

PROYECTO SCRUM-PYTHON

ESTUDIANTES

**ALEJANDRO RINCÓN PÉREZ
JAIME ANDRÉS PRADA TORRES
JHOAN SEBASTIAN DIAZ ARDILA
JULIÁN DAVID PIÑEROS CORONADO
MARIA CAMILA DIAZ TOLEDO**

DOCENTES INVOLUCRADO

JUAN CARLOS MARIÑO MORANTES

**CAMPUSLANDS
SALON U2
RUTA JAVA
FLORIDABLANCA
2024**

PROYECTO SCRUM

1. SITUACIÓN PROBLEMA

El gerente de “Enigma Store” tiene problemas con la actualización y modernización de su sistema de control de ventas; debido a la nueva solicitud por parte del gobierno la empresa debe entregar la facturación a sus clientes por correo electrónico y mantener en sus archivos las facturas digitales que se enviaran puesto que la facturación en papel debe darse por terminada. “Enigma Store” no cuenta con un inventario digital para poder realizar el proceso completamente digital puesto que su inventario se mantiene en libros escritos a mano.

Se encomendó el proyecto por la empresa de venta de ropa llamada “Enigma Store” para la elaboración de un programa que pueda registrar las ventas en facturas y permita la elaboración de un inventario por parte del administrador, donde este pueda crear, modificar, pero no eliminar inventario añadido, la empresa declara que desea tener una sección para el administrador y otra para el vendedor donde el pueda declarar la facturación y el administrador verificar las ganancias, los reportes de facturación y ambos visualizar el inventario de productos en stock, así como los descontinuados.

2. LEVANTAMIENTO DE REQUERIMIENTOS

Identificación de las Partes Interesadas

Stakeholders

- Cliente: Empresa Enigma Store
- Usuarios Finales: Administrador, Empleado de ventas, Clientes, Contador, Desarrolladores, Clientes potenciales del sistema.

Roles y Responsabilidades

- Cliente: Proveer información y validar requerimientos.
- Usuarios Finales: Probar el sistema y proveer retroalimentación.

Metodología de Levantamiento de Requerimientos

Técnicas Utilizadas

- Entrevistas con usuarios clave.
- Cuestionarios.
- Análisis de documentos existentes.

Proceso de Recopilación

- Reuniones iniciales con stakeholders de nivel operativo.
- Análisis y consolidación de la información.

3. REQUERIMIENTOS

3.1. Requerimientos Funcionales

Descripción de Funcionalidades

- El sistema debe contar con 2 entradas,

Administrador:

- Debe crear un inventario añadiendo los artículos por descripción y cantidad de ese mismo artículo
- Debe agregar el valor del costo del artículo y el precio de venta
- Debe poder modificar toda la información del artículo pero no borrar ningún código de prenda, ni repetirla.
- Debe verificar la información de las facturas.
- Debe verificar la información de ganancias.

Vendedor:

- Debe registrar las facturas
- Debe especificar los artículos de venta en la factura
- Debe adicionar el porcentaje del IVA
- Debe añadir descuentos de ser necesario.

3.2. Requerimientos No Funcionales

Rendimiento

- El sistema debe responder en menos de 3 segundos para la mayoría de las operaciones.

Seguridad

- Autenticación mediante usuario y contraseña.
- Autorización basada en roles.

Requerimientos Técnicos

Plataforma

- El sistema se desarrollará en Python.

Base de datos

- La información se guardará en archivos tipo .Json

4. HISTORIAS DE USUARIO CON CRITERIOS DE ACEPTACIÓN

EPICA GESTIÓN DE INVENTARIO

HISTORIA DE USUARIO						
Prioridad: Alta						
CÓDIGO DEL REQUERIMIENTO:	E1-HU1	Actor	Administrador			
NOMBRE DEL REQUERIMIENTO	Crear y añadir nuevos artículos al inventario					
Descripción						
Como administrador, quiero poder añadir nuevos artículos al inventario de la tienda para tener un registro completo y actualizado de los productos disponibles.						
Funcionalidad						
Permitir al administrador agregar nuevos artículos con los siguientes campos: descripción, cantidad, marca y talla.						
Criterios de aceptación	<ol style="list-style-type: none">El sistema debe permitir que el administrador ingrese la descripción del artículo.El sistema debe permitir que el administrador ingrese la cantidad de artículos.El sistema debe permitir que el administrador ingrese la marca.El sistema debe permitir que el administrador ingrese la talla del artículo.					
Restricciones						
El administrador solo puede agregar un código único por cada tipo de prenda y esta no se debe repetir.						

HISTORIA DE USUARIO						
Prioridad: Alta						
CÓDIGO DEL REQUERIMIENTO:	E1-HU2	Actor	Equipo de Desarrollo			
NOMBRE DEL REQUERIMIENTO	Añadir costos y valor de venta a artículos del inventario					
Descripción						
Como administrador, quiero poder añadir los precios de costo y el valor de venta del artículo para tener un control de los gastos y las ventas.						
Funcionalidad						
Permitir al administrador agregar los campos de costo del artículo y valor de venta del artículo						
Criterios de aceptación	<ol style="list-style-type: none"> El sistema debe permitir que el administrador ingrese el costo de las prendas El sistema debe permitir que el administrador ingrese el precio de venta de las prendas. 					
Restricciones						
El valor del costo y valor de venta solo debe ser modificado por el administrador.						

HISTORIA DE USUARIO						
Prioridad: Media						
CÓDIGO DEL REQUERIMIENTO:	E1-HU3	Actor	Equipo de Desarrollo			
NOMBRE DEL REQUERIMIENTO	Actualización de inventario					
Descripción						
Como administrador, quiero poder actualizar y modificar todos los datos de los artículos pero no borrar el código creado, para realizar correcciones en caso de error al ingresarlo.						
Funcionalidad						
Permitir al administrador modificar todos los campos de los artículos creados pero no eliminar el artículo completo.						
Criterios de aceptación	<ol style="list-style-type: none"> El sistema debe permitir que el administrador modifique la descripción de la prenda. El sistema debe permitir que el administrador modifique la cantidad de prendas. El sistema debe permitir que el administrador modifique la marca. El sistema debe permitir que el administrador modifique la talla del artículo. El sistema debe permitir que el administrador modifique el costo de las prendas El sistema debe permitir que el administrador modifique el precio de venta de las prendas. El sistema no debe permitir eliminar todo el artículo ni el código base 					
Restricciones						
El sistema no debe permitir eliminar todo el artículo ni el código base.						

HISTORIA DE USUARIO						
Prioridad: Baja						
CÓDIGO DEL REQUERIMIENTO:	E1-HU4	Actor	Equipo de Desarrollo			
NOMBRE DEL REQUERIMIENTO	Verificación de Facturas					
Descripción						
Como administrador, quiero poder visualizar las facturas realizadas por el vendedor para tener un control de las ventas y poder brindar la información al contador.						
Funcionalidad						
Permitir al administrador visualizar las facturas generadas por el vendedor, donde se muestran los productos, y el pago realizado por el cliente.						
Criterios de aceptación	<ol style="list-style-type: none"> El sistema debe permitir al administrador verificar las facturas realizadas por el vendedor. El sistema debe permitir al administrador verificar los productos de la factura y los pagos realizados. 					
Restricciones						
Las facturas no deben ser modificadas ni borradas del sistema.						

HISTORIA DE USUARIO						
Prioridad: Baja						
CÓDIGO DEL REQUERIMIENTO:	E1-HU5	Actor	Equipo de Desarrollo			
NOMBRE DEL REQUERIMIENTO	Verificación de Ganancias					
Descripción						
Como administrador, quiero poder visualizar las ganancias de las ventas realizadas por el vendedor para tener un control de los gastos, dinero adquirido y poder brindar la información al contador.						
Funcionalidad						
Permitir al administrador visualizar las ganancias generadas por el vendedor, donde se muestran los pagos realizados en las facturas menos el costo de los artículos.						
Criterios de aceptación	1. El sistema debe permitir al administrador verificar las ganancias obtenidas de las facturas después de descontar el costo de los artículos vendidos.					
Restricciones						
El sistema no debe mostrar el valor del iva y no debe ser tomado en cuenta como ganancia						

EPICA GESTIÓN DE FACTURACIÓN

HISTORIA DE USUARIO						
Prioridad: Alta						
CÓDIGO DEL REQUERIMIENTO:	E2-HU1	Actor	Equipo de Desarrollo			
NOMBRE DEL REQUERIMIENTO	Creación de Facturas					
Descripción						
Como administrador, quiero que el vendedor pueda crear una factura que tenga la información del cliente para poder brindar la factura al cliente por correo electrónico.						
Funcionalidad						
Permitir al vendedor crear una factura para cada venta realizada donde se muestran los datos requeridos del cliente: Nombre, identificación, correo electrónico.						
Criterios de aceptación	<ol style="list-style-type: none">1. El sistema debe permitir al vendedor crear facturas con la información del cliente: Nombre, Identificación, Correo electrónico.2. El sistema debe mostrar la fecha de creación de la factura y el número de factura					
Restricciones						
El número de factura no puede ser borrada, ni modificada la fecha.						
El número de factura debe ser único y secuencial.						

HISTORIA DE USUARIO						
Prioridad: Alta						
CÓDIGO DEL REQUERIMIENTO:	E2-HU2	Actor	Equipo de Desarrollo			
NOMBRE DEL REQUERIMIENTO	Adición de productos a Facturas					
Descripción						
Como administrador, quiero que el vendedor pueda registrar los productos vendidos en las facturas de cada cliente, para tener un control de las ventas e inventario y poder brindar la factura al cliente con mayor especificidad por correo electrónico.						
Funcionalidad						
Permitir al vendedor añadir los productos a la factura para cada venta realizada donde se muestran la información del cliente, también debe mostrar la cantidad de artículos vendidos y el precio de venta						
Criterios de aceptación	<ol style="list-style-type: none"> 1. El sistema debe permitir al vendedor adicionar los productos vendidos en cada factura del cliente. 2. El sistema debe descontar estos productos del inventario. 3. El sistema debe mostrar al vendedor los valores de venta de los productos en cada factura del cliente. 4. El sistema debe permitir asignar la cantidad de productos vendidos por cada artículo 					
Restricciones						
El sistema no debe permitir modificaciones a los productos agregados						

HISTORIA DE USUARIO						
Prioridad: Alta						
CÓDIGO DEL REQUERIMIENTO:	E2-HU3	Actor	Equipo de Desarrollo			
NOMBRE DEL REQUERIMIENTO	Adición de porcentaje IVA a Facturas					
Descripción						
Como administrador, quiero que el vendedor pueda agregar el valor del iva para la factura, este puede ser variante por ende siempre debe solicitarlo, para tener un control de los pagos de clientes y poder brindar la factura al cliente con mayor especificidad por correo electrónico, mostrando el total a pagar con el iva.						
Funcionalidad						
Permitir al vendedor añadir el valor del iva a la venta de los productos en la factura para cada venta realizada donde se muestre el total a pagar sin iva y con iva						
Criterios de aceptación	<ol style="list-style-type: none"> El sistema debe permitir al vendedor adicionar el iva a la venta a los productos en la factura El sistema debe mostrar el valor total de la compra sin iva y otro con iva. 					
Restricciones						
El porcentaje del iva siempre debe ser mostrado incluso cuando este sea igual a 0						

HISTORIA DE USUARIO						
Prioridad: Baja						
CÓDIGO DEL REQUERIMIENTO:	E2-HU4	Actor	Equipo de Desarrollo			
NOMBRE DEL REQUERIMIENTO	Adición de descuento a Facturas					
Descripción						
Como administrador, quiero que el vendedor pueda agregar algún descuento para el valor de la compra sin iva, este puede ser variante por ende siempre debe solicitarlo, para tener un control de los pagos de clientes y poder brindar la factura al cliente con mayor especificidad por correo electrónico, mostrando el total a pagar con el valor con descuento y el valor a pagar con el iva.						
Funcionalidad						
Permitir al vendedor añadir el valor del descuento a la venta de los productos en la factura para cada venta realizada donde se muestre el total a pagar sin iva y con iva.						
Criterios de aceptación	<ol style="list-style-type: none"> El sistema debe permitir al vendedor adicionar un descuento a la venta de artículos antes del iva. El sistema debe mostrar el valor total de la compra sin iva y otro con iva. 					
Restricciones						
El descuento siempre se debe aplicar antes del IVA para no afectar al pago del cliente.						

5. METODOLOGÍA

El desarrollo del proyecto para la empresa “Enigma Store” se establece bajo la Metodología ágil de Kanban la cual permite abordar un enfoque integral para la gestión del proyecto buscando la transparencia en la realización de las actividades impuestas a los integrantes del equipo de desarrollo visualizando los avances y las tareas completadas dando comprensión a las entregas realizadas a los clientes y los servicios adquiridos, donde se permite una adaptabilidad a los posibles y nuevos requerimientos del cliente ajustando los tiempos y las entregas de valor para cumplir con las expectativas del cliente, satisfacer sus necesidades y afrontar cualquier desafío o imprevisto.

El tablero designado para la implementación y asignación de tareas será digital en la plataforma ClickUp donde se dara acceso mediante invitacion directa y el enlace <https://app.clickup.com/9013352392/v/b/t/9013352392>

El Scrum Master a cargo dirigira la asignacion de tareas de cada Epica y los tiempos de entrega para los integrantes del Scrum Team (grupo de desarrollo); Se realizarán Daily Standup para la verificación, asignación y validación de tareas, durante el Daily Standup se considerarán las problemáticas o limitaciones del desarrollo de las actividades asignadas a cada miembro del equipo para su pronta solución y continuidad en el desarrollo del Sprint. Dando por sentado el marco de trabajo Scrum las actividades se realizarán de acuerdo al Backlog establecido, generando un Backlog Sprint para cada entrega de valor y las entregas de cada Sprint serán priorizadas para dar valor al cliente donde todo el equipo se auto-organiza para cumplir las tareas del Sprint brindando colaboración entre sus mismos integrantes.

Fases del proyecto

- **Fase de planificación:**
 - Definición de objetivos
 - Identificación de requisitos
- **Fase de diseño:**
 - Arquitectura visual del programa
 - Diseño de la interfaz
 - Verificación de nueva base de datos
- **Fase de desarrollo:**
 - Elaboración de código
 - Integración de las tareas asignadas
 - Pruebas de integración

Roles y Responsabilidades

I. Product Owner (Jhoan Sebastian Diaz Ardila)

- Creación y Gestión del Product Backlog: Prioriza los requisitos del usuario y las historias de usuario en importancia y valor para los entregables, asegurando el trabajo enfocado a la satisfacción del cliente
- Definición de requisitos: Analiza y define los requisitos funcionales y no funcionales del proyecto siempre con base de las necesidades del cliente
- Aceptación de Incrementos: Revisa y acepta los incrementos del Sprint cerciorándose de cumplir con los criterios de aceptación y las restricciones contempladas.
- Comunicación con Stakeholders: Representa los intereses de los stakeholders y se asegura de que el equipo de desarrollo entienda claramente los requisitos y expectativas del proyecto.
- Toma de Decisiones: Toma decisiones rápidas y efectivas sobre el rumbo del proyecto, priorizando el valor y la viabilidad.

II. Scrum Master (Alejandro Rincon Perez)

- Facilitador del Proceso: Se asegura de que el equipo siga las prácticas de Scrum correctamente.
- Remoción de Obstáculos: Identifica y elimina cualquier impedimento que pueda afectar el progreso del equipo.
- Coordinación de Reuniones: Organiza y facilita reuniones diarias (daily stand-ups), revisiones de sprint (sprint reviews) y retrospectivas.
- Soporte al Equipo: Brinda apoyo y coaching al equipo para mejorar continuamente sus prácticas y rendimiento.

III. Scrum Team (Jaime Prada Torres, Julian Piñeros, Maria Camila Diaz)

- Implementación de Funcionalidades: Desarrollan el código del sistema para el proyecto, asegurando que cumpla con los criterios de aceptación
- Pruebas y Validación: Realizan pruebas unitarias e integradas para asegurar que las funcionalidades desarrolladas se implementen correctamente.
- Colaboración en Equipo: Trabajan en estrecha colaboración con el Scrum Master y el Product Owner, participando activamente en todas las reuniones de Scrum.
- Resolución de Problemas: Identifican y solucionan problemas técnicos, asegurando que el desarrollo del proyecto avance sin contratiempos.
- Mejora Continua: Participan en las retrospectivas para identificar áreas de mejora y aplicar las lecciones aprendidas en futuros sprints.

Product-Backlog

La información de las Épicas a entregar se establecen en la plataforma Click Up cada Epica cuenta con las diferentes historias de usuario y se priorizan dentro de cada entregable para el Sprint.

En este proyecto se dividió el Scrum team en los 2 entregables puesto que el tiempo fue bastante limitado para la entrega al cliente dando prioridad a la Épica-Gestión de Inventario donde se puso mayor empeño para poder proseguir con la segunda entrega que era dependiente de las historias de usuario anterior

Nombre	Persona asignada	Fecha límite	Prioridad	Opciones
EPICA-GESTIÓN DE INVENTARIO 5	J.A.	Hoy	Urgente	...
E1-HU1-Historia-Crear y añadir nuevos artículos al inventario	J.A.	Hoy	Urgente	...
E1-HU2-Historia-Añadir costos y valor de venta a artículos del inventario	J.A.	Hoy	Alta	...
E1-HU3-Historia-Actualización de inventario	J.A.	Hoy	Alta	...
E1-HU4-Historia-Verificación de Facturas	J.A.	Hoy	Normal	...
E1-HU5-Historia-Verificación de Ganancias	J.A.	Hoy	Baja	...
EPICA-GESTIÓN DE FACTURAS 4	J.A.	Hoy	Alta	...
E2-HU1-Historia Creación de Facturas	J.A.	Hoy	Urgente	...
E2-HU2-Historia-Adición de productos a Factura	J.A.	Hoy	Alta	...
E2-HU3-Historia-Adición de porcentaje IVA a Factura	J.A.	Hoy	Alta	...
E2-HU4-Historia-Adición de descuento a Facturas.	J.A.	Hoy	Normal	...

El Scrum Master toma las historias de usuario de cada Épica y las reduce a tareas para cada miembro del Scrum Team facilitando la comprensión y alcanzando los plazos de entrega de cada epica para cumplir con los objetivos y el valor a entregar al cliente,

Sprint-Backlog

The screenshot shows a Scrum project board with the following structure:

- Product Backlog:** Contains tasks like "Epica-Gestión Administrador" and "Historia-Debe crear un inventario añadiendo los artículos po...".
- Sprint Backlog:** Contains tasks like "Facturación Con IVA", "Función para ver artículos según características", and "Función para Registrar Facturas".
- Pending:** Contains tasks like "Validación de Ingreso correcto de datos", "Implementar cálculo de totales y descuentos.", and "Implementar cálculo de totales y descuentos.".
- In Progress:** Contains tasks like "Facturación por fecha".
- Testing:** Contains tasks like "Implementación de reportes de ganancias".
- Done:** Contains tasks like "Función Para relacionarse con el inventario".

El Sprint Backlog es una lista detallada y específica de tareas y actividades que el equipo Scrum se compromete a completar durante un Sprint determinado. Estas tareas son seleccionadas del Product Backlog durante la reunión de Sprint Planning y representan el trabajo necesario para cumplir con los objetivos del Sprint y satisfacer los criterios de aceptación definidos para las historias de usuario seleccionadas.

Propósitos clave del Sprint Backlog incluyen:

- Planificación y Priorización:** Durante la reunión de Sprint Planning, el equipo Scrum colabora con el Product Owner para seleccionar las historias de usuario del Product Backlog que serán abordadas durante el Sprint. Estas historias se descomponen en tareas más pequeñas y manejables que constituyen el Sprint Backlog.
- Visibilidad y Transparencia:** El Sprint Backlog proporciona una visión clara y transparente de las actividades planificadas para el Sprint en curso. Cada tarea es visible para todos los miembros del equipo Scrum, facilitando la colaboración y asegurando que todos estén alineados con los objetivos y compromisos del Sprint.
- Gestión del Trabajo Diario:** Durante el Sprint, el Sprint Backlog actúa como una guía dinámica que ayuda al equipo a gestionar su trabajo diario. Cada tarea está asignada a un miembro del equipo, quien se compromete a completarla dentro del Sprint, monitoreando su progreso durante las reuniones diarias (Daily Standups).
- Adaptabilidad y Flexibilidad:** A medida que avanza el Sprint, el equipo puede ajustar el Sprint Backlog según sea necesario para abordar cambios

emergentes, nuevas prioridades o lecciones aprendidas durante el desarrollo. Esto permite al equipo Scrum adaptarse rápidamente y optimizar su enfoque para maximizar el valor entregado al final del Sprint.

Creación y Gestión del Sprint Backlog

- Descomposición de Historias de Usuario:** Cada historia de usuario seleccionada del Product Backlog se descompone en tareas más pequeñas y específicas. Estas tareas deben ser lo suficientemente claras y detalladas para que los miembros del equipo puedan trabajar de manera autónoma en su implementación.
- Estimación y Asignación:** Una vez descompuestas, las tareas del Sprint Backlog son estimadas por el equipo en términos de esfuerzo y complejidad. Esta estimación ayuda a priorizar y asignar las tareas de manera efectiva entre los miembros del equipo, asegurando una distribución equitativa del trabajo y cumpliendo con los compromisos del Sprint.
- Herramientas de Gestión:** En "Enigma Store", se utiliza ClickUp como plataforma centralizada para la gestión del Sprint Backlog. Los tableros Kanban dentro de ClickUp permiten a los miembros del equipo visualizar el estado de cada tarea, moverlas a través de diferentes etapas (como "Por hacer", "En progreso" y "Hecho") y actualizar su progreso en tiempo real.
- Revisión y Actualización Continua:** Durante el Sprint, el equipo monitorea continuamente el progreso del Sprint Backlog. Durante las reuniones diarias (Daily Standups), se discuten los avances, se identifican los obstáculos y se toman decisiones para ajustar el plan según sea necesario.

Gestión y Evaluación de los Incrementos

- Sprint Review:**
 - Al finalizar cada Sprint, se lleva a cabo una reunión de Sprint Review donde el equipo Scrum presenta los incrementos desarrollados al Product Owner y a los stakeholders.
 - Durante esta reunión, se evalúa si los incrementos cumplen con los criterios de aceptación definidos para las historias de usuario seleccionadas y se recibe retroalimentación para ajustes futuros.
- Demostración del Producto:**
 - Durante la Sprint Review, el equipo Scrum demuestra las funcionalidades completadas y discute cualquier problema o ajuste necesario.

- Esto permite a los stakeholders visualizar el progreso del producto y asegurarse de que se están cumpliendo sus expectativas y necesidades.

6. EVIDENCIA DE PLANTEAMIENTO DE PLATAFORMA DE TRABAJO

Reuniones

Dia 1

Dia 2

https://drive.google.com/file/d/1Q0CMsIhGGVSswQH_8BTMho70mlmOg0Ee/view?usp=drive_link

Dia 3

https://drive.google.com/file/d/1Qv1FRh2DuY_7xY7qkIrUinPw2UVwykB/view?usp=sharing

Review

<https://drive.google.com/file/d/1R9-anWgkE7hvChYIXQvmUDs1LtvbbcZ/view?usp=sharing>

Dia uno Creación de Historias

Epica-Gestión Administrador	Hoy	...
Historia-Debe crear un inventario añadiendo los artículos por descripción y cantidad de ese mismo artículo	Hoy	...
Historia-Debe agregar el valor del costo del artículo y el precio de venta	Hoy	...
Historia-Debe poder modificar toda la información del artículo pero no borrar ningún código de prenda, ni repetirla.	Hoy	...
Historia-Debe verificar la información de las facturas.	Hoy	...
Historia-Debe verificar la información de ganancias.	Hoy	...
Epica-Gestión Vendedor	Hoy	...
Historia-Debe registrar las facturas	Hoy	...
Historia-Debe especificar los artículos de venta en la factura	Hoy	...
Historia-Debe adicionar el valor del iva	Hoy	...
Historia-Debe añadir descuentos de ser necesario.	Hoy	...

+ Agregar Tarea

Dia uno (Socialización de historias de Usuario)

The screenshot shows a Scrum board in ClickUp. The board has five columns: PRODUCT-BACKLOG, SPRINT BACKLOG, IN PROGRESS, TESTING, and DONE. The SPRINT BACKLOG column is highlighted in yellow. There are several user stories listed under each column, each with a status indicator (blue dot for To Do, green dot for Doing, red dot for Done) and a priority level (Hoy or Urgente). The left sidebar shows project navigation and a list of spaces, with 'Tienda Enigma' selected. A context menu is open over one of the stories in the SPRINT BACKLOG column.

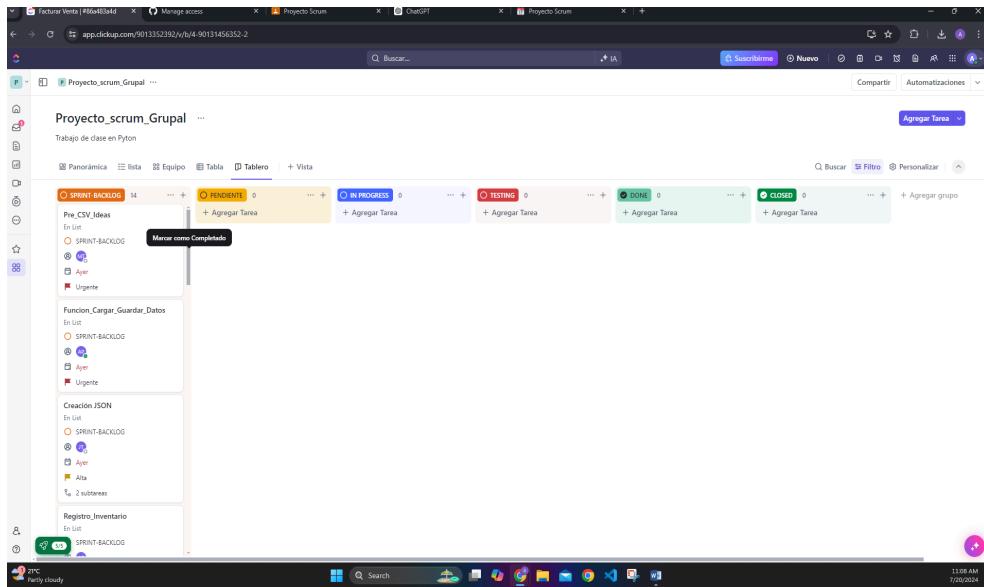
En la función de Facturación se deben tener en cuenta los...
En List
PRODUCT-BACKLOG
Hoy
Urgente

En La facturación debe estar el iva
En List
PRODUCT-BACKLOG
Hoy
Urgente

Función para ver artículos según características
En List
PRODUCT-BACKLOG
Hoy
Urgente

Función para Registrar Facturas
En List
PRODUCT-BACKLOG
Hoy
Urgente

Evidencia 2 de Socialización del historias



Gestión interna para la asignación usando tablas en click-up

The screenshot shows a ClickUp workspace titled "Tienda Enigma". On the left, there's a sidebar with project navigation and a "Tienda Enigma" board. The main area displays a table titled "Panorámica" with 21 rows of tasks. The columns include: #, NAME, PERSONA ASIGNADA, ESTADO, and FECHA LÍMITE. Most tasks are labeled "PRODUCT BACKLOG" under "ESTADO" and "Hoy" under "FECHA LÍMITE".

#	NAME	PERSONA ASIGNADA	ESTADO	FECHA LÍMITE	PROGRESO
1	Epic-Gestión Administrador	Jhoan Sebastian Diaz Aranda	PRODUCT BACKLOG	Hoy	■■■■■
2	Epic-Gestión Vendedores	Jhoan Sebastian Diaz Aranda	PRODUCT BACKLOG	Hoy	■■■■■
3	Crear formulario para añadir artículos.	Jamie Andres Prada Torres	PRODUCT BACKLOG	Hoy	■■■■■
4	Implementar validación de datos que no se repitan ID en el formulario.	Jamie Andres Prada Torres	PRODUCT BACKLOG	Hoy	■■■■■
5	Guardar datos en la base de datos.	Alejandro Rincon Perez	PRODUCT BACKLOG	Hoy	■■■■■
6	Crear formulario para añadir artículos.	Jamie Andres Prada Torres	PRODUCT BACKLOG	Hoy	■■■■■
7	Guardar datos en la base de datos.	Alejandro Rincon Perez	PRODUCT BACKLOG	Hoy	■■■■■
8	Crear función para editar artículos.	Alejandro Rincon Perez	PRODUCT BACKLOG	Hoy	■■■■■
9	Crear una función para modificar artículos.	Julian David Pifars Corradino	PRODUCT BACKLOG	Hoy	■■■■■
10	Crear Función Para Precio de Venta.	Julian David Pifars Corradino	PRODUCT BACKLOG	Hoy	■■■■■
11	Modificar costo.	Julian David Pifars Corradino	PRODUCT BACKLOG	Hoy	■■■■■
12	Crear función para generar facturas.	Alejandro Rincon Perez, Jamie Andres Prada Torres	PRODUCT BACKLOG	Hoy	■■■■■
13	Implementar cálculo de total.	Alejandro Rincon Perez	PRODUCT BACKLOG	Hoy	■■■■■
14	Función Para reloj cronómetro con el tiempo de ejecución.	Jamie Andres Prada Torres	PRODUCT BACKLOG	Hoy	■■■■■
15	Validación de ingreso correcto.	Julian David Pifars Corradino	PRODUCT BACKLOG	Hoy	■■■■■
16	Implementación de reportes.	Maria Camila Diaz Toledo	PRODUCT BACKLOG	Hoy	■■■■■
17	Facturación por fecha.	Maria Camila Diaz Toledo	PRODUCT BACKLOG	Hoy	■■■■■
18	Función para Registrar Facturas.	Jamie Andres Prada Torres	PRODUCT BACKLOG	Hoy	■■■■■
19	Función para ver artículos según su precio.	Maria Camila Diaz Toledo	PRODUCT BACKLOG	Hoy	■■■■■
20	En La facturación debe estar el número de factura.	Jamie Andres Prada Torres	PRODUCT BACKLOG	Hoy	■■■■■
21	En la Función de Facturación se debe tener la opción de pagar.	Jamie Andres Prada Torres	PRODUCT BACKLOG	Hoy	■■■■■

Día 2 Asignación de tareas Sprint Backlog

The screenshot shows the same ClickUp workspace "Tienda Enigma". The main area displays a table titled "Panorámica" with tasks categorized into four columns: "PENDING", "IN PROGRESS", "TESTING", and "DONE". Each column has a count of 0, 0, 0, and 0 respectively. The tasks listed in the table are identical to the ones in the Product Backlog, though some descriptions are partially cut off.

Estado	Tareas
PENDING	Crear formulario para añadir artículos. Implementar validación de datos que no se repitan ID en el formulario.
IN PROGRESS	Guardar datos en la base de datos.
TESTING	Crear formulario para añadir artículos.
DONE	

Tareas Pendientes

The screenshot shows a project management interface with a sidebar on the left containing navigation links like Inicio, Bandeja de entrada, Documentos, Panels, Clips, Hojas de horas, and Más. The main area displays a board with several columns:

- PRODUCT-BACKLOG**: Contains 2 items, one of which is "Epic-Gestión Administrador".
- SPRINT-BACKLOG**: Contains 0 items.
- PENDIENTE**: Contains 19 items, including "Función para ver artículos según características" and "Función para Registrar Facturas".
- IN PROGRESS**: Contains 0 items.
- TESTING**: Contains 0 items.
- DONE**: Contains 0 items.

Each card in the PENDIENTE column has a "Pendiente" status indicator and includes filters for Hoy and Urgente. A tooltip for the "Función para Registrar Facturas" card provides historical context: "Epic-Gestión Administrador Historia-Debe crear un inventario añadiendo los artículos por...".

Avances y progresos Click-up

The screenshot shows a Jira board titled "Tienda Enigma". The board is organized into several columns representing different stages of development:

- PRODUCT-BACKLOG**: Contains several items, some of which are expanded to show sub-tasks.
- SPRINT-BACKLOG**: Contains 0 items.
- PENDIENTE**: Contains 14 items, including:
 - Facturación Con IVA
 - Función para ver artículos según características
 - Función para Registrar Facturas
 - Validación de ingreso correcto de datos
 - Facturación por fecha
- IN PROGRESS**: Contains 3 items, including:
 - Función para ver artículos según características
 - Función para Registrar Facturas
 - Validación de ingreso correcto de datos
- TESTING**: Contains 2 items, including:
 - Implementación de reportes de ganancias.
 - Función Para relacionarse con el inventario
- DONE**: Contains 0 items.

The interface includes a sidebar with navigation links like "Inicio", "Bandeja de entrada", and "Panieres". At the bottom, there's a taskbar with icons for search, file, and other applications.

Avances y progresos Click-up

The screenshot shows a Click-up board titled "Tienda Enigma". The board is organized into several columns representing different stages of development:

- PENDIENTE**: Contains 2 items, including:
 - Crear una función para modificar Costos
 - Crear formulario para añadir artículos.
- IN PROGRESS**: Contains 9 items, including:
 - Validación de ingreso correcto de datos
 - Crear Funcion Para Precio de Venta
 - Guardar datos en la base de datos Json.
 - Implementar validación de datos que no se repitan ID en el...
 - Facturación Con IVA
 - Crear función para editar artículos Modificar Estado.
- TESTING**: Contains 8 items, including:
 - Implementación de reportes de ganancias.
 - Guardar datos en la base de datos Json.
 - Implementar validación de datos que no se repitan ID en el...
 - Facturación Con IVA
 - Crear función para editar artículos Modificar Estado.
- DONE**: Contains 0 items.

The interface includes a sidebar with navigation links like "Inicio", "Bandeja de entrada", and "Panieres". At the bottom, there's a taskbar with icons for search, file, and other applications.

Avances y progresos Click-up Testing

The screenshot shows a Click-up project board titled "Tienda Enigma". The board has six columns: PRODUCT-BACKLOG, SPRINT-BACKLOG, PENDIENTE, IN PROGRESS, TESTING, and DONE. Each column contains several tasks with status indicators (e.g., IN PROGRESS, TESTING, DONE) and due dates (e.g., Hoy, Urgente). A sidebar on the left lists project documents and panels.

- PRODUCT-BACKLOG:**
 - Epica-Gestión Administrador
 - História-Debe crear un inventario añadiendo los artículos po...
 - En List
 - PRODUCT-BACKLOG
 - Hoy
 - Urgente
 - 5 subtareas
- SPRINT-BACKLOG:**
 - + Agregar Tarea
- PENDIENTE:**
 - + Agregar Tarea
- IN PROGRESS:**
 - Guardar datos en la base de datos
 - Json
 - En List
 - IN PROGRESS
 - Hoy
 - Urgente
- TESTING:**
 - Facturación Con IVA
 - En List
 - IN PROGRESS
 - Hoy
 - Urgente
- DONE:**
 - Crear una función para modificar Costos
 - En List
 - TESTING
 - Hoy
 - Urgente

Avances y progresos Click-up

The screenshot shows a Click-up project board titled "Tienda Enigma". The board has six columns: PRODUCT-BACKLOG, SPRINT-BACKLOG, PENDIENTE, IN PROGRESS, TESTING, and DONE. Each column contains several tasks with status indicators (e.g., IN PROGRESS, TESTING, DONE) and due dates (e.g., Hoy, Urgente). A sidebar on the left lists project documents and panels.

- PRODUCT-BACKLOG:**
 - Epica-Gestión Administrador
 - História-Debe crear un inventario añadiendo los artículos po...
 - En List
 - PRODUCT-BACKLOG
 - Hoy
 - Urgente
 - 5 subtareas
- SPRINT-BACKLOG:**
 - + Agregar Tarea
- PENDIENTE:**
 - + Agregar Tarea
- IN PROGRESS:**
 - + Agregar Tarea
- TESTING:**
 - Función de Facturación con descuentos
 - En List
 - IN PROGRESS
 - Hoy
 - Urgente
- DONE:**
 - Función para ver artículos según características
 - En List
 - TESTING
 - Hoy
 - Urgente

Finalización de tareas Click up

The screenshot shows a Click Up project board titled 'Tienda Enigma'. The board has several columns: PRODUCT-BACKLOG, SPRINT-BACKLOG, PENDIENTE, IN PROGRESS, TESTING, and DONE. The PENDIENTE column is highlighted in yellow. On the left, there's a sidebar with navigation links like 'Inicio', 'Documentos', and 'Panieres'. A context menu is open over a task in the PENDIENTE column, listing options such as 'Agregar Tarea', 'Personalizar', and 'Agregar Tarea' again. Another context menu is visible on the right side of the board.

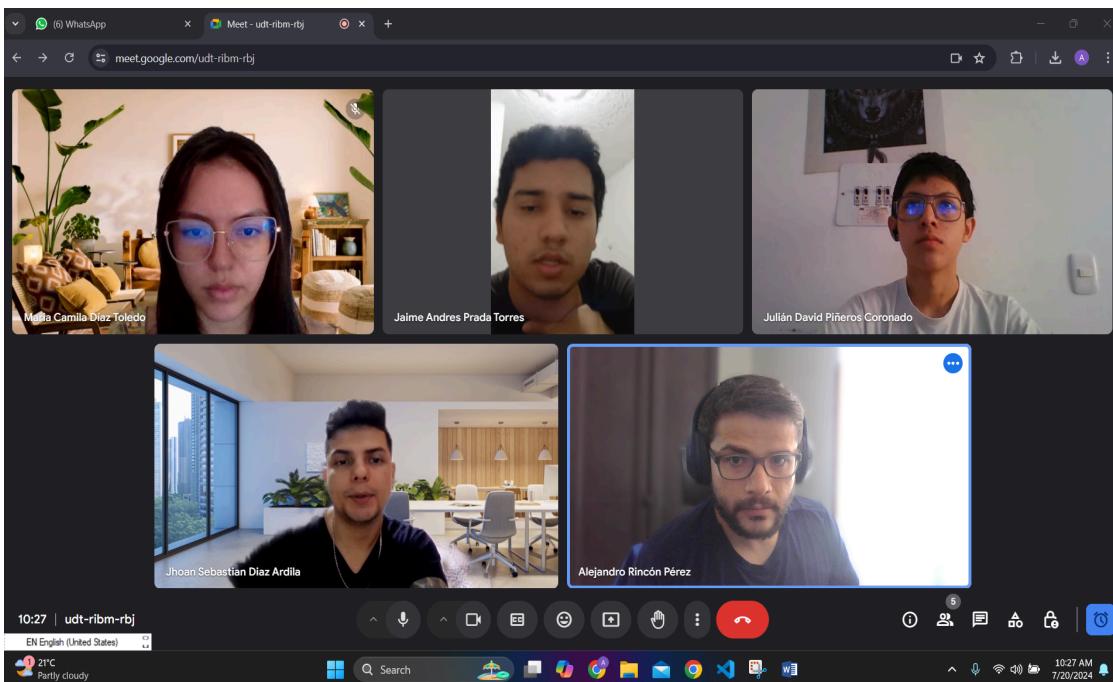
Administración de tiempos

A Click Up task card for the task 'Crear una función para modificar Costos'. The card includes fields for Estado (IN PROGRESS), Personas asignadas (empty), Fecha límite (Hoy), Prioridad (Urgente), Duración estimada (Vacio), Registrar el tiempo (0:00:18), Etiquetas (empty), Relaciones (empty), and a description section with 'Agregar descripción' and 'Escribe con IA' buttons.

Reunión dia 1



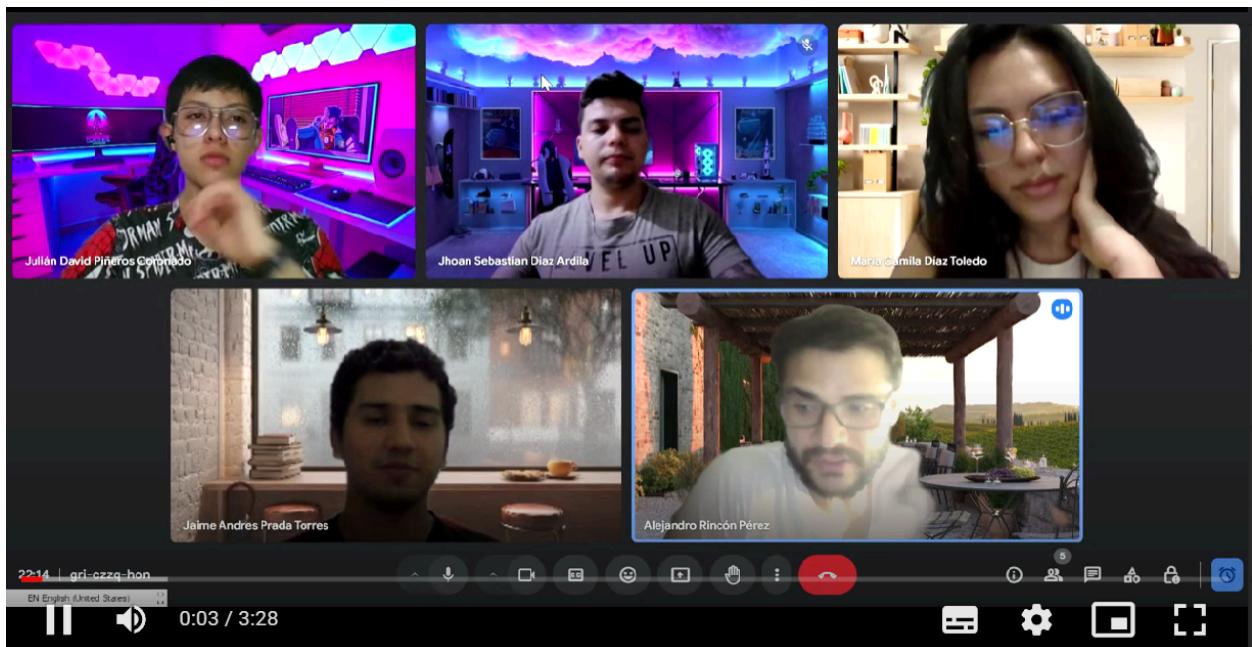
Reunión dia 2



Reunión dia 3



Review



Cumpliento de Tareas con software util para el Cliente

Tarea - Crear formulario para añadir artículos.

```

def Registro_prenda_nueva():
    cargar_datos(Ruta_JSON_INVENTARIO, inventario)

    ID_Ropa = generate_idp()
    info_elemento = {}

    # Descripción del producto
    while True:
        descripcion = input("Ingrese la descripción del producto: ").strip()
        if descripcion:
            info_elemento["descripcion"] = descripcion.upper()
            break
        else:
            print("La descripción no puede estar vacía. Por favor, ingresa una descripción válida.")

    # Marca del producto
    while True:
        marca = input("Ingrese la Marca del Producto: ").strip()
        if marca:
            info_elemento["Marca"] = marca.upper()
            break
        else:
            print("La marca no puede estar vacía. Por favor, ingresa una marca válida.")

    # Cantidad de artículos
    while True:
        cantidad = input("Cantidad de artículos a registrar: ").strip()
        if cantidad.isdigit() and int(cantidad) > 0:
            info_elemento["cantidad"] = int(cantidad)
            break
        else:
            print("La cantidad debe ser un número. Inténtalo de nuevo.")

    # Talla del producto
    while True:
        talla = input("Ingrese la talla del Producto: ").strip()

```

```

{
    "JaimePradali, yesterday * Test: Prueba Factura": [
        {
            "P001": {
                "descripcion": "PANTALON",
                "Marca": "X",
                "cantidad": "10",
                "Talla": "30",
                "estado": "Activo",
                "costo": "100",
                "precio": "150"
            },
            "P002": {
                "descripcion": "CAMISA",
                "Marca": "X",
                "cantidad": 8,
                "Talla": "S",
                "estado": "Activo",
                "costo": 40,
                "precio": 100
            },
            "P003": {
                "descripcion": "CHAQUETA",
                "Marca": "S",
                "cantidad": 11,
                "Talla": "L",
                "estado": "Activo",
                "costo": 50,
                "precio": 100
            },
            "P004": {
                "descripcion": "PANTALON",
                "Marca": "LEVELS",
                "cantidad": 0,
                "Talla": "32",
                "estado": "Activo",
                "costo": 50,
                "precio": 100
            }
    ]
}

```

Tarea - Implementar validación de datos que no se repitan ID en el inventario.

```

def generate_idp(prefix='P', width=3):
    global current_idP, inventario

    if current_idP is None:
        cargar_datos(Ruta_JSON_INVENTARIO, inventario)
        if inventario:
            max_id = max(int(key[1:]) for key in inventario.keys())
            current_idP = max_id + 1
        else:
            current_idP = 1

    new_id = f"{prefix}{current_idP:0{width}}"
    current_idP += 1
    return new_id

def producto_existe(descripcion, marca, talla):
    for item_id, item_info in inventario.items():
        if (item_info["descripcion"] == descripcion.upper() and
            item_info["Marca"] == marca.upper() and
            item_info["Talla"] == talla.upper()):
            return True
    return False

```

Tarea - Guardar datos en la base de datos Json.

```

import json      JaimePrada11, yesterday • feat: Prueba Facturas

Inventario = {}
Facturacion = {}

Ruta_JSON_INVENTARIO = "Inventario.json"
RUTA_JSON_FACTURAS = "Facturacion.json"

def guardar_datos(Ruta_JSON, Datos):
    try:
        contenido = json.dumps(Datos, indent=4, ensure_ascii=False)
        with open(Ruta_JSON, "w", encoding='utf-8') as file:
            file.write(contenido)
    except Exception as e:
        print(f"Error al guardar los datos: {e}")

def cargar_datos(Ruta_JSON, Datos):
    try:
        with open(Ruta_JSON, mode="r", encoding='utf-8') as archivo:
            Datos.update(json.load(archivo))
    except Exception as e:
        print(f"Error al cargar Datos {e}")

```

Tarea - Crear función para editar artículos Modificar Estado.

```
def Verificar_Estado():
    print("""[VERIFICAR ESTADO]"""")
    cargar_datos(Ruta_JSON_INVENTARIO, Inventario)
    ID_Producto = input("Id del producto: ")
    if ID_Producto not in Inventario:
        print("El ID del producto no existe en el inventario.")
        return
    if Inventario[ID_Producto]["cantidad"] > 0:
        print("Estado: Activo")
        Inventario[ID_Producto]["estado"] = "Activo"
        guardar_datos(Ruta_JSON_INVENTARIO, Inventario)
        print("Información guardada")
        print("*****")
        return
    elif Inventario[ID_Producto]["cantidad"] == 0:
        print("Estado del producto: Sin stock")
        try:
            Pregunta_Estado = int(input("1. Para agregar elementos 2. Para descontinuar el producto 3. Para salir: "))
            if Pregunta_Estado == 1:
                Cambio_Cantidad()
                return
            elif Pregunta_Estado == 2:
                Inventario[ID_Producto]["estado"] = "Descontinuado"
                guardar_datos(Ruta_JSON_INVENTARIO, Inventario)
                print("Información guardada")
                print("*****")
            elif Pregunta_Estado == 3:
                return
            else:
                print("Opción no válida.")
                Verificar_Estado()
        except ValueError:
            print("Entrada no válida, por favor ingrese un número.")
            Verificar_Estado()
```

Tarea - Crear una función para modificar Costos

```
def Cambio_Costo():
    print("""[CAMBIO DE COSTO]"""")
    cargar_datos(Ruta_JSON_INVENTARIO, Inventario)
    ID_Producto = input("Id del producto: ")
    if ID_Producto not in Inventario:
        print("El ID del producto no existe en el inventario.")
        return
    print("El costo actual del producto es:")
    print(Inventario[ID_Producto]["costo"])
    print("¿Desea cambiar el costo del producto?")
    try:
        Pregunta = int(input("Si desea cambiar el costo, ingrese 1 para SI, 2 para NO, 3 para Salir: "))
        if Pregunta == 1:
            try:
                Costo_nuevo = float(input("Nuevo costo: "))
                Inventario[ID_Producto]["costo"] = Costo_nuevo
                guardar_datos(Ruta_JSON_INVENTARIO, Inventario)
                print("Información guardada")
                print("*****")
            except ValueError:
                print("Entrada no válida, el costo debe ser un número.")
        elif Pregunta == 2:
            print("No se realizaron cambios.")
        elif Pregunta == 3:
            print("Saliendo sin cambios.")
        else:
            print("Opción no válida.")
    except ValueError:
        print("Entrada no válida, por favor ingrese un número.")
```

Tarea - Crear una función para modificar Precios

```
def Cambio_Precio():
    cargar_datos(Ruta_JSON_INVENTARIO, Inventario)
    ID_Producto = input("Id del producto: ")
    if ID_Producto not in Inventario:
        print("El ID del producto no existe en el inventario.")
        return
    print("El precio actual del producto es:")
    print(Inventario[ID_Producto]["precio"])
    print("¿Desea cambiar el precio del producto?")

    try:
        Pregunta = int(input("Si desea cambiar el precio, ingrese 1 para SI, 2 para NO, 3 para Salir: "))
        if Pregunta == 1:
            try:
                precio_nuevo = float(input("Nuevo precio: "))
                Inventario[ID_Producto]["precio"] = precio_nuevo
                guardar_datos(Ruta_JSON_INVENTARIO, Inventario)
                print("Información guardada")
                print("*****")
            except ValueError:
                print("Entrada no válida, el precio debe ser un número.")
        elif Pregunta == 2:
            print("No se realizaron cambios.")
        elif Pregunta == 3:
            print("Saliendo sin cambios.")
        else:
            print("Opción no válida.")
    except ValueError:
        print("Entrada no válida, por favor ingrese un número.")
```

Tarea - Crear una función para modificar Talla

```
def Cambio_Talla():
    print("CAMBIO DE TALLA")
    cargar_datos(Ruta_JSON_INVENTARIO, Inventario)
    ID_Producto = input("Id del producto: ")
    if ID_Producto not in Inventario:
        print("El ID del producto no existe en el inventario.")
        return

    print("Talla actual del producto:")
    print(Inventario[ID_Producto]["Talla"])
    print("¿Desea cambiar la Talla del Producto?")
    print("Recuerde que la talla está ligada al ID del producto.")

    try:
        Pregunta = int(input("Si desea cambiar la Talla, ingrese 1. para SI, 2. Para NO, 3. Salir: "))
        if Pregunta == 1:
            Talla_nueva = input("Talla a Cambiar: ")
            Inventario[ID_Producto]["Talla"] = Talla_nueva
            guardar_datos(Ruta_JSON_INVENTARIO, Inventario)
            print("Información Guardada")
            print("*****")
        elif Pregunta == 2:
            print("No se realizaron cambios.")
        elif Pregunta == 3:
            print("Saliendo sin cambios.")
        else:
            print("Opción no válida.")
    except ValueError:
        print("Entrada no válida, por favor ingrese un número.")
```

ⓘ You have Wi
your system.
extension fro

Tarea - Crear función para generar facturas.

```
< def facturas():
    factura = {}
    cargar_datos(RUTA_JSON_FACTURAS, Facturacion)
    cargar_datos(Ruta_JSON_INVENTARIO, Inventario)

    IDFactura = generate_id()
    factura["fecha"] = pedir_fecha()

    while True:
        id_usuario = input("ID del usuario: ").strip()
        if id_usuario.isdigit() and len(id_usuario) >= 5:
            factura["ID"] = id_usuario
            break
        else:
            print("El ID del usuario debe ser un numero y minimo de 5 digitos.")

    while True:
        nombre = input("Nombre: ").strip().upper()
        if nombre:
            factura["nombre"] = nombre
            break
        else:
            print("El nombre no puede estar vacío.")

    while True:
        email = input("Ingresa el email: ").strip().upper()
        if validar_email(email):
            factura["Email"] = email
            break
        else:
            print("Email no válido. Por favor, ingresa un email correcto.")

    while True:
        telefono = input("Ingresa el celular (7 a 10 dígitos): ").strip()
        if telefono.isdigit() and 7 <= len(telefono) <= 10:
            factura["telefono"] = telefono
            break
        else:
            print("El teléfono móvil debe contener entre 7 y 10 números.")

def facturas():
    factura["telefono"] = telefono
    break
else:
    print("El teléfono móvil debe contener entre 7 y 10 números.")

factura["Productos"] = {}
total_costo = 0
total_subtotal = 0

while True:
    ID_Producto = input("ID del producto (o escribe 'fin' para terminar): ").strip().upper()
    if ID_Producto.upper() == 'FIN':
        break

    if ID_Producto in Inventario:
        cantidad_disponible = Inventario[ID_Producto]["cantidad"]

        if cantidad_disponible == 0:
            print("Producto sin stock disponible.")
            continue

        while True:
            try:
                cantidad_solicitada = int(input("Cantidad del producto: ").strip())
                if cantidad_solicitada <= 0:
                    print("La cantidad debe ser un número positivo.")
                elif cantidad_solicitada > cantidad_disponible:
                    print(f"No hay suficiente cantidad en inventario.\nCantidad disponible: {cantidad_disponible}")
                while True:
                    respuesta = input(f"\nDesea proceder con la cantidad disponible: {cantidad_disponible} ?")
                    if respuesta == 's':
                        cantidad_a_facturar = cantidad_disponible
                        break
                    elif respuesta == 'n':
                        print("Producto no añadido a la factura.")
                        cantidad_a_facturar = 0
            except ValueError:
                print("Por favor, ingresa un número entero.")

            if cantidad_a_facturar > cantidad_disponible:
                print("Cantidad solicitada excede la cantidad disponible en inventario.")


            factura["Productos"].append({
                "ID_Producto": ID_Producto,
                "cantidad": cantidad_a_facturar
            })
            total_costo += cantidad_a_facturar * Inventario[ID_Producto]["precio"]
            total_subtotal += cantidad_a_facturar * Inventario[ID_Producto]["precio"]
```

Tarea - Implementar cálculo de totales y descuentos.

```
if total_subtotal > 0:
    while True:
        try:
            descuento_porcentaje = float(input("Ingresa el descuento en porcentaje (0-100): ").strip())
            if 0 <= descuento_porcentaje <= 100:
                break
            else:
                print("El descuento debe estar entre 0 y 100.")
        except ValueError:
            print("Por favor, ingresa un valor numérico para el descuento.")

    descuento_monto = total_subtotal * (descuento_porcentaje / 100)
    subtotal_con_descuento = total_subtotal - descuento_monto
    factura["Costo"] = total_costo
    factura["SubTotal"] = total_subtotal
    factura["descuento"] = descuento_monto
    factura["iva"] = subtotal_con_descuento * 0.19
    factura["Total"] = subtotal_con_descuento + factura["iva"]

    Facturacion[IDFactura] = factura
    try:
        guardar_datos(RUTA_JSON_FACTURAS, Facturacion)
        guardar_datos(Ruta_JSON_INVENTARIO, Inventario)
        print("Factura creada exitosamente.")
    except IOError as e:
        print(f"Error al guardar los datos: {e}")
    else:
        print("No se añadieron productos a la factura.")
```

ⓘ You have Windows Subsystem for Linux installed on your system. Do you want to enable the Microsoft Store extension from Microsoft?

Tarea - Reportes de inventario

```
def mostrar_stock():
    cargar_datos(Ruta_JSON_INVENTARIO, Inventario)
    print("""
+-----+-----+-----+-----+-----+
| CODIGO | DESCRIPCIÓN | MARCA | CANTIDAD | TALLA | ESTADO | COSTO | PRECIO |
+-----+-----+-----+-----+-----+
""")

    print(f'{{"CODIGO":<10} {"DESCRIPCIÓN":<15} {"MARCA":<10} {"CANTIDAD":<10} {"TALLA":<10} {"ESTADO":<10} {"COSTO":<10} {"PRECIO":<10}}')
    print("-" * 90)
    for codigo, detalles in Inventario.items():
        print(f'{codigo:<10} {detalles["descripcion"]:<15} {detalles["Marca"]:<10} {detalles["cantidad"]:<10} {detalles["Talla"]:<10} {detalles["Estado"]:<10} {detalles["Costo"]:<10} {detalles["Precio"]:<10}')
    print("-" * 90)

def mostrar_stock_especifico(criterio, valor):
    cargar_datos(Ruta_JSON_INVENTARIO, Inventario)

    print("""
+-----+-----+-----+-----+-----+
| CODIGO | DESCRIPCIÓN | MARCA | CANTIDAD | TALLA | ESTADO | COSTO | PRECIO |
+-----+-----+-----+-----+-----+
""")
```

```

def mostrar_stock_especifico(criterio, valor):
    cargar_datos(Ruta_JSON_INVENTARIO, Inventario)

    print("""
    _____
    |       |
    |   INVENTARIO   |
    |       |
    """)

    print(f"{'CODIGO':<10} {'DESCRIPCIÓN':<15} {'MARCA':<10} {'CANTIDAD':<10} {'TALLA':<10} {'ESTADO':<10} {'COSTO':<10} {'PRECIO':<10}")
    print("-" * 90)

    encontrado = False
    for codigo, detalles in Inventario.items():
        if criterio in detalles and valor.upper() in str(detalles[criterio]).upper():
            print(f"{codigo:<10} {detalles['descripcion']:<15} {detalles['Marca']:<10} {detalles['cantidad']:<10} {detalles['Talla']:<10} {detalles['estado']:<10} {detalles['costo']:<10} {detalles['precio']:<10}")
            encontrado = True

    if not encontrado:
        print("No se encontraron resultados para el criterio de búsqueda.")

    print("-" * 90)

```

```

def realizar_busqueda():

    while True:
        print("MOSTRAR INVENTARIO")
        print("****70")
        print('1. Ver información stock total')
        print('2.Ver información articulo específico')

        try:
            opc = int(input("Digite el número de la opción que desea elegir: "))
            if opc == 1:
                mostrar_stock()
                continuar=int(input('¿Desea seguir buscando? \n1.Sí \n2.No \n'))
                if continuar == 1:
                    continue
                elif continuar == 2:
                    print('saliendo...')
                    break
                elif opc != 1 or 2:
                    print('Opción no valida')
            elif opc == 2:
                cargar_datos(Ruta_JSON_INVENTARIO, Inventario)
                criterios = ['descripcion', 'Marca', 'cantidad', 'Talla', 'estado', 'costo', 'precio']
                n = 0
                for i in criterios:
                    n = n+1
                    print(n,i)
                eleccion = int(input('Digite el número de la opción que desea elegir: '))
                eleccion = eleccion-1
                elegido = criterios[eleccion]
                criterio = str(elegido)
                valor = input(f'Escriba la descripción que se encuentra en {criterios[eleccion]}: ')
                mostrar_stock_especifico(criterio, valor)
                continuar=int(input('¿Desea seguir buscando? \n1.Sí \n2.No \n'))
                if continuar == 1:

```

Consola-Producto final

Menú-Principal

```
Selecciona
*****
1. Administrador
2. Vendedor
3. Cerrar sesion
Ingresa la opcion deseada: 
```

Menu -Administrador

```
Bienvenido Administradore
Selecciona ----->
1. Registrar producto
2. Actualizar producto
3. Ver Inventario
4. Ver Reportes
5. Cerrar sesion
Ingresa la opcion: 
```

Agregar-items al inventario

```
Bienvenido Administradore
Selecciona ----->
1. Registrar producto
2. Actualizar producto
3. Ver Inventario
4. Ver Reportes
5. Cerrar sesion
Ingresa la opcion: 1
Ingrese la descripcion del producto: Camisa Manga larga
Ingrese la Marca del Producto: Levis
Cantidad de articulos a registrar: 300
Ingrese la talla del Producto: M
Ingrese el costo del producto: 200
Ingrese el precio de venta del producto: 300
*****
Informacion Guardada
```

Reporte de inventario



CODIGO	DESCRIPCIÓN	MARCA	CANTIDAD	TALLA	ESTADO	COSTO	PRECIO
P001	PANTALON	X	10	30	Activo	100	150
P002	CAMISA	X	8	S	Activo	40	100
P003	CHAQUETA	S	11	L	Activo	50	100
P004	PANTALON	LEVIS	0	32	Activo	50	100
P005	VESTIDOS	X	25	32	Activo	50	70
P006	GORRO	TOMMY	25	UNICA	Activo	30	50
P007	CAMISETA	H	10	M	Activo	35	70
P009	PANTALON	LEVIS	10	M	Activo	25	75
P010	CAMISA MANGA LARGA LEVIS		300	M	Activo	200	300

¿Desea seguir buscando?

1.Sí
2.No

Facturación



FACTURA_ID	FECHA	ID	NOMBRE	PRODUCTOS	Cantidad	Talla	Precio	SUBTOTAL	DESCUENTO	IVA	TOTAL
F001	20/07/2024	123456	ANDRES	P001	3	S	60	180	4.2	34.2	210
F002	20/07/2024	23654	ALEJANDRO	P005	3	S	60	180	4.2	34.2	210
F003	12/12/2024	121212	Alejo	P002	3	S	60	180	4.2	34.2	210
P004	3	S	60	P004	3	S	60	180	4.2	34.2	210
F004	20/07/2024	ANDRES	J@GMAIL.COM	P001	2	30	200	1250	250.0	190.0	1190.0
P004	5	32	100	P004	5	32	100	1250	250.0	190.0	1190.0
P007	5	M	70	P007	5	M	70	1250	250.0	190.0	1190.0
F005	21/07/2024	32456	andres	P001	8	30	200	1600	0.0	304.0	1904.0
F006	25/07/2024	3654	EDUARD	P004	3	32	100	1210	0.0	229.9	1439.9
P005	11	32	70	P005	11	32	70	1210	0.0	229.9	1439.9
P007	2	M	70	P007	2	M	70	1210	0.0	229.9	1439.9
F007	26/07/2024	321456	ANDRES	P006	5	UNICA	50	250	37.5	40.375	252.875
F008	26/07/2024	321456	CAMILA	P002	10	S	60	920	460.0	87.4	547.4
P003	4	L	80	P003	4	L	80	920	460.0	87.4	547.4
F009	30/07/2024	123456	ANDRES	P002	5	S	60	300	0.0	57.0	357.0

0 Git Graph

You, 14 hours ago Ln 122, Col 26 Spaces: 4 UTF-8 CRLF Python 3.12.3 64-bit ⚡ Prettier

