Manual Tecnico
Call Center Admin Práctica #1
LAB LFP B+

Christian Javier Rivas Arreaga
202303204

Sistema de Gestión de Call Center

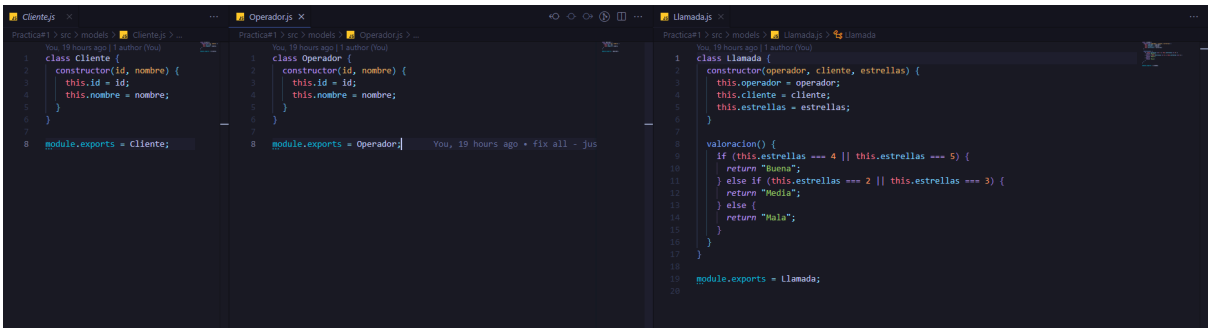
Aplicación desarrollada en Node.js puro para el análisis y gestión de llamadas de un centro de atención telefónica. El sistema permite cargar, procesar y generar reportes en formato HTML de las llamadas realizadas.

Tecnologías utilizadas:
- Node.js (JavaScript ES6+)
- FileSystem API nativa
- Path API nativa
- Estructuras de datos: Map, Set, Arrays

Arquitectura en Capas - Patrón MVC Modificado

src/
├── models/ → Entidades de negocio (Cliente, Operador, Llamada)
├── services/ → Lógica de negocio y servicios
├── utils/ → Utilidades y helpers
├── data/ → Almacenamiento de datos (CSV)
└── output/ → Archivos generados (HTML)

Flujo de datos
CSV → FileManager → CallCenter → HtmlExporter → HTML



Servicios Principales

FileManager.js - Procesamiento de CSV

// Función principal: cargarArchivo()
// Input: path (string) - Ruta del archivo CSV
// Output: Array<Llamada> - Array de objetos Llamada

Proceso de conversión CSV:
1. Lectura de archivo con fs.readFileSync()
2. Split por líneas (\n)
3. Parsing de cada línea: idOperador,nombreOperador,estrellas,idCliente,nombreCliente
4. Conversión de estrellas: "x;x;-;-" → conteo de "x"
5. Creación de objetos Operador, Cliente, Llamada
6. Return array de llamadas

CallCenter.js **- Lógica de Negocio**

// Métodos principales y su complejidad temporal:

cargarLlamadas(llamadas) → O(n) - Carga datos en memoria
listarHistorial() → O(n) - Mapeo de llamadas
contarValoraciones() → O(n) - Conteo y cálculo porcentajes
llamadasPorEstrellas() → O(n) - Agrupación por estrellas
listadoOperadores() → O(n) - Extracción única con Map
listadoClientes() → O(n) - Extracción única con Map
rendimientoOperador(nombre) → O(n) - Filtrado y cálculos
operadorConMejorValoracion() → O(n) - Iteración y comparación

```
65  rendimientoOperador(nombreOperador) {
66    const totalLlamadas = this.llamadas.length;
67    const llamadasAtendidas = this.llamadas.filter(
68      (l) => l.operador.nombre === nombreOperador
69    ).length;
70    const porcentajeAtencion =
71      totalLlamadas === 0 ? 0 : (llamadasAtendidas / totalLlamadas) * 100;
72    return {
73      operador: nombreOperador,
74      llamadasAtendidas,
75      porcentajeAtencion: porcentajeAtencion.toFixed(2) + "%",
76    };
77  }
```

```
  operadorConMejorValoracion() {
    const valoraciones = {};
    this.llamadas.forEach((l) => {
      if (!valoraciones[l.operador.id]) {
        valoraciones[l.operador.id] = {
          total: 0,
          conteo: 0,
          nombre: l.operador.nombre,
        };
      }
      valoraciones[l.operador.id].total += l.estrellas;
      valoraciones[l.operador.id].conteo++;
    });

    let mejorOperador = null;
    let mejorPromedio = 0;

    for (let id in valoraciones) {
      const promedio = valoraciones[id].total / valoraciones[id].conteo;
      if (promedio > mejorPromedio) {
        mejorPromedio = promedio;
        mejorOperador = valoraciones[id].nombre;
      }
    }

    return { operador: mejorOperador, promedio: mejorPromedio };
  }
}
```

Algoritmo de eliminación de duplicados:

```
43    listadoOperadores() {
44      const operadoresMap = new Map();      You, 12 minutes ago • all fix ~ just missing docs
45      this.llamadas.forEach((l) => {
46        operadoresMap.set(l.operador.id, {
47          id: l.operador.id,
48          nombre: l.operador.nombre,
49        });
50      });
51      return Array.from(operadoresMap.values());
52    }
```

GENERACIÓN DE REPORTES HTML

HtmlExporter.js - Plantillas dinámicas

```
class HtmlExporter {
  static generarTabla(datos, columnas, titulo, nombreArchivo) {
    let html = `
    <html>
    <head>
      <meta charset="UTF-8">
      <title>${titulo}</title>
      <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        h1 { text-align: center; }
        table { width: 80%; margin: auto; border-collapse: collapse; }
        th, td { border: 1px solid #ccc; padding: 8px; text-align: center; }
        th { background-color: #f4f4f4; }
      </style>
    </head>
    <body>
      <h1>${titulo}</h1>
      <table>
        <thead>
          <tr>${columnas.map((c) => `<th>${c}</th>`).join("")}</tr>
        </thead>
        <tbody>      You, 7 hours ago • all fix …
          ${datos
            .map(
              (fila) => `
            <tr>${columnas.map((c) => `<td>${fila[c]}</td>`).join("")}</tr>
            `
            )
            .join("")}
        </tbody>
      </table>
    </body>
    </html>
        `;
```

MANEJO DE RUTAS Y ARCHIVOS

```
  const outputDir = path.join(__dirname, "../output");
  if (!fs.existsSync(outputDir)) {
    fs.mkdirSync(outputDir);
  }

  fs.writeFileSync(path.join(outputDir, nombreArchivo), html, "utf-8");
  console.log(`Archivo ${nombreArchivo} generado en la carpeta output.`);
}
```