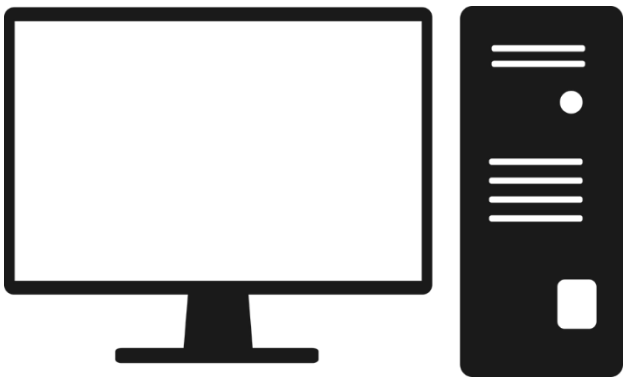
ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA
Universidad de Córdoba

EPSC

UNIVERSIDAD DE CÓRDOBA



Universidad de Córdoba

Departamento de Informática y Análisis Numérico

Área de Ingeniería del Software, Conocimiento y Bases de Datos

Memoria final de Prácticas del Grupo 11

Grupo 11

Memoria final de prácticas

Resumen del Documento

Memoria final de las prácticas llevadas a cabo durante el primer cuatrimestre en la asignatura de Ingeniería del Software, donde se detallan los pasos seguidos durante la realización de la solución ofrecida a un problema de gestión de una Clínica.

Gonzalo Reyes Criado
i72recrg@uco.es

Rafael Román de los Reyes
i62rolor@uco.es

Juan José Ropero Cerro
i82rocej@uco.es

Tabla de contenido

1. Descripción del problema y del sistema

Descripción del problema

El problema presentado consiste en la informatización de una clínica médica, dónde se tiene el inconveniente de que siempre se han usado sistemas analógicos como papel y archivadores para llevar la historia de los pacientes y el registro de estos, y el dueño quiere ahora actualizar la base de datos y la forma de organizar la clínica.

Para ello, se ha de crear un sistema informático que permita guardar un registro de pacientes y de sus historiales médicos. El programa debe permitir al administrador del sistema visualizar todos los datos de una manera sencilla y agradable a la vista a través del terminal.

Se pretende crear este sistema usando leguaje C++ y sistema de listas y ficheros de texto plano para almacenar la información. En el presente documento se detallarán los pasos seguidos y los distintos componentes presentes en el programa para poder gestionar la base de datos de pacientes y todos sus historiales además de las citas.

Descripción del sistema

En nuestro sistema se detallan tres clases principales definidas como sigue:

- **Paciente**:
  - Nombre: string.
  - Apellidos: string.
  - Fecha de Nacimiento: string
  - Domicilio: string.
  - Teléfono: int.
  - Edad: int
  - Historial: Historial
- **Cita:**
  - FechayHora: tm_struct.
  - Descripcion: string.
  - Paciente: Paciente.
- **Historial:**
  - Enfermedad: string.
  - Síntomas: string.
  - Tratamiento: string.
  - Duración tratamiento: string.

A continuación, se detallará el proceso que hemos seguido para gestionar estos datos de manera correcta.

2. Especificación de los requisitos mediante una clasificación y codificación de los mismos para una posterior validación

Durante la recogida de requisitos, obtuvimos la siguiente información:

Requisitos funcionales

1. Registrar un paciente (1).
2. Actualizar información de un paciente (2).
3. Dar de baja a un paciente (3).

4. Mostrar un paciente (1).
5. Mostrar a todos los pacientes registrados (3).
6. Crear una cita (1).
7. Modificar una cita (2).
8. Mostrar una cita (2).
9. Mostrar la lista de todas las citas (3).
10. Añadir datos al historial del paciente (1).
11. Modificar datos del historial del paciente (2).
12. Volcar datos a fichero (1).

La prioridad de los requisitos va indicada entre paréntesis, siendo 1 la prioridad más alta y 5 la prioridad más baja.

Requisitos no funcionales

1. Todos los datos de la clínica se guardan de forma local.
2. Con cada modificación, el sistema debe guardar los datos para que no se produzcan pérdidas en caso de apagado accidental.
3. No se establecen mecanismos de seguridad en la aplicación adicionales a los otorgados por el usuario o el Sistema Operativo.
4. Como se ha establecido anteriormente, la interfaz se basa en la consola de comandos.
5. La aplicación se ha implementado en C++.
6. La base de datos no está encriptada y se guarda en ficheros de texto plano.
7. La interfaz debe de ser clara y entendible pues el cliente manifiesta no tener grandes conocimientos informáticos.
8. La implementación del programa debe de ser modular.

3. Actores, casos de uso y priorización de estos

Actores

Los actores que se han tenido en cuenta a la hora de la realización de la documentación y casos de uso son los siguientes:

- Principales: Secretario / Doctor / Administrador del Sistema.
- Secundarios: Paciente.

4. Especificación de los casos de uso

Insertar paciente

ID: 001 **Descripción**: Se introducen los datos de un paciente en memoria local.

Actores principales: Administrador **Actores secundarios**: Paciente

Precondiciones:

- Ninguna

Flujo principal:

1. El administrador desea insertar los datos de un paciente
2. El administrador selecciona la opción de inserción de un paciente en el menú principal
3. El administrador introduce el nombre, apellidos, dirección, fecha de nacimiento, teléfono y tipo de cliente.
4. El sistema ejecuta la función de guardar datos (volcar los datos modificados de memoria principal a la base de datos).

Postcondiciones:

- El paciente insertado debe existir en la base de datos.

Flujos alternativos:

4.a. Si el paciente ya existe, se muestra un mensaje de error.

Modificar paciente

ID: 002 **Descripción**: Se modifica alguno de los datos del paciente.

Actores principales: Administrador **Actores secundarios**: Paciente

Precondiciones:

- Debe existir el paciente cuyos datos se van a modificar.

Flujo principal:

1. El administrador desea modificar los datos de un paciente
2. El administrador selecciona la opción de modificar paciente en el menú principal
3. El administrador selecciona el campo o campos que quiere modificar.
4. El sistema pide permisos para la modificación.
5. El administrador modifica dichos campos.
6. El sistema actualiza los datos modificados en la base de datos.

Postcondiciones:

- Los datos modificados están actualizados en la base de datos.

Flujos alternativos:

4.a. Si el paciente no existe, se da la opción de insertarlo.

Eliminar paciente

ID: 003 **Descripción**: Se borran los datos relativos a un paciente. **Actores principales**: Administrador **Actores secundarios**: Paciente

Precondiciones:

- El paciente debe existir en la base de datos.

Flujo principal:

1. El administrador desea borrar los datos de un paciente
2. El administrador selecciona la opción de borrar paciente en el menú principal
3. El sistema pide permisos para borrar los datos al administrador
4. El sistema actualiza la base de datos borrando al paciente.

Postcondiciones:

- El paciente no existe en la base de datos.

Flujos alternativos:

4.a. Si el paciente no existe, se muestra un mensaje de error.

Buscar paciente

ID: 004 **Descripción**: Se busca un paciente y se muestran por pantalla sus datos.

Actores principales: Administrador **Actores secundarios**: Paciente

Precondiciones:

- Debe existir algún paciente en la base de datos.

Flujo principal:

1. El administrador desea buscar los datos de un paciente.
2. El administrador selecciona la opción de buscar paciente el menú principal.
3. El administrador busca el paciente por nombre y apellidos.

Postcondiciones:

- El sistema muestra por pantalla los datos del paciente.

Flujos alternativos:

4.a. Si no existe el usuario, se muestra un mensaje de error

Mostrar lista de pacientes

ID: 05 **Descripción**: El sistema muestra todos los pacientes de la base de datos.

Actores principales: Usuario **Actores secundarios**: Pacientes

Precondiciones:

- Necesario que exista 1 Paciente como mínimo

Flujo principal:

1. El usuario desea consultar los datos de todos pacientes.
2. El usuario inicia el programa y solicita la vision de todos los pacientes.
3. El sistema muestra por pantalla los datos de todos los usuario en la base de datos.

Postcondiciones:

- Se muestran al usuario todos los pacientes y sus datos.

Flujos alternativos:

4.a. Si no existe ningun paciente, el sistema no muestra ningun paciente.

Crear cita

ID: 06 **Descripción**: El sistema creara una cita para un paciente

Actores principales: Usuario **Actores secundarios**: Paciente

Precondiciones:

- Ninguna

Flujo principal:

1. El usuario desea crear una cita para un paciente.
2. El usuario inicia el programa y solicita crear una cita.
3. El usuario introduce todos los datos del paciente.
4. El sistema confirma la creacion de la cita y la muestra por pantalla.

Postcondiciones:

- Se confirma la cita creada y se la muestra al usuario

Flujos alternativos:

6.a. Si los datos no se introducen correctamente, el sistema muestra un mensaje de error

Modificar cita

ID: 07 **Descripción**: El sistema modifica la cita del paciente deseado.

Actores principales: Usuario **Actores secundarios**: Paciente

Precondiciones:

- Debe existir la cita del paciente que se desea modificar.

Flujo principal:

1. El usuario desea modificar una cita para un paciente.
2. El usuario inicia el programa y solicita modificar una cita
3. El usuario introduce el nombre y apellidos del paciente que busca modificar la cita.
4. El usuario modifica los datos deseados de la cita del paciente.
5. El sistema confirma la modificacion de la cita y la muestra por pantalla.

Postcondiciones:

- Se confirma la cita modificada y se la muestra al usuario.

Flujos alternativos:

7.a. Si la cita a modificar no existe, el sistema muestra un mensaje de error

Mostrar cita

ID: 08 **Descripción**: El sistema muestra la cita de un paciente

Actores principales: Usuario. **Actores secundarios**: Paciente.

Precondiciones:

- Necesario que exista el paciente del cual se desea mirar su cita.

Flujo principal:

1. El usuario desea consultar la cita de un paciente.
2. El usuario inicia el programa y solicita una vision de la cita de un paciente.
3. El usuario introduce el nombre y apellido del paciente del cual desea ver su cita.
4. El sistema muestra por pantalla los datos del paciente y su cita prevista.

Postcondiciones:

- Se muestran al usuario la cita del paciente deseada.

Flujos alternativos:

8.a. Si no existe el paciente, el sistema muestra un mensaje de error.

Mostrar lista de citas

ID: 009

Descripción: Muestra la lista de citas que tiene el profesional en la semana pedida.

Actores principales: Usuario.

Actores secundarios: Paciente.

Precondiciones:

- Debe haber como mínimo una (1) cita en la base de datos.

Flujo principal:

1. El sistema solicita la semana sobre la que se quiere hacer la consulta.
2. El usuario introduce el número del día que corresponde al lunes de esa semana.
3. El sistema busca y recopila todas las citas de la semana pedida.
4. El sistema muestra las citas por pantalla en orden cronológico.

Postcondiciones:

- El sistema muestra las citas correspondientes a esa semana ordenadas cronológicamente y en una línea por cada cita.

Flujos alternativos:

- 3.a. Si no hay ninguna cita ese día, se muestra una advertencia sobre ello.

Añadir datos al historial de paciente

ID: 010.

Descripción: Permite añadir datos adicionales al historial médico del paciente.

Actores principales: Usuario.

Actores secundarios: Paciente.

Precondiciones:

- El paciente debe existir en la base de datos.

Flujo principal:

1. El usuario especifica sobre qué paciente quiere añadir datos al historial.
2. El sistema busca dicho paciente en el historial.
3. El sistema muestra por pantalla los resultados de la búsqueda.
4. El usuario confirma sobre qué paciente quiere realizar la adición.
5. El sistema le pregunta qué quiere añadir.
6. El usuario realiza las adiciones pertinentes.
7. El sistema guarda los datos.

Postcondiciones:

- La Postcondición que sea necesaria.

Flujos alternativos:

- 2.a. El paciente no se encuentra.

Modificar datos del historial de paciente

ID: 011.

Descripción: Permite al usuario modificar los datos del historial médico del paciente.

Actores principales: Usuario.

Actores secundarios: Paciente.

Precondiciones:

- El paciente debe existir en la base de datos.

Flujo principal:

1. El usuario especifica sobre qué paciente quiere añadir datos al historial.
2. El sistema busca dicho paciente en el historial.
3. El sistema muestra por pantalla los resultados de la búsqueda.
4. El usuario confirma sobre qué paciente quiere realizar la modificación.
5. El sistema le pregunta qué quiere modificar.
6. El usuario realiza las modificaciones pertinentes.
7. El sistema guarda los datos.

Postcondiciones:

- Los datos quedarán modificados.

Flujos alternativos:

002.a. Si no se encuentra el paciente, se muestra una advertencia. 006.a. Si las modificaciones no se corresponden con lo que se quería modificar, se muestra un error.

Volcar datos a fichero

ID: 012.

Descripción: Volcado de datos desde la memoria principal al archivo de la base de datos.

Actores principales: Usuario.

Actores secundarios: *No hay.*

Precondiciones:

- Debe haberse realizado al menos un (1) cambio.
- Debe existir la base de datos.

Flujo principal:

1. El sistema lee todo lo que se ha cambiado.
2. El sistema vuelca todo en el archivo de la base de datos.
3. Los guarda.

Postcondiciones:

- Los datos serán volcados en el archivo.

Flujos alternativos:

002.a. No existe el archivo o está dañado.

Eliminar una cita

ID: 13 **Descripción:** El sistema elimina la cita de un paciente

Actores principales: Usuario. **Actores secundarios:** Paciente.

Precondiciones:

- Necesario que exista el paciente, y una cita asociada al paciente

Flujo principal:

1. El usuario desea consultar la cita de un paciente.
2. El usuario inicia el programa y solicita una vision de la cita de un paciente.
3. El usuario introduce el nombre y apellido del paciente del cual desea ver su cita.
4. El sistema elimina la cita asociada al paciente dado.

Postcondiciones:

- Devuelve un mensaje de confirmacion de eliminacion.

Flujos alternativos:

8.a. Si no existe la cita del paciente, el sistema muestra un mensaje de error.

Los componentes que conforman un caso de uso son:

1. **ID:** identificador numérico de la función, coincide con el identificador de los casos de uso.
2. **Descripción:** Una breve descripción del caso de uso en cuestión.
3. **Actores:** se le llama actor a toda entidad externa al sistema que guarda una relación con éste y que le demanda una funcionalidad. Esto incluye a los operadores humanos pero también incluye a todos los sistemas externos, además de entidades abstractas, como el tiempo.
4. **Precondiciones:** acciones que deben realizarse antes poder entrar en el flujo principal. 17
5. **Flujo principal:** acciones que se realizan en el sistema en caso de que el funcionamiento sea el correcto.
6. **Postcondiciones:** modificaciones en el sistema o el programa que ocurrirán tras la finalización correcto de la función.
7. **Flujos alternativos:** acciones que ocurrirán en el sistema o en el programa en caso de que el flujo principal no funcione correctamente.

5. Validación de los Casos de Uso con los Requisitos Funcionales

Insertar paciente

ID: 001 **Nombre**: Insertar Paciente

Prioridad (de 1 a 10): 10

Responsable: Gonzalo Reyes Criado

Descripción

Como *administrador* quiero *insertar un paciente* para *poder tener guardados sus datos la base de datos*

Validación

- Se deben insertar datos en todos los campos para poder guardar al paciente.
- El paciente no puede repetirse.

Modificar paciente

ID: 002 **Nombre**: Modificar Paciente

Prioridad (de 1 a 10): 5

Responsable: Gonzalo Reyes Criado

Descripción

Como *administrador* quiero *modificar los datos de un paciente* para *poder tener actualizada la base de datos*

Validación

- El paciente debe seguir existiendo en la base de datos.
- Si la modificación se ha completado, alguno de los datos relativos al paciente deben de haber cambiado en la base de datos.

Eliminar paciente

ID: 003 **Nombre**: Eliminar Paciente

Prioridad (de 1 a 10): 10

Responsable: Gonzalo Reyes Criado

Descripción

Como *administrador* quiero *eliminar un paciente* para *tener actualizada la base de datos*

Validación

- Se deben poder eliminar cualquier usuario.

Buscar paciente

ID: 004 **Nombre**: Buscar Paciente

Prioridad (de 1 a 10): 10

Responsable: Gonzalo Reyes Criado

Descripción

Como *administrador* quiero *buscar un paciente* para *saber si existe y sus datos*

Validación

- Si existe se debe poder manipular.
- Si existe, debe poder verse físicamente en la base de datos.

Mostrar lista de pacientes

ID: 05 **Nombre:** Mostrar lista de Pacientes

Prioridad: Media

Responsable: Juan José Ropero Cerro

Descripción

Como *usuario* quiero *ver todos los pacientes* para *consultar su información y sus citas previstas*

Validación

- Se debe poder mostrar cualquier usuario.
- Los resultados de la búsqueda pueden mostrar cero resultados.
- Se deben mostrar todos los pacientes.
- Se deben mostrar los datos de los pacientes.
- Se deben mostrar las citas de los pacientes.

Crear cita

ID: 06 **Nombre:** Crear cita

Prioridad: Alta

Responsable: Juan José Ropero Cerro

Descripción

Como *usuario* quiero *crear una nueva cita* para *citar a un paciente un determinado dia*

Validación

- Se debe poder crear una cita a cualquier usuario de la base de datos.
- Los resultados de la creacion pueden ser aceptado o rechazados.
- Se deben mostrar el paciente deseado.
- Se debe poder almacenar la cita del paciente.

Modificar cita

ID: 07 **Nombre:** Modificar cita

Prioridad: Alta

Responsable: Juan José Ropero Cerro

Descripción

Como *usuario* quiero *buscar la cita de un paciente* para *modificar su cita prevista.*

Validación

- Se debe poder mostrar cualquier usuario de la base de datos.
- Los resultados de la búsqueda pueden mostrar cero o un resultado.
- Se deben mostrar el paciente deseado.
- Se deben mostrar la cita del paciente a modificar.
- Se ha de poder modificar los parametros de la cita del paciete deseado.

Mostrar cita

ID: 08 **Nombre**: Mostrar cita del Paciente

Prioridad: Media

Responsable: Juan José Ropero Cerro

Descripción

Como *usuario* quiero *ver la cita del paciente deseado* para *consultar su información y su cita prevista*

Validación

- Se debe poder mostrar cualquier usuario de la base de datos.
- Los resultados de la búsqueda pueden mostrar cero o un resultado.
- Se deben mostrar el paciente deseado.
- Se deben mostrar los datos del paciente.
- Se deben mostrar la cita del paciente.

Mostrar lista de citas

ID: 009 **Nombre**: Mostrar la lista de citas.

Prioridad: Alta.

Responsable: Rafael Román de los reyes.

Descripción

Como *administrador* quiero *mostrar la lista de todas las citas* para *poder ver de un vistazo todas las citas que tengo durante la semana.*

Validación

- El sistema debe mostrar todas las citas.
- Deben de estar ordenadas de la más próxima (la siguiente al momento en el que se ve) hasta la última de la semana.
- Deben mostrar sólo la información esencial en la vista rápida (día, paciente, hora).

Añadir datos al historial de paciente

ID: 010 **Nombre**: Añadir datos al historial del Paciente.

Prioridad: Alta.

Responsable: Rafael Román de los reyes.

Descripción

Como *administrador* quiero *poder añadir datos al historial del paciente* para *poder ir actualizando su historia clínica conforme pasan las citas.*

Validación

- El sistema debe permitir buscar un paciente en concreto.
- Una vez encontrado, debe permitir añadir los datos de que quiera el facultativo de forma cómoda.
- Debe guardar todos los datos de inmediato.

Modificar datos del historial de paciente

ID: 011 **Nombre**: Modificar datos del historial de Paciente.

Prioridad: Alta.

Responsable: Rafael Román de los reyes.

Descripción

Como *administrador* quiero *poder modificar datos del historial del paciente* para *poder corregir cualquier error o tratamiento que se haya puesto con anterioridad.*

Validación

- El sistema debe permitir buscar un paciente en concreto.
- Una vez encontrado, debe permitir modificar los datos de la historia de forma cómoda.
- Debe guardar todos los datos de inmediato.

Volcar datos a fichero

ID: 012 **Nombre**: Volcar datos a fichero.

Prioridad: Urgente.

Responsable: Rafael Román de los reyes.

Descripción

Como *administrador* quiero *poder guardar todos mis datos* para *que sean conservados y en caso de fallo del sistema puedan ser recuperados facilmente.*

Validación

- El sistema debe permitir guardar todos los datos.
- Además debe hacerlo de forma periódica.
- Debe permitir crear una copia de los mismos.

Eliminar una cita

ID: 13 **Nombre**: Eliminar Cita

Prioridad: Alta

Responsable: Juan José Ropero Cerro

Descripción

Como *usuario* quiero *eliminar la cita de un paciente*

Validación

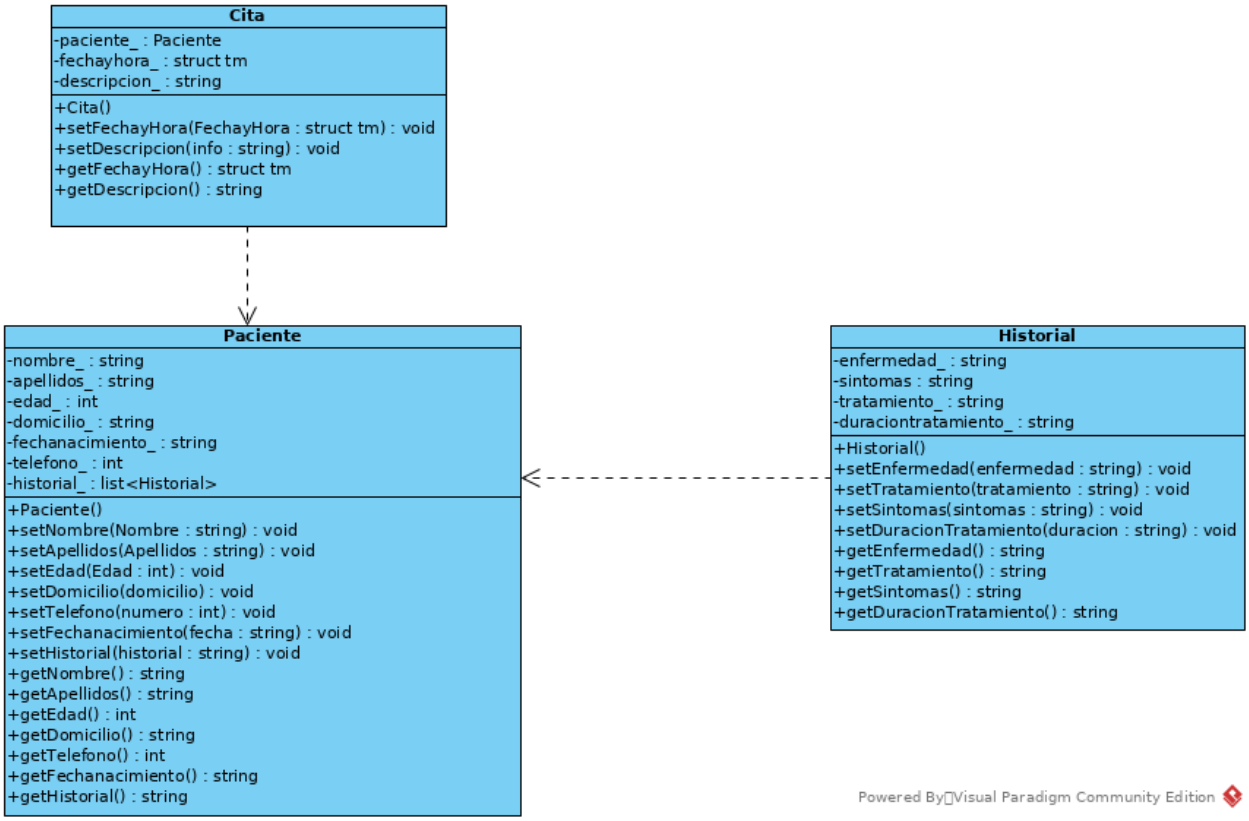
- Se debe poder eliminar la cita de cualquier paciente de la base de datos.
- Los resultados de la eliminacion pueden ser aceptado o rechazados.
- Se debe poder eliminar la cita del paciente.

6. Representación, descripción y especificación de las entidades de información que manejara el sistema

Diagrama de clases

El diagrama de clases en un tipo de diagrama estático que describe la estructura general de un sistema sin entrar en detalles sobre su implementación, donde podemos observar información adicional sobre los atributos o métodos.

En este tipo de diagramas también podemos observar las relaciones entre clases del sistema.



7. Validación de las Clases con los Casos de Uso

La validación de las clases con los casos de uso se puede ver escribiendo cada caso de uso y las clases en las que está involucrado, pero hemos valorado mejor usar una presentación tabular que, de un vistazo y en la misma página, permite ver la validación de las clases con cada caso de uso.

Se observa la siguiente Matriz Casos de uso / clases:

Caso de Uso	**Paciente**	**Cita**	**Historial**
Insertar un paciente.	**X**		**X**
Modificar un paciente.	**X**		
Eliminar un paciente.	**X**		**X**
Mostrar un paciente.	**X**		
Mostrar una lista de todos los pacientes.	**X**		
Crear una cita.		**X**	
Modificar una cita.		**X**	
Mostrar una lista de citas.		**X**	
Añadir datos al Historial del Paciente.			**X**
Modificar datos del historial del paciente.			**X**
Volcar los datos al fichero.	**X**	**X**	**X**
Eliminar una cita.		**X**	

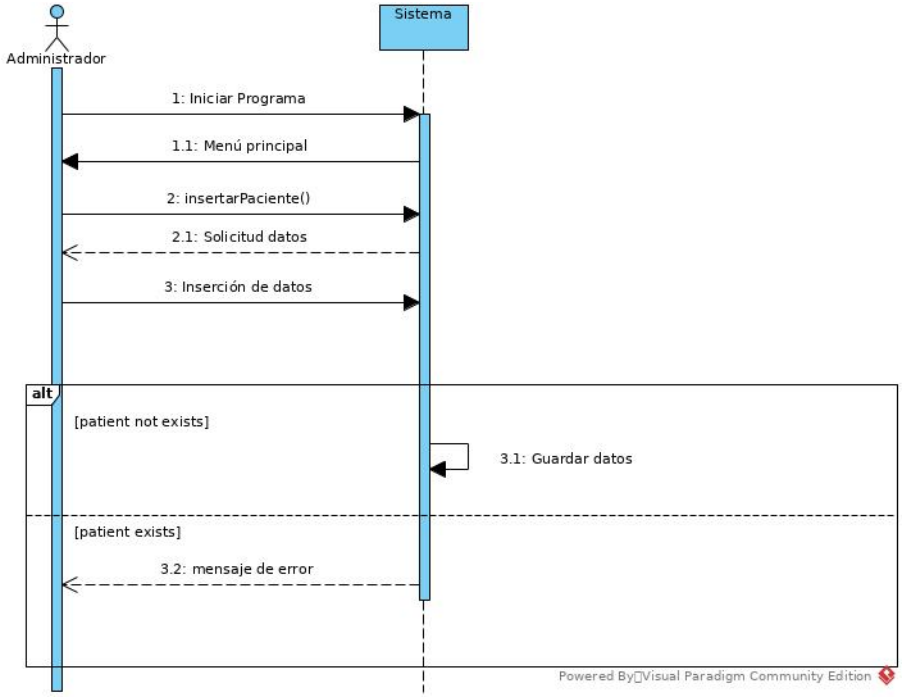
Consideramos que insertar y eliminar un paciente también influye en el historial pues cuando se da de alta o baja a un paciente también estamos creando indirectamente su historial. También consideramos que el volcado de datos a fichero, al tener cada una de las clases su propia función para guardar los datos, incumbe a todas las clases creadas.

Como las citas pueden ser con alguien que aún no esté dado de alta en el sistema, no hemos creído conveniente que el caso de uso correspondiente a las citas tenga relación con la clase paciente.

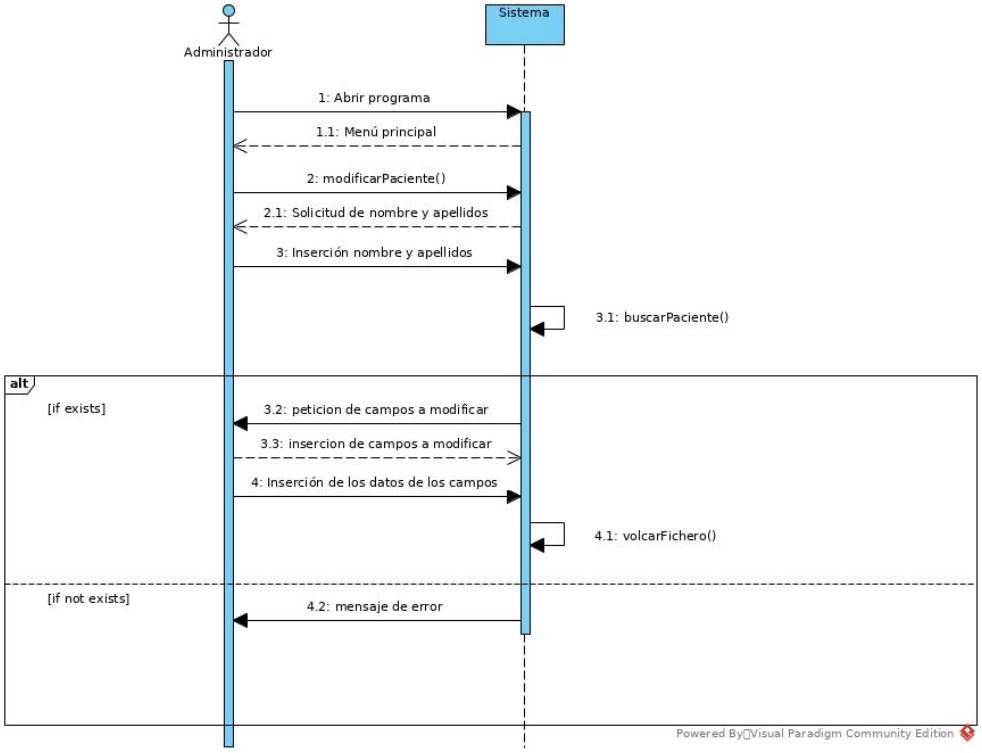
8. Descripción de los escenarios de interacción que se produzcan en el sistema para llevar a cabo la funcionalidad descrita en los Casos de Uso más complejos

Vamos a mostrar los diagramas de secuencia para todas las actividades detalladas anteriormente.

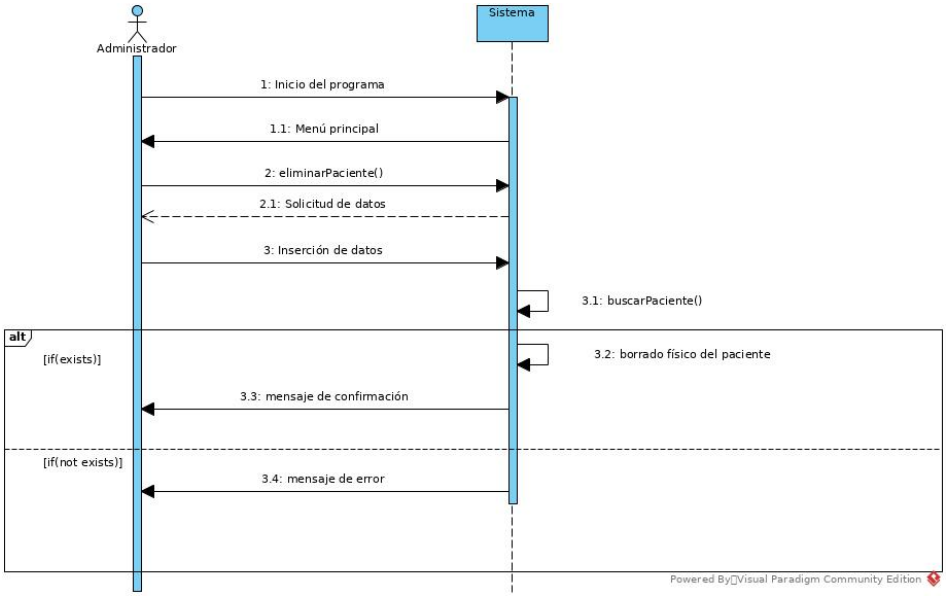
Insertar paciente



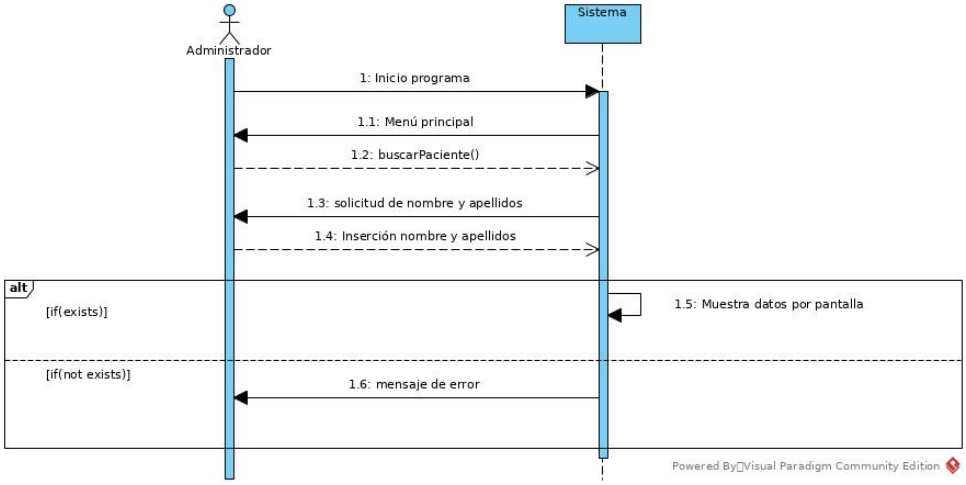
Modificar paciente



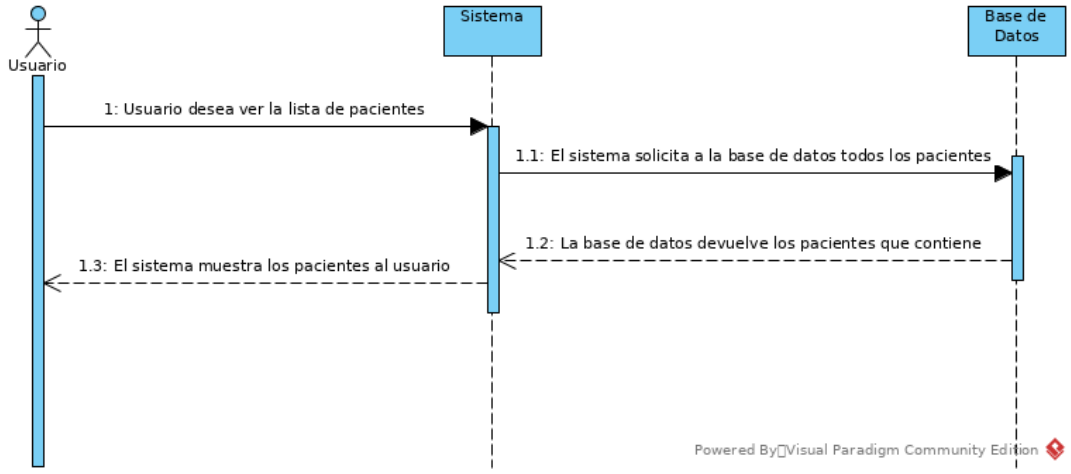
Eliminar paciente



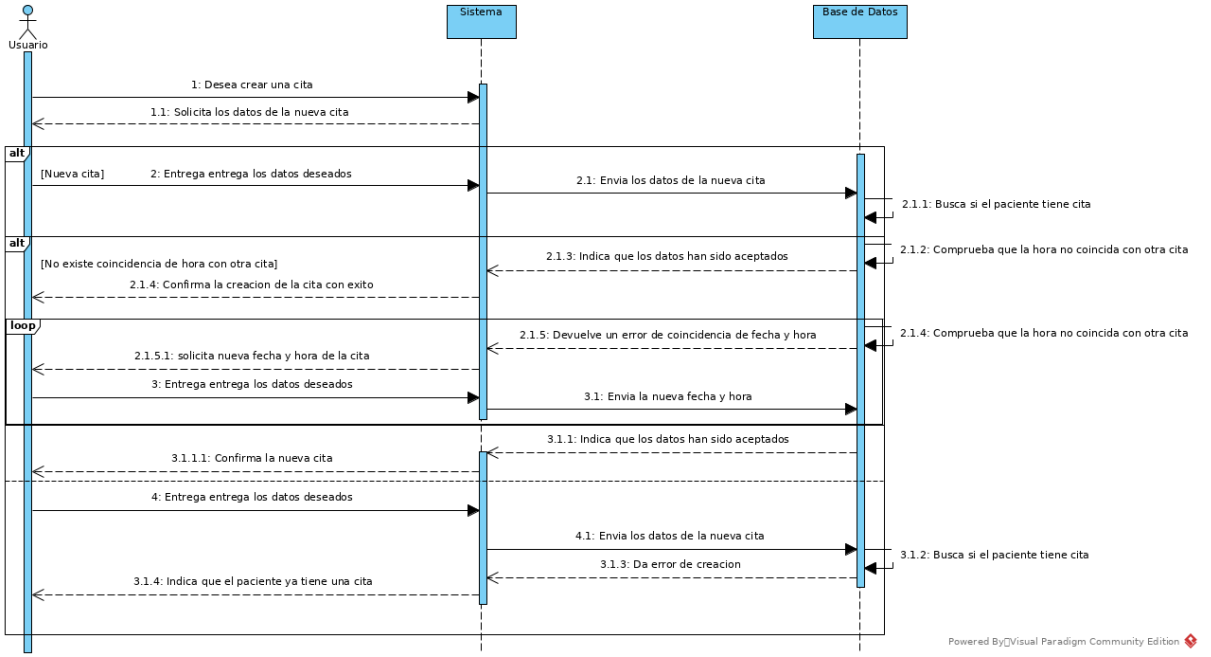
Buscar paciente



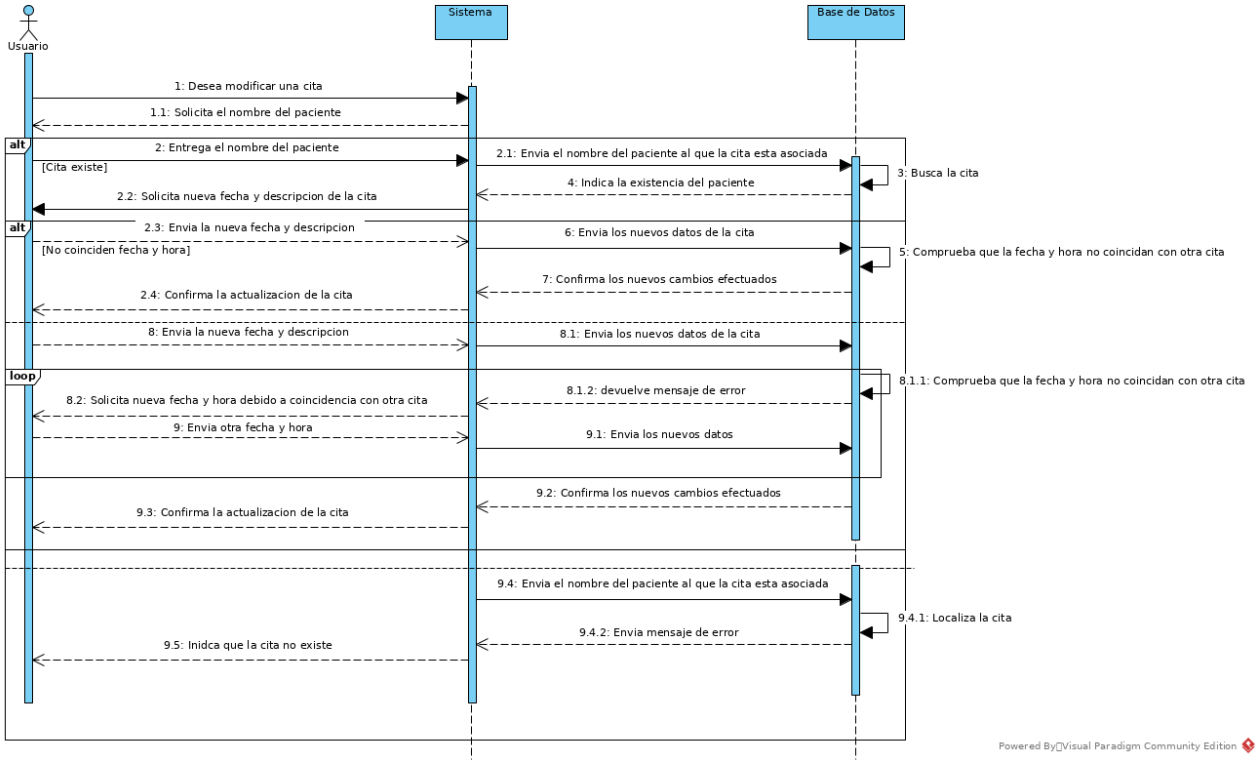
Mostrar lista de pacientes



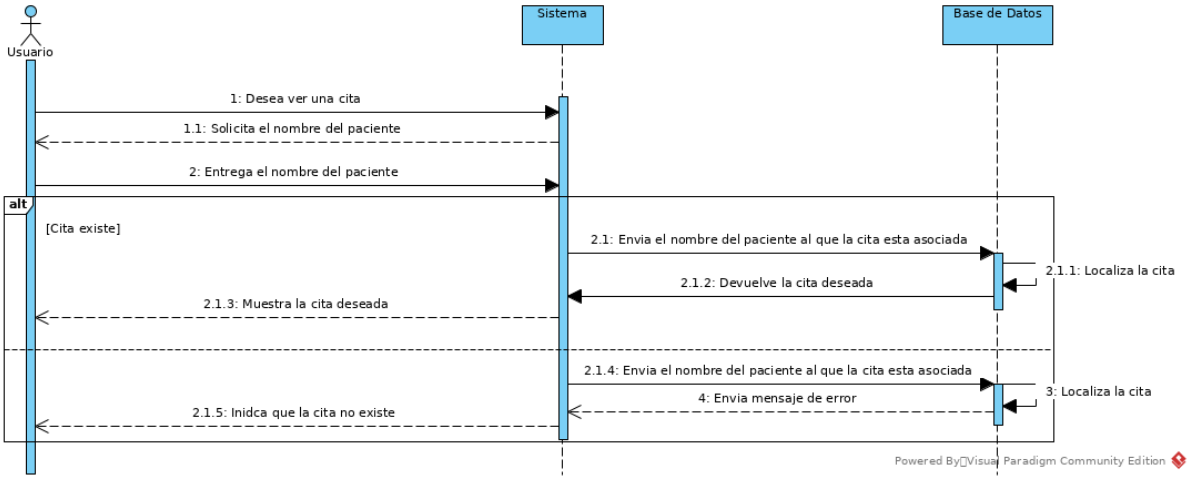
Crear cita



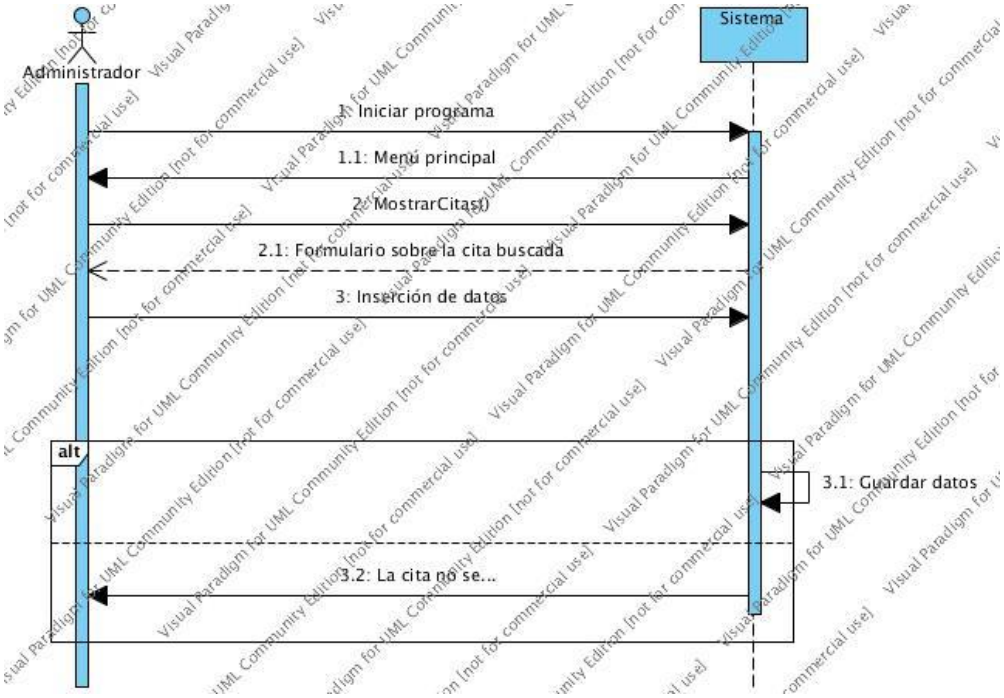
Modificar cita



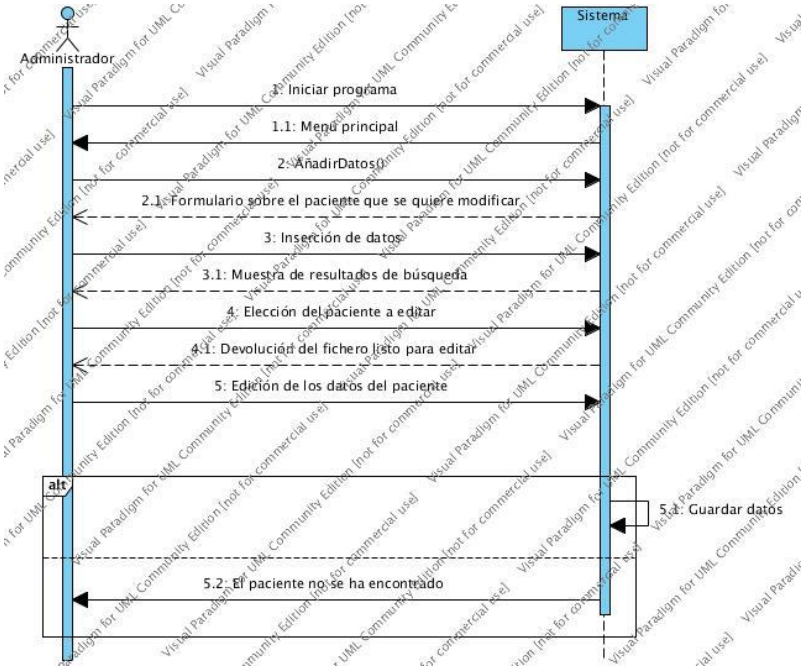
Mostrar cita



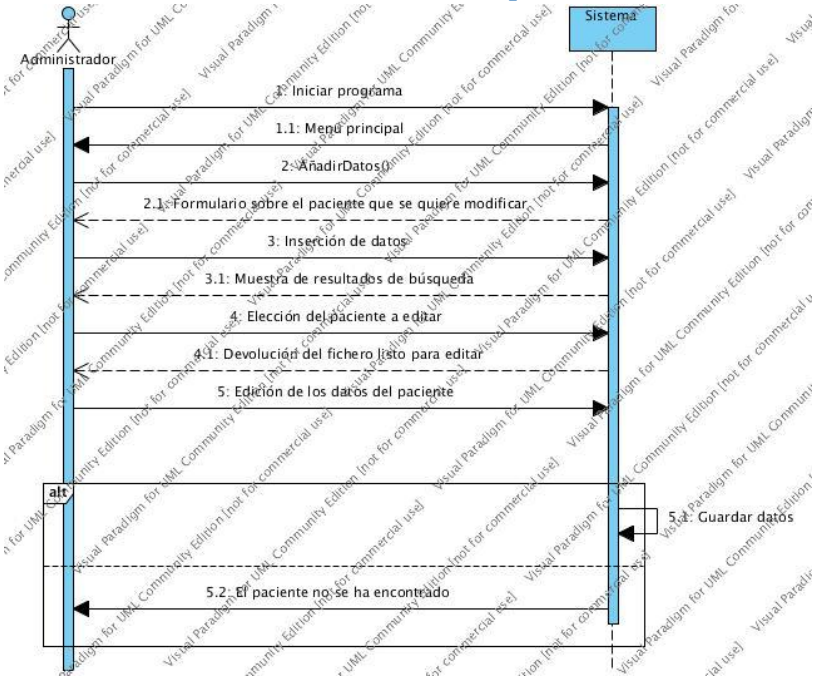
Mostrar lista de citas



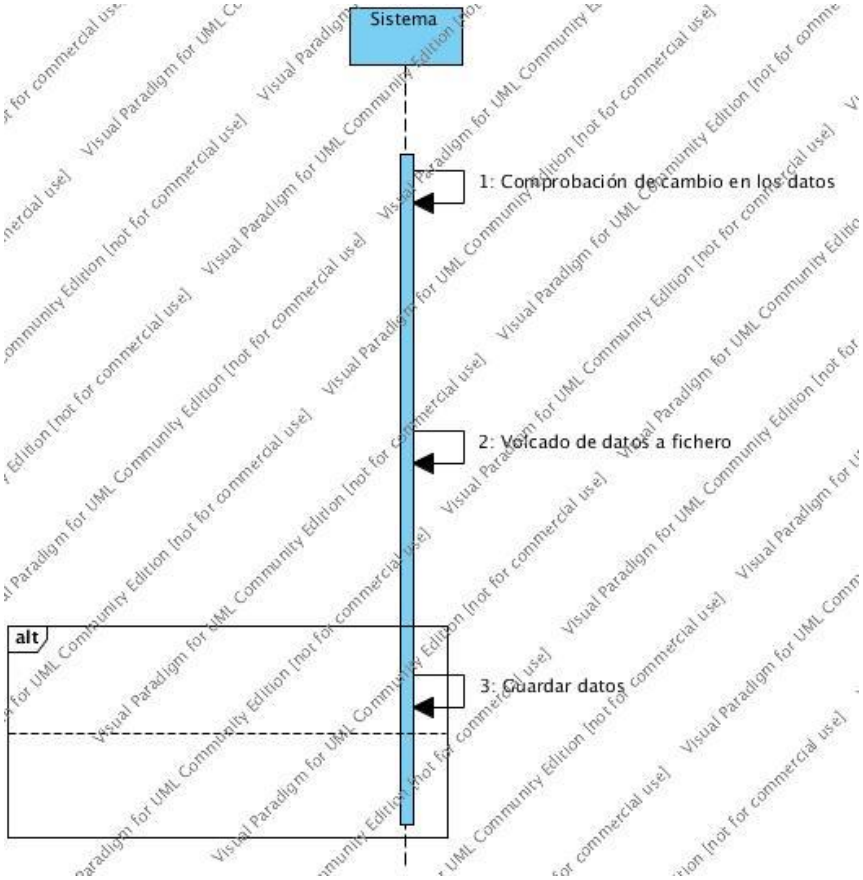
Añadir datos al historial de paciente



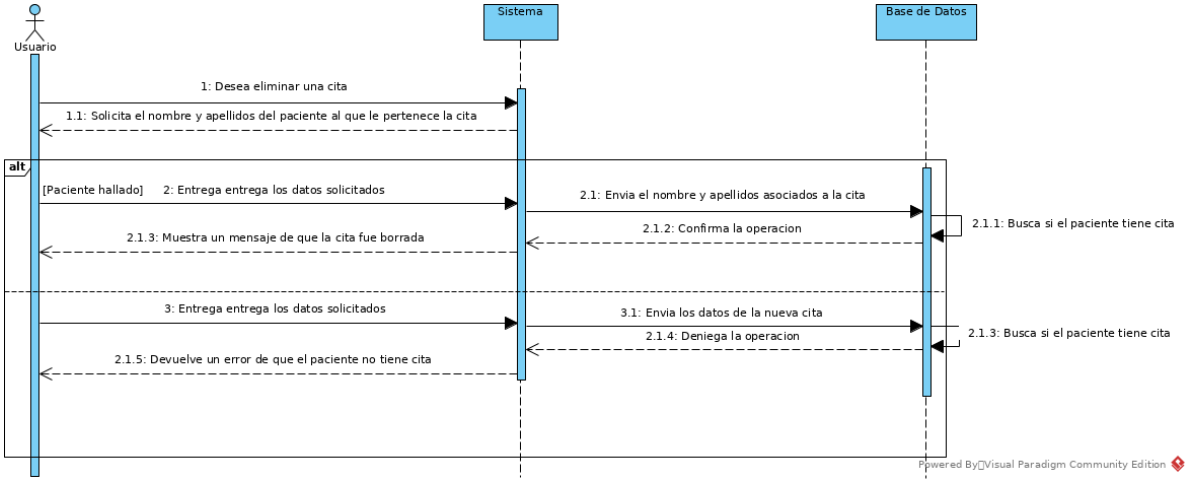
Modificar datos del historial de paciente



Volcar datos a fichero



Eliminar una cita



9. Implementación, refinamiento y pruebas usando la metodología SCRUM con las iteraciones que sean necesarias

Durante el curso hemos intentado seguir una metodología de trabajo SCRUM. Hemos creado los Sprints y hemos intentado seguirlos, pero, como es obvio, la carga de trabajo de otras asignaturas ha provocado retrasos que ha hecho que en algunas ocasiones no siguiéramos la línea temporal impuesta por SCRUM, por lo que los tiempos cambiaron respecto a los especificados en TAIGA.

Primer Sprint – 19 de noviembre al 24 de noviembre

En este primer sprint definimos el proyecto y creamos el árbol de archivos de trabajo para comenzar el desarrollo.

Tarea	**Asignación**	**Puntos**	**Realización**
Preparación de árbol de trabajo	Todos	1	Todos, creamos nuestras ramas y preparamos los archivos para trabajar
Definición clase Paciente	Gonzalo	1	Gonzalo
Definición clase Cita	Juanjo	1	Juanjo, corrigen Gonzalo y Rafael
Definición clase Historial	Gonzalo	1	Gonzalo
Agregados de clase Cita	Rafael	1	Rafael
Reparto de Casos de Uso	Rafael	1	Todos

Este primer Sprint fue cumplido en su tiempo previsto.

Segundo sprint – 24 de noviembre al 1 de diciembre

En el segundo Sprint se crearon el esqueleto de todas las clases para comenzar a hacer pruebas de aplicación de forma progresiva.

Tarea	**Asignación**	**Puntos**	**Realización**
Desarrollo de la clase Cita	Juanjo	3	Juanjo, corrección por todos
Desarrollo de la clase Paciente	Gonzalo	3	Gonzalo, corrección por todos
Desarrollo de la clase Historial	Gonzalo	3	Gonzalo, corrección por todos, se detectan errores que se subsanarán en el próximo sprint
Comienzo de la memoria de proyecto	Rafel	2	Rafael

Tercer sprint – 1 de diciembre al 15 de diciembre

En el tercer sprint se pulieron algunos detalles que quedaron sin resolver en el anterior sprint y se retocaron algunos elementos de las clases que incumbían a nuestro proyecto.

Tarea	**Asignación**	**Puntos**	**Realización**
Mejora de la clase Cita	Juanjo	2	Juanjo, corrección por todos
Mejora de la clase Historial	Gonzalo	4	Gonzalo y Rafael corrección por todos, sigue habiendo errores.
Adición de datos a Memoria	Rafel	2	Rafael
Pruebas de aplicación	Juanjo Gonzalo	5	Juanjo y Gonzalo con corrección por parte de todo el equipo

Cuarto sprint – 20 de enero a 2 de febrero

En el cuarto sprint se retocaron algunos aspectos de la aplicación, sobre todo la clase historial para fusionarla con la de paciente. También se finalizó la documentación.

Tarea	**Asignación**	**Puntos**	**Realización**
Corrección clase Paciente	Gonzalo	3	Gonzalo, corrección por todos
Fusión clase Historial con Paciente	Gonzalo	3	Gonzalo, corrección por todos, la clase historial sigue sin funcionar.
Finalización memoria	Rafel	4	Rafael

La clase historial nos ha dado siempre una gran cantidad de errores desde que comenzamos a codificarla. Hemos intentado subsanar los errores como hemos podido, pero finalmente tuvimos que eliminar la funcionalidad. Creemos que el error se encuentra en la forma en la que lo tenemos todo implementado y la gran cantidad de exámenes y prácticas que tenemos actualmente nos han impedido buscar una solución a tiempo.