**Tecnicatura:** Técnico Superior en Desarrollo Web y Aplicaciones Digitales.

**Espacios Curriculares: Proyecto Integrador**

**Tema:** PROYECTO: Planificador de Proyectos Empresariales

**Docente:** Veronica Nievas

**Integrantes**:

*Alexis Gabriel Baracat Peralta 40107091*

*Janet Pilar Molina DNI 35109986*

*Cristian Felipe Pasquevich DNI 40807865*

*Natalia Ana Martorina DNI 34542698*

*Juan Daniel Opazo DNI 35747302*

**Año: 2025**

**Proyecto:**

# STOCKPOINT

***Desarrollado by MACHINA DEVELOPS***

Un sistema diseñado para la gestión de stock para pequeñas empresas o Pymes facilitando y automatizando tareas relacionadas con el stock.



[**FICHA DEL DOCUMENTO 3**](#_30j0zll)

[**CONTENIDO 4**](#_1fob9te)

[**1**](#_2jxsxqh) **INTRODUCCIÓN 6**

[**1.1**](#_2et92p0) **Propósito 6**

[**1.2**](#_tyjcwt) **Alcance 6**

**1.3 Herramientas 6**

[**1.4**](#_z337ya) **Personal involucrado 6**

[**1.5**](#_3j2qqm3) **Definiciones, acrónimos y abreviaturas 7**

[**1.6**](#_1y810tw) **Referencias 7**

[**1.7**](#_17dp8vu) **Resumen 7**

[**2**](#_26in1rg) **DESCRIPCIÓN GENERAL 7**

[**2.1**](#_4i7ojhp) **Perspectiva del producto 7**

[**2.3**](#_2xcytpi) **Características de los usuarios 8**

[**2.4**](#_1ci93xb) **Restricciones 9**

[**3**](#_3whwml4) **REQUISITOS ESPECÍFICOS 9**

[**3.2**](#_2bn6wsx) **Product Backlog 17**

[3.2.1](#_qsh70q) Product Backlog **17**

[**3.3**](#_3as4poj) **Sprints 19**

[3.3.1](#_1pxezwc) Sprint 1 **19**

[3.3.2](#_49x2ik5) Sprint 2 **20**



**PROYECTO INTEGRADOR**

***-Especificaciones requisitos de software-***

**1 Introducción**

Este documento es una Especificación de Requisitos Software (ERS) para el Sistema de información para la gestión de procesos y control de inventarios. Esta especificación se ha estructurado basándose en las directrices dadas por el estándar IEEE Práctica Recomendada para Especificaciones de Requisitos Software ANSI/IEEE 830, 1998.

**1.1 Propósito**

El presente documento tiene como propósito definir las especificaciones funcionales, para el desarrollo de un sistema de información web que permitirá **Control de Stock para pequeños comercios y PYMEs**, destinado a propietarios y empleados de negocios como almacenes, kioscos, depósitos y talleres. El sistema permitirá registrar, consultar y administrar el inventario de forma sencilla y eficiente.

**1.2 Alcance**

El sistema proporcionará:

* Registro y actualización de productos.
* Control de entradas y salidas de stock.
* Alertas de stock bajo.
* Generación de reportes.
* Gestión básica de usuarios.

Será accesible desde un navegador web, con interfaz amigable y adaptable a

distintos niveles de conocimiento técnico

**1.4 Personal involucrado**

| Nombre | Juan Opazo |
| --- | --- |
| **Rol** | Product Owner |
| **Categoría Profesional** | Estudiante |
| **Responsabilidad** |  |
| **Información de contacto** | juanse.361@gmail.com |

| Nombre | Alexis Baracat |
| --- | --- |
| **Rol** | Scrum master |
| **Categoría Profesional** | Estudiante |
| **Responsabilidad** |  |
| **Información de contacto** | mail |

| Nombre | Natalia Martorina |
| --- | --- |
| **Rol** | Developers |
| **Categoría Profesional** | Estudiante |
| **Responsabilidad** |  |
| **Información de contacto** | mail |

| Nombre | Pilar Molina |
| --- | --- |
| **Rol** | Developers |
| **Categoría Profesional** | Estudiante |
| **Responsabilidad** |  |
| **Información de contacto** | mail |

**1.5 Definiciones, acrónimos y abreviaturas**

| ***Nombre*** | ***Descripción*** |
| --- | --- |
| **Usuario** | Administrador, Empleado |
| **SIS-I** | Sistema de Control de Stock para pequeños comercios y PYMEs. |
| **ERS** | Especificación de Requisitos Software |
| **RF** | Requerimiento Funcional |
| **RNF** | Requerimiento No Funcional |
| **FTP** | Protocolo de Transferencia de Archivos |
| **Moodle** | Aula Virtual |



**1.6Referencias**

| **Título del Documento** | **Referencia** |
| --- | --- |
| Standard IEEE 830 - 1998 | IEEE |

**1.7 Resumen**

Este documento describe la Especificación de Requisitos de Software para el sistema **Stock Point**, una aplicación web destinada a pequeños comercios y PYMEs para la gestión de inventarios.

El sistema permitirá registrar productos, controlar entradas y salidas de stock, emitir alertas de bajo inventario y generar reportes.

El proyecto seguirá la metodología ágil Scrum, comenzando con un Sprint 0 para definir alcance, diseño y backlog inicial. El backend se desarrollará en Python, la base de datos será MySQL y el frontend se implementará con HTML, CSS y JavaScript.

Con esta solución se busca optimizar la gestión de inventarios, reducir pérdidas por faltantes o excesos de mercadería y facilitar la toma de decisiones.

**2 Descripción general**

**2.1 Perspectiva del producto**

El sistema **Stock Point** será un producto diseñado para operar en entornos web, accesibles desde cualquier dispositivo con conexión a internet y navegador moderno, sin necesidad de que el cliente requiera un software adicional.

Permitirá a los usuarios realizar operaciones de alta, baja y modificación de productos, registrar movimientos de stock y generar reportes, todo desde una interfaz simple y adaptada a distintos niveles de experiencia técnica.

El sistema está integrado a pequeños comercios y PyMes, contemplando distintos roles de uso, con diferentes permisos, pensado como una solución segura, capaz de crecer en cantidad de usuarios y productos.



**2.2 Características de los usuarios**

| **Tipo de usuario** | Administrador |
| --- | --- |
| **Formación** | Manejo de PC |
| **Actividades** | Alta/baja de productos, gestión de usuarios,análisis de reportes,control de stock |

| **Tipo de usuario** | Empleado |
| --- | --- |
| **Formación** | Manejo de PC |
| **Actividades** | Registro de ventas, control de entradas y salidas de productos, actualización de inventario |

**2.3 Restricciones**

* El sistema funcionará únicamente en navegadores compatibles (Chrome, Firefox, Edge).
* Se requiere conexión a internet para su uso.
* Compatibilidad con pantallas de escritorio y tablet.
* Backend desarrollado en **Python** con frameworks como Flask o FastAPI
* Base de datos: **MySQL**.
* Frontend con **HTML5, CSS3 y JavaScript**.

**3 Requisitos específicos**

REQUERIMIENTOS FUNCIONALES

**RF-01:** Registrar nuevos productos con nombre, código SKU, precio y stock mínimo.

**RF-02:** Modificar y eliminar productos existentes.

**RF-03:** Registrar movimientos de stock (entradas, salidas, ajustes).

**RF-04:** Consultar stock actual y filtrar por nombre, código o categoría.

**RF-05:** Generar alertas de bajo stock.

**RF-06:** Exportar reportes de stock a PDF o Excel.

**RF-07:** Gestionar usuarios con roles y permisos.

REQUERIMIENTOS NO FUNCIONALES

**RNF-01:** Interfaz amigable y responsiva.

**RNF-02:** Tiempo de respuesta máximo de 2 segundos por operación.

**RNF-03:** Acceso seguro con autenticación de usuario y contraseñas encriptadas.

**RNF-04:** Backup automático diario de la base de datos.

**RNF-05:** Disponibilidad del sistema mínima del 99% mensual

**Product Backlog**

**US01:** Como administrador, quiero registrar productos para tenerlos en el inventario.

**US02:** Como administrador, quiero registrar entradas y salidas de productos para mantener actualizado el stock.

**US03:** Como administrador, quiero recibir alertas cuando un producto esté bajo el stock mínimo.

**US04:** Como administrador, quiero generar reportes para analizar el estado del inventario.

**US05:** Como empleado, quiero poder registrar ventas para que el stock se descuente automáticamente.

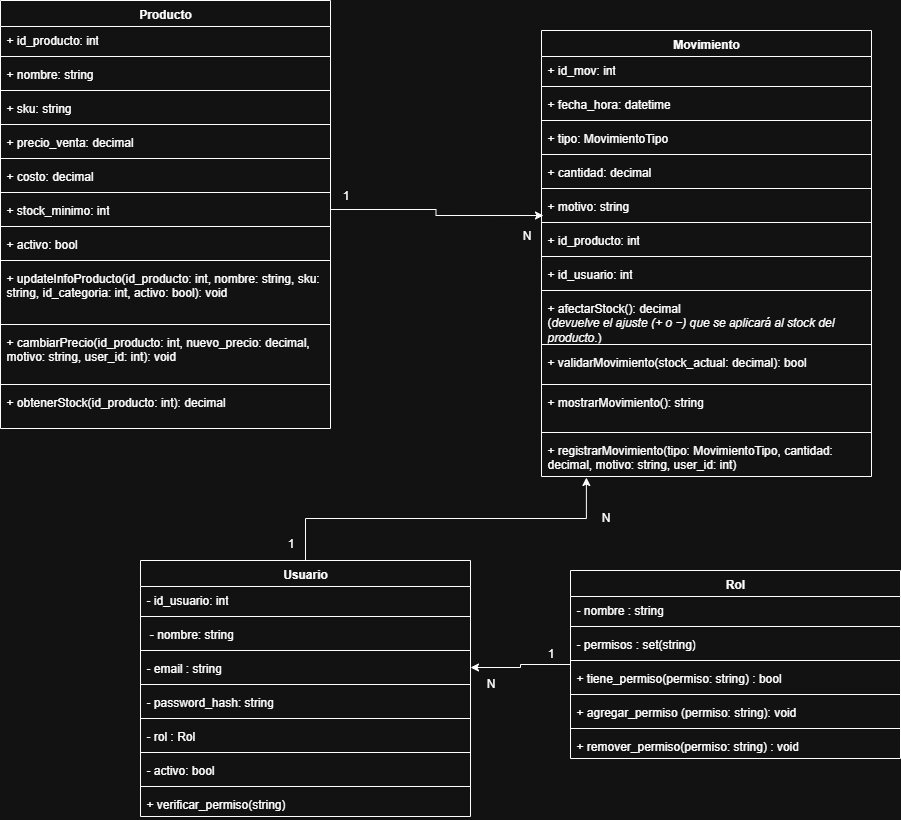
**US06:** Como administrador, quiero crear y gestionar usuarios para definir permisos de acceso.

**Sprints.**

| **N° de sprint** | 00 |
| --- | --- |
| **Sprint Backlog** | US01: Registrar productos en el inventario  1. Diseñar formulario de alta de productos. 2. Validar campos obligatorios (nombre, SKU, costo, precio, stock mínimo). 3. Conectar el formulario con la base de datos. 4. Implementar validaciones de duplicados (SKU único). 5. Guardar el producto en DB. 6. Probar creación y persistencia de productos.   **US02:** **Como administrador, quiero registrar entradas y salidas de productos para mantener actualizado el stock.**  1.Crear interfaz para registrar movimiento.  2.Definir lógica de actualización del stock (suma/resta).  3.Validar cantidades ingresadas (no negativas, no mayores a stock disponible en salida).  4.Guardar movimiento en DB con fecha y usuario.  5.Probar entradas y salidas con diferentes escenarios.  **US03: Como administrador, quiero recibir alertas cuando un producto esté bajo el stock mínimo.**  1.Definir condición de alerta (stock\_actual < stock\_mínimo).  2.Implementar notificación visual o por mensaje en sistema.  3.Configurar alerta automática al consultar inventario o realizar movimientos.  4.Probar con productos en diferentes niveles de stock.  **US04:** **Como administrador, quiero generar reportes para analizar el estado del inventario.**  1.Diseñar consultas SQL para obtener datos de inventario.  2.Crear módulo de generación de reportes (tablas, resúmenes).  3.Implementar filtros (por fecha, por producto).  4.Agregar exportación de reportes (PDF/Excel).  5.Probar integridad y actualización de datos en reportes.  **US05:** **Como empleado, quiero poder registrar ventas para que el stock se descuente automáticamente**.  1.Crear formulario de ventas.  2.Implementar lógica de descuento automático del stock.  3.Validar existencia suficiente de stock antes de confirmar.  4.Guardar venta en DB.  5.Probar flujo completo de venta y actualización.    **US06:** **Como administrador, quiero crear y gestionar usuarios para definir permisos de acceso.**  1.Diseñar formulario de registro de usuario.  2.Implementar roles (Administrador, Empleado).  3.Asignar permisos a cada rol.  4.Implementar validaciones de acceso según rol.  5.Probar usuarios con diferentes roles y accesos. |
| **Responsabilidades** |  |
| **Calendario** |  |
| **Inconvenientes:** | |

**Anexo I**

**-DIAGRAMA UML**

  
  
 ***Explicación de Clases y Métodos - Proyecto Control de Stock***

**Clase Producto:** La clase Producto representa la ficha base de un artículo. Contiene la información estable del producto (nombre, SKU, costo, precio de venta, stock mínimo, etc.). No guarda el stock en vivo, sino que el stock se calcula a partir de sus movimientos.  
Métodos:   
- update\_Info\_Producto(...): Permite actualizar la información básica del producto como nombre, sku, categoría y estado activo.  
 - cambiar\_Precio(...): Registra un cambio de precio, exigiendo un motivo y el usuario que lo realiza. - registrar Movimiento(...): Asocia un movimiento (entrada, salida o ajuste) al producto.  
 - obtener\_Stock(...): Devuelve el stock actual del producto sumando/restando sus movimientos.

**Clase Movimiento**: Representa cualquier cambio en el stock de un producto. Cada movimiento queda registrado con fecha, tipo, cantidad, motivo, producto afectado y usuario responsable.   
Métodos: - afectarStock(): Devuelve el ajuste (+ o -) que se aplicará al stock del producto. - validarMovimiento(...): Verifica si el movimiento es válido (ej: no permite stock negativo). - mostrarMovimiento(): Devuelve un texto legible con la información del movimiento. - registrarMovimiento(...): Crea y guarda un movimiento con sus datos principales.

Clase Usuario: La clase Usuario representa a una persona que accede al sistema.

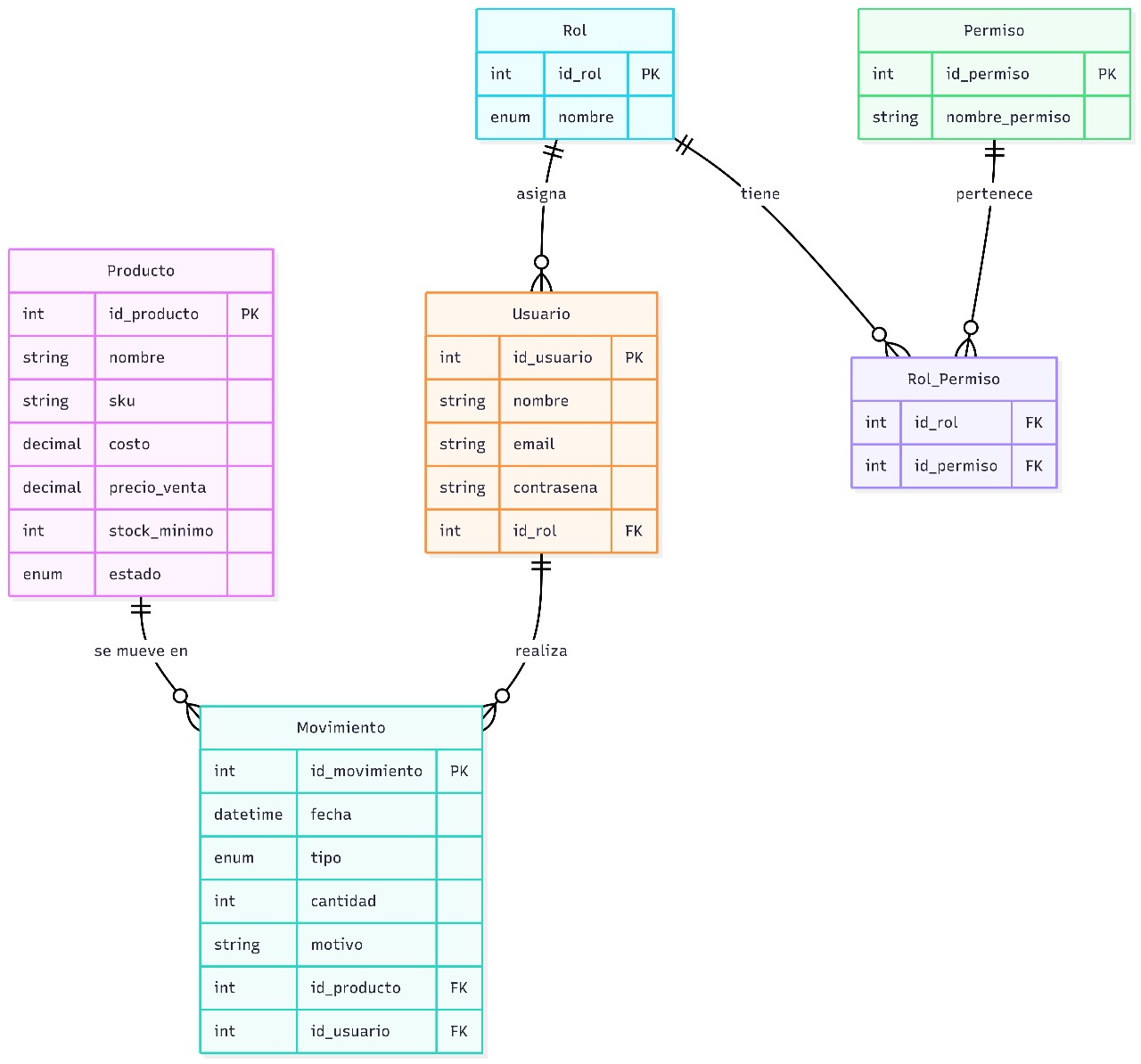
Cada usuario tiene un rol asignado que define qué operaciones puede ejecutar.  
Método:  
verificar\_permiso(...): Consulta el rol del usuario y verifica si tiene habilitado el permiso solicitado. Si no lo tiene, lanza un error de permiso denegado.

**Clase Rol**: La clase Rol define los permisos de un grupo de usuarios (ej: Administrador, Empleado). Un rol agrupa un conjunto de permisos en forma de strings.  
Métodos:   
tiene\_permiso(...): Verifica si un permiso específico existe dentro del rol.  
agregar\_permiso(...): Añade un nuevo permiso al rol.   
remover\_permiso(...): Quita un permiso del rol.

Relaciones entre Clases - Un Producto puede tener muchos Movimientos (1 a N). - Cada Movimiento está asociado a un Usuario (N a 1). - Un Usuario tiene un Rol (N a 1). - Un Rol puede estar asignado a varios Usuarios (1 a N).  
 Este diseño separa claramente las responsabilidades y permite un control de stock confiable y seguro.

**ANEXO II**

**-Diagrama Entidad Relación-**



Explicación:

El diagrama representa un sistema de **gestión de stock con usuarios, roles y permisos**:

**Entidad Producto (lo que se va a controlar en el inventario)**

- **Atributos:**

* + id\_producto (PK)
  + nombre
  + sku (código único de producto)
  + costo
  + precio\_venta
  + stock\_minimo
  + estado (activo / inactivo)

**Entidad Usuario (persona que utiliza el sistema)**

- **Atributos:**

* + id\_usuario (PK)
  + nombre
  + email
  + contraseña
  + id\_rol (FK) → Relación con la tabla **Rol**.

**Entidad Rol**

* **Atributos:**
  + id\_rol (PK)
  + nombre (ejemplo: administrador, empleado) Define los **roles de los usuarios**.

**Entidad Permiso (los que puede tener un rol)**

- **Atributos:**

* + id\_permiso (PK)
  + nombre\_permiso (ej: crear producto, modificar, borrar, entre otros)

**Entidad Rol\_Permiso (relación intermedia)**

- **Atributos:**

* + id\_rol (FK)
  + id\_permiso (FK)

Relación **muchos a muchos** entre **Rol** y **Permiso** (un rol puede tener varios permisos y un permiso puede pertenecer a varios roles)

**Entidad Movimiento (del stock)**

- **Atributos:**

* + id\_movimiento (PK)
  + fecha
  + tipo (entrada, salida, ajuste…)
  + cantidad
  + motivo
  + id\_producto (FK)
  + id\_usuario (FK)

Cada movimiento está relacionado con:

* **un producto** (id\_producto)
* **un usuario** (id\_usuario).

**Relaciones:**

**Usuario – Rol:** Un usuario tiene un rol (N:1).Un rol puede asignarse a muchos usuarios.

**Rol – Permiso (Rol\_Permiso):** Relación muchos a muchos (N:M)

**Usuario – Movimiento:** Un usuario **realiza** movimientos (1:N).

**Producto – Movimiento:** Un producto está asociado a varios movimientos (1:N).