

## FRONT PAGE

Nombre Alumno / DNI	Leonardo Antonio Vespa Zagorscak / Y5867066X
Título del Programa	PHE APPLIED COMPUTING & ARTIFICIAL INTELLIGENCE
N.º Unidad y Título	UNIT 20. APPLIED PROGRAMMING & DESIGN PRINCIPLES
Año académico	2024-2025
Profesor de la unidad	ÁNGEL BRAVO
Título del Assignment	APPLIED PROGRAMMING & DESIGN PRINCIPLES
Día de emisión	26/11/2024
Día de entrega	21/01/2025
Declaración del estudiante	<p>Certifico que la presentación del assignment es completamente mi propio trabajo y entiendo completamente las consecuencias del plagio.</p> <p>Entiendo que hacer una declaración falsa es una forma de mala práctica.</p> <p><b>Fecha: 26/11/24</b></p> <p><b>Firma del alumno:</b></p> <p><i>Leonardo Vespa</i></p>

**Plagio**

*El plagio es una forma particular de hacer trampa. El plagio debe evitarse a toda costa y los alumnos que infrinjan las reglas, aunque sea inocentemente, pueden ser sancionados. Es su responsabilidad asegurarse de comprender las prácticas de referencia correctas. Como alumno de nivel universitario, se espera que utilice las referencias adecuadas en todo momento y mantenga notas cuidadosamente detalladas de todas sus fuentes de materiales para el material que ha utilizado en su trabajo, incluido cualquier material descargado de Internet. Consulte al profesor de la unidad correspondiente o al tutor del curso si necesita más consejos.*



## Contenido

Módulos (Entrega01).....	3
Módulo Paciente .....	3
Módulo Médico .....	4
Módulo Citas Médicas .....	5
Módulo Generación de Reportes .....	6
Módulo manejo de Bases de Datos.....	6
Códigos (Entrega02) .....	8



Con el objetivo de organizar la estructura del código, se divide en módulos cada funcionalidad clave descrita en el proyecto del AB.

Estas mismas funcionalidades, serán descritas en un diagrama de flujo para entender las principales funciones y decisiones de cada módulo.

## Módulos (Entrega01)

### Módulo Paciente

Se asigna una clase a Pacientes (class Paciente) y permite agregar a un paciente nuevo, así como gestionar su información.

#### Atributos:

- Nombre: Nombre y apellidos del paciente.
- ID: Identificador único (Por ejemplo, el DNI, NIE o Pasaporte)
- FechaIngreso: Fecha en la que el paciente fue dado de alta en el sistema.
- historialClinico: Vector (lista) que almacena entradas de textos registrando el historial clínico del paciente.

#### Métodos

- altaPaciente: método usado para crear nuevo paciente:
  - Parámetros: nombre, ID, y fechaIngreso del paciente.
  - Función: Almacenar datos en los atributos del paciente para crear el registro.
- bajaPaciente: Utilizado para eliminar paciente
  - Función: Eliminar los datos del paciente, así como liberar la memoria ocupada.
- modificarDatos: Permite modificar la información del paciente:
  - Parámetros: recibe nuevoNombre, nuevaFechaIngreso y nuevoID.
  - Función: Actualiza los atributos con los nuevos datos.
- buscarPaciente: Permite buscar un paciente en la BBDD:
  - Parámetros: mediante un criterio (nombre, ID o fecha de ingreso) realiza la búsqueda de un paciente.



- Función: Busca si el criterio coincide con alguno de los pacientes almacenados en la BBDD.
- registrarHistorial: Registra una entrada en el historial clínico del paciente.
  - Parámetro: recibe una entrada, siendo un nuevo registro para el historial.
  - Función: Agrega la entrada al vector historialClinico.

## Módulo Médico

Se asigna una clase a Médicos (class Medico) para registrar un nuevo médico, y gestionar toda su información.

### Atributos:

- Nombre: Nombre y apellidos del médico.
- ID: Identificador único para cada médico (Por ejemplo, DNI, NIE o Pasaporte).
- Especialidad: Área de trabajo del médico.
- Disponibilidad: Estado que indica si el médico tiene disponibilidad para nuevas citas.

### Métodos:

- altaMedico: Crea un nuevo registro de médico.
  - Parámetros: recibe nombre, ID y especialidad del médico.
  - Función: Asigna los datos a cada atributo y marca la disponibilidad como "true".
- bajaMedico: Elimina el registro del médico
  - Función: Elimina los datos del médico, así como liberar el espacio ocupado en la BBDD.
- asignarEspecialidad: Asigna la especialidad del médico.
  - Parámetro: especialidadNueva es el nombre de la nueva especialidad.
  - Función: Asigna el valor al atributo especialidad.
- listarMedicosPorCriterio: Genera una lista de médico según el criterio utilizado
  - Parámetro: criterio puede ser especialidad o disponibilidad.



## Módulo Citas Médicas

Se asigna una clase (class Citas) y gestiona las citas asignadas a los pacientes y a los médicos, permitiendo ordenarlas y modificarlas.

### Atributos:

- CitaID: Identificador único de la cita.
- pacienteID: Identificador del paciente que necesita la cita.
- medicoID: identificación del médico asignado.
- Fecha: fecha y hora de la cita.
- Urgencia: prioridad de la cita que puede ser “alta” o “baja”.
- Estado: estado de la cita que puede ser “activa”, “completada” o “cancelada”.

### Métodos:

- asignarCita: asigna una nueva cita a un paciente y médico.
  - Parámetros: recibe citaID, pacienteID, medicoID, fecha y urgencia.
  - Función: Asigna los valores a los atributos correspondientes y marca la cita como “activa”.
- cancelarCita: Cambia el estado del atributo estado a “cancelada”.
  - Función: cambia el estado del atributo estado a “cancelada”
- ordenarCita: Ordena una lista de citas por fecha o urgencia.
  - Parámetros: reciba una lista citas y un criterio.
  - Función: implementa un algoritmo de ordenación para organizar la lista de citas según el criterio.
- modificarCita: Modifica la fecha o urgencia de una cita existente.
  - Parámetros: nuevaFecha y nuevaUrgencia.
  - Función: Actualiza la fecha y la urgencia de la cita con los valores nuevos.



## Módulo Generación de Reportes

Se asigna una clase (class Reporte) y genera los reportes solicitados como de los pacientes atendidos, citas pendientes y pacientes con enfermedades crónicas.

### Métodos:

- **generarReportePacientesAtendidos:** Genera el reporte de los pacientes atendidos en un rango de fechas.
  - **Parámetros:** Recibe una lista de pacientes, fechaInicio y fechaFin.
  - **Función:** Filtra los pacientes atendidos dentro del rango de fecha solicitado en el reporte.
- **reporteCitasPendientes:** Genera un listado de citas pendientes incluyendo a los médicos o la especialidad.
  - **Parámetro:** recibe una lista de citas y el medicoID.
  - **Función:** filtra las citas pendientes que estén en estado “activa” y las muestra según el medicoID.
- **reportePacientesCronicos:** Genera el reporte de los pacientes con enfermedades crónicas.
  - **Parámetros:** recibe una lista de pacientes.
  - **Función:** Filtra los pacientes con historial clínico que indica enfermedades crónicas.

## Módulo manejo de Bases de Datos

Se asigna una clase (class Archivo) para la gestión de la base de datos, guardando y recuperando todos los datos de los pacientes, médicos y citas. También, se encarga de hacer respaldos de la base de datos.

### Métodos

- **guardarDatos:** Guarda todos los datos en archivos de texto o binarios.
  - **Parámetros:** recibe listas de pacientes, médicos y citas.
  - **Función:** abre el archivo en modo escritura y almacena los datos de cada objeto- Utiliza un método datos() para extraer la información de cada objeto como una cadena de texto.



- cargarDatos: Carga los datos desde archivos al iniciar el programa.
  - Parámetros: recibe referencias a las listas de pacientes, médicos y citas.
  - Función: abre el archivo en modo lectura y recupera cada dato, reconstruyendo los objetos y cargándolos en las listas correspondientes.
- Backup: Crea una copia de seguridad de los datos.
  - Función: Copia el archivo principal en un archivo de respaldo con un nombre diferente, permitiendo restaurar los datos en caso de fallo.

Los pseudocódigos de cada módulo serán entregados en archivos .txt en la misma entrega del classroom, así como el diagrama de flujo, entregado en formato .jpg.

También, se subirá toda la documentación a este repositorio de GitHub : [Unit20 MSMK](#)

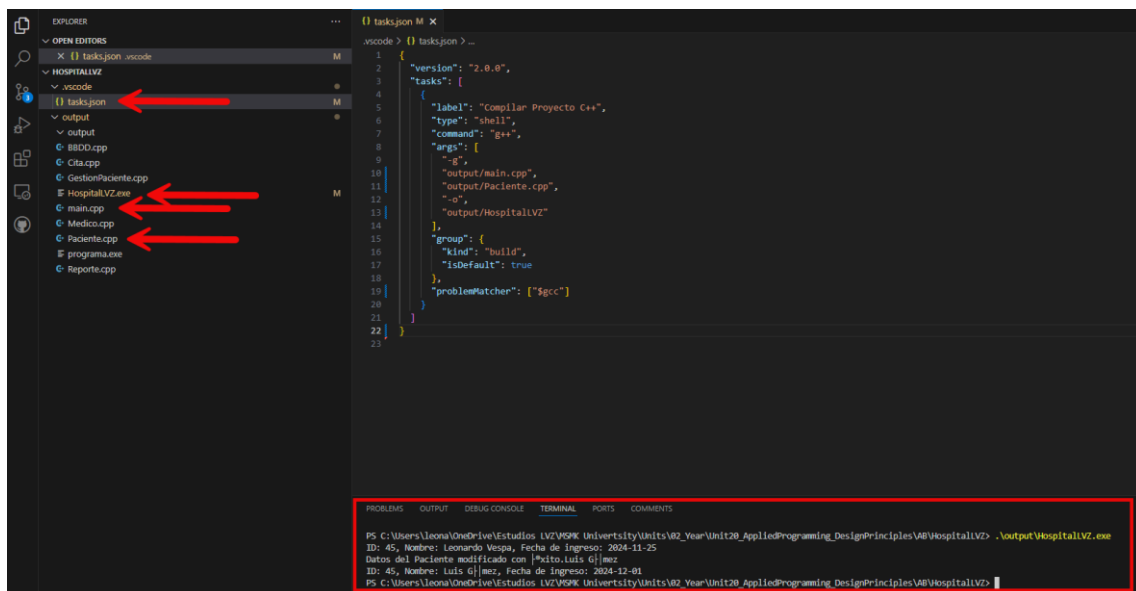
## Códigos (Entrega02)

Para la presente entrega, se han desarrollado una primera versión de los códigos para cada una de las clases, los cuales se harán entrega por [GitHub](#).

La clase donde se ha realizado el mayor desarrollo y pruebas ha sido la de Paciente.cpp con el objetivo de ver que funcionara de la manera más correcta posible.

Para ello, se han realizado dos pruebas:

1 – Clase Paciente.cpp y main.cpp por separado: Aquí, se buscaba ver el comportamiento al compilar y ejecutar el programa HospitalLVZ.exe, y el resultado fue:



The image shows a screenshot of the Visual Studio Code editor. On the left, the Explorer sidebar displays the project structure with files like task.json, output, BDDO.cpp, Cita.cpp, GestionPaciente.cpp, HospitalLVZ.exe, main.cpp, Medico.cpp, Paciente.cpp, programa.exe, and Reporte.cpp. Red arrows point to task.json, HospitalLVZ.exe, main.cpp, and Paciente.cpp. The main editor area shows the content of task.json, which defines a task for compiling and running the program. The terminal at the bottom shows the execution of HospitalLVZ.exe, displaying patient data and a confirmation message.

```
1 {
2   "version": "2.0.0",
3   "tasks": [
4     {
5       "label": "Compilar Proyecto C++",
6       "type": "shell",
7       "command": "g++",
8       "args": [
9         "-g",
10        "output/main.cpp",
11        "output/Paciente.cpp",
12        "-o",
13        "output/HospitalLVZ"
14      ],
15      "group": {
16        "kind": "build",
17        "isDefault": true
18      },
19      "problemMatcher": ["$gcc"]
20    }
21  ]
22 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
PS C:\Users\leona\OneDrive\Estudios\LIVMSMK University\Units\B2_Year\Unit20_AppliedProgramming_Design\Principles\AB\HospitalLVZ> .\output\HospitalLVZ.exe
ID: 45, Nombre: Leonardo Vespa, Fecha de ingreso: 2024-11-25
Datos del Paciente modificado con |*ito.Luis G|mez
ID: 45, Nombre: Luis G|mez, Fecha de ingreso: 2024-12-01
PS C:\Users\leona\OneDrive\Estudios\LIVMSMK University\Units\B2_Year\Unit20_AppliedProgramming_Design\Principles\AB\HospitalLVZ>
```

Imagen 1 - Resultado de ejecutar HospitalLVZ.exe. Fuente: Autor

Se mostro con éxito cuando se dio de alta un paciente y cuando fueron modificados sus datos.



2 – Clase Paciente.cpp y main.cpp juntas: Para poder ejecutar este archivo, se creó otro archivo llamado GestionPaciente.cpp donde se insertó el archivo main.cpp para poder crear un menú que permitiera seleccionar que se quería realizar dentro de la función Paciente.cpp Para ello se ejecuta el programa llamado GestionPaciente.exe:

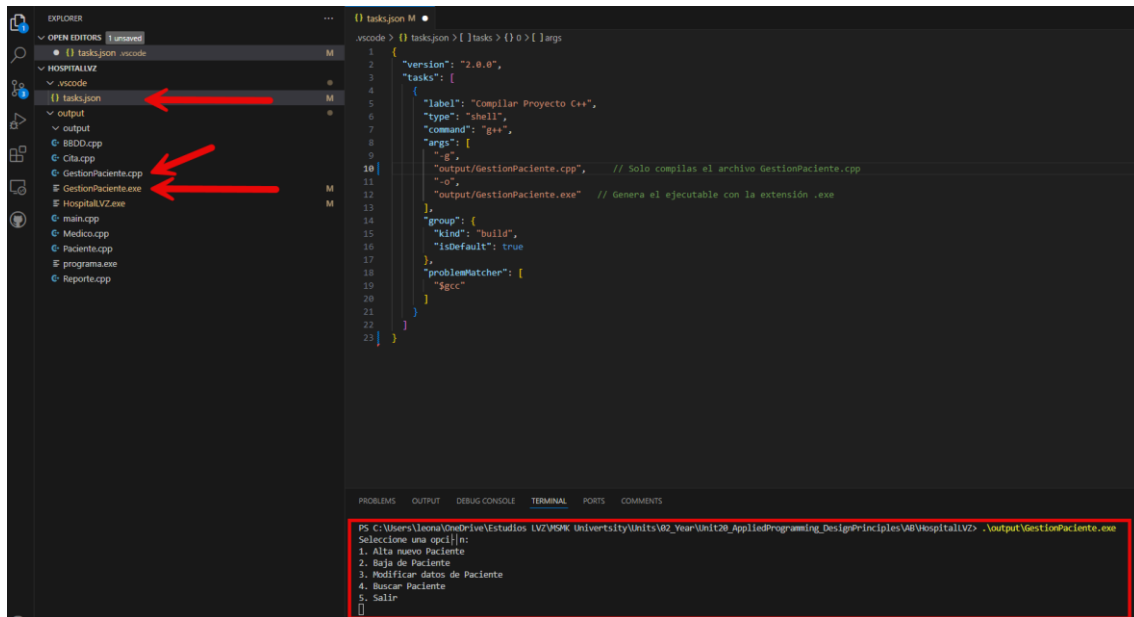


Imagen 2 - Resultado de ejecutar GestionPaciente.exe. Fuente: Autor

Se mostro con éxito el menú con las distintas opciones, así como la navegación por el.