Documentación AB FINAL

1. Introducción

Este proyecto consiste en un sistema de gestión hospitalaria donde encontramos clases como pacientes, médicos y citas en código C++. Tienen diversas funcionalidades con pequeñas diferencias, registrar, modificar, eliminar, buscar, listar y estados. Además, se pueden manejar citas médicas asignadas a pacientes y médicos únicos. Todos los datos registrados se guardan en archivos para su posterior recuperación.

2. Características principales

Menú Médicos:

- Registro de nuevos médicos al sistema
- Modificación de la información del médico seleccionado
- Eliminación de los médicos seleccionándolos por ID
- Gestión del estado de los médicos (activo/inactivo)
- Buscador de médicos por ID
- Listado de todos los médicos

Menú Pacientes:

- Registro de nuevos pacientes al sistema
- Modificación de la información del paciente seleccionado
- Eliminación de los pacientes seleccionándolos por ID
- Gestión del estado de los pacientes (activo/inactivo)
- Buscador de pacientes por ID
- Listado de todos los pacientes

Menú Citas:

- Registro de nuevas citas
- Modificación de la cita seleccionada
- Eliminación de la cita seleccionada
- Listado de citas de urgencia
- Listado de todas las citas

Funcionalidades globales:

- Validación de fecha
- Validación de género
- Validación de vacío
- Guardado de datos en archivos

3. Requisitos del sistema

Software:

- Compilador C++: Estándar C++11 o superior
- Sistema operativo: Windows, macOS, Linux
- IDE: Visual Studio / Code,CLion, etc

4. Instrucciones de instalación y uso

Instalación:

Puede o clonar el repositorio del proyecto o compilar los archivos fuente a través de los siguientes comandos:

- Clonación: git clone https://github.com/guillemsmk/ABProgramacion cd ABProgramacion
- Compilar archivos fuente: g++ main.cpp medico.cpp paciente.cpp citas.cpp -o sistema.propio

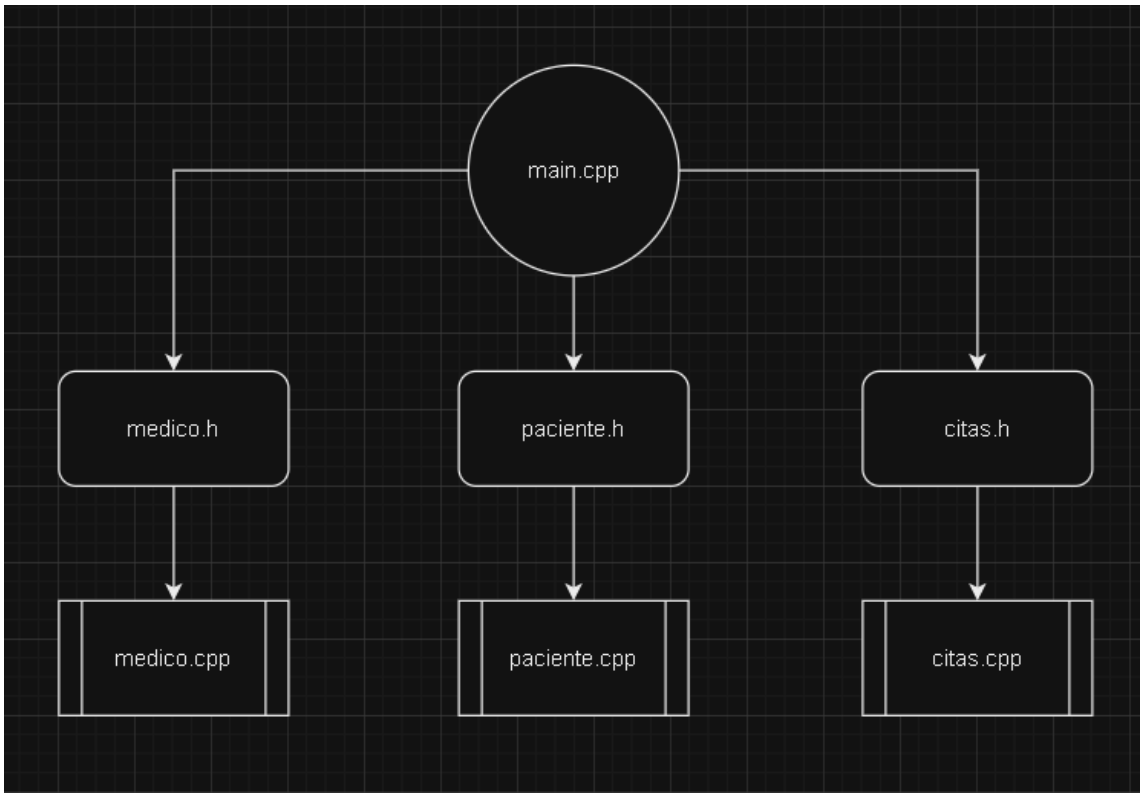
Uso:

Ejecutas el programa y verás una interfaz simple donde simplemente tienes que insertar él número referente a la acción seleccionada. Los datos se guardan automáticamente.

5. Estructura

El archivo principal es el “main.cpp”, contiene lo más principal del código, gestiona todo el trabajo y relaciona las interacciones entre módulos. Posteriormente tenemos por clase un archivo “.cpp” y otro “.h”. Médico contiene los atributos y las funciones específicas de la clase. Paciente funciona igual que médico. Y luego Citas gestiona la programación y los detalles de las citas médicas. El código está estructurado de forma modular facilitando el uso, el mantenimiento y la futura escalabilidad.

6. Diagrama general del proyecto



7. Clases y Funciones

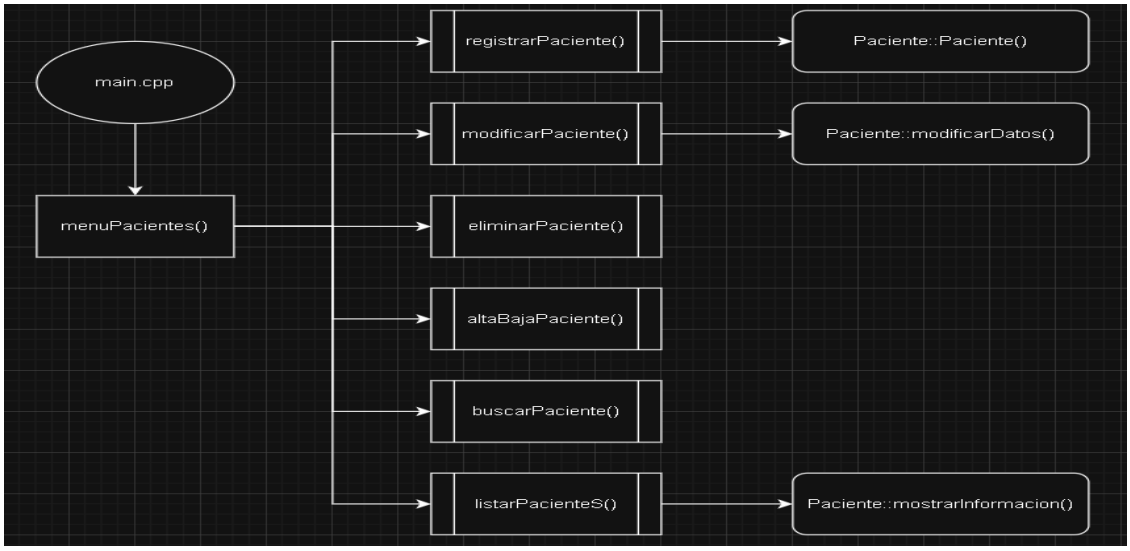
Clase Paciente:

- Atributos:
    - int IDPaciente > identificador único de los pacientes
    - string nombre
    - string dirección
    - string genero
    - string fechaNacimiento
    - string diagnostico
    - bool estado > (Activo/Inactivo)

- Miembros:
    - Paciente > constructor
    - int getIDPaciente > devuelve el ID del paciente
    - string getNombre ():
    - string getDireccion() const:
    - string getGenero () const:
    - string getFechaNacimiento() const:
    - string getDiagnostico() const:
    - string getEstado() const:

- void modificarDatos() > permite modificar los datos, en concreto, nombre, dirección y diagnóstico
- void darDeAlta() > permite cambiar el estado a Activo
- void darDeBaja() > permite cambiar el estado a Inactivo
- void mostrarInformacion() > imprime la información completa del paciente

- Diagrama de las llamadas:



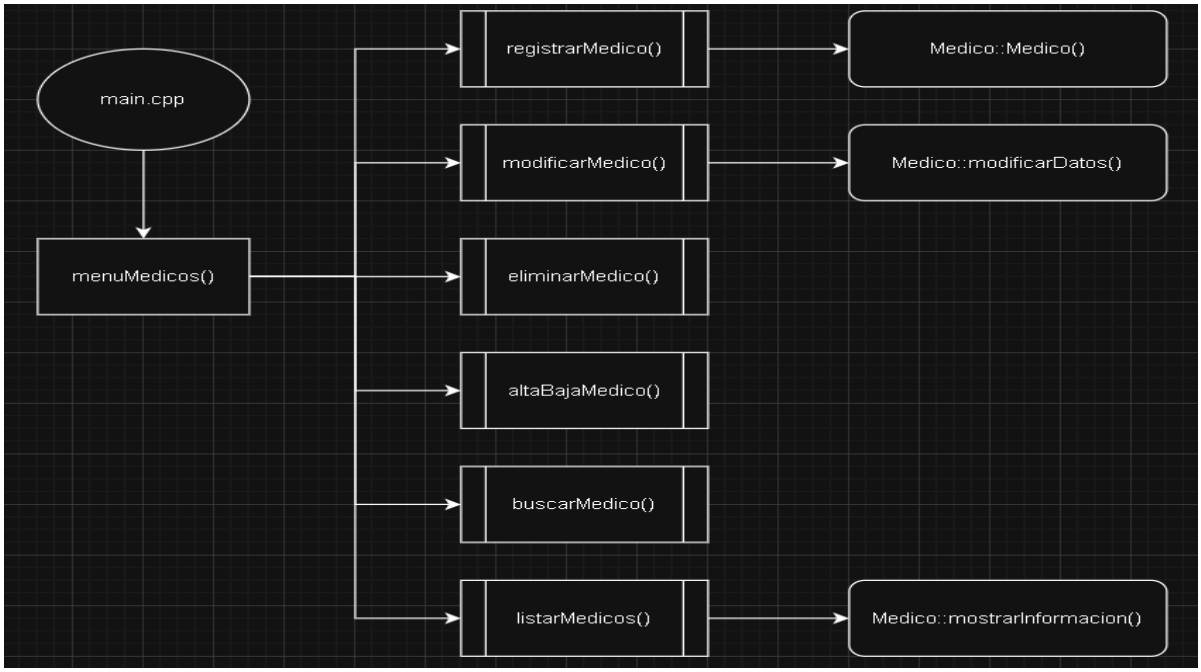
Clase Médico:

- Atributos:
  - int IDMedico > identificador único de los médicos
  - string nombre
  - string dirección
  - string genero
  - string especialidad
  - bool estado > (Activo/Inactivo)

- Miembros:
  - Medico > constructor
  - int getIDMedico > devuelve el ID del médico
  - string getNombre ():
  - string getDireccion() const:
  - string getGenero () const:
  - string getEspecialidad() const:
  - string getEstado() const:
  - void modificarDatos() > permite modificar los datos, en concreto, nombre, dirección y especialidad
  - void darDeAlta() > permite cambiar el estado a Activo

- void darDeBaja() > permite cambiar el estado a Inactivo
- void mostrarInformacion() > imprime la información completa del médico

- Diagrama de las llamadas:



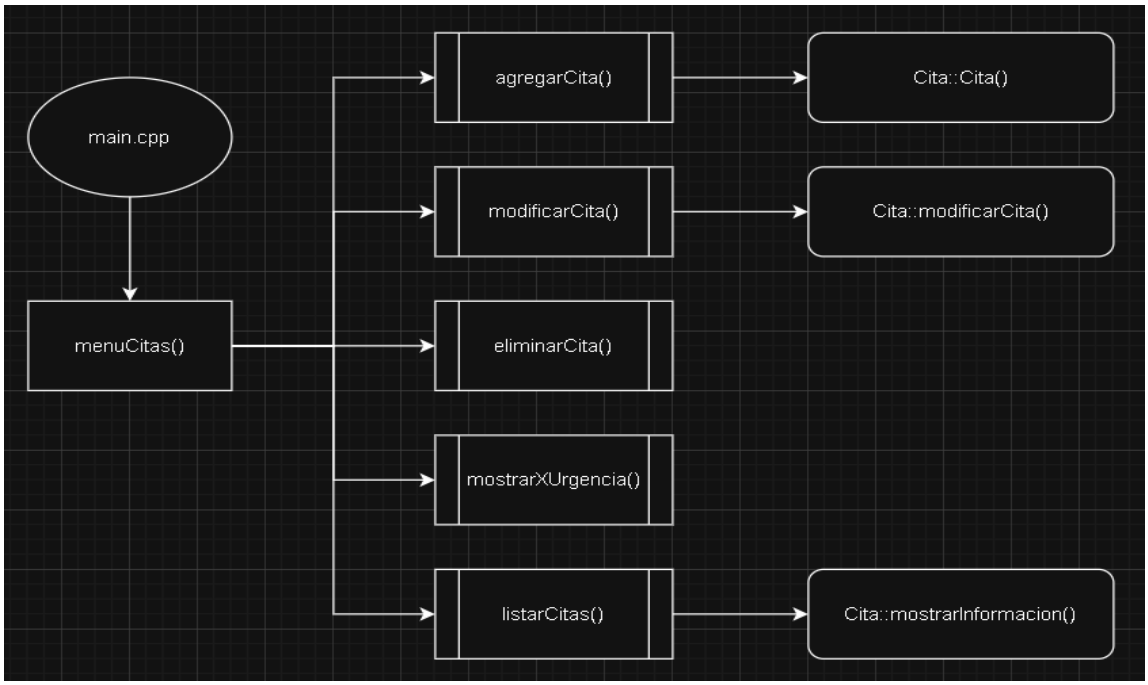
Clase Médico:

- Atributos:
  - int IDCita > identificador único de las citas
  - Paciente paciente es el objeto del paciente asociado a la cita
  - Medico medico es el objeto del médico asociado a la cita
  - string fechaCita
  - string hora
  - string motivo
  - bool urgencia

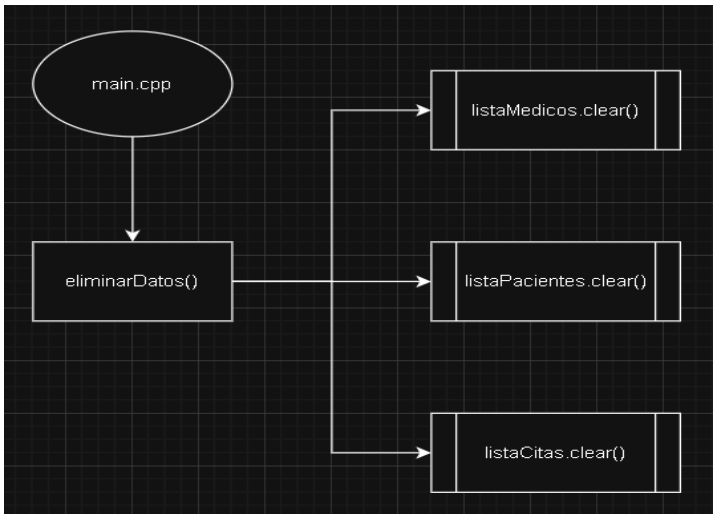
- Miembros:
  - Cita > constructor
  - int getIDCita > devuelve el ID de la cita
  - Paciente getP(): devuelve el paciente asociado a la cita elegida
  - Medico getM(): devuelve el médico asociado a la cita elegida
  - string getF() const: devuelve la fecha asociada a la cita elegida
  - string getH() const: devuelve la hora asociada a la cita elegida
  - string getMo () const: devuelve el motivo asociado a la cita elegida
  - bool esUrgente() const: devuelve si la cita es urgente o no

- void modificarCita() > permite modificar los datos, en concreto, fecha, hora, motivo y urgencia
- void mostrarInformacion() > imprime la información completa de la cita

- Diagrama de las llamadas:



Función eliminarDatos: Esta función se encarga de eliminar todos los datos de los archivos .txt.



8. Almacenamiento de datos

He utilizado archivos .txt, los cuales se generan automáticamente si no hay uno existente. El formato de los archivos es el siguiente:

Pacientes: ID | Nombre | Dirección | Género | Fecha de Nacimiento | Diagnóstico | Estado

Médicos: ID | Nombre | Dirección | Género | Especialidad | Estado

Citas: IDCita | IDPaciente | IDMédico | Día | Hora | Motivo | Urgencia

9. Decisión del diseño

El proyecto está organizado de forma modular como he dicho anteriormente para que sea fácil de navegar, además de que posteriormente puede ser extendido fácilmente. Los archivos ".txt" permiten tener un registro de los datos sin requerir una base de datos.

Existen validaciones en algunas entradas al registrar o modificar clases. Tanto el género, como las fechas, como dejar una entrada vacía no está permitido por lo tanto salta un mensaje si no lo completas correctamente.

Existen los estados para saber que o quién está activo o inactivo sin necesidad de eliminar sus datos, pero si que existe la opción para realizar un borrado completo.