	Computación	**Docente:** Ing. Daniel Díaz MsC.
	Programación Aplicada	**Período Lectivo: 64**

INFORME PROYECTO PRIMER PARCIAL – TIENDA ELECTROYECH

CARRERA: Computación		**ASIGNATURA: Programación Aplicada**
GRUPO:	3	**INFORME PROYECTO PRIMER PARCIAL – PROGRAMA FACTURACIÓN**
INTEGRANTES:	**Andy Mora, Steven Paredes, Pablo Carrera, Cristian Narváez, Alisson Valencia**	

OBJETIVO GENERAL:
Desarrollar un sistema integral de gestión de inventario y ventas para la tienda de electrónica "ElectroTech" que optimice las operaciones internas, mejore la experiencia del cliente en el punto de venta y proporcione herramientas eficientes para la administración de stock, ventas y proveedores.

OBJETIVOS ESPECIFICOS:

- Implementar un módulo de gestión de inventario, clientes y proveedores que permita a los empleados agregar, actualizar y eliminar estos mismos, asegurando un seguimiento preciso del stock, clientes y proveedores disponibles y la automatización de alertas para estos.

- Desarrollar una interfaz de punto de venta (POS) que facilite el proceso de ventas, permitiendo la selección de clientes, productos y el cálculo automático de totales, impuestos y descuentos, mejorando la eficiencia y precisión en las transacciones diarias.

- Implementar un sistema de seguridad y autenticación robusto que controle el acceso al sistema mediante credenciales válidas, roles y permisos específicos para cada usuario, garantizando la protección de datos sensibles.

- Desarrollar una base de datos para registrar y almacenar toda la información asociada con la tienda. Esto incluye detalles como los nombres de los clientes, los nombres de los proveedores, y el inventario disponible. De esta manera, se podrá gestionar y acceder a toda la información de una forma estructurada y ordenada, facilitando la organización y administración de los datos.

- Analizar cómo la clase GGrafics puede ser utilizada para mejorar el diseño y la interacción de interfaces en aplicaciones, evaluando su capacidad para crear elementos gráficos animados y visualmente atractivos que optimicen la experiencia del usuario y faciliten una presentación más impactante y eficiente de la información en aplicaciones de interfaz de usuario.

REQUERIMENTOS FUNCIONALES	**Gestión de Inventario:** • Agregar nuevos productos con detalles como nombre, descripción, precio, cantidad en stock y proveedor.
	Punto de Venta: • Realizar ventas seleccionando cliente y productos, registrando la cantidad vendida y calculando automáticamente el total, incluyendo impuestos y descuentos.
	Gestión de Proveedores: • Registrar proveedores con nombres completos, email, DNI, código, teléfono y razón social.
	Gestión de Clientes: • Mantener una base de datos de clientes con información relevante y permitir agregar y actualizar datos de clientes.
	Gestión de Empleados: • El administrador puede enrolar empleados y recuperar contraseñas.

	Interfaz de Usuario: • Diseño visual utilizando los colores institucionales: violeta, blanco y azul. • Prototipado de la aplicación considerando diseño, paleta de colores, tipografía e iconos. • Interfaz intuitiva y fácil de usar, con retroalimentación visual clara.
	Seguridad y Autenticación: • Implementar un sistema de autenticación seguro con roles y permisos específicos. • Permitir a los usuarios actualizar sus contraseñas.
ARQUITECTURA DEL SISTEMA - *Características del programa.*	

Arquitectura MVC:
El sistema se construirá bajo el modelo de arquitectura Modelo-Vista-Controlador (MVC), un paradigma que facilita la separación de preocupaciones en el desarrollo de aplicaciones. La **lógica de negocio** se encapsula en el Modelo, que maneja los datos y reglas de negocio; la **presentación** se gestiona a través de la Vista, encargada de mostrar los datos al usuario; y el **control** se lleva a cabo mediante el Controlador, que actúa como intermediario, procesando las entradas del usuario y actualizando el Modelo y la Vista en consecuencia.
Esta arquitectura permite una mayor modularidad, facilitando el mantenimiento y escalabilidad del sistema, además de promover el desarrollo colaborativo y la reutilización de componentes.

Base de Datos:
El sistema opera utilizando una base de datos que ha sido creada con la aplicación MySQL. Esta aplicación es fundamental para el almacenamiento, la gestión y la recuperación de toda la información contenida en la base de datos. MySQL permite organizar los datos de manera eficiente y acceder a ellos cuando sea necesario, asegurando que toda la información esté disponible y bien estructurada para su uso en el sistema.

GGrafics:
El sistema funciona empleando la clase GGrafics para crear un diseño de la interfaz de la tienda que sea más animado y atractivo. Al utilizar GGrafics, se puede mejorar significativamente la apariencia visual de la presentación, haciendo que la interfaz sea más dinámica y cautivadora para los usuarios. Esta clase permite incorporar elementos gráficos avanzados y animaciones, lo que contribuye a una experiencia de usuario más envolvente y visualmente agradable.

Validación y Expresiones Regulares:
El sistema implementará una validación de entrada utilizando expresiones regulares, lo que garantiza que los datos ingresados por los usuarios cumplan con los formatos predefinidos. Esto es crucial para mantener la integridad de la base de datos y evitar errores comunes como entradas mal formateadas o inconsistencias.
Por ejemplo, se utilizarán expresiones regulares para validar correos electrónicos, números de teléfono, códigos de productos y otros campos críticos.

Parámetros del Sistema:
El sistema estará totalmente parametrizado, lo que permitirá una configuración dinámica de sus componentes sin necesidad de modificar el código fuente. Esto incluirá parámetros para ajustar impuestos, descuentos, alertas de stock y configuraciones específicas de la tienda.
La parametrización facilitará la personalización del sistema según las necesidades cambiantes de "ElectroTech" y permitirá una rápida adaptación a nuevas políticas o regulaciones.

Algunos de los conceptos de Programación:

- **Herencia:** Las clases derivadas en el sistema heredan características y comportamientos de clases base, promoviendo la reutilización del código y facilitando la extensión de funcionalidades. Por ejemplo, una clase base Producto podría tener clases derivadas como Electrodomestico y Dispositivo Movil que hereden propiedades comunes, pero implementen comportamientos específicos.
- **Polimorfismo:** Este principio permite que diferentes clases se comporten de manera distinta cuando se llama a métodos idénticos, dependiendo del contexto. Esto se logrará mediante métodos sobrecargados y sobrescritos, permitiendo a las subclases definir su propia versión de un método.
- **Encapsulamiento:** Los datos y métodos se encapsularán dentro de clases, restringiendo el acceso directo a los detalles internos y exponiendo solo lo necesario a través de interfaces públicas. Esto protegerá la integridad de los datos y reducirá la posibilidad de errores por manipulación indebida.
- **Boxing y Unboxing:** Se utilizará boxing y unboxing para las operaciones en ventas, lo cual implica convertir tipos de datos primitivos en objetos (boxing) y viceversa (unboxing). Esto permitirá una gestión más flexible y eficiente de los tipos de datos, facilitando operaciones como la manipulación de precios y cantidades durante el proceso de venta.
- **Librerías:** El desarrollo del sistema se apoyará en una serie de tecnologías y herramientas modernas, incluyendo una librería personalizada (libreriaV2) que contiene clases genéricas y métodos para manejo de archivos y validaciones.

Diseño del proyecto:

- Optamos por desarrollar una variedad de interfaces con diseños limpios y funcionales, asegurando que el usuario no se pierda al utilizar el programa. Utilizamos WindowBuilder de Java, para crear interfaces

visualmente atractivas. Además, contamos con una clase ViewComponents que centraliza métodos para mejorar la apariencia y usabilidad de las interfaces.

- El sistema está dividido en dos roles principales: ADMINISTRADOR y EMPLEADO, cada uno con sus propias interfaces específicas adaptadas a sus necesidades y responsabilidades.

A continuación, una imagen del Login principal para apreciar su diseño:

- Se opto por un diseño con colores agradables, interfaces llamativas sin perder de lado la parte limpia de la vista para que sea atractivo para el usuario.

Pagina Web: https://josumirandae.github.io/TechVantajeSolutionss.github.io/

RESULTADOS OBTENIDOS:

1. **Mayor Eficiencia en el Desarrollo:**
   El uso de la librería genérica contribuyó significativamente a la eficiencia del desarrollo al proporcionar componentes predefinidos y funciones comunes que se pueden reutilizar en todo el sistema. Esto permitió a los desarrolladores enfocarse en la lógica de negocio específica en lugar de tener que codificar repetidamente funcionalidades básicas. Además, la facilidad de diseño proporcionada por WindowBuilder y la clase ViewComponents agilizó el proceso de creación de interfaces de usuario, permitiendo la rápida iteración y revisión de los diseños.

2. **Validación de Datos:**
   La implementación de expresiones regulares para validar la entrada de datos mejoró la seguridad y la integridad de los datos del sistema al garantizar que se ingresen únicamente datos válidos. La retroalimentación visual mediante la coloración de campos de entrada (verde para datos válidos, rojo para datos inválidos) no solo ayuda al usuario a corregir errores fácilmente, sino que también ayuda a prevenir la inserción de datos incorrectos o maliciosos en el sistema.

3. **Optimización del Control de Inventarios para Electrotech:**
   La aplicación desarrollada ofrece a Electrotech una solución integral para gestionar eficientemente su inventario. Mediante el seguimiento detallado de los productos, incluyendo su cantidad, ubicación y movimientos, la empresa puede garantizar un control preciso de sus existencias en todo momento. Esto permite una gestión proactiva de la reposición de productos y una reducción significativa de pérdidas debido a excesos o faltantes en el inventario. Además, la generación de informes y análisis en tiempo real proporciona a Electrotech una visión profunda de su inventario, facilitando la toma de decisiones informadas para optimizar los niveles de stock y maximizar la eficiencia operativa.

CONCLUSIONES:

- **Importancia del Diseño Intuitivo:** Un diseño de interfaz limpio y bien estructurado es crucial para la usabilidad del sistema. Los usuarios pueden operar el software de manera más efectiva cuando las interfaces son intuitivas y fáciles de navegar. La retroalimentación visual mediante colores (verde para confirmación, rojo para errores) y notificaciones claras ayuda a guiar al usuario a través del proceso, aumentando su satisfacción y productividad.

- **Control en la entrada de los campos:** La validación mediante expresiones regulares contribuye a garantizar que la información garantizada por los usuarios este correctamente escrita o definida. Las notificaciones visuales sobre la validez de los datos proporcionados, con indicadores de color, no solo mejoran la experiencia del usuario, sino que también ayudan a prevenir errores y posibles vulnerabilidades.

- **Estandarización y Modularidad en el Desarrollo:** La creación de componentes reutilizables y estandarizados, como los proporcionados por la librería genérica utilizada, no solo acelera el desarrollo sino que también asegura consistencia en el diseño. La modularidad en el desarrollo permite una mejor organización del código y facilita futuras actualizaciones y mantenimiento del sistema.

RECOMENDACIONES:

- **Comentarios Claros y Concisos:** Asegurarse siempre de incluir comentarios claros y concisos en el código para facilitar su comprensión y mantenimiento futuro. Esto ayudará a otros desarrolladores (y a nosotros mismos) a entender rápidamente el propósito y el funcionamiento de cada sección del código, lo que facilitará la identificación y corrección de posibles problemas.

- **Orden con las interfaces y claridad en nombres:** Siempre mantener un orden coherente para las respectivas clases, más cuando se trabaja con una gran cantidad de clases. Siempre nombres claros y precisos para evitar confusiones al momento de programar.

A continuación, el respectivo diagrama de clases: