

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
2. Objetivo del sistema
3. Diseño e implementación
 - 3.1. Modelo de Clases (UML)
 - 3.2. Entidades principales
 - 3.3. Servicios implementados
 - 3.4. Manejo de excepciones
 - 3.5. Persistencia
4. Funcionalidades del sistema
 - 4.1. Gestión de Clientes
 - 4.2. Gestión de Servicios
 - 4.3. Gestión de Empleados
 - 4.4. Gestión de Turnos
 - 4.5. Reportes
5. Programación estructurada aplicada
6. Programación Orientada a Objetos aplicada
7. Uso de colecciones y estructuras
8. Capturas de Pantalla
9. Conclusión
10. Participación por integrante

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

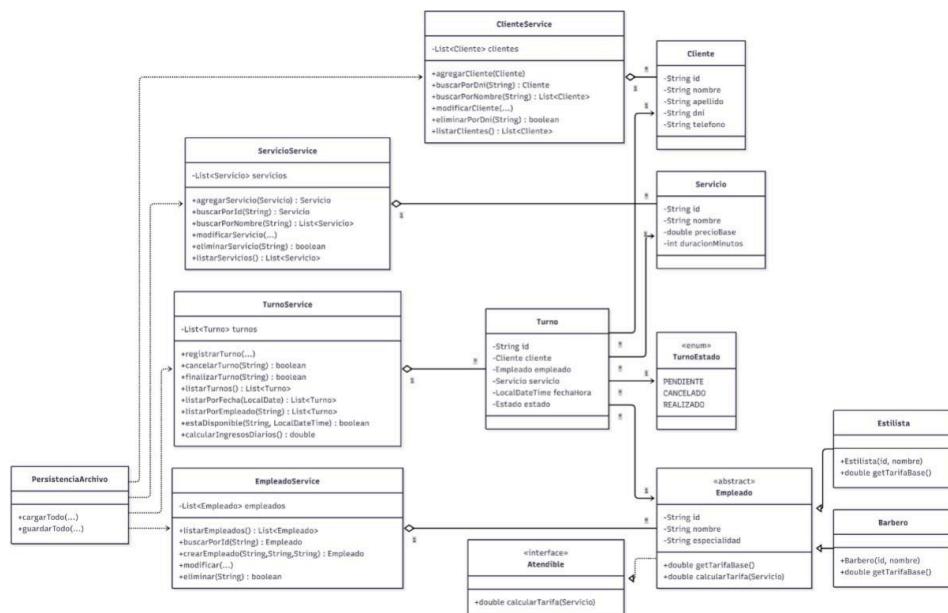
El objetivo es modelar el funcionamiento administrativo de una peluquería moderna, permitiendo:

- 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:

```
-- Gestión de Servicios --
1. Agregar servicio
2. Buscar servicio por ID
3. Buscar servicio por nombre
4. Modificar servicio
5. Eliminar servicio
6. Listar servicios
0. Volver
Opción: 1
Nombre: Peluqueria
Precio: 20000
Duración en minutos: 60
Servicio creado: Servicio: 53fceb4 | Peluqueria | $20000.00 | 60 min
```

Imagen 1 – Gestión de Servicios:

```

Turnos pendientes:

21/11/2025 19:00
Servicio: Peluquería
Cliente: Kevin Bizjan (DNI 43205402)
Empleado: Jorge (Estilista)
Estado: PENDIENTE
ID: T614276

--- Reportes ---
1. Ingresos del día
2. Turnos pendientes
3. Turnos realizados hoy
4. Ingresos por empleado (hoy)
5. Ingresos por servicio (hoy)
6. Horas trabajadas por empleado (hoy)
7. Ranking de servicios más vendidos (hoy)
0. Volver

```

Imagen 2 – Gestión de turnos

The screenshot shows the Eclipse IDE interface. On the left, the project structure for 'PELUQUERIA-TFI' is displayed, showing packages like 'data', 'src/app', 'src/menu', 'model', and 'exceptions', along with various Java files such as 'Main.java', 'MenuClientes.java', 'MenuEmpleados.java', etc. On the right, a terminal window is open, showing the command 'PS C:\Users\Kevin\Desktop\Trabajo Integrador Programación 1.0.8.9-hotspot\bin\java.exe' being run, followed by the output of a Java application. The application's output shows a menu for 'PELUQUERÍA FULLSTACK - MENÚ PRINCIPAL' with options 1 through 7 and a 'Salir' option.

```

PELUQUERIA-TFI
└── data
    ├── clientes.json
    ├── empleados.json
    ├── servicios.json
    └── turnos.json
  ↘ out
  └── src
      └── app
          └── menus
              ├── Main.java
              ├── MenuClientes.java
              ├── MenuEmpleados.java
              ├── MenuReportes.java
              ├── MenuServicios.java
              └── MenuTurnos.java
      └── exceptions
      └── model
          ├── Barbero.java
          ├── Cliente.java
          ├── Empleado.java
          ├── Estilista.java
          ├── Servicio.java
          └── Turno.java

```

```

15  public class Main {
16      public static void main(String[] args) {
17          // Crear menús
18          MenuClientes menuClientes = new MenuClientes();
19          MenuServicios menuServicios = new MenuServicios();
20          MenuEmpleados menuEmpleados = new MenuEmpleados();
21
22          menuClientes.mostrar();
23          menuServicios.mostrar();
24          menuEmpleados.mostrar();
25
26          int opcion = menuClientes.elegir();
27
28          switch (opcion) {
29              case 1:
30                  menuClientes.gestionClientes();
31              break;
32
33              case 2:
34                  menuServicios.gestionServicios();
35              break;
36
37              case 3:
38                  menuEmpleados.gestionEmpleados();
39              break;
40
41              case 4:
42                  menuTurnos.gestionTurnos();
43              break;
44
45              case 5:
46                  menuReportes.gestionReportes();
47              break;
48
49              case 0:
50                  System.out.println("Saliendo del programa...");
51                  System.exit(0);
52              break;
53
54              default:
55                  System.out.println("Opción no válida.");
56              break;
57          }
58      }
59  }

```

```

PELUQUERÍA FULLSTACK - MENÚ PRINCIPAL
1. Gestión de clientes
2. Gestión de servicios
3. Gestión de empleados
4. Gestión de turnos
5. Reportes
0. Salir
Opción: []

```

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 mantenable.

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.