Documentación general proyecto Final de curso análisis de sistemas

George Alexander Rodríguez Benítez, Kiara María Orellana Yón, Melissa Jazmín Alvarez González, José Alejandro Oscal Morales, Wallington Lesther Eliu Mente López, Luis Fernando cardona campos, Velvet Julissa González Moreno, Abner Julián Abimael Canté Rodríguez, Enso Fabricio Gonzalez Moreno, Kevin Alejandro Rodas Balcárcel, Dionicio Véliz Cerón, Jorge Alejandro Lemus Ramírez, Andy Alejandro Girón Loaiza.

Universidad Mariano Gálvez de Guatemala

junio 03 de 2023

Notas de los autores

George Alexander Rodríguez Benítez, Kiara María Orellana Yón, Melissa Jazmín Alvarez González, José Alejandro Oscal Morales, Wallington Lesther Eliu Mente López, Luis Fernando cardona campos, Velvet Julissa González Moreno, Abner Julián Abimael Canté Rodríguez, Enso Fabricio Gonzalez Moreno, Kevin Alejandro Rodas Balcárcel, Dionicio Véliz Cerón, Jorge Alejandro Lemus Ramírez, Andy Alejandro Girón Loaiza.

Facultad de Ingeniería en Sistemas, Universidad Mariano Gálvez

La correspondencia relacionada con esta investigación debe ser dirigida a nombre de los autores, Universidad Mariano Gálvez, Sanarate

# Índice

[Índice ii](#_Toc136723098)

[1. Toma de requerimientos 1](#_Toc136723099)

[1.1.1Planteamiento del problema 1](#_Toc136723100)

[1.1.2. Objetivos para resolver 2](#_Toc136723101)

[1.1.3. Entrevista 2](#_Toc136723102)

[1.1.4. Funcionalidades 4](#_Toc136723103)

[1.1.5 Presupuesto 5](#_Toc136723104)

[1.2. Diagramas de flujo 6](#_Toc136723105)

[1.2.1. Diagramación de actividades 6](#_Toc136723106)

[1.3. Casos de uso 7](#_Toc136723107)

[1.3.1. Diagramas de secuencias 7](#_Toc136723108)

[1.4. Base de datos 10](#_Toc136723109)

[1.4.1. Modelo entidad - relación 10](#_Toc136723110)

[1.5. Base de datos (SQL) 10](#_Toc136723111)

[1.5.5. MySQL script 10](#_Toc136723112)

[1.6 División del proyecto 34](#_Toc136723113)

[Proyect Manager 34](#_Toc136723114)

[Frontend – aplicación 34](#_Toc136723115)

[Backend – API 34](#_Toc136723116)

[Base de datos 35](#_Toc136723117)

[Servidores 35](#_Toc136723118)

[Documentación 35](#_Toc136723119)

[1.6.1. Paleta de colores 35](#_Toc136723120)

[Acerca de #030300 36](#_Toc136723121)

[Acerca de #DB8300 37](#_Toc136723122)

[Acerca de #EAA001 37](#_Toc136723123)

[Acerca de #F9C700 38](#_Toc136723124)

[Acerca de #FFE11D 39](#_Toc136723125)

[1.6.2. Logotipo 40](#_Toc136723126)

[2 Backend API REST 40](#_Toc136723127)

[2.1 Lenguaje: PHP 40](#_Toc136723128)

[2.1 .1 Configuración de las rutas de API 40](#_Toc136723129)

[2.1.2 Pruebas de desarrollo por Postman 42](#_Toc136723130)

[2.1.3. Implementación de seguridad por tokens 42](#_Toc136723131)

[2.2 Frontend 43](#_Toc136723132)

[2.2.3 Login 43](#_Toc136723133)

# 1. Toma de requerimientos

## 1.1.1Planteamiento del problema

En la tienda de herramientas "Ferrimundo", actualmente se lleva a cabo la gestión del inventario de manera manual, utilizando registros en papel. Cada vez que se efectúa una compra o venta de productos, se requiere una actualización física del inventario mediante anotaciones en hojas de papel. Este enfoque presenta diversos desafíos y limitaciones que afectan la eficiencia y precisión en la administración del inventario. Por lo tanto, se plantea la necesidad de implementar una solución tecnológica, como una aplicación de inventario, para abordar estos problemas y mejorar el funcionamiento de la tienda.

El principal problema al utilizar registros en papel para gestionar el inventario es la falta de precisión y la falta de actualización en tiempo real de los niveles de stock. El proceso manual es susceptible a errores humanos, como anotaciones incorrectas, omisiones o registros duplicados.

En la actualidad, las ventas en línea son una parte crucial de la estrategia comercial de muchas empresas, incluyendo las tiendas de herramientas. Por lo tanto, surge la necesidad de desarrollar una aplicación específica que permita el control del inventario, así como la gestión de compras a proveedores y ventas, adaptada a las necesidades y características particulares de la tienda de herramientas. Esto permitirá aprovechar las oportunidades que brinda un sistema de inventario más eficiente.

## 1.1.2. Objetivos para resolver

* Mejorar la exactitud y eficiencia en el manejo del inventario. Se pretende disminuir los errores humanos con el fin de lograr una gestión de inventario más precisa y eficiente.
* Asegurar que los niveles de inventario se mantengan precisos y actualizados en tiempo real.
* Llevar un control de los proveedores y las compras realizadas a los mismos.
* Llevar un control de las ventas realizadas a los clientes, así como un control de los clientes.

## 1.1.3. Entrevista

Cliente: ¡Buena tarde! Gusto en conocerlo, soy el dueño de una ferretería Ferrimundo y estoy interesado en una aplicación móvil para que mis empleados puedan llevar un control de inventario. ¿Puedes ayudarme con esto?

Desarrollador: ¡Buenos días! Claro, estaré encantado de ayudarte con eso. ¿Podrías proporcionarme más detalles sobre cómo deseas que funcione la aplicación y qué características específicas te gustaría incluir?

Cliente: Por supuesto. En primer lugar, necesitamos una función para registrar todos los productos en nuestro inventario, incluyendo detalles como nombre, descripción y precio de venta. Además, sería genial si pudiéramos adjuntar imágenes a cada producto para facilitar su identificación.

Desarrollador: Entendido. ¿Hay alguna otra información relevante que te gustaría almacenar para cada producto en el inventario?

Entrevistador: Sí, necesitaríamos un sistema que actualice automáticamente el inventario cuando se realicen compras o ventas.

Desarrollador: Entendido. ¿Hay alguna función específica que desees para la gestión de proveedores y compras?

Cliente: Sí, necesitaríamos una función que registre los proveedores, incluyendo información de contacto. Sería útil poder asociar productos específicos a cada proveedor y llevar un historial de compras realizadas a cada uno.

Desarrollador: Entendido. ¿Y en cuanto a la gestión de ventas y clientes? ¿Qué características esperas tener?

Cliente: Para la gestión de ventas, necesitaríamos una función para registrar cada transacción, incluyendo detalles del cliente, productos vendidos, cantidades y precios.

Desarrollador: Muy bien. Entonces, en resumen, necesitaríamos una aplicación móvil que permita registrar y actualizar el inventario de productos, gestionar proveedores y compras, y registrar ventas y clientes. Además, generación de informes.

Cliente: Exactamente. ¿Qué tipo de seguridad utilizará la aplicación?

Desarrollador: Implementaremos medidas de seguridad adicionales para proteger las consultas y transacciones en la aplicación. Utilizaremos tokens de autenticación para validar la identidad de los usuarios y garantizar que solo las personas autorizadas puedan acceder a ciertas funcionalidades. Antes de procesar cualquier consulta, verificaremos la validez del token.

Cliente: Eso me tranquiliza y por último ¿Cuál será el proceso de desarrollo y cuánto tiempo tomará?

Desarrollador: Trabajaremos en estrecha colaboración durante el proceso de desarrollo. El tiempo exacto dependerá del alcance y las funcionalidades específicas que desees incluir en la aplicación. Sin embargo, podemos establecer un cronograma realista y mantenerte informado sobre el progreso en cada etapa del desarrollo.

Cliente: Perfecto. Estoy emocionado por ver cómo esta aplicación ayudará mi negocio.

## 1.1.4. Funcionalidades

Gestión de inventario. El sistema debe ser capaz de mantener un registro detallado del inventario de la ferretería, incluyendo información como el nombre del producto, la descripción, el precio, la cantidad disponible, estado y el proveedor.

Búsqueda y filtrado. Permitir al usuario buscar productos específicos.

Catálogo de productos. Mostrar un catálogo completo de productos disponibles.

Procesamiento de productos. El sistema debe permitir al administrador procesarlos productos. Debe ser capaz de modificar los productos, ingresa productos, puede buscar productos dentro del sistema y habilitar o deshabilitar productos.

Control de ventas. Permitir al usuario registrar las ventas y ajustar cantidades.

Control de clientes. Permitir al usuario gestionar a sus clientes. Debe ser capaz de ingresar nuevo cliente, modificar y eliminar cliente.

Control de proveedores. Permitir al usuario gestionar sus proveedores. Debe ser capaz de ingresar nuevo proveedor, modificar y eliminar proveedor.

Historial de compras y de ventas: Permitir al usuario llevar un historial completo de sus compras al proveedor y sus ventas a los clientes.

Seguridad. El sistema debe contar con medidas de seguridad robustas para proteger la información confidencial de la ferretería, incluyendo datos de clientes y de ventas.

## 1.1.5 Presupuesto

Desarrollo de la aplicación:

Análisis y diseño de la aplicación: $250.00

Desarrollo de la funcionalidad de control de inventario en tiempo real: $625.00

Desarrollo de la interfaz de usuario y experiencia de usuario: $365.00

Desarrollo de backend y API:: $250.00

Desarrollo de frontend y aplicación móvil: $500.00

Integración de seguridad: $250.00

Pruebas y aseguramiento de calidad: $250.00

Total, para el desarrollo de la aplicación: $2500.00

Mantenimiento y soporte:

Mantenimiento mensual: $62.5/mes

Soporte técnico: $6.25 /hora

Costo anual estimado para el mantenimiento y soporte: $865.00

Costos adicionales:

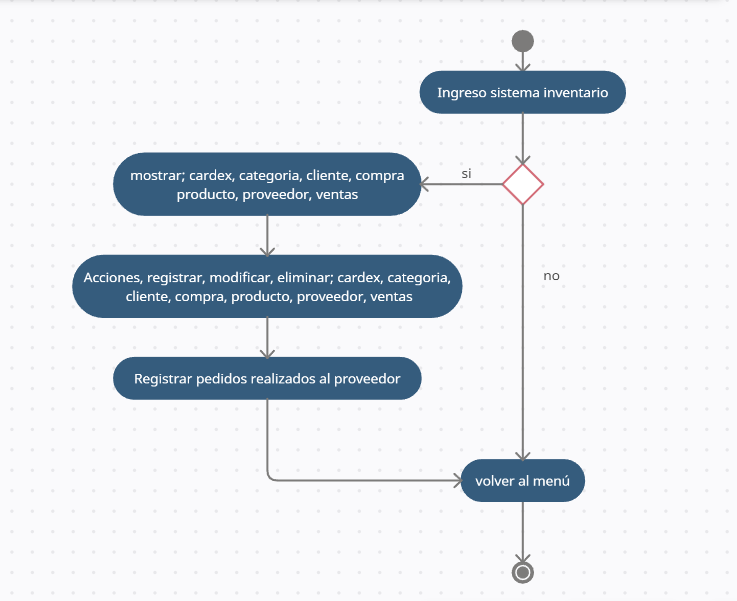
Hosting y almacenamiento en la nube: $62.5/año

Registro de dominio: $25.00/año

Total de costos adicionales: $87.50 /año

## 1.2. Diagramas de flujo

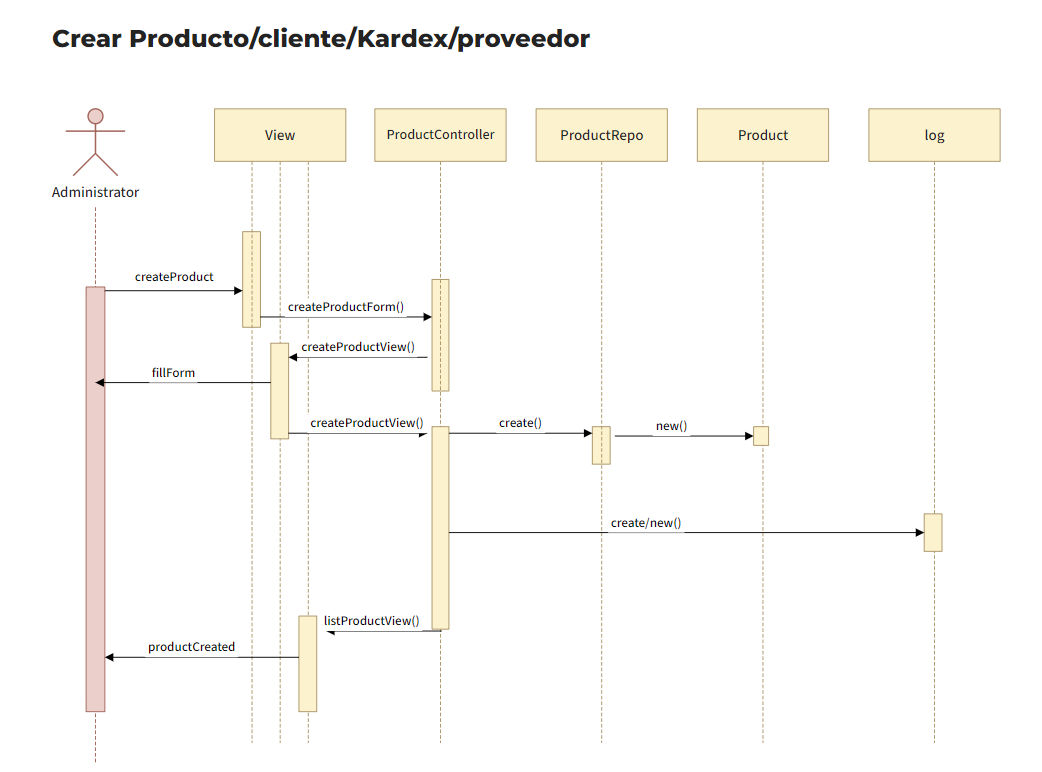
### 1.2.1. Diagramación de actividades

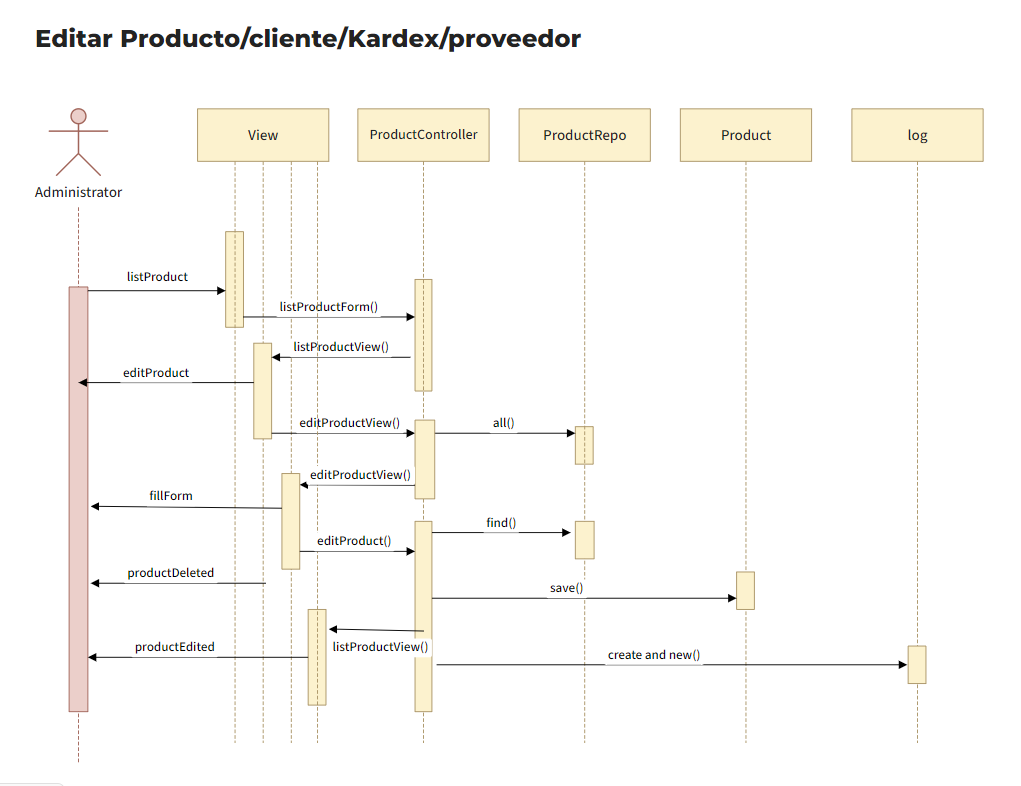


## 1.3. Casos de uso

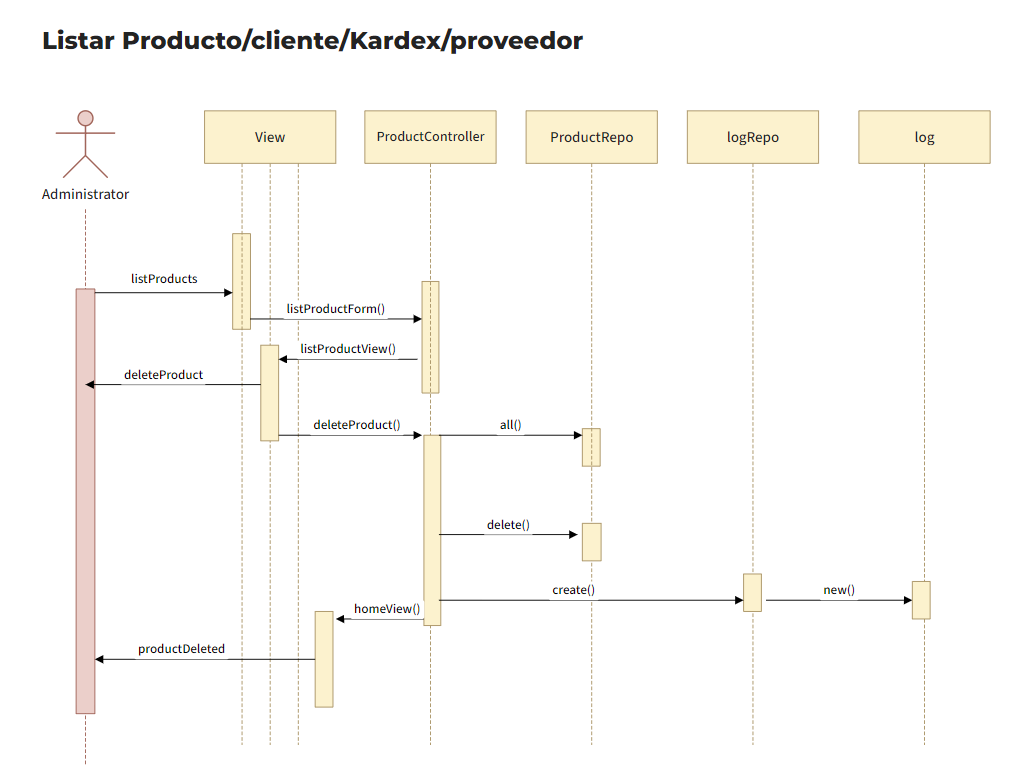
### 1.3.1. Diagramas de secuencias

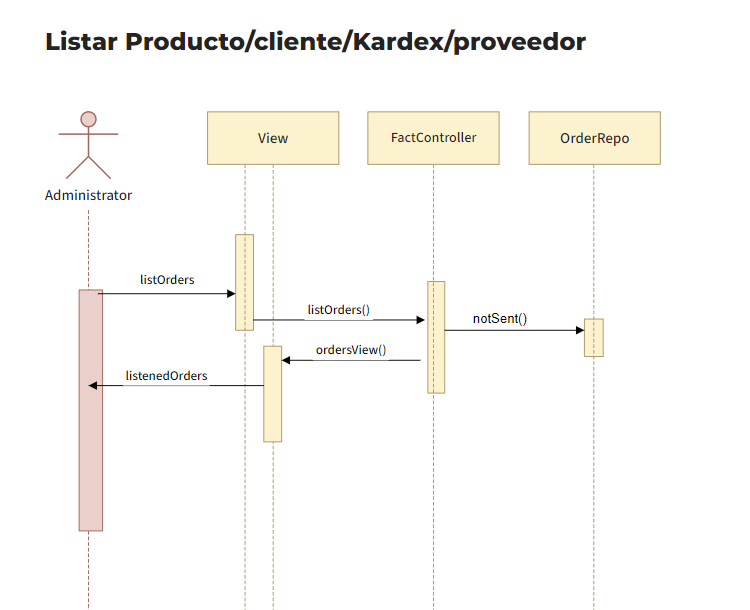
Crear





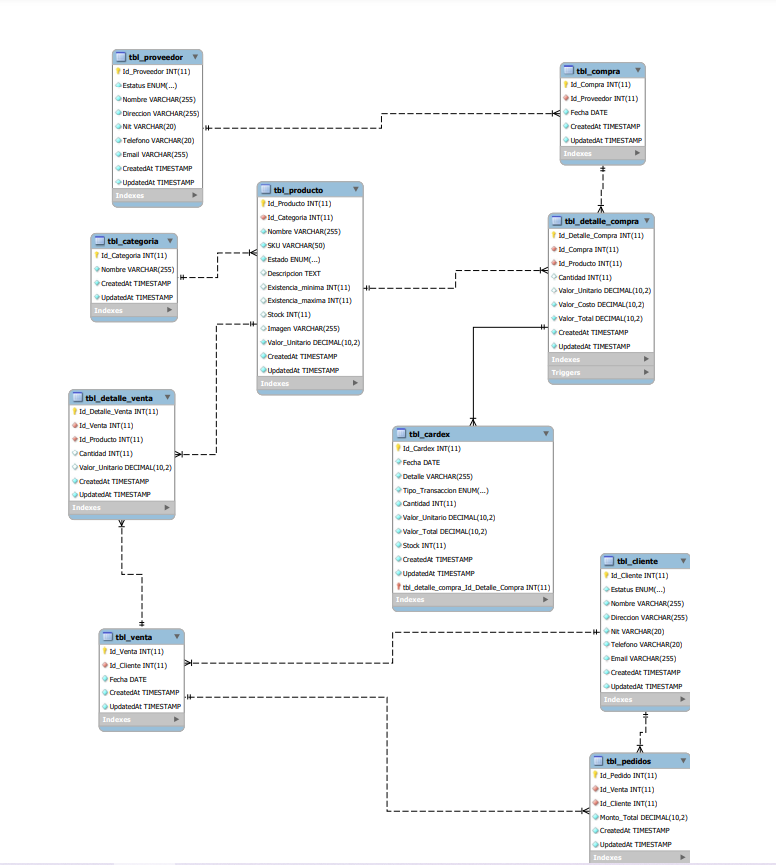
ELIMINAR PRODUCTO/CLIENTE/KARDEX/PROVEEDOR





## 1.4. Base de datos

### 1.4.1. Modelo entidad - relación



## 1.5. Base de datos (SQL)

### 1.5.5. MySQL script

CREATE DATABASE inventory\_control;

USE inventory\_control;

-- Tabla: tbl\_proveedor

CREATE TABLE tbl\_proveedor (

Id\_Proveedor INT PRIMARY KEY AUTO\_INCREMENT,

Estatus ENUM('activo', 'inactivo') NOT NULL,

Nombre VARCHAR(255) NOT NULL,

Direccion VARCHAR(255) NOT NULL,

Nit VARCHAR(20) NOT NULL,

Telefono VARCHAR(20) NOT NULL,

Email VARCHAR(255) NOT NULL,

CreatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UpdatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

-- Tabla: tbl\_categoria

CREATE TABLE tbl\_categoria (

Id\_Categoria INT PRIMARY KEY AUTO\_INCREMENT,

Nombre VARCHAR(255) NOT NULL,

CreatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UpdatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

-- Tabla: tbl\_producto

CREATE TABLE tbl\_producto (

Id\_Producto INT PRIMARY KEY,

Id\_Categoria INT NOT NULL,

Nombre VARCHAR(255) NOT NULL,

SKU VARCHAR(50) NOT NULL,

Estado ENUM('activo', 'inactivo') NOT NULL,

Descripcion TEXT,

Existencia\_minima INT,

Existencia\_maxima INT,

Stock INT,

Imagen VARCHAR(255),

Valor\_Unitario DECIMAL(10, 2) NOT NULL,

CreatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UpdatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (Id\_Categoria) REFERENCES tbl\_categoria (Id\_Categoria)

);

-- Tabla: tbl\_cliente

CREATE TABLE tbl\_cliente (

Id\_Cliente INT PRIMARY KEY AUTO\_INCREMENT,

Estatus ENUM('activo', 'inactivo') NOT NULL,

Nombre VARCHAR(255) NOT NULL,

Direccion VARCHAR(255) NOT NULL,

Nit VARCHAR(20) NOT NULL,

Telefono VARCHAR(20) NOT NULL,

Email VARCHAR(255) NOT NULL,

CreatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UpdatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

-- Tabla: tbl\_compra

CREATE TABLE tbl\_compra (

Id\_Compra INT PRIMARY KEY AUTO\_INCREMENT,

Id\_Proveedor INT NOT NULL,

Fecha DATE NOT NULL,

CreatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UpdatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (Id\_Proveedor) REFERENCES tbl\_proveedor (Id\_Proveedor)

);

-- Tabla: tbl\_detalle\_compra

CREATE TABLE tbl\_detalle\_compra (

Id\_Detalle\_Compra INT PRIMARY KEY AUTO\_INCREMENT,

Id\_Compra INT NOT NULL,

Id\_Producto INT NOT NULL,

Cantidad INT DEFAULT NULL,

Valor\_Unitario DECIMAL(10, 2),

Valor\_Costo DECIMAL(10, 2) NOT NULL,

Valor\_Total DECIMAL(10, 2) NOT NULL,

CreatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UpdatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (Id\_Compra) REFERENCES tbl\_compra (Id\_Compra),

FOREIGN KEY (Id\_Producto) REFERENCES tbl\_producto (Id\_Producto)

);

-- Tabla: tbl\_venta

CREATE TABLE tbl\_venta (

Id\_Venta INT PRIMARY KEY AUTO\_INCREMENT,

Id\_Cliente INT NOT NULL,

Fecha DATE NOT NULL,

CreatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UpdatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (Id\_Cliente) REFERENCES tbl\_cliente (Id\_Cliente)

);

-- Tabla: tbl\_detalle\_venta

CREATE TABLE tbl\_detalle\_venta (

Id\_Detalle\_Venta INT PRIMARY KEY AUTO\_INCREMENT,

Id\_Venta INT NOT NULL,

Id\_Producto INT NOT NULL,

Cantidad INT,

Valor\_Unitario DECIMAL(10, 2),

CreatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UpdatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (Id\_Venta) REFERENCES tbl\_venta (Id\_Venta),

FOREIGN KEY (Id\_Producto) REFERENCES tbl\_producto (Id\_Producto)

);

-- Tabla: tbl\_cardex

CREATE TABLE tbl\_cardex (

Id\_Cardex INT PRIMARY KEY AUTO\_INCREMENT,

Fecha DATE NOT NULL,

Detalle VARCHAR(255) NOT NULL,

Tipo\_Transaccion ENUM('compra', 'venta') NOT NULL,

Cantidad INT NOT NULL,

Valor\_Unitario DECIMAL(10, 2) NOT NULL,

Valor\_Total DECIMAL(10, 2) NOT NULL,

Stock INT NOT NULL,

CreatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UpdatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

DELIMITER $$

CREATE TRIGGER trigger\_compra AFTER INSERT ON tbl\_detalle\_compra FOR EACH ROW

BEGIN

-- Variables para almacenar los valores necesarios

DECLARE cantidad INT;

DECLARE tipo\_transaccion VARCHAR(10);

DECLARE nombre\_producto VARCHAR(255);

-- Obtener la cantidad y el nombre del producto

SELECT NEW.Cantidad, p.Nombre INTO cantidad, nombre\_producto

FROM tbl\_detalle\_compra dc

INNER JOIN tbl\_producto p ON dc.Id\_Producto = p.Id\_Producto

WHERE dc.Id\_Detalle\_Compra = NEW.Id\_Detalle\_Compra;

-- Establecer el tipo de transacción como 'compra'

SET tipo\_transaccion = 'compra';

-- Actualizar el stock en la tabla tbl\_producto

UPDATE tbl\_producto

SET Stock = Stock + cantidad

WHERE Id\_Producto = NEW.Id\_Producto;

-- Insertar registro en tbl\_cardex

INSERT INTO tbl\_cardex (Fecha, Detalle, Tipo\_Transaccion, Cantidad, Valor\_Unitario, Valor\_Total, Stock)

VALUES (CURDATE(), nombre\_producto, tipo\_transaccion, cantidad, NEW.Valor\_Unitario, cantidad \* NEW.Valor\_Unitario, (SELECT Stock FROM tbl\_producto WHERE Id\_Producto = NEW.Id\_Producto));

END$$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER trigger\_venta AFTER INSERT ON tbl\_detalle\_venta

FOR EACH ROW

BEGIN

-- Variables para almacenar los valores necesarios

DECLARE cantidad INT;

DECLARE tipo\_transaccion VARCHAR(10);

-- Obtener la cantidad y el tipo de transacción

SELECT NEW.Cantidad, 'venta' INTO cantidad, tipo\_transaccion

FROM tbl\_detalle\_venta

WHERE Id\_Detalle\_Venta = NEW.Id\_Detalle\_Venta;

-- Actualizar el stock en la tabla tbl\_producto

UPDATE tbl\_producto

SET Stock = Stock - cantidad

WHERE Id\_Producto = NEW.Id\_Producto;

-- Insertar registro en tbl\_cardex

INSERT INTO tbl\_cardex (Fecha, Detalle, Tipo\_Transaccion, Cantidad, Valor\_Unitario, Valor\_Total, Stock)

VALUES (CURDATE(), (SELECT Nombre FROM tbl\_producto WHERE Id\_Producto = NEW.Id\_Producto), tipo\_transaccion, cantidad, NEW.Valor\_Unitario, cantidad \* NEW.Valor\_Unitario, (SELECT Stock FROM tbl\_producto WHERE Id\_Producto = NEW.Id\_Producto));

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE sp\_insertar\_proveedor(

IN p\_Estatus ENUM('activo', 'inactivo'),

IN p\_Nombre VARCHAR(255),

IN p\_Direccion VARCHAR(255),

IN p\_Nit VARCHAR(20),

IN p\_Telefono VARCHAR(20),

IN p\_Email VARCHAR(255)

)

BEGIN

DECLARE record\_count INT;

SET record\_count = (SELECT COUNT(\*) FROM tbl\_proveedor WHERE Nombre = p\_Nombre);

IF record\_count > 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Error: El proveedor ya está registrado.';

ELSE

INSERT INTO tbl\_proveedor (Estatus, Nombre, Direccion, Nit, Telefono, Email, CreatedAt)

VALUES (p\_Estatus, p\_Nombre, p\_Direccion, p\_Nit, p\_Telefono, p\_Email, CURDATE());

END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE sp\_insertar\_categoria(

IN p\_Nombre VARCHAR(255)

)

BEGIN

DECLARE record\_count INT;

SET record\_count = (SELECT COUNT(\*) FROM tbl\_categoria WHERE Nombre = p\_Nombre);

IF record\_count > 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Error: La categoría ya está registrada.';

ELSE

INSERT INTO tbl\_categoria (Nombre, CreatedAt)

VALUES (p\_Nombre, CURDATE());

END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE sp\_insertar\_cliente(

IN p\_Estatus ENUM('activo', 'inactivo'),

IN p\_Nombre VARCHAR(255),

IN p\_Direccion VARCHAR(255),

IN p\_Nit VARCHAR(20),

IN p\_Telefono VARCHAR(20),

IN p\_Email VARCHAR(255)

)

BEGIN

DECLARE record\_count INT;

SET record\_count = (SELECT COUNT(\*) FROM tbl\_cliente WHERE Nombre = p\_Nombre);

IF record\_count > 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Error: El cliente ya está registrado.';

ELSE

INSERT INTO tbl\_cliente (Estatus, Nombre, Direccion, Nit, Telefono, Email, CreatedAt)

VALUES (p\_Estatus, p\_Nombre, p\_Direccion, p\_Nit, p\_Telefono, p\_Email, CURDATE());

END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE sp\_insertar\_compra(

IN p\_Id\_Proveedor INT,

IN p\_Fecha DATE

)

BEGIN

DECLARE record\_count INT;

SET record\_count = (SELECT COUNT(\*) FROM tbl\_compra WHERE Id\_Proveedor = p\_Id\_Proveedor AND Fecha = p\_Fecha);

IF record\_count > 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Error: La compra ya está registrada.';

ELSE

INSERT INTO tbl\_compra (Id\_Proveedor, Fecha, CreatedAt)

VALUES (p\_Id\_Proveedor, p\_Fecha, CURRENT\_TIMESTAMP);

END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE sp\_insertar\_detalle\_compra(

IN p\_Id\_Compra INT,

IN p\_Id\_Producto INT,

IN p\_Cantidad INT,

IN p\_Valor\_Unitario DECIMAL(10, 2),

IN p\_Valor\_Costo DECIMAL(10, 2),

IN p\_Valor\_Total DECIMAL(10, 2)

)

BEGIN

INSERT INTO tbl\_detalle\_compra (Id\_Compra, Id\_Producto, Cantidad, Valor\_Unitario, Valor\_Costo, Valor\_Total, CreatedAt)

VALUES (p\_Id\_Compra, p\_Id\_Producto, p\_Cantidad, p\_Valor\_Unitario, p\_Valor\_Costo, p\_Valor\_Total, CURRENT\_TIMESTAMP);

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE sp\_insertar\_venta(

IN p\_Id\_Cliente INT,

IN p\_Fecha DATE

)

BEGIN

DECLARE record\_count INT;

SET record\_count = (SELECT COUNT(\*) FROM tbl\_venta WHERE Id\_Cliente = p\_Id\_Cliente AND Fecha = p\_Fecha);

IF record\_count > 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Error: La venta ya está registrada.';

ELSE

INSERT INTO tbl\_venta (Id\_Cliente, Fecha, CreatedAt)

VALUES (p\_Id\_Cliente, p\_Fecha, CURRENT\_TIMESTAMP);

END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE sp\_insertar\_detalle\_venta(

IN p\_Id\_Venta INT,

IN p\_Id\_Producto INT,

IN p\_Cantidad INT,

IN p\_Valor\_Unitario DECIMAL(10, 2)

)

BEGIN

INSERT INTO tbl\_detalle\_venta (Id\_Venta, Id\_Producto, Cantidad, Valor\_Unitario, CreatedAt)

VALUES (p\_Id\_Venta, p\_Id\_Producto, p\_Cantidad, p\_Valor\_Unitario, CURRENT\_TIMESTAMP);

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE sp\_insertar\_producto(

IN p\_Id\_Producto INT,

IN p\_Id\_Categoria INT,

IN p\_Nombre VARCHAR(255),

IN p\_SKU VARCHAR(50),

IN p\_Estado ENUM('activo', 'inactivo'),

IN p\_Descripcion TEXT,

IN p\_Existencia\_Minima INT,

IN p\_Existencia\_Maxima INT,

IN p\_Stock INT,

IN p\_Imagen VARCHAR(255),

IN p\_Valor\_Unitario DECIMAL(10, 2)

)

BEGIN

DECLARE product\_count INT;

-- Verificar si el producto ya existe

SELECT COUNT(\*) INTO product\_count FROM tbl\_producto WHERE Id\_Producto = p\_Id\_Producto;

IF product\_count > 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'El producto ya existe.';

ELSE

-- Insertar el producto

INSERT INTO tbl\_producto (

Id\_Producto,

Id\_Categoria,

Nombre,

SKU,

Estado,

Descripcion,

Existencia\_Minima,

Existencia\_Maxima,

Stock,

Imagen,

Valor\_Unitario

)

VALUES (

p\_Id\_Producto,

p\_Id\_Categoria,

p\_Nombre,

p\_SKU,

p\_Estado,

p\_Descripcion,

p\_Existencia\_Minima,

p\_Existencia\_Maxima,

p\_Stock,

p\_Imagen,

p\_Valor\_Unitario

);

SELECT 'Producto insertado correctamente.' AS Message;

END IF;

END $$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE sp\_editar\_proveedor(

IN p\_Id\_Proveedor INT,

IN p\_Estatus ENUM('activo', 'inactivo'),

IN p\_Nombre VARCHAR(255),

IN p\_Direccion VARCHAR(255),

IN p\_Nit VARCHAR(20),

IN p\_Telefono VARCHAR(20),

IN p\_Email VARCHAR(255)

)

BEGIN

DECLARE v\_Count INT;

-- Verificar si el proveedor existe

SELECT COUNT(\*) INTO v\_Count

FROM tbl\_proveedor

WHERE Id\_Proveedor = p\_Id\_Proveedor;

-- Si el proveedor no existe, mostrar mensaje de error

IF v\_Count = 0 THEN

SELECT 'El proveedor no existe' AS Error;

ELSE

-- Actualizar los datos del proveedor

UPDATE tbl\_proveedor

SET Estatus = p\_Estatus,

Nombre = p\_Nombre,

Direccion = p\_Direccion,

Nit = p\_Nit,

Telefono = p\_Telefono,

Email = p\_Email,

UpdatedAt = CURRENT\_TIMESTAMP

WHERE Id\_Proveedor = p\_Id\_Proveedor;

SELECT 'Proveedor actualizado correctamente' AS Result;

END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE sp\_editar\_categoria(

IN p\_Id\_Categoria INT,

IN p\_Nombre VARCHAR(255)

)

BEGIN

DECLARE v\_Count INT;

-- Verificar si la categoría existe

SELECT COUNT(\*) INTO v\_Count

FROM tbl\_categoria

WHERE Id\_Categoria = p\_Id\_Categoria;

-- Si la categoría no existe, mostrar mensaje de error

IF v\_Count = 0 THEN

SELECT 'La categoría no existe' AS Error;

ELSE

-- Actualizar el nombre de la categoría

UPDATE tbl\_categoria

SET Nombre = p\_Nombre,

UpdatedAt = CURRENT\_TIMESTAMP

WHERE Id\_Categoria = p\_Id\_Categoria;

SELECT 'Categoría actualizada correctamente' AS Result;

END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE sp\_editar\_producto(

IN p\_Id\_Producto INT,

IN p\_Id\_Categoria INT,

IN p\_Nombre VARCHAR(255),

IN p\_SKU VARCHAR(50),

IN p\_Estado ENUM('activo', 'inactivo'),

IN p\_Descripcion TEXT,

IN p\_Existencia\_Minima INT,

IN p\_Existencia\_Maxima INT,

IN p\_Stock INT,

IN p\_Imagen VARCHAR(255),

IN p\_Valor\_Unitario DECIMAL(10, 2)

)

BEGIN

DECLARE v\_Count INT;

-- Verificar si el producto existe

SELECT COUNT(\*) INTO v\_Count

FROM tbl\_producto

WHERE Id\_Producto = p\_Id\_Producto;

-- Si el producto no existe, mostrar mensaje de error

IF v\_Count = 0 THEN

SELECT 'El producto no existe' AS Error;

ELSE

-- Verificar si la categoría existe

SELECT COUNT(\*) INTO v\_Count

FROM tbl\_categoria

WHERE Id\_Categoria = p\_Id\_Categoria;

-- Si la categoría no existe, mostrar mensaje de error

IF v\_Count = 0 THEN

SELECT 'La categoría no existe' AS Error;

ELSE

-- Actualizar los datos del producto

UPDATE tbl\_producto

SET Id\_Categoria = p\_Id\_Categoria,

Nombre = p\_Nombre,

SKU = p\_SKU,

Estado = p\_Estado,

Descripcion = p\_Descripcion,

Existencia\_minima = p\_Existencia\_Minima,

Existencia\_maxima = p\_Existencia\_Maxima,

Stock = p\_Stock,

Imagen = p\_Imagen,

Valor\_Unitario = p\_Valor\_Unitario,

UpdatedAt = CURRENT\_TIMESTAMP

WHERE Id\_Producto = p\_Id\_Producto;

SELECT 'Producto actualizado correctamente' AS Result;

END IF;

END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE sp\_editar\_cliente(

IN p\_Id\_Cliente INT,

IN p\_Estatus ENUM('activo', 'inactivo'),

IN p\_Nombre VARCHAR(255),

IN p\_Direccion VARCHAR(255),

IN p\_Nit VARCHAR(20),

IN p\_Telefono VARCHAR(20),

IN p\_Email VARCHAR(255)

)

BEGIN

DECLARE v\_Count INT;

-- Verificar si el cliente existe

SELECT COUNT(\*) INTO v\_Count

FROM tbl\_cliente

WHERE Id\_Cliente = p\_Id\_Cliente;

-- Si el cliente no existe, mostrar mensaje de error

IF v\_Count = 0 THEN

SELECT 'El cliente no existe' AS Error;

ELSE

-- Actualizar los datos del cliente

UPDATE tbl\_cliente

SET Estatus = p\_Estatus,

Nombre = p\_Nombre,

Direccion = p\_Direccion,

Nit = p\_Nit,

Telefono = p\_Telefono,

Email = p\_Email,

UpdatedAt = CURRENT\_TIMESTAMP

WHERE Id\_Cliente = p\_Id\_Cliente;

SELECT 'Cliente actualizado correctamente' AS Result;

END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE sp\_actualizar\_proveedor\_inactivo(IN p\_Id\_Proveedor INT)

BEGIN

IF EXISTS(SELECT \* FROM tbl\_proveedor WHERE Id\_Proveedor = p\_Id\_Proveedor AND Estatus = 'activo') THEN

UPDATE tbl\_proveedor SET Estatus = 'inactivo' WHERE Id\_Proveedor = p\_Id\_Proveedor;

SELECT 'Proveedor actualizado a inactivo correctamente.' AS Message;

ELSE

SELECT 'El proveedor no existe o ya está inactivo.' AS Message;

END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE sp\_actualizar\_cliente\_inactivo(IN p\_Id\_Cliente INT)

BEGIN

IF EXISTS(SELECT \* FROM tbl\_cliente WHERE Id\_Cliente = p\_Id\_Cliente AND Estatus = 'activo') THEN

UPDATE tbl\_cliente SET Estatus = 'inactivo' WHERE Id\_Cliente = p\_Id\_Cliente;

SELECT 'Cliente actualizado a inactivo correctamente.' AS Message;

ELSE

SELECT 'El cliente no existe o ya está inactivo.' AS Message;

END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE sp\_actualizar\_producto\_inactivo(IN p\_Id\_Producto INT)

BEGIN

DECLARE v\_Stock INT;

SELECT Stock INTO v\_Stock FROM tbl\_producto WHERE Id\_Producto = p\_Id\_Producto;

IF EXISTS(SELECT \* FROM tbl\_producto WHERE Id\_Producto = p\_Id\_Producto AND Estado = 'activo') THEN

IF v\_Stock > 0 THEN

SELECT 'No se puede actualizar a inactivo. El producto tiene stock disponible.' AS Message;

ELSE

UPDATE tbl\_producto SET Estado = 'inactivo' WHERE Id\_Producto = p\_Id\_Producto;

SELECT 'Producto actualizado a inactivo correctamente.' AS Message;

END IF;

ELSE

SELECT 'El producto no existe o ya está inactivo.' AS Message;

END IF;

END$$

DELIMITER ;

## 1.6 División del proyecto

## Proyect Manager

* George Rodríguez

## Frontend – aplicación

* Coordinador: Alejandro Girón
* Integrante: Dionicio Véliz
* Integrante: Jorge Alejandro
* Integrante: Velvet González

## Backend – API

* Coordinador: Abner Canté
* Integrante: Luis Fernando Cardona

## Base de datos

* Coordinador: Kevin Rodas
* Integrante: Melissa Álvarez

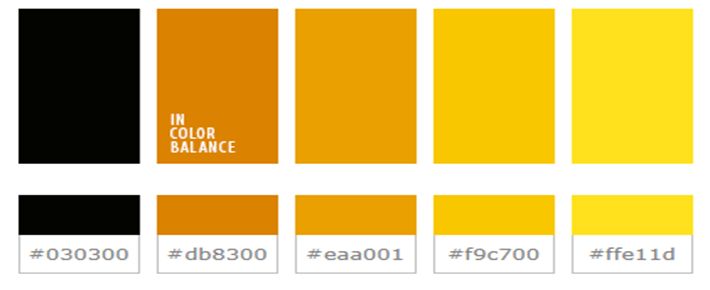
## Servidores

* Coordinador: Oscal Morales
* Integrante: Wallinton Eliu

## Documentación

* Coordinador: Kiara Orellana
* Integrante: Enso González

### 1.6.1. Paleta de colores



## Acerca de #030300

Principio del formulario



Final del formulario

|  |  |
| --- | --- |
| **HEX** | #030300 |
| **NOMBRE** | - |
| **TONO** | negro |
| **RGB** | 3 3 0 |
| **HSV** | 60° 100% 1% |
| **CMYK** | 0% 0% 100% 98% |
|  |  |

#030300 es negro . El componente de #030300 es RGB (3 3 0). El color complementario de #030300 es #000003. El negro es el color que recuerda una fuerte voluntad. El negro es efectivo para dirigir un diseño limpio y potente. Es el color para aclarar, pero también el color que recuerda una sensación de inquietud y opresión.

## Acerca de #DB8300

Principio del formulario



Final del formulario

|  |  |
| --- | --- |
| **HEX** | #DB8300 |
| **NOMBRE** | - |
| **TONO** | naranja fuerte |
| **RGB** | 219 131 0 |
| **HSV** | 35° 100% 85% |
|  |  |
| **CMYK** | 0% 40% 100% 14% |
|  |  |

#DB8300 es naranja fuerte. El componente de #DB8300 es RGB (219 131 0). El color complementario de #DB8300 es #0058DB. El naranja es el color de la vitalidad y la felicidad. Su diseño puede provocar el efecto de motivar las compras. Es un color equilibrado, en el que no existe la presión que tiene, por ejemplo, el rojo.

## Acerca de #EAA001

Principio del formulario



Final del formulario

|  |  |
| --- | --- |
| **HEX** | #EAA001 |
| **NOMBRE** | - |
| **TONO** | naranja vívido |
| **RGB** | 234 160 1 |
| **HSV** | 40° 99% 91% |
| **CMYK** | 0% 31% 99% 8% |

#EAA001 es naranja vívido. El componente de #EAA001 es RGB (234 160 1). El color complementario de #EAA001 es #014BEA. El naranja es el color de la vitalidad y la felicidad. Su diseño puede provocar el efecto de motivar las compras. Es un color equilibrado, en el que no existe la presión que tiene, por ejemplo, el rojo.

## Acerca de #F9C700

Principio del formulario



Final del formulario

|  |  |
| --- | --- |
| **HEX** | #F9C700 |
| **NOMBRE** | - |
| **TONO** | naranja vívido |
| **RGB** | 249 199 0 |
| **HSV** | 47° 100% 97% |
| **CMYK** | 0% 20% 100% 2% |
|  |  |

#F9C700 es naranja vívido. El componente de #F9C700 es RGB (249 199 0). El color complementario de #F9C700 es #0031F9. El naranja es el color de la vitalidad y la felicidad. Su diseño puede provocar el efecto de motivar las compras. Es un color equilibrado, en el que no existe la presión que tiene, por ejemplo, el rojo.

## Acerca de #FFE11D

Principio del formulario



Final del formulario

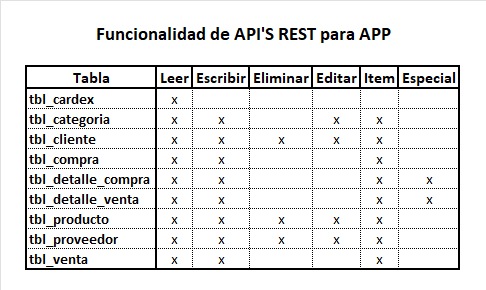
|  |  |
| --- | --- |
| **HEX** | #FFE11D |
| **NOMBRE** | - |
| **TONO** | amarillo brillante |
| **RGB** | 255 225 29 |
| **HSV** | 52° 88% 100% |
| **CMYK** | 0% 11% 88% 0% |

#FFE11D es amarillo brillante. El componente de #FFE11D es RGB (255 225 29). El color complementario de #FFE11D es #1D3AFF. El amarillo es un estímulo fuerte Es el color de la esperanza y las sonrisas que recuerdan al sol. Debido a que es un color con energía, usándolo como acentuación le da al diseño diversión y vitalidad.

## 1.6.2. Logotipo



# 2 Backend API REST



## 2.1 Lenguaje: PHP

### 2.1 .1 Configuración de las rutas de API

/{$folder}/cardex

/{$folder}/categorias

/{$folder}/clientes

/{$folder}/compras

/{$folder}/detalleCompras

/{$folder}/detalleVentas

/{$folder}/productos

/{$folder}/proveedores

/{$folder}/ventas

/api/cardex: “Ruta del cardex”

/api/categorías: “Ruta de Categorías”

/api/clientes: “Ruta de clientes”

/api/compras: “Ruta de compras”

/api/detalleCompras: “Ruta de detalle de compra”

/api/detalleventas: “Ruta de detalle de ventas”

/api/productos “Ruta de productos”

/api/proveedores “Ruta de proveedores”

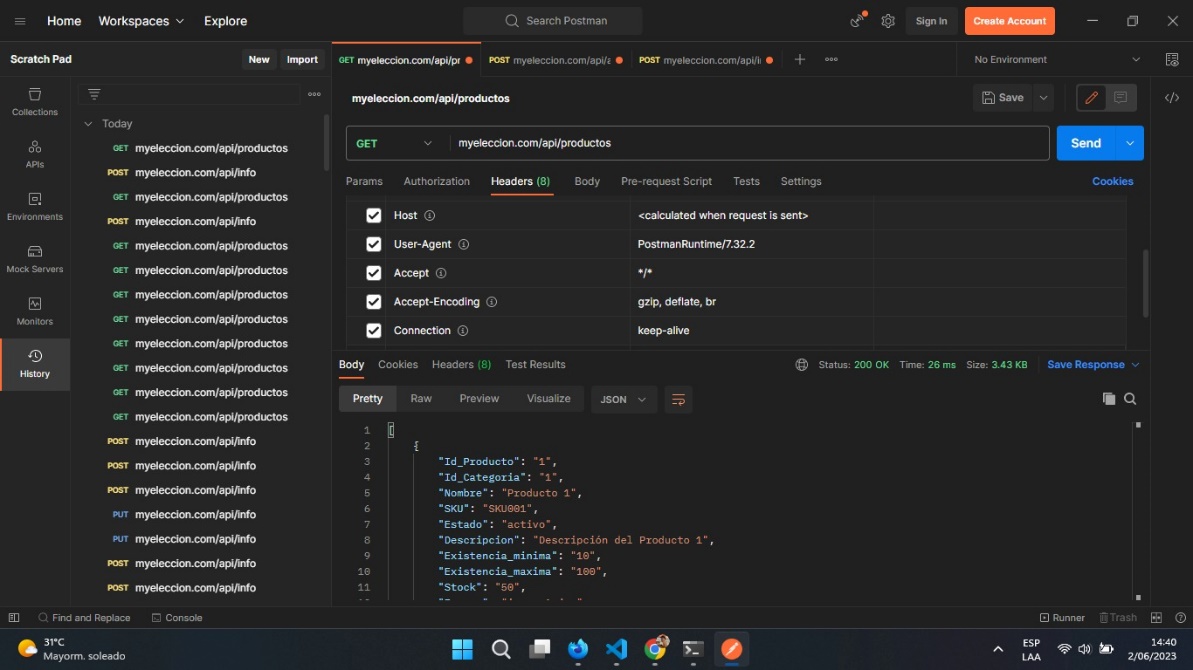
/api/ventas “Ruta de ventas”

/api/usuarios “Ruta de usuarios”

/api/auth “Ruta de autenticación”

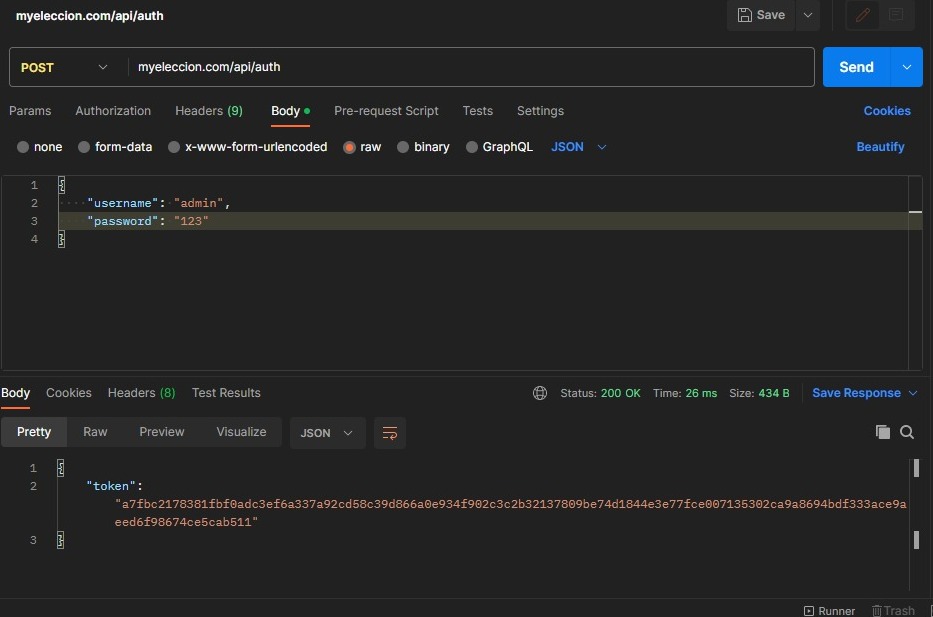
/api/info “Ruta de información”

### 2.1.2 Pruebas de desarrollo por Postman

Validación de Token Correctamente

### 2.1.3. Implementación de seguridad por tokens

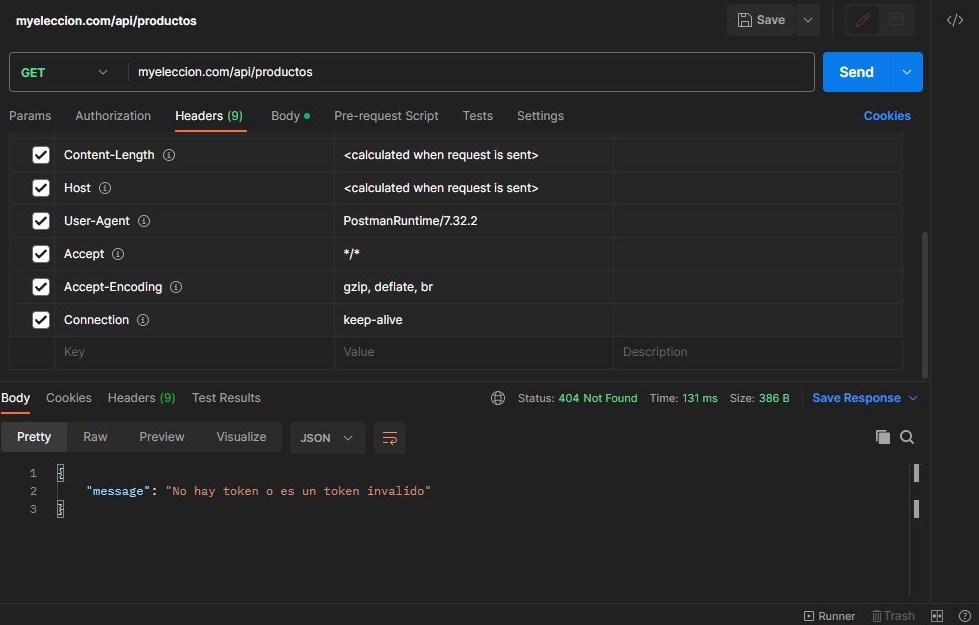
Por cada login el sistema genera un token nuevo.



T

Token: 447c5b8f076e280962491fd457143614ce10a6072354218a2d3d0ae6950db8fdc90f91aa56ceac7bc058796219840ce63fde7860ea5f6a1d1348016079daebc1

Sino se envia el token correctamente sucede lo siguiente:



## 2.2 Frontend

### 2.2.3 Login

1.8.5. Esquema de permisos

1.9. Presentación y Conversión a APK

1.9.4. Utilización de claves publicas

ssh debian@181.189.131.142 -p 727

Equipo de Servidores

IP pública 181.189.131.142

Equipo 1 - Gateway API

IP privada 192.168.28.50

Puerto SSH 750

Puerto https 80 público 80

Servicios Apach

Equipo 2 - Security

IP privada 192.168.28.51

Puerto SSH 751

Puerto DB 3306 público 851

Puerto https 80 público 8081

Servicios Apache, MySQL

IP privada 192.168.28.52

Puerto SSH 752

Puerto DB 3306 público 852

Puerto https 80 público 8082

Servicios Apache, MySQL

* Gateway
* Apache
* php
* security
* apache
* mysql
* php

Tercero

* apache
* mysql
* php
* data sql

.52

api app

.52 var/www(\*)

Se crearon usuarios para las base de datos

CREATE USER 'data'@'localhost' IDENTIFIED BY 'drF12345fa@';

GRANT ALL PRIVILEGES ON \* . \* TO 'data'@'localhost';

Se cargo el archivo APP\_structure.sql, APP\_data.sql, APP\_trigger.sql y APP\_procedure.sql

security .51

api auth

.net

auth.sql

CREATE USER 'seguridad'@'localhost' IDENTIFIED BY 'mdsF12345@';

GRANT ALL PRIVILEGES ON \* . \* TO 'seguridad'@'localhost';

Se cargo la base de datos auth.sql

Gateway

50 api gateway

1.9.6. Presentación de Aplicativo