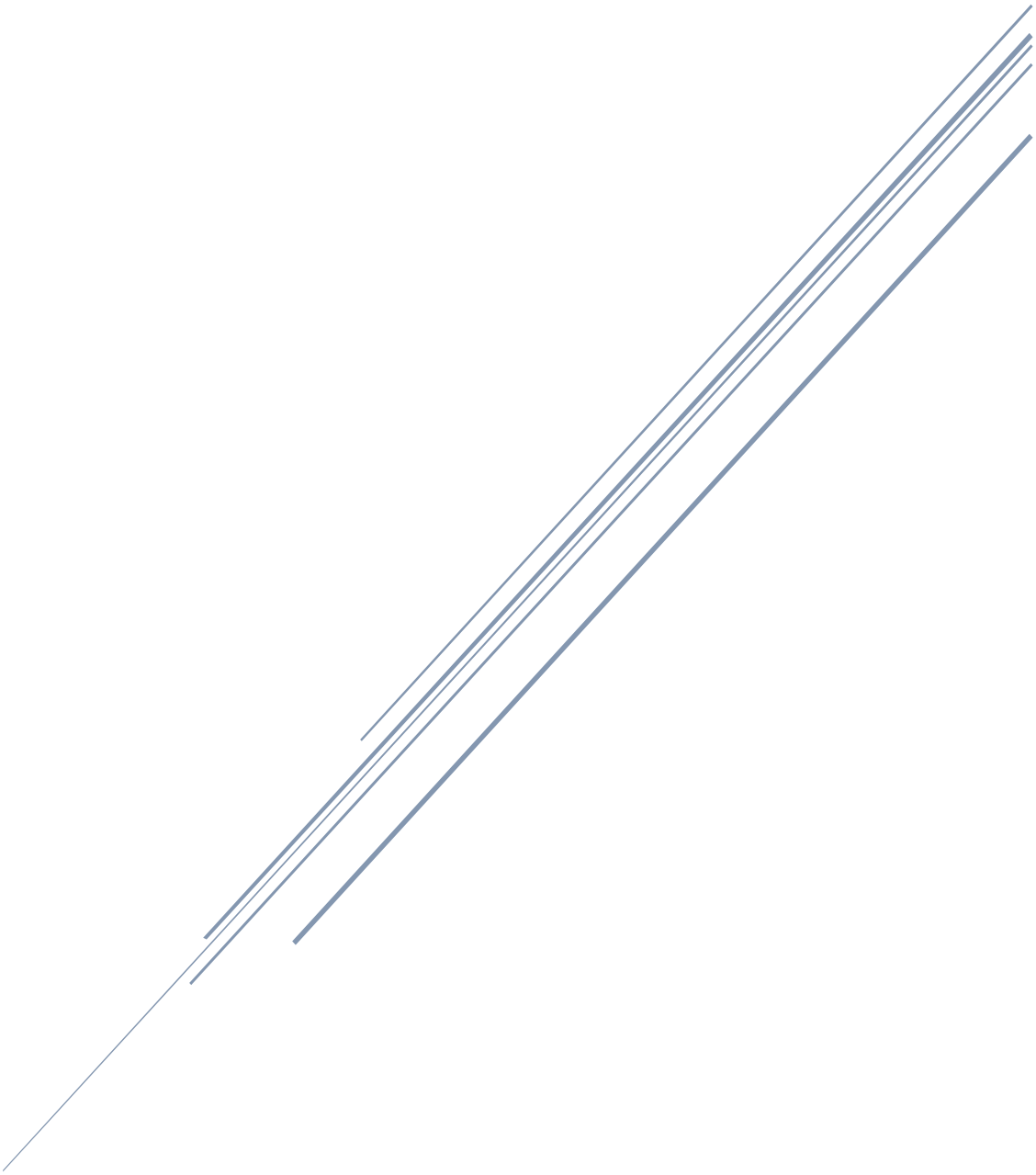
MEMORIA PRACTICAS IS

Agenda electrónica



Alberto Gamez Albarran

Ingenieria del Software. Convocatoria Julio.

Indice

Identificación de requisitos.

Requisitos funcionales

• RF000_DATOS – El sistema permitirá almacenar los siguientes datos de un paciente: Nombre,

Apellidos, DNI, fecha de nacimiento, teléfono y citas.

Citas, es un atributo de múltiple valores, compuesto por los siguientes atributos: fecha, hora, motivo.

• RF001_BUSCAR – El sistema permitirá la búsqueda por nombre de un paciente.

• RF002_INSERTAR – El sistema permitirá modificar los datos de un paciente.

• RF003_MODIFICAR – El sistema permitirá modificar los datos de un paciente.

• RF004_ELIMINAR – El sistema permitirá modificar los datos de un paciente.

• RF005_LISTAR – El sistema permitirá listar por pantalla toda la agenda de pacientes y citas.

• RF006_PERSISTENCIA – El sistema almacenará la agenda en el disco constantemente de manera que esté disponible cuando se vuelva a cargar el programa.

Requisitos no funcionales

• RNF000_INTERFAZ - El programa se utilizará a través del terminal.

• RNF001_LENGUAJE - El programa estará escrito en C++.

• RNF002_CALIDAD - El programa deberá incluir pruebas de unidad.

• RNF003_ALMACENAMIENTO - El programa almacenará el contenido en un fichero de texto.

• RNF004_VOLUMEN – El programa tendrá capacidad para unos 200 pacientes.

• RNF005_SO – El programa deberá funcionar en GNU/Linux

A través de los requisitos funcionales, generamos los casos de uso e historias de usuario, los cuales a su vez, terminaran dando una visión general del programa en forma de diagrama de clases.

Casos de uso

Caso de uso: **Acceso a la interfaz**
ID: 01
Breve descripción: El trabajador podrá iniciar la aplicación usando una interfaz de consola con varias opciones
Actores principales: Trabajador.
Nivel: General
Pre-condiciones: Ninguna.
Flujo principal: 1) El trabajador accede a la aplicación. 2) El trabajador dispone de varias opciones y un menú de salida. 3) El sistema dispondrá de un sistema de copias de seguridad manuales.
Post-condiciones: 1 Ninguna.
Flujos alternativos: Ninguno.

Caso de uso: **Dar de alta paciente**
ID: 02
Breve descripción: Trabajador solicita la inclusión de un nuevo paciente en el sistema.
Actores principales: Trabajador.
Actores secundarios: Paciente.
Pre-condiciones: Ninguna.
Flujo principal: 1) Trabajador accede al sistema 2) Accede al menú y maca la opción de dar de alta paciente. 3) El trabajador introduce los siguientes datos: Nombre, DNI, fecha de nacimiento, teléfono, citas.
Post-condiciones: 2 Preso introducido en Sistema.
Flujos alternativos: Ninguno.

Caso de uso: **Modificar Paciente**
ID: 03
Breve descripción: Trabajador solicita modificar los datos de un paciente.
Actores principales: Trabajador.
Actores secundarios: Paciente.
Pre-condiciones: El paciente ya existe con anterioridad en el sistema.
Flujo principal: 1) El trabajador accede al menú del sistema. 2) Solicita modificar los datos de un paciente ya existente. 3) El sistema solicita al trabajador el DNI del paciente. 4) El sistema devuelve la ficha actual del paciente y permite la modificación de la totalidad de los datos de este. 5) El trabajador modifica los datos que sean necesarios. 6) El sistema muestra mensaje de confirmación.
Post-condiciones: Paciente modificado.
Flujos alternativos: 4.1) El sistema indica que no existe ningún paciente con el DNI facilitado. 4.2) El sistema permite al usuario elegir entre la opción de introducir otro DNI o salir de la opción de modificar paciente.

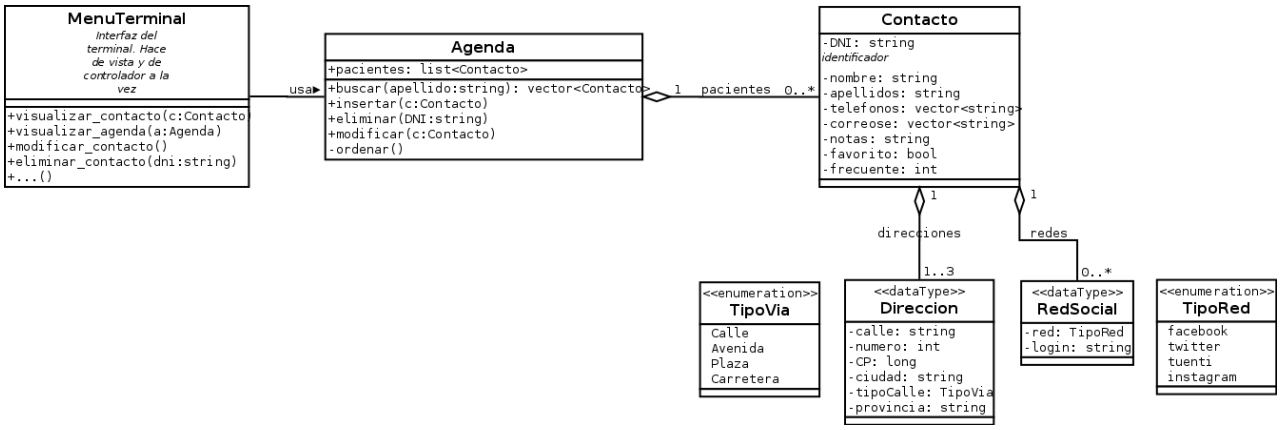
Caso de uso: **Borrar Paciente**
ID: 04
Breve descripción: Trabajador solicita dar de baja un paciente.
Actores principales: Trabajador.
Actores secundarios: Paciente.
Pre-condiciones: Existe el paciente con anterioridad en el sistema.
Flujo principal: 1) El trabajador accede al sistema. 2) Marca la opción de borrado. 3) El sistema solicita el DNI del paciente a eliminar. 4) El trabajador introduce el DNI. 5) El sistema confirmará la decisión de borrar totalmente los datos del paciente. 6) El trabajador confirmará 7) Se mostrara un mensaje de confirmación de borrado.
Post-condiciones: 3 Paciente borrado del sistema.
Flujos alternativos: 4.1) El DNI introducido por el trabajador no existe. 4.2) El sistema permite la opción de introducir nuevo DNI o salir de la opción de borrado. 6.1) El trabajador decide no borrar la ficha del paciente, con lo que se le devuelve al menú.

Caso de uso: **Buscar Paciente**
ID: 05
Breve descripción: Trabajador solicita la búsqueda de un paciente por su DNI.
Actores principales: Trabajador.
Actores secundarios: Paciente.
Pre-condiciones: Existe el paciente con anterioridad en el sistema.
Flujo principal: 1) El trabajador accede al menú del sistema. 2) El trabajador accede a la opción de búsqueda. 3) El sistema solicitará el DNI al trabajador para localizar al paciente. 4) El trabajador introduce el DNI del paciente. 5) El sistema devuelve la ficha del paciente por pantalla. 6) El sistema pasados unos segundos dará la opción de volver al menú principal.
Post-condiciones: Ninguna.
Flujos alternativos: 5.1) El DNI del paciente no existe, con lo cual el sistema dará la opción de introducir otro DNI o salir al menú principal.

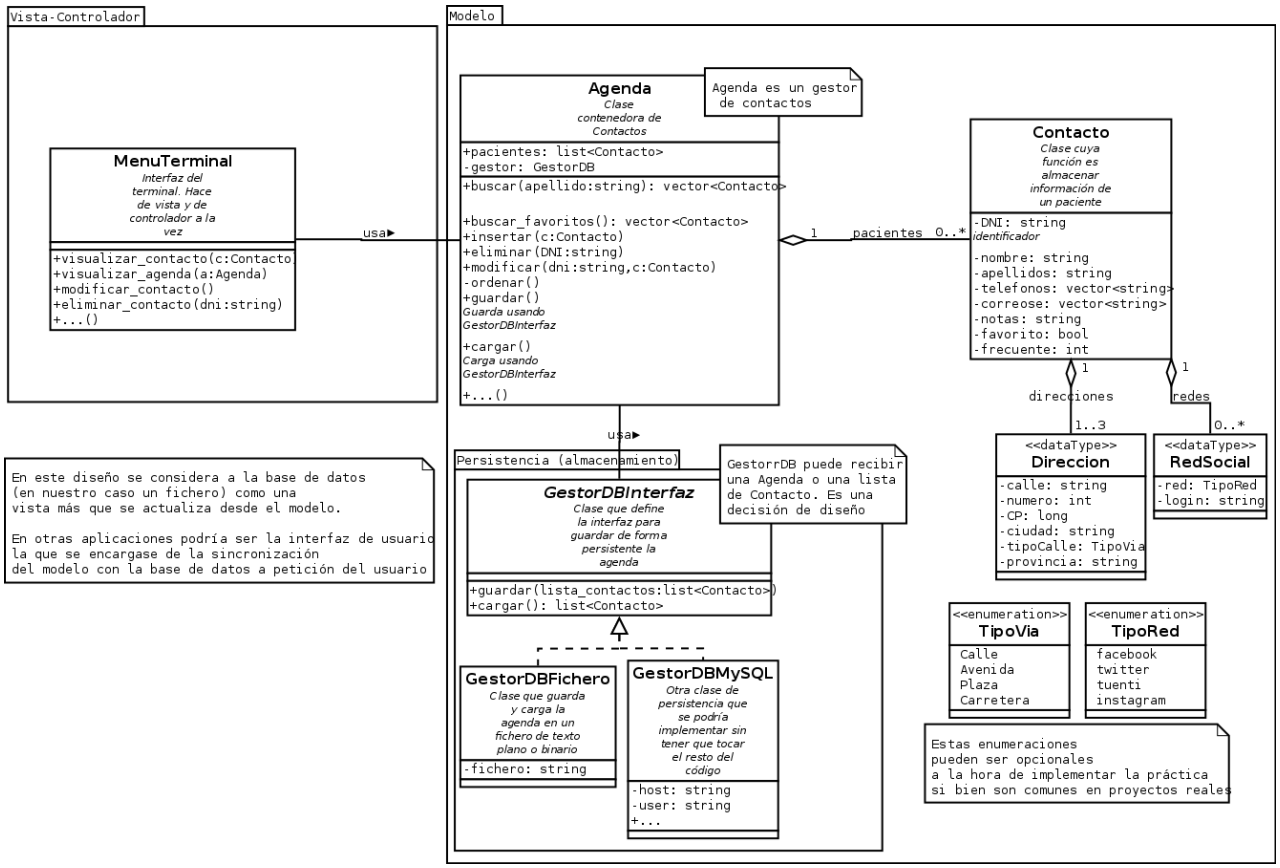
Caso de uso: **Almacenar datos pacientes.**
ID:06
Breve descripción: Trabajador solicita guardado de los datos de la sesión de forma manual.
Actores principales: Trabajador.
Actores secundarios: Ninguno.
Pre-condiciones: Ninguna.
Flujo principal: 1) El trabajador accede al sistema. 2) El trabajador elige la opción de exportación a base de datos en el menú. 3) El sistema mostrará mensaje de confirmación. 4) El trabajador confirmará 5) El sistema mostrará mensaje de validación del guardado de datos.
Post-condiciones: Datos del sistema exportados al fichero.
Flujos alternativos: 4.1) El trabajador no confirma la grabación de datos, con lo cual, el sistema le devuelve al menú principal.

Diagrama de clases

Diagrama de clases básico



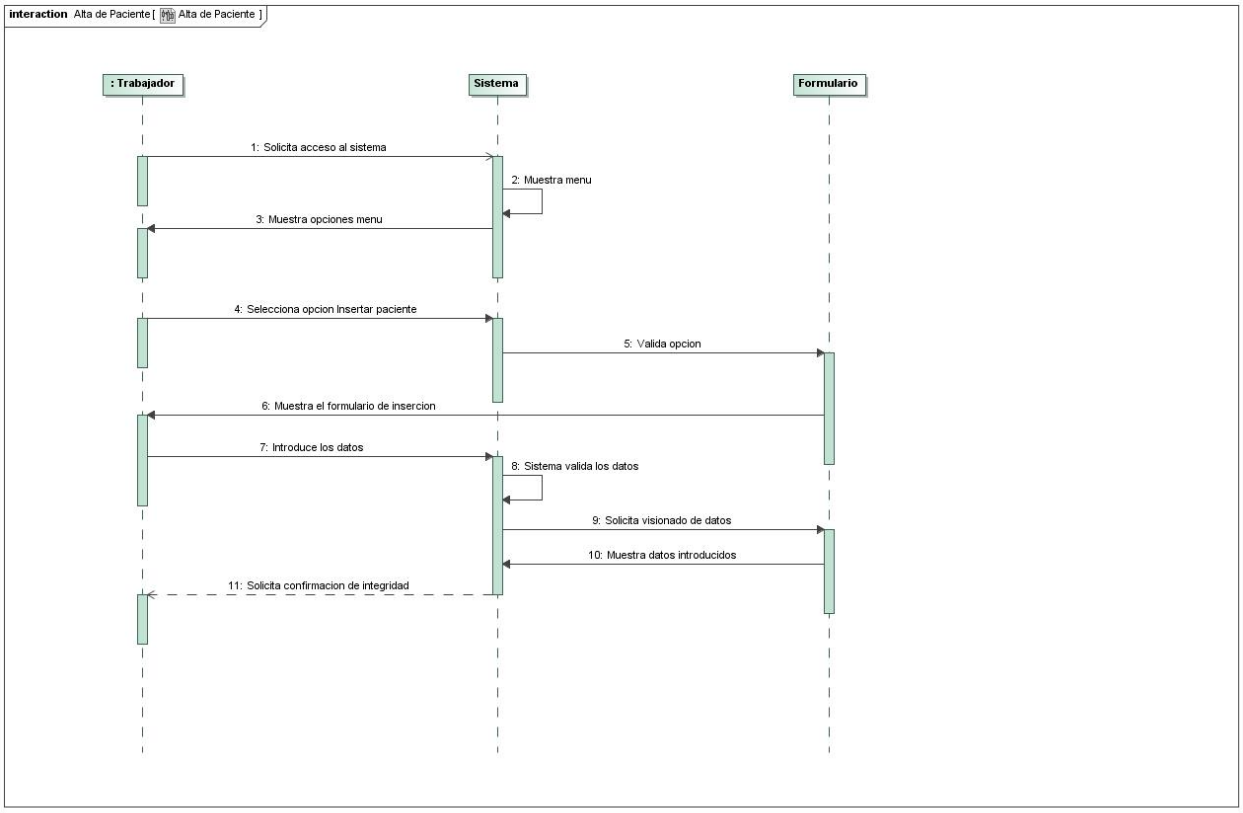
Diagramada de clases aplicando SOLID



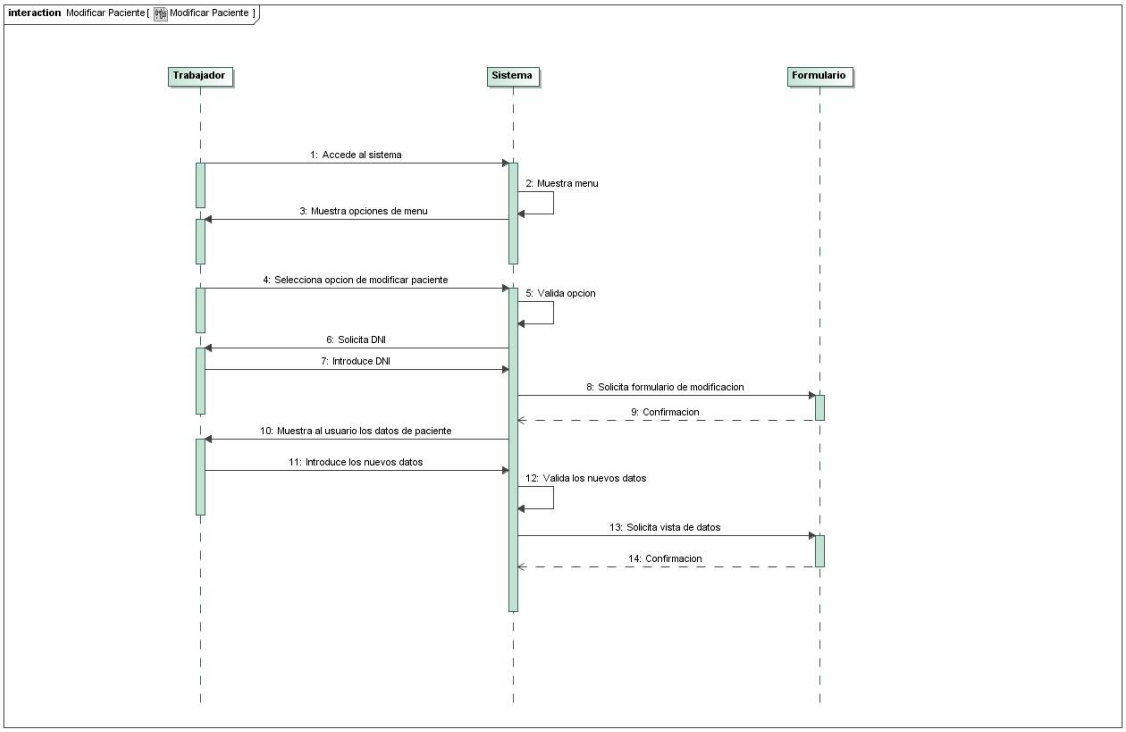
Diagramas de secuencia

Una vez identificados los requisitos tanto funcionales como no funcionales y sus casos de uso, ya podemos generar, en base a estos, los diagramas de secuencia que indicaran como se realizará el flujo de información entre los stakeholders.

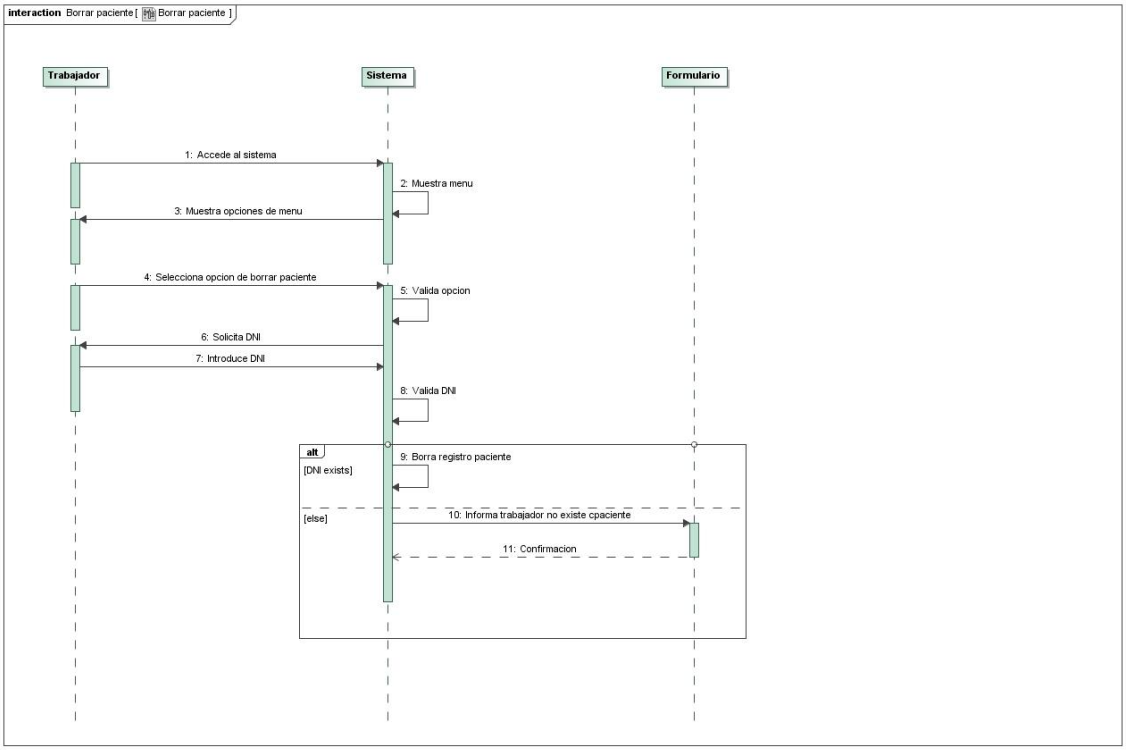
Dar de alta paciente



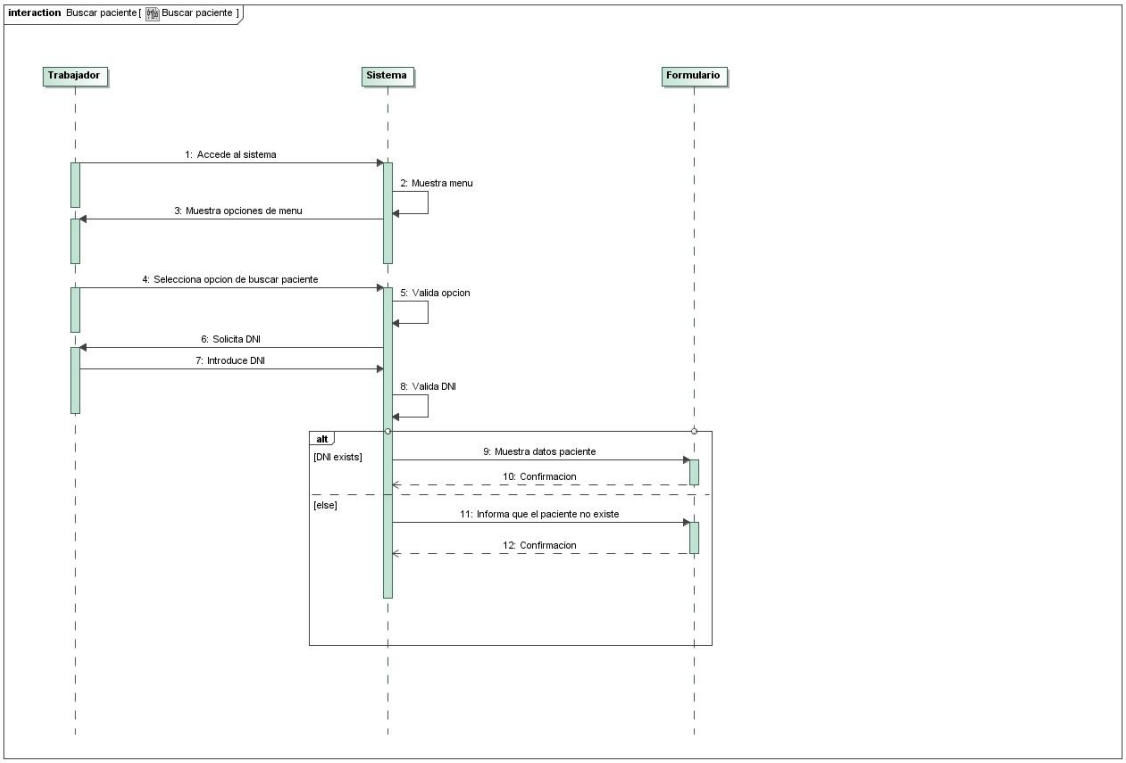
Modificar datos pacientes



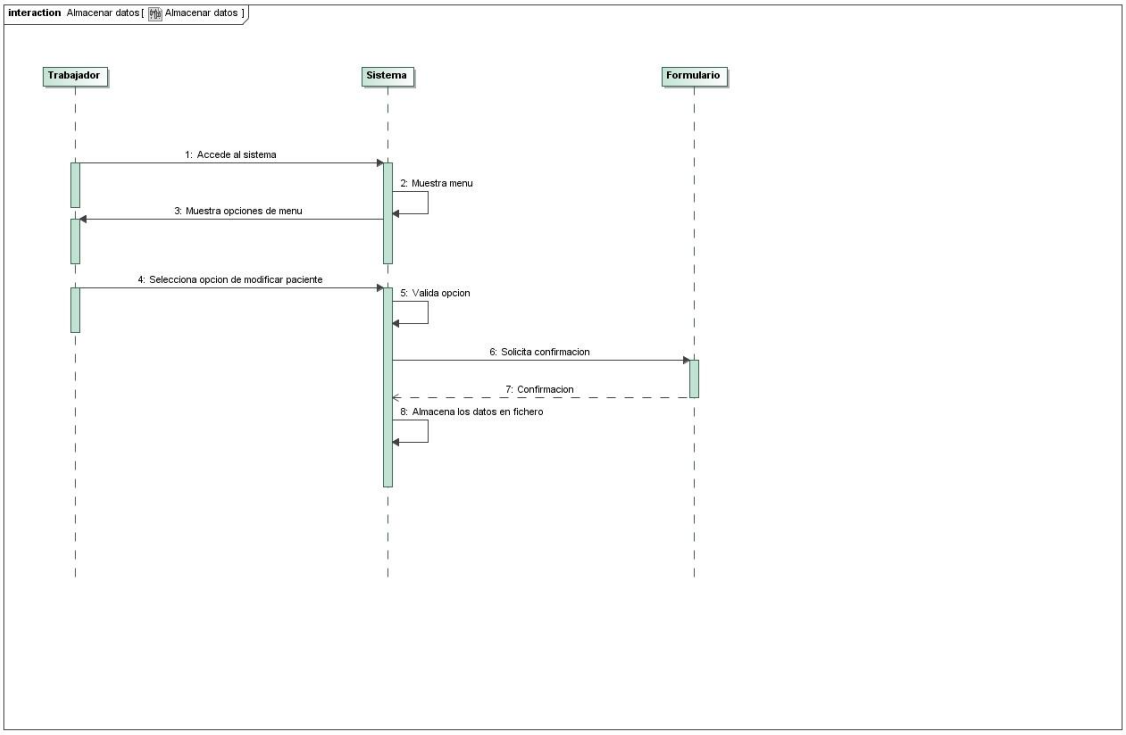
Borrar paciente



Buscar paciente



Almacenar datos



Historias de usuario

Aunque por motivos obvios no se está utilizando una metodología SCRUM para el desarrollo de la práctica, a continuación se expondrán varias historias de usuario para generar un índice de prioridades en la implementación de funcionalidades.

Estas historias de usuario, tendrán una estimación aproximada de tiempo de desarrollo, expresada en minutos y una prioridad ordenada ascendente desde 1 (mínima prioridad) hasta 1000.

Creación de interfaz

ID: R001	Nombre: Interfaz
Estimación: 10	Descripción: Como usuario, necesito una agenda funcional con una interfaz simple que se muestre por consola y que me permita dar de alta, modificar, buscar y borrar mis contactos.
Prioridad: 800	
Pruebas de aceptación:	El cliente dispondrá de una interfaz en consola donde realizar las tareas descritas, después de cualquier iteración se le devolverá automáticamente a esta para poder continuar con la ejecución, también dispondrá de una opción de salida y almacenamiento.
Dependencias:	RNF000, RF005

Funcionalidad de alta de paciente

ID: R002	Nombre: Alta de paciente
Estimación: 30	Descripción: Como usuario, necesito poder dar de alta a mis clientes en la agenda, indicando los principales campos como: Nombre, apellidos, Fecha de nacimiento, Teléfono, Email, Redes y un campo libre para concretar la cita.
Prioridad: 600	
Pruebas de aceptación:	El cliente accederá a un formulario de alta donde podrá dar de alta a sus pacientes de forma rápida y sencilla, estos cambios no se guardaran de forma automática. Comprobaremos la consistencia de datos que se guardan en el fichero, así como su posibilidad de utilización por el resto de funciones. Si al dar de alta el cliente el sistema devuelve error, se informará si es por integridad de datos, de no serlo se

	mostrará un mensaje para que contacte con su administrador.
Dependencias:	RF000, RF001,RF002

Modificar paciente

ID: R003	Nombre: Modificar paciente
Estimacion: 30	Descripcion: Como usuario, necesito tener la funcionalidad de poder modificar un paciente, tras haberlo dado de alta previamente, ya sea para actualizar algún dato o simplemente corregir un error de entrada.
Prioridad: 200	
Pruebas de aceptación:	El cliente dispondrá de una interfaz en consola donde podrá acceder a una opción de modificación a través del DNI del paciente, una vez dentro, deberá confirmar la modificación de los datos del paciente y de ser así, se modificará la ficha completa de este, reescribiendo todos los datos.
Dependencias:	RNF000, RF001, RF003

Borrar paciente

ID: R004	Nombre: Borrar paciente
Estimacion: 10	Descripcion: Como usuario necesito poder eliminar los registros de los pacientes en caso de que se prevea que no volverá a necesitar nuestro servicio.
Prioridad: 100	
Pruebas de aceptación:	El cliente dispondrá de una interfaz en consola donde podrá dar de baja a un paciente, se mostrará mensaje de confirmación antes de darlo de baja de forma definitiva.
Dependencias:	RNF000, RF001,RF002,RF003

Buscar paciente

ID: R005	Nombre: Buscar paciente
Estimacion: 60	Descripcion: Como usuario necesito acceder a los registros de los pacientes y poder visualizarlos en pantalla.
Prioridad: 700	
Pruebas de aceptación:	El cliente dispondrá de una interfaz en consola donde podrá visualizar los datos del paciente, accederá a la ficha en concreto a través del DNI, el mismo algoritmo de búsqueda de paciente se utilizará en el resto de funcionalidades del sistema.
Dependencias:	RF000,RF001,RF002,RF003,RF004,RF005

Almacenar datos

ID: R006	Nombre: Almacenar datos
Estimacion: 30	Descripcion: Como usuario necesito que los datos que genere, se mantengan en el sistema de forma semiautomática.
Prioridad: 400	
Pruebas de aceptación:	El cliente dispondrá de una opción dentro de la interfaz del programa para poder guardar los datos siempre que lo desee, además al salir el sistema preguntará como medida de seguridad si desea realizar un guardado de los datos. Este se realizar en un fichero binario.
Dependencias:	RF000,RF001,RF002,RF003,RF004,RF005

En base a las historias de usuario podemos generar la siguiente tabla de prioridades (back log):

Historia	Importancia	Descripcion
Interfaz	800	Generación interfaz sistema
Dar de alta paciente	600	Alta paciente en sistema
Modificar paciente	200	Modificar los datos del pac.
Borrar paciente	100	Eliminar paciente de sistema
Buscar paciente	700	Buscar paciente en sistema
Almacenar datos	400	Almacenar datos en fich.

Como se puede observar en el back log, hemos asignado la máxima prioridad a la generación de la interfaz, desde la cual podremos ir montando el resto de funcionalidades de la agenda, en segundo lugar buscar paciente, debido a que esta funcionalidad la considero critica al ser necesaria en el resto de componentes, dar de alta paciente se le asigna una prioridad alta,

porque es necesario tener un cliente creado al menos para poder ir realizando las pruebas de integridad y sobre todo para que el resto de las funciones tengan utilidad. Por otro lado, tenemos las funciones de borrar paciente y modificar paciente, con muy bajo nivel de prioridad, debido a que aunque son necesarias para la entrega final al cliente, no lo son tanto de cara a las pruebas y a una versión inicial del proyecto.

Codigo C++ practica

```
#include<stdio.h>

#include<conio.h>

#include<iostream>

#define bool int

#define true 1

#define false 0

using namespace std;



FILE *archivo, *temporal;


void AGREGAR_PERSONA()

{

	int dni, registro;

	char apellido[20], nombre[20], telefono[15], resp;

	bool no_encontrado;

	do{

		if((archivo = fopen("datos.ag","r")) == NULL)

		{

			cout<<"\n No se Encuentra el Archivo!";

			cout<<"\n\n Pulse una tecla para continuar...";

			getch();

		}else{

			//clrscr();

			no_encontrado = true;
```

```
                cout<<"\n Introduzca el dni: "; cin>> registro;
                while ((!feof(archivo)) && (no_encontrado))
                {
                        fscanf(archivo,"%d %s %s %s", &dni, &apellido, &nombre,
&telefono);
                        if(registro == dni)
                        {
                                no_encontrado = false;
                        }
                }
                fclose(archivo);
                if(no_encontrado)
                {
                        archivo = fopen("datos.ag","a");
                        dni = registro;
                        cout<<" Introduzca el Apellido: "; cin>> apellido;
                        cout<<" Introduzca el Nombre: "; cin>> nombre;
                        cout<<" Introduzca el Telefono: "; cin>> telefono;
                        fprintf(archivo,"%d %s %s %s\n", dni, apellido, nombre,
telefono);
                }else{
                        cout<<"\n El paciente ya existe en la agenda";
                }
                cout<<"\n\n Desea registrar otro paciente? S/N: "; resp = getch();
                fclose(archivo);
        }
    }while((resp == 's') || (resp == 'S'));
```

```
} //Fin del procedimiento AGREGAR_PERSONA


void CONSULTAR_REGISTRO()

{

	int registro, dni; // Se declaran enteros porque si se dejan como string luego no se puede hacer la comparacion normal sino por comparacion de cadenas.

	char apellido[20], nombre[20], telefono[15], resp;

	bool no_encontrado;

	do{

		if((archivo = fopen("datos.ag","r")) == NULL)

		{

			cout<<"\n No se Encuentra el Archivo!";

			cout<<"\n\n Pulse una tecla para continuar...";

			getch();

		}else{

			//clrscr();

			no_encontrado = true;

			cout<<"\n Introduzca la dni a Consultar: "; cin>>registro;

			while((!feof(archivo)) && (no_encontrado))

			{

				fscanf(archivo,"%d %s %s %s", &dni, &apellido, &nombre, &telefono);

				if(registro == dni)

				{

					no_encontrado = false;

				}

			}
```

```
                        if(no_encontrado)
                        {
                                cout<<"\n No existe ningun paciente con el DNI indicado\n\n";
                        }else{
                                cout<<"\n Paciente encontrado \n\n";
                                cout<<" dni:   " <<dni <<"\n";
                                cout<<" Apellido: " <<apellido <<"\n";
                                cout<<" Nombre:   " <<nombre <<"\n";
                                cout<<" Telefono: " <<telefono <<"\n\n";
                        }
                        cout<<" Desea realizar otra consulta? S/N: "; resp = getch();
                        fclose(archivo);
                }
        }while((resp == 's') || (resp == 'S'));
} //Fin del procedimiento CONSULTAR_REGISTRO


void ELIMINAR_PERSONA()
{
        int registro, dni;
        char resp, apellido[20], nombre[20], telefono[15], eliminar;
        bool no_encontrado;
        do{
                if ((archivo = fopen("datos.ag","r")) == NULL)
                {
                        cout<<"\n No se encuentra el archivo";
                        cout<<"\n\n Pulse una tecla para continuar...";
```

```
                getch();
        }else{
                //clrscr();
                no_encontrado = true;
                cout<<"\n Introduzca el DNI del paciente: "; cin>>registro;
                while((!feof(archivo)) && (no_encontrado))
                {
                        fscanf(archivo,"%d %s %s %s", &dni, &apellido, &nombre,
&telefono);
                        if(registro == dni)
                        {
                                no_encontrado = false;
                        }
                }
                if(no_encontrado)
                {
                        cout<<"\n No existe ningun registro con el DNI indicado \n\n";
                }else{
                        cout<<"\n Registro encontrado\n\n";
                        cout<<" DNI:   " <<dni <<"\n";
                        cout<<" Apellido: " <<apellido <<"\n";
                        cout<<" Nombre:   " <<nombre <<"\n";
                        cout<<" Telefono: " <<telefono <<"\n\n";
                        fclose(archivo);
                        cout<<" Desea eliminar este paciente? S/N: "; eliminar =
getch();
                }
```

```
                    if ((eliminar == 's') || (eliminar == 'S'))
                    {
                            if((archivo = fopen("datos.ag","r")) == NULL)
                            {
                                    cout<<"\n No se encuentra el archivo";
                                    cout<<"\n\n Pulse una tecla para continuar...";
                                    getch();
                            }else{
                                    temporal = fopen("temporal.ang","w");
                                    while(!feof(archivo))
                                    {
                                            fscanf(archivo,"%d %s %s %s\n", &dni,
&apellido, &nombre, &telefono);
                                            if (registro != dni)
                                            {
                                                    fprintf(temporal,"%d %s %s %s\n", dni,
apellido, nombre, telefono);
                                            }
                                    }
                                    fclose(temporal);
                                    fclose(archivo);
                                    if((temporal = fopen("temporal.ang","r")) == NULL)
                                    {
                                            cout<<"\n ERROR: No se encuentra el archivo";
                                            cout<<"\n\n Pulse una tecla para continuar...";
                                            getch();
                                    }else{
```

```
                                                archivo = fopen("datos.ag","w");
                                                while(!feof(temporal))
                                                {
                                                        fscanf(temporal,"%d %s %s %s\n",
&dni, &apellido, &nombre, &telefono);
                                                        fprintf(archivo,"%d %s %s %s\n", dni,
apellido, nombre, telefono);
                                                }
                                                fclose(temporal);
                                                fclose(archivo);
                                                cout<<"\n\n Se ha eliminado el registro";
                                                //Vaciamos el Archivo temporal
                                                temporal = fopen("temporal.ang","w");
                                                fclose(temporal);
                                        }
                                }
                        }
                        cout<<"\n\n ¿Desea eliminar otro registro? S/N: "; resp = getch();
                }
        }while((resp == 's') || (resp == 'S'));
} //Fin del procedimiento ELIMINAR_REGISTRO

void VACIAR_AGENDA()
{
        char resp;
        cout<<"\n\n Se procedera al borrado total de la agenda, ¿Esta seguro? S/N: "; cin>>
resp;
```

```
	if ((resp == 's') || (resp == 'S'))
	{
		if((archivo = fopen("datos.ag","w")) == NULL)
		{
			cout<<"\n No se encuentra el archivo";
			cout<<"\n\n Pulse una tecla para continuar...";
			getch();
		}else{
			fclose(archivo);
			cout<<"\n Datos de agenda eliminados";
			cout<<"\n\n Pulse una tecla para continuar...";
			getch();
		}
	}
} //Fin del procedimiento VACIAR_AGENDA

void DESPLEGAR_AGENDA()
{
	char dni[8], apellido[20], nombre[20], telefono[15];
	if((archivo = fopen("datos.ag","r")) == NULL)
	{
		cout<<"\n No se encuentra el archivo";
		cout<<"\n\n Pulse una tecla para continuar...";
		getch();
	}else{
		//clrscr();
		cout<<"\n";
```

```
                cout<<"  DNI       APELLIDO       NOMBRE       TELEFONO\n\n";
                while(!feof(archivo))
                {
                        fscanf(archivo,"%s %s %s %s\n", &dni, &apellido, &nombre,
&telefono);
                        cout<<" "<<dni<<"     "<<apellido<<"        "<<nombre<<"
"<<telefono<<"\n";
                }
                fclose(archivo);
                cout<<"\n\n Pulse una tecla para continuar...";
                getch();
        }
} //Fin del procedimiento DESPLEGAR_AGENDA

void MENU()
{
        cout<<"\n  AGENDA ELECTRONICA \n";
        cout<<" 1- Agregar un paciente \n";
        cout<<" 2- Buscar un paciente \n";
        cout<<" 3- Listar todos los pacientes \n";
        cout<<" 4- Eliminar un paciente \n";
        cout<<" 5- Vaciar la Agenda \n\n";
        cout<<" S- Salir.\n";
} // Fin del procedimiento MENU

int main ()
{
```

```
char op;
bool salir = false;
do{
	//clrscr();
	if ((archivo = fopen("datos.ag","r")) == NULL)
	{
		archivo = fopen("datos.ag","w");
		fclose(archivo);
		cout<<"\n\n Archivo creado con exito\n\n";
		MENU();
		cout<<"\n Seleccione una opcion: "; op = getch();
	}else{
		MENU();
		cout<<"\n Seleccione una opcion: "; op = getch();
	}
	switch (op)
	{
		case '1':
			AGREGAR_PERSONA();
			break;
		case '2':
			CONSULTAR_REGISTRO();
			break;
		case '3':
			DESPLEGAR_AGENDA();
			break;
```

```
                    case '4':
                        ELIMINAR_PERSONA();
                        break;
                    case '5':
                        VACIAR_AGENDA();
                        break;
                    case 's': case 'S':
                        salir = true;
                        break;
                }
        }while(salir == false);
} // Fin del main (void)
```