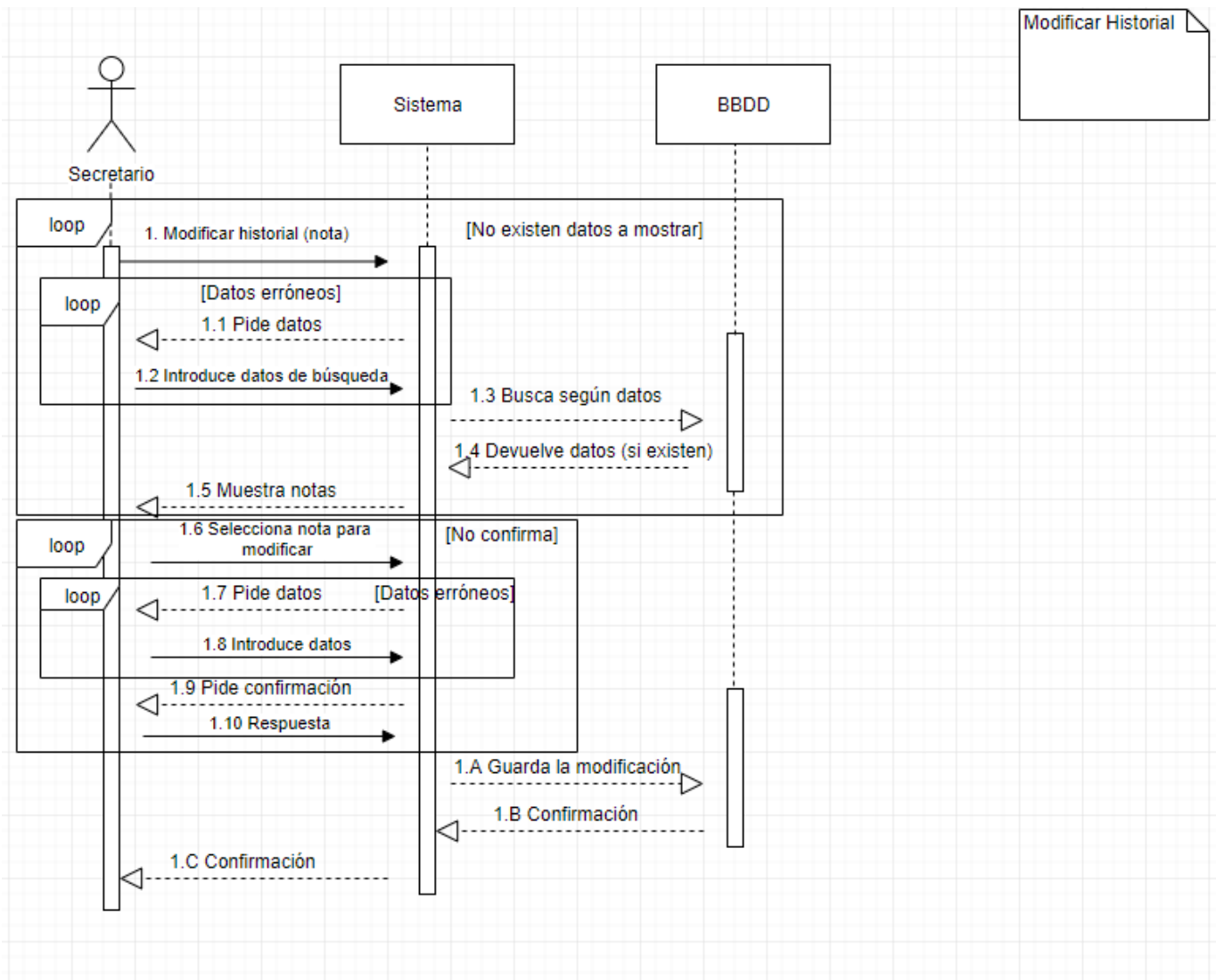
Añadir cita



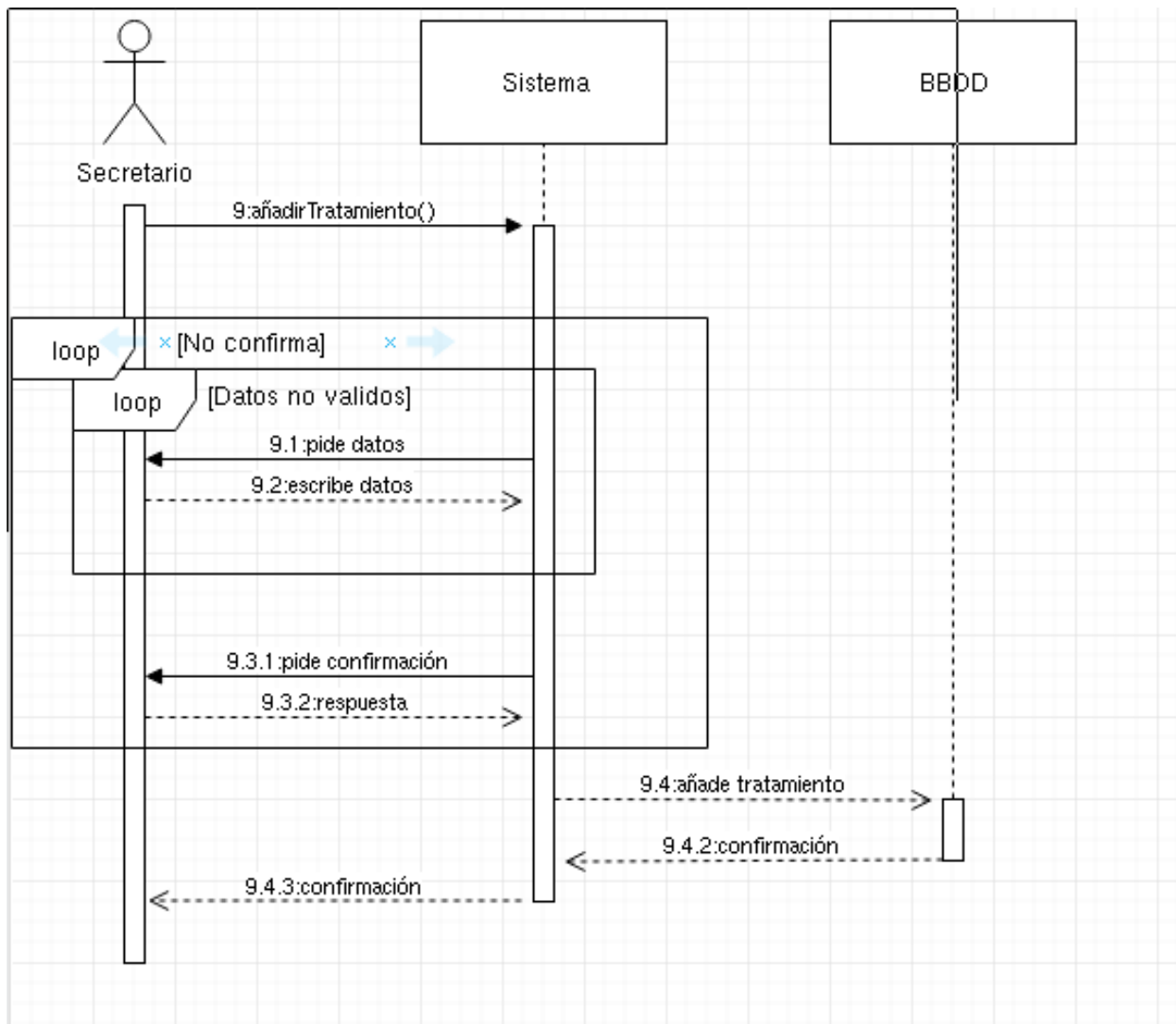
-Añadir cita



-Modificar tratamiento:



-Modificar historial:



-Añadir tratamiento:

6. Metodología, implementación, refinamiento y pruebas finales.

6.1 Introducción

Para concluir con la documentación sobre el desarrollo del software, se hablará sobre los métodos usados, problemas durante el desarrollo y un histórico sobre cómo hemos creado el programa.

6.2 Metodología

En el desarrollo de todo el software hemos utilizado la metodología SCRUM, por lo cual es conveniente que primero se haga un resumen muy breve sobre dicho método de trabajo.

En SCRUM se pretende crear unas prácticas para trabajar en equipo de la manera más óptima posible. Se suele utilizar en entornos complejos donde se necesitan resultados lo más pronto posible, donde no hay requisitos específicos, por ello, características como la innovación son fundamentales. También se utiliza SCRUM para solventar situaciones que están provocando que un proyecto no se esté desarrollando todo lo bien que debería.

SCRUM se basa en unos periodos de tiempo cortos, pero muy eficientes (normalmente de 1 a 2 semanas). Estos periodos se llaman iteraciones, y primero han de ser planificadas, luego ejecutadas guiadas por el Scrum Master, y al final, debe haber una revisión de dicha iteración en la que se presenta al cliente el trabajo realizado y luego se realiza una sesión retrospectiva para analizar el trabajo en esa iteración.

Para el desarrollo de este proyecto se ha utilizado la aplicación Taiga.io que nos permite gestionar el “product backlog” y los “sprints” realizados.



A través de esta aplicación hemos podido asignar prioridad a las tareas y ajustarlas en tres sprints, uno por semana. El Scrum Master y los miembros del team repartieron el trabajo en función de sus habilidades personales y comenzamos el trabajo siguiendo los principios de SCRUM y utilizando la aplicación anterior hasta la finalización del proyecto.

6.3 Implementación

Para la implementación como hablé anteriormente hemos utilizado el lenguaje C++ y una estructura por clases interrelacionadas.

Para comenzar la codificación del programa lo primero fue determinar y programar los métodos de la clase base que en este caso es “Paciente”. Conseguimos crear métodos eficientes para satisfacer las necesidades del cliente e incluso añadir algunas funcionales para que resulte más cómodo al usuario.

Mientras se desarrollaba y se terminaba la clase “Paciente” con sus métodos el resto del team fue desarrollando en paralelo las clases “Tratamiento” y “Registro” que empezaron a dar problemas debido a las herencias, interrelaciones en el código, entre otros problemas que luego se comentarán en el refinamiento del proyecto. Decidimos dejar “Cita” y “Nota” para el final debido a su simplicidad, eso quiere decir que dedicamos una gran parte del tiempo a las tres clases ya mencionadas anteriormente.

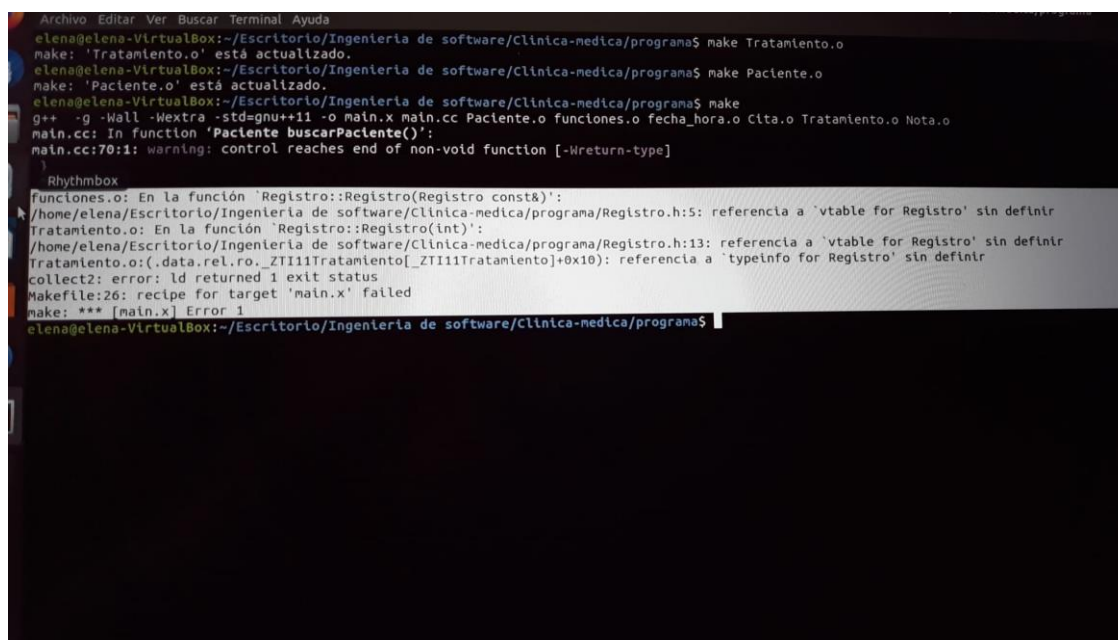
Al llevar “Tratamiento” y “Registro” a un punto avanzado del proyecto empezaron a surgir nuevas ideas para implementar a “Paciente” de forma que fuera una base más compleja para el programa y pudiera tener más funcionalidades. Las dos primeras clases mencionadas empezaron a generar conflictos debido a que, como ya se comentó en el análisis estructural, la clase “Registro” es una clase pivote, por lo cual posee funciones virtuales y variables estáticas que fueron utilizadas para facilitar la reutilización de código y el uso eficiente de la memoria.

Según iba quedando lo anterior terminado, se empezaron a desarrollar en paralelo las dos últimas clases “Cita” y “Nota” mientras que el Scrum Master empezó a desarrollar el programa principal en el que ir implementando todo lo que se había creado hasta ahora. Como se puede suponer, volvieron a surgir problemas en las relaciones entorno al programa principal. Tras un tiempo trabajando en cómo eliminar esos obstáculos para poder seguir el desarrollo, terminamos las cinco clases que componen el proyecto y ya solo quedó juntarlo en el programa principal y empezar el refinamiento de todo.

6.4 Refinamiento

Al dejar todo ya perfectamente estructurado, empezó un proceso de reparación de errores mediante un refinamiento de todas las clases. El primer impedimento que se debe comentar fue la necesidad de definir algunas funciones dentro de ficheros en los que no deberían estar por restricciones del lenguaje C++, ese error fue fácilmente solventado mediante unas cuantas redefiniciones en los ficheros fuente.

El mayor problema se presentó cuando se comenzó con la implementación de las clases “Tratamiento” y “Registro”. Como ya se ha comentado anteriormente “Registro” es una clase pivote de la que heredan otras clases, entre ellas “Tratamiento”. Es por ello que el uso de funciones virtuales complicó un poco las interrelaciones, debido a una pequeña falta de conocimiento sobre toda la complejidad de dichas funciones y todos sus posibles usos.



```
Archivo Editar Ver Buscar Terminal Ayuda
elena@elena-VirtualBox:~/Escritorio/Ingenieria de software/Clinica-medica/programa$ make Tratamiento.o
make: 'Tratamiento.o' está actualizado.
elena@elena-VirtualBox:~/Escritorio/Ingenieria de software/Clinica-medica/programa$ make Paciente.o
make: 'Paciente.o' está actualizado.
elena@elena-VirtualBox:~/Escritorio/Ingenieria de software/Clinica-medica/programa$ make
g++ -g -Wall -Wextra -std=gnu++11 -o main.x main.cc Paciente.o funciones.o fecha_hora.o Cita.o Tratamiento.o Nota.o
main.cc: In function 'Paciente buscarPaciente()':
main.cc:70:1: warning: control reaches end of non-void function [-Wreturn-type]
}
Rhythmbox
funciones.o: En la función 'Registro::Registro(Registro const&)':
/home/elena/Escritorio/Ingenieria de software/Clinica-medica/programa/Registro.h:5: referencia a 'vtable for Registro' sin defnir
Tratamiento.o: En la función 'Registro::Registro(int)':
/home/elena/Escritorio/Ingenieria de software/Clinica-medica/programa/Registro.h:13: referencia a 'vtable for Registro' sin defnir
Tratamiento.o:(.data.rel.ro._ZTI11Tratamiento[_ZTI11Tratamiento+0x10]): referencia a 'typeinfo for Registro' sin defnir
collect2: error: ld returned 1 exit status
Makefile:26: recipe for target 'main.x' failed
make: *** [main.x] Error 1
elena@elena-VirtualBox:~/Escritorio/Ingenieria de software/Clinica-medica/programa$
```

Ese problema de referencia retrasó mucho el desarrollo del programa, porque cualquier cambio debía depurarse en todo el programa de forma muy minuciosa, ya que son muchas líneas de código y deben revisarse todas para evitar que arreglar una función, cause problemas en otra. Después de muchas horas de investigación sobre funciones virtuales, comprendimos que la función estaba bien programada y por ello empezamos a buscar el error en otro lugar, al final el archivo makefile estaba defectuoso y no incluía los archivos.o necesarios para que esas referencias encontraran a los objetos a los que referenciaban.

Tras superar ese error se pudo continuar con el refinamiento del programa, dejando los mensajes de la consola bien preparados con un display adecuado, de forma no resulte molesto ni ortopédico leer los mensajes del sistema por pantalla.

Unos días más tarde cuando se procedió a pulir la clase “Cita” volvimos a tener el mismo error de referencia “vtable for “ sin definir. Igual que se consiguió arreglar la primera vez, de la misma forma se arregló para la clase cita y ya no tuvimos problemas de referencia en ninguna otra función.

Pero surgió otro inconveniente en cuanto a la manera de mostrar las citas, debido a que al mostrarlas decidimos que se mostrarán por ID, porque para poder mostrar el nombre completo del paciente sin necesidad de un atributo en la clase “Cita”, esta debería acceder a ficheros y datos de la clase paciente; así que como la estructura de nuestro programa nos demuestra que eso sería muy poco eficiente se descartó esta idea. Por otra parte, se barajó la opción de añadir un atributo nombre y apellidos a la clase “Cita” pero esto daría lugar a la duplicación datos y cambios profundos que no se pudieron realizar por falta de tiempo.

Para terminar, se tuvieron que corregir algunos errores ortográficos y problemas en el display de los datos junto a otros errores menores.

El resto del refinamiento consistió en depurar algunos avisos recomendables para reducir la carga en CPU y el uso de memoria eficiente.

6.5 Pruebas finales

En cuanto a las últimas pruebas realizadas al programa decidimos que no íbamos a utilizar Google Test, debido a que nuestro programa principalmente es un entorno gráfico que está basado en impresiones y escaneos de datos por pantalla y teclado. Así que decidimos que la mejor manera de probar el programa sería haciendo nosotros nuestra propia clínica médica. De esta forma estuvimos varias horas utilizando todas las posibles opciones del programa con diferentes parámetros muy aleatorios que pudieran desembocar en bugs o errores del sistema, hasta que no quedó ninguno.

Encontramos un problema con una de las listas que causaba el aborto del programa debido a conteo negativo en la misma, esta parte la descubrimos mientras probábamos la función de añadir tratamiento de forma manual; así como varios bucles infinitos.

Nos habría gustado realizar un testeo más intensivo del programa, pero el refinamiento y el debugging del programa nos llevó demasiado tiempo, así que no nos quedó más remedio que probarlo a la vieja usanza.

No obstante, si decidiéramos lanzar el software al mercado, estaría programada una actualización de lanzamiento para arreglar todos los posibles errores que hayan escapado a nuestras pruebas. Aunque seguramente sean mínimos siguen siendo errores y deben ser corregidos.

Tras completar las pruebas dimos por finalizado el proyecto y se entregó al cliente.

Fecha	Versión	Autores
19/12/2019	1.0	Rafael Varona Serrano Elena Ire Martín Luque Javier Roldán Ortiz