1.Resumen.

Este proyecto de QuantifyQ tiene el objetivo de desarrollar una plataforma sencilla, utilizando el lenguaje de programación Java, enfocado en la gestión productos con ingreso de usuarios existentes o que se puedan crear. La aplicación implementará la arquitectura MVC para garantizar una separación clara entre la lógica de negocio, la interfaz gráfica y el manejo de eventos, siguiendo los principios de la POO como abstracción, encapsulamiento, herencia y polimorfismo.

La interfaz de la aplicación estará construida con componentes gráficos como JFrame, JPanel, JLabel, JButton y JTextField, permitiendo una experiencia de usuario intuitiva. En la pantalla de inicio se solicitará un usuario y contraseña para acceder al sistema, si no se encuentra en él sistema tendrá la opción de crear un usuario desde esta parte. Una vez autenticado, el usuario será dirigido al menú principal, desde donde podrá:

- Visualizar, crear, actualizar y eliminar datos de productos del inventario.
- Consultar créditos del sistema, como parte del reconocimiento al desarrollador.

El desarrollo de esta plataforma responde a las historias de usuario definidas previamente y busca cumplir con criterios de aceptación concretos, así como con entregables funcionales y documentados.

2.Palabras Claves.

Java: Lenguaje de programación principal del proyecto.
NetBeans: Entorno de desarrollo usado para construir la aplicación.
MVC: Arquitectura recomendada para organizar el proyecto.
CRUD: Las funcionalidades clave del sistema (Crear, Leer, Actualizar, Eliminar).
Inventario: Objetivo principal del proyecto (gestión de productos y existencias).

3.Abstract.

This project aims to develop a simple desktop portal using the Java programming language, allowing the management of users and inventory products through a graphical interface. The application is designed under the MVC architecture and applies core OOP principles such as abstraction, encapsulation, inheritance, and polymorphism, promoting good software development practices.

The system allows the creation of users who can access the portal via login credentials and interact with the application through a user-friendly interface built with components such as JFrame, JPanel, JButton, JLabel, and JTextField. Once logged in, users can perform full CRUD operations (Create, Read, Update, Delete) on product records and navigate through features like user management and credit display

4.Introducción.

QuantifyQ es una microempresa colombiana en desarrollo desde el año 2025 con el objetivo de ofrecer soluciones tecnológicas y de análisis enfocadas en la optimización de procesos. Desde su creación hacemos la apuesta por la innovación y el desarrollo de herramientas digitales accesibles para pequeñas y medianas empresas que buscan mejorar la trazabilidad y eficiencia en sus operaciones.

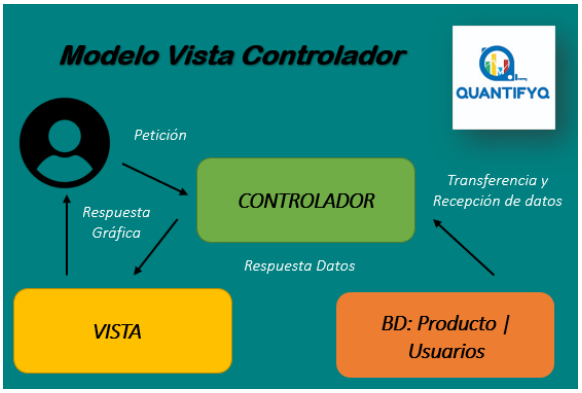
El proyecto actual, desarrollado como parte directa a el manejo de inventarios, está diseñado y pensado en una aplicación de escritorio funcional, que permita gestionar productos con ingreso de portal de usuarios, teniendo conexión directa con bases de datos, construyendo un sistema de inventario ágil, intuitivo y adaptable a las necesidades de sus aliados comerciales.

5.MVC (Modelo Vista Controlador).

- **Modelo**: Contiene las clases que representan los datos del sistema, como Producto y Usuario. También gestiona la conexión y comunicación con la base de datos (guardar, consultar, actualizar y eliminar registros).

- **Vista**: Incluye todos los elementos gráficos que interactúan con el usuario, como las pantallas de inicio de sesión, créditos, el apartado de productos. Estos elementos están construidos con componentes JFrame, JPanel, JTable, JTextField, entre otros.

- **Controlador**: Es el encargado de recibir las acciones del usuario (por ejemplo, hacer clic en “Crear producto” o “Iniciar sesión”), procesarlas y coordinar la respuesta adecuada entre el modelo y la vista. También valida datos y gestiona el flujo de navegación dentro del sistema.



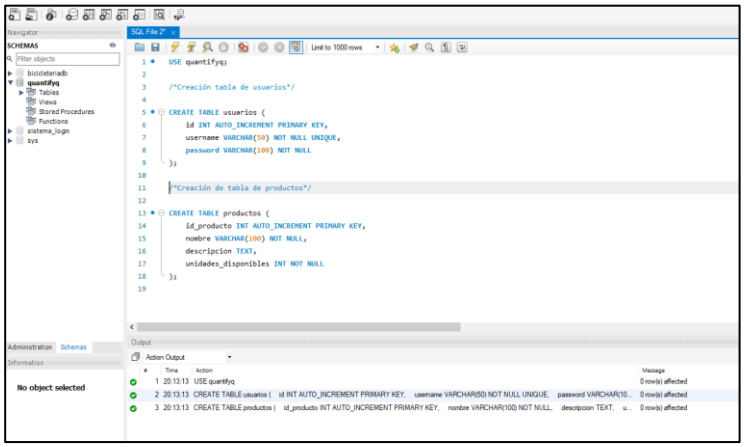
6.Diccionario de datos.

Se realiza la creación de la base de datos de QuantifyQ para el almacenamiento de las tablas de usuarios y productos que serán usadas en el desarrollo de la aplicación en Java.

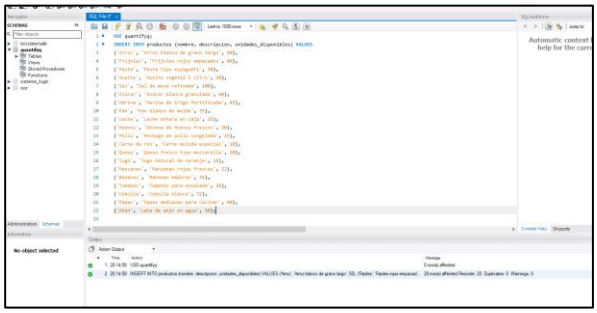
Tabla	Campo	Tipo de Dato	Restricciones	Descripción
usuarios	id	INT	AUTO_INCREMENT, PRIMARY KEY	Identificador único del usuario
usuarios	username	VARCHAR(50)	NOT NULL, UNIQUE	Nombre de usuario (único)
usuarios	password	VARCHAR(100)	NOT NULL	Contraseña del usuario
productos	id_producto	INT	AUTO_INCREMENT, PRIMARY KEY	Identificador único del producto
productos	nombre	VARCHAR(100)	NOT NULL	Nombre del producto
productos	descripcion	TEXT	-	Descripción del producto
productos	unidades_disponibles	INT	NOT NULL	Cantidad disponible en inventario

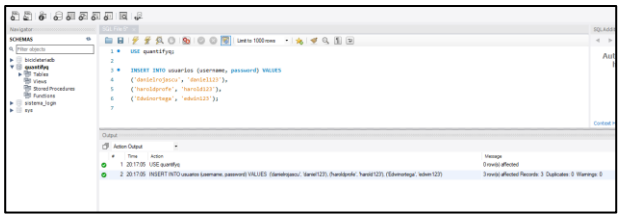
Creación de la base de datos:

Creación de las tablas:



Ingesta de datos en la tabla de productos y usuarios:





7.Desarrollo Proyecto.

1. Creamos un proyecto nuevo: Java with Ant - Java Applitation.
2. Declaramos el nombre: QuantifyqApp.
3. Desactivamos la opción del Main.
4. Creamos el Proyecto.
5. Creamos los paquetes: source packages - New - Java Package y creamos los siguientes:

- db: clase con la conexión a la base de datos.
- models: clases usuario y contraseña.
- views: JFrame From de (Crear usuarios, Créditos, Inicio, Productos).
- main: clase main.

6.Descargamos la librería de conexión necesaria, en este caso MySQL.

- https://dev.mysql.com/downloads/connector/j/
- Seleccionamos el ZIP Archive.
- Cargamos al proyecto, para esto expandimos nuestro proyecto, opción de librerías, clic derecho "Add JAR/Folder" ubicamos él .jar de conexión e importamos.

7.Comenzamos con la creación de las clases y diseñadores.

- Creación ConexionBD.Java: en el paquete db, damos clic derecho y seleccionamos new + JavaClass.
- Creación Usuario.Java: en el paquete models, damos clic derecho y seleccionamos new + JavaClass.
- Creación Producto.Java: en el paquete models, damos clic derecho y seleccionamos new + JavaClass.
- Creación Main.Java: en el paquete main, damos clic derecho y seleccionamos new + JavaClass.
- Creación JFrame From: en el paquete views, damos clic derecho y seleccionamos new + JFrame From

Inicio.Java:

- JLabel con la imagen del logo.
- JLabel título.
- JLabel para Usuario.
- JTextField para el usuario digitar.
- JLabel para Contraseña.
- JPasswordField para el usuario digitar.
- JButton para Iniciar sesión.
- JButton para Crear usuario.
- JButton para Créditos @.
- JButton para Documentación GitHub.

Variables:

- JTextField usuario: txtUsuario
- JPasswordField: txtContrasena
- JButton Iniciar sesión: btnIniciarSesion

CrearUsuario.Java:

- JLabel con la imagen del logo.
- JLabel título.
- JLabel para Usuario.
- JTextField para el usuario digitar.
- JLabel para Contraseña.
- JPasswordField para el usuario digitar.
- JButton Crear Usuario

Variables:

- JTextField usuario: txtNuevoUsuario
- JPasswordField: txtNuevaContrasena
- JButton Iniciar sesión: btnGuardarUsuario

Creditos.Java:

- JLabel con la imagen del logo.
- JLabel título.
- JLabel para los mensajes de créditos.
- JButton cerrar venta

Variables:

- JButton Cerrar: btnCerrarCreditos

Productos.Java:

- JLabel con la imagen del logo.
- JLabel título.
- JTable (JScrollPane) tabla que interactúa con la base de datos.
- JButton Ver Productos.
- JButton Crear Productos.
- JButton Actualizar Productos.
- JButton Eliminar Productos.

Variables:

- JButton Ver: btnVerProductos
- JButton Crear: btnCrearProducto
- JButton Actualizar: btnActualizarProducto
- JButton Eliminar: btnEliminarProducto
- JTable: tablaProductos

8.Definición de P.O.O y Conceptos.

- Abstracción: Es la manera donde represento lo más importante de algo de una clase.
  Proyecto: En las clases producto, usuario y models donde realizo la creación de atributos o métodos (nombre, descripción, contraseña, etc.)

```
public class Producto {
    private int id;
    private String nombre;
    private String descripci
    private int unidades;
```

- Encapsulamiento: Protejo los datos en una clase y que su ingreso sea de manera controlada.
  Proyecto: esta en las mismas clases, pero es donde aplicamos los Get o Set.

```
// Getters y Setters
public int getId() { return id; }
public void setId(int id) { this.id = id; }

public String getNombre() { return nombre; }
public void setNombre(String nombre) { this.nombre = nombre; }

public String getDescripcion() { return descripcion; }
public void setDescripcion(String descripcion) { this.descripcion = descripcion; }

public int getUnidades() { return unidades; }
public void setUnidades(int unidades) { this.unidades = unidades; }
```

- Polimorfismo: Puedo usar un mismo método o acción de diferentes formas.
  Proyecto: Cuando uso el método de ActionPerformed ya que realiza diferentes cosas según la venta que tenga.

```
185     }
186
187     private void btnCrearProductoActionPerformed(java.awt.event.ActionEvent evt) {
```

- Herencia: Cuando una case hereda cosas de otra.
  Proyecto: en el proyecto de nosotros creemos que cuando se crean los diseñadores en todas aplicamos el JFrame.
- JFrame: Contenedor de diseño (todo lo que está en recuadro morado)

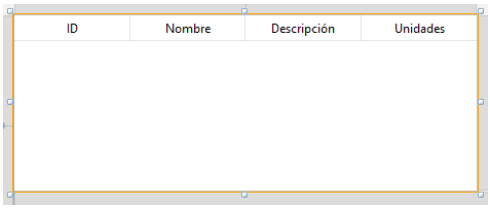
- JLabel: Objeto para almacenar las imágenes o textos fijos.

- JButton: Objeto que ejecuta una acción programada, en el caso nuestra muestra, crear, actualiza, etc. Según lo que le asignemos en el script.

- JTextField: Objeto que permite al usuario digitar valores visibles (Ver que escribe).

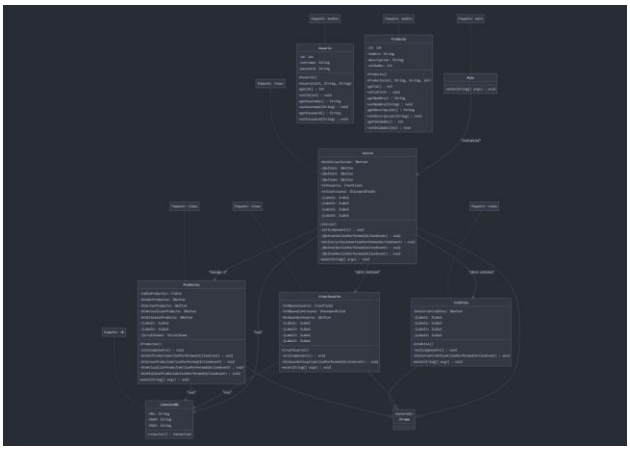
- JPasswordField: Objeto que permite al usuario digitar, pero no ver lo que escribe.

- JTable - JScrollPane: Objeto que visualizar una tabla y en esta misma que se active la barra de desplazamiento.



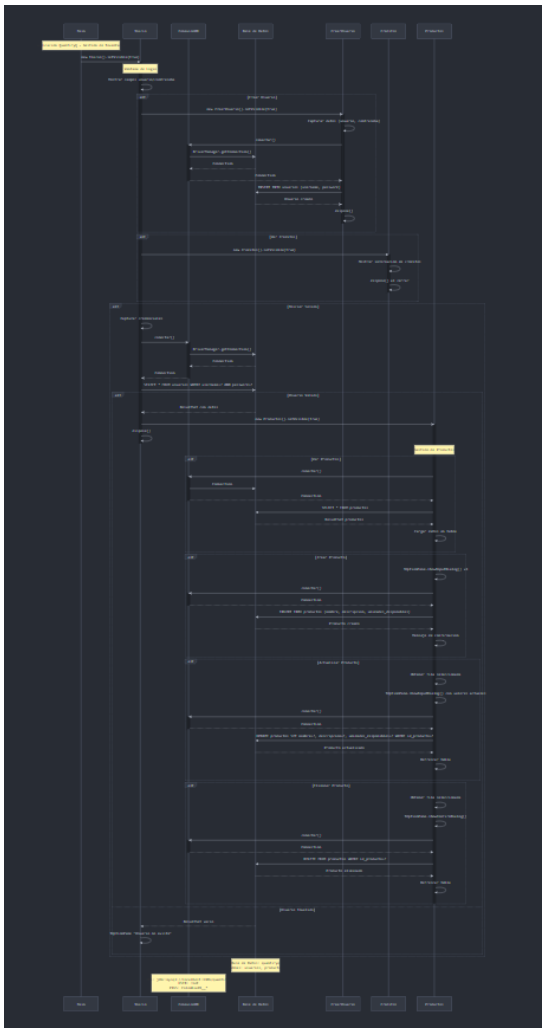
9.Diagrama de Clases.

En la documentación de GitHub de deja el siguiente archivo para visualizarlo mejor (diagrama_clases_quantifyq.mermaid).



10.Secuencia de datos.

En la documentación de GitHub de deja el siguiente archivo para visualizarlo mejor (quantifyq_sequence_diagram.mermaid).



11.Casos de uso

Flujo de eventos principal		
Paso	**Acción del usuario**	**Respuesta del sistema**
1	El usuario abre la aplicación QuantifyQ	Se muestra la ventana de inicio.
2	Ingresa su usuario y contraseña	El sistema verifica las credenciales contra la base de datos.
3	Clic en "Iniciar sesión"	Si las credenciales son correctas, se abre la ventana de productos.
4	Clic en "Ver productos"	El sistema consulta la base de datos y muestra los productos en una tabla.
5	Clic en "Crear producto"	Se solicitan nombre, descripción y unidades; el producto se guarda en la BD.
6	Clic en "Actualizar producto"	Se selecciona un producto, se editan los campos y se actualizan en la BD.
7	Clic en "Eliminar producto"	Se selecciona un producto y se elimina tras confirmar.
8	Clic en "Créditos"	Se abre una ventana con el mensaje de créditos.
9	Clic en "Ir al sitio web"	El sistema abre el navegador con la URL definida.

Flujo Alternativo		
Paso	**Acción del usuario**	**Respuesta del sistema**
3A	Usuario no existe o contraseña incorrecta	Muestra el mensaje "Usuario no existe"
3B	El usuario no está registrado	Puede hacer clic en "Crear usuario"
5A	Se deja algún campo vacío al crear producto	El sistema muestra mensaje de validación
6A	No se selecciona ningún producto	Muestra "Selecciona un producto para actualizar."
7A	No se selecciona ningún producto	Muestra "Selecciona un producto para eliminar."

Reglas de negocio aplicadas

- Solo usuarios autenticados pueden acceder al módulo de productos.
- No se permite crear productos sin nombre, descripción o unidades válidas.
- No se puede actualizar o eliminar si no se selecciona un producto de la tabla.

13.Perfiles de creadores

Marcos Daniel Rojas Cuervo: **Estudiante de Ingeniería de Software | Especialista en Inteligencia de Negocios y Datos**

Daniel es estudiante de tercer semestre de Ingeniería de Software con una sólida trayectoria de 9 años en el mundo de los datos y la inteligencia de negocios. A lo largo de su carrera, ha liderado y participado en el desarrollo de soluciones analíticas utilizando herramientas como Power BI, Qlik View, Qlik Sense, Qlik Cloud y Looker Studio.

Cuenta con experiencia en procesos ETL mediante SSIS, herramientas de integración de datos y administración de plataformas Qlik, gestionando configuraciones, tareas, licencias y entornos completos a través de QMC.

Actualmente, está ampliando su perfil hacia el desarrollo de software, trabajando con tecnologías como HTML, CSS, JavaScript, Java y Python, así como con servicios de AWS como S3, Redshift, Athena y Glue. Su enfoque combina conocimientos técnicos en programación con una visión analítica centrada en el uso estratégico de los datos.

Tiene un nivel de inglés básico/intermedio, lo que le permite desenvolverse en contextos técnicos con soltura.

Edwin Ricardo Ortega Cuervo: **Estudiante de Ingeniería de Software | Analista de Datos con Enfoque en BI y Visualización**

Edwin es estudiante de Ingeniería de Software con una fuerte orientación hacia el análisis y visualización de datos. Su perfil se caracteriza por la capacidad para organizar, segmentar y presentar información con precisión y alto valor agregado.

Tiene experiencia en el diseño de tableros de control en Excel, así como en el desarrollo de modelos analíticos utilizando herramientas de Business Intelligence como Power BI y Qlik. Ha aplicado estas soluciones en distintas industrias, aportando indicadores clave para la toma de decisiones estratégicas.

Edwin combina habilidades analíticas con un enfoque práctico en herramientas de productividad como Excel, Word y PowerPoint, fortaleciendo su perfil como generador de reportes e informes de alto impacto.

Su visión está centrada en promover la transformación digital a través del uso inteligente de los datos, con énfasis en sostenibilidad, rentabilidad y alfabetización digital.