# TFG

# OPTISTOCK


Curso: Desarrollo de Aplicaciones Multiplataforma

Alumno: Rafael Auxilia Esquina

Profesor: Javier Martín

# 1. Motivación

El proyecto **OptiStock** surge de una problemática real en las pequeñas y medianas empresas (PYMES), donde la falta de herramientas tecnológicas accesibles y eficientes genera importantes desafíos en la gestión de inventarios y ventas. Estos problemas no solo afectan la operatividad diaria, sino que también limitan la capacidad de crecimiento y competitividad de las empresas.

**Problemas Identificados**

1. **Falta de control en el inventario:**
    - Las empresas suelen depender de métodos manuales como hojas de cálculo o registros en papel, lo que incrementa los errores humanos y dificulta el seguimiento del stock.
    - No contar con un control en tiempo real puede llevar a desabastecimientos o exceso de inventario, lo que afecta directamente las ventas y los costos operativos.
2. **Procesos manuales ineficientes:**
    - La generación manual de facturas, tickets y reportes consume tiempo valioso que podría destinarse a actividades estratégicas.
    - Los procesos manuales no permiten una integración fluida entre las ventas físicas y online.
3. **Dificultades en la toma de decisiones:**
    - Sin acceso a datos estructurados y análisis, los gerentes no pueden tomar decisiones informadas sobre compras, promociones o estrategias de ventas.
    - La falta de herramientas de análisis limita la capacidad de identificar tendencias de mercado o productos de alta rotación.
4. **Falta de integración tecnológica:**
    - Muchas PYMES no cuentan con una plataforma centralizada que unifique la gestión de productos, inventarios y ventas.
    - La falta de integración entre los diferentes procesos operativos genera redundancias y errores.
5. **Experiencia del cliente deficiente:**
    - Los clientes esperan procesos de compra rápidos, tanto en tienda física como online.
    - Una interfaz poco intuitiva o la falta de seguimiento de pedidos puede afectar la fidelización de los clientes.

## Motivación del Proyecto

OptiStock se plantea como una solución integral que aborde estas problemáticas mediante el desarrollo de un sistema tecnológico que permita:

- **Centralizar la gestión** de productos, inventarios y ventas en una única plataforma.
- **Facilitar el seguimiento de stock** en tiempo real, con alertas automáticas para niveles bajos.
- **Optimizar procesos** de venta física y online, reduciendo tiempos y errores.
- **Mejorar la experiencia del cliente** mediante interfaces intuitivas y funcionales.
- **Automatizar tareas repetitivas**, como la generación de facturas, tickets y reportes.
- **Proporcionar herramientas de análisis** que permitan a los gerentes tomar decisiones basadas en datos.

Con estas características, OptiStock busca empoderar a las PYMES para que puedan competir en un entorno cada vez más digitalizado.

## Motivation

**Project OptiStock** arises from a real problem faced by small and medium-sized enterprises (SMEs), where the lack of accessible and efficient technological tools creates significant challenges in inventory and sales management. These issues not only affect daily operations but also limit the capacity for growth and competitiveness of companies.

**Identified Problems**

1. **Lack of inventory control:**
    - Companies often rely on manual methods such as spreadsheets or paper records, which increases human errors and makes stock tracking difficult.
    - Not having real-time control can lead to stockouts or excess inventory, directly affecting sales and operational costs.
2. **Inefficient manual processes:**
    - The manual generation of invoices, tickets, and reports consumes valuable time that could be spent on strategic activities.
    - Manual processes do not allow for a smooth integration between physical and online sales.

3. **Difficulties in decision-making:**
    - Without access to structured data and analysis, managers cannot make informed decisions about purchases, promotions, or sales strategies.
    - The lack of analytical tools limits the ability to identify market trends or high-turnover products.
4. **Lack of technological integration:**
    - Many SMEs do not have a centralized platform that unifies the management of products, inventories, and sales.
    - The lack of integration between different operational processes generates redundancies and errors.
5. **Poor customer experience:**
    - Customers expect fast purchasing processes, both in physical stores and online.
    - An unintuitive interface or lack of order tracking can affect customer loyalty.

**Project Motivation**

OptiStock is proposed as a comprehensive solution that addresses these issues by developing a technological system that allows:

- Centralizing the management of products, inventories, and sales on a single platform.
- Facilitating real-time stock tracking, with automatic alerts for low levels.
- Optimizing physical and online sales processes, reducing time and errors.
- Improving customer experience through intuitive and functional interfaces.
- Automating repetitive tasks, such as generating invoices, tickets, and reports.
- Providing analytical tools that enable managers to make data-driven decisions.

# 2. Objetivos Propuestos

El proyecto OptiStock tiene como objetivo principal desarrollar un sistema completo y robusto que permita gestionar inventarios y ventas de manera eficiente. Este objetivo general se desglosa en objetivos específicos que abarcan diferentes aspectos del sistema.

## Objetivo General

Diseñar e implementar un sistema integral de gestión de inventarios y ventas, con soporte para aplicaciones de escritorio y web, que sea escalable, accesible y adaptable a las necesidades de pequeñas y medianas empresas.

## Objetivos Específicos

### Gestión de Productos

- Crear un módulo de catálogo que permita:
    - Registrar nuevos productos con atributos como nombre, descripción, precio, IVA, descuento, stock, categoría, marca y modelo.
    - Editar y eliminar productos existentes.
    - Clasificar productos por categorías para facilitar su búsqueda y organización.
    - Asignar precios y promociones de manera sencilla.

### Gestión de Inventarios

- Implementar un sistema de control de stock que permita:
    - Registrar ubicaciones físicas de los productos dentro de la empresa.
    - Realizar inventarios físicos periódicos para verificar el stock real.
    - Registrar movimientos de productos (entradas y salidas) con detalles como tipo de movimiento, cantidad y fecha.
    - Generar alertas automáticas cuando los niveles de stock sean bajos.

**Ventas**

- Diseñar un módulo de ventas que incluya:
    - Gestión de ventas físicas mediante un punto de venta (POS).
    - Generación automática de tickets y facturas en formato PDF.
    - Registro de un historial de ventas para análisis posterior.
    - Gestión de devoluciones de productos de manera eficiente.

**E-commerce**

- Desarrollar una tienda online que permita:
    - A los clientes navegar por el catálogo de productos de manera intuitiva.
    - Realizar compras online con un proceso de checkout simplificado.
    - Hacer seguimiento de pedidos en tiempo real.
    - Consultar el historial de compras realizadas.

**Gestión de Recursos Humanos**

- Crear un módulo para la administración de empleados que permita:
    - Registrar datos personales y laborales de los empleados (DNI, nombre completo, dirección, teléfono, salario, etc.).
    - Generar nóminas mensuales con desglose de salario bruto, deducciones y salario neto.
    - Gestionar turnos de trabajo y roles dentro de la empresa.

**Órdenes de Compra**

- Implementar un sistema para gestionar relaciones con proveedores, incluyendo:
    - Registro de órdenes de compra.
    - Seguimiento del estado de las órdenes (pendiente, completada, cancelada).
    - Recepción de mercancías y actualización automática del stock.

**Reportes y Análisis**

- Diseñar un módulo de análisis que permita:
    - Generar informes personalizados sobre ventas, inventarios y desempeño de productos.
    - Exportar reportes en formatos PDF y Excel.
    - Visualizar gráficos y estadísticas en tiempo real.

**Impacto Esperado**

Con la implementación de estos objetivos, OptiStock permitirá a las PYMES:

- Reducir errores operativos.
- Mejorar la eficiencia en la gestión de inventarios y ventas.
- Incrementar la satisfacción del cliente.
- Tomar decisiones empresariales más informadas.

# 3. Metodología Utilizada

Para garantizar el éxito del proyecto, se adoptó la metodología ágil **SCRUM**, que se caracteriza por ser iterativa e incremental. SCRUM permite dividir el desarrollo en ciclos cortos llamados *sprints*, lo que facilita la adaptación a cambios y la entrega continua de valor.

**Ventajas de SCRUM**

- Permite una comunicación constante entre los miembros del equipo.
- Facilita la priorización de tareas según las necesidades del cliente.
- Promueve la mejora continua mediante la retroalimentación al final de cada sprint.

**Roles en SCRUM**

1. **Product Owner** (Rafael Auxilia Esquina)**:**
    - Representa al cliente y define los requisitos del sistema.
    - Prioriza las tareas en el backlog del producto.
2. **Scrum Master** (Rafael Auxilia Esquina)**:**
    - Asegura el cumplimiento de la metodología SCRUM.
    - Elimina impedimentos que puedan afectar al equipo de desarrollo.
3. **Equipo de Desarrollo** (Rafael Auxilia Esquina)**:**
    - Implementa las funcionalidades del sistema.
    - Participa activamente en las reuniones diarias y en la planificación de sprints.

# 4. Backlog

**1. Gestión de Clientes**

- Como administrador, quiero registrar y gestionar información de clientes para poder ofrecer un mejor servicio y seguimiento. Como cliente, quiero poder ver mi historial de compras para tener un registro de mis transacciones.

**2. Gestión de Proveedores**

- Como administrador, quiero registrar proveedores y sus datos de contacto para facilitar la comunicación y gestión de órdenes de compra. Como administrador, quiero recibir notificaciones cuando un proveedor actualice su información para mantener la base de datos actualizada.

**3. Gestión de Inventarios**

- Como administrador, quiero controlar el stock de productos para evitar sobreventas y asegurar disponibilidad. También quiero recibir alertas cuando el stock de un producto esté bajo para reabastecerlo a tiempo. También quiero recibir alertas cuando el stock de un producto esté bajo para reabastecerlo a tiempo. Como empleado, quiero registrar ubicaciones de productos en el almacén para facilitar la búsqueda.

**4. Ventas**

- Como administrador, quiero generar facturas automáticamente tras cada venta para simplificar el proceso. Como vendedor, quiero gestionar ventas en módulos físicos y online para ofrecer flexibilidad a los clientes y quiero ofrecer diferentes métodos de pago para facilitar la compra a los clientes. Como cliente, quiero poder realizar devoluciones de productos de manera sencilla y quiero recibir confirmaciones de mis pedidos por correo electrónico para tener un registro de mi compra.

**3. Promociones y Descuentos**

- Como administrador, quiero crear y gestionar promociones para incentivar las ventas durante temporadas específicas. Como cliente, quiero ver las promociones disponibles al navegar en la tienda online para aprovechar descuentos.

**4. Gestión de Devoluciones**

- Como vendedor, quiero registrar devoluciones de productos de manera sencilla para mantener el inventario actualizado. Como cliente, quiero iniciar una devolución en línea para facilitar el proceso sin tener que ir a la tienda.

**5. Informes Personalizados**

- Como administrador, quiero generar informes personalizados sobre ventas, inventarios y desempeño de productos para tomar decisiones informadas. Como gerente, quiero exportar informes a Excel y PDF para compartir con otros miembros del equipo.

**6. Seguridad y Accesos**

- Como administrador, quiero gestionar roles y permisos de usuarios para asegurar que solo las personas autorizadas tengan acceso a información sensible. Como usuario, quiero iniciar sesión de manera segura para proteger mis datos y transacciones.

## 5. Tecnologías y Herramientas Utilizadas

El desarrollo de OptiStock ha requerido la integración de múltiples tecnologías y herramientas que aseguran un sistema robusto, funcional y escalable. A continuación, se detallan las tecnologías utilizadas en cada componente del sistema.

**Lenguajes de Programación**

1. **C#:**
    - Utilizado para el desarrollo de la aplicación de escritorio.
    - Elegido por su compatibilidad con .NET Framework y su capacidad para construir aplicaciones robustas con interfaces gráficas avanzadas.

2. **JavaScript:**
    - Implementado en la lógica del cliente para la interfaz web.
    - Permite una experiencia de usuario interactiva y fluida.

3. **HTML/CSS:**
    - Lenguajes base para la estructura y diseño de la interfaz web.
    - CSS se complementó con Bootstrap para garantizar un diseño responsive.


4. **SQL:**
    - Lenguaje utilizado para la gestión y consulta de la base de datos relacional.


**Bases de Datos**

1. **MySQL:**
    - Sistema de gestión de bases de datos relacional elegido por su escalabilidad, seguridad y facilidad de uso.
    - Almacena toda la información del sistema, desde productos y ventas hasta movimientos de inventario y datos de empleados.

**Frameworks y Bibliotecas**

1. **WPF (Windows Presentation Foundation):**
    - Framework utilizado para desarrollar la interfaz gráfica de la aplicación de escritorio.
    - Permite diseñar interfaces modernas y personalizables.
2. **Bootstrap:**
    - Framework CSS que facilita el diseño responsive y consistente de la interfaz web.


3. **Express.js:**
    - Framework de Node.js utilizado para gestionar el backend de la aplicación web.
    - Ofrece una arquitectura sólida para manejar peticiones HTTP y APIs REST.


4. **iTextSharp:**
    - Biblioteca utilizada para la generación automática de documentos PDF, como tickets y facturas.
5. **EPPlus:**
    - Biblioteca para la creación de archivos Excel, utilizada en la generación de reportes de ventas e inventarios.

**Herramientas de Desarrollo**

1. **Visual Studio 2022:**
    1. IDE principal para el desarrollo de la aplicación de escritorio en C#.


2. **Visual Studio Code:**
    1. Editor de código utilizado para el desarrollo de la interfaz web y el backend con Node.js.


3. **MySQL Workbench:**
    1. Herramienta para el diseño, modelado y gestión de la base de datos.


4. **Git:**
    1. Sistema de control de versiones que permite realizar un seguimiento de los cambios en el código fuente.

5. **GitHub:**
    1. Plataforma utilizada para alojar el repositorio del proyecto y facilitar la colaboración entre los miembros del equipo.


## 6. Estimación de Recursos y Planificación

La planificación del proyecto OptiStock fue esencial para asegurar el cumplimiento de los plazos y la asignación eficiente de recursos. Se utilizó un **diagrama de Gantt** para organizar las tareas y estimar su duración.

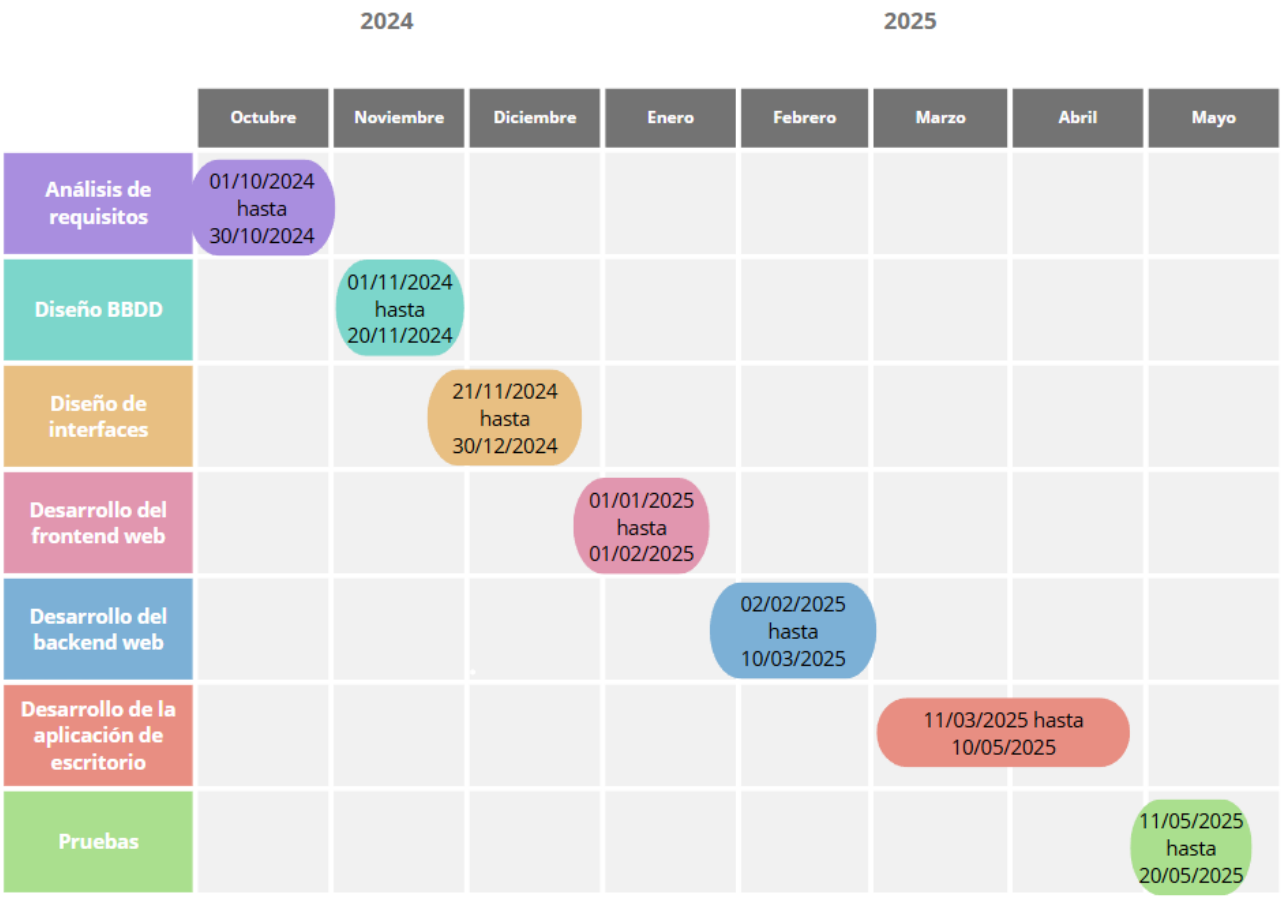
**Recursos Humanos**

1. **Desarrollador Full Stack** (Rafael Auxilia Esquina)**:**
    - Responsable del desarrollo del backend, frontend web y la aplicación de escritorio.
2. **Diseñador UI/UX** (Rafael Auxilia Esquina)**:**
    - Encargado de diseñar las interfaces de usuario y garantizar una experiencia intuitiva.
3. **Tester** (Rafael Auxilia Esquina)**:**
    - Responsable de realizar pruebas funcionales y de usabilidad para garantizar la calidad del sistema.

**Recursos Técnicos**

- Equipos de desarrollo (ordenadores con software especializado).
- Servidores para el alojamiento web y la base de datos.

# 7. Diagrama de Gantt



## 8. Fases del proyecto

1. Análisis de Requisitos

    - Identificación de necesidades del usuario.
    - Recopilación de requisitos funcionales y no funcionales para definir el alcance del proyecto.
    - Documentación de requisitos utilizando herramientas como Google Docs para una mejor organización.

2. Diseño de la Base de Datos (BBDD)

    - Creación del esquema de la base de datos utilizando MySQL Workbench para la modelización de datos.
    - Normalización de la estructura para evitar redundancias y mejorar la eficiencia.
    - Establecimiento de relaciones y restricciones en MySQL para garantizar la integridad de los datos.
    - Implementación inicial de la base de datos en un servidor de pruebas.

3. Diseño de Interfaces

    - Creación de wireframes y prototipos con Figma para visualizar la interfaz antes del desarrollo.
    - Diseño UI/UX centrado en la usabilidad y accesibilidad para garantizar una experiencia intuitiva.
    - Ajustes iterativos en el diseño basados en feedback de usuarios y pruebas de concepto.

4. Desarrollo del Frontend Web

    - Implementación en HTML, CSS y Bootstrap para la estructura y estilos de la web.
    - Uso de JavaScript para interactividad y una mejor experiencia de usuario. o Integración con la API REST del backend mediante fetch para la comunicación con la base de datos.
    - Optimización del diseño responsivo para compatibilidad con dispositivos móviles.

5. Desarrollo del Backend Web

    - Creación de la API con Express.js en Node.js para gestionar peticiones y respuestas del sistema.
    - Implementación de endpoints REST para operaciones CRUD sobre productos y usuarios.
    - Conexión con la base de datos MySQL para almacenamiento y recuperación de datos.

6. Desarrollo de la Aplicación de Escritorio

    - Implementación en C# con WPF para el diseño de la interfaz gráfica de la aplicación de escritorio.
    - Uso del patrón MVVM para una mejor separación entre la lógica de negocio y la interfaz.
    - Conexión con la base de datos MySQL a través de Entity Framework. o Integración con funcionalidades como ventas y gestión de inventarios.

7. Pruebas

    - Pruebas de integración para garantizar la comunicación efectiva entre el frontend y el backend.
    - Corrección de errores y optimización del rendimiento.

## 9. Enlace de Mockup

https://www.figma.com/design/y1R3mYap8RsWD6kYrvwqhC/Untitled?node-id=0
1&p=f&t=FiriJF1SoVSJclSs-0

## 10. Arquitectura general del sistema

El sistema **OptiStock** ha sido diseñado siguiendo una arquitectura modular y escalable, que favorece la separación de responsabilidades y facilita tanto el mantenimiento como la evolución futura del software. Esta arquitectura permite que cada componente del sistema pueda ser desarrollado, probado y modificado de manera independiente, minimizando el impacto de los cambios y mejorando la robustez general de la aplicación.

**Descripción del diagrama:**

- La **Interfaz Gráfica** (WPF) comunica todas las acciones del usuario a la capa de lógica de negocio.
- La **Lógica de Negocio** procesa las peticiones, aplica reglas, valida datos y solicita o actualiza información en la base de datos.
- La **Base de Datos** almacena de forma persistente toda la información relevante del sistema (productos, ventas, clientes, inventarios, etc.).

### 10.1 Descripción de Capas

#### 10.1.1 Capa de Presentación

- **Tecnología:** Windows Presentation Foundation (WPF)
- **Responsabilidad:**
    - Proporcionar una interfaz gráfica amigable, moderna e intuitiva.
    - Gestionar la interacción con el usuario: formularios, menús, botones, cuadros de diálogo, listas, etc.
    - Enviar las acciones y datos introducidos por el usuario a la capa de lógica de negocio.
    - Mostrar los resultados, mensajes, errores o informes generados por el sistema.
- **Ventajas:**
    - Permite una experiencia de usuario rica y personalizable.
    - Facilita la integración de gráficos, tablas y exportación de datos.
    - El uso de XAML y data binding facilita la separación entre interfaz y lógica.

#### 10.1.2 Capa de Lógica de Negocio

- **Tecnología:** C# (clases, servicios, modelos)
- **Responsabilidad:**
    - Implementar la lógica central del sistema: gestión de productos, inventario, ventas, usuarios, etc.
    - Validar la información recibida desde la interfaz antes de interactuar con la base de datos.
    - Coordinar el flujo de datos entre la presentación y la persistencia.
    - Aplicar reglas de negocio, cálculos (por ejemplo, descuentos, IVA, stock mínimo), y gestionar los permisos de acceso según el rol del usuario.
- **Ejemplo de responsabilidades:**
    - Comprobar que el EAN de un producto sea válido antes de permitir su registro.
    - Calcular el precio final de un producto aplicando IVA y descuento.
    - Registrar movimientos de stock tras una venta o una recepción de mercancía.

- **Ventajas:**
    - Centraliza la lógica, permitiendo su reutilización y facilitando las pruebas unitarias.
    - Permite la evolución del sistema sin modificar la interfaz ni la base de datos.

#### 10.1.3 Capa de Acceso a Datos

- **Tecnología:** MySqlConnector (para C#)
- **Responsabilidad:**
    - Gestionar la conexión y comunicación con la base de datos MySQL.
    - Ejecutar consultas SQL, procedimientos almacenados, inserciones, actualizaciones y borrados.
    - Mapear los resultados de la base de datos a objetos del sistema (modelos C#).
- **Ejemplo de responsabilidades:**
    - Insertar un nuevo cliente en la tabla clientes.
    - Actualizar el stock de un producto tras una venta.
    - Consultar el historial de ventas para generar informes.
- **Ventajas:**
    - Permite centralizar y optimizar las operaciones de acceso a datos.
    - Facilita el cambio de motor de base de datos en el futuro si fuera necesario.
    - Mejora la seguridad al evitar inyecciones SQL mediante el uso de parámetros.

#### 10.1.4 Ventajas de la Arquitectura Modular

- **Facilidad de mantenimiento:** Cada módulo puede ser modificado de forma aislada.
- **Escalabilidad:** Permite añadir nuevas funcionalidades (por ejemplo, un módulo web o móvil) sin reescribir la aplicación.
- **Reutilización:** La lógica de negocio puede ser reutilizada por otras interfaces (por ejemplo, una API REST).
- **Seguridad:** La separación de capas permite implementar controles de acceso y validaciones en distintos niveles.
- **Pruebas:** Facilita la realización de pruebas unitarias y de integración.

### 10.1.5 Ejemplo de Flujo de Datos

1. **El usuario** pulsa el botón “Registrar venta” en la interfaz WPF.
2. **La capa de presentación** recoge los datos del formulario y los envía a la lógica de negocio.
3. **La lógica de negocio** valida los datos, calcula totales y solicita a la capa de datos registrar la venta.
4. **La capa de datos** inserta la venta y los detalles en las tablas correspondientes de MySQL.
5. **La lógica de negocio** recibe la confirmación y devuelve el resultado a la interfaz.
6. **La interfaz** muestra un mensaje de éxito y genera el ticket PDF para el cliente.

### 10.1.6 Posibles Mejoras Futuras

- Implementación de una **API REST** para permitir acceso desde aplicaciones móviles o web.
- Separación física de las capas en diferentes proyectos o microservicios.
- Uso de un ORM (como Entity Framework) para facilitar la gestión de entidades complejas.
- Incorporación de un sistema de caché para mejorar el rendimiento en consultas frecuentes.

# 11. Estructura y Modelo de la Base de Datos

La base de datos del sistema **OptiStock** constituye el núcleo de la gestión de la información de la aplicación. Está diseñada bajo un modelo relacional, normalizado y robusto, que permite la integridad referencial entre los diferentes módulos de la aplicación (productos, ventas, inventario, compras, clientes, trabajadores, etc.).

A continuación, se presenta un análisis detallado de las tablas principales, sus relaciones y su función dentro del sistema.

### 11.1 Descripción de las Tablas Principales

#### 11.1.1 Tabla categorías

- **Propósito:** Clasificar los productos para facilitar su búsqueda y organización.
- **Campos:**
    - Id: INT, clave primaria, autoincremental.
    - Nombre: VARCHAR(100), nombre único de la categoría.

**Ejemplo de uso:**
Permite agrupar productos bajo categorías como “Papelería”, “Hogar y Baño”, “Juguetes”, etc.

---

#### 11.1.2 Tabla productos

- **Propósito:** Almacenar toda la información relevante de cada producto gestionado en la empresa.
- **Campos:**
    - Id: INT, clave primaria, autoincremental.
    - Nombre: VARCHAR(255), nombre del producto.
    - Precio: FLOAT(10,2), precio base sin IVA.
    - Iva: FLOAT(5,2), porcentaje de IVA aplicado.
    - PrecioTotal: FLOAT(10,2), calculado automáticamente como Precio + (Precio * Iva / 100).
    - Descripcion: TEXT, descripción detallada.
    - CategoriaId: INT, clave foránea a categorias(Id).
    - Stock: INT, cantidad disponible.
    - StockInicial: INT, stock inicial al dar de alta el producto.
    - Imagen: VARCHAR(255), ruta o enlace de la imagen del producto.
    - FechaCreacion: DATETIME, fecha de alta del producto.

- Descuento: FLOAT(5,2), porcentaje de descuento aplicado.
- EAN: VARCHAR(13), código de barras único.
- Marca: VARCHAR(30), marca del producto.
- Modelo: VARCHAR(30), modelo del producto.
- CantidadMaxima, CantidadMinima: INT, para gestión avanzada de stock.

**Relaciones:**

- Cada producto pertenece a una sola categoría (CategoriaId).
- Cada producto puede estar en varias ubicaciones (inventario y Ubicaciones).

---

#### 11.1.3 Tabla Ubicaciones

- **Propósito:** Definir las ubicaciones físicas (estanterías, zonas, etc.) donde se almacenan los productos.
- **Campos:**
  - Id: INT, clave primaria.
  - ProductoId: INT, referencia a productos(Id).
  - Nombre: VARCHAR(20), identificador o código de la ubicación.

**Nota:** En versiones futuras se puede separar la relación Producto-Ubicación para permitir ubicaciones sin producto asignado.

---

#### 11.1.4 Tabla inventario

- **Propósito:** Registrar el stock real de cada producto en cada ubicación física.
- **Campos:**
  - Id: INT, clave primaria.
  - ProductoId: INT, referencia a productos(Id).
  - UbicacionId: INT, referencia a Ubicaciones(Id).
  - Cantidad: INT, cantidad registrada.
  - UltimoInventario: DATETIME, fecha del último inventario realizado.
  - Recuento: INT, recuento físico.
  - UbicacionRecuento: VARCHAR(50), descripción de la ubicación en el recuento.

---

#### 11.1.5 Tabla movimientos y movimientosWeb

- **Propósito:** Registrar todos los movimientos de stock (entradas, salidas, ventas, devoluciones, compras, etc.), tanto en tienda física como en la web.
- **Campos:**
    - Id: INT, clave primaria.
    - ProductoId/ProductoEAN: VARCHAR(20), referencia al producto (EAN).
    - FechaMovimiento: DATETIME, fecha y hora del movimiento.
    - TipoMovimiento: VARCHAR(50), tipo de movimiento (“Venta Física”, “Compra”, “Devolución”, etc.).
    - Cantidad: INT, cantidad afectada.

**Relaciones:**

- Referencia a productos por EAN, permitiendo trazabilidad absoluta.

---

#### 11.1.6 Tabla clientes y clienteweb

- **Propósito:** Almacenar los datos de los clientes físicos y online respectivamente.
- **Campos:**
    - Id: INT, clave primaria.
    - NombreCompleto, DNI, Telefono, Email, Direccion, Ciudad, etc.
    - FechaNacimiento, FechaAlta: Fechas para gestión de antigüedad y promociones.

---

#### 11.1.7 Tabla ventas y HistorialVentas

- **Propósito:** Registrar las ventas realizadas y sus detalles.
- **Campos en ventas:**
    - Id: INT, clave primaria.
    - DniCliente: VARCHAR(15), referencia al cliente.
    - NumeroDocumento: VARCHAR(20), número de ticket o factura.
    - Fecha: DATETIME.
    - Total: DECIMAL(10,2).
- **Campos en HistorialVentas:**
    - Id: INT, clave primaria.
    - NumeroDocumento: referencia a ventas(NumeroDocumento).

- Producto: referencia a productos(EAN).
- Cantidad: INT.

---

#### 11.1.8 Tabla Devoluciones y DetalleDevolucion

- **Propósito:** Registrar devoluciones de productos y sus detalles.
- **Campos en Devoluciones:**
  - Id: INT, clave primaria.
  - DniCliente, NumeroDocumento, Motivo, Fecha.
- **Campos en DetalleDevolucion:**
  - Id: INT, clave primaria.
  - DevolucionId: referencia a Devoluciones(Id).
  - ProductoId: referencia a productos(EAN).
  - Cantidad: INT.

---

#### 11.1.9 Tabla OrdenesDeCompra y DetallesOrdenDeCompra

- **Propósito:** Gestionar las compras a proveedores y los productos adquiridos.
- **Campos en OrdenesDeCompra:**
  - Id, NumeroOrden, Proveedor, Estado, FechaApertura.
- **Campos en DetallesOrdenDeCompra:**
  - Id, OrdenDeCompraId, ProductoId, EAN, Cantidad, PrecioUnitario.

---

#### 11.1.10 Tabla Proveedores

- **Propósito:** Almacenar información de proveedores.
- **Campos:**
  - Id, Nombre, Contacto, Telefono, Email, Direccion, Ciudad, Provincia, CodigoPostal, Pais, TipoProveedor, Notas, SitioWeb, FechaRegistro.

#### 11.1.11 Tabla Trabajadores, Nomina, Vacaciones

- **Propósito:** Gestión de recursos humanos, nóminas y vacaciones.
- **Campos en Trabajadores:**
    - Id, NombreCompleto, FechaNacimiento, DNI, Telefono, Email, Direccion, FechaContratacion, Salario, Usuario, Contraseña, NumeroSeguridadSocial, CategoriaProfesional, Departamento.
- **Campos en Nomina:**
    - Id, TrabajadorId, Mes, Anio, SalarioBruto, Deducciones, SalarioNeto.
- **Campos en Vacaciones:**
    - Id, TrabajadorId, FechaInicio, FechaFin, DiasTotales.

---

#### 11.1.12 Tablas de pedidos web: clienteweb, pedidos_web, detalle_pedido

- **Propósito:** Gestión de la tienda online, pedidos y detalles de compra.
- **Campos en clienteweb:**
    - Id, Nombre, Apellido, Direccion, Ciudad, Correo, Contraseña.
- **Campos en pedidos_web:**
    - Id, NumeroPedido, ClienteWebId, PrecioTotal, FechaPedido, Estado.
- **Campos en detalle_pedido:**
    - Id, PedidoId, ProductoId, Cantidad.

### 11.2 Ventajas del Modelo de Datos

- **Escalabilidad:** Permite añadir nuevas tablas o campos sin afectar la integridad del sistema.
- **Flexibilidad:** Soporta tanto ventas físicas como online, gestión de inventario avanzado y control de recursos humanos.
- **Seguridad:** El uso de claves foráneas y restricciones únicas evita inconsistencias y duplicidades.
- **Trazabilidad:** Toda acción relevante queda registrada en la base de datos, permitiendo auditorías y análisis históricos.

# 12.Prototipos de código y sus explicaciones

## 12.1. Inserción de un nuevo producto:

```
using (var conexion = new MySqlConnection(cadenaConexion))
{
    conexion.Open();
    string sql = @"INSERT INTO productos (Nombre, Precio, Iva, Descripcion, CategoriaId, Stock,
Imagen, Descuento, EAN, Marca, Modelo)
                  VALUES (@nombre, @precio, @iva, @descripcion, @categoriaId, @stock, @imagen,
@descuento, @ean, @marca, @modelo)";
    using (var cmd = new MySqlCommand(sql, conexion))
    {
        cmd.Parameters.AddWithValue("@nombre", producto.Nombre);
        cmd.Parameters.AddWithValue("@precio", producto.Precio);
        cmd.Parameters.AddWithValue("@iva", producto.Iva);
        cmd.Parameters.AddWithValue("@descripcion", producto.Descripcion);
        cmd.Parameters.AddWithValue("@categoriaId", producto.CategoriaId);
        cmd.Parameters.AddWithValue("@stock", producto.Stock);
        cmd.Parameters.AddWithValue("@imagen", producto.Imagen);
        cmd.Parameters.AddWithValue("@descuento", producto.Descuento);
        cmd.Parameters.AddWithValue("@ean", producto.EAN);
        cmd.Parameters.AddWithValue("@marca", producto.Marca);
        cmd.Parameters.AddWithValue("@modelo", producto.Modelo);
        cmd.ExecuteNonQuery();
    }
}
```

Este fragmento muestra cómo se inserta un producto nuevo desde la aplicación. Se utilizan parámetros para evitar inyecciones SQL y asegurar la integridad de los datos. Todos los campos relevantes del producto se envían a la base de datos.

12.2. Actualización del Stock tras una Venta

```
public void ActualizarStock(string ean, int cantidadVendida)
{
    using (var conexion = new MySqlConnection(cadenaConexion))
    {
        conexion.Open();
        string sql = "UPDATE productos SET Stock = Stock - @cantidadVendida WHERE EAN = @ean";
        using (var cmd = new MySqlCommand(sql, conexion))
        {
            cmd.Parameters.AddWithValue("@cantidadVendida", cantidadVendida);
            cmd.Parameters.AddWithValue("@ean", ean);
            cmd.ExecuteNonQuery();
        }
    }
}
```

Cuando se realiza una venta, el sistema descuenta automáticamente la cantidad vendida del stock del producto correspondiente. Esto mantiene el inventario actualizado en tiempo real.

12.3. Registrar un Movimiento de Stock

```
public void RegistrarMovimiento(string ean, string tipoMovimiento, int cantidad)
{
    using (var conexion = new MySqlConnection(cadenaConexion))
    {
        conexion.Open();
        string sql = @"INSERT INTO movimientos (ProductoId, TipoMovimiento, Cantidad)
                       VALUES (@ean, @tipoMovimiento, @cantidad)";
        using (var cmd = new MySqlCommand(sql, conexion))
        {
            cmd.Parameters.AddWithValue("@ean", ean);
            cmd.Parameters.AddWithValue("@tipoMovimiento", tipoMovimiento);
            cmd.Parameters.AddWithValue("@cantidad", cantidad);
            cmd.ExecuteNonQuery();
        }
    }
}
```

Cada vez que hay una entrada o salida de stock (venta, compra, devolución, ajuste de inventario), se registra un movimiento en la tabla movimientos, lo que permite la trazabilidad completa del producto.

12.4. Consulta del Historial de Ventas de un Cliente

```
SELECT v.NumeroDocumento, v.Fecha, v.Total, hv.Producto, hv.Cantidad
FROM ventas v                                    .
JOIN HistorialVentas hv ON v.NumeroDocumento = hv.NumeroDocumento
WHERE v.DniCliente = '12345678A'
ORDER BY v.Fecha DESC;
```

Esta consulta permite obtener todas las compras realizadas por un cliente, mostrando los productos adquiridos, cantidades, fecha y total de cada venta.

12.5. Inserción de una Orden de Compra y sus Detalles

```
// Insertar la orden de compra
string sqlOrden = @"INSERT INTO OrdenesDeCompra (NumeroOrden, Proveedor, Estado)
                    VALUES (@numeroOrden, @proveedor, @estado)";
using (var cmd = new MySqlCommand(sqlOrden, conexion))
{
    cmd.Parameters.AddWithValue("@numeroOrden", orden.NumeroOrden);
    cmd.Parameters.AddWithValue("@proveedor", orden.Proveedor);
    cmd.Parameters.AddWithValue("@estado", orden.Estado);
    cmd.ExecuteNonQuery();
    int idOrden = (int)cmd.LastInsertedId;

    // Insertar los detalles de la orden
    foreach (var detalle in orden.Detalles)
    {
        string sqlDetalle = @"INSERT INTO DetallesOrdenDeCompra (OrdenDeCompraId, ProductoId, EAN,
Cantidad, PrecioUnitario)
                              VALUES (@ordenId, @productoId, @ean, @cantidad, @precioUnitario)";
        using (var cmdDetalle = new MySqlCommand(sqlDetalle, conexion))
        {
            cmdDetalle.Parameters.AddWithValue("@ordenId", idOrden);
            cmdDetalle.Parameters.AddWithValue("@productoId", detalle.ProductoId);
            cmdDetalle.Parameters.AddWithValue("@ean", detalle.EAN);
            cmdDetalle.Parameters.AddWithValue("@cantidad", detalle.Cantidad);
            cmdDetalle.Parameters.AddWithValue("@precioUnitario", detalle.PrecioUnitario);
            cmdDetalle.ExecuteNonQuery();
        }
    }
}
```

Este código muestra cómo se inserta una orden de compra y sus detalles asociados, asegurando la integridad referencial entre ambas tablas.

12.6. Cambiar o aplicar Descuento a un Producto

```
public void ActualizarDescuentoProducto(int productoId, float nuevoDescuento)
{
    using (var conexion = new MySqlConnection(cadenaConexion))
    {
        conexion.Open();
        string sql = "UPDATE productos SET Descuento = @nuevoDescuento WHERE Id = @productoId";
        using (var cmd = new MySqlCommand(sql, conexion))
        {
            cmd.Parameters.AddWithValue("@nuevoDescuento", nuevoDescuento);
            cmd.Parameters.AddWithValue("@productoId", productoId);
            cmd.ExecuteNonQuery();
        }
    }
}
```

Permite modificar el descuento aplicado a un producto, lo que impactará en el precio final mostrado al cliente y en los informes de ventas.

12.7. Exportación de Datos a Excel

```
public void ActualizarDescuentoProducto(int productoId, float nuevoDescuento)
{
    using (var conexion = new MySqlConnection(cadenaConexion))
    {
        conexion.Open();
        string sql = "UPDATE productos SET Descuento = @nuevoDescuento WHERE Id = @productoId";
        using (var cmd = new MySqlCommand(sql, conexion))
        {
            cmd.Parameters.AddWithValue("@nuevoDescuento", nuevoDescuento);
            cmd.Parameters.AddWithValue("@productoId", productoId);
            cmd.ExecuteNonQuery();
        }
    }
}
```

En OptiStock, la exportación de listados (productos, inventario, ventas, etc.) a Excel facilita el análisis y la presentación de información para auditoría, gestión o informes.

12.8. Generación de Tickets o Facturas en PDF

```
using iTextSharp.text;
using iTextSharp.text.pdf;
using System.IO;

public void GenerarPDFTicket(string dniCliente, List<DVenta> productosEnVenta, double totalCompra,
DateTime fechaCompra, string nombreEmpresa, string ticketId)
{
    string rutaArchivo = $"Ticket_{ticketId}.pdf";
    Document document = new Document();
    PdfWriter.GetInstance(document, new FileStream(rutaArchivo, FileMode.Create));
    document.Open();

    // Encabezado
    document.Add(new Paragraph(nombreEmpresa, FontFactory.GetFont(FontFactory.HELVETICA_BOLD,
16)));
```

```
document.Add(new Paragraph($"Ticket: {ticketId}"));
document.Add(new Paragraph($"Fecha: {fechaCompra.ToShortDateString()}"));
document.Add(new Paragraph($"Cliente: {dniCliente}"));
document.Add(new Paragraph(" "));

// Tabla de productos
PdfPTable table = new PdfPTable(4);
table.AddCell("Producto");
table.AddCell("Cantidad");
table.AddCell("Precio Unitario");
table.AddCell("Subtotal");

foreach (var producto in productosEnVenta)
{
    table.AddCell(producto.Nombre);
    table.AddCell(producto.Cantidad.ToString());
    table.AddCell(producto.PrecioUnitario.ToString("C"));
    table.AddCell((producto.Cantidad * producto.PrecioUnitario).ToString("C"));
}
```

```
    document.Add(table);

    // Total
    document.Add(new Paragraph(" "));
    document.Add(new Paragraph($"TOTAL: {totalCompra.ToString("C")}",
FontFactory.GetFont(FontFactory.HELVETICA_BOLD, 14)));

    document.Close();
}
```

Al finalizar una venta, el sistema genera automáticamente un ticket o factura en PDF para el cliente, usando la librería iTextSharp.