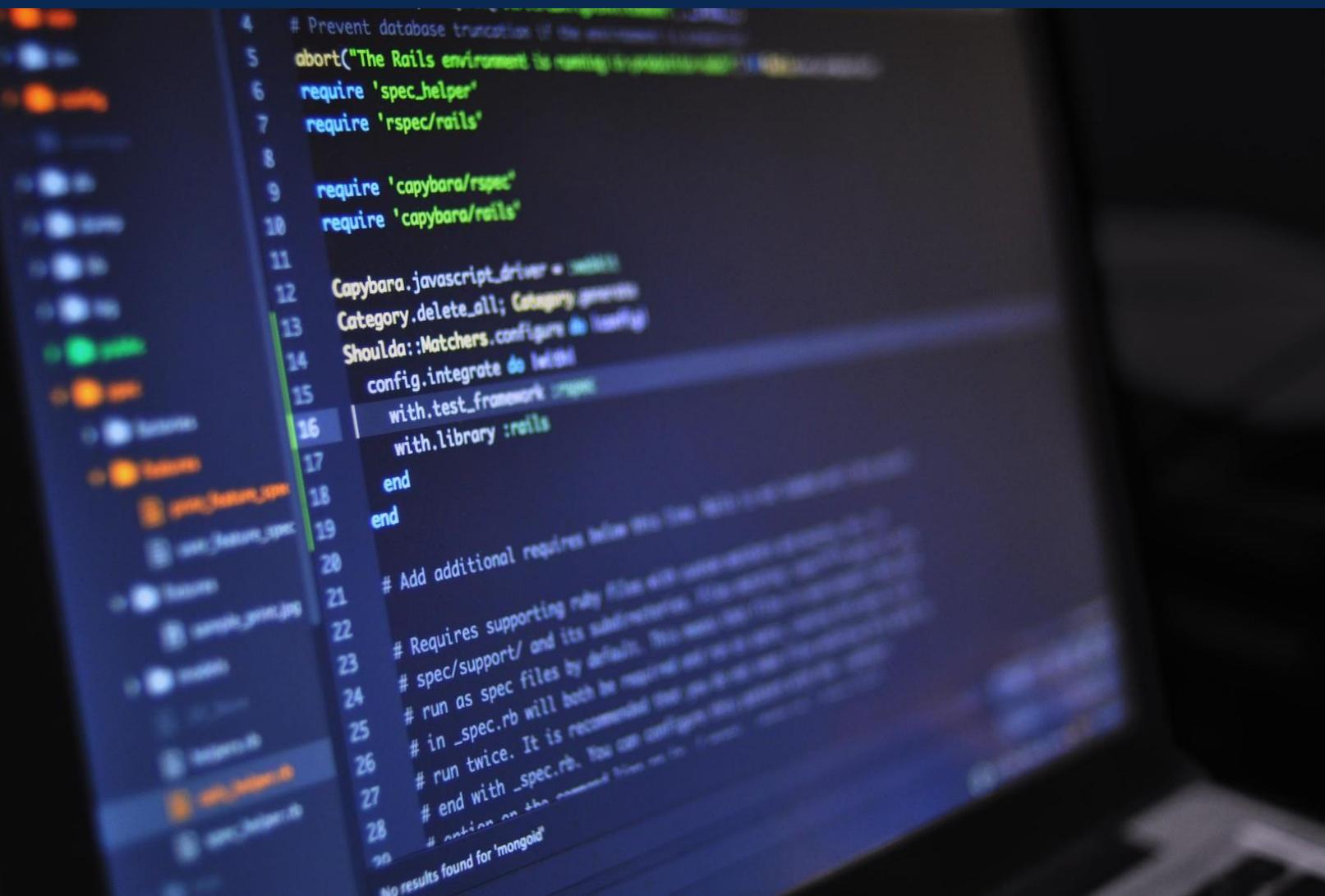


PROYECTO FINAL - ANÁLISIS Y REPORTES DE ÓRDENES

SOL SHEYLA YUCRA VILCACHAGUA

OBJETIVO DEL PROYECTO

- PROCESAR ARCHIVOS DE ÓRDENES Y GENERAR REPORTES AGREGADOS Y DETALLADOS.
- VALIDAR CREDENCIALES DE ACCESO Y MANEJAR ERRORES CRÍTICOS DE I/O O FORMATO.
- PRESENTAR MÉTRICAS POR CATEGORÍA, MÉTODO DE PAGO, ESTADO, CIUDAD, PROVEEDOR Y AÑO FISCAL.



```
4 # Prevent database truncation if the database is corrupt
5 abort("The Rails environment is running in production mode")
6 require 'spec_helper'
7 require 'rspec/rails'
8
9 require 'capybara/rspec'
10 require 'capybara/rails'
11
12 Capybara.javascript_driver = :webkit
13 Category.delete_all; Category.create!(name: "Electronics", description: "Electronics category")
14 Shoulda::Matchers.configure do |config|
15   config.integrate do |with|
16     with.test_framework :rspec
17     with.library :rails
18   end
19 end
20
21 # Add additional requires below this line if necessary
22
23 # Requires supporting files within the same directory as this file if you want to
24 # run as spec/ and its subdirectories. You can remove this line if you don't need
25 # it.
26 # in _spec.rb will both be required by default. If you specify your own include
27 # in _spec.rb, then that one will be used instead.
28 # end with _spec.rb. You can avoid writing this line if you don't need it.
29
30 # Note: mongoid needs to be required on the command line before running tests
31 #       or mongoid will not be available in the test environment
32
33 # No results found for 'mongoid'
```

- Lenguaje: Java (clases en paquete Modelo y Controlador).
- IDE recomendado: NetBeans / IntelliJ / Eclipse.
- Entrada: archivos de texto (CSV con cabecera).
- Salida: reportes por consola (System.out.printf).

TECNOLOGÍAS Y HERRAMIENTAS





ESTRUCTURA DEL REPOSITORIO (SEGÚN CAPTURAS)

```
└── Data/
    ├── Datos.txt
    └── Usuario.txt

└── src/
    ├── Modelo/
        ├── OrdenBase
        ├── DetalleFinanciero
        ├── ProductoVendido
        ├── InformacionLogistica
        ├── EntidadesRelacionadas
        ├── RegistroMaestro
        └── Usuario

    ├── Controlador/
        └── ControladorOrdenes.java

    └── Vista/
        └── Principal.java
```

- Archivo de entrada: CSV con al menos 15 columnas (se espera cabecera)

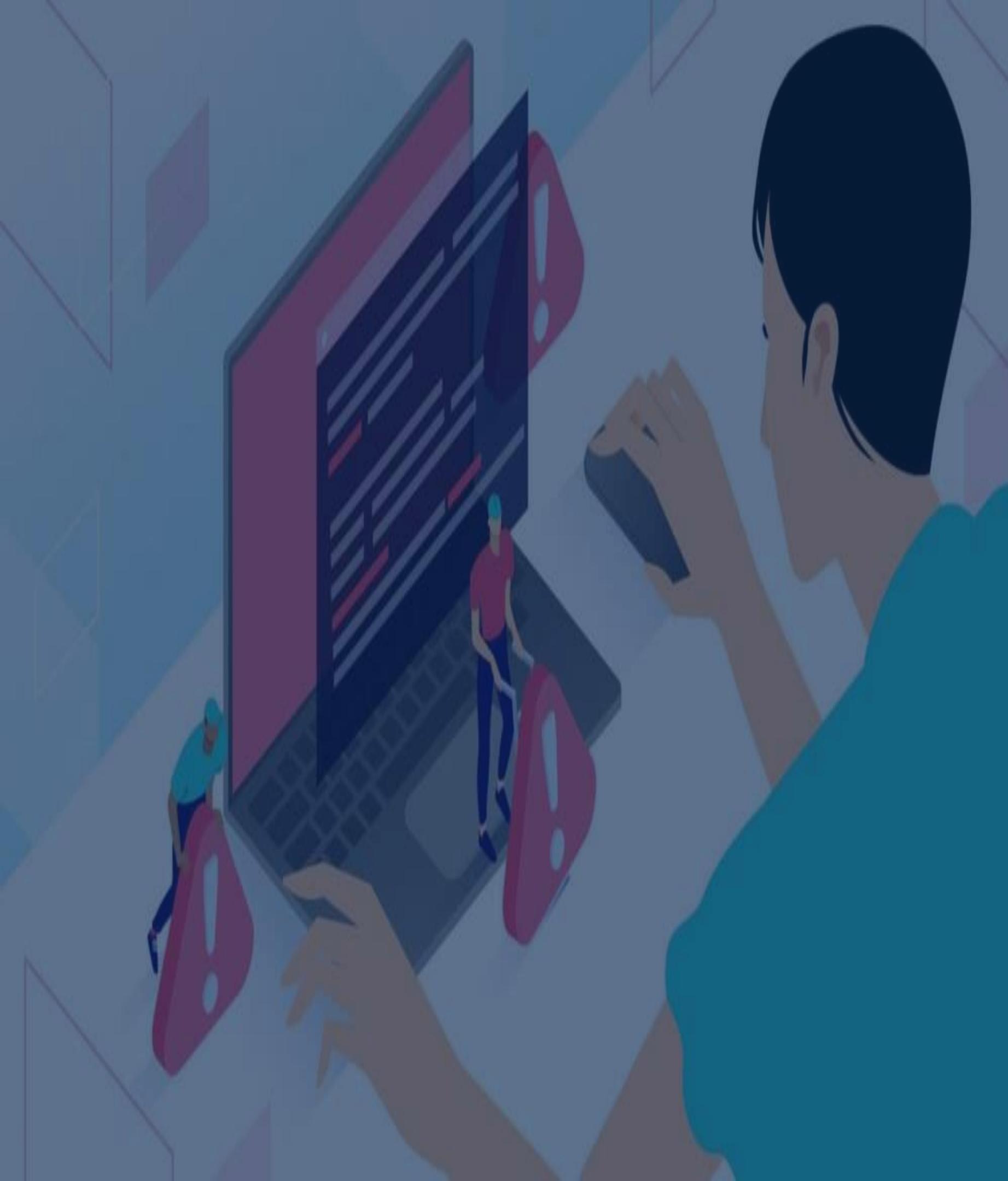
- **Controlador.ControladorOrdenes**: Lee archivo, valida usuario, carga registros y genera 7 reportes.
- **Modelo.RegistroMaestro**: Agrega objetos compuestos (OrdenBase, DetalleFinanciero, ProductoVendido, InformacionLogistica, EntidadesRelacionadas).
- **Modelo.DetalleFinanciero**: importe, costo de envío y método de pago.
- **Modelo.ProductoVendido**: nombre, cantidad, categoría.
- **Modelo.OrdenBase**: id, número y fecha de orden.

CLASES PRINCIPALES Y RESPONSABILIDADES



FLUJO DE DATOS (ALTO NIVEL)

- 1) Validación de credenciales leyendo Usuario.txt (usuario,contraseña).
- 2) Contar líneas del CSV (omitar cabecera).
- 3) Leer cada línea, parsear campos y construir RegistroMaestro.
- 4) Ejecutar distintos reportes que agregan/filtran por categoría, método, estado, ciudad, proveedor y año fiscal.



MANEJO DE ERRORES Y VALIDACIONES

- Clase anidada ExcepcionProcesamientoDatos con indicador esCritico.
- Validaciones: existencia de archivo, cabecera, número mínimo de campos (≥ 15).
- Advertencias: líneas con campos faltantes (se imprimen y se omiten).
- Errores críticos: fallos de I/O o NumberFormatException — se propagan como excepciones.

REPORTES IMPLEMENTADOS (7)

- 1. Reporte anual — monto total y detalle por orden.
- 2. Por categoría — total productos y porcentaje.
- 3. Por método de pago — monto total y porcentaje.
- 4. Por estado — conteo de órdenes y porcentaje.
- 5. Por ciudad — monto total por ciudad.
- 6. Por proveedor — conteo de órdenes por proveedor.
- 7. Por año fiscal — listado detallado para un año dado



- 1. Clonar el repo: git clone
https://github.com/SolisTipoJhanMarco/Proyecto_final.git

- 2. Abrir en NetBeans/Eclipse y ejecutar la clase Principal (src/Vista/Principal.java).

- 3. Asegurar que Data/Datos.txt y Data/Usuario.txt estén en la ruta esperada.

- 4. Parámetros: nombre del archivo de datos y del archivo de usuario (según implementación de Principal).

CÓMO EJECUTAR (SUGERIDO)



- Reportes impresos por consola con tablas formateadas (System.out.printf).

- Resumen: totales, porcentajes y listas detalladas por año fiscal.

- Capturas de pantalla: agregar salidas reales aquí (reemplazar esta diapositiva).

SALIDA / DEMO (EJEMPLO)



- El sistema procesa archivos CSV y genera reportes útiles para análisis comercial.

- Buenas prácticas: validar más columnas, usar colecciones dinámicas (Map) en lugar de arreglos con MAX_GRUPOS.

- Mejoras: exportar a CSV/Excel, interfaz gráfica o web, pruebas unitarias y manejo avanzado de errores.

CONCLUSIONES Y MEJORAS PROPUESTAS



GRACIAS!