Fecha de entrega: 6 de agosto del 2022 Hora de entrega: 7:00 pm

II Proyecto Programado

(ISB-21) PROGRAMACIÓN IV Profesor: Luis Daniel Vargas Gómez Nombre de los autores: Aleida Rivera Borge y Ronny Rafael Rodríguez Gómez Universidad Internacional San Isidro Labrador

Pag 2 Tabla de contenido

Pag1 Portada

Pag3 Resumen ejecutivo

Pag4 Introducción

5. Objetivo general

6 Objetivo específicos

7 Descripción del problema

8 Desarrollo del sistema

Diseño del sistema

Arquitectura General del Sistema

. Implementación y pruebas

Pag 12 conclusiones

10 recomendaciones

Pag Bibliografía (Referencias en formato APA)

Nota: Este documento está sujeto a cambios

Resumen ejecutivo

Introducción

Este proyecto consiste en el desarrollo de una aplicación de escritorio en Java, utilizando Swing como biblioteca principal para la creación de interfaces gráficas. El objetivo es construir un sistema que permita registrar, consultar, modificar y eliminar información relacionada con Clientes, Vehículos, Servicios y Órdenes de Trabajo, empleando archivos de texto (.txt ) como sistema de persistencia.

Además, se realiza una gestión ordenada de versiones mediante Git, y se genera la presente documentación técnica con enfoque modular y orientado a objetos.

5 Objetivo general

Objetivos del Proyecto

- Aplicar conceptos fundamentales de Java como clases, objetos, herencia, polimorfismo, interfaces y colecciones genéricas.

- Utilizar Swing para la creación de una interfaz gráfica intuitiva y funcional.

- Implementar persistencia de datos usando archivos de texto.

- Organizar el desarrollo con control de versiones usando Git.

- Documentar adecuadamente la arquitectura, funcionalidades y estructura del proyecto.

6 Objetivo específicos

7 Descripción del problema

8 desarrollo del sistema

Arquitectura General del Sistema

El sistema está estructurado en cuatro módulos principales:

* Gestión de Clientes
* Gestión de Vehículos
* Gestión de Servicios
* Gestión de Órdenes de Trabajo

Cada módulo está representado por clases específicas que manejan la lógica de datos, vistas Swing personalizadas y controladores encargados de los eventos de usuario.

Además, se emplea una entidad intermedia (Detalle\_Servicio) que gestiona la relación muchos-a-muchos entre Servicios y Órdenes de Trabajo, sin embargo, esta anuente a cambios para mejores que se adapten a la funcionalidad del código.

En cuanto a la persistencia de Datos

La persistencia se realiza mediante archivos de texto .txt. Cada módulo tiene su archivo correspondiente, con estructura clara para facilitar lectura, escritura y modificación.

Las operaciones incluyen:

* Escritura secuencial de nuevos registros.
* Lectura completa para mostrar en la interfaz.
* Actualización de datos mediante reescritura del archivo.
* Eliminación lógica o física de líneas según el caso.

Relaciones entre Entidades

Relaciones Lógicas del Sistema:

* Un Cliente puede tener múltiples Vehículos.
* Cada Vehículo está asociado a un único Cliente.
* Una Orden de Trabajo se vincula con un Cliente, un Vehículo y uno o más Servicios.
* La relación entre Órdenes y Servicios se maneja mediante una tabla intermedia (Detalle\_Servicio).

Las llaves primarias (ID\_Cliente, Placa, ID Servicio, ID\_Orden) y llaves foráneas aseguran integridad en la vinculación de registros. Llaves y Relación entre Entidades

| **Entidad Origen** | **Relación** | **Entidad Destino** | **Cardinalidad** |
| --- | --- | --- | --- |
| Cliente | tiene | Vehículo | 1: N |
| Cliente | tiene | Orden Trabajo | 1: N |
| Vehículo | pertenece a | Cliente | N:1 |
| Vehículo | está en | Orden Trabajo | 1: N |
| Orden Trabajo | usa | Detalle Servicio | 1: N |
| Servicio | está en | Detalle Servicio | N:1 |
|  |  |  |  |

## Diagrama Er Interfaz Gráfica Segundo Proyecto Programado

Diagrama

El contenido generado por IA puede ser incorrecto.

Interfaz Gráfica (Swing)

La interfaz se organiza en paneles por módulo, cada uno con:

* **Formulario** de entrada de datos (JTextField, JComboBox, JTextArea)
* **Tabla** para visualización (JTable)
* **Botones de acción** (JButton): Registrar, Consultar, Modificar, Eliminar
* **Mensajes de estado** (JLabel) y validaciones visuales
* Diseño limpio con layouts (BorderLayout, GridBagLayout, etc.) y estilos básicos

Cada vista permite interacción directa con el usuario, mostrando retroalimentación inmediata y permitiendo navegación entre módulos, dicha vista se pondrá a prueba para su aprobación, y esta anuente a cambios.

# Elaboración de la interfaz (Vista)

Diagrama de flujo: Gestión de Clientes (versión estructurada)

Inicio del Módulo

[Inicio: Módulo de Clientes]

↓

[Mostrar panel de opciones: Registrar, Consultar, Modificar, Eliminar]

**Registro de Cliente**

[Usuario selecciona "Registrar"]

↓

[Ingreso de datos: Nombre, Teléfono, Dirección]

↓

[Validar campos vacíos o inválidos]

↓

¿Datos válidos?

→ No → [Mostrar error en JLabel]

→ Sí → [Guardar en archivo .txt + Mostrar confirmación]

Componentes:

* btnRegistrar – botón para abrir el formulario de registro
* btnConsultar – botón para abrir la tabla de clientes
* btnModificar – botón para modificar cliente
* btnEliminar – botón para eliminar cliente

Componentes del Formulario:

* lblID – etiqueta para mostrar el ID generado es solo texto fijo por ahora
* txtNombre – campo para nombre
* txtTelefono – campo para teléfono
* txtDireccion – campo para dirección
* btnRegistrar – botón para guardar
* lblMensaje – etiqueta vacía para mostrar mensajes de validación es solo texto fijo por ahora

¿Qué pasa después de llenar?

Una vez que el usuario llena los datos y hace clic en Registrar (botón), el comportamiento esperado es:

1. Validar que los campos no estén vacíos
2. Generar un ID si aplica.
3. Guardar los datos en archivo .txt.
4. Limpiar el formulario o mostrar un mensaje de "Cliente registrado".

Ese formulario podría ser reutilizado también para:

* Cargar los datos de un cliente ya existente para modificarlos
* Pero para eso lo llamaría el **botón Modificar**, no Registrar

Consulta de Clientes

[Usuario selecciona "Consultar"]

↓

[Leer archivo de clientes]

↓

[Mostrar lista en JTable]

Componentes:

* tablaClientes – JTable (solo con columnas ID, Nombre, Teléfono, Dirección) **Aquí rellene 2 columnas con información, para hacer una prueba, sin embargo, aun no hago el llamado de toda la interfaz, sigo en proceso.**
* btnCerrar – para cerrar la ventana

Modificar y eliminar cliente desde la misma interfaz (ConsultaClientes.java)

La idea es seleccionar una fila en la tabla y luego:

* Ver los datos seleccionados
* Editarlos y guardar cambios
* Eliminar ese cliente si desea

**Modificación de Cliente**

[Usuario selecciona un cliente en JTable]

↓

[Mostrar datos actuales en formulario]

↓

[Modificar campos y confirmar cambios]

↓

[Actualizar archivo + Mostrar mensaje]

**Eliminación de Cliente**

[Seleccionar cliente desde JTable]

↓

[Confirmar eliminación]

↓

[Eliminar línea del archivo]

↓

[Actualizar vista + Mostrar confirmación]

**Resultado final**

ConsultaClientes.java debería tener:

* Una tabla (tablaClientes)
* Campos de texto que se llenarán al seleccionar una fila
* Botones para modificar o eliminar ese cliente

**Conexiones lógicas (PK y uso futuro de FK)**

* Cada cliente tendrá un ID\_Cliente único.
* Este ID será usado más adelante en los módulos de Vehículos y Órdenes como llave foránea, así que en esta vista debe generarse y visualizarse correctamente.

Control de Versiones

Se emplea Git para:

* Seguimiento de cambios en el código fuente: En espera del Codigo
* Manejo de ramas para nuevas funcionalidades; Detallar la rama
* Control de versiones estable en cada avance
* Documentación de commits significativa

Conclusiones

El proyecto refleja una correcta aplicación de los principios de la programación orientada a objetos, con enfoque modular y funcional. La interfaz gráfica fue desarrollada para facilitar la experiencia del usuario, mientras que la persistencia mediante archivos brinda una solución accesible para el manejo de datos sin base de datos.

10 recomendaciones

Pag Bibliografía (Referencias en formato APA)

<https://lucid.app/lucidchart/78ba35f3-5f89-4728-bc7c-7928112db8ab/edit?page=0_0&invitationId=inv_08121f2d-35dd-469b-af0d-c051ba69e917#>