SISTEMA DE GESTIÓN - PELUQUERÍA FULLSTACK

Trabajo Práctico Integrador – Programación II
Tecnicatura Universitaria en Programación – UTN FRRE
Año lectivo 2025 – Comisión 1 (Turno Mañana)
Profesor: Facundo Uferer
Integrantes: Kevin Bizjan – Thomas Stagliano – Sebastián Flores

ÍNDICE

1. Introducción

El presente informe describe el desarrollo del *Sistema de Gestión de Peluquería Fullstack*, un proyecto implementado como Trabajo Práctico Integrador para la materia **Programación II**.

El sistema funciona por consola y permite administrar clientes, empleados, servicios y turnos de una peluquería realista.

Se utilizó Visual Studio, Java 21, programación estructurada, POO, colecciones, manejo de archivos y excepciones personalizadas, siguiendo las consignas obligatorias del trabajo.

2. Objetivo del sistema

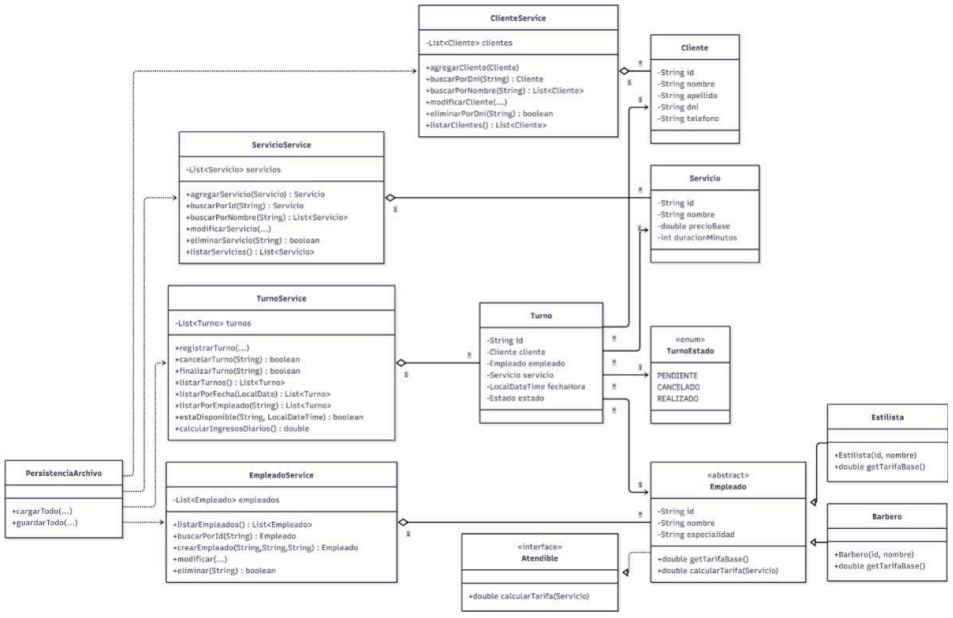
El objetivo es modelar el funcionamiento administrativo de una peluquería moderna, permitiendo:

- Registrar clientes
- Registrar y modificar empleados
- Definir servicios con precio y duración
- Agendar turnos verificando disponibilidad
- Emitir reportes del estado actual

El sistema opera íntegramente desde consola.

3. Diseño e implementación

3.1. Modelo UML



Incluye clases: Cliente, Servicio, Empleado (abstracta) → Barbero, Estilista,Turno ,Services: ClienteService, ServicioService, EmpleadoService, TurnoService ,Excepciones personalizadas, PersistenciaArchivo.

3.2. Entidades principales

- **Cliente**
  Nombre, apellido, DNI, teléfono.
- **Empleado**
  ID, nombre, especialidad (Barbero/Estilista).
- **Servicio**
  ID, nombre, precio, duración.
- **Turno**
  Relación Cliente – Empleado – Servicio + fecha/hora.

3.3. Servicios implementados

Cada entidad tiene un *Service* con las operaciones CRUD: Agregar, Buscar, Modificar, Eliminar, Listar.

Además: TurnoService implementa validación de disponibilidad y ClienteService utiliza un **array auxiliar** para cumplir el trabajo.

3.4. Manejo de excepciones

Se agregaron excepciones personalizadas como:

ClienteDuplicadoException

ElementoNoEncontradoException

ServicioInvalidoException

TurnoNoDisponibleException

EmpleadoNoEncontradoException

Todas integradas en menús con mensajes claros para el usuario.

3.5. Persistencia

El sistema guarda y carga automáticamente: clientes.json, servicios.json, empleados.json. turnos.json.

Mediante la clase **PersistenciaArchivo** utilizando Gson.

4. Funcionalidades del sistema

4.1. Gestión de Clientes

- Alta con validaciones de DNI, nombre y teléfono
- Búsqueda por DNI o por nombre parcial
- Modificación
- Eliminación
- Listado general

4.2. Gestión de Servicios

- Creación con validación de precio y duración
- Edición
- Eliminación
- Búsqueda
- Listado general

4.3. Gestión de Empleados

- Alta de Barbero o Estilista
- Validación de tipo
- Búsqueda por ID
- Listado general

4.4. Gestión de Turnos

1. Selección guiada de cliente, empleado y servicio
2. Validación de fecha futura
3. Validación de disponibilidad del empleado
4. Cancelación
5. Finalización
6. Listado por día, por empleado o global

4.5. Reportes:

Se implementaron distintos tipos de reportes para el usuario como Ingresos del día, Turnos pendientes, Turnos realizados hoy, Ingresos por empleado y servicio, horas trabajadas por empleado y ranking de servicios más vendidos.

5. Programación estructurada aplicada

Uso intensivo de `Scanner`
Estructuras `if`, `switch`, `for`, `do-while`, `forEach`
Recursividad indirecta (menú repetitivo)

Uso de `break`, `continue`
Manipulación de Strings
Métodos con parámetros y reutilización

6. Programación Orientada a Objetos aplicada

Encapsulamiento en todas las clases modelo

Herencia: `Empleado` → `Barbero` / `Estilista`

Polimorfismo: `calcularTarifa()` definido en la interfaz Atendible

Interfaces: `Atendible`

Uso de lambdas y streams: `removeIf`, `anyMatch`, `filter`, `forEach`

7. Uso de colecciones y estructuras

El sistema utiliza: **ArrayList** para todas las entidades, **Arrays** en ClienteServicio, **Streams** y métodos funcionales, Ordenamientos por criterios, Iteradores implícitos

8. Capturas de pantalla Código:

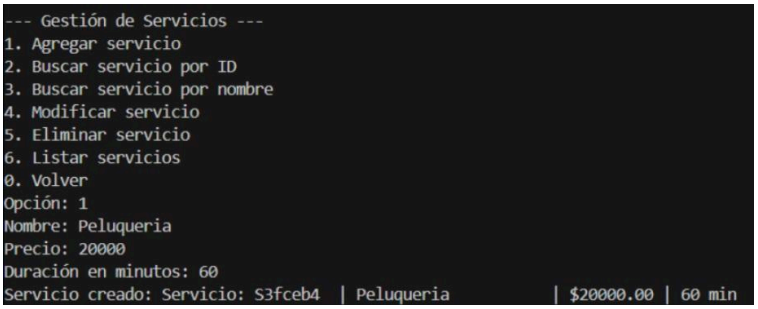


Imagen 1 – Gestión de Servicios:

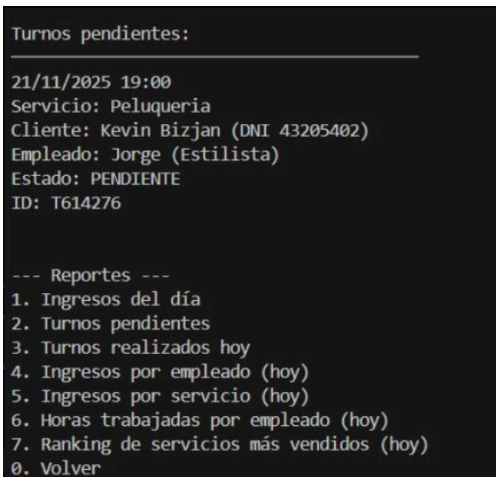


Imagen 2 – Gestión de turnos

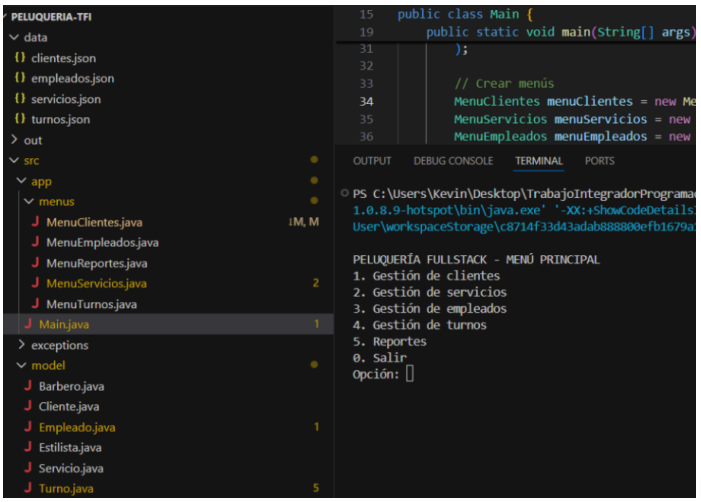


Imagen 3 – Menú principal

9. Conclusión

El sistema cumple completamente los requisitos del Trabajo Práctico Integrador, integrando programación estructurada, POO, colecciones, manejo de excepciones personalizadas, persistencia de datos y un diseño modular mantenible.

Tratamos de representar un caso realista y funcional de un sistema administrativo de peluquería, con validaciones robustas y una estructura clara que facilita futuras extensiones.

10. Participación por integrantes:

Kevin Bizjan: Implementación general del sistema. Menú principal, TurnoService completo. Validaciones y excepciones. Persistencia y estructura del proyecto. Diagrama UML y documentación.

Thomas Stagliano. Desarrollo de ClienteService y ServicioService. Validaciones de datos. Manejo de búsqueda y filtrado. Reportes y pruebas de funcionamiento.

Sebastián Flores. Desarrollo de EmpleadoService. Clases Barbero y Estilista. Interfaz Atendible. Colaboración en menú de turnos y testing.