Avance 3 – Documento de Diseño de Interfaz y Validaciones

Ian Ramos, Cristhofer Villarreal, Jean Llantén e Ivan Justavino

Se debe entregar un **documento estructuarado** que incluya:

1. Diseño de Interfaz de Usuario (UI/UX)

1. Diseño de Interfaz (UI/UX)

Sistema de Gestión del Hospital Universitario UTP

1.1 Concepto General de la Interfaz

El Sistema de Gestión del Hospital Universitario UTP utiliza una **interfaz de usuario basada en consola**, diseñada para ser clara, intuitiva y fácil de usar. El objetivo principal del diseño es permitir que los usuarios interactúen con el sistema de forma ordenada, minimizando errores y facilitando el acceso a las funcionalidades principales del sistema hospitalario.

La interfaz está pensada para **usuarios administrativos y personal del hospital**, quienes no necesariamente poseen conocimientos técnicos avanzados.

1.2 Principios de Diseño Aplicados

El diseño de la interfaz sigue los siguientes principios de UX:

- **Simplicidad:** Menús claros con opciones numeradas.
- **Consistencia:** Todos los menús siguen la misma estructura.
- **Claridad:** Mensajes comprensibles para el usuario.
- **Prevención de errores:** Validaciones antes de procesar la información.
- **Retroalimentación inmediata:** El sistema informa cada acción realizada.

1.3 Pantalla de Inicio

Al iniciar el sistema, el usuario visualiza un mensaje de bienvenida que identifica claramente la institución:

==========
SISTEMA DE GESTIÓN HOSPITAL UNIVERSITARIO
Universidad Tecnológica de Panamá
==========

Esta pantalla refuerza la identidad institucional del sistema y orienta al usuario desde el inicio.

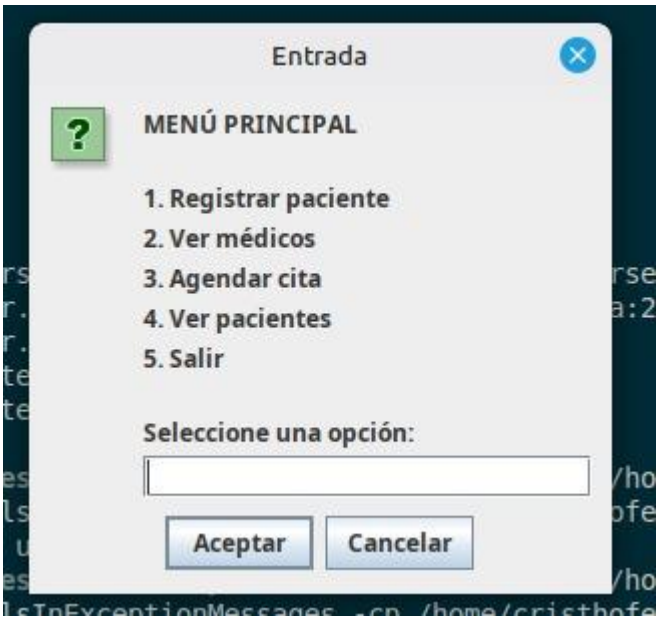
1.4 Menú Principal

El menú principal permite acceder a las funcionalidades principales del sistema:

1. Registrar Paciente
2. Registrar Médico
3. Agendar Cita
4. Gestionar Recetas
5. Ver Reportes
6. Salir
Seleccione una opción:

Características del menú:

- Uso de números para facilitar la selección.
- Validación de opciones incorrectas.
- Mensajes de error claros en caso de selección inválida.



1.5 Flujo de Interacción del Usuario

El flujo de navegación del sistema es el siguiente:

1. El usuario visualiza el menú principal.
2. Selecciona una opción.
3. El sistema solicita los datos necesarios.
4. Se validan los datos ingresados.
5. El sistema confirma la acción realizada.
6. Retorna automáticamente al menú principal.

Este flujo garantiza una experiencia ordenada y sin confusión.

1.6 Mensajes del Sistema

El sistema muestra mensajes informativos en todo momento:

- Confirmación de registros exitosos
- Mensajes de error claros y descriptivos
- Indicaciones sobre el formato correcto de los datos

Ejemplo:

Paciente registrado correctamente.
Presione ENTER para continuar...

1.7 Accesibilidad y Usabilidad

El diseño de la interfaz considera aspectos básicos de accesibilidad:

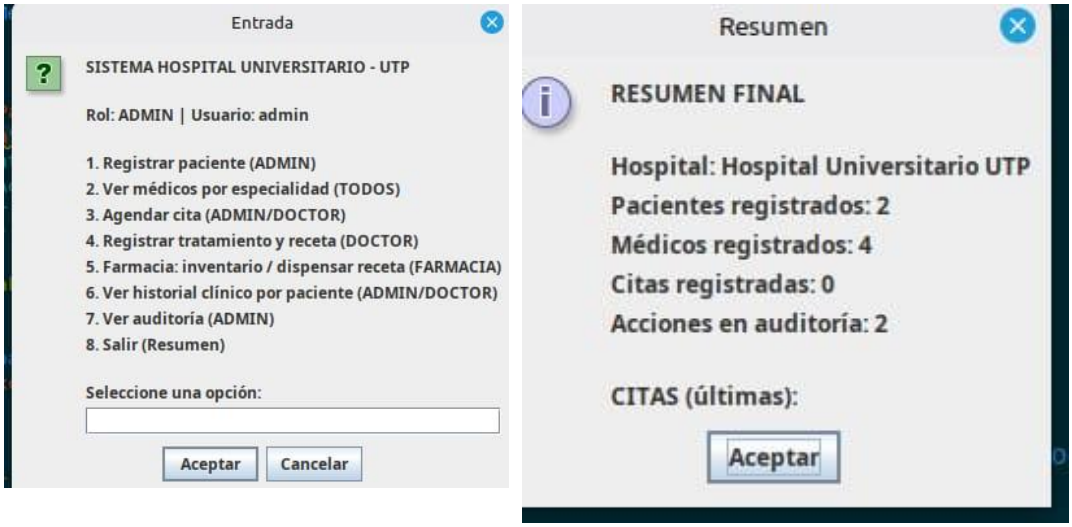
- Texto legible
- Lenguaje sencillo
- Navegación guiada
- Evita saturación de información

Esto permite que cualquier usuario pueda utilizar el sistema con una mínima curva de aprendizaje.

1.8 Justificación del Diseño

Aunque el sistema no utiliza una interfaz gráfica (GUI), la interfaz por consola cumple con los requisitos del proyecto académico, permitiendo demostrar:

- Lógica de negocio
- Organización del sistema
- Correcta interacción usuario–sistema
- Aplicación de validaciones y control de flujo.
-



2. Validaciones por cada entrada de usuario.

Sistema de Gestión del Hospital Universitario UTP

En el sistema se implementan validaciones para garantizar la correcta entrada de datos por parte del usuario, evitando errores de ejecución y asegurando la integridad de la información registrada. A continuación, se detallan las validaciones aplicadas en cada funcionalidad del sistema.

2.1 Validaciones del Menú Principal

Tabla de Validaciones

Nº	Validación	Descripción	Dónde se implementa

1	Opción de menú numérica	Verifica que la opción ingresada sea un número entero válido.	Menú principal (`try / catch NumberFormatException`)
2	Opción dentro del rango	Evita ejecutar opciones inexistentes del menú.	`switch(op)`
3	Nombre de paciente válido	Nombre no vacío, mínimo 3 caracteres y solo letras y espacios.	Registro de paciente – `case 1`
4	Paciente no duplicado	No permite registrar pacientes con el mismo nombre.	Registro de paciente – `case 1`
5	Fecha de nacimiento válida	Fecha en formato `YYYY-MM-DD`.	Registro de paciente – `case 1`
6	Fecha no futura	Impide fechas de nacimiento posteriores a la fecha actual.	Registro de paciente – `case 1`
7	Rol autorizado	Restringe acciones según rol (ADMIN, DOCTOR, FARMACIA).	En cada opción del menú
8	Existencia de pacientes	No permite agendar citas sin pacientes registrados.	Agendar cita – `case 3`
9	Fecha/hora válida de cita	Formato correcto `dd/MM/yyyy HH:mm`.	Agendar cita – `case 3`
10	Cita no en el pasado	Evita agendar citas con fechas anteriores a la actual.	Agendar cita – `case 3`
11	Evitar cita duplicada	Un paciente no puede tener dos citas en la misma fecha y hora.	Agendar cita – `case 3`
12	Evitar solapamiento	Un médico no puede tener dos citas en la misma fecha y hora.	Agendar cita – `case 3`
13	Tratamiento válido	El tratamiento debe tener mínimo 5 caracteres.	Tratamiento y receta – `case 4`
14	Receta con medicamentos	La receta debe contener al menos un medicamento.	Tratamiento8 y receta – `case 4`
15	Medicamento válido	El nombre del medicamento no puede ser vacío.	Farmacia – `case 5`
16	Cantidad válida	La cantidad ingresada debe ser un número entero.	Farmacia – `case 5`
17	Stock no negativo	Evita que el inventario quede en valores negativos.	Farmacia – `case 5`
18	Receta no dispensada	Impide dispensar una receta más de una vez.	Farmacia – `case 5`
19	Stock suficiente	Valida existencia de stock antes de dispensar receta.	Farmacia – `case 5`
20	Cancelación segura	Si el usuario cancela una ventana, el sistema regresa al menú.	En todos los `JOptionPane`

Tabla de Excepciones Manejadas

Nº	Excepción	Cuándo ocurre	Manejo realizado
1	`NumberFormatException`	Cuando el usuario ingresa texto en campos numéricos (menú, stock).	Se muestra mensaje de error y se retorna al menú.
2	`DateTimeParseException`	Cuando el formato de fecha o fecha/hora es incorrecto.	Se notifica al usuario y se solicita un nuevo valor.

Manejo General de Errores

El sistema implementa manejo de excepciones para evitar fallos inesperados:

- NumberFormatException
- InputMismatchException
- DateTimeParseException
- Validaciones lógicas personalizadas

Cuando ocurre un error, el sistema muestra un mensaje claro al usuario y retorna al menú principal sin finalizar la ejecución del programa.

3. Actualización basada en retroalimentació

- Comentarios Recibidos y Mejoras Implementadas

Avances #1 y #2 – Sistema de Gestión de Hospital Universitario

Avance #1

Comentarios Recibidos y Correcciones Aplicadas

1. Corrección en la Clase Médico

Comentario del profesor:
El método agendarCita(c: Cita): boolean en la clase Medico asignaba demasiada responsabilidad a esta clase. La lógica de agendamiento de citas es compleja, ya que implica validar disponibilidad, evitar duplicidad y posibles solapamientos de horarios, lo cual no corresponde a una entidad de dominio.

Corrección implementada:
Se eliminó la lógica de agendamiento de citas de la clase Medico y se trasladó a una clase

controladora central denominada GestorCitas.
En la clase Medico, el método fue reemplazado por obtenerCitasEnFecha(fecha), cuya responsabilidad se limita únicamente a devolver las citas asociadas a un médico en una fecha determinada.

Con esta corrección se aplicó el principio de **responsabilidad única (SRP)**, mejorando la organización y mantenibilidad del código.

2. Corrección en la Clase Historial Clínico

Comentario del profesor:
La clase HistorialClinico hacía uso de una lista de objetos EntradaHistorial, sin que dicha clase estuviera definida en el diagrama UML ni en la lista de clases.

Corrección implementada:
Se definió formalmente la clase EntradaHistorial, incorporando los atributos clave:

- fecha: DateTime
- tipo: String (Cita, Tratamiento, Resultado)
- referenciaID: String

Esta corrección permitió completar el modelo conceptual y garantizar coherencia entre el UML y la implementación en código.

3. Corrección en la Clase Auditoría

Comentario del profesor:
El método registrarAccion() utilizaba un parámetro de tipo String para representar la fecha, lo cual no era adecuado para un registro de auditoría.

Corrección implementada:
Se ajustó el método para que el manejo de la fecha se realice mediante un atributo timestamp de tipo DateTime, ya existente en la clase.
De esta manera, se eliminó la necesidad de pasar la fecha como parámetro, mejorando la precisión y claridad del diseño.

4. Inclusión de Clases Controladoras Faltantes

Comentario del profesor:
Se indicó la ausencia de clases controladoras necesarias para centralizar la lógica del sistema.

Corrección implementada:
Se agregaron las siguientes clases:

- **GestorCitas:**
  Encargada de orquestar la creación de citas médicas, validando disponibilidad del médico y evitando duplicidad por paciente.
- **GestorPacientes:**
  Responsable del registro de pacientes y de la creación del historial clínico asociado.
- **GeneradorReportes:**
  Se creó para centralizar la generación de reportes del sistema, eliminando esta responsabilidad de la clase Administrador.

Estas incorporaciones mejoraron la arquitectura del sistema y separaron correctamente la lógica de negocio.

Avance #2

Comentarios Recibidos y Correcciones Aplicadas

1. Método Incorrecto en la Clase Médico

Comentario del profesor:
La clase Medico aún contenía el método emitirReceta() vacío, cuando debía enfocarse únicamente en métodos relacionados con sus citas.

Corrección implementada:
Se eliminó el método vacío emitirReceta() y se implementó correctamente el método obtenerCitasEnFecha(), alineando la clase Medico con su responsabilidad real dentro del sistema.

2. Implementación Incompleta del Método obtenerCitasEnFecha()

Comentario del profesor:
Aunque la lógica de agendamiento estaba en GestorCitas, el método obtenerCitasEnFecha() no estaba implementado en la clase Medico.

Corrección implementada:
Se implementó el método obtenerCitasEnFecha() en la clase Medico, permitiendo que el gestor de citas consulte directamente las citas del médico para validar disponibilidad.

3. Corrección Confirmada en la Clase Auditoría

Comentario del profesor:
Se indicó que el método registrarAccion() ya no debía recibir la fecha como parámetro.

Corrección implementada:
Esta corrección fue aplicada correctamente, utilizando el atributo timestamp interno de la clase Auditoria.

4. Implementación de Relaciones del Modelo

Comentario del profesor:
No se evidenciaba la implementación completa de todas las relaciones descritas en las tablas y el UML.

Corrección implementada:
Se revisaron y aplicaron las relaciones entre clases en el código, asegurando coherencia entre:

- Paciente y Historial Clínico
- Médico y Citas
- Gestores y entidades del dominio

Esto garantizó que el modelo UML reflejara fielmente la implementación real del sistema.

Las correcciones realizadas a partir de los comentarios recibidos en los Avances #1 y #2 permitieron mejorar significativamente la estructura del sistema, aplicando principios de diseño orientado a objetos, una correcta separación de responsabilidades y una mayor coherencia entre el diseño UML y el código implementado. Estas mejoras fortalecen la calidad, escalabilidad y mantenibilidad del sistema desarrollado.

4. Entrega técnica

- Subir el documento en formato **PDF** al repositorio de GitHub (en carpeta /docs).
- Adjuntar el mismo PDF en la **tarea de Teams**.
- Comprimir todos los archivos **.java** en un ZIP y subirlo a la **tarea de Teams**.

Nota: Este avance **se debe entregar un día antes de su presentación**, forma parte del **25% de la nota semestral** (promedio con Avances 1 y 2).

Rúbrica:

Criterio	Peso (%)	Excelente (100-90%)	Bueno (89-70%)	Regular (69-50%)	Insuficiente (49-0%)
Documento de diseño de interfaz	25%	Prototipos completos, flujos claros, justificación técnica sólida.	Prototipos adecuados, flujos definidos, pero con omisiones menores.	Prototipos básicos, flujos confusos o incompletos.	Diseño ausente, desorganizado o irrelevante.
Validaciones descritas	25%	Todas las entradas validadas, con ejemplos de errores, mensajes claros y manejo de excepciones.	La mayoría de las entradas validadas, pero faltan detalles o ejemplos.	Validaciones limitadas, sin cobertura completa.	Validaciones ausentes o mal implementadas.
Actualización tras retroalimentación	20%	Se listan y corrigen todos los comentarios previos con evidencia concreta en código y diseño.	Se atienden los comentarios principales, pero sin detalle exhaustivo.	Se mencionan algunos comentarios, pero sin cambios claros.	No se considera la retroalimentación previa.
Entrega técnica	15%	Todo subido correctamente a GitHub y Teams, ZIP organizado, estructura clara.	Entrega completa con pequeños errores de formato u organización.	Faltan archivos o están mal etiquetados.	Entrega incompleta o incorrecta.
Claridad y profesionalismo	15%	Documento bien redactado, sin errores, con estilo técnico, ortografía y formato impecables.	Legible, pero con errores menores de redacción o formato.	Redacción confusa, varios errores de forma.	Documento de baja calidad, difícil de seguir.

Total: 100% de este avance (equivalente al 25% de la nota semestral).