


PORTADA

Nombre Alumno / DNI	Francisco Javier Roig Minguell / 05439210D
Título del Programa	2º PD CYBER SECURITY / COMPUTER SCIENCE / DATA SCIENCE
Nº Unidad y Título	UNIT 20 - APPLIED PROGRAMMING & DESIGN PRINCIPLES
Año académico	2024/2025
Profesor de la unidad	ANGEL BRAVO
Título del Assignment	
Día de emisión	27/11/2024
Día de entrega	21/01/2025
Nombre IV y fecha	
Declaración del estudiante	<p>Certifico que la presentación del assignment es completamente mi propio trabajo y entiendo completamente las consecuencias del plagio. Entiendo que hacer una declaración falsa es una forma de mala práctica.</p> <p>Fecha: 20/01/2025</p> <p>Firma del alumno: Javier Roig Minguell</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.

Documento Explicativo de Desarrollo del Proyecto

Introducción

Este proyecto ha sido desarrollado con el propósito de gestionar de manera eficiente la información en un entorno hospitalario. Se han implementado funcionalidades clave para la gestión de pacientes, médicos y citas, además de un sistema para almacenar y recuperar datos utilizando archivos. Este documento describe el diseño, la implementación y las decisiones técnicas tomadas durante el desarrollo.

Estructura del Proyecto

El proyecto está compuesto por múltiples módulos para organizar las funcionalidades:

1. **Clase Paciente:** Permite manejar la información personal de los pacientes, incluyendo su historial clínico.
 2. **Clase Médico:** Administra los datos de los médicos y su disponibilidad.
 3. **Clase Cita:** Gestiona las citas médicas, incluyendo información sobre el paciente, el médico, la fecha y si la cita es urgente.
 4. **Clase Historial:** Registra eventos específicos en el historial clínico de los pacientes.
 5. **Clase Servicio:** Contiene información sobre los servicios médicos disponibles y sus costos.
 6. **Módulo de manejo de archivos (archivos.cpp):** Se encarga de almacenar y recuperar información de pacientes, asegurando persistencia de datos.
 7. **Módulo de gestión de citas (gestion_citas.cpp):** Proporciona funciones para ordenar y organizar citas.
 8. **Módulo de gestión de pacientes (gestion_pacientes.cpp):** Incluye operaciones para agregar, buscar y eliminar pacientes.
-

Detalles Técnicos

Clase Paciente

Se diseñó con atributos como el nombre, ID único y fecha de ingreso. Incluye métodos para:

- Agregar eventos al historial clínico.
- Mostrar historial.
- Obtener datos básicos del paciente (nombre, ID, fecha de ingreso).

Clase Médico

Incluye atributos como nombre, especialidad y disponibilidad. Sus métodos permiten:

- Asignar especialidad.
- Cambiar disponibilidad.
- Recuperar información básica del médico.

Clase Cita

Gestiona las citas médicas con atributos como paciente, médico, fecha, hora y urgencia.

Métodos clave:

- Mostrar detalles de la cita.
- Verificar si es urgente.

Manejo de Archivos

El módulo de archivos utiliza las clases <fstream> para persistencia de datos. Ejemplo:

- **Guardar pacientes:** Escribe información de cada paciente en un archivo.
- **Cargar pacientes:** Lee un archivo y reconstruye la lista de pacientes en memoria.

Gestión de Citas

El módulo incluye funciones para ordenar citas por fecha, asegurando una experiencia optimizada al mostrar citas pendientes.

Gestión de Pacientes

Proporciona funcionalidades para:

- Agregar nuevos pacientes a la base de datos.
- Buscar pacientes por su ID.
- Eliminar pacientes de la base de datos.

Decisiones de Diseño

Modularidad

El proyecto se dividió en módulos independientes para facilitar la escalabilidad y el mantenimiento.

Orientación a Objetos

Se utilizó programación orientada a objetos (POO) para modelar entidades del mundo real como pacientes, médicos y citas. Esto mejora la legibilidad y reutilización del código.

Persistencia

El manejo de archivos asegura que la información importante no se pierda al cerrar la aplicación.
