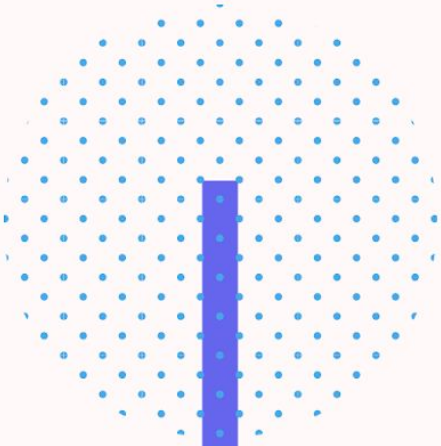
ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA INFORMÁTICA

MARTA ESCRIBANO LÓPEZ
JESÚS BUENO RUIZ
LUÍS ANERI DELGADO



ÍNDICE:

INTRODUCCIÓN:

La evolución hacia puestos de empleo mucho más avanzados tecnológicamente ha supuesto un cambio positivo. Gracias a esto, se ha logrado simplificar los procesos de producción y las formas en las que se desempeñan algunas tareas. Además, ha permitido llevar a cabo mejoras en las condiciones de trabajo, aumentar la seguridad y favorecer la comunicación y el flujo de intercambio de información. De hecho, durante los últimos años se ha experimentado un importante cambio en el ámbito laboral y de los recursos humanos, ya que las empresas demandan con más frecuencia candidatos preparados para afrontar los retos que las nuevas tecnologías han impuesto.

Aumenta la eficiencia, la productividad y ofrece información en tiempo real: en algunos sectores, el desarrollo de la tecnología ha sido uno de los factores más importantes para conseguir empresas modernas, competitivas, eficientes y con grandes índices de productividad. De la misma forma, el acceso a información en tiempo real sobre los indicadores de las compañías ha permitido mejorar y anticipar la toma de decisiones.

Brinda más opciones de atraer talento: teniendo en cuenta que muchas personas utilizan canales digitales para buscar empleo, la presencia de las empresas en estos lugares resulta prácticamente imprescindible. Desde el punto de vista de los recursos humanos, estas opciones permiten atraer talento en mayor medida y conocer mucho mejor a los candidatos.

La confluencia entre empleador y trabajador es cada vez más estrecha, teniendo en cuenta los beneficios que aporta la tecnología en este tipo de relaciones laborales. Mientras que el primero ve reducidos sus costes de producción, el segundo disfruta de mejores condiciones a la hora de conciliar su vida profesional con la personal. En resumen, la tecnología ha contribuido a hacer más fácil el día a día tanto a empresas (permitiendo un crecimiento mayor y más rápido), como a trabajadores (que son más eficientes y productivos) y a profesionales de los recursos humanos (capaces de gestionar y elaborar estrategias mucho más complejas).

1.- Descripción del sistema/problema.

Nuestro sistema está adaptado a las necesidades de una clínica a la hora de guardar, mostrar y analizar la información de sus pacientes. Nuestro cliente desea poder tener toda esa información en un programa y poder manejarla desde ahí. El programa guarda la información en ficheros de texto y se ha realizado en el lenguaje de programación C++.

Una vez analizado y comprendido el problema de nuestro cliente. Nuestro objetivo es que con este programa intentaremos facilitarle a nuestro cliente la extracción de la información al igual que la implementación de esta.

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

Continuando con el desarrollo de nuestro programa, vamos a exponer los requisitos que se deberán cumplir. Estos serán clasificados por Datos que gestiona el sistema, requisitos funcionales y requisitos no funcionales:

Datos que Gestiona el Sistema :

- Datos de la cita:
    - Fecha
    - Estado
    - DNI paciente
    - Notas

- Datos de usuario:
    - Nombre y apellidos
    - DNI
    - Dirección postal
    - Teléfono
    - Fecha de Nacimiento
    - Compañía
    - Alta médica

- Historial Médico
    - Fecha de la anotación
    - Anotación
    - DNI Paciente
    - Estado de la Cita

- Historial de Tratamiento
    - Fecha de inicio
    - Fecha de parada
    - Nombre Tratamiento
    - Dni Paciente
    - Estado de Tratamiento

REQUISITOS FUNCIONALES :

- El programa debe ser capaz de buscar al paciente por nombre o DNI y que el programa muestre los datos del usuario.
- Se debe de poder añadir una cita.
- Se debe de poder modificar la cita.
- Se debe de poder dar de alta a un usuario.
- Se debe de poder acceder al historial médico y al historial de tratamiento.
- Debe de haber una opción que añada un tratamiento.
- Debe de haber una opción que cancele un tratamiento y actualice el historial.
- Se debe de poder añadir datos de la cita al historial médico.
- Debe de poder mostrar una lista con todos los pacientes.
- Se debe poder mostrar las citas de cualquier día.
- Se debe de poder visualizar las citas de hoy.

REQUISITOS NO-FUNCIONALES :

- Almacenamiento ilimitado de pacientes de la clínica.
- Lenguaje de implementación C++.
- El historial médico del paciente no se puede borrar.
- El historial de tratamiento no se puede borrar.
- Programa Compatible con Linux.

3.- Actores, Casos de uso y priorización de los mismos.

A continuación haremos una breve descripción de los casos de uso implementados y la importancia de estos mismos.

- 01.- Buscar usuario por el DNI. Introduciremos el DNI del usuario que deseamos buscar y el sistema lo muestra. El actor principal es el administrador y el secundario el paciente. No tiene ninguna precondición. Su flujo principal sería:
    - El administrador desea consultar los datos de un paciente.
    - El administrador abre el cuadro de diálogo de búsqueda en el menú principal.
    - El administrador introduce el DNI del paciente.
    - El sistema muestra por pantalla los datos del paciente.

Se tiene que mostrar al administrador las distintas operaciones relativas al paciente. Y como opción alternativa, si el paciente que se ha buscado no existe, se debe de mostrar un mensaje de error.

- 02.- Modificar el usuario. Introducimos el DNI del usuario que deseamos modificar y el sistema te da la opción de editar los datos. El actor principal es el administrador y el secundario sería el paciente. Una precondición es que ya exista el paciente que se quiera modificar en el sistema. Su flujo principal sería:
    - El administrador desea modificar los datos de un paciente.
    - El administrador abre el cuadro de diálogo de modificar en el menú principal.
    - El administrador introduce el DNI del paciente.
    - El programa le pregunta al administrador por los datos que quiere variar.
    - El administrador los introduce y el programa lo guarda.

- Se muestra el administrador con los datos modificados.

Se debe mostrar al administrador los datos del paciente que ya han sido modificados. Al igual que en caso anterior, si el paciente buscado no existe se deberá de mostrar un mensaje de error.

- 03.- Añadir cita. Desde la opción de citas aparece la opción de añadir una cita nueva. El actor principal es el administrador y el secundario el usuario. Las precondiciones serían que la fecha y hora a la que se le quiere asignar la nueva cita estén libres y que el usuario al que se le quiere asignar la cita deberá estar registrado en el sistema. Sus flujos principales serían:
  - El administrador desea crear una cita para un paciente.
  - El administrador accede a la pestaña de citas.
  - El administrador clickea en el botón añadir una cita.
  - El programa le pregunta al administrador la fecha que desee para fijar una cita.
  - El programa le pregunta al administrador la hora que desee para fijar la cita.
  - Si no está ocupado el programa pide al administrador que introduzca el nombre y apellidos del paciente.
  - Se muestra al administrador la cita con fecha, hora y paciente.

Las post condiciones que deberá tener sería que la cita queda guardada en los datos de citas, el usuario no tendrá permitido solicitar una cita hasta que concluya o se modifique esta y se comprobará que la nueva cita no solapa con otra ya existente. Como flujo alternativo debemos decir que si el espacio de tiempo está ocupado, deberá mostrarse un error , si el usuario no existe o ya posee una cita se deberá mostrar un mensaje de error especificando el porqué.

- 04.- Modificar Cita. Desde el apartado citas aparece la opción de modificar una cita ya existente. Los actores son los mismos que en el caso anterior. La cita debe de estar previamente fijada. Sus flujos principales son:
    - El administrador desea modificar una cita para un paciente ya creada.
    - El administrador accede a la pestaña de citas.
    - El administrador clickea en el botón modificar una cita.
    - El programa le pregunta al administrador el paciente que tiene asignada la cita que se desea modificar.
    - El programa le pregunta al administrador la fecha que desee para fijar la nueva cita.
    - El programa le pregunta al administrador la hora que desee para fijar la nueva cita.
    - Se muestra al administrador la cita con fecha, hora y paciente modificada.

La cita debe de quedar guardada en los datos de cita. Algún flujo alternativo sería que si el usuario no existe o la nueva fecha solicitada coincide con otra, se deberá mostrar un mensaje de error aclarando cual es el problema.

- 05.- Añadir paciente. Desde el apartado pacientes dar de alta a un nuevo paciente. El actor principal sería el administrador y el actor secundario el o los paciente/s. Para este caso no existe ninguna precondición. Su flujo principal sería:
    - El administrador desea añadir un nuevo paciente al sistema (darle de alta).
    - El administrador accede a la pestaña Pacientes.
    - El administrador clickea en el botón dar de alta.
    - El programa muestra un formulario con todos los datos personales del usuario.
    - El administrador rellenará los datos a mano.
    - El administrador le da al botón de aceptar.
    - Se muestra al administrador la ficha del nuevo paciente con sus datos personales.

Como post condición, el paciente debe quedar registrado en los datos del sistema junto con sus datos personales y el programa debe de haber creado un apartado de historial médico y de tratamiento para el usuario. El flujo alternativo si el administrador no rellena un campo obligatorio se mostrará un mensaje de error y si existe un usuario con el mismo DNI se mostrará otro mensaje de error.

- 06.- Mostrar lista. Desde el apartado de pacientes, el administrador le da a la opción de mostrar lista. No hay ninguna precondición y el flujo principal sería:
    - El administrador desea visualizar una lista con todos los pacientes registrados para un vistazo rápido.
    - El administrador accede a la pestaña de 'lista de usuarios'.
    - El programa muestra una lista ordenada por fecha de alta en el sistema.
    - El programa muestra en la lista nombre, apellidos, dni, estado en cada fila.

En esta opción, no hay ninguna post condición ni tampoco ningún flujo alternativo.

- 07.- Visualizar historial médico. Desde el apartado de Citas visualizar un ‘time-line’ de la historia médica del paciente. El actor principal es el administrador y los actores secundarios serían los pacientes. Una precondición sería que el paciente estuviese registrado en el Sistema. El flujo principal sería:
    - El administrador desea visualizar el historial médico del paciente.
    - El administrador accede a la pestaña historial médico.
    - El administrador busca por el DNI al paciente deseado.
    - El programa muestra un ‘time-line’ del historial del paciente.

  No existe ninguna precondición ni tampoco ningún flujo alternativo.

- 08.- Visualizar el historial del tratamiento. Desde el apartado de Tratamientos visualizamos los tratamientos que el paciente ha recibido. El actor principal es el administrador y los secundarios los pacientes. La precondición establecida sería que el paciente tiene que estar registrado en el Sistema. El flujo principal sería:
    - El administrador desea visualizar el historial de tratamiento del paciente.
    - El administrador accede a la pestaña historial de tratamiento.
    - El administrador busca por DNI al paciente deseado.
    - El programa muestra un 'time-line' del historial del tratamiento del paciente.

No existe ninguna post condición ni tampoco ningún flujo alternativo para este caso.

- 9.- Añadir tratamiento. Desde el apartado Tratamientos añadir un tratamiento para el paciente. El actor principal es el administrador y los secundarios el paciente. Las precondiciones establecidas son que el paciente debe de estar registrado en el sistema y el paciente debe estar sin ningún tratamiento recetado. El flujo principal sería:
    - El administrador desea añadir un tratamiento médico al paciente.
    - El administrador accede a la pestaña tratamientos.
    - El administrador accede al apartado añadir tratamiento.
    - El administrador busca DNI al paciente deseado.
    - El programa abre un cuadro de diálogo para añadir tratamiento.
    - El administrador escribe el tratamiento médico, la fecha en la que se ha recetado y le da a aceptar.

El tratamiento se queda almacenado en el historial del paciente y puede ser visualizado posteriormente y el tratamiento puede ser cancelado también. Los flujos alternativos serían que si el administrador busca un paciente que no está registrado, aparecerá un mensaje de error indicando que no existe ningún paciente con esos atributos registrado. Y que si el administrador introduce la fecha en un formato no correcto, aparecerá un mensaje de error.

- 10.- Modificar el tratamiento. Desde el apartado de Tratamiento Médico modificar un tratamiento ya creado. Los actores son los mismo que se han repetido con anterioridad. Las precondiciones son que el paciente debe estar registrado en el sistema y que el paciente debe tener algún tratamiento recetado. El flujo principal sería:
    - El administrador desea modificar un tratamiento médico de un paciente.
    - El administrador accede a la pestaña tratamientos.
    - El administrador accede al apartado modificar tratamiento.
    - El administrador busca por DNI al paciente deseado.
    - El programa abre un cuadro de diálogo para modificar tratamiento.
    - El programa muestra el historial médico del paciente, así como el tratamiento en vigor con su respectiva fecha.
    - El administrador escribe el nuevo tratamiento médico del paciente y la fecha en la que se ha modificado y le da a aceptar.

El tratamiento se queda almacenado en el historial del paciente y puede ser visualizado posteriormente, el tratamiento no puede ser borrado y el tratamiento mostrará la fecha en la que ha sido modificado. El flujo alternativo es que si el administrador busca un paciente que no está registrado, aparecerá un mensaje de error indicando que no existe ningún paciente con esos atributos registrado. Si el paciente buscado no tiene ningún tratamiento recetado, aparecerá la pestaña añadir tratamiento. Si el administrador introduce la fecha en un formato no correcto, aparecerá un mensaje de error.

- 11.- Dar de alta a un paciente. Desde el apartado Pacientes dar de alta a un paciente registrado. Se mantiene los actores, y la precondición sería que el paciente debe de estar registrado. Los flujos principales son:
    - El administrador desea dar de alta a un paciente al sistema.
    - El administrador accede a la pestaña Pacientes.
    - El administrador pincha la opción dar de alta a un paciente.
    - El administrador busca por DNI al paciente.
    - Cambia su estado de paciente actual actual a paciente alta.

El paciente debe quedar registrado en el sistema pero con el estado de ‘paciente de alta’. Y no existe ningún flujo alternativo.

- 12.- Mostrar listas de citas de hoy. Desde la opción de citas se mostrará las citas asignadas para el día de hoy. Debe de existir una cita para que el día deseado sea visualizado. El flujo principal sería:
    - El administrador desea visualizar una lista con todos los pacientes los cuales tiene cita hoy.
    - El administrador accede a la pestaña de citas, donde podrá acceder a las hoy.
    - El programa muestra una lista ordenada por fecha horaria todas las citas del día.
    - El programa muestra en la lista hora de la cita, dni, estado en cada fila.

No existe ninguna post condición ni tampoco ningún flujo alternativo para este caso.

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

Matriz requisitos funcionales (RF) - casos de uso (CU): cada RF debe quedar cubierto por al menos un CU. Con esto nos aseguramos que todas las funcionalidades requeridas son tenidas en cuenta.

Los requisitos funcionales de forma enumerada serían:

1. *El programa debe ser capaz de buscar al paciente por DNI y que el programa muestre los datos del usuario.*
2. *Se debe de poder añadir una cita.*
3. *Se debe de poder modificar la cita.*
4. *Se debe de poder dar de alta a un usuario.*
5. *Se debe de poder acceder al historial médico y al historial de tratamiento.*
6. *Debe de haber una opción que añada un tratamiento.*
7. *Debe de haber una opción que cancele un tratamiento y actualice el historial.*
8. *Se debe de poder añadir datos de la cita al historial médico.*
9. *Debe de poder mostrar una lista con todos los pacientes.*
10. *Se debe poder mostrar las citas de cualquier día.*
11. *Se debe de poder visualizar las citas de hoy.*

Y los casos de uso enumerados serían:

1. *Buscar usuario*
2. *Modificar usuario*
3. *Añadir Cita*
4. *Modificar Cita*
5. *Añadir Paciente*
6. *Mostrar lista*
7. *Visualizar Historial Médico*
8. *Visualizar historial de Tratamiento*
9. *Añadir Tratamiento*
10. *Modificar Tratamiento*
11. *Dar de alta*
12. *Mostrar Lista de citas de hoy*

CASOS DE USO													
R. Func		1	2	3	4	5	6	7	8	9	10	11	12
	1	✓											
	2			✓		✓							
	3				✓								
	4											✓	
	5							✓	✓				
	6									✓	✓		
	7										✓		
	8												
	9						✓						
	10				✓								
	11											✓	✓

6.- Representación, descripción y especificación de los entidades de información que manejará el sistema (Diagrama de Clases).

Nuestro diagrama de clases estará compuesto por:

CLASE: CITA

DESCRIPCIÓN DE LA CLASE:

Esta clase es la clase cita que posee la información sobre la fecha de la cita, un identificador de la misma y un apartado de notas sobre la evaluación del paciente.

DATOS:

- Nota *(string)*: **cadena de tamaño máximo 300 caracteres que contiene alguna anotación sobre la cita del paciente*
- Fecha *(class fecha)*: **fecha de la cita para un paciente*
- ID *(int)*: dato numérico que sirve para ayudar a buscar la cita mas rapida
- DNI *(string)* DNI de la persona a la que pertenece la cita

MÉTODOS:

- Cita(Fecha fecha, string dni) Constructor, recibe por parámetros la fecha y dni de la persona a la que pertenece la cita, y asigna una id aleatoria
- getCita() Observador de la clase Cita
- GetNota() Visualiza la nota de la cita
- setNota(string nota) recibe una cadena por parámetros y la copia en el atributo nota de la cita
- getFecha() visualiza la fecha de la cita deseada

- setFecha(class fecha) Recibe una clase fecha y modifica la fecha del paciente por esta nueva.
- getID() Observador de el ID de la cita
- setID(int ID) Recibe un entero de identificación del paciente.
- getDNI(string DNI) observador de DNI
- setDNI(string DNI) Recibe un nuevo dni por parámetro y modifica el del paciente por este.
- addCita(lista) Añade una nueva cita en una lista.
- deleteCita(dni, fecha, lista) Recibe el dni del paciente, la fecha a la que pertenece la cita y la borra de la lista.

CLASE: FECHA

DESCRIPCIÓN DE LA CLASE:

Esta clase es una clase auxiliar que sirve para almacenar los diferentes tipos de fecha de otras clases.

DATOS:

- Año *(int)*
- Mes *(int)*
- Dia *(int)*
- Hora *(int)*
- minuto *(int)*

MÉTODOS:

- Fecha(dia, mes, año) Constructor, recibe obligatoriamente el dia, mes y año.
- getFecha() Observador la fecha.

- getDia() Observador de la variable dia.
- getMes() Observador de la variable mes.
- getAño() Observador de la variable año.
- getHora() Observador de la variable hora.
- getMinuto() Observador de la variable minuto.
- setDia(int dia) Recibe un parámetro de tipo entero y lo guarda en día_.
- setMes(int mes) Recibe un parámetro de tipo entero y lo guarda en mes_.
- setAño(int año) Recibe un parámetro de tipo entero y lo guarda en año_.
- setHora(int hora) Recibe un parámetro de tipo entero y lo guarda en hora_.
- setMinuto(int minuto) Recibe un parámetro de tipo entero y lo guarda en minuto_.

CLASE: PERSONA

DESCRIPCIÓN DE LA CLASE:

Esta clase es la clase persona que representa a cada uno de los pacientes registrados en el sistema de la clínica y que por tanto posee cada uno de los datos de cada paciente. Además de las funciones de modificar cada uno de los datos, borrar a un paciente, obtener el DNI y mostrar el paciente.

DATOS:

- Nombre *(string)*: cadena de tamaño máximo 50 caracteres que contiene el nombre del paciente.
- Apellido *(string)*: cadena de tamaño máximo 80 caracteres que contiene los apellidos del paciente
- DNI *(string)*: cadena de 9 caracteres que contiene 8 números y una letra y consiste en el NIF del paciente
- Teléfono *(int)*: número entero de 9 dígitos que corresponde al número de teléfono del paciente
- Dirección *(string)*: Consiste en una cadena de una longitud de unos 100 caracteres de longitud que contiene la dirección postal del paciente
- Fecha de Nacimiento *(struct fecha)*: fecha de nacimiento del paciente
- Compañía *(string)*: Compañía médica del paciente (seguro médico o seguridad social).
- Alta médica *(bool)*: Es true si el paciente tiene el alta médica, y false si tiene la baja médica.
- Lista de Tratamientos(list <Tratamiento>) Archiva todos los tratamientos de la persona

MÉTODOS:

- Persona() Constructor, recibe por parámetros todos los datos del paciente nuevo e inicializa el nuevo paciente (todos son obligatorios)
- getNombre() observador de nombre
- getApellidos() observador de apellidos
- getDNI() observador de DNI
- getTelefono() observador de teléfono
- getDireccion() observador de dirección
- getFecha() observador de fecha
- getCompañia() observador de compañía
- setNombre(string nombre) Recibe un nombre por parámetro y modifica el del paciente por este nuevo.
- setApellidos(string apellido) Recibe el apellido por parámetro y modifica el del paciente por este nuevo.
- setDNI(string DNI) Recibe un nuevo dni por parámetro y modifica el del paciente por este.
- setTelefono(int telefono) recibe el teléfono nuevo como parámetro y modifica el ya existente por este.
- setDireccion(string direccion) Recibe una nueva dirección por parametro y modifica la ya existente.
- setFecha(class fecha) Recibe una clase de tipo fecha y modifica la fecha de el paciente por esta nueva.
- setCompañia(string compañía) Recibe la nueva compañía por parametro y modifica la ya existente.
- setAltaMedica(bool alta) Recibe un parámetro de tipo bool y modifica el alta médica del paciente.
- addPersona(persona, lista) Recibe un objeto de tipo Persona y lo guarda en una lista.
- deletePersona(dni, lista) Recibe el dni del paciente que se quiere eliminar, y lo borra de la lista.
- modificarPersona(dni, persona, lista) Recibe el dni de la persona que se quiere modificar, un objeto tipo Persona con nuevos valores, y lo guarda en la lista.

- mostrarPersonas(lista) Muestra todos los pacientes registrados en la lista.
- mostrarPersona(dni, lista) Recibe el dni del paciente que se desea mostrar, lo busca en la lista y lo muestra por pantalla.
- darAlta(dni, lista) Recibe el dni del paciente al que se desea dar de alta y cambia su estado en la lista.
- darBaja(dni, lista) Recibe el dni del paciente al que se desea dar de baja y cambia su estado en la lista.
- mostrarTratamientos() (void) muestra todos los tratamientos de un paciente.
- mostrarTratamiento()(void) muestra un tratamiento especifico
- addTratamiento()(bool) Devuleve ‘true’ si consigue añadir el paciente, ‘false’ en caso contrario.
- bool finalizarTratamiento(string dni, string nombre) pone la fecha de finalización de un paciente.
- bool cancelarTratamiento(string dni, string nombre) Cancela un tratamiento.
- datosTratamientos(Tratamiento &t)(void) Introduce manualmente los datos del tratamiento.
- modificarTratamiento(string dni, string nombre, Tratamiento t)(bool) Modifica un tratamiento si este se modifica con exito devuelve ‘true’ y si no ‘false’
- mostrarTratamiento(string dni, string nombre)(void) devuelve true si existe el tratamiento especifico y lo muestra, false si no.

CLASE: TRATAMIENTO

DESCRIPCIÓN DE LA CLASE:

Esta clase es la clase tratamiento que posee datos como la fecha de inicio, de final y el nombre del tratamiento. Sirve para almacenar la información de un tratamiento de un paciente.

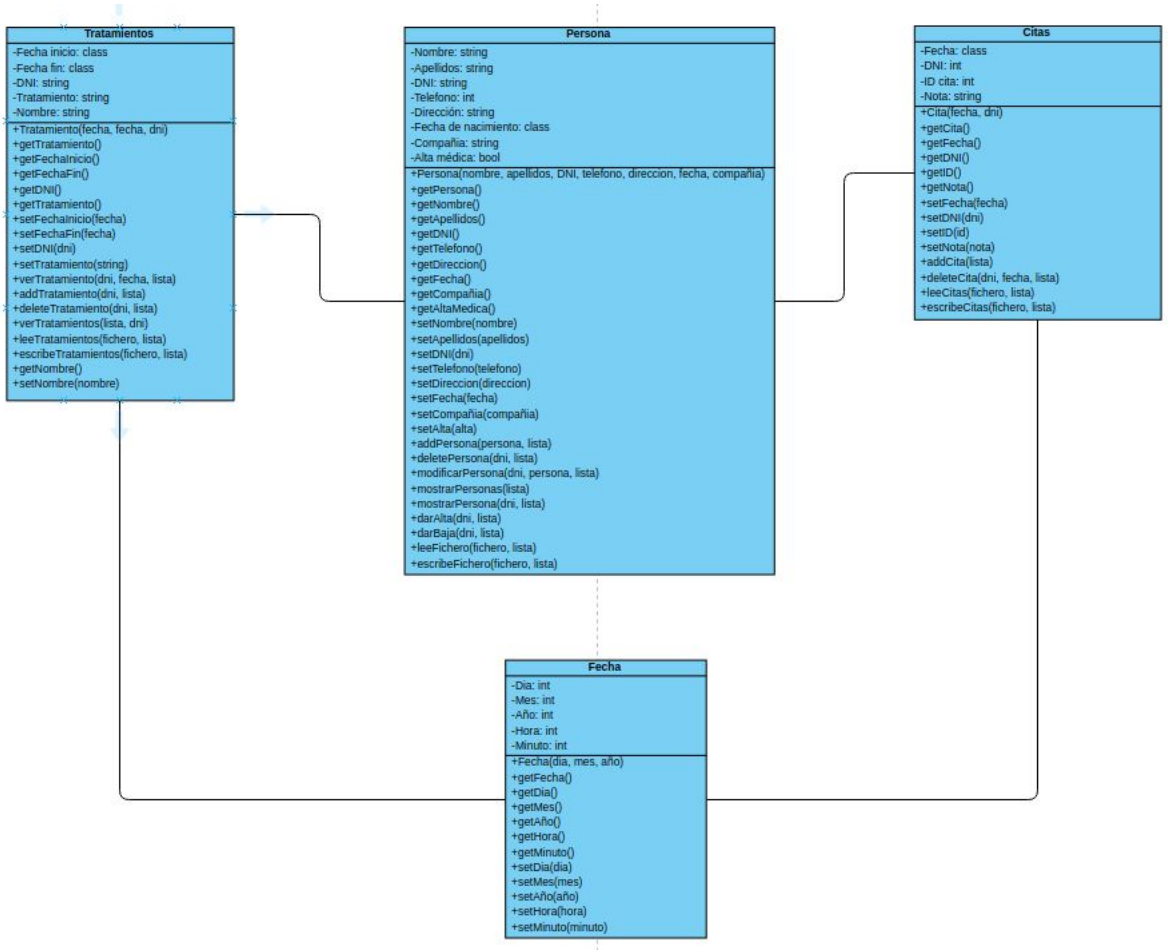
DATOS:

- Nombre *(string)*: nombre del tratamiento
- Fecha de Inicio *(struct fecha)*: fecha de inicio del tratamiento
- Fecha de Fin *(struct fecha)*: fecha en la que se termina el tratamiento
- Tratamiento *(string)* cadena de unos 300 caracteres que guarde detalles sobre el tratamiento en caso de que sea necesario
- DNI *(string)* DNI de la persona a la que pertenece el tratamiento

MÉTODOS:

- Tratamiento() Constructor, recibe por parámetros la fecha de inicio, dni del paciente y el nombre del tratamiento.
- getFechainicio() Observador de la fecha de inicio
- getTratamiento() Observador del tratamiento
- getFechaFin() Observador de la fecha de finalización
- getNombre() Observador del nombre.
- getDNI() Observador del dni.
- setTratamiento(string tratamiento) Modificador del campo tratamiento.
- setNombre(string nombre) Modificador del campo nombre.
- setFechaInicio(class fecha) Modificador del campo fecha inicio.
- setFechaFin(class fecha) Modificador del campo fecha fin.
- setDNI(string dni) Modificador del campo dni.
- deleteTratamiento(string dni, lista) Recibe el dni del paciente y elimina el tratamiento de la lista.

Tras presentar y explicar las clases, nuestro diagrama de clases quedaría de esta forma, complementando y reafirmando lo anteriormente escrito.



7.- Validación de las Clases con los Casos de Uso.

Los casos de uso enumerados serían:

1.- Buscar usuario

2.- Modificar usuario

3.- Añadir Cita

4.-Modificar Cita

5.- Añadir Paciente

6.- Mostrar lista

7.- Visualizar Historial Médico

8.- Visualizar historial de Tratamiento

9.- Añadir Tratamiento

10.- Modificar Tratamiento

11.- Dar de alta

12.- Mostrar Lista de citas de hoy

Y las clases enumeradas serían:

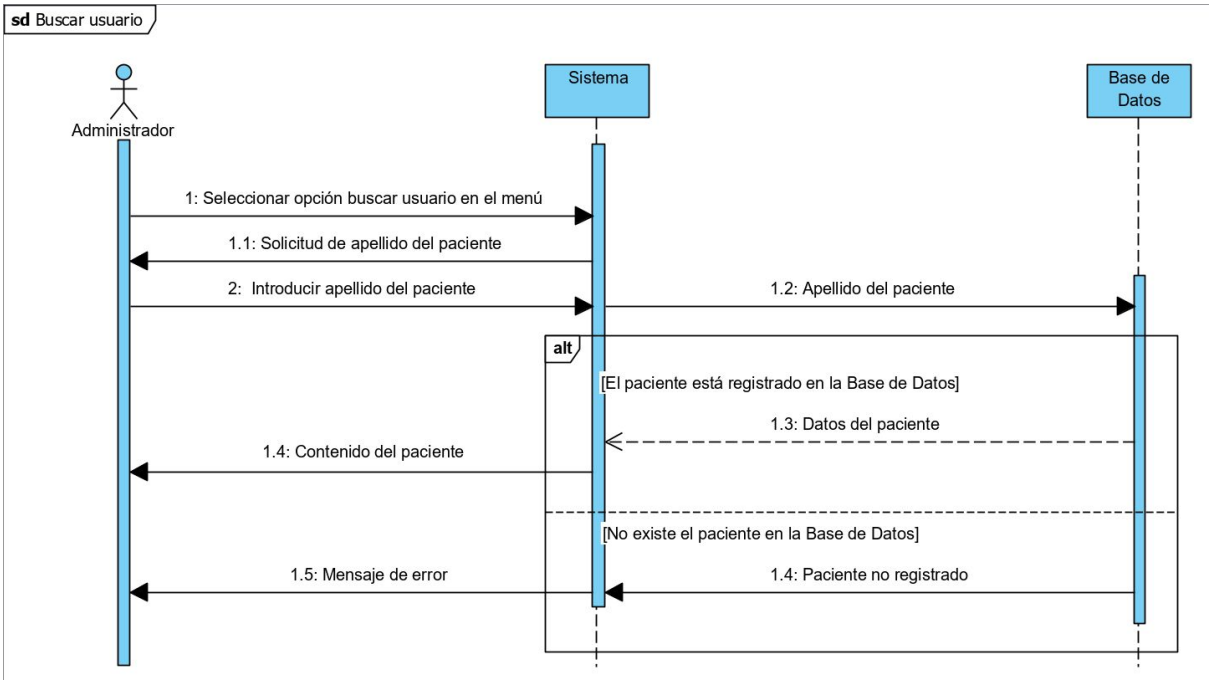
1. *Cita*
2. *Fecha*
3. *Persona*
4. *Tratamiento*

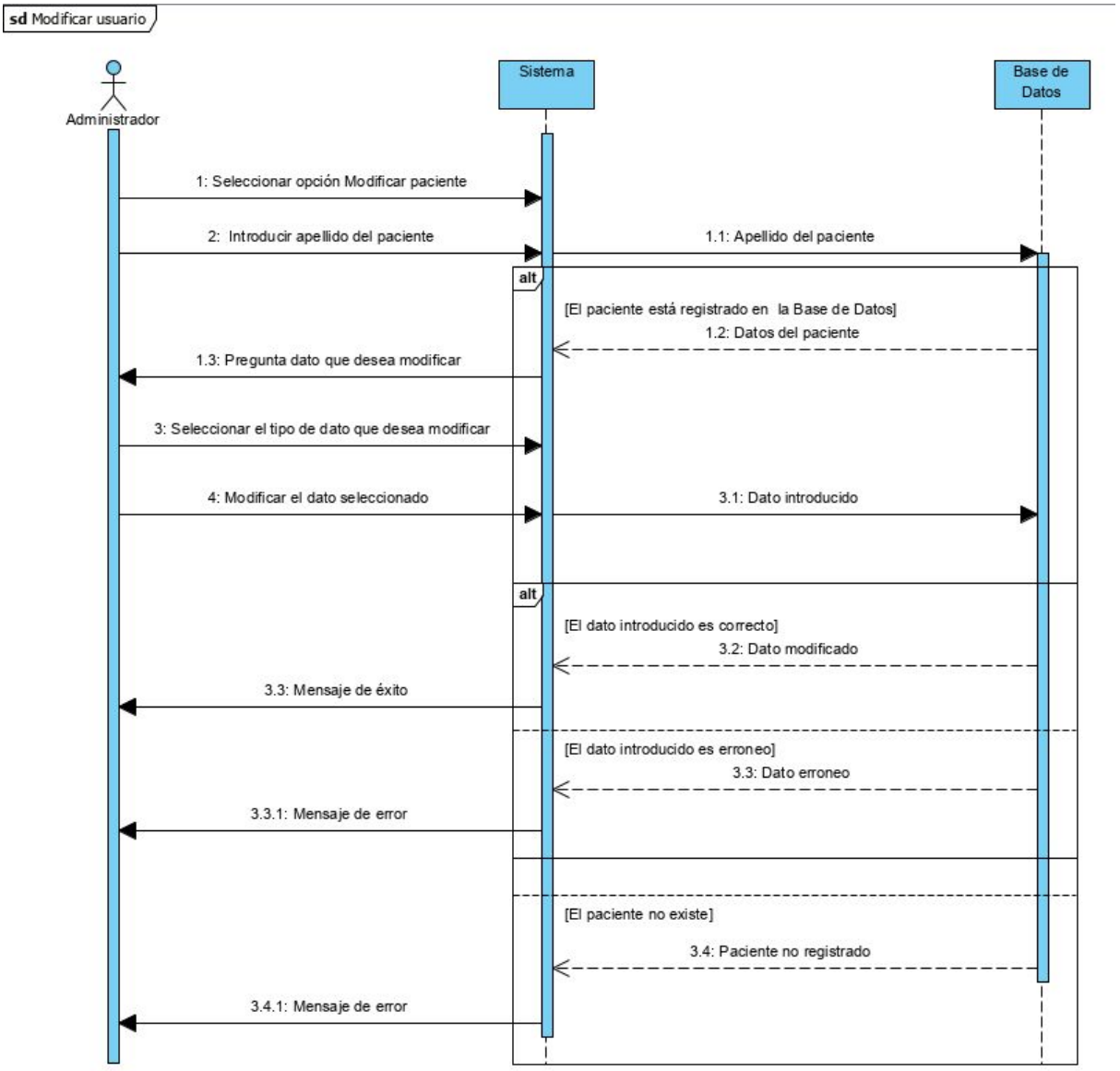
Matriz casos de uso (CU) - clases: cada caso de uso debe tener asignada una clase al menos, en caso contrario, faltar´ıa mejorar la definición de la clase, o la creación de otra.

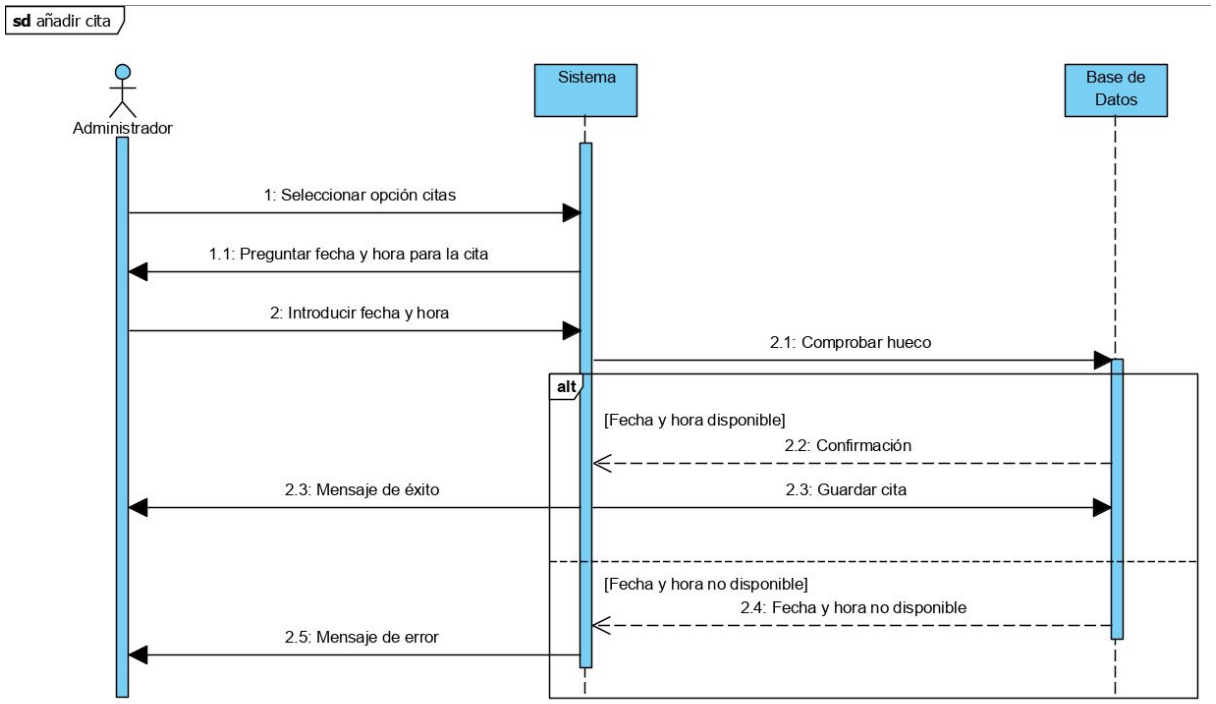
	CASOS DE USO											
CLASES		1	2	3	4	5	6	7	8	9	10	11	12
	1			✓	✓								✓
	2			✓	✓								✓
	3	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓
	4								✓	✓	✓		

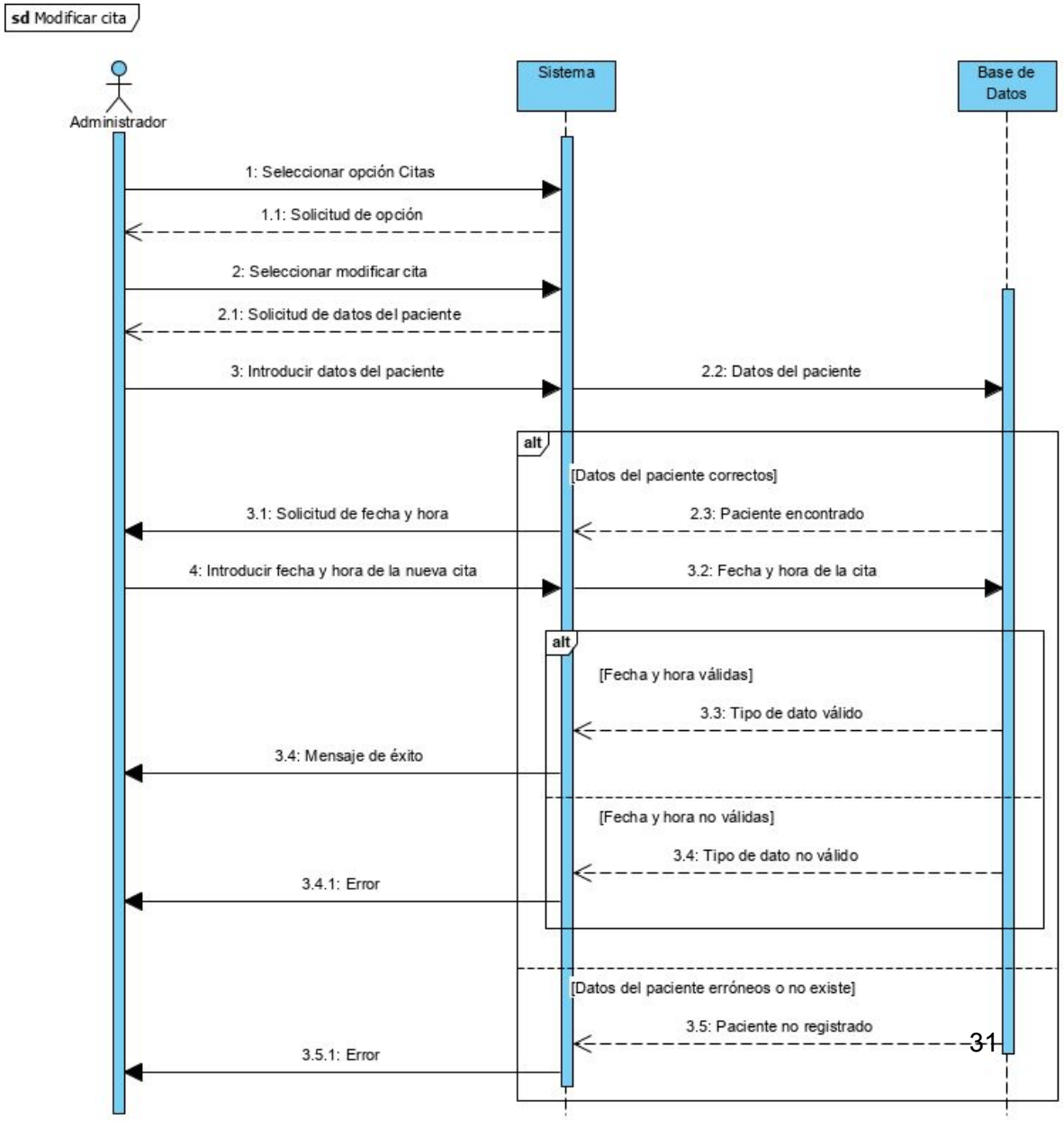
8.- Descripción de los escenarios de interacción que se produzcan en el sistema para llevar a cabo la funcionalidad descrita (Diagramas de Secuencia) para los Casos de Uso más complejos.

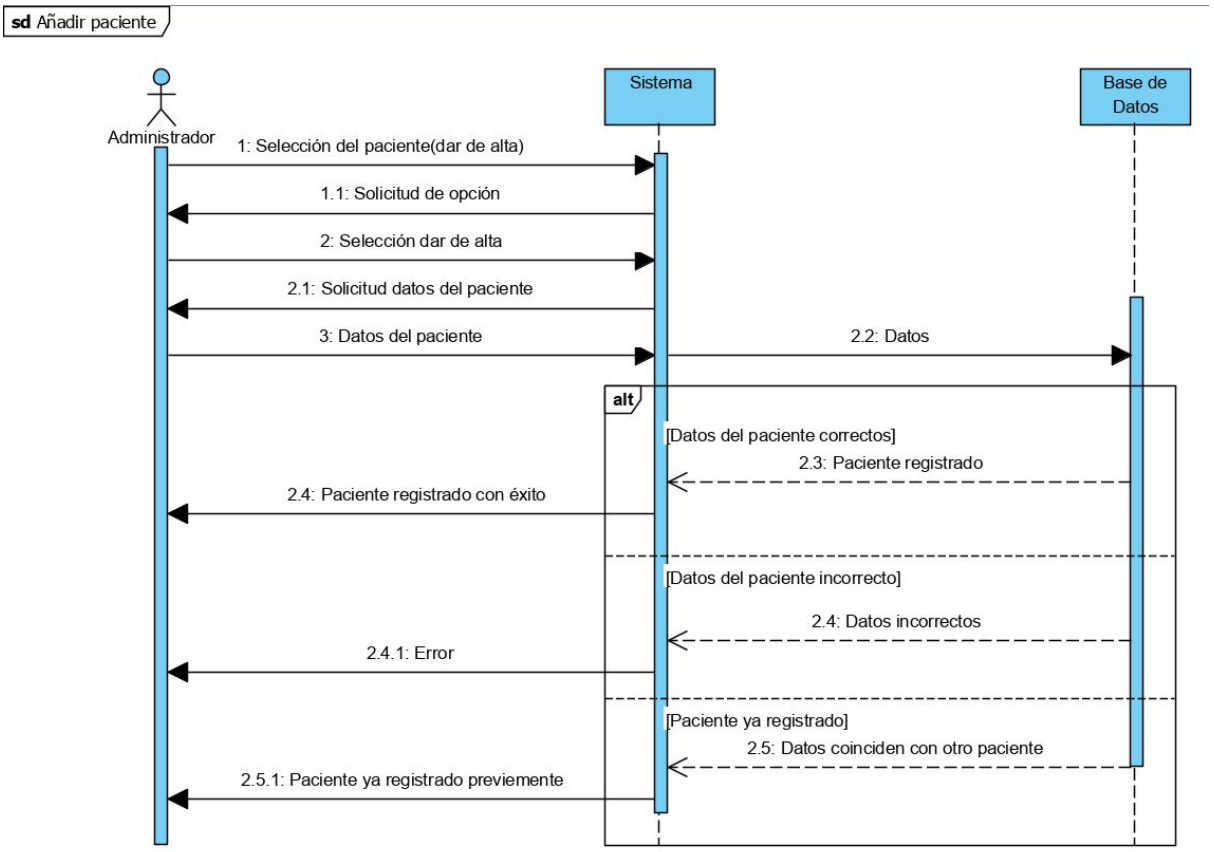
Como sabemos un diagrama de secuencia describe el comportamiento dinámico del sistema y muestra la mecánica de la interacción durante una acción concreta. Los diagramas de secuencia que hemos planteado son:

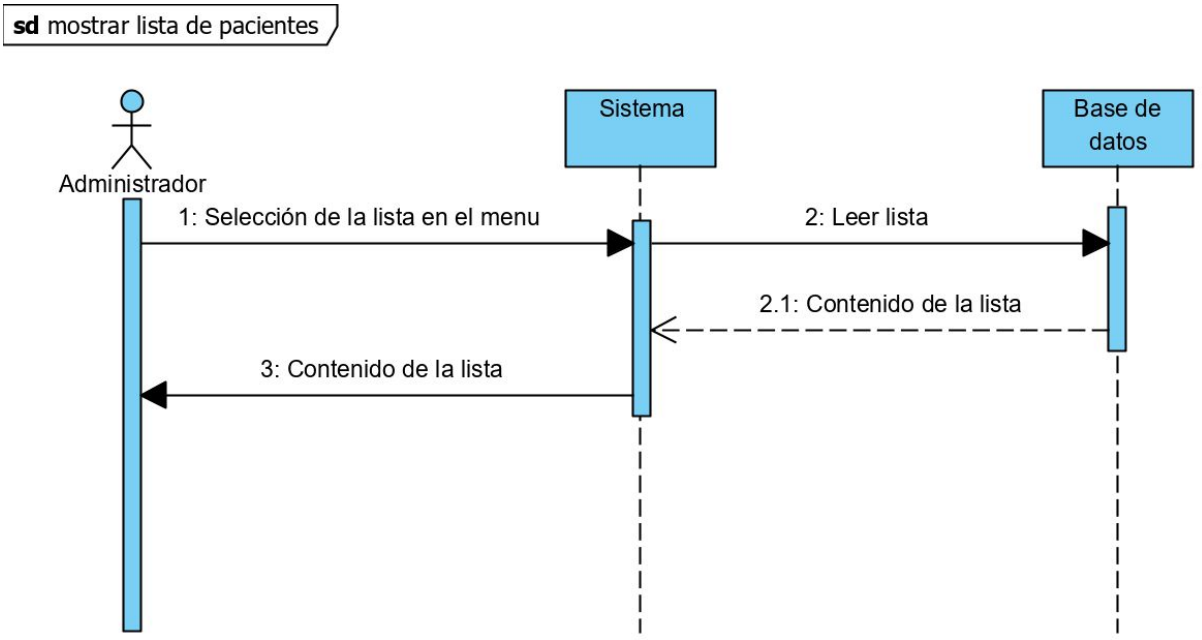


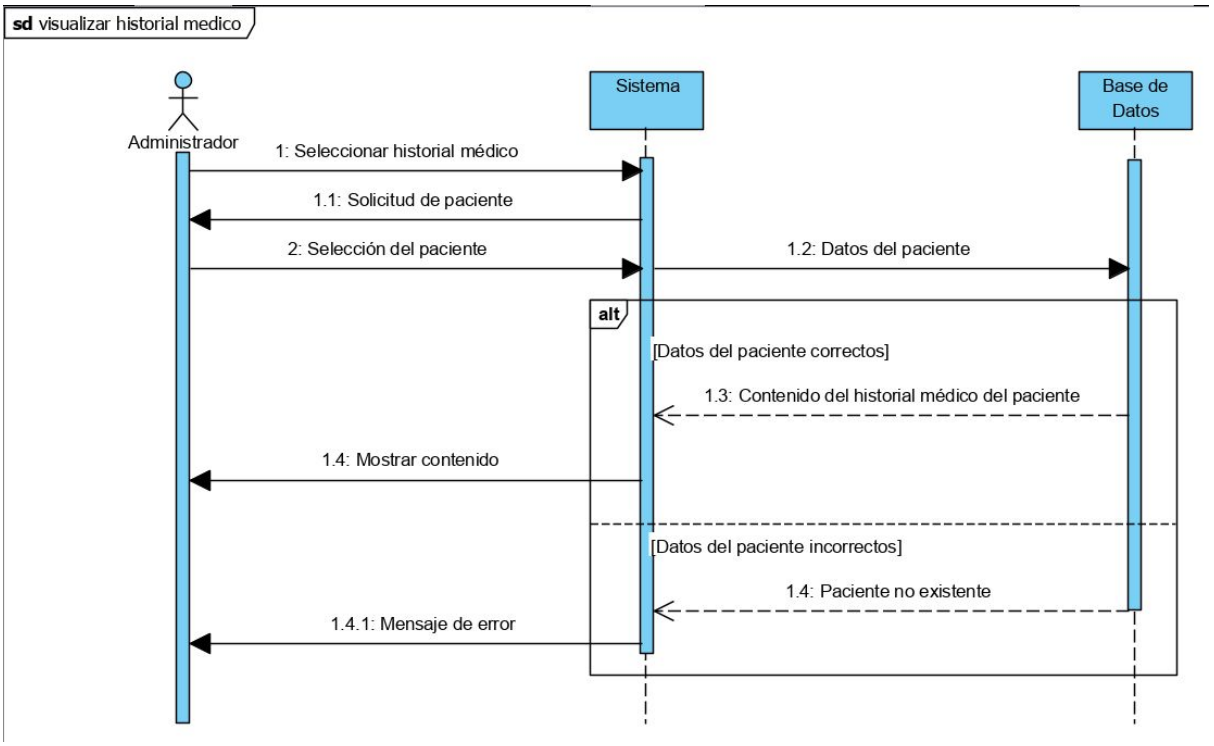

sd Modificar usuario
Administrador
Sistema
Base de Datos
1: Seleccionar opción Modificar paciente
2: Introducir apellido del paciente
1.1: Apellido del paciente
alt
[El paciente está registrado en la Base de Datos]
1.2: Datos del paciente
1.3: Pregunta dato que desea modificar
3: Seleccionar el tipo de dato que desea modificar
4: Modificar el dato seleccionado
3.1: Dato introducido
alt
[El dato introducido es correcto]
3.2: Dato modificado
3.3: Mensaje de éxito
[El dato introducido es erroneo]
3.3: Dato erroneo
3.3.1: Mensaje de error
[El paciente no existe]
3.4: Paciente no registrado
3.4.1: Mensaje de error

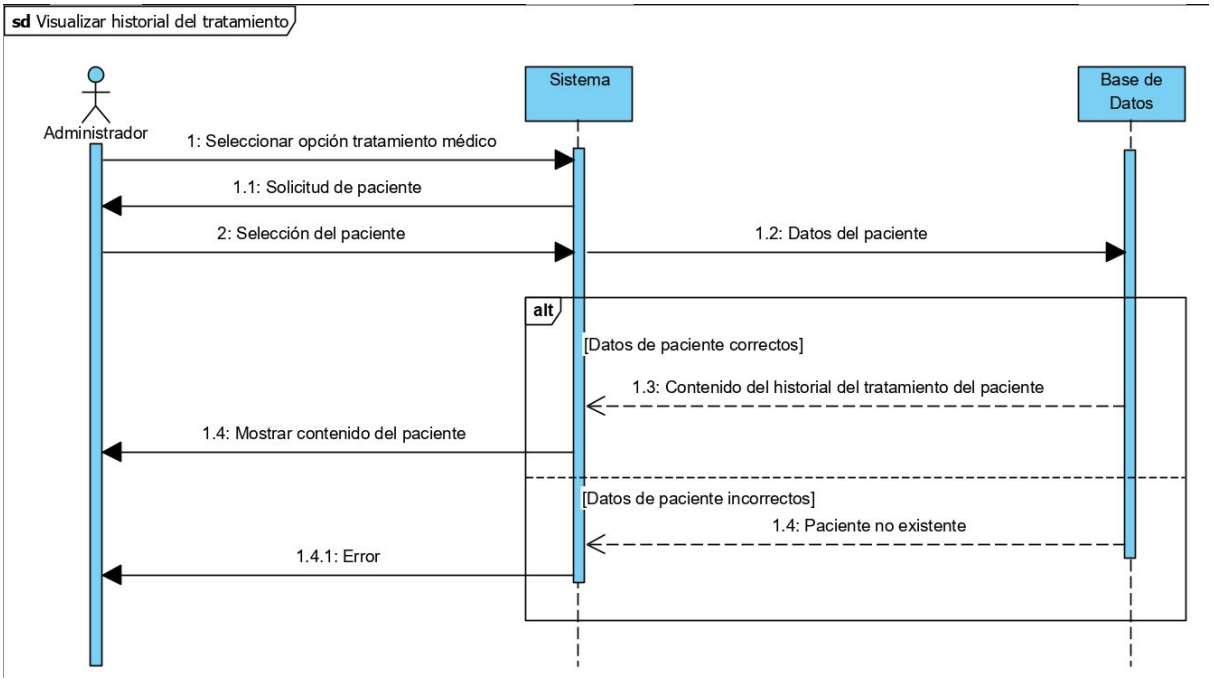

sd añadir cita
Administrador
Sistema
Base de Datos
1: Seleccionar opción citas
1.1: Preguntar fecha y hora para la cita
2: Introducir fecha y hora
2.1: Comprobar hueco
alt
[Fecha y hora disponible]
2.2: Confirmación
2.3: Mensaje de éxito
2.3: Guardar cita
[Fecha y hora no disponible]
2.4: Fecha y hora no disponible
2.5: Mensaje de error

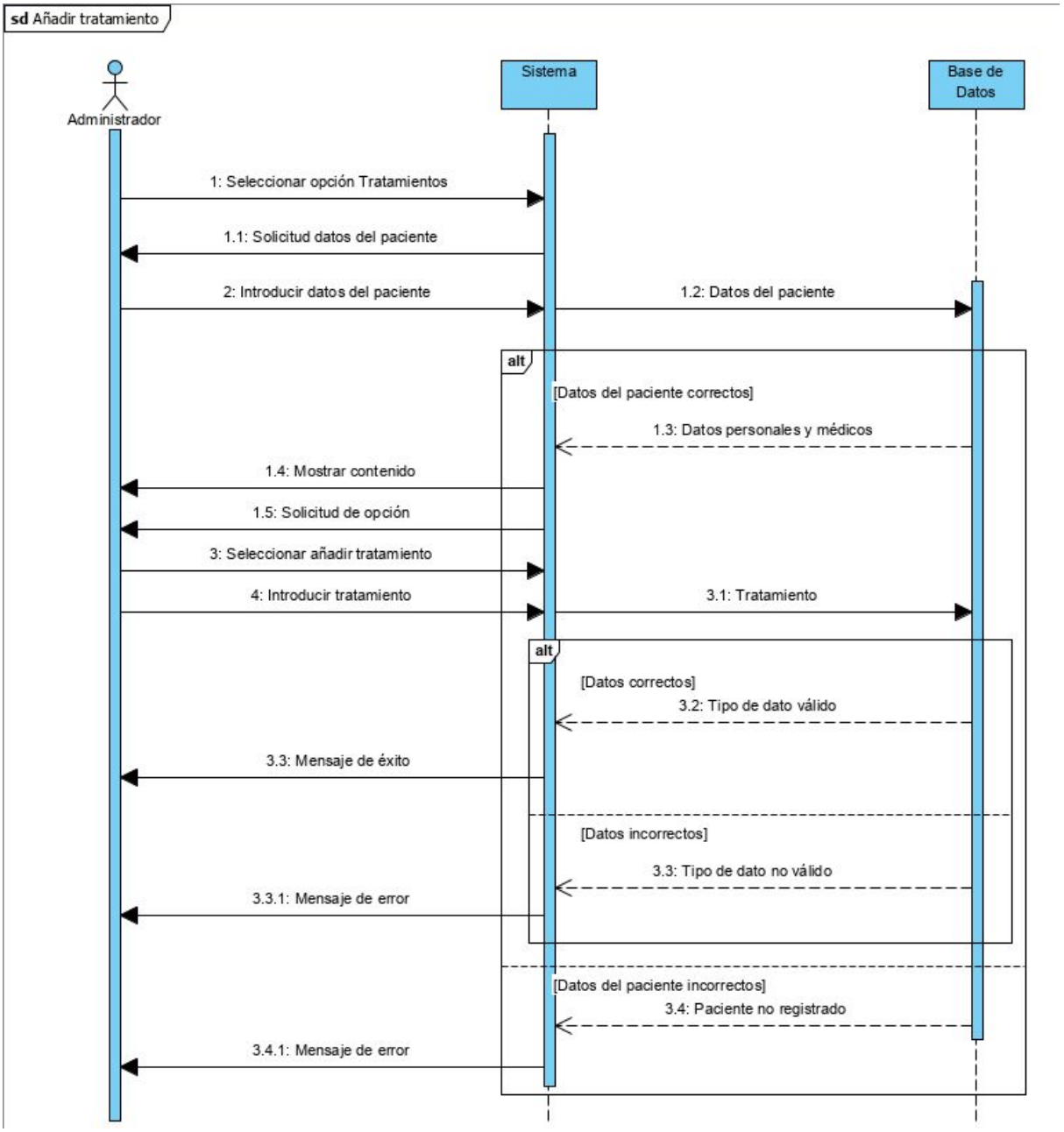

sd Modificar cita
Administrador
Sistema
Base de Datos
1: Seleccionar opción Citas
1.1: Solicitud de opción
2: Seleccionar modificar cita
2.1: Solicitud de datos del paciente
3: Introducir datos del paciente
2.2: Datos del paciente
alt
[Datos del paciente correctos]
3.1: Solicitud de fecha y hora
2.3: Paciente encontrado
4: Introducir fecha y hora de la nueva cita
3.2: Fecha y hora de la cita
alt
[Fecha y hora válidas]
3.3: Tipo de dato válido
3.4: Mensaje de éxito
[Fecha y hora no válidas]
3.4: Tipo de dato no válido
3.4.1: Error
[Datos del paciente erróneos o no existe]
3.5: Paciente no registrado
3.5.1: Error

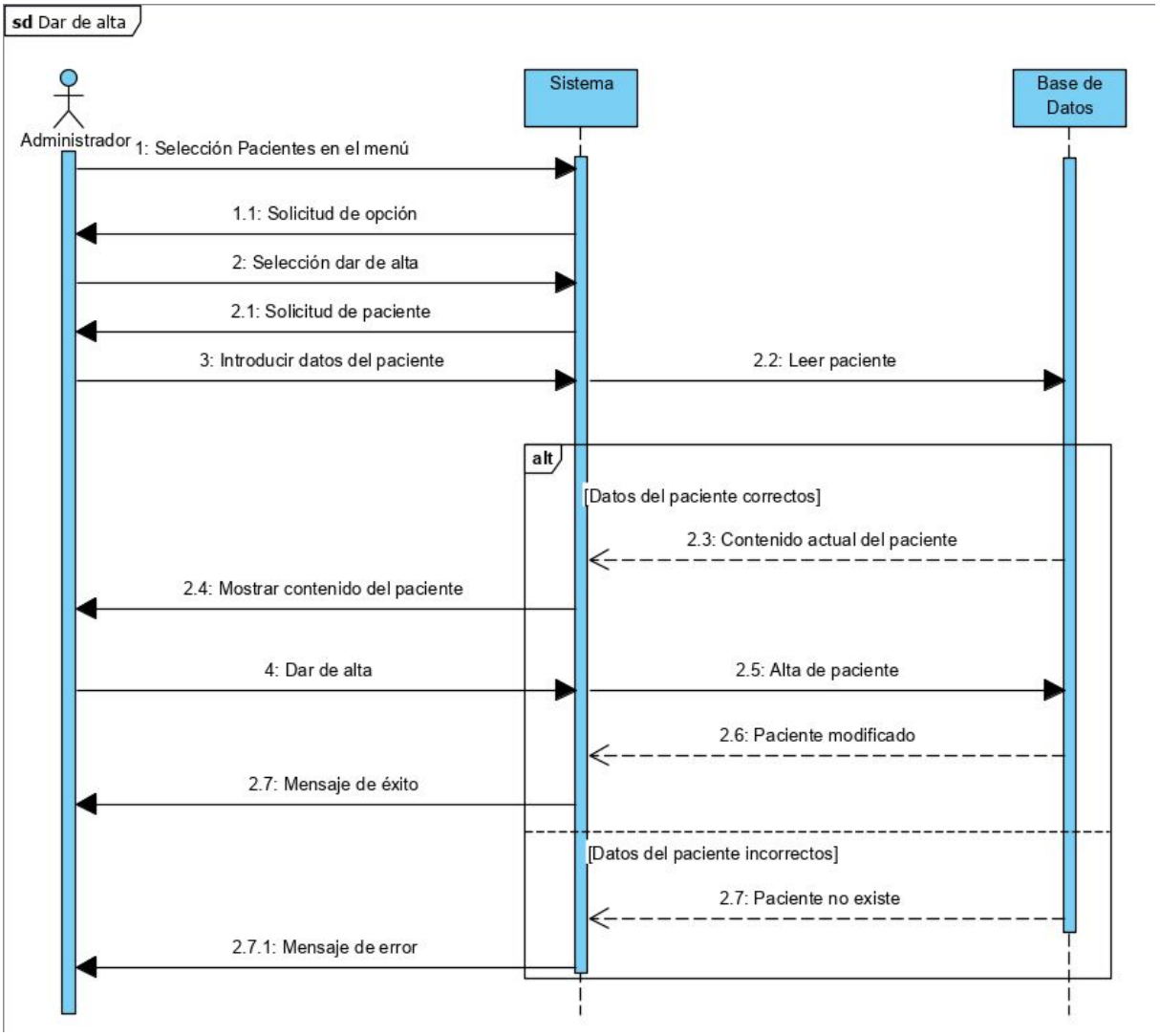

sd Añadir paciente
Administrador
Sistema
Base de Datos
1: Selección del paciente(dar de alta)
1.1: Solicitud de opción
2: Selección dar de alta
2.1: Solicitud datos del paciente
3: Datos del paciente
2.2: Datos
alt
[Datos del paciente correctos]
2.3: Paciente registrado
2.4: Paciente registrado con éxito
[Datos del paciente incorrecto]
2.4: Datos incorrectos
2.4.1: Error
[Paciente ya registrado]
2.5: Datos coinciden con otro paciente
2.5.1: Paciente ya registrado previemente


sd mostrar lista de pacientes
Administrador
Sistema
Base de datos
1: Selección de la lista en el menu
2: Leer lista
2.1: Contenido de la lista
3: Contenido de la lista


sd visualizar historial medico
Administrador
Sistema
Base de Datos
1: Seleccionar historial médico
1.1: Solicitud de paciente
2: Selección del paciente
1.2: Datos del paciente
alt
[Datos del paciente correctos]
1.3: Contenido del historial médico del paciente
1.4: Mostrar contenido
[Datos del paciente incorrectos]
1.4: Paciente no existente
1.4.1: Mensaje de error


sd Visualizar historial del tratamiento
Administrador
Sistema
Base de Datos
1: Seleccionar opción tratamiento médico
1.1: Solicitud de paciente
2: Selección del paciente
1.2: Datos del paciente
alt
[Datos de paciente correctos]
1.3: Contenido del historial del tratamiento del paciente
1.4: Mostrar contenido del paciente
[Datos de paciente incorrectos]
1.4: Paciente no existente
1.4.1: Error


sd Añadir tratamiento
Administrador
Sistema
Base de Datos
1: Seleccionar opción Tratamientos
1.1: Solicitud datos del paciente
2: Introducir datos del paciente
1.2: Datos del paciente
alt
[Datos del paciente correctos]
1.3: Datos personales y médicos
1.4: Mostrar contenido
1.5: Solicitud de opción
3: Seleccionar añadir tratamiento
4: Introducir tratamiento
3.1: Tratamiento
alt
[Datos correctos]
3.2: Tipo de dato válido
3.3: Mensaje de éxito
[Datos incorrectos]
3.3: Tipo de dato no válido
3.3.1: Mensaje de error
[Datos del paciente incorrectos]
3.4: Paciente no registrado
3.4.1: Mensaje de error


sd Dar de alta
Administrador
Sistema
Base de Datos
1: Selección Pacientes en el menú
1.1: Solicitud de opción
2: Selección dar de alta
2.1: Solicitud de paciente
3: Introducir datos del paciente
2.2: Leer paciente
alt
[Datos del paciente correctos]
2.3: Contenido actual del paciente
2.4: Mostrar contenido del paciente
4: Dar de alta
2.5: Alta de paciente
2.6: Paciente modificado
2.7: Mensaje de éxito
[Datos del paciente incorrectos]
2.7: Paciente no existe
2.7.1: Mensaje de error

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

Como sabemos, SCRUM es una metodología ágil que se basa en entregas parciales y frecuentes de un producto para obtener resultados con rapidez. Para poder llevar esto a cabo y haciendo uso del taiga.io, hemos hecho nuestras entregas en 4 partes, o en 4 sprints, en los que poco a poco hemos ido desarrollando paso por paso nuestro programa, remarcando que había que corregir, que cosas eran más importantes que otras y poniéndonos de acuerdo los unos con los otros de manera que cada uno tuviese una tarea y el tiempo conforme a lo que necesitaba.

Nuestras entregas, correcciones y sprints quedan reflejados en el taiga.io que hemos adjuntado junto con este archivo.