



TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE APIZACO

NOMBRE DEL PROYECTO:

MÓDULO DE VENTAS DE UNA TIENDA VIRTUAL PARA EL
INSTITUTO TECNOLÓGICO DE APIZACO.

OPCIÓN QUE CORRESPONDA:

TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL

PARA OBTENER EL TÍTULO PROFESIONAL DE:

INGENIERIA EN TECNOLOGÍAS DE LA INFORMACIÓN Y
COMUNICACIONES

PRESENTA:

JUAN VICENTE VAZQUEZ ESPINOZA

NOMBRE DEL ASESOR:

DR. JUAN RAMOS RAMOS



Instituto Tecnológico de Apizaco
Sistemas y Computación

Apizaco, Tlaxcala, **10/septiembre/2025**

Oficio No. S.C./OPV/0521/25

**ASUNTO: Liberación de proyecto
para titulación integral**

ISRAEL JORGE SÁNCHEZ SÁNCHEZ
JEFE DEL DEPTO. DE DIVISIÓN DE ESTUDIOS PROFESIONALES
P R E S E N T E.

POR ESTE MEDIO LE INFORMO QUE HA SIDO LIBERADO EL SIGUIENTE PROYECTO PARA LA TITULACIÓN INTEGRAL.

NOMBRE DEL EGRESADO:	VAZQUEZ ESPINOZA JUAN VICENTE
CARRERA:	INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES
NO. DE CONTROL:	20370974
NOMBRE DEL PROYECTO:	Módulo de ventas de una tienda virtual para el Instituto Tecnológico de Apizaco
PRODUCTO U OPCIÓN:	TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL

AGRADEZCO DE ANTEMANO SU VALIOSO APOYO EN ESTA IMPORTANTE ACTIVIDAD PARA LA FORMACIÓN PROFESIONAL DE NUESTROS EGRESADOS.

A T E N T A M E N T E

Excelencia en Educación Tecnológica.
Pensar para Servir, Servir para Triunfar®

MIQUELINA SÁNCHEZ PULIDO
JEFA DEL DEPTO. DE SISTEMAS Y COMPUTACIÓN

ccp. Coordinación de carrera
Expediente
MSP/mrol*



JUAN RAMOS RAMOS
ASESOR INTERNO



2025
Año de
**La Mujer
Indígena**

Av. Instituto Tecnológico No. 418, San Andrés Ahuashuatepec,
Municipio de Tzompantepec, Tlaxcala, Mex. C.P. 90491
Tel. (241) 4172010 Ext. 115, e-mail: sistemas@apizaco.tecnm.mx
tecnm.mx | apizaco.tecnm.mx





Instituto Tecnológico de Apizaco
División de Estudios Profesionales

Tzompantepec, Tlaxcala, **02/octubre/2025**

JUAN VICENTE VAZQUEZ ESPINOZA
EGRESADO(A) DE LA CARRERA DE INGENIERÍA
EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES
CON NÚMERO DE CONTROL 20370974.
P R E S E N T E.

Por este medio, me permito informar a usted que por aprobación de la comisión revisora asignada para valorar el trabajo que mediante la opción **TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL** y cuyo tema es: **"MÓDULO DE VENTAS DE UNA TIENDA VIRTUAL PARA EL INSTITUTO TECNOLÓGICO DE APIZACO"**, conforme a lo establecido en el procedimiento para la obtención del Título Profesional en el Instituto Tecnológico, la División de Estudios Profesionales a mi cargo le otorga la:

AUTORIZACIÓN DE IMPRESIÓN

Debiendo entregar **2 USB** del mismo, apropiadamente empastados, a la brevedad, para presentar su Acto de Recepción Profesional.

Sin otro particular por el momento me despido.

ATENTAMENTE

Excelencia en Educación Tecnológica®

Pensar para Servir, Servir para Triunfar®

CECILIA YARELY ARREGUÍN LARA
JEFA DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES



ccp. Archivo
CYAL/Gsc*



2025
Año de
La Mujer
Indígena

Av. Instituto Tecnológico No. 418, San Andrés Ahuashuatepec, Municipio de
Tzompantepec, Tlaxcala, Mex. C.P. 90491 Tel. (241) 4172010 Ext. 304. e-mail:
dep_titulacion@apizaco.tecnm.mx | apizaco.tecnm.mx



AGRADECIMIENTOS

Al Instituto Tecnológico de Apizaco, le agradezco la oportunidad de estudiar la carrera de ingeniería en tecnologías de la información y comunicaciones. Quiero expresar mi más sincero agradecimiento a los maestros que contribuyeron grandemente a mi educación con su dedicación, paciencia y atención. A mi asesor interno, Dr. Juan Ramos Ramos, quien ha sido mi guía y apoyo en este corto, pero significativo tiempo. Los consejos, recomendaciones y comentarios proporcionados me ayudaron a enfrentar y superar los desafíos y alcanzar los objetivos establecidos. Gracias al asesor externo la Lic. Miquelina Sánchez Pulido al permitirme realizar mi residencia profesional en su área de trabajo. Les agradezco por permitirme utilizar los recursos que tienen y aplicar mis conocimientos en mi residencia.

Mis palabras de agradecimiento son poca cosa en comparación con la gratitud que siento por quienes han sido parte importante e insustituible en esto, por lo que hacer la siguiente mención. A mi familia por estar a mi lado en cada paso de la vida y por el apoyo incondicional durante estos años de estudio. A comprender lo que significa estudiar para un grado como el mío. A mis amigos, por acompañarme y soportar los estrados de mi vida escolar y cotidiana. A mi profesión, que siempre me ha dado un propósito y una lección que aprender. A cada persona que he conocido, a cada experiencia que he tenido y a todo lo que he aprendido. Gracias.

RESUMEN

Por medio del presente, se expone la historia del desarrollo del Módulo de Ventas en la tienda virtual del Instituto Tecnológico de Apizaco. Durante mi residencia profesional, realizada desde enero hasta junio de 2025, realicé el proyecto en cuestión. Utilizando Django, encontré la manera de simplificar y organizar todo el proceso de las ventas: desde el registro de la venta misma, la conexión con las pasarelas de pago y la creación de los comprobantes, hasta la sincronización con el inventario además la forma de emitir los reportes detallados y asegurarse de que cada transacción sea segura. A lo largo de 22 semanas, de desarrollo en la institución, junto con mis asesores tanto internos como externos, me lancé a analizar qué me hacía falta para el sistema, a crear su estructura, a realizar la programación de interfaces amigables para el usuario, a integrar herramientas hechas por otras personas, a probar la seguridad del sistema, y al final lanzarlo.

Como resultado, se ha recibido un módulo funcional que hace que el proceso de compra en la tienda virtual se realice más fácil y, también, simplificó a la gerencia en este trabajo. Todo esto, además, está cubierto por diagramas que muestran cómo funciona el sistema desde el punto de vista técnico. A lo largo de este proceso, no solo conocí sobre la programación y la administración de proyectos, sino que también adquirí habilidades de colaboración y creatividad para problemas resueltos.

Este proyecto mejora la tienda virtual del Centro del Instituto y señala su deseo por producir profesionales punteros y futuristas.

ÍNDICE

PRELIMINARES -----	1
AGRADECIMIENTOS -----	1
RESUMEN-----	2
CAPÍTULO 1: GENERALIDADES DEL PROYECTO -----	13
Introducción. -----	13
Descripción de la empresa u organización-----	14
Misión -----	14
Visión-----	14
Valores-----	15
Puesto o área del trabajo el estudiante. -----	15
Rol del Estudiante-----	16
Problemas por resolver. -----	17
Falta de un sistema centralizado -----	17
Control Inadecuado del Inventario -----	18
No Integración con Sistemas de Pago-----	18
Objetivos -----	19
Objetivo General -----	19
Objetivos Específicos -----	19
Justificación -----	20
Beneficios al Instituto -----	21
Beneficios Educativos -----	22
CAPÍTULO 2: MARCO TEÓRICO -----	23
Fundamentos teóricos -----	23

Comercio Electrónico	24
Definición y Evolución	24
Componentes de un Sistema de E-commerce	25
Desafíos del E-commerce	26
Modelo Teórico de Adopción	27
Framework Django	28
Arquitectura MVT	29
Funcionamiento MTV de Django.	29
Ventajas de Django en E-commerce	31
JavaScript y CSS	32
Introducción a JavaScript	32
Introducción a CSS	32
Bases de Datos Relacionales	33
Selección de Motor de Base de Datos	34
Pasarelas de Pago	35
Conceptos Generales	35
Gestión de Inventarios	38
Principios de Gestión	38
Lógica de Stock	39
Seguridad Informática	42
Amenazas en E-commerce.	42
Medidas Implementadas	43
Normativo Mexicano	46
Cumplimiento Específico	46
Diseño de Interfaz de Usuario	47

Tecnologías Actuales para el desarrollo web-----	49
CAPÍTULO 3: DESARROLLO -----	51
Introducción-----	51
Propósito de la Metodología-----	52
Metodología Kanban -----	52
Implementación de Kanban-----	53
Beneficios de Kanban -----	55
Desglose de Actividades -----	56
Inicio -----	56
Inducción -----	57
Asignación del Proyecto -----	59
Recopilación de Requisitos Funcionales y No Funcionales-----	61
Análisis y Documentación del Módulo de Ventas-----	64
Requerimientos de usuario -----	66
Diseño de Arquitectura y Base de Datos-----	68
Elaboración de Historias de Usuario -----	78
Diagramas de flujo-----	85
Flujo de Gestión del Administrador -----	85
Flujo de interacción del Usuario -----	86
Casos de uso -----	89
Configuración del Ambiente de Desarrollo-----	100
Creación de aplicaciones: -----	102
App Ecommerce -----	102
App Orders-----	111
App Social -----	124

Desarrollo del Core de Ventas-----	132
Integración con Sistema de Inventarios-----	134
Desarrollo de Interfaces de Usuario-----	136
Integración de Pasarelas de Pago-----	137
Sistema de Notificaciones-----	139
Flujo del módulo de Ventas-----	141
CAPÍTULO: 4 RESULTADOS-----	143
Django en el módulo de ventas-----	143
JavaScript y CSS en el Módulo de Ventas-----	144
SQLite en el Módulo de Ventas-----	145
Registro y Administración Eficiente de Ventas-----	146
Integración Segura con Pasarelas de Pago-----	149
Sistema de Comprobantes-----	151
Seguridad y Privacidad de la Información-----	153
Administración y Gestión del Sistema (Panel de Administración)-----	155
Descripción De la Interfaz de Usuario-----	158
Usuario-----	158
Inicio de sesión-----	158
Carrito-----	158
Checkout-----	160
Payments-----	161
Métodos de pago-----	162
Panel del usuario.-----	163
Mi cuenta/Notificaciones-----	164
Mi cuenta/Mis órdenes de compra-----	165

Administrador -----	166
Inicio sesión-----	166
Dashboard principal -----	167
Sección Notificaciones -----	168
Sección Ordenes/ validar ordenes -----	169
Sección Ordenes/ ordenes validadas -----	170
Sección Ordenes/ ordenes canceladas-----	171
Manual de Usuario-----	172
CONCLUSIONES -----	173
Competencias Desarrolladas -----	175
Competencias Técnicas -----	175
Competencias Blandas -----	176
Competencias Profesionales -----	177
REFERENCIAS-----	178
ANEXOS-----	179
Anexo A: Pruebas -----	179
Anexo B: Manual de Usuario -----	179

INDICE FIGURAS

Figura 1 Flujo de trabajo en la arquitectura MTV (Model-Template-View) de Djangow ...	30
Figura 2 Flujo de una compra desde que el usuario la inicia hasta su confirmación o cancelación.....	41
Figura 3 Diagrama que ilustra el flujo de un diagrama de secuencia seguro que atienda la solicitud aceptada proveniente del usuario, a través de medidas de protección de datos y de sistema	45
Figura 4 Diagrama que representa la estructura de un template HTML organizado en tres secciones principales: Header, Main y Footer	50
Figura 5 Repositorio en GitHub.....	54
Figura 6 se presenta un tablero Kanban configurado en la plataforma Trello, el cual organiza las tareas del proyecto en diferentes columnas: Backlog, En Desarrollo, Pausado y Concluido.....	60
Figura 7 Diagrama de casos de uso del sistema para dos actores: Administrador y Cliente. El administrador realiza funciones de gestión y supervisión, Por su parte, el cliente interactúa con el sistema para explorar productos, realizar compras y solicitar cambios.....	65
Figura 8 Estructura de la base de datos del sistema.....	69
Figura 9. Diagrama_flujo del Administrador	86
Figura 10 Diagrama_flujo Usuario.	88
Figura 11 Proceso_Autenticación.....	90
Figura 12 Proceso_Mis Ordenes.....	92
Figura 13 Proceso_Checkout.	93
Figura 14 Proceso_Pago	95
Figura 15 Panel_admin	96
Figura 16 Panel_usuario.....	98
Figura 17 Notificacion	99
Figura 18 instalación de Python.....	100
Figura 19 Creación de un entorno virtual	100
Figura 20 Instalación de bibliotecas.....	101
Figura 21 Configuración SQLite	102

Figura 22 Ecommerce _asgi.py.....	104
Figura 23 Ecommerce _context_processors.py.....	105
Figura 24 Ecommerce _settings.py	106
Figura 25 Ecommerce _URLS.py.....	108
Figura 26 Ecommerce _Views.py.....	109
Figura 27 Ecommerce _wsgi.py	110
Figura 28 Orders _wsgi.py.....	112
Figura 29 Orders _ admin.py	114
Figura 30 Orders _ app.py	115
Figura 31 Orders _ forms.py.....	117
Figura 32 Orders _ models.py.....	119
Figura 33 Orders _ tasks.py	120
Figura 34 Orders _ urls.py	121
Figura 35 Orders _ urls.py	124
Figura 36 Social _ admin.py	125
Figura 37 Social _ admin.py	127
Figura 38 Social _ admin.py	128
Figura 39 Social _ admin.py	129
Figura 40 Social _ admin.py	130
Figura 41 Social _ admin.py	131
Figura 42 Flujo del proceso de venta.....	134
Figura 43 Diagrama de Flujo - Integración con Inventarios.....	135
Figura 44 Diagrama de Navegación: navegación del usuario a través de las interfaces.....	137
Figura 45 Diagrama de Secuencia - Proceso de Pagos	139
Figura 46 Diagrama de Flujo – Notificaciones por Correo electrónico	140
Figura 47 Diagrama de Flujo – Módulo_ventas	142
Figura 48 Diagrama que muestra el modelo relacional, donde un usuario pueda generar una orden	146
Figura 49 Vista place_order	147
Figura 50 formulario (OrderForm)	148
Figura 51 Modelo Order: Función0.....	150

Figura 52 Vista Payment:.....	150
Figura 53 UserProfile.....	152
Figura 54 Decorador @login_required	153
Figura 55 Decorador @staff_member_required	154
Figura 56 Gestión de Pedidos	156
Figura 57 Comprobantes.....	157
Figura 58 Inicio_sesión_usuario:	158
Figura 59 Interfaz de carrito.	159
Figura 60 Pantalla de checkout.....	160
Figura 61 pantalla Payments.....	161
Figura 62 Métodos de pago.....	162
Figura 63 Panel “Mi cuenta”	163
Figura 64 Mi cuenta/Notificaciones.....	164
Figura 65 Mi cuenta/Mis órdenes de compra	165
Figura 66 Incio_Sesión administradores.....	166
Figura 67 Dashboard administrador.....	167
Figura 68 Notificaciones_admin	168
Figura 69 Ordenes/validar_ordenes/admin.....	169
Figura 70 Ordenes/Ordenes_Validadas /admin.....	170
Figura 71 Ordenes/Ordenes_canceladas/admin.....	171
Figura 72 M_Inicio_sesión_usuario.....	180
Figura 73 M_Carrito	181
Figura 74 M_ Checkout.	182
Figura 75 M_ Paymentes.	183
Figura 76 M_Mi cuenta/Notificaciones	184
Figura 77 M_Mi cuenta/Mis órdenes de compra.....	185
Figura 78 M_Inicio_admin.	186
Figura 79 M_Panel_admin.....	187
Figura 80 M_Notificaciones_admin.....	188
Figura 81 M_Ordenes/validar_ordenes/admin.	189
Figura 82 M_Ordenes/Ordenes_validadas/admin.	190

Figura 83 M_Ordenes/Ordenes_canceladas /admin	191
Figura 84 M_Reportes/admin.	192

INDICE TABLAS

Tabla 1 Requisitos funcionales del módulo de ventas, los cuales definen las capacidades esenciales del sistema	62
Tabla 2 Requisitos no funcionales del módulo de ventas, los cuales establecen criterios de calidad	63
Tabla 3 Historias de Usuario: Registrar Nueva Venta	79
Tabla 4 Historias de Usuario: Consultar Historial de Ventas	79
Tabla 5: Historias de Usuario: Cancelar Venta	80
Tabla 6: Historias de Usuario: Generar Comprobante	80
Tabla 7: Historias de Usuario: Gestionar Estado de Pagos	81
Tabla 8: Historias de Usuario: Actualizar Stock Automáticamente	81
Tabla 9: Historias de Usuario: Verificar Disponibilidad de Productos	82
Tabla 10: Historias de Usuario: Alertas de Stock Bajo	82
Tabla 11: Historias de Usuario: Integrar Pasarelas de Pago	83
Tabla 12: Historias de Usuario: Notificar Estado de Pagos	83
Tabla 13: Historias de Usuario: Implementar Seguridad de Datos	84
Tabla 14: Historias de Usuario: Restringir Acceso a Datos	84

INDICE TABLAS

Tabla 1 Requisitos funcionales del módulo de ventas, los cuales definen las capacidades esenciales del sistema	62
Tabla 2 Requisitos no funcionales del módulo de ventas, los cuales establecen criterios de calidad.	63
Tabla 3 Historias de Usuario: Registrar Nueva Venta	79
Tabla 4 Historias de Usuario: Consultar Historial de Ventas	79
Tabla 5: Historias de Usuario: Cancelar Venta	80
Tabla 6: Historias de Usuario: Generar Comprobante	80
Tabla 7: Historias de Usuario: Gestionar Estado de Pagos.....	81
Tabla 8: Historias de Usuario: Actualizar Stock Automáticamente	81
Tabla 9: Historias de Usuario: Verificar Disponibilidad de Productos	82
Tabla 10: Historias de Usuario: Alertas de Stock Bajo	82
Tabla 11: Historias de Usuario: Integrar Pasarelas de Pago.....	83
Tabla 12: Historias de Usuario: Notificar Estado de Pagos	83
Tabla 13: Historias de Usuario: Implementar Seguridad de Datos	84
Tabla 14: Historias de Usuario: Restringir Acceso a Datos	84

CAPÍTULO 1: GENERALIDADES DEL PROYECTO

Introducción.

El comercio electrónico ha mejorado la forma en que las instituciones educativas y otros servicios controlan sus servicios y productos a través de plataformas capacitadas, seguras y efectivas. Es dentro de esta plataforma que el Instituto Tecnológico de Apizaco en la estructura mexicana del Tecnológico Nacional de México decidió instalar una tienda virtual. El módulo de ventas es el producto principal de la tienda virtual de las operaciones de la residencia profesional del semestre enero-junio de 2025.

El módulo de ventas implementa un sistema de programación Django y complementos Tecnología para la grabación y control de ventas, despliegue de gateway de pago a través de sistemas de transacciones financieras, esquemas de designación de comprobantes, destino de la creación, sincronización de la mercadería en existencia, producción de informes y grabaciones actualizados, y certificación de seguridad en forma de transacciones. Como respuesta a los problemas encontrados por el comercio electrónico, la administración de inventarios, la conciencia fiscal y la protección de datos financieros. Un trabajo de reflexión que incluye una detallada y crítica descripción del desarrollo del módulo de ventas, su justificación, lugar metodológico, entre otras cosas, está dentro del reporte, los diagramas técnicos dentro de las aplicaciones (Ecommerce, Orders, Carousel y Social etc.) de clases, flujo, rutas que llevan a la arquitectura del sistema de vivienda profesional.

Descripción de la empresa u organización

Tecnológico Nacional de México / Instituto Tecnológico de Apizaco.

El Instituto Tecnológico de Apizaco, es una institución de educación superior tecnológica perteneciente al Tecnológico Nacional de México se ha consolidado como una de las principales instituciones educativas de la región, ofreciendo 12 programas de ingeniería con diversas especialidades. el Instituto tiene como misión formar profesionistas competitivos con alta capacidad técnica y humanística.

Domicilio

El Instituto Tecnológico de Apizaco se ubica en la **Av. Instituto Tecnológico No. 418, San Andrés Ahuashuatepec, Tzompantepec, Tlaxcala, C.P. 90491.**

Misión

Su **misión** es formar profesionistas líderes, emprendedores, e innovadores, comprometidos con el desarrollo económico y social sostenible.

Visión

Su **visión** es ser una institución de vanguardia, formadora de ciudadanos globales con competencias interculturales.

Valores

Sus **valores** incluyen:

Honestidad: se demuestra día a día a través de la transparencia con la ejecución de las actividades laborales cotidianas.

Responsabilidad: es la actitud que caracteriza al personal en el cumplimiento de su tarea con eficacia y eficiencia en beneficio de la comunidad.

Lealtad: es la actitud que caracteriza al personal del Instituto Tecnológico de Apizaco, aplica sus conocimientos y habilidades en el logro de los metas institucionales reflejados en el lema: “Pensar para servir, servir para triunfar”.

Puesto o área del trabajo el estudiante.

Este proyecto se desarrolló en el Departamento de Sistemas y Computación junto con el apoyo del Departamento de Comunicación y difusión, área correspondiente a la encargada en proporcionar los requerimientos funcionales y los productos necesarios y el soporte operativo esencial para la creación del módulo de ventas, lo cual permitió alinear el desarrollo y diseño de una solución técnica con las necesidades institucionales, garantizando que ese módulo creado refleje los objetivos institucionales y estratégicos.

Rol del Estudiante

La participación y el rol del estudiante en el desarrollo del módulo de ventas giraron en torno a un conjunto de funciones que contribuyeron al diseño y desarrollo de una solución sólida, y apegado a las necesidades institucionales. Algunas de las funciones desempeñadas, comprende: analizar requerimientos para el módulo de ventas, lo cual incluye recopilar y evaluar los insumos funcionales, evaluar su relevancia; diseñar e implementar soluciones técnicas a través del framework Django que incluye la estructura de modelos como Order y vistas personalizadas, integrar pasarelas de pagos seguros y módulos de inventario sincronizados con bases de datos para la gestión, tanto como desarrollar interfaces de usuario y paneles de gestión administrativa mediante el uso del framework Bootstrap, al objeto de optimizar la interfaz, pruebas de seguridad a través de decoradores como `@login_required` y `@csrf_protect` y pruebas de rendimiento, para determinar la viabilidad de los componentes desarrollados; y documentar procesos significativamente, a través de reportes técnicos que facilitan la transferencia de conocimientos.

Problemas por resolver.

El desarrollo del módulo de ventas para la futura tienda en línea para el Instituto Tecnológico de Apizaco se ha encontrado con varios problemas significativos que deben abordarse para asegurar un sistema eficiente y basado en un enfoque seguro y las necesidades dadas institución. Los problemas, clasificados de más a menos significativos, siguen siendo esenciales para el diseño general de la plataforma de comercio electrónico que respalde el logro de los objetivos institucionales. A continuación, se describen de manera detallada:

Falta de un sistema centralizado

La falta de un sistema centralizado para procesar las transacciones de venta es otro desafío significativo para la operación de la tienda virtual. En este sentido, sin un módulo automático, las ventas tendrán que confiar en procesos manuales. La carga de trabajo adicional podría exponer las transacciones al error humano. Es probable que también sea difícil hacer un seguimiento y verificar las transacciones realizadas. Esto reduciría la eficiencia administrativa y dañaría la satisfacción del cliente. Como parte del desarrollo de este proyecto en el Instituto Tecnológico de Apizaco, el Módulo de Ventas busca solucionar estas limitaciones, ofreciendo una solución tecnológica que garantice honestidad al transparentar cada transacción, responsabilidad al asegurar un manejo preciso de datos y lealtad al fortalecer el vínculo con la comunidad educativa mediante un sistema accesible y eficiente.

Control Inadecuado del Inventario

La ausencia de un mecanismo eficiente para administrar el inventario podría provocar discrepancias en la disponibilidad de productos, como la venta de artículos que ya no están en stock. Esto generaría insatisfacción entre los clientes y complicaciones operativas para el instituto. Un módulo de ventas debe estar vinculado de manera básica al módulo de inventario para actualizar el stock en tiempo real tras cada transacción, asegurando que los productos ofrecidos en el carrito estén disponibles.

No Integración con Sistemas de Pago

Sin pasarelas de pago compatibles, la tienda se verá limitada en términos de opciones de pago, lo cual puede desincentivar a los compradores. Aunque los pagos directos son comunes, y los compradores de la tienda pueden realizar transferencias bancarias, los usuarios esperan métodos seguros y accesibles. Esta limitación hace que la tienda se vea menos profesional y sus opciones no sean versátiles.

Objetivos

Objetivo General

Elaborar e instaurar un Módulo de Ventas para la Tienda Online del Instituto Tecnológico de Apizaco que permita llevar a cabo de forma segura y eficiente los procesos de comercialización. Este debe contar con: registro de ventas, pasarelas de pago, comprobantes, sincronización con existencias y automatización de cambio. Todo el proceso de comercialización debe ser amigable en la experiencia y facilidad para el usuario y las operaciones administrativas.

Objetivos Específicos

- Registrar y administrar todas las ventas realizadas en la tienda virtual, asegurando trazabilidad y precisión.
- Integrar pasarelas de pago seguras (transferencia, apartado) para garantizar transacciones confiables.
- Implementar un sistema de generación de comprobantes
- Sincronice el módulo de ventas con el de inventarios para actualizar las existencias automáticamente.
- Informes detallados del estado de las ventas y los pagos para agilizar la toma de decisiones.
- Proteger y mantener la privacidad de la información financiera mediante encriptación y validaciones.

Justificación

Partir de que el e-commerce permite al instituto mantener contacto con algunos ex_alumnos ofreciendo productos, souvenir. El comercio electrónico ya se ha establecido dentro de las estrategias clave y las funcionalidades de la dirección en términos de su gestión para cualquier gestor o institución educativa ya que aumenta su alcance, ahorra algunos recursos y otorga capacidad de rentabilidad en productos y servicios. Para el Instituto Tecnológico de Apizaco, la creación de la tienda virtual es una oportunidad de presencia en línea, para fomentar programas institucionales y para adquirir recursos generados por el comercio para fines académicos. El módulo de ventas es un espacio para transacciones individuales que forma un componente autónomo de un sistema de comercio electrónico y es responsable del procesamiento de transacciones y la creación de registros relacionados. El presente trabajo consiste en proponer un diseño de este módulo.

El módulo de ventas abordará esta limitante al:

Centralizar el registro de ventas en un sistema unificado mediante la aplicación Orders con el modelo Order, garantizando un manejo transparente de las transacciones; sincronizar el módulo de ventas con el módulo de inventarios y el carrito a través de señales en OrderProduct y CartItem para actualizar los niveles de stock en tiempo real y reflejar los productos seleccionados; integrar pasarelas de pago seguras (transferencia, apartado), protegidas con decoradores como @csrf_protect y alineadas con las normativas del CFDI; generar comprobantes electrónicos utilizando la biblioteca ReportLab, que incluyen datos como order_number y montos para asegurar la trazabilidad; proporcionar reportes detallados a través de la vista sales_report, ofreciendo análisis de tendencias y métricas clave para apoyar la toma de decisiones con claridad; e implementar medidas de seguridad robustas,

incluyendo SSL/TLS, validaciones de entrada y decoradores como @login_required, para proteger los datos sensibles conforme a la LFPDPPP.

Beneficios al Instituto

El módulo de ventas será una adición clave al Instituto Tecnológico de Apizaco; será extremadamente ventajoso desde la perspectiva estratégica y otorgará al Instituto grandes ventajas en la gestión de una tienda virtual que cumpliría con los objetivos institucionales. Algunos de estos beneficios incluyen:

1. Eficiencia en la administración; la venta se convierte en un proceso automático, evitando errores humanos y agilizando la eficiencia de los gerentes de ventas para manejar varias transacciones a la vez.
2. Confianza del cliente; varias pasarelas de pago y medidas de seguridad adicionales pueden aplicarse para mejorar la seguridad y la experiencia del cliente, fomentando la fidelidad del consumidor.
3. Planificación estratégica; la capacidad de recibir datos analíticos le brindará una visión de lo que funciona y le permitirá planificar lo que necesita mejorar.
4. Viabilidad; la tienda en línea que reinvierte las ganancias de la primera inversión en el futuro puede aumentar la financiación para más proyectos y actualizaciones.

Beneficios Educativos

El módulo de ventas no es solo una oportunidad formativa única, sino también un proyecto donde los estudiantes del Instituto Tecnológico de Apizaco pueden aplicar sus conocimientos en la práctica y adquirir habilidades claves en los siguientes ámbitos:

- La programación web: pueden diseñar la interfaz de usuario y desarrollar la lógica que subyace a la operación de la tienda virtual
- Administración de bases de datos: adquieren la capacidad de procesar la información en los registros de ventas, el inventario y los ‘carritos de compra’, manteniéndola íntegra y eficiente.
- Ciberseguridad: aprenden las habilidades que aplicarán para asegurar transacciones seguras y proteger la información confidencial.
- Gerencia de proyectos: pueden planificar y realizar un proyecto del área de tecnología que tenga resultados tangibles y de alta calidad.

Tal experiencia no solo mejorará su educación académica, sino que también les brindará habilidades críticas para el éxito profesional en los campos de comercio electrónico y software

CAPÍTULO 2: MARCO TEÓRICO

Fundamentos teóricos

En resumen, el Módulo de Ventas del Software Propuesto para la tienda virtual del Instituto Tecnológico de Apizaco, en esencia, se ha basado en los referentes teóricos abordados en las líneas precedentes. En su conjunto, el marco conceptual ha tratado los aspectos clave acerca de los principios más relevantes en los ámbitos técnicos y teóricos del comercio electrónico, frameworks de desarrollo web, bases de datos, pasarelas de pago, seguridad informática, inventarios, interfaz de usuario y regulaciones. Los conceptos abordados en esta área se han implementado como fundamentos y justificación para la selección de decisiones técnicas para hacer el desarrollo del proyecto robusto, escalable y profesional.

El comercio electrónico se consolida como un modelo de negocio estratégico, permitiendo al Instituto diversificar sus canales de venta, optimiza la experiencia del usuario y además garantizar una fuente de ingreso sostenible que de por sí contribuya a la continuidad de la comunidad estudiantil. Por lo que, los conceptos abordados en esta área se han traducido en un desarrollo robusto, escalable y en sí profesional, lo que se basa incluso en la adopción de herramientas innovadoras de desarrollo como Django para la estructura backend y Bootstrap 5 para las interfaces responsivas, transmitiendo un enfoque renovador notable, alineado a su ideal de seguir formando profesionistas innovadores.

Comercio Electrónico

Definición y Evolución

El término comercio electrónico, que se abrevia e-commerce, se refiere a la negociación en la compra y venta de bienes, servicios e información a través de la tecnología digital; a menudo implica la conexión en línea de consumidores y vendedores por Internet. Laudon y Traver (2021) mencionan, que, en su opinión, hay cuatro tipos principales de e-commerce.

B2B, que es Business to Business, negocio a negocio en español. B2B, es en el que una empresa compra los productos de otra empresa. B2C, se entiende Business to Consumer, es el comercio entre empresa y consumidor. C2C se entiende por Consumer to Consumer, es el comercio donde las transacciones se realizan directamente entre individuos sin intervención de intermediarios. Y hay algo llamado M-commerce, es una compra, venta de bienes, y servicios a través de dispositivos móviles, tales como teléfonos inteligentes o tabletas.

Ha pasado los años, el comercio electrónico se ha expandido con la ayuda de tecnologías innovadoras como la banda ancha, los dispositivos móviles, y las pasarelas de pago seguras. Según AMVO (2023), en México, en el año 2023, un aumento del 27% en el comercio electrónico, donde el valor de las ventas en línea es de \$ 528 bn. Entonces la apertura de la tienda en línea fue justificada por el deseo de mejorar la presencia en la web del Instituto y la tienda abierta a granel mayor mercado en aumento.

Componentes de un Sistema de E-commerce

Un sistema de comercio electrónico consta de varios componentes interrelacionados (Chaffey, 2022):

- **Carrito de la compra:** Permite al usuario seleccionar los productos antes de finalizar la compra y comprarlos de una forma más cómoda, haciendo que la experiencia sea mucho más simple.
- **Buscador efectivo:** Buscador que permita encontrar producto de una forma mucho más sencilla.
- **Catálogo de Productos:** Una base de datos que almacena la información de productos (nombre, precio, stock).
- **Pasarelas de Pago:** Facilitan transacciones seguras.
- **Gestión de Pedidos:** Registra transacciones.
- **Inventarios:** Actualiza el stock en tiempo real.
- **Comprobantes:** Genera comprobantes conforme a normativas.
- **Seguridad:** Protege datos sensibles de los usuarios.

El módulo de ventas integra algunos de estos componentes, con un enfoque en la gestión de pedidos, pagos, inventarios, optimizando la experiencia del usuario.

Desafíos del E-commerce

El comercio electrónico sufre múltiples dificultades técnicas, operativas y normativas (Turban et al., 2021) que necesitan soluciones sólidas para asegurar a la plataforma el éxito. Los permisos junto a los inconvenientes mencionados anteriormente, ampliar información de fuentes en línea relevantes para obtener una visión generalizada, por ejemplo:

- **Seguridad.** La protección contra los delincuentes de los ordenadores es vital en el comercio electrónico, ya que los delincuentes de los ordenadores cibernéticos amenazan continuamente con inyecciones de SQL (SQLi), Script of Cross-Site (XSS), y el conjunto del robot y los envíos de denegación (DDoS). Para hacer frente a este problema, el proyecto ha tomado medidas como: Validaciones estrictas con Django, cifrado de datos con HTTPS, autenticación multi-paso, y WAF.
- Los problemas de **usabilidad** también son significativos. Un usuario que no es familiar con la tecnología debe ser capaz de acceder a la interfaz. Según el Baymard Institute (2023), 70 % de todos los carritos de compra en línea en un estudio realizado por el Instituto Baymard terminaron abandonándose porque la interfaz era propensa a formularios complicados o poco claros procesos de pago.
- **Escalabilidad:** Las plataformas de comercio electrónico deberían también ser capaces de ser escalables para poder afrontar picos de tráfico máximo frecuentes, como el Black Friday o ventas masivas. De acuerdo con un análisis de Cloudflare, el tráfico de sitios de e-commerce puede incrementar hasta un 400% durante estos eventos (Cloudflare, 2023). Por lo anterior, se integraron servicios en la nube como AWS o Azure y herramientas de cacheo, como Redis, en la arquitectura modular del

módulo de ventas, desarrollado con Django, para permitir manejar millones de usuarios simultáneos sin sacrificar rendimiento.

- **Cumplimiento Normativo:** Otras legislaciones son también necesarias; por ejemplo, la LFPDPPP y GDPR son necesarias para anexarse a la legislación fiscal y de protección de datos de usuarios nacionales e internacionales respectivamente.
- Integración con **Sistemas Externos:** La integración con pasarelas de pago, son igualmente complicadas, así como sistemas de inventario en tiempo real.

Estos desafíos se abordaron utilizando el framework Django, que permitió una arquitectura segura y escalable, junto con validaciones estrictas en formularios y transacciones, pruebas de seguridad y el cumplimiento de las normativas internacionales.

Modelo Teórico de Adopción

En su clásico Technology Acceptance Model (TAM) de Davis (1989) explica la aceptación de tecnologías por parte de los usuarios basada en la percepción de utilidad y facilidad de uso. Justificado por las compras reales de módulos, respectivamente, esta base teórica podría mostrarse en base a la conveniencia del acceso a componentes y la intuición de la interfaz, principalmente para estudiantes y la comunidad.

Tecnologías para el desarrollo.

Framework Django

Django es un framework de desarrollo web de alto nivel basado en Python, diseñado para crear aplicaciones seguras, escalables y mantenibles (Holovaty & Kaplan-Moss, 2022).

Su filosofía DRY (Don't Repeat Yourself) y sus características integradas, como:

- ORM (Object-Relational Mapping) el cual mapea objetos Python a tablas de bases de datos.
- MVC/MTV (Model Template View) divide la lógica de negocio, la presentación, el middleware de seguridad y el panel administrativo.
- CSRF (Cross Site Request Forgery protection), y el Admin Panel es una interfaz automática para gestionar los datos.
- Sistema de Enrutamiento, Django facilita la creación de URLs claras y organizadas, permitiendo acceder a diferentes partes del sistema, como el listado de productos o los reportes, de forma sencilla e intuitiva.
- Reportes Automáticos, Django facilita la generación de formatos como PDF, Word y Excel, utilizando bibliotecas como ReportLab.

Todo esto lo hacen ideal para proyectos de e-commerce donde la robustez, escalabilidad y seguridad son fundamentales para el éxito. En este proyecto, se utilizara Django para desarrollar las aplicaciones principales del módulo de ventas: Ecommerce, Orders y la sincronización con otras como Social:

Arquitectura MVT

Las siglas en ingles MVT corresponden a Model (Modelo), Template (Plantilla) y View (Vista) Figura 1.

- Model: Maneja todo lo relacionado con la información, esto incluye como acceder a esta, la validación, relación entre los datos y su comportamiento. Define la estructura de datos (ej. 'Order ', 'Payment ').
- Template: Gestiona la presentación (HTML con Jinja2).
- View: Es un enlace entre el modelo y el template. Decide qué información será mostrada y por cual template. Contiene la lógica de negocio (ej. 'placeorder ').

Esta arquitectura permitirá separar la lógica de negocio del módulo de ventas (e. g., validación de pagos) de su presentación (e.g., interfaz de usuario), facilitando el mantenimiento y la escalabilidad.

Funcionamiento MTV de Django.

Al momento de hacer click en un enlace o escribir una dirección, lo primero que se accede es al mapa de URLs (también conocido como URL map o URL conf), en este archivo cada ruta está asociado con una view, si se necesita algún dato se solicitara este a model el cual a su vez generara la consulta a la base de datos, cuando los datos han sido traídos estos son enviados al template que contiene la lógica de presentación para estos. Luego de "pintar" la página esta se enviá al cliente (navegador) que hizo la solicitud (Figura 1).

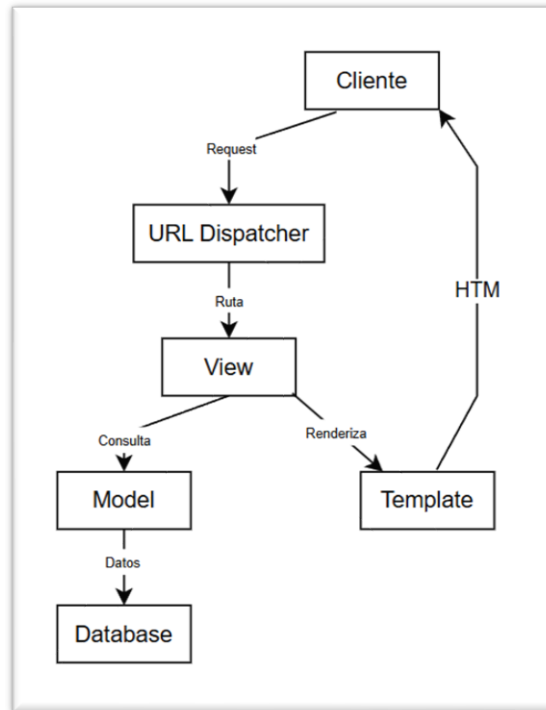


Figura 1 Flujo de trabajo en la arquitectura MTV (Model-Template-View) de Django

Ventajas de Django en E-commerce

Django es un framework de alto nivel basado en Python que se utiliza comúnmente para proyectos E-commerce debido a su robustez, seguridad y escalabilidad (Django Project, 2023). Para este proyecto, varias de las ventajas que Django son:

- **Seguridad incorporada:** Django viene con varios mecanismos incorporados para protegerse contra ataques comunes como Cross-Site Request Forgery y Cross-Site scripting. También viene con un ORM incorporado que proporciona defensas contra inyecciones SQL.
- **Escalable:** Django fue diseñado teniendo en cuenta la escalabilidad. Se puede escalar vertical y horizontalmente y escalar horizontal significa que puede manejar los procesos como una lista de tareas, por lo que el servidor no estará sobrecargado. Incorpora una amplia gama de soporte de base de datos incluido.
- **Modular:** Django no es sólo un solo proyecto sino un conjunto de varios proyectos. Esto se divide en una aplicación que puede ser utilizada dentro de su proyecto o en un usuario fuera del proyecto. Esto consume un desarrollo estructurado.
- **Comunidad:** la comunidad es el aspecto más esencial de cualquier marco web. Django también es apoyado por una comunidad muy fuerte que puede ayudarlo en sus proyectos. Como está documentado, por lo que también puedes encontrar pilas en Internet.

En conjunto, estas características hacen de Django una excelente elección para proyectos de e-commerce que requieren fiabilidad, rapidez en el desarrollo, seguridad y escalabilidad.

JavaScript y CSS

Introducción a JavaScript

JavaScript es un lenguaje de programación de alto nivel interpretado y orientado a objetos desarrollado por Brendan Eich en 1995. Este es un lenguaje que hace posible el comportamiento de una página web ya que estas son algo más que un texto estático si no que permiten al usuario interactuar. JavaScript contiene una biblioteca estándar de objetos, como Array, Date y Math, y un conjunto básico de elementos del lenguaje como operadores, estructuras de control y declaraciones.

Como elemento del ecosistema web, JavaScript se ejecuta del lado del cliente a través de los navegadores. Por ende, permite desde la manipulación del DOM y la carga de contenido sin recargar la página completa hasta la gestión de eventos para optimizar la experiencia de los usuarios y la integración con API para funcionalidades avanzadas. Otras de sus funcionalidades se suman como bibliotecas, por ejemplo, jQuery, que simplifica tareas difíciles como animaciones y manipulaciones de elementos, y frameworks modernos como React. No obstante, en proyectos como el Módulo de Ventas, se destaca por el código propio para mejorar el rendimiento y la flexibilidad. Por su parte, esta tecnología permite la innovación a través de interfaces interactivas y eficientes.

Introducción a CSS

CSS que es la abreviatura de cascading style sheet que significa en español hojas de estilo en cascadas es un lenguaje de estilo que se usa para describir la apariencia y el formato de su documento HTML. Su propósito principal es separar el contenido de la estructura del

documento HTML de la apariencia de diseño, lo que permite a los desarrolladores crear sitios web atractivos y visualmente agradables a través de la elección de colores, fuentes, esquema y animación, entre otras cosas.

CSS3 introdujo características más avanzadas como las media queries para diseños responsivos, el flexbox y grid para diseños flexibles y las transiciones para efectos dinámicos, con estas características, resulta más sencillo crear un sitio para dispositivos móviles..

Bases de Datos Relacionales

La base de datos relacional modela los datos en tablas, que también se llaman entidades, y están relacionadas entre sí. Esto se logra mediante claves primarias, que son identificadores únicos por tabla, y claves foráneas, que son referencias a claves primarias de otra tabla.

Esto permite estructurar los datos de una manera significativa, preservar la integridad referencial y realizar consultas complejas utilizando SQL (Elmasri & Navathe, 2020). En el módulo de ventas, se crea o implementa el modelo relacional para describir adecuadamente las siguientes cinco entidades clave: **pedidos, pago, producto, inventario y usuario**.

- Un pedido con un usuario y varios productos
- Un pago relacionado directamente con un pedido para reflejar el método de pago, el estado del pago y el comprobante.
- Producto que tiene nombre, categoría, precio, disponibilidad y variante.

- Inventarios que se controlan por disponibilidad por producto y variante.
- Un usuario con su pedido.

El diagrama ERD de Entidad-Relación Figura 2, muestra la relación lógica entre las entidades y da una idea de cómo fluye la información. También asegura que todas las operaciones en la DB sigan los principios de normalización adecuados.

Selección de Motor de Base de Datos

En desarrollo, se utilizó **SQLite** por su simplicidad: SQLite es una herramienta de software libre, que permite almacenar información en dispositivos empujados de una forma sencilla, eficaz, potente, rápida y en equipos con pocas capacidades de hardware, como puede ser una PDA o un teléfono celular. SQLite implementa el estándar SQL92 y también agrega extensiones que facilitan su uso en cualquier ambiente de desarrollo. Esto permite que SQLite soporte desde las consultas más básicas hasta las más complejas del lenguaje SQL, y lo más importante es que se puede usar tanto en dispositivos móviles como en sistemas de escritorio, sin necesidad de realizar procesos complejos de importación y exportación de datos, ya que existe compatibilidad al 100% entre las diversas plataformas disponibles, haciendo que la portabilidad entre dispositivos y plataformas sea transparente.

Pasarelas de Pago

Conceptos Generales

Una pasarela de pagos es una plataforma tecnológica automatizada que sirve como intermediaria de las transacciones financieras electrónicas. Las pasarelas de pagos para empresas, independientemente del modo de negociación, les brinda la posibilidad de efectuar, aceptar, tramitar y controlar distintos métodos de pago por remisión de crédito, crédito, carteras virtuales y demás de forma segura.

La pasarela de pagos conecta al cliente, la empresa y sus instituciones financieras en una sola plataforma, y generalmente cobra una comisión por cada transacción procesada (Schneider, 2021).

El proyecto integró una serie de diferentes métodos de pago a través de una pasarela para permitir que los usuarios pudieran realizar sus compras de forma sencilla, rápido y segura. Esta integración fue fundamental para mejorar la experiencia del usuario, así como facilitar un flujo de compra suave y fiable. Las capacidades clave integradas incluyeron:

- Soporte para varios métodos de pago, lo que incluyó: transferencias bancarias, a la vez que se posibilitaban otros métodos, como pagos en efectivo, que se requerían según las limitaciones o preferencias del usuario;
- Confirmación automática de transacciones exitosas con generación de notificaciones y cambio de estatus de la orden;
- Flujos de comprobantes, donde los pagos posteriores debían ser confirmados por los usuarios con comprobantes como el comprobante de transferencia.

- Ordenes automáticas que se cancelan cuando no se cumplen las condiciones que las rigen dentro del límite de tiempo establecido, por ejemplo, 24 horas para subir el comprobante en pagos por transferencia bancaria, o 72 horas para validar un orden bajo el formato de apartado.
- Notificaciones por correo electrónico que informan a los usuarios y a las partes interesadas, usuario y administrador, sobre el estado de las compras y confirmaciones.
- Procesamiento seguro y encriptado de la información de pago en un base de datos cumpliendo con las mejores prácticas y pintura de contraseñas para evitar riesgos a la privacidad de los datos de los clientes.

Gracias a esta integración, se logró una solución de pagos que facilita las compras en línea.

En el caso del sistema considerado se contemplan dos métodos principales

- **Transferencia de Banco:** esta opción implica realizar un depósito proveniente de su cuenta bancaria en forma manual en el csv. Una vez creada la orden, el usuario tiene la obligación de cargar una confirmación de transferencia, que se lleva a cabo dentro de las 24 horas siguientes. A continuación, el sistema verifica la información y confirma la transacción. Si el usuario no entrega el recibo en el plazo convenido, el pedido se cancela automáticamente
- **Apartado:** esta opción implica al usuario hacer una reserva en el producto y el pago se confirma en la tienda en persona. Esta opción es más adecuada para aquellos que consideran apropiado pagar a tienda. Cabe señalar los métodos de transacción

deberán ser flexibles y seguros. Por un lado, esto hace que la compra y todo el proceso sea más conveniente para los usuarios, por otro lado, agrega sentido a este sistema en términos de transferencia de control.

Seguridad en Pagos

Para asegurar la protección de los datos y transacciones de los usuarios, se han implementado fuertes medidas de seguridad. Entre ellas: “” GENERAL””

- Cifrado SSL/TLS de todo el tráfico entre el navegador y el servidor del usuario.

De esa manera, se mantendrá la confidencialidad e integridad de los datos transmitidos, incluyendo la información de pago, y las credenciales de los usuarios. Reforzando esto, todas las formas de pago diferido, como transferencias bancarias y apartado, exigen un comprobante de la operación al usuario para su procesamiento.

Estos aseguran que un no pago no se traduzca en un bloqueo de inventario y evita fraudes y errores operativos. En su conjunto, estas medidas fortalecen la confianza del usuario al tiempo que proporcionan a los administradores herramientas eficientes de control.

Gestión de Inventarios

Principios de Gestión

Además, como se mencionó anteriormente, la gestión de inventarios es otro aspecto vital del comercio electrónico. Ayuda a garantizar que el stock disponible refleje de cerca la disponibilidad real de productos y evita problemas relacionados con la sobreventa o el desabasto. Para implementar el módulo de ventas descrito, estos principios clave se utilizaron, junto con las mejores prácticas obtenidas de una amplia variedad de fuentes especializadas.

- Sincronización en tiempo real. El módulo de ventas se implementó utilizando Django para sincronizar el inventario con las transacciones y cambiar los niveles de stock automáticamente después de cada compra. De esta manera, los clientes solo ven el haber exacto, reduciendo el riesgo de sobreventa.
- Control de Stock Mínimo: A través del método de par levels o niveles mínimos de inventario, se estableció un umbral crítico para cada producto que notificaba al administrador que el stock estaba por debajo de lo esperado. Dado que SkuVault llama la atención sobre la cantidad de “demanda” perdida y el “tiempo de reabastecimiento”, esta técnica mantiene un nivel de inventario para evitar desabastos.
- Automatización de Reabastecimiento: el sistema envía alertas automatizadas basadas en puntos de reorden calculados con datos de ventas y tiempos de entrega recomendados

- El módulo también puede generar órdenes de compra automática para productos específicos

Estos principios, soportados por la robustez de Django y validados por prácticas de la industria, aseguran una gestión eficiente del inventario, fortaleciendo la operatividad del módulo de ventas y la satisfacción del cliente.

Lógica de Stock

La lógica de stock: deberá describir cómo se deben administrar las existencias de productos de acuerdo con las entradas del usuario, que pueden ser comprar, cancelar o devolver. Estará basado en algunas teorías del SCM y los principios RDBMS para garantizar la precisión, la escalabilidad y robusto los errores.

Stock conceptual de diseño: de acuerdo con el sistema, las existencias de productos deberán estar destinadas a representarse mediante un atributo numérico que se asocia a un artículo específico y refleja la cantidad activa disponible para realizar la venta.

Para hacerlo, existirá un atributo de entero conocido como stock, cuyo valor se actualizará en el sistema a lo largo del tiempo en función de las transacciones subyacentes.

El diseño general incluye:

- Stock inicial: es la cantidad en el sistema para cada producto en el momento del registro.

- Umbral de alerta: el nivel más bajo de stock en el que un usuario debe recibir notificaciones.
- Historial de movimientos: se almacenan diferentes cambios aquí para seguir su seguimiento y auditoría.

El modelo de diseño conceptual (Figura 2) resuelve el diseño de entidad de un producto, cuyos atributos incluyen nombre, precio, stock y umbral de alerta. Estos datos se guardan y se accede a ellos a través de una base de datos relacional para consulta y actualización.

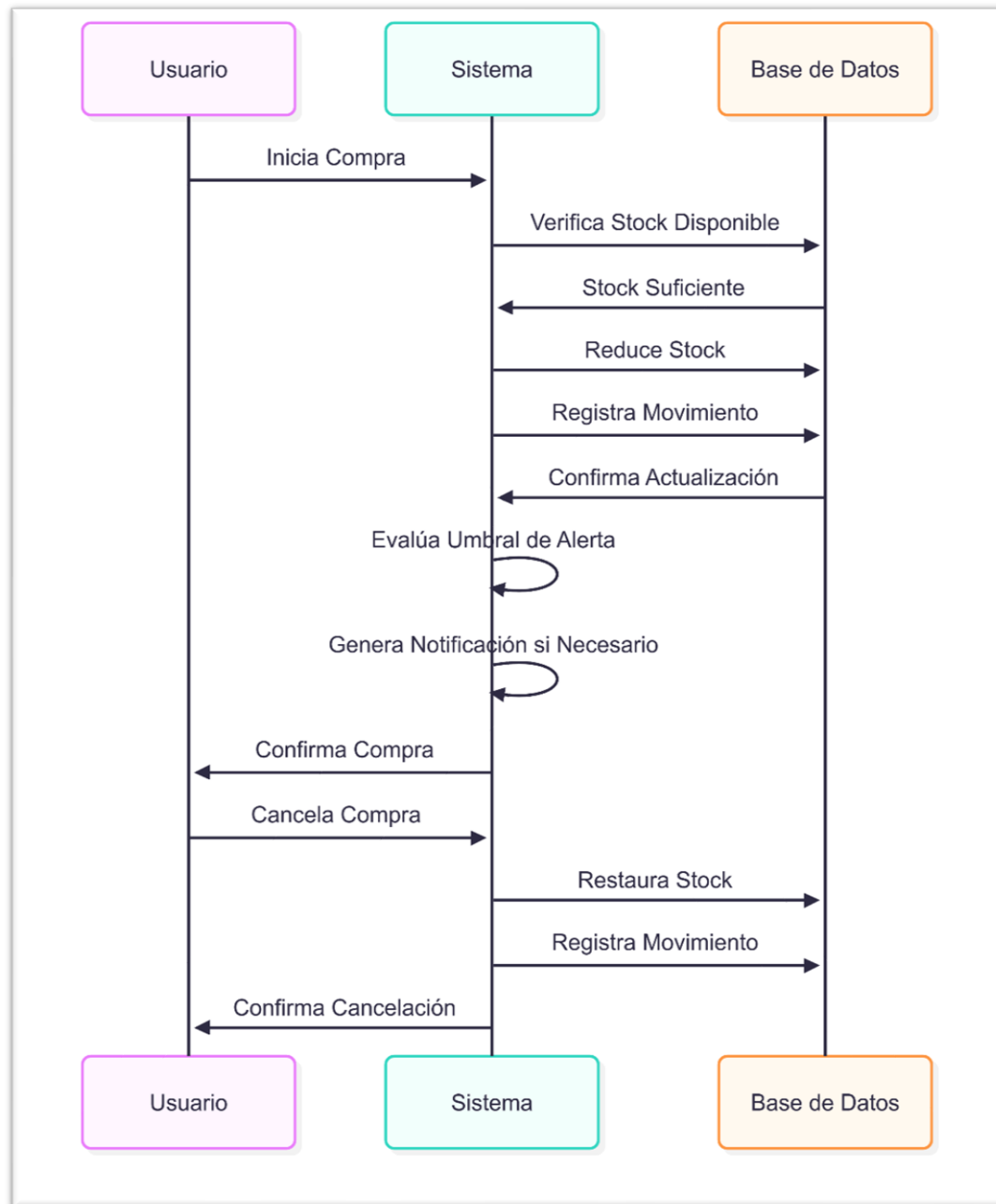


Figura 2 Flujo de una compra desde que el usuario la inicia hasta su confirmación o cancelación.

Seguridad Informática

Amenazas en E- commerce.

Cabe mencionar que referidos a amenazas en e-commerce no se trata de solo hipótesis, sino de riesgos reales que existen y que puede dañar la confianza y la reputación de un comercio electrónico. Tal es el caso de los ataques de los hackers que roban la información personal sensible, los fraudes en pagos o los fallos técnicos que molestan a los usuarios.

Entender estos peligros es el primer paso para protegernos, y aquí exploraremos cómo identificarlos y mitigarlos. Las amenazas planteadas a continuación podrían afectar Luxcer, comprometiendo su funcionamiento y la base de usuarios (OWASP, 2023);

- SQL injection: el atacante inserta consultas maliciosas en los campos de entrada para manipular la base de datos, con ello accede a información confidencial o modifica registro
- Solicitud Cross-Site Request Forgery: un ataque en el que se engaña a un usuario en línea para que realice envíos no deseados en un sistema autenticado que el usuario tiene derecho a realizar, como autorización de pagos o acceso al sistema para realizar transacciones no deseadas.
- Scripting en sitios cruzados: el ataque en el que se inyectan scripts maliciosos en los sitios web que se ejecutan en el lado del navegador del usuario, lo que permite la transferencia de sesiones a los contenidos del atacante o la redirección a su sitio web.
- Filtraciones de datos: El acceso no autorizado a información sensible debido a la configuración incorrecta del sistema o la lógica inadecuada.

- Ataques de denegación de servicio: el esfuerzo por interrumpir la utilización del servicio intensificando el rendimiento de la carga del sistema.
- Phishing: La forma de fraude en línea en la que se suplanta la identidad de la persona o entidad en línea.
- Escalada de privilegios: cómo las vulnerabilidades se benefician para obtener acceso como miembro administrativo.

Como la mayoría de estas amenazas son muy relevantes en el mercado y abarca temas financieros, personales y operativos validamos, el módulo de ventas garantiza la seguridad y continuidad completas.

Medidas Implementadas

El módulo deberá estar protegido frente a las diversas amenazas que se presentan en el comercio electrónico, tales como ataques de inyección SQL, csrf, XSS, DDoS o fuentes de filtración de información, entre otros, por lo que se diseñó un robusto paquete de seguridad en el que se aplicaron medidas desarrolladas conforme a estándares de la industria tales como OWASP Top 10 e ISO/IEC 27001, basadas en las capacidades propias del framework Django complementadas con protección establecida por tecnologías adicionales como se muestra en la (Figura 3).

Las protecciones nativas: Django, como framework de desarrollo web de alto nivel, tiene una cantidad significativa de protecciones que se implementaron en el módulo de ventas para asegurarse de no sufrir ataques. Aquí hay algunos nativos que podrían reducirse gracias a los ya existentes.

- ORM (Mapeo Objeto-Relacional): El mapeo relacional de objetos de Django permite la interacción con la base de datos a través de los objetos de Python, no las consultas directas SQL, lo que reduce significativamente el riesgo de inyección SQL, que es una de las primeras en la lista de las amenazas de software web críticas. OWASP (2023);
- Middleware CSRF: el “middleware” Cross-Site-Request Forgery de Django protege al usuario de acción no autorizada acerca de su cuenta, por ejemplo, el pago de su tarjeta sin su consentimiento;
- Salida de HTML: La salida de HTML evita inyecciones de scripts maliciosos en el lenguaje del navegador. Es decir, el código no intencionado no será ejecutado en el “client side” debido al hecho de que las comillas angulares y otros caracteres no válidos, como <, >, y & se escaparán automáticamente antes de que se muestren en la “UI”.

Protecciones Adicionales: Además de las capacidades nativas de Django incluyen protecciones adicionales para protegerse de amenazas y cumplir con los estándares de la industria, como OWASP Top 10 e ISO/IEC 27001.

- Entrada Validación: La validación estricta de la entrada en el lado del cliente y el servidor reduce los riesgos de entrada maliciosa como RFI, LFI, intentos de inyección y manipulación de formularios. Se trabaja con Escape HTML y ORM para una defensa multi-capa.

Cifrado de Datos Sensibles: El cifrado de datos sensibles deberá proteger la confidencialidad de la información transmitida y almacenada, como credenciales de usuarios

y detalles de transacciones. Mediante el uso de **SSL/TLS** para el tráfico de red y cifrado en la base de datos para datos críticos

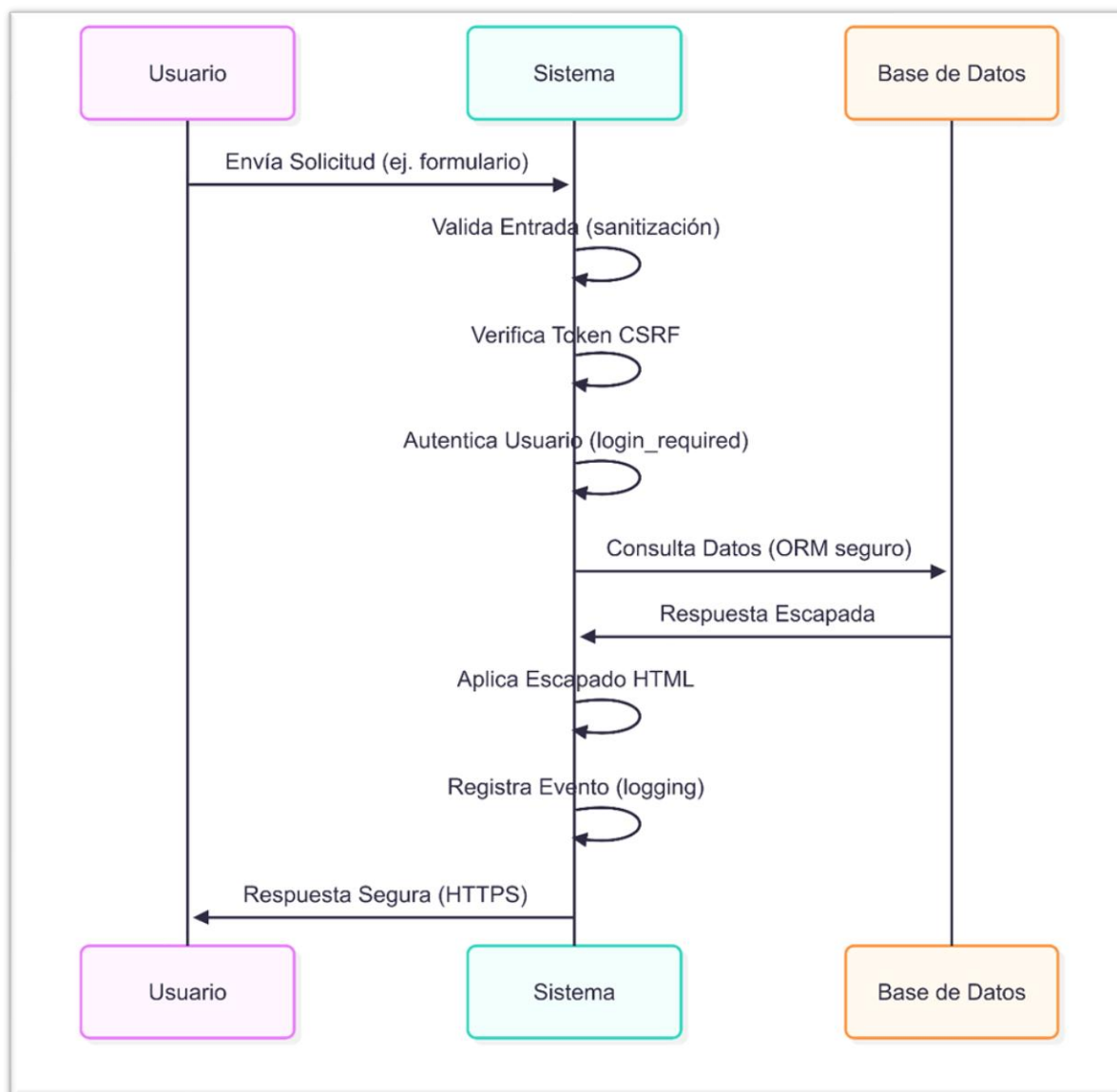


Figura 3 Diagrama que ilustra el flujo de un diagrama de secuencia seguro que atiende la solicitud aceptada proveniente del usuario, a través de medidas de protección de datos y de sistema.

Normativo Mexicano

El sistema de ventas desarrollado cumplirá la Ley de Protección de Datos Personales, que estipula tanto los principios y las obligaciones que se deben de seguir en el tratamiento de datos personales para garantizar el derecho a la intimidad y la autodeterminación informativa de las personas.

Cumplimiento Específico

- **Consentimiento:** ante la recopilación, tratamiento o transferencia de datos del usuario se obtiene el consentimiento previo y expreso del individuo, cumpliendo con el principio de licitud del artículo 6 de la LFPDPPP. Esquema: Formularios con casilla de aceptación de uso de datos personales.
- **Almacenamiento:** se utilizan bases de datos con mecanismos de seguridad técnica, física y administrativa como cifrado, control de acceso y de uso. Se trata de un sistema alineado al principio de responsabilidad.
- **Aviso de Privacidad:** se informa al usuario con un Aviso de Privacidad claro y accesible sobre qué datos se van a tratar, con qué finalidad, y los medios donde ejercer sus derechos ARCO.

Diseño de Interfaz de Usuario

Para el diseño se utilizarán los principios de usabilidad de Jakob Nielsen de la interfaz (Nielsen, 2020), así como los principios modernos de diseño centrado en el usuario (UX) y diseño visual (UI) como se muestra en la (Figura 4). Estos principios ayudaran a diseñar un nivel de usuario efectivo, accesible y agradable. Los principios más aplicables que se utilizaran son:

Funcionalidad Saylor la cual definirá el término funcional como «el grado en que una teoría psicológica puede aplicarse a su comportamiento, incluidas las incidencias numéricas». De hecho, el objetivo de cualquier funcionamiento de este tipo es promover la eficiencia de la plataforma de usuario.

- Simplicidad: Las interfaces son claras y mínimas, lo que prioriza la disposición de los elementos del uso correcto de los espacios en blanco y un recorrido directo.
- Consistencia: Usando Bootstrap mantiene un solo estilo visual a través del sistema. Esto incluye colores de elementos, conjunto de fuentes e iconografía y formato de elementos como botones o formularios. Todos los objetos que los usuarios pueden ver y tocar se comportan de la misma manera, lo que permite a los usuarios aprender cómo trabajan y no perder el tiempo intentando descubrir cosas nuevas cuando cambian los contextos.
- Feedback Inmediato: Responder con un mensaje claro de confirmación, error o advertencia del sistema a las acciones del usuario, tales como enviar un formulario, iniciar la sesión o subir un archivo. Estos mensajes permiten al usuario saber si su intento de interactuar fue o no exitoso, y si falló, qué necesita hacer para corregirlo.

- **Accesibilidad:** Se incluyeron prácticas en el caso de la restricción de la captación de imagen con la cámara electrónica de una persona.
 - Texto alternativo (alt) en imágenes.
 - Orientaciones adecuadas de contraste entre texto y fondo.
 - Soporte esperable de navegación por teclado.
 - Preferencia del uso semántico de etiquetas HTML que facilitan la navegación asistida.
- **Estado del Sistema Visible:** informa al usuario sobre el estado de las indicaciones de carga, órdenes y notificaciones visibles. Cuánto está pasando en el sistema
- **Prevención de Errores:** Se validan datos del lado del cliente y del servidor para evitar errores comunes. Te textos de ejemplo o de valores esperados son usados en los formularios para simplificar la tarea del usuario.
- **Control y Libertad del Usuario:** Se permite a los usuarios navegar de manera libre los recorridos viendo los pasos anteriores. Por ejemplo: Editar datos antes de confirmar una orden.

Tecnologías Actuales para el desarrollo web

Para el desarrollo de la interfaz se optó por algunas tecnologías ya mencionadas anteriormente con el fin de presentar una interfaz funcional, simplificada y accesible.

Bootstrap: Se trata de un framework CSS de código abierto que puede utilizarse para crear una interfaz de usuario responsiva y definida por el usuario, optimizada de tal forma que permita visualizar la pantalla en cualquier dispositivo independientemente de su tamaño. Esto se utilizó en el módulo de ventas para garantizar que el diseño pueda visualizarse sin cambiar la capacidad de respuesta de la interfaz de usuario; de este modo, sus usuarios clave incluyen estudiantes, profesores y público en general.

Chart.js: Chart.js es una biblioteca JavaScript de código abierto que se utiliza para crear gráficos interactivos y visualmente atractivos que representan gráficamente el conjunto de datos deseado. Chart.js se instaló en el módulo de ventas para permitir a los administradores ver informes visuales en la sección Dashboard. Esto permite a los administradores ver información vital sobre el rendimiento de la tienda virtual.

Jinja2: Jinja2 es un motor de plantillas se utiliza en el lenguaje de programación Python para generar contenido HTML dinámico y es accesible por Django para entregar vistas adecuadas en el frontend del lado del cliente. Era obligatorio en el módulo de ventas para producir plantillas dinámicas, como catálogos de productos, detalles de pedidos y notificaciones de transacciones.

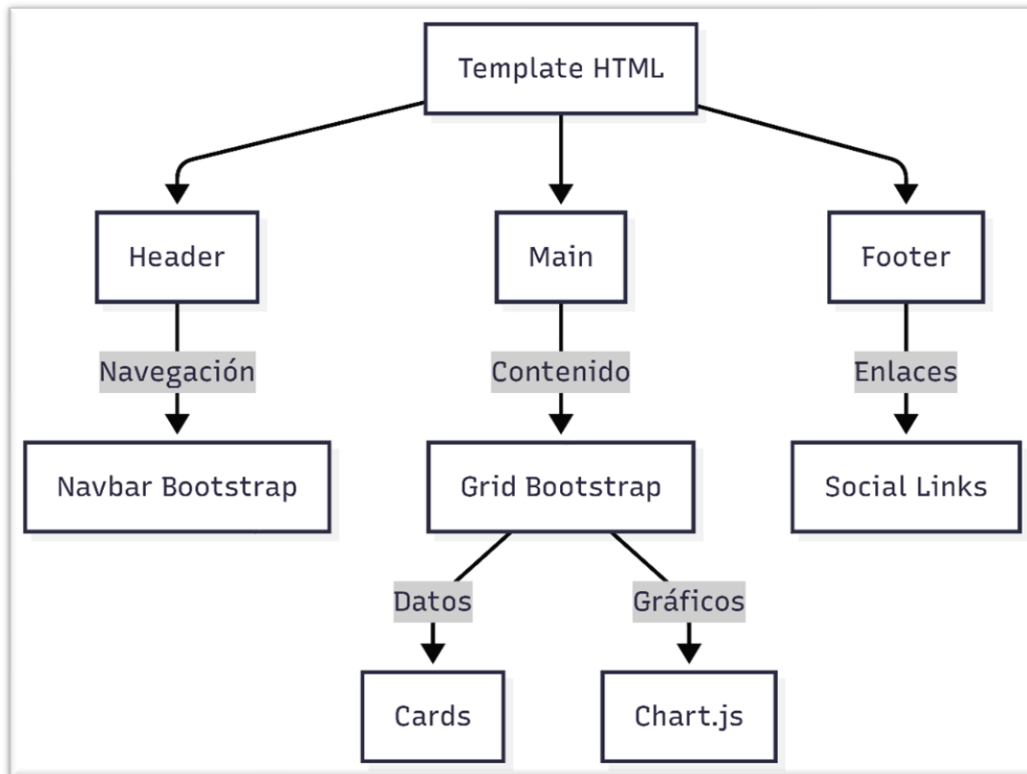


Figura 4 Diagrama que representa la estructura de un template HTML organizado en tres secciones principales: Header, Main y Footer.

CAPÍTULO 3: DESARROLLO

Introducción

En este informe se encuentra el desarrollo de un módulo de ventas para la tienda virtual del Instituto Tecnológico de Apizaco. Este módulo se integró a los módulos del carrito e inventario. La sección incluye el propósito general del proceso, la metodología empleada, las actividades llevadas a cabo en estas etapas y cómo se relacionan con los objetivos del proyecto.

El propósito general fue implementar un sistema seguro, escalable y eficiente que automatice transacciones comerciales e incluya mejoras de inventario y experiencia del usuario y del administrador de la plataforma del Instituto, que constaba de estudiantes, profesores y público en general. Kanban es un enfoque estrecho y flexible de la administración de proyectos que se utilizó, ofrece una manera flexible e iterativa para priorizar tareas, abordar desafíos técnicos y garantizar que las entregas sean continuas. Las actividades abarcan bothAnálisis de Requerimientos, Diseño de Arquitectura del Sistema, y Aplicaciones Desarrollo en Django, Ecommerce, Órdenes, Carros, Tienda y Social, Pasarelas de Pago, Pruebas de Unidad y Pruebas, y Documentación Técnica detallada, entre otras.

A continuación, las etapas, sus objetivos específicos, los procedimientos llevados a cabo, y los diagramas correspondientes explican los modelos, las clases y los flujos de datos del sistema.

Propósito de la Metodología

Metodología Kanban

La metodología Kanban es un sistema de gestión de tareas que forma parte de las metodologías ágiles. Su propósito principal es supervisar y optimizar el flujo de trabajo desde el inicio hasta la finalización de las tareas, asegurando un proceso continuo y eficiente.

Seleccionamos la metodología Kanban con el fin de gestionar el desarrollo del Módulo de Ventas, todo debido a su flexibilidad, capacidad para visualizar el flujo de trabajo y enfoque en la entrega continua.

Kanban permitió:

- Visualización del progreso: Un tablero Kanban reflejó el estado de las tareas en tiempo real.
- Priorización dinámica: Las tareas se ajustaron según las necesidades del Instituto y la retroalimentación de los asesores.
- Reducción de cuellos de botella: Limitar el trabajo en curso (WIP) aseguró un enfoque en la calidad.
- Colaboración efectiva: Facilitó la comunicación con el asesor interno (Dr. Juan Ramos Ramos) y externo (Lic. Miquelina Sánchez Pulido).

Implementación de Kanban

El desarrollo del módulo de ventas para la tienda virtual del Instituto Tecnológico de Apizaco se manejó con una metodología Kanban. Dicha metodología se estructura de tal manera para garantizar el trabajo continuo en un flujo y apoyada por un conjunto de herramientas tecnológicas que impulsaron la realización técnica del sistema. El proceso Kanban se estructuró en un tablero digital creado en Trello Figura 6, que tiene cuatro columnas que reflejan el estado de cada tarea: Por hacer, con actividades programadas; En progreso, con tareas en ejecución; En revisión, con entregables completados que esperan validación; y completado, con tareas validadas por un asesor.

Cada tarea se representó con tarjetas en Trello, en las cuales se incluía una descripción, un responsable, un nivel de importancia y una estimación temporal. En reuniones se revisaba el avance, así como revisar el desempeño verificando los entregables, priorizando tareas y planificando el siguiente ciclo de trabajo. Con este modelo fue posible garantizar un flujo continuo sobre el tablero a través de tiempos de realización de ciclo que promediaron los tres días por tarea con el 85% de tareas completadas en ese tiempo, lo que refleja la efectividad del método.

Algunos de los mecanismos que respaldaron esta metodología y su foto técnica son las siguientes herramientas: Trello para la gestión de las tareas y su avance, GitHub para el control de versiones y códigos compartidos; se configuró un repositorio para que el mantenimiento y seguimiento de los cambios de las funciones se dieran (Figura 5).

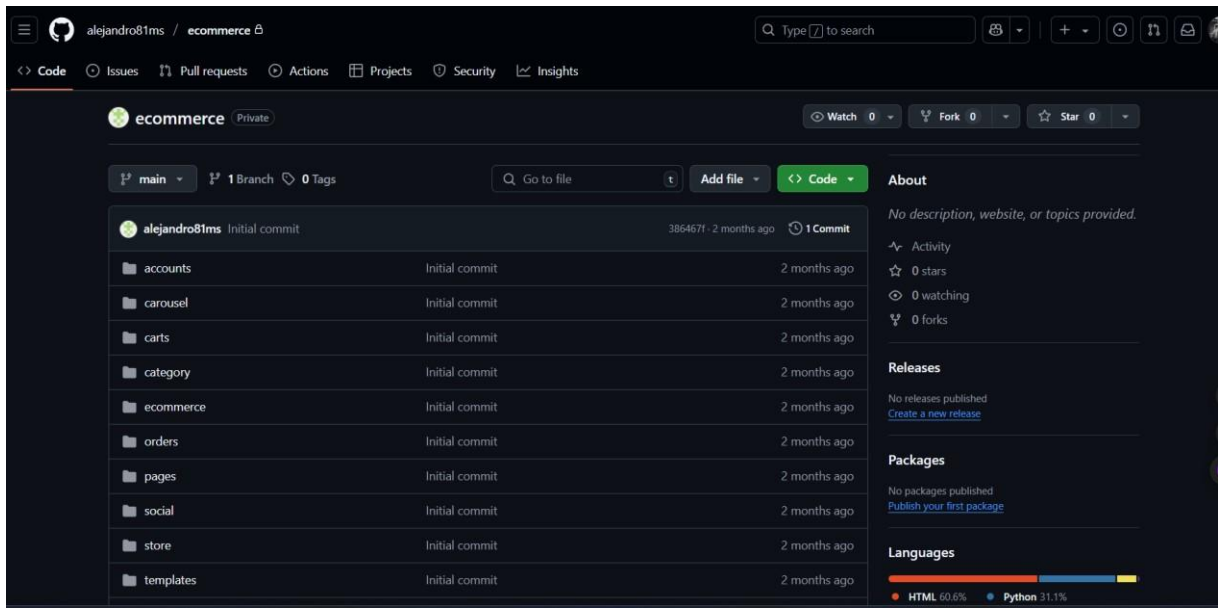


Figura 5 Repositorio en GitHub.

Django framework principal del desarrollo web; permitió la creación de aplicaciones como E. commerce y Orders; SQLite, para la gestión de base de datos en desarrollo, aunque fue aconsejado el uso de PostgreSQL en producción por su escalabilidad. También se utilizó Bootstrap 5 para interfaces responsivas y accesibles; que permitió mejorar la experiencia de usuario en el carrito de compras y el catálogo de productos, y ReportLab para la generación de comprobantes en formato PDF, siguiendo los requerimientos de facturación electrónica CFDI.

Beneficios de Kanban

Los principios de Kanban se traducen en una serie de **ventajas clave** para cualquier empresa que implemente este sistema.

- Flexibilidad: Permitió ajustar prioridades según retroalimentación del Departamento de Comunicación y Difusión.
- Transparencia: los plazos de entrega se vuelven más predecibles y confiables. Todo el equipo sabe exactamente cuál es su trabajo y en qué parte del ciclo se encuentra, por lo que el nivel de coordinación es mucho mejor y hay menos incertidumbre.
- Eficiencia: Kanban evita la sobreproducción y el uso ineficiente de los recursos. Además de optimizar el uso de materiales, también contribuyó a reducir el desperdicio y mejorar la eficiencia del proyecto en general.
- Control de las tareas: La metodología facilita un control más preciso del esfuerzo y la planificación, lo que resulta en una mayor productividad.
- Iteración rápida: Las revisiones aseguraron entregables funcionales.

Desglose de Actividades

A continuación, se detalla cada actividad del cronograma, con su propósito, procedimiento, resultados, códigos relevantes, y diagramas asociados, basados en las aplicaciones Orders y Ecommerce.

Inicio

Con el propósito central de familiarizarse con el Instituto Tecnológico de Apizaco, comprender su estructura organizativa, normativas internas y proyectos tecnológicos en curso, y sentar las bases sólidas para el desarrollo del módulo de ventas de la tienda virtual, se llevó a cabo una actividad inicial. Esta actividad comenzó con una reunión de presentación con el equipo de trabajo, incluyendo al asesor Dr. Juan Ramos Ramos, quien reviso los lineamientos institucionales, aclarando objetivos y expectativas para alinear el proyecto con las necesidades del Instituto.

Se analizó documentación clave, como la misión, visión y valores del Instituto, junto con normativas internas y proyectos tecnológicos existentes, lo que permitió contextualizar el rol del módulo de ventas dentro del entorno institucional. Como resultado, se elaboró un informe inicial que detalló la estructura organizativa, destacando departamentos como Sistemas y Computación y Comunicación y Difusión, y describió el contexto tecnológico, identificando necesidades comerciales clave, como la automatización de transacciones. Para garantizar una colaboración efectiva, se establecieron canales de comunicación estructurados, implementando reuniones para revisiones, un grupo de WhatsApp para coordinaciones rápidas con el Departamento de Comunicación y Difusión, y un tablero Trello

compartido para dar seguimiento a las tareas iniciales, promoviendo transparencia y eficiencia. Estos resultados, validados por los asesores, aseguraron una comprensión clara del Instituto y sus prioridades, proporcionando una base sólida para el desarrollo del módulo de ventas, alineado con los objetivos institucionales y las necesidades comerciales del Instituto Tecnológico de Apizaco.

Inducción

Con el propósito central de analizar las problemáticas y necesidades del área asignada y comprender los procesos relacionados con el desarrollo del módulo de ventas de la tienda virtual del Instituto Tecnológico de Apizaco, se llevó a cabo una actividad de inducción. Esta actividad comenzó con un análisis exhaustivo de las problemáticas comerciales mediante sesiones de trabajo con el Departamento de Comunicación y Difusión, donde se identificaron necesidades clave, como la falta de un sistema integrado de ventas, el control manual de inventarios y la ausencia de comprobantes automatizados, además de definir a los usuarios principales, incluyendo clientes como estudiantes, docentes y público general, así como administradores encargados de la gestión operativa.

Para garantizar una base sólida, llevamos a cabo una capacitación, enfocada en herramientas como Django para el desarrollo web, complementando con formación en metodologías ágiles, específicamente Kanban, y el uso de Trello para la gestión de tareas asignadas. Igualmente, se revisaron los procesos administrativos, analizando flujos de ventas

manuales, generación de reportes y gestión de inventarios, lo que nos permitió documentar las prácticas existentes y detectar áreas de mejora, como la automatización de tareas repetitivas y la optimización del seguimiento de transacciones.

Como resultado, se elaboró un informe detallado que sintetizó los problemas identificados, incluyendo la falta de integración entre ventas e inventarios y la necesidad de reportes automáticos, y se estableció un dominio inicial de las herramientas y normativas necesarias, validado por el asesor Dr. Juan Ramos Ramos. Estos resultados, logrados mediante un enfoque estructurado y colaborativo con el personal administrativo, sentaron las bases para el diseño del módulo de ventas, asegurando que el proyecto abordara las necesidades del Instituto Tecnológico de Apizaco y estuviera alineado con sus objetivos comerciales y tecnológicos.

Asignación del Proyecto

Con el propósito central de definir los objetivos, alcances y cronograma del proyecto para el desarrollo del módulo de ventas y obtener la aprobación del plan de trabajo, se llevó a cabo una actividad. Esta actividad comenzó con la revisión del anteproyecto, donde se delinearon el objetivo general de crear un sistema integrado de ventas y objetivos específicos, como implementar el registro de transacciones, integrar pasarelas de pago, generar comprobantes, sincronizar inventarios, producir reportes analíticos y garantizar medidas de seguridad, alineando estas metas con la misión del Instituto de promover innovación y responsabilidad social. Para estructurar el desarrollo, se adaptó una Gráfica de Gantt que detalló un cronograma de 22 semanas, abarcando 12 actividades principales, desde el análisis de requisitos hasta la integración con módulos de inventario, asignando plazos y responsables claros para cada tarea. Complementando esta planificación, se delimitaron los requisitos funcionales y programados mediante sesiones de trabajo con stakeholders clave, como administradores, estudiantes y docentes, recopilando necesidades como una interfaz intuitiva y automatización de procesos comerciales.

El plan de trabajo resultante fue presentado al asesor Dr. Juan Ramos Ramos, quien proporciono retroalimentación para ajustar plazos y priorizar funcionalidades, como la generación automática de comprobantes. Para gestionar las tareas de manera ágil, se configuró un tablero Kanban en Trello (Figura 6), estructurado con columnas para Listos para iniciar, En desarrollo, Pausado y Concluido, incorporando actividades iniciales como la instalación del entorno de desarrollo con Django, la redacción del documento de requerimientos y el diseño preliminar de la base de datos.

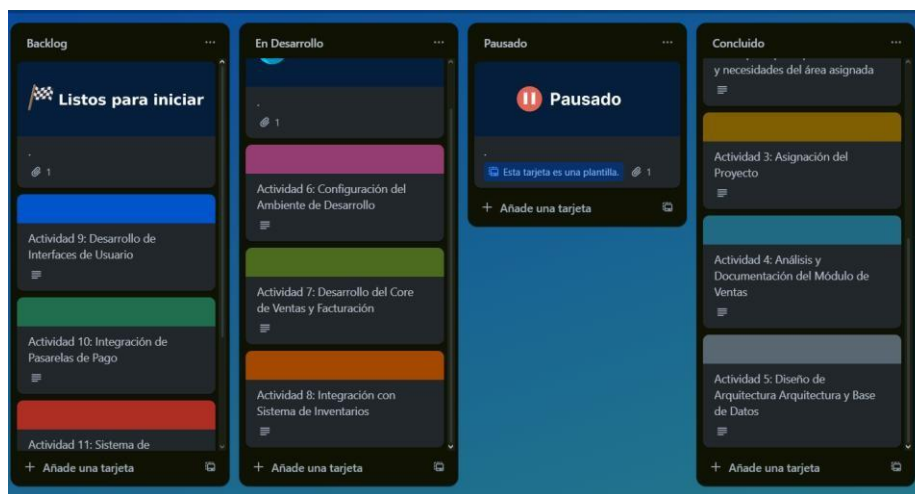


Figura 6 se presenta un tablero Kanban configurado en la plataforma Trello, el cual organiza las tareas del proyecto en diferentes columnas: Backlog, En Desarrollo, Pausado y Concluido.

Este plan, validado por los asesores, aseguró un desarrollo estructurado y transparente, alineado con los valores de honestidad y compromiso ético del Instituto, sentando las bases para un módulo de ventas funcional, seguro y adaptado a las necesidades comerciales del Instituto Tecnológico de Apizaco.

Recopilación de Requisitos Funcionales y No Funcionales

Para definir los requisitos del módulo de ventas, se realizó la recopilación de información a través de entrevistas, a los stakeholders clave, como el jefe del Departamento de Comunicación y Difusión.

ID	MODULO	DESCRIPCIÓN
RF-01	Ventas	El sistema debe registrar cada transacción de venta (Usuario que realizó la compra, Productos adquiridos con sus respectivas cantidades y precios, Fecha y hora de la transacción.)
RF-02	Ventas	Al finalizada la venta, el sistema debe generar automáticamente un comprobante digital en formato PDF.
RF-03	Ventas	El sistema debe permitir pagos mediante transferencia bancaria, apartado (72 horas)
RF-04	Ventas	Antes de completar la compra, el sistema debe validar la información de pago y confirmar la transacción de forma segura.
RF-05	Ventas	En caso de error en la transacción, el sistema debe notificar al usuario y permitir reintentarlo.
RF-06	Ventas	El sistema debe actualizar automáticamente el inventario al completar una venta, reduciendo la cantidad de productos vendidos.
RF-07	Ventas	Si el stock de un producto es cero, el sistema debe deshabilitar su compra y notificar al administrador.

RF-08	Ventas	El sistema debe generar reportes de ventas con productos vendidos, estado de pagos (completados, pendientes, Cancelados), fechas, y montos totales.
RF-09	Ventas	Los reportes deben ser exportables en PDF y CSV.
RF-10	Ventas	El sistema debe mantener un historial de ventas accesible para administradores y clientes en sus perfiles.
RF-11	Ventas	El sistema debe permitir buscar y filtrar transacciones por fecha, cliente, producto, o estado de pago.
RF-12	Ventas	Los usuarios deben recibir notificaciones en tiempo real sobre confirmación de compra, errores de transacción, y estado del pedido.
RF-13	Ventas	Las notificaciones deben enviarse por correo electrónico o dentro del sistema.
RF-14	Ventas	Todas las transacciones deben usar SSL/TLS para encriptar datos sensibles.
RF-15	Ventas	El sistema debe implementar autenticación y control de acceso por roles (cliente, administrador).
RF-16	Ventas	El sistema debe contar con autenticación y control de acceso por roles (cliente, administrador, auditor, etc.).

Tabla 1 Requisitos funcionales del módulo de ventas, los cuales definen las capacidades esenciales del sistema.

ID	MODULO	DESCRIPCIÓN
01	RNF- Ventas	El sistema debe ser accesible desde navegadores web y dispositivos móviles sin pérdida de funcionalidad.
02	RNF- Ventas	El sistema debe ser capaz de procesar al menos 100 transacciones simultáneas sin afectar el rendimiento
03	RNF- Ventas	El tiempo de respuesta para registrar una venta no debe superar los 2 segundos.
04	RNF- Ventas	El sistema debe realizar copias de seguridad automáticas de la base de datos cada 24 horas.
05	RNF- Ventas	El sistema debe garantizar una disponibilidad del 99.5% durante el horario comercial.
06	RNF- Ventas	La interfaz debe ser intuitiva, con un diseño basado en buenas prácticas de UX/UI.

Tabla 2 Requisitos no funcionales del módulo de ventas, los cuales establecen criterios de calidad.

Análisis y Documentación del Módulo de Ventas

Para cumplir con el fin mencionado de recolectar requerimientos específicos, definir reglas de negocio y la identificación de integraciones para el desarrollo del módulo de ventas, se realizó una actividad de análisis. Esta consistió en:

Recolección de requerimientos: se llevaron a cabo sesiones con el Departamento de Comunicación y Difusión para identificar los requerimientos; en cuanto a las necesidades de los usuarios,

Definir reglas de negocio: se definió que las ventas se registrarían con estados de “New”, “Pending”, “Completed” y “Canceled”, los pagos en transferencia se validarían en 24 horas y los pagos en apartados en 72 horas y que el inventario se restablecería automáticamente una vez se validara el pago y el stock y emitiría notas de crédito según la política de devolución o cambio según la institución.

Como se muestra en la (Figura 7) se generó un diagrama de casos de uso que describe a los actores principales y las interacciones que tienen entre sí, como registrar ventas o consultar inventarios; y se elaboró un documento de requerimientos en el que se recopilaban necesidades funcionales, como la generación de comprobantes, y no funcionales.

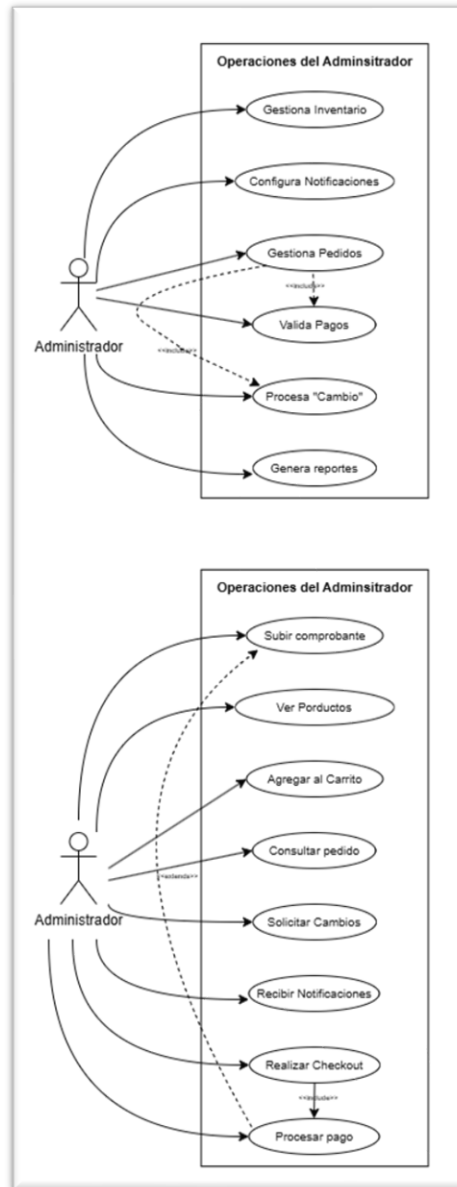


Figura 7 Diagrama de casos de uso del sistema para dos actores: Administrador y Cliente. El administrador realiza funciones de gestión y supervisión, Por su parte, el cliente interactúa con el sistema para explorar productos, realizar compras y solicitar cambios.

Requerimientos de usuario

Los requisitos de usuario para el módulo de ventas de la tienda virtual del Instituto Tecnológico de Apizaco se recopilaron a través de entrevistas. Esto implica explorar las percepciones y experiencias de los usuarios sobre los procesos comerciales tal como existen actualmente, incluidas las compras y el manejo de los pedidos.

Gestión de Ventas

1. El usuario puede registrar nuevas ventas en el sistema, proporcionando productos, cantidad, precio y método de pago.
2. Historial de ventas consultable.
3. Cancelación de ventas, si es necesario con base en la política.
4. Comprobante de pago oficial, como una factura o ticket generado automáticamente después de cada compra completada.
5. El usuario podrá gestionar el estado de los pagos

Integración con Inventarios

6. Se actualiza el stock automáticamente cuando se genera una venta
7. Disponibilidad de productos en tiempo real
8. Se generarán alertas cuando el stock de un producto sea insuficiente o esté próximo a agotarse.

Métodos de Pago

9. Se implementará la integración con diversas pasarelas de pago para ofrecer opciones como transferencias bancarias.
10. Validación e información de pago segura almacenada.
11. Estado de pagos consultable y notificaciones de pago rechazado o “en proceso”.

Seguridad y Protección de Datos

12. Protección de información personal
13. Acceso controlado a información de ventas o pagos solo por usuarios autorizados
14. Se cumplirán normativas de protección de datos aplicables.

Comprobantes

15. Comprobantes detallados de cada venta
16. Comprobantes exportables en PDF.

Automatización y Notificaciones

18. Notificaciones automáticas a los clientes
19. Se gestionarán procesos de cambio según las políticas establecidas por la tienda.

Diseño de Arquitectura y Base de Datos

Se llevó a cabo una actividad con el objetivo principal de diseñar la arquitectura técnica de operación del módulo de ventas, como el modelo relacional, las integraciones de servicios y la documentación técnica. Se empezó por definir un modelo de datos relacional en Django, identificando entidades claves como Order, Payment, OrderProduct, Cart, CartItem, Product y ProductVariation (con atributos correspondientes como estado de la orden en Order, precio en Product, etc.) y relaciones vía claves foráneas como entre Order y User en Account para asociar transacciones con usuarios, Payment para registrar métodos de pago, etc, o entre CartItem y Product para manejar el carrito de compras.

Para garantizar la funcionalidad del sistema, se diseñaron integraciones de servicios, implementando notificaciones automáticas mediante el módulo EmailMessage de Django, configurado con un servidor SMTP para enviar correos a clientes y administradores sobre estados de pedidos, y desarrollando señales Django, como post_save en el modelo Order, para actualizar automáticamente el stock en el modelo Product tras confirmar una venta, asegurando sincronización en tiempo real con el módulo de inventarios.

Como parte de la documentación técnica, se elaboró un diagrama UML-Entidad-Relación (ERD) (Figura 8) que ilustró las entidades, sus atributos y relaciones, para representar gráficamente los flujos de datos entre los módulos de ventas, carritos e inventarios. Este diseño aseguró una arquitectura robusta y escalable, alineada con las necesidades comerciales del Instituto Tecnológico de Apizaco, proporcionando una base técnica sólida para el desarrollo e integración del módulo de ventas con otros componentes del sistema.

- id: Clave primaria (BigAutoField, PK).
- username: Nombre de usuario (CharField, 150 caracteres, único).
- email: Correo electrónico (EmailField, único).
- first_name: Nombre (CharField, 50 caracteres).
- last_name: Apellido (CharField, 50 caracteres).
- is_admin: Indicador de administrador (BooleanField, predeterminado False).

Propósito: Representa las cuentas de usuarios y administradores, sirviendo como base para autenticación y autorización en el módulo de ventas.

2. **UserProfile** (Perfil de Usuario):

Atributos:

- id: Clave primaria (BigAutoField, PK).
- user: Relación uno a uno con Account (OneToOneField, FK).
- phone: Teléfono (CharField, 15 caracteres, regex="^\d{10}\$").
- street_address: Dirección (CharField, 100 caracteres).
- zip_code: Código postal (CharField, 5 caracteres, regex="^\d{5}\$").

Propósito: Extiende los datos del usuario para fines de envío y facturación (CFDI), asegurando cumplimiento normativo (LFPDPPP).

3. **Carousel** (Carrusel Promocional):

Atributos:

- id: Clave primaria (BigAutoField, PK).
- title: Título (CharField, 200 caracteres).
- description: Descripción (TextField, blank).
- is_active: Indicador activo (BooleanField, predeterminado True).

Propósito: Gestiona carruseles para promociones en el catálogo, mejorando la visibilidad de productos en el e-commerce.

4. **CarouselImage** (Imagen de Carrusel):

Atributos:

- id: Clave primaria (BigAutoField, PK).
- carousel: Relación con Carousel (ForeignKey, FK).
- image: Imagen (ImageField, upload_to='carousel_images/').
- order: Orden (PositiveIntegerField).

Propósito: Almacena imágenes para carruseles, permitiendo ordenamiento visual en la interfaz de ventas.

5. **Notification** (Notificación):

Atributos:

- id: Clave primaria (BigAutoField, PK).
- user: Relación con Account (ForeignKey, FK).

- message: Mensaje (TextField).
- is_read: Leído (BooleanField, predeterminado False).
- created_at: Fecha de creación (DateTimeField, auto_now_add).

Propósito: Maneja notificaciones para eventos como pagos confirmados, integradas con el seguimiento de órdenes.

6. **Payment** (Pago):

Atributos:

- id: Clave primaria (BigAutoField, PK).
- user: Relación con Account (ForeignKey, FK).
- payment_id: ID de pago (CharField, 100 caracteres, único).
- payment_method: Método de pago (CharField, 50 caracteres, choices=PAYMENT_METHODS).
- amount: Monto (DecimalField, 10,2).
- status: Estado (CharField, 50 caracteres, choices=PAYMENT_STATUSES).
- proof: Comprobante (FileField, upload_to='proofs/').
- created_at: Fecha de creación (DateTimeField, auto_now_add).

Propósito: Registra pagos y comprobantes, soportando pasarelas y validaciones fiscales (CFDI).

7. **Cart** (Carrito):

Atributos:

- id: Clave primaria (BigAutoField, PK).
- cart_id: ID de carrito (CharField, 250 caracteres, único).
- created_at: Fecha de creación (DateTimeField, auto_now_add).

Propósito: Gestiona carritos temporales para compras, sincronizados con ventas.

8. **CartItem** (Ítem de Carrito):

Atributos:

- id: Clave primaria (BigAutoField, PK).
- user: Relación con Account (ForeignKey, FK).
- product: Relación con Product (ForeignKey, FK).
- variation: Relación con ProductVariation (ForeignKey, FK, null).
- cart: Relación con Cart (ForeignKey, FK).
- quantity: Cantidad (PositiveIntegerField).
- is_active: Activo (BooleanField, predeterminado True).
- is_processed: Procesado (BooleanField, predeterminado False).

Propósito: Detalla ítems en el carrito, facilitando la transición a pedidos.

9. **Category** (Categoría):

Atributos:

- id: Clave primaria (BigAutoField, PK).
- name: Nombre (CharField, 50 caracteres).
- slug: Slug (SlugField, único).

Propósito: Clasifica productos para una navegación organizada en el catálogo.

10. **Product** (Producto):

Atributos:

- id: Clave primaria (BigAutoField, PK).
- product_name: Nombre (CharField, 100 caracteres).
- slug: Slug (SlugField, único).
- price: Precio (DecimalField, 10,2).
- stock: Stock (PositiveIntegerField).
- is_available: Disponible (BooleanField, predeterminado True).
- category: Relación con Category (ForeignKey, FK).
- created_at: Fecha de creación (DateTimeField, auto_now_add).

Propósito: Gestiona el catálogo de productos, sincronizado con ventas e inventarios.

11. **ProductVariation** (Variación de Producto):

Atributos:

- id: Clave primaria (BigAutoField, PK).
- product: Relación con Product (ForeignKey, FK).
- color: Color (CharField, 50 caracteres).
- stock: Stock (PositiveIntegerField).
- is_active: Activo (BooleanField, predeterminado True).

Propósito: Maneja variaciones de productos (ej., colores) para un control preciso de inventarios.

12. **Order** (Orden):

Atributos:

- id: Clave primaria (BigAutoField, PK).
- user: Relación con Account (ForeignKey, FK).
- payment: Relación con Payment (ForeignKey, FK).
- order_number: Número de orden (CharField, 20 caracteres, único).
- order_total: Total (DecimalField, 10,2).
- tax: Impuesto (DecimalField, 5,2).
- status: Estado (CharField, 50 caracteres, choices=STATUS).
- is_ordered: Ordenada (BooleanField, predeterminado False).
- created_at: Fecha de creación (DateTimeField, auto_now_add).

- `updated_at`: Fecha de actualización (DateTimeField, `auto_now`).
- `street_address`: Dirección (CharField, 100 caracteres).
- `zip_code`: Código postal (CharField, 5 caracteres, `regex="^\d{5}$"`).

Propósito: Registra y gestiona pedidos, central para el módulo de ventas.

13. **OrderProduct** (Producto en Orden):

Atributos:

- `id`: Clave primaria (BigAutoField, PK).
- `order`: Relación con Order (ForeignKey, FK).
- `product`: Relación con Product (ForeignKey, FK).
- `variation`: Relación con ProductVariation (ForeignKey, FK, null).
- `quantity`: Cantidad (PositiveIntegerField).
- `product_price`: Precio (DecimalField, 10,2).

Propósito: Detalla los productos en un pedido, facilitando la sincronización con inventarios.

14. **Review** (Reseña):

Atributos:

- `id`: Clave primaria (BigAutoField, PK).
- `user`: Relación con Account (ForeignKey, FK).

- **product:** Relación con Product (ForeignKey, FK).
- **rating:** Calificación (PositiveIntegerField).
- **comment:** Comentario (TextField).
- **created_at:** Fecha de creación (DateTimeField, auto_now_add).

Propósito: Captura retroalimentación de usuarios sobre productos, mejorando la calidad del catálogo.

Relaciones entre Entidades

- **Account - UserProfile:** Uno a uno, para extender datos del usuario.
- **Account - Order, Payment, CartItem, Review, Notification:** Uno a muchos, un usuario puede tener múltiples pedidos, pagos, ítems, reseñas y notificaciones.
- **Order - Payment:** Uno a uno, cada pedido tiene un pago asociado.
- **Order - OrderProduct:** Uno a muchos, un pedido contiene varios productos.
- **OrderProduct - Product, ProductVariation:** Uno a uno (o muchos), cada ítem se asocia a un producto y su variación.
- **Cart - CartItem:** Uno a muchos, un carrito contiene varios ítems.
- **CartItem - Product, ProductVariation:** Uno a uno, cada ítem se asocia a un producto y su variación.
- **Category - Product:** Uno a muchos, una categoría agrupa múltiples productos.

- **Product - ProductVariation, Review:** Uno a muchos, un producto tiene variaciones y reseñas.
- **Carousel - CarouselImage:** Uno a muchos, un carrusel contiene múltiples imágenes.

Elaboración de Historias de Usuario

Se abordaron historias de usuario para el módulo de ventas mediante sesiones de trabajo con el jefe del Departamento de Sistemas y Computación, administradores y estudiantes. Se diseñó un proceso simple para identificar las necesidades de los usuarios. Este proceso permitió entender qué esperan los usuarios de la tienda, como una compra fácil o reportes rápidos.

Las tablas 3 a 14 presentan las historias de usuario correspondientes las especificaciones del módulo de ventas. En esta definición, se incluye la descripción y los criterios de aceptación para su desarrollo.

Historias de Usuario				
Historia de Usuario	Descripción Historia de Usuarios		Criterios de Aceptación	
1	Como	Administrador	El sistema debe permitir seleccionar productos del catálogo existente	
	Quiero	Registrar nuevas ventas en el sistema especificando detalles como productos, cantidad, precio y método de pago.	Debe calcular automáticamente el total de la venta según las cantidades y precios	Debe permitir seleccionar entre múltiples métodos de pago
	Para	Mantener un control preciso de las	El sistema debe	Se debe mostrar

		transacciones comerciales realizadas.	validar que los datos ingresados sean correctos antes de confirmar la venta	una confirmación al finalizar el registro exitoso de la venta
--	--	---------------------------------------	---	---

Tabla 3 Historias de Usuario: Registrar Nueva Venta

Historias de Usuario				
Historia de Usuario	Descripción Historia de Usuarios		Criterios de Aceptación	
2	Como	Administrador	El sistema debe mostrar un listado de ventas ordenado cronológicamente	
	Quiero	Consultar el historial de ventas aplicando filtros por fecha, cliente y estado de pago.	Debe permitir aplicar filtros por rango de fechas, cliente específico y estado de pago	Debe permitir ver el detalle completo de una venta al seleccionarla
	Para	Analizar el comportamiento de las ventas y dar seguimiento a transacciones específicas.	Debe mostrar información resumida de cada venta (fecha, cliente, monto, estado)	Los resultados deben actualizarse inmediatamente al aplicar los filtros

Tabla 4 Historias de Usuario: Consultar Historial de Ventas

Historias de Usuario				
Historia de Usuario	Descripción Historia de Usuarios		Criterios de Aceptación	
3	Como	Administrador	Solo se pueden cancelar ventas dentro del tiempo establecido por las políticas	

	Quiero	Cancelar ventas en caso de errores respetando las políticas establecidas.	El sistema debe poder cancelar correctamente	Debe existir un nivel de autorización para cancelaciones.
	Para	Corregir transacciones incorrectas sin afectar la integridad del sistema.	El sistema debe registrar quién realizó la cancelación y cuándo	El inventario debe actualizarse automáticamente tras la cancelación

Tabla 5: Historias de Usuario: Cancelar Venta

Historias de Usuario				
Historia de Usuario	Descripción Historia de Usuarios		Criterios de Aceptación	
4	Como	Administrador	Los comprobantes deben cumplir con los requisitos.	
	Quiero	Que se generen automáticamente comprobantes oficiales de pago tras cada transacción exitosa.	El sistema debe generar tickets según la solicitud del cliente	Los comprobantes deben incluir todos los detalles.
	Para	Cumplir con requisitos y proporcionar evidencia de compra al cliente.	Debe ser posible enviar el comprobante por correo electrónico	Se debe poder reimprimir el comprobante en caso necesario

Tabla 6: Historias de Usuario: Generar Comprobante

Historias de Usuario			
Historia de Usuario	Descripción Historia de Usuarios		Criterios de Aceptación
5	Como	Administrador	El sistema debe permitir marcar pagos como pendientes, parciales o completados

	Quiero	Gestionar el estado de los pagos de las ventas realizadas.	Debe ser posible registrar pagos.	Se debe visualizar claramente el estado de pago en el registro de la venta
	Para	Tener control sobre la situación financiera de cada transacción.	El sistema debe calcular automáticamente.	Debe existir una vista de los cambios en el estado de pago

Tabla 7: Historias de Usuario: Gestionar Estado de Pagos

Integración con inventarios				
Historia de Usuario	Descripción Historia de Usuarios		Criterios de Aceptación	
6	Como	Administrador	El stock debe actualizarse en tiempo real al confirmar una venta	
	Quiero	Que el sistema actualice automáticamente el stock tras cada venta.	El sistema debe registrar el movimiento en el historial de inventario	En caso de cancelación, el stock debe restaurarse correctamente
	Para	Mantener sincronizado el inventario con las ventas realizadas.	Si ocurre un error en la actualización, debe notificarse al administrador	La integración debe funcionar incluso en caso de ventas simultáneas

Tabla 8: Historias de Usuario: Actualizar Stock Automáticamente

Historia de Usuario	Descripción Historia de Usuarios		Criterios de Aceptación	
7	Como	Administrador	El sistema debe mostrar el stock disponible al seleccionar un producto	
	Quiero	Verificar la disponibilidad de productos en tiempo real antes de completar una venta,	Debe alertar visualmente cuando se intente vender más unidades que las disponibles	Debe impedir finalizar ventas con cantidades mayores al stock disponible
	Para	Evitar comprometer productos que no están en existencia.	La información de disponibilidad debe actualizarse en tiempo real	Debe mostrar la fecha estimada de reposición para productos agotados

Tabla 9: Historias de Usuario: Verificar Disponibilidad de Productos

Historias de Usuario				
Historia de Usuario	Descripción Historia de Usuarios		Criterios de Aceptación	
8	Como	Administrador	El sistema debe definir umbrales de stock mínimo para cada producto	
	Quiero	Recibir alertas cuando el stock de un producto sea insuficiente o esté próximo a agotarse.	Las alertas deben notificarse en el sistema	Las alertas deben incluir información del producto.
	Para	Gestionar oportunamente la reposición y evitar ruptura de stock.	Debe existir un panel de control para visualizar todos los productos en alerta	Las alertas deben categorizarse según la rotación del producto

Tabla 10: Historias de Usuario: Alertas de Stock Bajo

Métodos de pago				
Historia de Usuario	Descripción Historia de Usuarios		Criterios de Aceptación	
9	Como	Administrador	El sistema debe integrarse con pasarelas de pago diferentes	
	Quiero	Contar con integración a diversas pasarelas de pago.	La interfaz de pago debe ser consistente independientemente del método seleccionado	El proceso de pago debe realizarse sin salir del sistema
	Para	Ofrecer múltiples opciones de pago.	Debe mostrarse claramente las comisiones aplicables por cada método	El sistema debe confirmar la transacción exitosa o indicar el motivo de rechazo

Tabla 11: Historias de Usuario: Integrar Pasarelas de Pago

Historias de Usuario				
Historia de Usuario	Descripción Historia de Usuarios		Criterios de Aceptación	
10	Como	Administrador	El sistema debe mostrar el estado actual de cada transacción (Nuevo, Aceptado, Completado, Cancelado)	
	Quiero	Consultar el estado de los pagos y recibir notificaciones sobre transacciones rechazadas o en proceso	Las notificaciones deben llegar en tiempo real por correo y en el sistema	Debe existir un panel para gestionar todas las transacciones.
	Para	Dar seguimiento oportuno a situaciones que requieran atención.	El sistema debe proporcionar información detallada.	Debe permitir la visualización desde la interfaz.

Tabla 12: Historias de Usuario: Notificar Estado de Pagos

Seguridad y Protección de Datos				
Historia de Usuario	Descripción Historia de Usuarios		Criterios de Aceptación	
11	Como	Administrador	El sistema debe implementar autenticación para accesos críticos	
	Quiero	Implementar medidas de seguridad para proteger la información personal y financiera.	Toda la información sensible debe transmitirse y almacenarse cifrada	Debe existir un registro de auditoría de accesos a información sensible
	Para	Garantizar la confidencialidad e integridad de los datos sensibles.	El sistema debe cumplir con estándares reconocidos de seguridad informática	Debe implementarse un plan de respuesta a incidentes de seguridad

Tabla 13: Historias de Usuario: Implementar Seguridad de Datos

Historias de Usuario				
Historia de Usuario	Descripción Historia de Usuarios		Criterios de Aceptación	
12	Como	Administrador	El sistema debe implementar roles y permisos granulares	
	Quiero	Que solo los usuarios autorizados puedan acceder a la información de ventas y pagos,	Los accesos deben estar basados en el principio de mínimo privilegio	Debe existir un proceso formal de autorización para asignar permisos
	Para	Mantener la confidencialidad y evitar accesos indebidos a datos sensibles.	El sistema debe registrar todos los intentos de acceso exitosos y fallidos	Se debe implementar bloqueo automático tras múltiples intentos fallidos

Tabla 14: Historias de Usuario: Restringir Acceso a Datos

Diagramas de flujo

Flujo de Gestión del Administrador

La (Figura 9) ilustra el proceso integral de interacción de un administrador dentro del panel de control, el cual abarcan desde el ingreso inicial hasta la administración de múltiples secciones como lo son las cuentas de usuarios, listas de deseos, carritos de compra, órdenes de pedido, productos, etc. Comienza con la autenticación del administrador, seguida del acceso al área de administración, donde puede elegir una sección específica para supervisar. Incluye capacidades para revisar, modificar información existente, validar, Cancelar, etc. Asimismo, incorpora un paso de verificación previo a la eliminación de cualquier registro y permite la personalización de datos detallados, como los perfiles de usuarios, los artículos dentro de los carritos, las especificaciones de las órdenes y el contenido de las páginas.

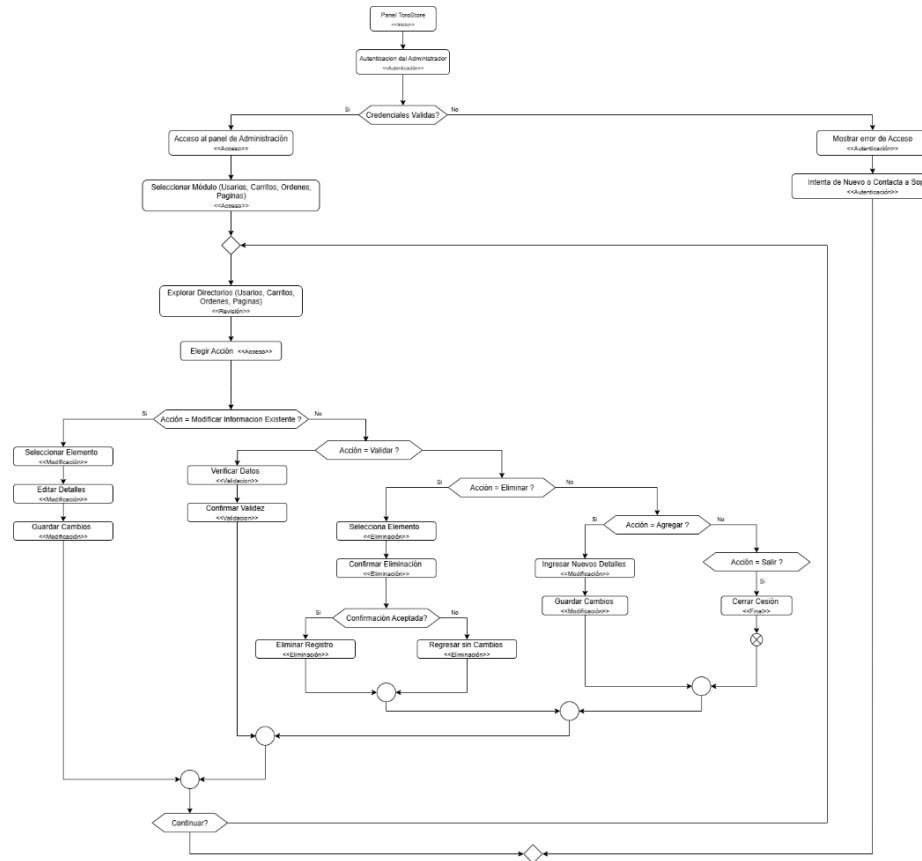


Figura 9. Diagrama _flujo del Administrador.

Flujo de interacción del Usuario

La (Figura 10) ilustra el proceso de interacción del usuario dentro de la tienda virtual, el cual abarcan desde el ingreso inicial hasta la realización de múltiples acciones como lo son registro, autenticación, exploración de productos, gestión de carrito, lista de deseos y ordenes, etc. Iniciando con el ingreso/registro, explorar productos, seleccionar y agregar items al

carrito/lista de deseos, así como también visualizar las ordenes generadas. Además, incluye procesos de edición, actualización y gestión.

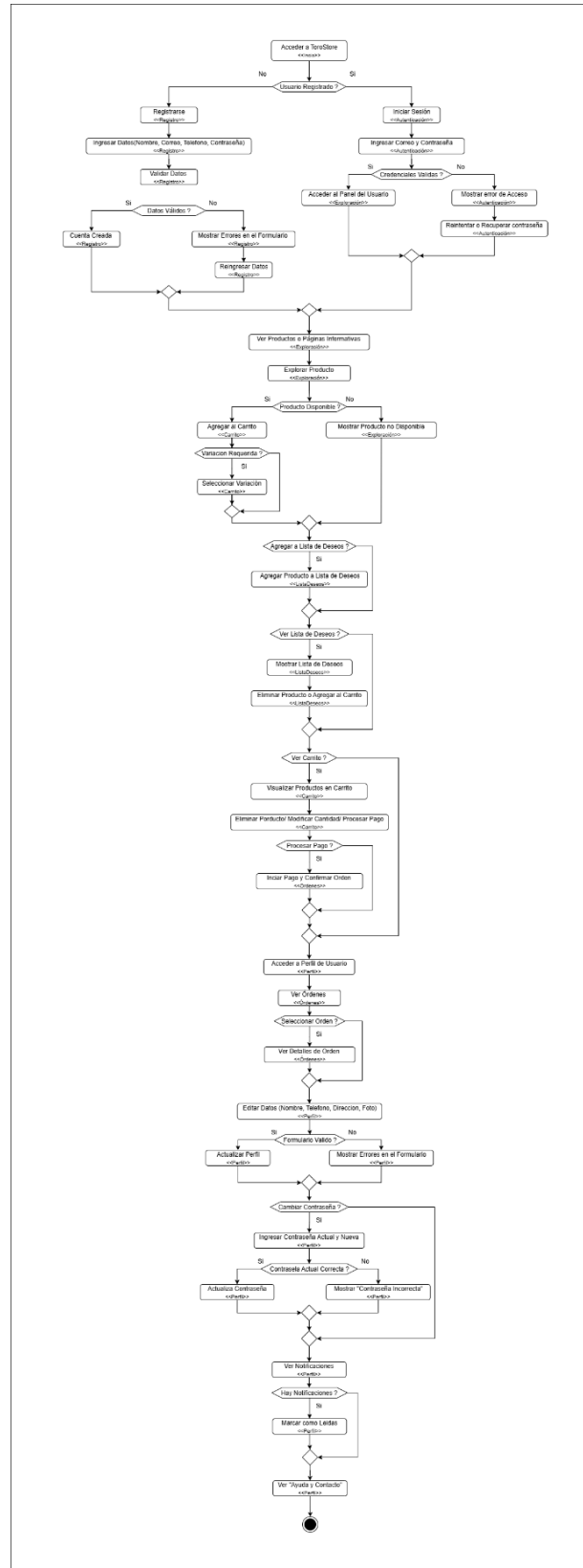


Figura 10 Diagrama flujo Usuario.

Casos de uso

Los casos de uso son la clave para entender cómo las personas interactuarán con el Módulo de Ventas. De esta manera, podemos describir los “subprocesos” realizados por los alumnos, profesores o usuarios, hasta la finalización del proceso de compra y/o cambio. En realidad, siempre estamos hablando de la misma cosa, y no de herramientas separadas, pero esta aproximación es útil para trazar un “mapa” de procesos primarios, encontrar lagunas y asegurarnos de que todo está funcional.

Caso de Uso 1: Autenticación (Figura 11)

Actor Principal: Usuario y Administrador

Precondiciones: El sistema debe estar activo y accesible, con cuentas registradas para Usuario o Administrador.

Flujo Principal:

1. El Usuario o Administrador accede a la página de inicio de sesión.
2. Ingresa su correo electrónico y contraseña.
3. El sistema verifica las credenciales.
4. Se redirige al panel correspondiente (usuario o administrativo) tras una autenticación exitosa.

Flujos Alternativos:

- Si las credenciales son incorrectas, el sistema muestra un mensaje de error y deniega el acceso
- **Postcondiciones:** El Usuario o Administrador queda autenticado y tiene acceso al módulo.

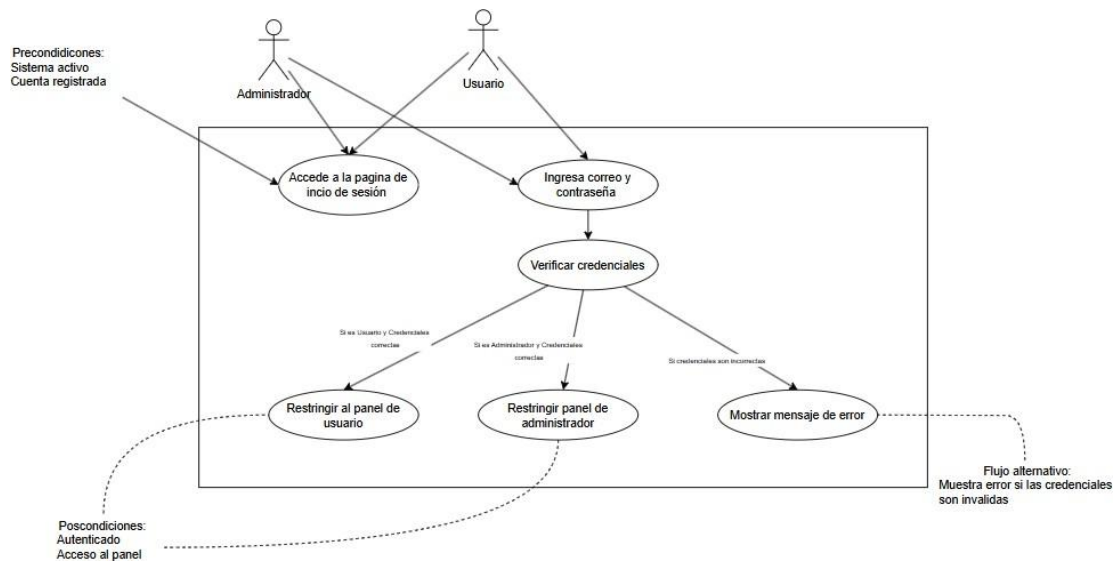


Figura 11 Proceso_Autenticación

Caso de Uso 2: Ordenes (Figura 12)

Actor Principal: Usuario

Precondiciones: El Usuario debe estar autenticado y tener al menos una orden registrada.

Flujo Principal:

1. Accede a "Mi Cuenta" y selecciona "Mis Ordenes".
2. El sistema muestra un historial de órdenes con detalles.
3. El Usuario selecciona una orden para ver detalles.
4. El sistema muestra los detalles de la orden.

Flujos Alternativos:

- Si no hay órdenes, el sistema muestra un mensaje de "Sin Órdenes de compra" y sugiere explorar el catálogo.

Postcondiciones:

El Usuario ve el estado de su compra actualizado.

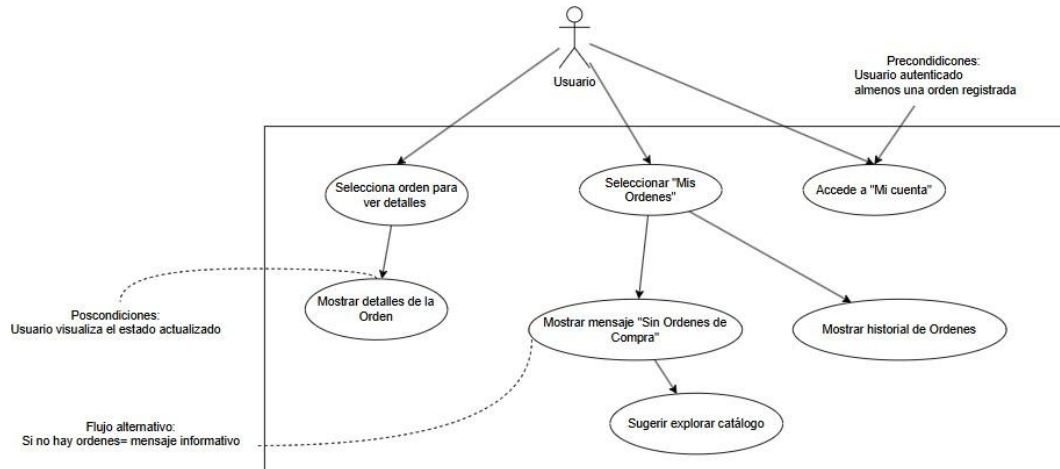


Figura 12 Proceso_Mis Ordenes

Caso de Uso 3: Checkout (Figura 13)

Actor Principal: Usuario

Precondiciones: El Usuario debe estar en la pantalla de Checkout con productos seleccionados.

Flujo Principal:

1. El usuario accede al Checkout
2. El sistema muestra detalles del pedido y datos del usuario
3. El usuario visualiza los datos.
4. El usuario valida datos.

5. El sistema valida datos y continua con el proceso

Flujos Alternativos:

- Si los datos son inválidos, el sistema muestra errores específicos y pide correcciones.
- Si hay campos vacíos, se resalta cada campo faltante con una alerta.

Postcondiciones:

Los datos validados están listos para procesar la compra.

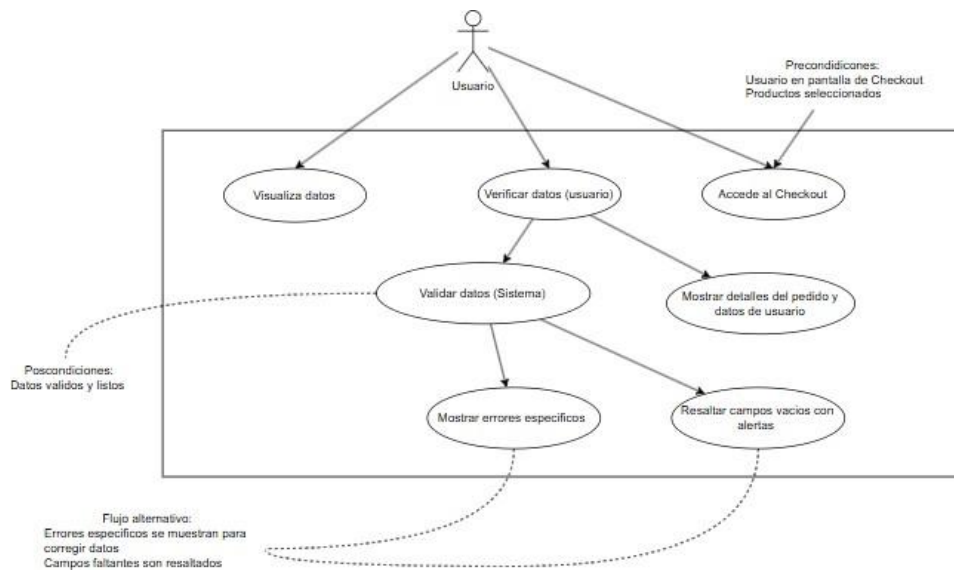


Figura 13 Proceso _ Checkout.

Caso de Uso 4: Pago (Figura 14)

Actor Principal: Usuario

Precondiciones: El Usuario debe estar en la pantalla de pago con datos validados.

Flujo Principal:

1. El Usuario accede a la pantalla de pago.
2. Selecciona un método (PayPal, transferencia, apartado).
3. El sistema muestra las instrucciones o redirige a la pasarela.
4. Confirma la selección.

Flujos Alternativos:

- Si el método no está disponible (ej., PayPal fuera de servicio), se ofrecen alternativas como transferencia.
- Si hay un error en la redirección, se muestra un mensaje para reintentar tras 30 segundos.

Postcondiciones: El método de pago está seleccionado para procesar. Esto asegura **honestidad** al informar opciones, **responsabilidad** al validar pagos, y **lealtad** al facilitar compras.

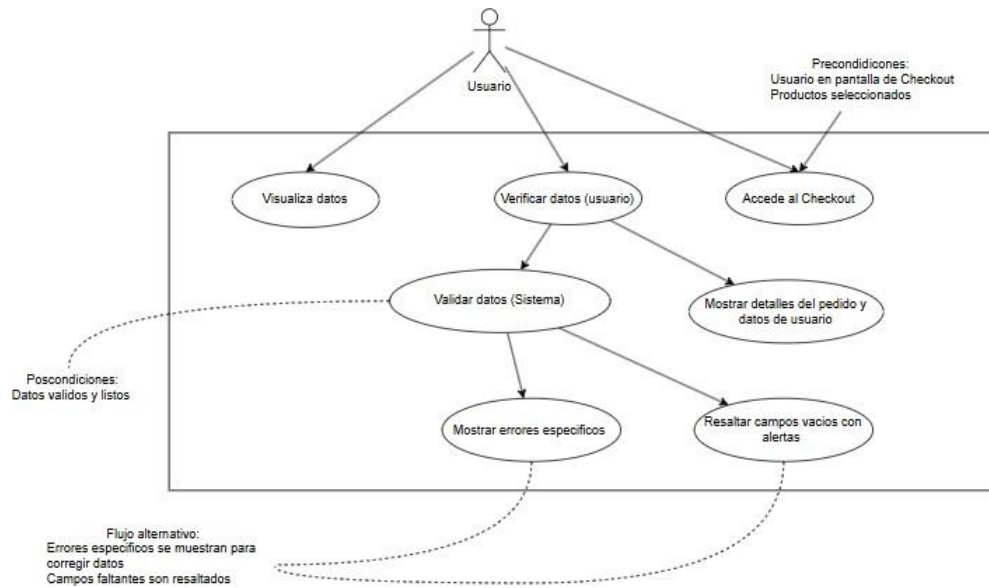


Figura 14 Proceso_Pago

Caso de Uso 5: Panel_admin (Figura 15)

Actor Principal: Administrador

Precondiciones: El Administrador debe estar autenticado y tener acceso al panel administrativo.

Flujo Principal:

1. El Administrador inicia sesión.
2. Accede al panel y selecciona "Visualización de Órdenes".
3. El sistema muestra una tabla con todas las órdenes (ID, estado, fecha).

4. El Administrador valida ordenes

Flujos Alternativos:

- Si no hay órdenes, se muestra un mensaje de "Sin Ordenes"
- Si hay un error de carga, se solicita recargar o revisar la conexión.

Postcondiciones: El Administrador ve la lista de órdenes actualizada. Esto promueve **honestidad** al mostrar datos reales, **responsabilidad** al mantener registros, y **lealtad** al facilitar la gestión.

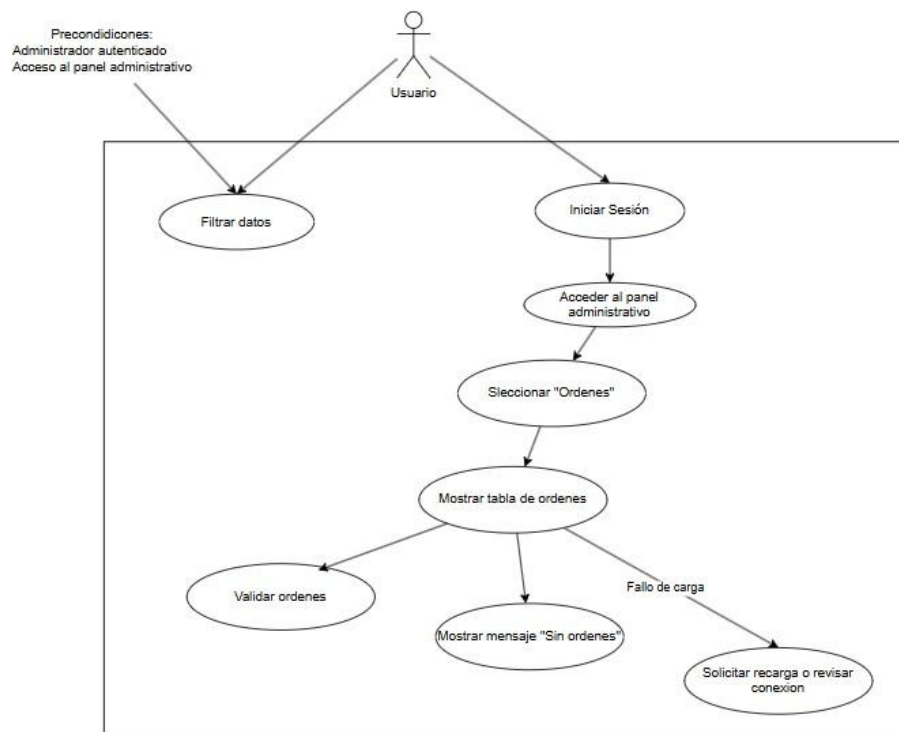


Figura 15 Panel_admin

Caso de Uso 6: Panel_usuario (Figura 16)

Actor Principal: Usuario

Precondiciones: El actor debe estar autenticado, y la orden debe estar en estado "Pendiente" o "New".

Flujo Principal:

1. El Usuario inicia sesión.
2. Accede a la lista de órdenes (panel "Mi Cuenta").
3. Selecciona una orden y elige "Cancelar".
4. El sistema confirma la cancelación y actualiza el estado.

Flujos Alternativos:

- Si hay un error en la actualización, se solicita reintentar tras verificar la conexión.

Postcondiciones: La orden queda cancelada, y el inventario se ajusta. Esto asegura **honestidad** al notificar cambios, **responsabilidad** al gestionar recursos, y **lealtad** al respetar las necesidades.

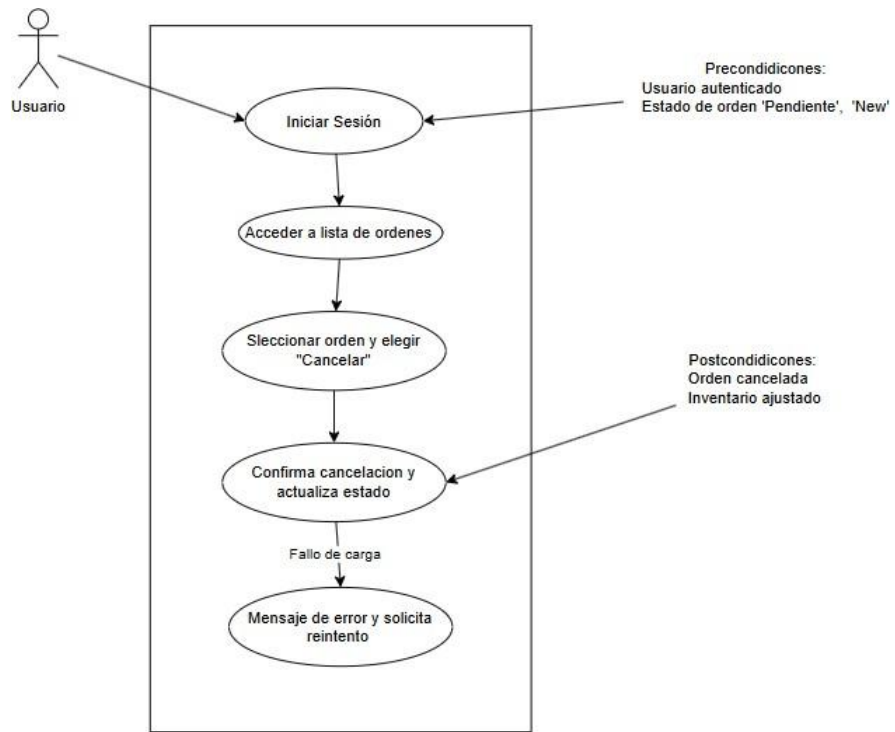


Figura 16 Panel_usuario

Caso de Uso 7: Notificación (Figura 17)

Actor Principal: Usuario y Administrador

Precondiciones: Debe haber un evento relevante (ej., nueva orden, pago confirmado).

Flujo Principal:

1. El sistema detecta un evento (ej., orden procesada).
2. Envía una notificación al Usuario o Administrador (correo o panel).
3. El receptor revisa la notificación.

Flujos Alternativos:

- Si el correo falla, se envía una notificación en el panel y se registra el fallo.

Postcondiciones: El Usuario o Administrador está informado del evento. Esto

asegura **honestidad** al comunicar cambios, **responsabilidad** al garantizar entrega, y

lealtad al mantener contacto.

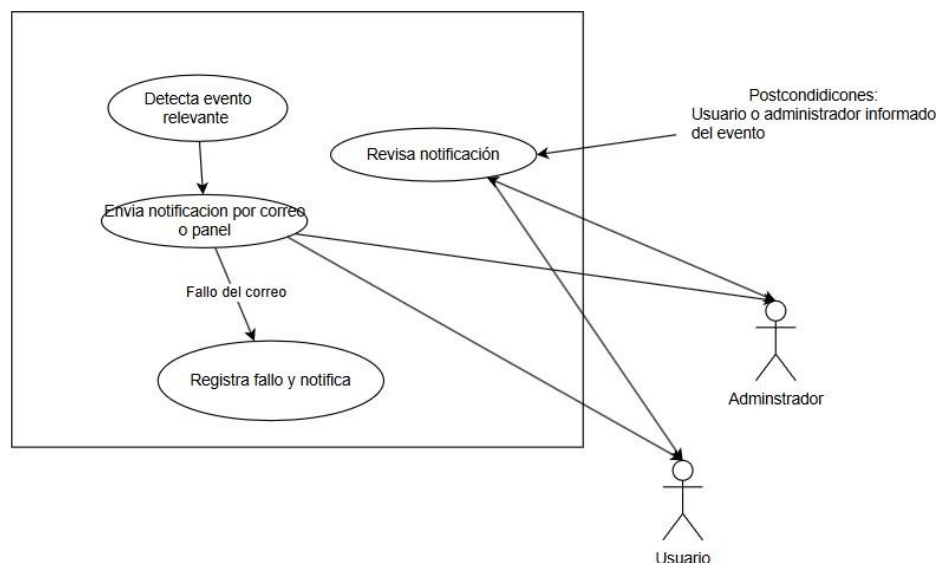


Figura 17 Notificación

Configuración del Ambiente de Desarrollo

Con el fin de hacer una adecuada preparación del ambiente de desarrollo completo a fin de instalar el módulo de ventas, se realizó una acción que inició con la instalación de Python (Figura 18) y la creación de un entorno virtual a fin de aislar las dependencias de los proyectos (Figura 19).

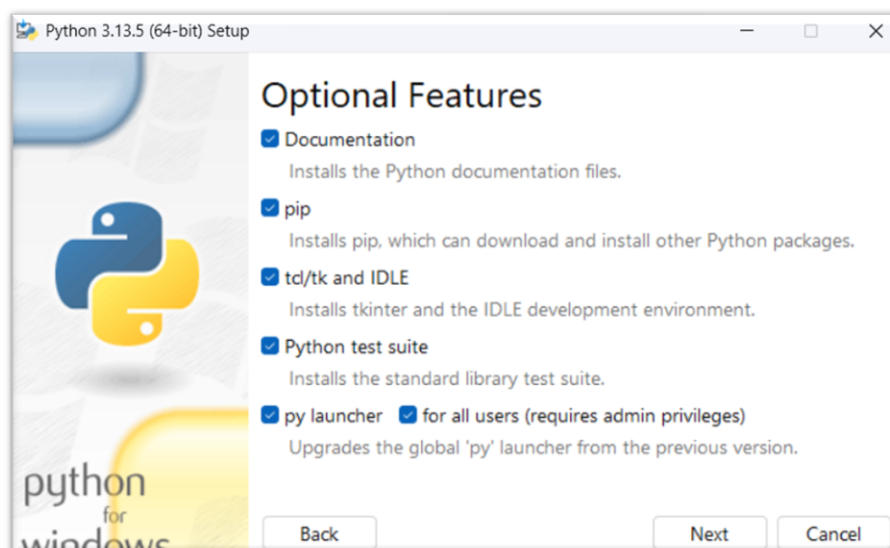
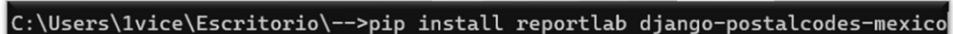


Figura 18 instalación de Python.

```
C:\Users\lvice\Escritorio\-->python -m venv venv|
```

Figura 19 Creación de un entorno virtual

Luego, se instaló Django y unas bibliotecas necesarias como reportlab para la generación de comprobantes en PDF, pillow para el manejo de imágenes, y django-postalcodes-mexico para la gestión de códigos postales, a través del comando `pip install` (Figura 20). Así, por medio de `django-admin startproject`, se creó un proyecto de Django; posterior, se configuraron las aplicaciones claves que serán utilizadas como Orders, Carts, Store, Ecommerce, Carousel y Social, por medio de `python manage.py startapp`, las cuales estarán subordinadas a las funcionalidades del sistema.



```
C:\Users\lvice\Escritorio\-->pip install reportlab django-postalcodes-mexico
```

Figura 20 Instalación de bibliotecas

Para la gestión de datos, en el archivo `settings.py` del proyecto se configuró SQLite como motor de base de datos, logrando simplificar el desarrollo (Figura 21). Después, con `python manage.py makemigrations` y `python manage.py migrate` se generaron las migraciones iniciales y se estructuraron los modelos iniciales necesarios para algunas entidades como los pedidos y productos.

```
# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Figura 21 Configuración SQLite

Creación de aplicaciones:

Para la creación de cada aplicación utilizada se realizó por medio del comando python manage.py startapp *nombre de la app*

- python manage.py startapp Ecommerce
- python manage.py startapp Orders
- python manage.py startapp Social

App Ecommerce

La aplicación ecommerce es el núcleo central de un proyecto de comercio electrónico basado en Django, integrando sub-aplicaciones como cuentas, tienda, carritos, pedidos, categorías, listas de deseos, elementos sociales, páginas estáticas, carruseles, notificaciones y reportes. Ofrece configuraciones clave como soporte WSGI/ASGI, base de datos SQLite, templates con procesadores de contexto personalizados, autenticación segura con modelo de usuario extendido, manejo de sesiones, emails para notificaciones, logging para depuración y vistas para la página principal y un dashboard administrativo para métricas como ganancias,

pedidos e ingresos. Incluye enrutamiento de URLs, manejo de archivos estáticos/media y seguridad con middlewares (CSRF, autenticación), asegurando un sitio escalable y amigable.

A continuación, se resumen los archivos .py principales:

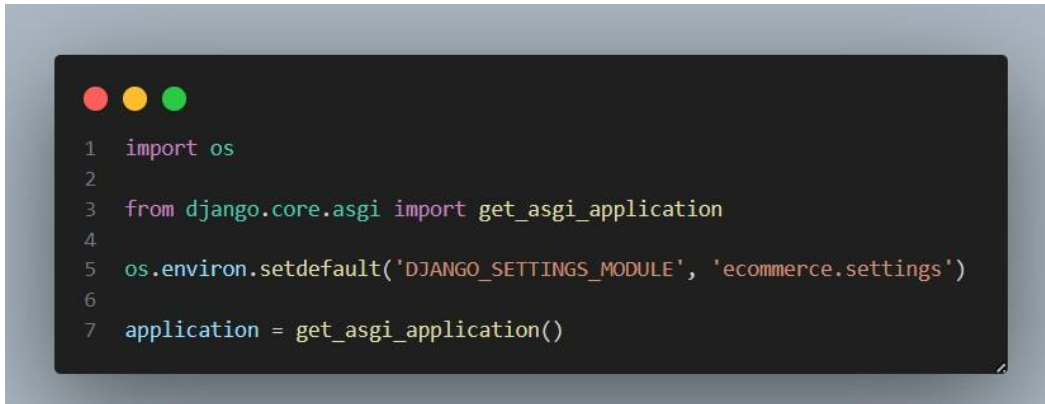
(asgi.py)

Este archivo configura el servidor ASGI (Asynchronous Server Gateway Interface) para el proyecto, permitiendo el manejo de solicitudes asíncronas en Django, útil para características como WebSockets o canales en tiempo real. (Figura 22)

Métodos/funciones clave:

- `get_asgi_application()`: Importado de Django, retorna la aplicación ASGI principal configurada con el módulo de settings del proyecto.
- `os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'ecommerce.settings')`: Establece la variable de entorno para cargar los settings de Django si no está definida.

Propósito: Exponer la aplicación como un callable ASGI nombrado `application`, facilitando el despliegue en servidores como Daphne o Uvicorn para entornos asíncronos. Es esencial para producción en setups modernos, aunque el proyecto también soporta WSGI síncrono.



```
1 import os
2
3 from django.core.asgi import get_asgi_application
4
5 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'ecommerce.settings')
6
7 application = get_asgi_application()
```

Figura 22 Ecommerce_asgi.py

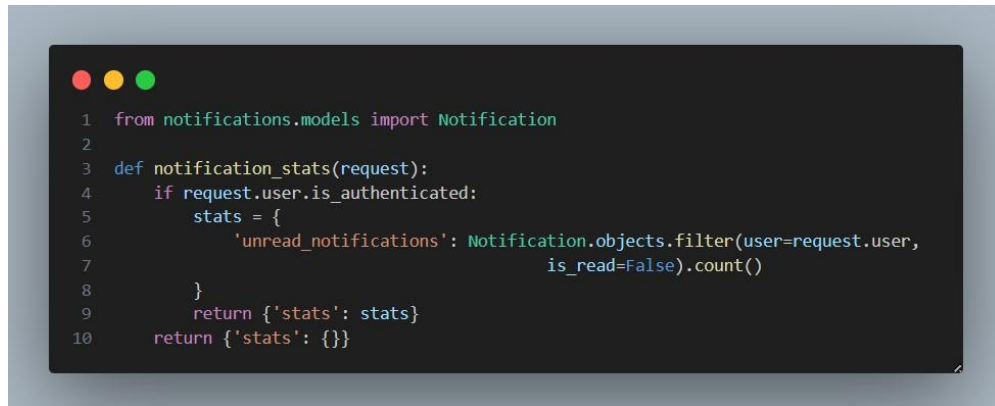
context_processors.py

Define procesadores de contexto personalizados que inyectan datos globales en todas las plantillas, como estadísticas de notificaciones no leídas para usuarios autenticados. (Figura 23)

Métodos/funciones clave:

- `notification_stats(request)`: Verifica si el usuario está autenticado; si lo está, cuenta las notificaciones no leídas del modelo `Notification` y las retorna en un diccionario `stats`. Si no, retorna un diccionario vacío.

Propósito: Proporcionar datos dinámicos (como contadores de notificaciones) disponibles en cualquier template sin repetir código en vistas, mejorando el modularidad y la experiencia del usuario al mostrar alertas en tiempo real.

A screenshot of a code editor window with a dark background and light-colored text. The code is Python and is part of a file named `Ecommerce_context_processors.py`. It defines a function `notification_stats(request)` that checks if a user is authenticated and returns a dictionary with notification statistics. The code is as follows:

```
1 from notifications.models import Notification
2
3 def notification_stats(request):
4     if request.user.is_authenticated:
5         stats = {
6             'unread_notifications': Notification.objects.filter(user=request.user,
7                                                                 is_read=False).count()
8         }
9     return {'stats': stats}
10    return {'stats': {}}
```

Figura 23 *Ecommerce_context_processors.py*

settings.py

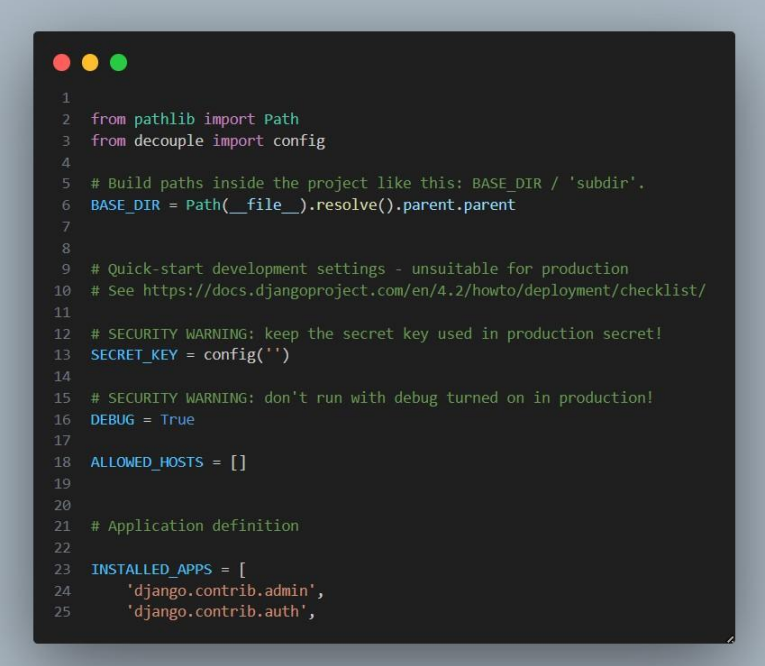
Archivo central de configuración de Django que define variables globales para el proyecto, como apps instaladas, middleware, base de datos, templates, autenticación, emails y logging. (Figura 24)

Campos/claves principales:

- **SECRET_KEY**: Clave secreta para seguridad (cargada desde env con decouple).
- **INSTALLED_APPS**: Lista de apps Django y personalizadas (e.g., 'store', 'carts', 'orders').
- **MIDDLEWARE**: Lista de middlewares para manejo de sesiones, CSRF, autenticación y timeouts.

- **TEMPLATES:** Configuración de templates, incluyendo context processors personalizados.
- **DATABASES:** Config de SQLite por defecto.
- **AUTH_USER_MODEL:** Modelo de usuario personalizado ('accounts.Account').
- **EMAIL_*:** Configuración SMTP para emails (usando Gmail).
- **LOGGING:** Configuración de loggers para vistas y forms en 'store', con handlers para consola y archivo.

Propósito: Centralizar todas las configuraciones del proyecto para facilitar el mantenimiento, seguridad y escalabilidad. Define comportamientos como timeouts de sesión (e.g., 1 hora estándar, 14 días extendido), rutas de archivos estáticos/media, y tags para mensajes de error.



```

1
2 from pathlib import Path
3 from decouple import config
4
5 # Build paths inside the project like this: BASE_DIR / 'subdir'.
6 BASE_DIR = Path(__file__).resolve().parent.parent
7
8
9 # Quick-start development settings - unsuitable for production
10 # See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/
11
12 # SECURITY WARNING: keep the secret key used in production secret!
13 SECRET_KEY = config('')
14
15 # SECURITY WARNING: don't run with debug turned on in production!
16 DEBUG = True
17
18 ALLOWED_HOSTS = []
19
20
21 # Application definition
22
23 INSTALLED_APPS = [
24     'django.contrib.admin',
25     'django.contrib.auth',

```

Figura 24 Ecommerce _settings.py

urls.py

Define las rutas URL principales del proyecto, mapeando patrones a vistas o incluyendo URLs de otras apps. (Figura 25)

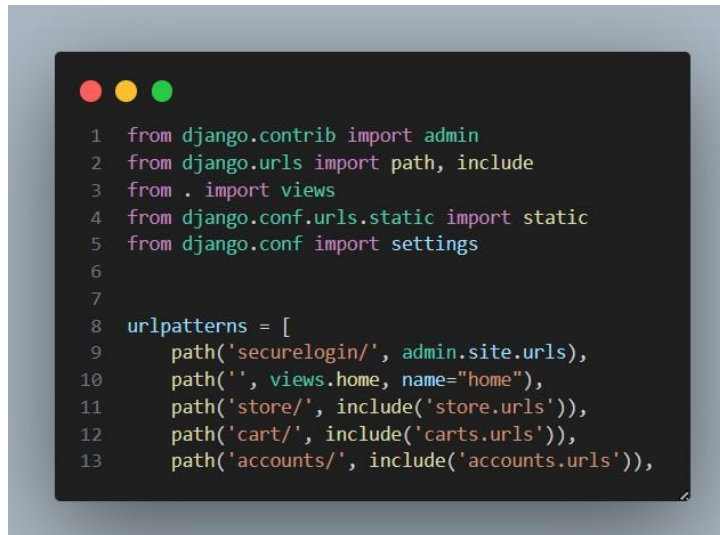
Campos/claves principales:

- `urlpatterns`: Lista de paths, incluyendo admin, vistas raíz y includes de apps como 'store', 'carts', etc.

Métodos/funciones clave:

- `path()`: Define rutas específicas, e.g., `path("", views.home, name="home")` para la página principal.
- `include()`: Incluye URLs de sub-apps, e.g., `path('store/', include('store.urls'))`.
- `static()`: Sirve archivos media en desarrollo.

Propósito: Actuar como enrutador central, conectando URLs a vistas correspondientes y habilitando el servicio de archivos estáticos/media durante el desarrollo, asegurando una navegación estructurada.



```
1 from django.contrib import admin
2 from django.urls import path, include
3 from . import views
4 from django.conf.urls.static import static
5 from django.conf import settings
6
7
8 urlpatterns = [
9     path('securelogin/', admin.site.urls),
10    path('', views.home, name="home"),
11    path('store/', include('store.urls')),
12    path('cart/', include('carts.urls')),
13    path('accounts/', include('accounts.urls')),
```

Figura 25 Ecommerce_URLS.py

views.py

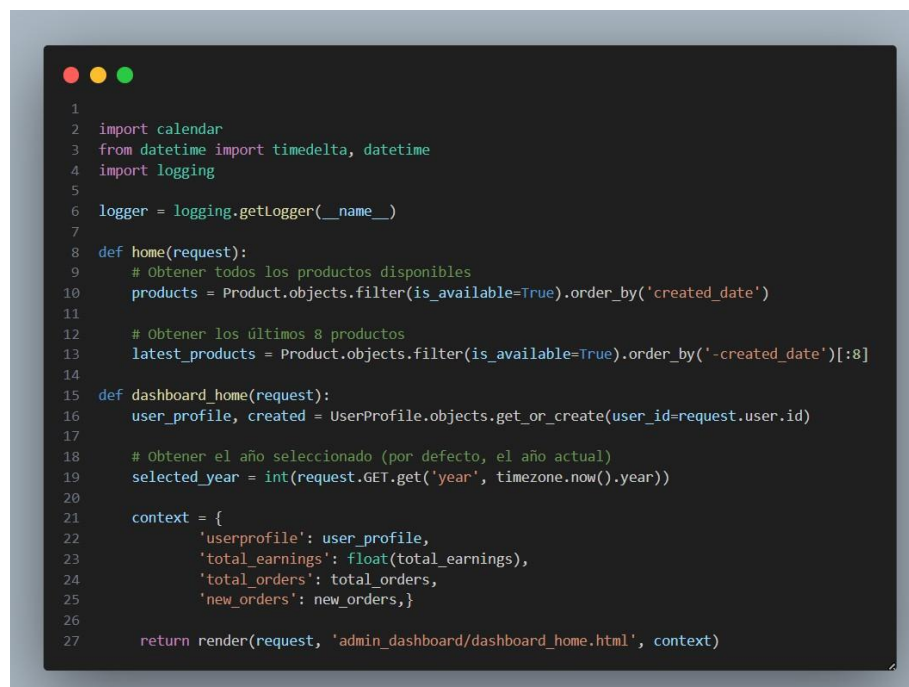
Contiene vistas para la página principal (home) y el dashboard administrativo, manejando lógica de negocio como obtener productos, reviews, carruseles y métricas de ventas. (Figura 26)

Métodos/funciones clave:

- `home(request)`: Obtiene productos disponibles, últimos productos, carruseles y reviews; asocia reviews por producto y renderiza 'home.html'.
- `dashboard_home(request)`: Restringida a staff; calcula métricas como ganancias, pedidos, clientes, ingresos por mes/año, comparaciones, ventas totales y pedidos

recientes; renderiza 'admin_dashboard/dashboard_home.html' con datos para gráficos y tablas.

Propósito: Procesar solicitudes HTTP, consultar modelos (e.g., Product, Order, Account) y preparar contextos para templates. La vista home enfoca la experiencia del usuario final, mientras que dashboard_home proporciona analítica para admins, usando logging para depuración.



```
1
2 import calendar
3 from datetime import timedelta, datetime
4 import logging
5
6 logger = logging.getLogger(__name__)
7
8 def home(request):
9     # Obtener todos los productos disponibles
10    products = Product.objects.filter(is_available=True).order_by('created_date')
11
12    # Obtener los últimos 8 productos
13    latest_products = Product.objects.filter(is_available=True).order_by('-created_date')[:8]
14
15 def dashboard_home(request):
16     user_profile, created = UserProfile.objects.get_or_create(user_id=request.user.id)
17
18     # Obtener el año seleccionado (por defecto, el año actual)
19     selected_year = int(request.GET.get('year', timezone.now().year))
20
21     context = {
22         'userprofile': user_profile,
23         'total_earnings': float(total_earnings),
24         'total_orders': total_orders,
25         'new_orders': new_orders,}
26
27     return render(request, 'admin_dashboard/dashboard_home.html', context)
```

Figura 26 Ecommerce _Views.py

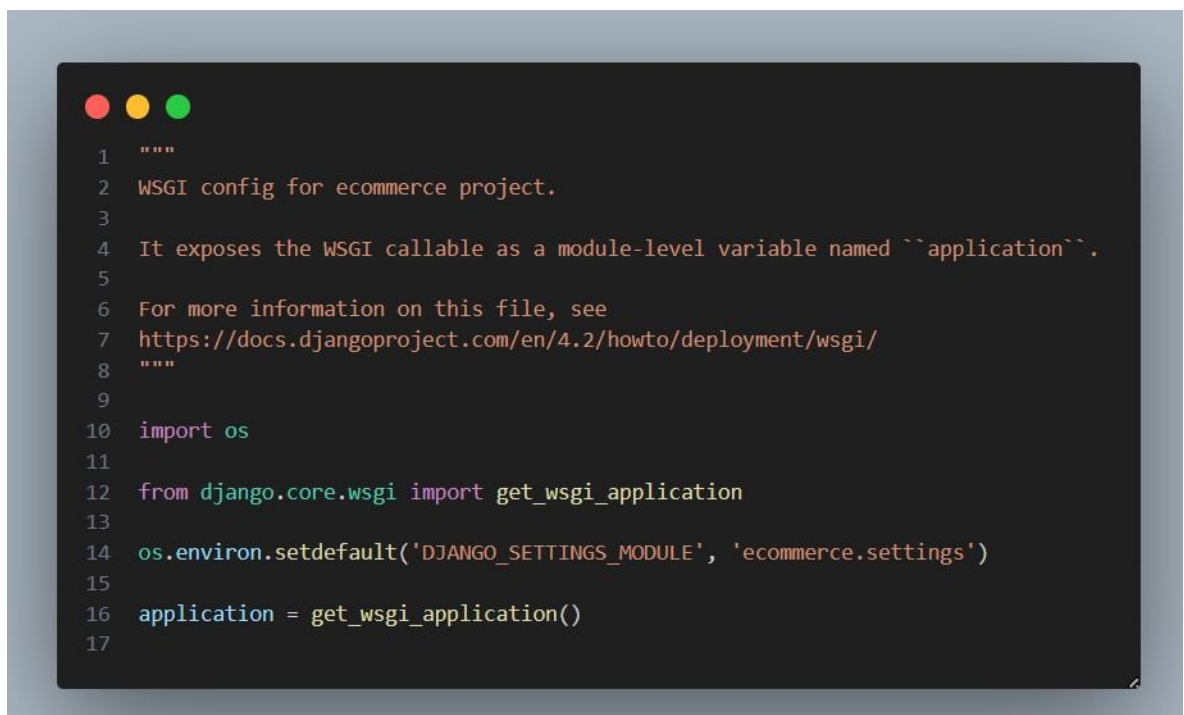
wsgi.py

Configura el servidor WSGI (Web Server Gateway Interface) para el proyecto, permitiendo el manejo de solicitudes síncronas en entornos como Gunicorn o Apache. (Figura 27)

Métodos/funciones clave:

- `get_wsgi_application()`: Importado de Django, retorna la aplicación WSGI principal configurada con los settings.
- `os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'ecommerce.settings')`: Establece la variable de entorno para los settings si no está definida.

Propósito: Exponer la aplicación como un callable WSGI nombrado `application`, facilitando el despliegue en servidores síncronos. Es el counterpart de ASGI para compatibilidad con setups tradicionales.



```
1  """
2  WSGI config for ecommerce project.
3
4  It exposes the WSGI callable as a module-level variable named ``application``.
5
6  For more information on this file, see
7  https://docs.djangoproject.com/en/4.2/howto/deployment/wsgi/
8  """
9
10 import os
11
12 from django.core.wsgi import get_wsgi_application
13
14 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'ecommerce.settings')
15
16 application = get_wsgi_application()
17
```

Figura 27 `Ecommerce_wsgi.py`

App Orders

La aplicación orders gestiona todo el proceso de pedidos en el proyecto de comercio electrónico, desde la creación y validación hasta la cancelación, entrega y generación de documentos como bauchers. Permite a los usuarios colocar pedidos, seleccionar métodos de pago, subir comprobantes y rastrear el estado de sus órdenes, con notificaciones y correos automáticos para cada etapa. Incluye un sistema administrativo para validar pagos, cancelar o eliminar órdenes, y monitorear estadísticas como órdenes pendientes o validadas. Utiliza tareas en segundo plano para cancelar pedidos expirados, soporta formularios personalizados para datos de envío, y genera PDFs con detalles de transacciones usando ReportLab, asegurando un flujo de trabajo eficiente y seguro con transacciones atómicas y validaciones estrictas. A continuación, se resumen los archivos .py principales de la aplicación orders:

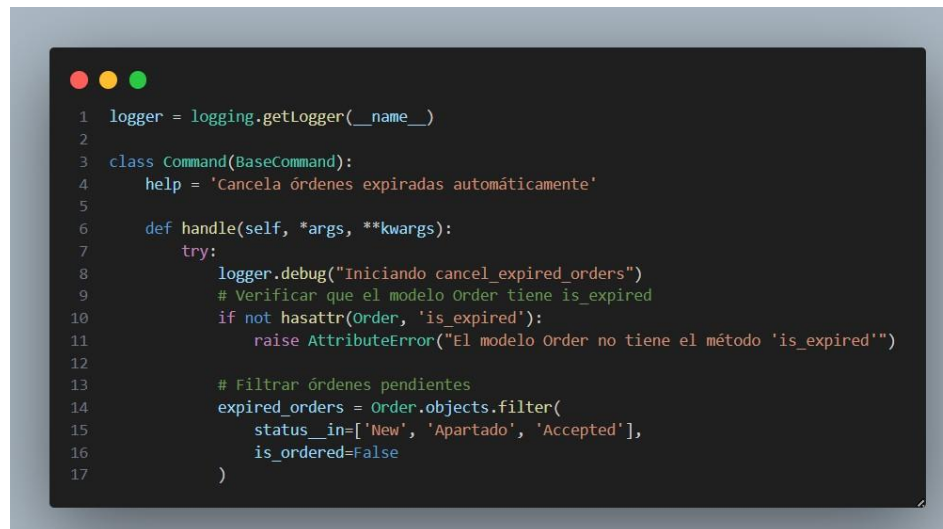
management/commands/cancel_expired_orders.py

Comando de gestión Django que cancela automáticamente órdenes expiradas basadas en su estado y tiempo de creación, restaurando stock y enviando correos de notificación. (Figura 28)

Métodos/funciones clave:

- `handle(self, *args, **kwargs)`: Filtra órdenes pendientes, verifica si están expiradas usando `order.is_expired()`, cancela las que cumplen (actualiza estado a 'Cancelled', restaura stock vía señal), envía correo HTML de cancelación y cuenta las canceladas; maneja errores con logging y transacciones atómicas.

Propósito: Automatizar la limpieza de órdenes no completadas (e.g., transferencias sin comprobante después de 24 horas o apartados después de 72 horas), asegurando que el stock se libere y los usuarios sean notificados, mejorando la gestión de inventario y la experiencia del usuario.

A screenshot of a code editor with a dark background and light-colored text. The code is written in Python and defines a Django command class. It includes a logger, a class definition for 'Command' inheriting from 'BaseCommand', and a 'handle' method. The 'handle' method contains logic to filter expired orders based on status and whether they are ordered.

```
1 logger = logging.getLogger(__name__)
2
3 class Command(BaseCommand):
4     help = 'Cancela órdenes expiradas automáticamente'
5
6     def handle(self, *args, **kwargs):
7         try:
8             logger.debug("Iniciando cancel_expired_orders")
9             # Verificar que el modelo Order tiene is_expired
10            if not hasattr(Order, 'is_expired'):
11                raise AttributeError("El modelo Order no tiene el método 'is_expired'")
12
13            # Filtrar órdenes pendientes
14            expired_orders = Order.objects.filter(
15                status__in=['New', 'Apartado', 'Accepted'],
16                is_ordered=False
17            )
```

Figura 28 Orders _wsgi.py

admin.py

Registra modelos de la app en el panel admin de Django, con personalizaciones como inlines, campos calculados, filtros y fieldsets para una interfaz administrativa intuitiva. (Figura 29)

Campos/claves principales:

- `list_display`: Listas de campos visibles en vistas de lista (e.g., `'order_number'`, `'full_name'` para `Order`).
- `fieldsets`: Grupos de campos organizados (e.g., 'Información del cliente', 'Dirección' para `OrderAdmin`).

Métodos/funciones clave:

- `municipality(self, obj)`: Retorna el municipio del objeto `Order`.
- `get_state_display(self, obj)`: Retorna el display legible del estado.
- `full_address(self, obj)`: Formatea y retorna la dirección completa como HTML.
- Clases como `OrderAdmin`, `PaymentAdmin`, `OrderProductAdmin`: Extienden `ModelAdmin` con inlines (e.g., `OrderProductInline`), filtros, búsquedas y campos `readonly`.

Propósito: Facilitar la administración de pagos, órdenes y productos de órdenes en el backend, permitiendo a admins ver y editar datos clave como direcciones completas o estados, con optimizaciones para usabilidad como paginación y filtros.

```

1  from django.contrib import admin
2
3
4  class OrderProductInline(admin.TabularInline):
5      model = OrderProduct
6      readonly_fields = ('user', 'product', 'variation', 'quantity', 'product_price', 'ordered', 'created_at', 'updated_at')
7      extra = 0
8
9
10 class OrderAdmin(admin.ModelAdmin):
11     def municipality(self, obj):
12         return obj.municipality
13
14
15
16 class PaymentAdmin(admin.ModelAdmin):
17     list_display = ['payment_id', 'user', 'payment_method', 'transaction_type', 'amount', 'status', 'created_at']
18     list_display_links = ['payment_id']
19     list_filter = ['payment_method', 'transaction_type', 'status']
20     search_fields = ['payment_id', 'user_email', 'reference_number']
21     list_per_page = 20
22
23 class OrderProductAdmin(admin.ModelAdmin):
24     list_display = ['order', 'product', 'variation', 'quantity', 'product_price', 'ordered', 'created_at']
25     list_display_links = ['order', 'product']
26     list_filter = ['ordered']
27     search_fields = ['order__order_number', 'product__product_name']

```

Figura 29 Orders _ admin.py

app.py

Configuración de la app Django, estableciendo el campo auto por defecto y iniciando tareas en segundo plano al cargar la app. (Figura 30)

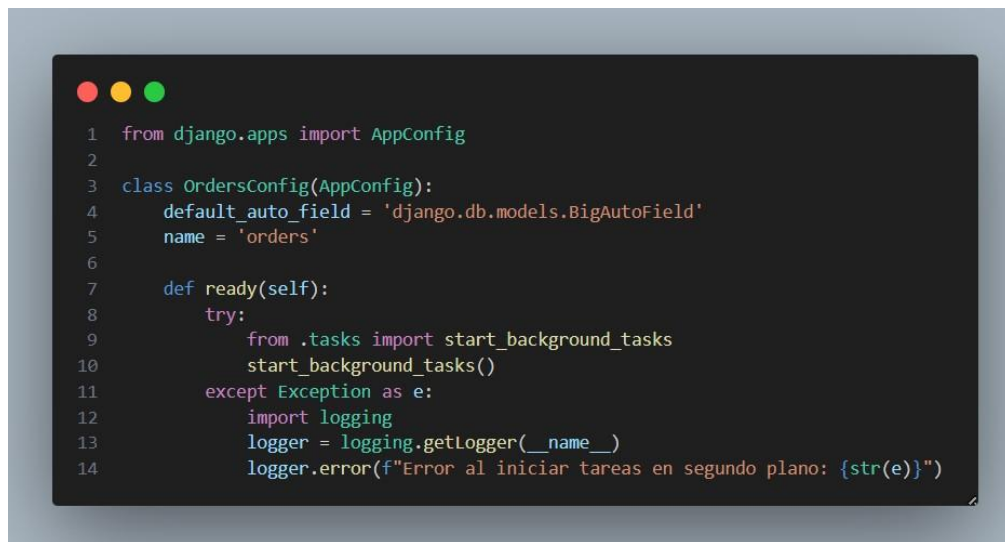
Campos/claves principales:

- default_auto_field: 'django.db.models.BigAutoField'.
- name: 'orders'.

Métodos/funciones clave:

- `ready(self)`: Intenta importar y ejecutar `start_background_tasks` de `tasks.py` para iniciar hilos de fondo; captura y loguea errores.

Propósito: Inicializar la app y asegurar que las tareas programadas (como cancelación de órdenes expiradas) se lancen automáticamente al inicio del servidor, promoviendo operaciones asíncronas sin intervención manual.



```
1 from django.apps import AppConfig
2
3 class OrdersConfig(AppConfig):
4     default_auto_field = 'django.db.models.BigAutoField'
5     name = 'orders'
6
7     def ready(self):
8         try:
9             from .tasks import start_background_tasks
10            start_background_tasks()
11        except Exception as e:
12            import logging
13            logger = logging.getLogger(__name__)
14            logger.error(f"Error al iniciar tareas en segundo plano: {str(e)}")
```

Figura 30 `Orders_app.py`

forms.py

Define un formulario para crear órdenes, con campos para datos personales, dirección y notas, incluyendo validaciones personalizadas para teléfono y código postal. (Figura 31)

Campos/claves principales:

- Campos como `first_name`, `last_name`, `phone` (con pattern para 10 dígitos), `email`, `street_address`, `neighborhood`, `zip_code` (con pattern para 5 dígitos), `use_saved_address`.

Métodos/funciones clave:

- `_init_(self, *args, **kwargs)`: Añade clases CSS a widgets de campos (excepto checkbox).
- `clean(self)`: Valida datos limpios; hace campos de dirección obligatorios si no se usa guardada, verifica formatos de teléfono y `zip_code` con regex, y añade errores si fallan.

Propósito: Proporcionar un formulario seguro y user-friendly para recopilar información de órdenes durante el checkout, asegurando datos válidos (e.g., formatos mexicanos para teléfono y CP) y opción de reutilizar perfiles guardados para eficiencia.

```

1 from django import forms
2 from .models import Order
3 import re
4
5 class OrderForm(forms.Form):
6     # Campos de Order
7     first_name = forms.CharField(
8         max_length=50,
9         label='Nombre(s)',
10        widget=forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Nombre(s)})
11    )
12    last_name = forms.CharField(
13        max_length=50,
14        label='Apellidos',
15        widget=forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Apellidos'})
16    )
17    phone = forms.CharField(
18        max_length=15,
19        label='Teléfono',
20        help_text='Debe tener 10 dígitos (ejemplo: 5512345678).',
21        widget=forms.TextInput(attrs={
22            'class': 'form-control',
23            'pattern': r'^\d{10}$',
24            'placeholder': 'Ejemplo: 5512345678'
25        })
26    def __init__(self, *args, **kwargs):
27        super(OrderForm, self).__init__(*args, **kwargs)
28        for field in self.fields:
29            if field != 'use_saved_address':
30                self.fields[field].widget.attrs['class'] = 'form-control'

```

Figura 31 Orders _forms.py

models.py

Define modelos para pagos, órdenes, productos de órdenes y detalles bancarios, con validaciones, choices para estados/métodos, y señales para manejar cancelaciones. (Figura 32)

Campos/claves principales:

- En Payment: payment_method (choices como 'PayPal'), status (choices como 'Pending'), amount, proof (FileField).
- En Order: status (choices como 'New'), state (choices de estados mexicanos), order_total, zip_code, neighborhood.
- En OrderProduct: quantity, product_price, ordered.
- En BankDetails: bank_name, account_number, clabe.

Métodos/funciones clave:

- is_expired(self) (en Order): Verifica si la orden ha expirado basado en método de pago y tiempo transcurrido.
- full_address(self) (en Order): Concatena partes de dirección en una cadena legible.
- restore_stock(self) (en OrderProduct): Restaura stock al producto o variación usando transacciones atómicas.
- handle_order_cancellation (señal receiver): Restaura stock de todos OrderProduct al cambiar Order a 'Cancelled'.
- clean(self): Valida campos como usuario activo, formatos de payment_id/order_number/zip_code con regex.

Propósito: Modelar la estructura de datos para el flujo de órdenes, asegurando integridad con validaciones (e.g., stock disponible), lógica de expiración y automatización

de acciones como restauración de inventario al cancelar, integrándose con otros apps como accounts y store.

```
1  django.db import models
2  class Order(models.Model):
3      STATUS = (
4          ("New", "Nuevo"),
5          ("Accepted", "Aceptado"),
6          ("Completed", "Completado"),
7          ("Cancelled", "Cancelado"),
8          ("Apartado", "Apartado"),
9      )
10
11  def is_expired(self):
12      if self.payment:
13          now = timezone.now()
14          time_diff = now - self.created_at
15          if self.status == 'New':
16              if self.payment.payment_method == 'BankTransfer' and not self.payment.proof:
17                  return time_diff > timedelta(hours=24) # 24 horas
18              if self.payment.payment_method == 'Apartado':
19                  return time_diff > timedelta(hours=72) # 72 horas
20          return False
21
22  class OrderProduct(models.Model):
23      order = models.ForeignKey(Order, on_delete=models.CASCADE)
24      user = models.ForeignKey(Account, on_delete=models.CASCADE)
25
26  def handle_order_cancellation(sender, instance, **kwargs):
27      """Restaura el stock de los OrderProduct asociados cuando un Order cambia a status='Cancelled'."""
28      if instance.status == "Cancelled":
29          with transaction.atomic():
30              for order_item in instance.orderproduct_set.all():
31                  order_item.restore_stock()
32                  order_item.ordered = False
33                  order_item.save()
```

Figura 32 Orders _models.py

tasks.py

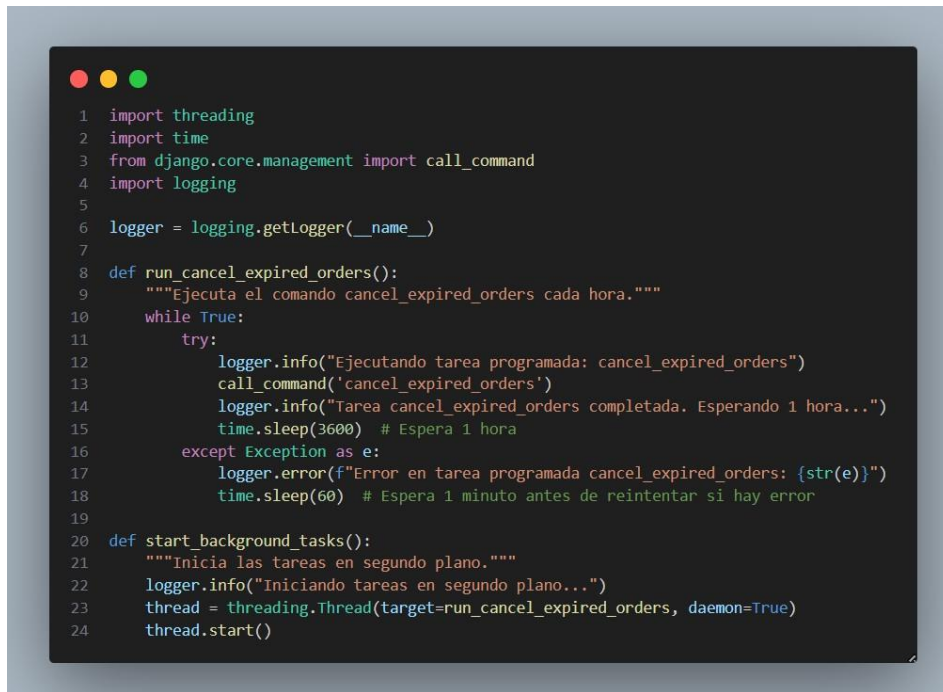
Implementa tareas en segundo plano para ejecutar comandos periódicos, como cancelar órdenes expiradas cada hora. (Figura 33)

Campos/claves principales: No aplica (es un módulo de funciones).

Métodos/funciones clave:

- `run_cancel_expired_orders()`: Bucle infinito que llama al comando 'cancel_expired_orders' cada 3600 segundos (1 hora), con logging de errores y reintentos.
- `start_background_tasks()`: Inicia un hilo daemon para ejecutar `run_cancel_expired_orders`.

Propósito: Habilitar operaciones automatizadas y asíncronas sin bloquear el servidor principal, asegurando que órdenes expiradas se procesen regularmente para mantener el sistema actualizado.



```

1  import threading
2  import time
3  from django.core.management import call_command
4  import logging
5
6  logger = logging.getLogger(__name__)
7
8  def run_cancel_expired_orders():
9      """Ejecuta el comando cancel_expired_orders cada hora."""
10     while True:
11         try:
12             logger.info("Ejecutando tarea programada: cancel_expired_orders")
13             call_command('cancel_expired_orders')
14             logger.info("Tarea cancel_expired_orders completada. Esperando 1 hora...")
15             time.sleep(3600) # Espera 1 hora
16         except Exception as e:
17             logger.error(f"Error en tarea programada cancel_expired_orders: {str(e)}")
18             time.sleep(60) # Espera 1 minuto antes de reintentar si hay error
19
20 def start_background_tasks():
21     """Inicia las tareas en segundo plano."""
22     logger.info("Iniciando tareas en segundo plano...")
23     thread = threading.Thread(target=run_cancel_expired_orders, daemon=True)
24     thread.start()

```

Figura 33 `Orders_tasks.py`

Urls.py

Define rutas URL para vistas de la app, incluyendo acciones de usuario (e.g., colocar orden) y admin (e.g., validar órdenes). (Figura 34)

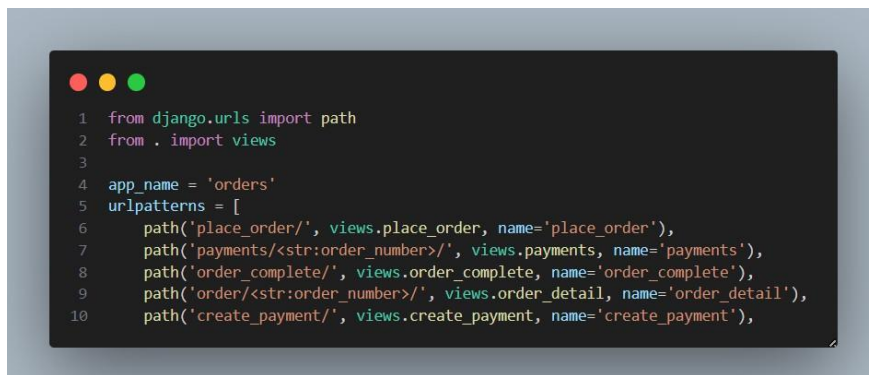
Campos/claves principales:

- `urlpatterns`: Lista de paths con names como 'place_order', 'admin_validate_orders'.

Métodos/funciones clave:

- `path()`: Mapea URLs a vistas (e.g., 'place_order/' a `views.place_order`).

Propósito: Enrutar solicitudes HTTP a las vistas correspondientes, separando lógica de usuario y admin, y facilitando navegación como 'orders:order_detail' para URLs nombradas.



```
1 from django.urls import path
2 from . import views
3
4 app_name = 'orders'
5 urlpatterns = [
6     path('place_order/', views.place_order, name='place_order'),
7     path('payments/<str:order_number>/', views.payments, name='payments'),
8     path('order_complete/', views.order_complete, name='order_complete'),
9     path('order/<str:order_number>/', views.order_detail, name='order_detail'),
10    path('create_payment/', views.create_payment, name='create_payment'),
```

Figura 34 Orders_urls.py

views.py

Contiene vistas para manejar el flujo de órdenes, desde checkout y pagos hasta vistas admin para validación, cancelación y generación de PDFs, con logging, transacciones y notificaciones. (Figura 35)

Métodos/funciones clave:

- `validate_zip_code(request)`: Valida código postal contra base de datos mexicana, retornando JSON con estado, municipio y colonias.
- `payments(request, order_number)`: Procesa pagos POST (crea Payment, OrderProduct, envía correos), o renderiza template GET; maneja métodos como BankTransfer.
- `place_order(request)`: Maneja formulario de orden, valida stock, crea Order y redirige a pagos; usa perfil guardado si aplica.
- `order_complete(request)`: Muestra confirmación de orden con productos y subtotal.
- `order_detail(request, order_number)`: Muestra detalles de orden, permite subir comprobante de pago.
- `cancel_order(request, order_number)`: Cancela orden de usuario, restaura stock.
- `admin_cancel_order(request, order_number)`: Cancela orden desde admin, notifica y envía correo.

- `admin_canceled_orders(request)`: Lista órdenes canceladas con paginación y estadísticas.
- `admin_delete_validated_order(request, order_number)`: Elimina orden validada desde admin.
- `generate_bank_transfer_pdf(request)`: Genera PDF con detalles de pago usando ReportLab.
- `user_orders(request)`: Lista órdenes de usuario, permite subir comprobantes.
- `admin_validate_orders(request)`: Valida órdenes pendientes desde admin, actualiza stock y envía notificaciones/correos.
- `mark_order_delivered(request, order_number)`: Marca orden como entregada, notifica y envía correo.

Propósito: Procesar solicitudes HTTP para todo el ciclo de vida de órdenes, integrando lógica de negocio como validaciones, emails, PDFs y notificaciones, con decoradores para autenticación y roles (e.g., `staff_member_required`), asegurando un flujo seguro y eficiente.

```

1 from django.shortcuts import render, redirect, get_object_or_404
2 from django.http import JsonResponse, HttpResponse
3 from django.contrib.auth.decorators import login_required
4 from django.db import transaction
5 from django.core.mail import EmailMessage
6
7 def handle_order_cancellation(sender, instance, **kwargs):
8     """Restaura el stock de los OrderProduct asociados cuando un Order cambia a status='Cancelled'."""
9     if instance.status == "Cancelled":
10         with transaction.atomic():
11             for order_item in instance.orderproduct_set.all():
12                 order_item.restore_stock()
13                 order_item.ordered = False
14                 order_item.save()
15
16 def place_order(request, total=0, quantity=0):
17     current_user = request.user
18     cart_items = CartItem.objects.filter(user=current_user, is_active=True, is_processed=False)
19     cart_count = cart_items.count()
20
21     if cart_count <= 0:
22         messages.info(request, "Tu carrito está vacío.")
23         return redirect('store')
24
25 def cancel_order(request, order_number):
26     try:
27         order = get_object_or_404(Order, order_number=order_number, user=request.user)
28
29         if order.status in ['Confirmed', 'Canceled', 'Completed', 'Delivered']:
30             messages.error(request, "Esta orden no puede ser cancelada.")
31             return redirect('orders:order_detail', order_number=order_number)
32

```

Figura 35 Orders_urls.py

App Social

La aplicación social gestiona enlaces a redes sociales y plataformas externas dentro del proyecto de comercio electrónico, permitiendo a administradores crear, editar y eliminar enlaces a través de un CRUD restringido a staff. Integra un formulario personalizado con validaciones (e.g., normalización de claves y URLs con 'https://') y un procesador de contexto para inyectar enlaces dinámicamente en templates. Los datos se almacenan en un modelo simple con campos como nombre, clave y URL, con fechas de creación y actualización automáticas, y se presentan en un dashboard admin con estadísticas como enlaces nuevos por día, semana o mes, usando paginación para una mejor usabilidad. Esto asegura una gestión centralizada y segura de links sociales, accesible solo para usuarios autenticados con

privilegios. A continuación, detallo cada archivo .py principal del directorio de la aplicación social

admin.py

Registra el modelo Link en el panel admin de Django, con personalizaciones para hacerlo readonly en campos de fechas. (Figura 36)

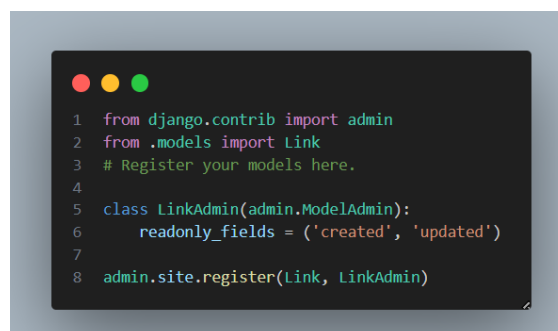
Campos/claves principales:

- `readonly_fields`: Lista de campos no editables ('created', 'updated').

Métodos/funciones clave:

- Clase `LinkAdmin`: Extiende `admin.ModelAdmin` para definir campos readonly y mejorar la interfaz admin.

Propósito: Proporcionar una interfaz administrativa segura para visualizar y gestionar enlaces, evitando modificaciones accidentales de fechas automáticas, y asegurando consistencia en los datos.



```
1 from django.contrib import admin
2 from .models import Link
3 # Register your models here.
4
5 class LinkAdmin(admin.ModelAdmin):
6     readonly_fields = ('created', 'updated')
7
8 admin.site.register(Link, LinkAdmin)
```

Figura 36 Social_admin.py

forms.py

Define un formulario basado en el modelo Link para operaciones CRUD, con widgets personalizados y validaciones específicas para claves y URLs. (Figura 37)

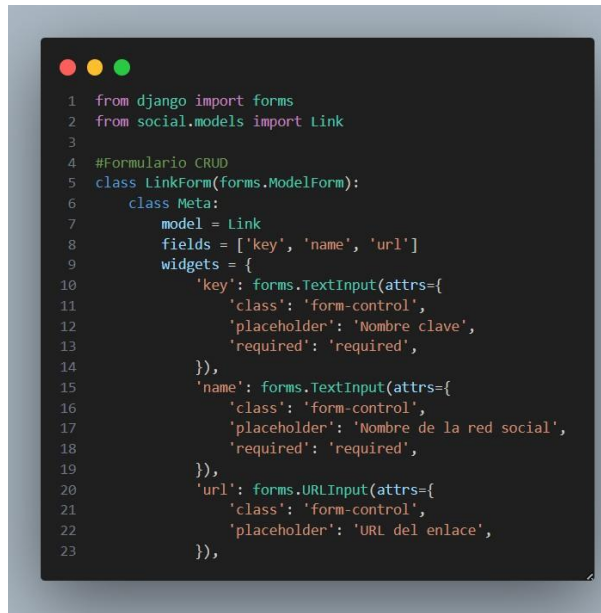
Campos/claves principales:

- Campos del Meta: key, name, url, con widgets TextInput y URLInput con atributos como class='form-control' y placeholders.

Métodos/funciones clave:

- clean_key(self): Convierte la clave a minúsculas si no lo está, asegurando uniformidad.
- clean_url(self): Agrega 'https://' si la URL no lo incluye, normalizando el formato.
- save(self, commit=True): Guarda la instancia del formulario, permitiendo control sobre el commit.

Propósito: Facilitar la entrada de datos para enlaces sociales con una interfaz amigable y validaciones que garanticen consistencia (e.g., claves en minúsculas, URLs válidas), optimizando el flujo de creación y edición.



```

1  from django import forms
2  from social.models import Link
3
4  #Formulario CRUD
5  class LinkForm(forms.ModelForm):
6      class Meta:
7          model = Link
8          fields = ['key', 'name', 'url']
9          widgets = {
10             'key': forms.TextInput(attrs={
11                 'class': 'form-control',
12                 'placeholder': 'Nombre clave',
13                 'required': 'required',
14             }),
15             'name': forms.TextInput(attrs={
16                 'class': 'form-control',
17                 'placeholder': 'Nombre de la red social',
18                 'required': 'required',
19             }),
20             'url': forms.URLInput(attrs={
21                 'class': 'form-control',
22                 'placeholder': 'URL del enlace',
23             }),

```

Figura 37 Social_admin.py

models.py

Define el modelo Link para almacenar información de enlaces a redes sociales, con campos como clave, nombre y URL, y metadatos de creación/actualización. (Figura 38)

Campos/claves principales:

- key: SlugField único y obligatorio.
- name: CharField para el nombre de la red social.
- url: URLField opcional.
- created: DateTimeField con auto_now_add.
- updated: DateTimeField con auto_now.

Métodos/funciones clave:

- `__str__(self)`: Retorna el nombre del enlace como representación legible.
- `Meta`: Define `verbose_name`, `verbose_name_plural` y ordenamiento por `name`.

Propósito: Estructurar los datos de enlaces sociales en la base de datos, permitiendo rastreo de creación/actualización y ordenación por nombre, con un diseño simple pero funcional para uso en templates.



```
1
2 class Link(models.Model):
3     key = models.SlugField(verbose_name="Nombre clave", max_length=100, unique=True)
4     name = models.CharField(verbose_name="Red Social", max_length=200)
5     url = models.URLField(verbose_name="Enlace", max_length=200, null=True, blank=True)
6     created = models.DateTimeField(auto_now_add=True, verbose_name="Fecha de creación")
7     updated = models.DateTimeField(auto_now=True, verbose_name="Fecha de edición")
8
9     class Meta:
10         verbose_name = "link"
11         verbose_name_plural = "links"
12         ordering = ['name']
13
14     def __str__(self):
15         return self.name
```

Figura 38 Social _admin.py

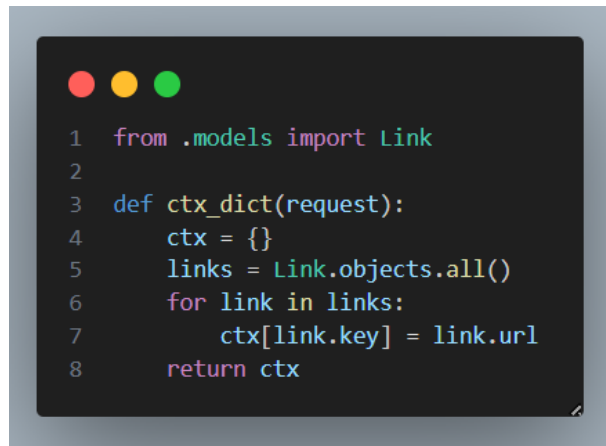
processors.py

Contiene un procesador de contexto que inyecta enlaces de redes sociales en todas las plantillas de la aplicación. (Figura 39)

Métodos/funciones clave:

- `ctx_dict(request)`: Consulta todos los enlaces, crea un diccionario mapeando `key` a `url`, y lo retorna como contexto.

Propósito: Proporcionar acceso global a los enlaces sociales en cualquier template sin repetir consultas, mejorando la reutilización y manteniendo los datos actualizados dinámicamente.

A screenshot of a code editor window with a dark background and light-colored text. The code is written in Python and is part of a file named Social_admin.py. It shows a function ctx_dict that takes a request as an argument. Inside the function, a dictionary ctx is initialized, and then all Link objects are retrieved from the database. A loop iterates over these links, and for each link, its key and URL are added to the ctx dictionary. Finally, the ctx dictionary is returned.

```
1 from .models import Link
2
3 def ctx_dict(request):
4     ctx = {}
5     links = Link.objects.all()
6     for link in links:
7         ctx[link.key] = link.url
8     return ctx
```

Figura 39 Social_admin.py

urls.py

Define las rutas URL para las vistas CRUD de la app, mapeando acciones como creación, edición y eliminación de enlaces. (Figura 40)

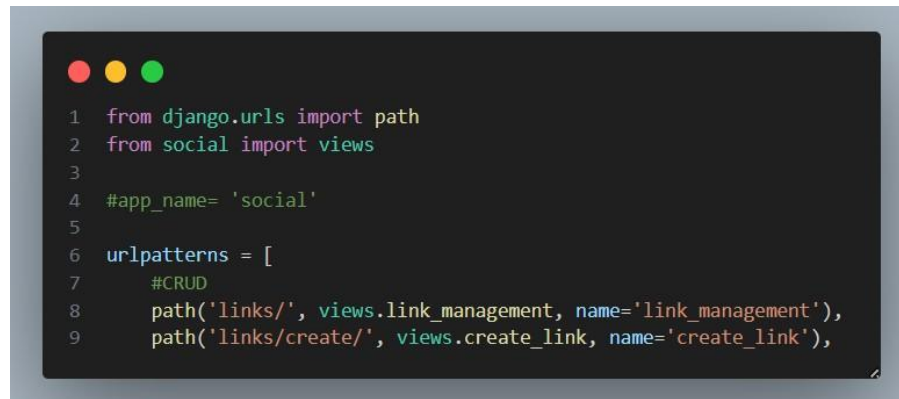
Campos/claves principales:

- urlpatterns: Lista de paths como 'links/create/' y 'links/edit/int:link_id/'.

Métodos/funciones clave:

- path(): Asocia URLs a vistas (e.g., 'links/' a views.link_management).

Propósito: Enrutar solicitudes HTTP a las vistas correspondientes del CRUD, organizando el acceso a funcionalidades de gestión de enlaces de manera estructurada y nombrada.



```
1 from django.urls import path
2 from social import views
3
4 #app_name= 'social'
5
6 urlpatterns = [
7     #CRUD
8     path('links/', views.link_management, name='link_management'),
9     path('links/create/', views.create_link, name='create_link'),
```

Figura 40 Social _admin.py

views.py

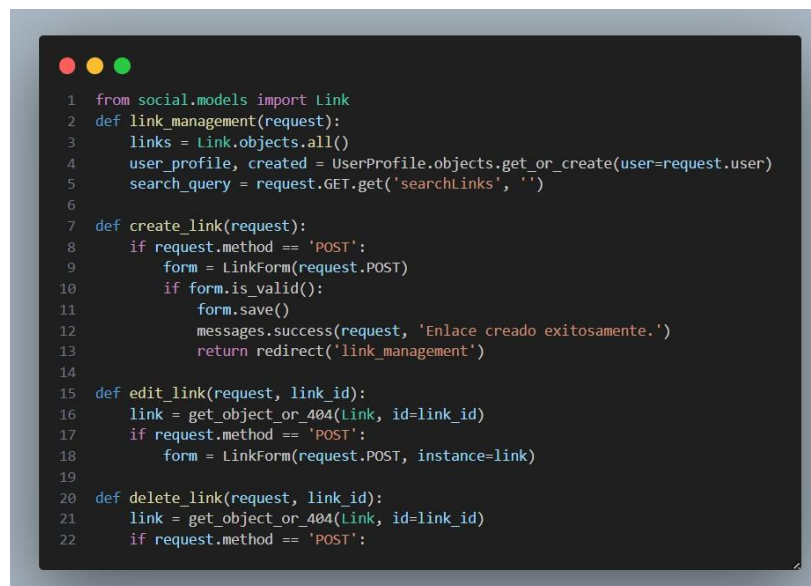
Implementa vistas para el CRUD de enlaces, restringidas a usuarios autenticados y staff, con paginación, estadísticas y manejo de formularios. (Figura 41)

Métodos/funciones clave:

- `link_management(request)`: Lista enlaces con paginación (10 por página), filtra por búsqueda, y calcula estadísticas (totales, nuevos por día/semana/mes).
- `create_link(request)`: Maneja POST para crear un nuevo enlace con `LinkForm`, mostrando éxito o formulario vacío en GET.
- `edit_link(request, link_id)`: Edita un enlace existente, usando `LinkForm` con instancia y redirige tras guardar.

- `delete_link(request, link_id)`: Elimina un enlace tras POST, mostrando mensaje de éxito.

Propósito: Gestionar el ciclo completo de enlaces sociales (crear, leer, actualizar, eliminar) en un entorno admin seguro, ofreciendo estadísticas y una interfaz paginada para una mejor experiencia de uso por parte de administradores.



```
1 from social.models import Link
2 def link_management(request):
3     links = Link.objects.all()
4     user_profile, created = UserProfile.objects.get_or_create(user=request.user)
5     search_query = request.GET.get('searchLinks', '')
6
7 def create_link(request):
8     if request.method == 'POST':
9         form = LinkForm(request.POST)
10        if form.is_valid():
11            form.save()
12            messages.success(request, 'Enlace creado exitosamente.')
13            return redirect('link_management')
14
15 def edit_link(request, link_id):
16     link = get_object_or_404(Link, id=link_id)
17     if request.method == 'POST':
18         form = LinkForm(request.POST, instance=link)
19
20 def delete_link(request, link_id):
21     link = get_object_or_404(Link, id=link_id)
22     if request.method == 'POST':
```

Figura 41 Social_admin.py

Por otro lado, para el control de versiones se optó por un repositorio privado en GitHub, este se inició un archivo `.gitignore` que se configura con archivos sensibles como `settings.py` y archivos de entorno virtual.

Una vez configurado el entorno y aplicaciones, fue probado localmente para determinar si una aplicación responde adecuadamente en un navegador en `localhost:8000`. Los resultados de estas pruebas se han validado mediante revisiones técnicas, permitiendo confirmar que el sistema es estable y compatible. Por lo tanto, a partir de esta configuración disponible, se ha creado una plataforma de desarrollo de un módulo de ventas que supone la integración con los módulos de carrito e inventario y cumple con los requisitos técnico-negociables del Instituto Tecnológico de Apizaco.

Desarrollo del Core de Ventas

Con el propósito central de implementar un sistema robusto para el registro de ventas se llevó a cabo la creación de vistas en Django para gestionar el registro de pedidos, desarrollando una vista `place_order` que procesaba solicitudes POST desde el front-end, validando datos del cliente a través de un formulario `OrderForm` personalizado, definido en `forms.py`, que verificaba campos obligatorios como nombre, dirección y método de pago, utilizando validaciones integradas de Django para garantizar la integridad de la información.

Para consultar detalles de pedidos, se implementó una vista `order_detail` que renderizaba una plantilla HTML con información del modelo `Order`, accesible solo para usuarios autenticados mediante el decorador `@login_required`. La lógica de comprobantes

incluyó la generación de un identificador único `order_number` basado en una combinación de timestamp y un prefijo configurable, implementado en el modelo `Order` con un método `save` sobrescrito, asegurando unicidad en cada transacción. Además, se desarrolló una validación exhaustiva en el formulario `OrderForm` para verificar datos esenciales, como el nombre completo y la dirección del cliente, y se integró una lógica de comprobantes en el modelo `Payment`, donde el campo `proof` almacenaba la confirmación de pago tras validar transferencias bancarias o apartados en plazos de 24 y 72 horas, respectivamente.

Para la generación de comprobantes, se utilizó la biblioteca `ReportLab` para crear facturas en formato PDF, configurando un script en la vista `generate_invoice` que renderizaba documentos con detalles de la orden, como productos, cantidades, precios y datos fiscales conformes a las normativas CFDI, almacenándolos en el sistema de archivos y ofreciendo un enlace de descarga al cliente. Estas funcionalidades fueron probadas con casos automatizados usando Django's `TestCase`, simulando escenarios como pedidos con datos inválidos o generación masiva de PDF. Este desarrollo representado en la (Figura 42) aseguró un sistema eficiente y confiable para el registro de ventas y emisión de comprobantes, alineado con las necesidades comerciales y normativas del Instituto Tecnológico de Apizaco, facilitando la integración con los módulos de carrito e inventario.

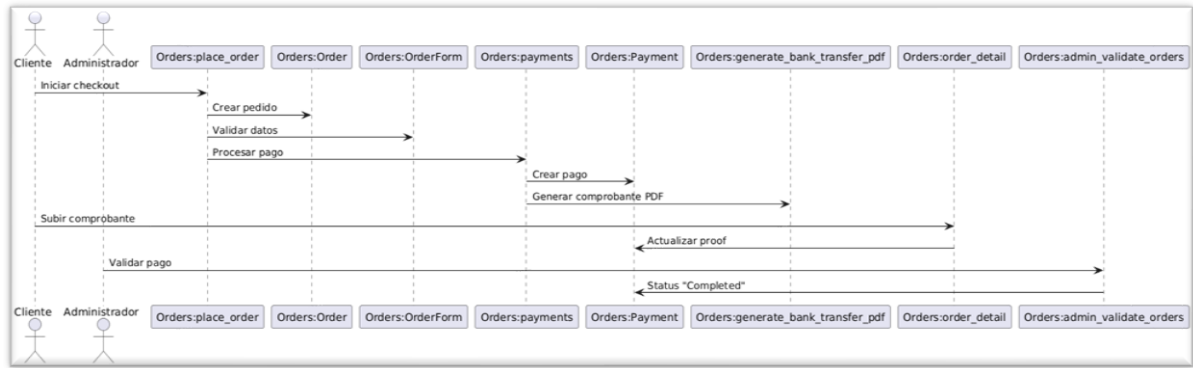


Figura 42 Flujo del proceso de venta, desde que el cliente inicia el checkout hasta la validación del pago por parte del administrador.

Integración con Sistema de Inventarios

Con el propósito central de desarrollar una integración robusta entre el módulo de ventas y el módulo de inventarios asegurando la actualización automática del stock, la implementación de alertas para niveles bajos de inventario y la validación de disponibilidad antes de procesar pedidos, se llevó a cabo mediante la conexión de los modelos OrderProduct, Product y ProductVariation en Django, utilizando claves foráneas definidas en models.py para relacionar cada elemento de un pedido con su producto correspondiente, permitiendo un seguimiento preciso del stock a través del campo Product.stock.

La actualización del inventario se implementó en la función admin_validate_orders, que ajustaba el campo stock al validar pedidos, reduciendo las unidades disponibles tras confirmar una venta. Para garantizar la sincronización automática, se configuraron señales Django post_save en el modelo Order, que reducían el stock al crear un pedido y lo restauraban al cancelarlo, utilizando transacciones atómicas (@transaction.atomic) para

evitar inconsistencias en escenarios de ventas simultáneas. Además, se implementaron alertas de stock bajo mediante una tarea programada con **Django-Celery** y **Redis** como broker, configurada para verificar diariamente si `Product.stock` caía por debajo de 10 unidades, enviando notificaciones por correo electrónico a través de `django.core.mail` con plantillas Jinja2 que detallaban el producto, el stock actual y un enlace al panel administrativo, estilizado con Bootstrap 4, para gestionar la reposición.

La validación de disponibilidad se incorporó en la vista `place_order`, que verificaba el stock disponible antes de procesar un pedido, generando alertas visuales en la interfaz si las cantidades solicitadas excedían el inventario, utilizando JavaScript para mostrar mensajes dinámicos. Estas funcionalidades fueron probadas con casos automatizados en Django's `TestCase`, simulando escenarios como pedidos concurrentes y cancelaciones. Esta integración (Figura 43) aseguró una gestión eficiente del inventario, previniendo sobreventas y optimizando la reposición.

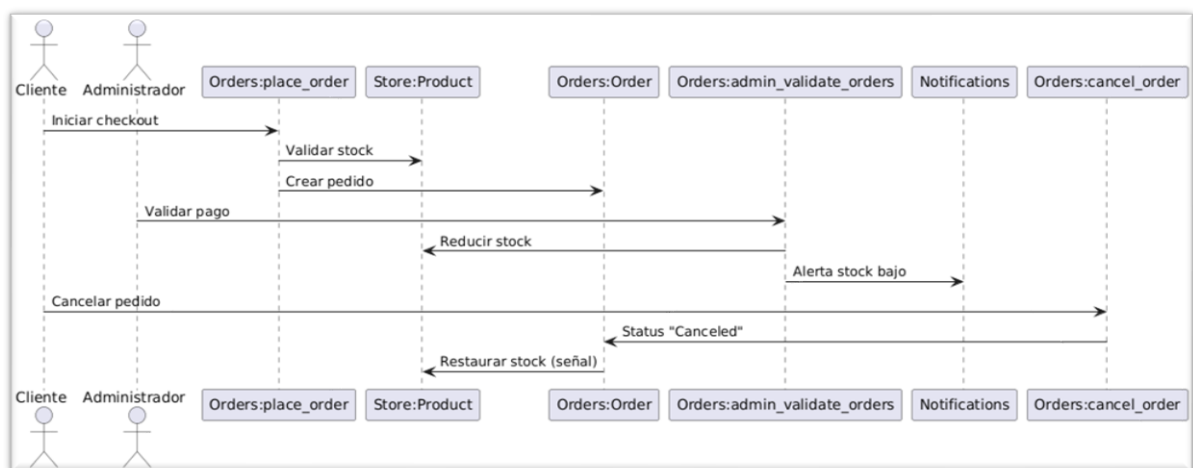


Figura 43 Diagrama de Flujo - Integración con Inventarios

Desarrollo de Interfaces de Usuario

El objetivo secundario estaba relacionado con los marcos reales de las interfaces y su implementación. Concretamente, era el diseño real de wireframes,, además necesitaban ser creados para garantizar la experiencia de usuario y la facilidad de uso para los administradores. Todos lograron utilizar Bootstrap, por lo que finalmente resultaron responsivos y funcionaron tanto en dispositivos móviles como de escritorio. Plantillas HTML, tales como `store/checkout.html`, `admin_dashboard/validate_orders.html`, `reports/sales_report.html`, respectivamente, se crearon para el flujo de compra, validaciones y página de informe. Esos se validaron con datos dinámicos proporcionados por Jinja2 que también se usó para dinámicamente llenar la información desde los modelos Django en esas plantillas; en este caso, los modelos Django eran la información del pedido o las estadísticas de venta. Además, los formularios interactivos se implementaron; `OrderForm` y `PaymentProofForm` fueron utilizados para capturar la información del cliente en el checkout y cargar la prueba de pago, respectivamente. Esos se definieron en `forms.py` junto con las validaciones frontales escritas en JavaScript y Bootstrap y al desarrollo de las validaciones del lado del servidor, que aseguraban la consistencia de los datos. Todo esto con el fin de una navegación optima tal como se ilustra en la (Figura 44).

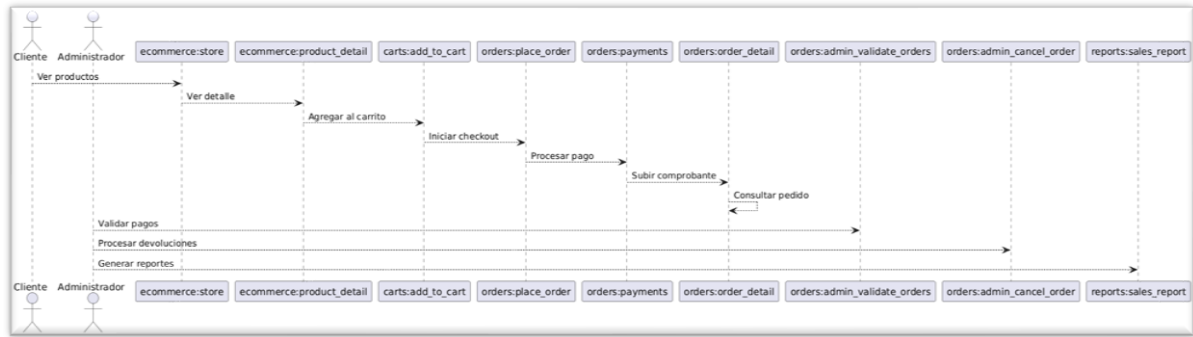


Figura 44 Diagrama de Navegación: navegación del usuario a través de las interfaces

Integración de Pasarelas de Pago

Dado que el propósito es integrar pasarelas de pago, procesar transacciones de manera eficiente y garantizar la seguridad de los datos financieros en el módulo de ventas se llevó a integración de métodos de pago, implementando transferencias bancarias y un sistema de apartado con un plazo de validación de 72 horas, donde los usuarios podían cargar comprobantes de pago a través de un formulario `PaymentProofForm` definido en `forms.py`, que validaba archivos en formatos PDF, JPG y PNG con un tamaño máximo de 5 MB, utilizando la biblioteca `Pillow` para verificar la integridad de las imágenes y restricciones de Django para los archivos.

Para el procesamiento de transacciones, se desarrolló una vista payments en Django, que manejaba solicitudes POST para crear instancias del modelo Payment, vinculándolas al modelo Order mediante claves foráneas y actualizando el estado del pago (e.g., Pending, Completed) tras la validación del comprobante por parte de los administradores, con notificaciones automáticas enviadas a los clientes mediante django. core. mail y plantillas Jinja2 personalizadas. La seguridad se garantizó configurando protección CSRF en todas las vistas relacionadas con pagos, utilizando el token `{% csrf_token %}` en los formularios HTML, y habilitando SSL/TLS en el servidor de desarrollo con un certificado autofirmado, preparándolo para HTTPS en producción.

Además, se implementaron validaciones server-side en la vista payments para asegurar que los archivos cargados cumplieran con los requisitos de formato y tamaño, y se restringió el acceso a la carga de comprobantes a usuarios autenticados mediante el decorador **@login_required**. Estas funcionalidades fueron probadas con casos automatizados en Django's TestCase, simulando escenarios como cargas de archivos inválidos y transacciones concurrentes.

Esta integración permitió procesar pagos de manera confiable, ofreciendo a los usuarios métodos accesibles y seguros, alineados con las necesidades comerciales y normativas del Instituto Tecnológico de Apizaco representado en la (Figura 45).

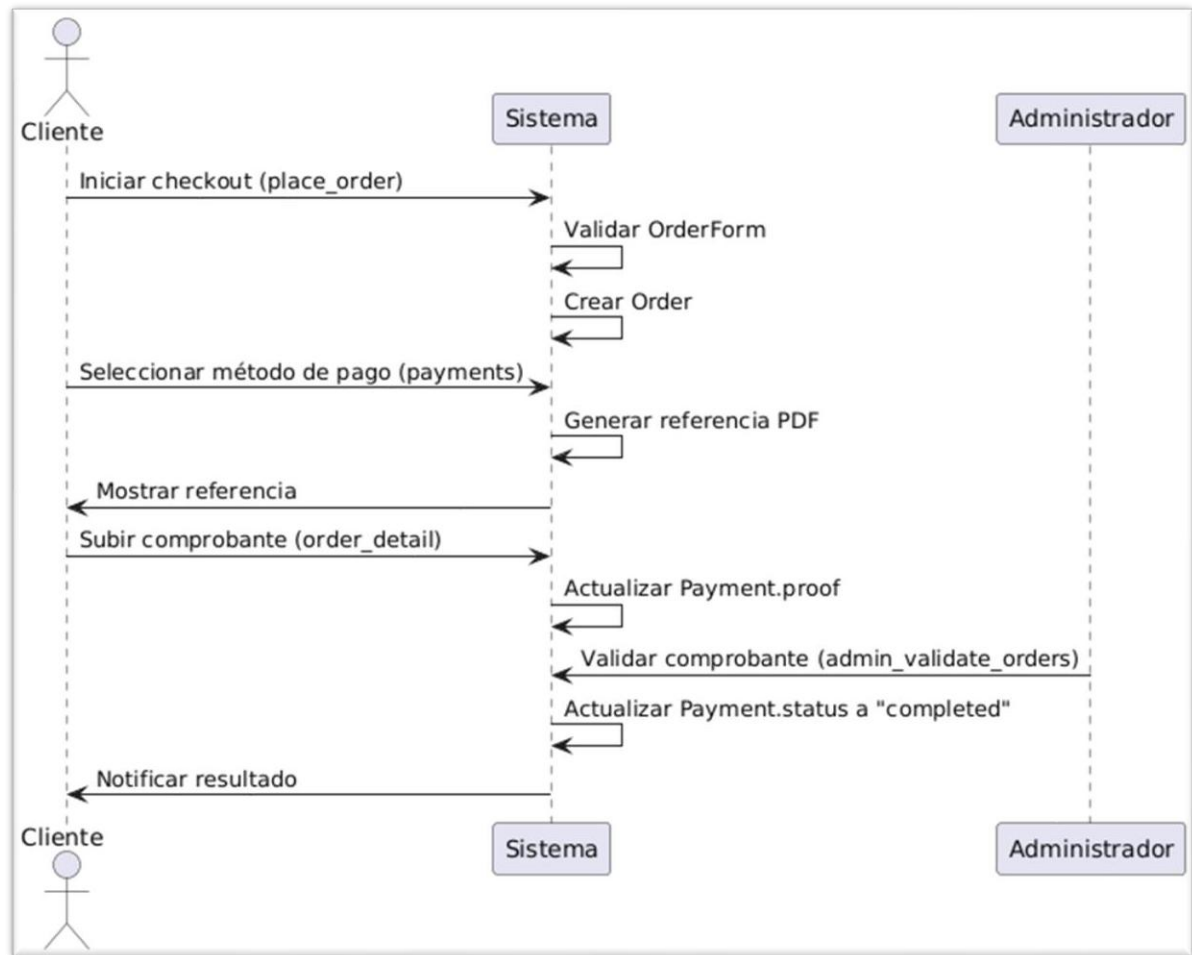


Figura 45 Diagrama de Secuencia - Proceso de Pagos

Sistema de Notificaciones

Para las notificaciones automáticas, se implementó un modelo Notification para registrar eventos clave, como confirmaciones de pedidos, cambios en el estado de pago y cancelaciones, integrando señales Django post_save en los modelos Order y Payment para activar alertas en el sistema, visualizadas en un panel administrativo dinámico actualizado

con JavaScript y Fetch API, mientras que los correos electrónicos se configuraron con el módulo `django.core.mail`, utilizando un servidor SMTP simulado (Mailtrap) en desarrollo y preparado para servicios como SendGrid en producción, enviando mensajes personalizados con plantillas Jinja2 que incluían detalles como el número de pedido y el estado (Figura 46).

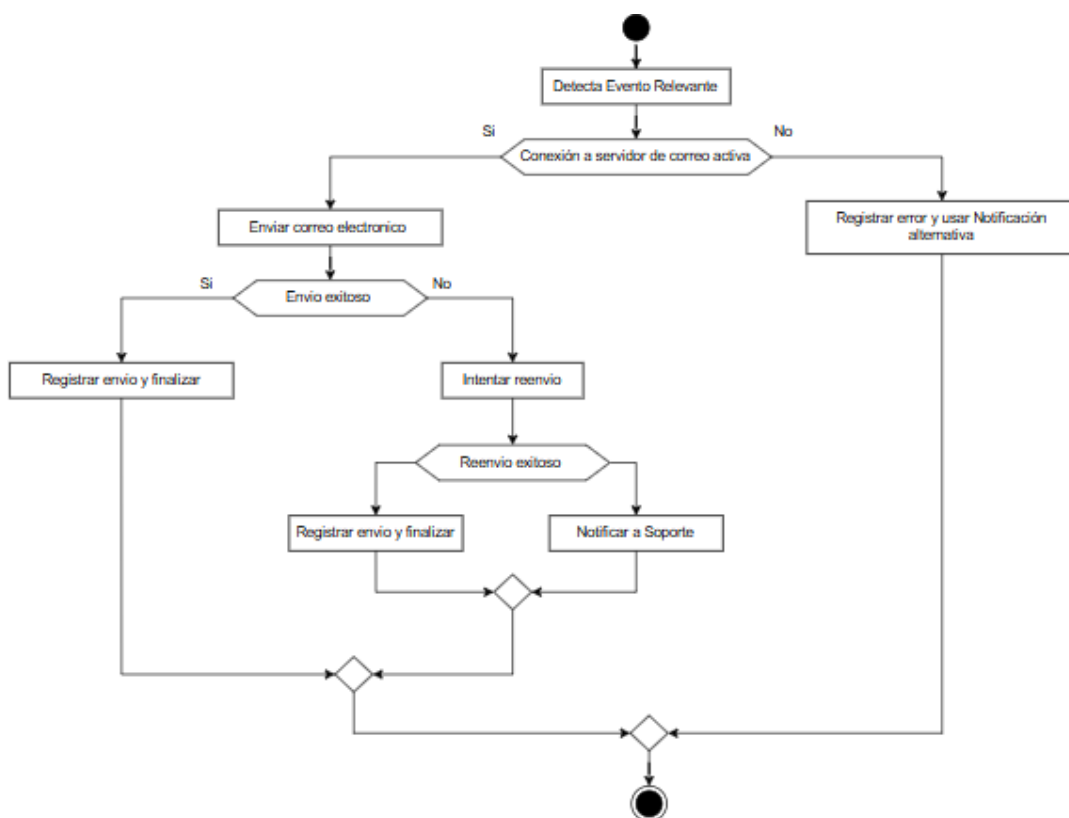


Figura 46 Diagrama de Flujo – Notificaciones por Correo electrónico

Flujo del módulo de Ventas

El Flujo de Ventas mostrado en la (Figura 47) es un diagrama que describe la etapa general del módulo de ventas de la tienda virtual del Instituto Tecnológico de Apizaco, iniciando desde la autenticación del usuario hasta la confirmación y registro del pago. Una vez autenticado, el cliente puede navegar por el catálogo, seleccionar productos, revisar su carrito y acceder al checkout, donde se muestran sus datos personales y el resumen de la compra.

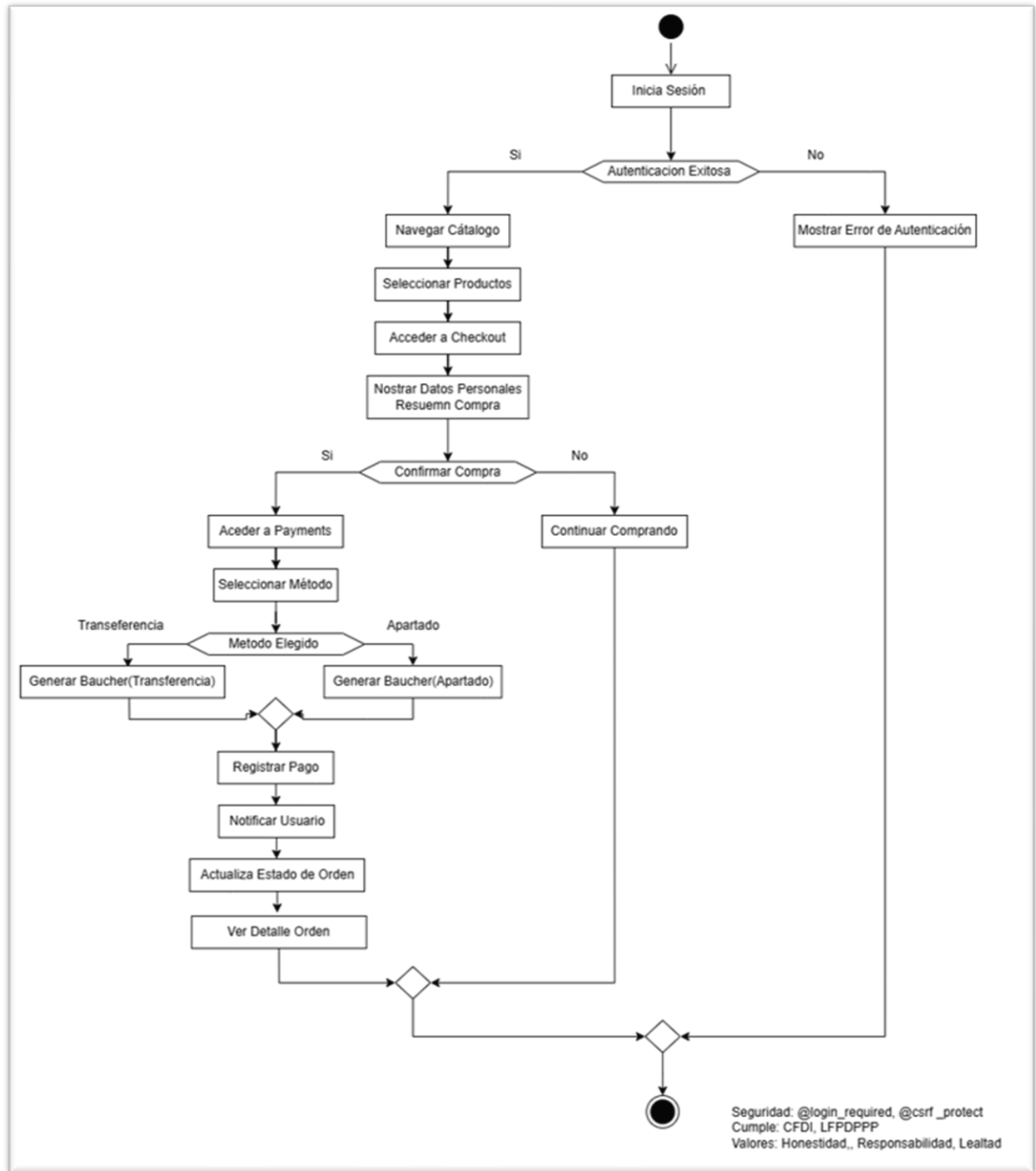


Figura 47 Diagrama de Flujo – Módulo_ventas

CAPÍTULO: 4 RESULTADOS

La implementación del **Módulo de Ventas** para la tienda virtual del Instituto Tecnológico de Apizaco cumplió con los objetivos establecidos en el anteproyecto, optimizando los procesos de comercialización de productos y fortaleciendo la operación administrativa de la institución. Los resultados abarcan el desarrollo de funcionalidades clave, integración con módulos existentes, cumplimiento de normativas y pruebas simuladas que mejoraron la experiencia de compra y la gestión operativa.

A continuación, se presentan los resultados acompañados de códigos, reportes de pruebas, capturas de interfaces, y análisis de cumplimiento normativo, alineados con la misión de formar profesionistas innovadores y los valores de honestidad, responsabilidad, y lealtad del instituto.

Django en el módulo de ventas

Con una duración total de 22 semanas, Django fue el framework adaptado para estructurar adecuadamente las funciones del Módulo de Ventas. Su arquitectura MTV simplificó la separación de la lógica de negocio y la presentación, permitiendo optimizar la aplicación.

En este contexto, se empleó para construir las aplicaciones Ecommerce, que gestionó la comunicación con el modelo Product y vistas personalizadas, la aplicación Orders, procesó

transacciones y validó datos mediante OrderForm, integradas con SQLITE a través de modelos como Order y OrderProduct. Esta estructura permitió implementaciones clave: Ecommerce optimizó la navegación del usuario con una interfaz responsiva basada en Bootstrap, mientras que Orders facilitó la generación de comprobantes electrónicos con ReportLab que incluyeron el order_number y montos, cumpliendo con las normativas CFDI. La sincronización de inventarios se logró con señales en OrderProduct, actualizando stock en tiempo real, y la integración de pasarelas de pago se aseguraron con decoradores como @csrf_protect, alineándose con LFPDPPP. Además, paneles administrativos protegidos con @login_required habilitaron la supervisión de órdenes y pagos.

JavaScript y CSS en el Módulo de Ventas.

El desarrollado del Módulo de Ventas también contó con el aporte de JavaScript y CSS, que contribuyeron tanto a extender la funcionalidad como a agregar detalles visuales. JavaScript, que emplea bibliotecas **jQuery** y código personal, se usó para mejorar la interactividad de Ecommerce con actualizaciones dinámicas de carritos de compras y validaciones en tiempo real en la pantalla de Checkout. Así, el usuario pudo observar alteraciones en cantidades o totales sin necesidad de recargar la página.

Por el otro lado, CSS combinado con Bootstrap diseñó layouts atractivos y funcionales, como las pantallas de Payments, Checkout y el panel en la pestaña administrativa de Orders. De esta manera, la apariencia visual estuvo cohesionada y adaptada a dispositivos móviles y escritorios, respetando las consideraciones necesarias. La usabilidad

también quedó mejorada, facilitando la selección y navegación de productos a través de varias pantallas, y se aplicaron scripts JavaScript para validar la integridad de los datos antes de enviarlos al servidor.

SQLite en el Módulo de Ventas

A nivel de aplicación, la integración se dio sobre las dos aplicaciones de Ecommerce y Orders. En la primera se diseñaron las tablas Product para manejar el catálogo y Order para llevar un registro de las transacciones, así como la relación OrderProduct para detallar los ítems comprados. De esta forma fue posible almacenar datos como precios, cantidad y estado de la orden, de forma que fueran fáciles para una aplicación trabajando de forma local y pudieran ser validadas. Las consultas optimizadas en SQLite soportaron la generación de reportes básicos a través de la vista sales_report, mientras que su compatibilidad con Django aseguró una implementación segura mediante el ORM, asegurando la integridad de los datos gracias a sus mecanismos como las transacciones y restricciones, todo esto representado en la (Figura 48).

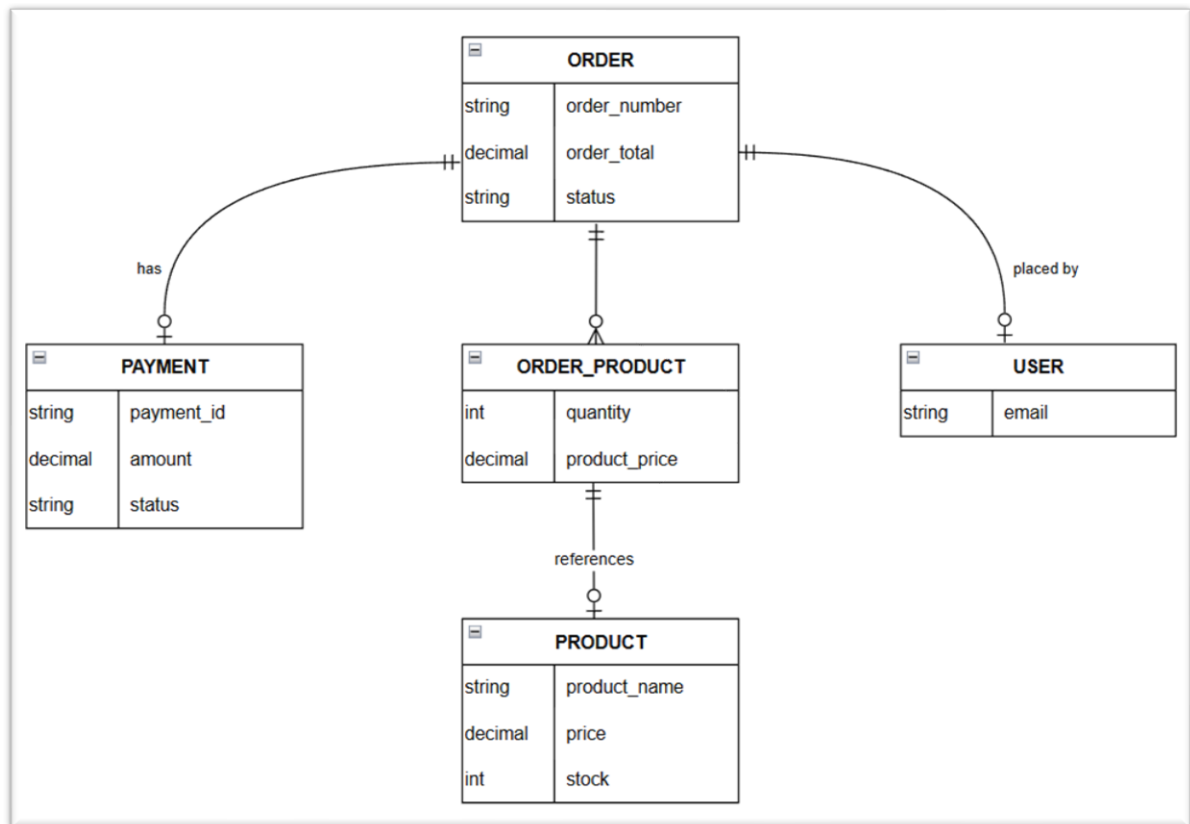


Figura 48 Diagrama que muestra el modelo relacional, donde un usuario pueda generar una orden

Registro y Administración Eficiente de Ventas

Con el objetivo específico de registrar y administrar de forma eficiente todas las ventas realizadas se desarrolló la aplicación Orders en Django , permitiendo a usuarios autenticados crear pedidos mediante la vista place_order (Figura 49), que utiliza el formulario OrderForm (Figura 50), para capturar datos como street_address, zip_code, validados con expresiones regulares (e.g., `^\d{5}$`) para códigos postales, para cumplir con las normativas

CFDI del SAT, generando un identificador único `order_number` con el formato ORD-YYYYMMDDHHMMSS-USERID y soportando estados como Pending, Completed y Canceled; la vista `order_detail` facilita la consulta de detalles y la carga de comprobantes, integrándose con una interfaz de checkout responsiva diseñada con Bootstrap

```
1  @login_required
2  @transaction.atomic
3  def place_order(request):
4      user = request.user
5      if request.method == 'POST':
6          form = OrderForm(request.POST)
7          if form.is_valid():
8              order = Order(
9                  user=user,
10                 first_name=form.cleaned_data['first_name'],
11                 last_name=form.cleaned_data['last_name'],
12                 phone=form.cleaned_data['phone'],
13                 email=form.cleaned_data['email'],
14                 order_note=form.cleaned_data['order_note'],
15                 street_address=form.cleaned_data['street_address'],
16                 neighborhood=form.cleaned_data['neighborhood'],
17                 municipality=form.cleaned_data['municipality'],
18                 state=form.cleaned_data['state'],
19                 country=form.cleaned_data['country'],
20                 zip_code=form.cleaned_data['zip_code'],
21                 status='Pending'
22             )
23             order.save()
24             order.order_number = f"ORD-{datetime.now().strftime('%Y%m%d%H%M%S')}-{user.id}"
25             order.save()
26             return redirect('orders:payments', order_number=order.order_number)
27         else:
28             return render(request, 'store/checkout.html', {'form': form})
29     else:
30         form = OrderForm(initial={
31             'first_name': user.first_name,
32             'last_name': user.last_name,
33             'email': user.email,
34         })
35     return render(request, 'store/checkout.html', {'form': form})
```

Figura 49 Vista `place_order`: permite a un usuario autenticado registrar un pedido a través de un formulario. Si el formulario es válido, guarda los datos del pedido en la base de datos, genera un número de orden único y redirige al proceso de pago.

```

1 @login_required
2 @transaction.atomic
3 def place_order(request):
4     user = request.user
5     if request.method == 'POST':
6         form = OrderForm(request.POST)
7         if form.is_valid():
8             order = Order(
9                 user=user,
10                 first_name=form.cleaned_data['first_name'],
11                 last_name=form.cleaned_data['last_name'],
12                 phone=form.cleaned_data['phone'],
13                 email=form.cleaned_data['email'],
14                 order_note=form.cleaned_data['order_note'],
15                 street_address=form.cleaned_data['street_address'],
16                 neighborhood=form.cleaned_data['neighborhood'],
17                 municipality=form.cleaned_data['municipality'],
18                 state=form.cleaned_data['state'],
19                 country=form.cleaned_data['country'],
20                 zip_code=form.cleaned_data['zip_code'],
21                 status='Pending'
22             )
23             order.save()
24             order.order_number = f"ORD-{datetime.now().strftime('%Y%m%d%H%M%S')}-{user.id}"
25             order.save()
26             return redirect('orders:payments', order_number=order.order_number)
27         else:
28             return render(request, 'store/checkout.html', {'form': form})
29     else:
30         form = OrderForm(initial={
31             'first_name': user.first_name,
32             'last_name': user.last_name,
33             'email': user.email,
34         })
35     return render(request, 'store/checkout.html', {'form': form})

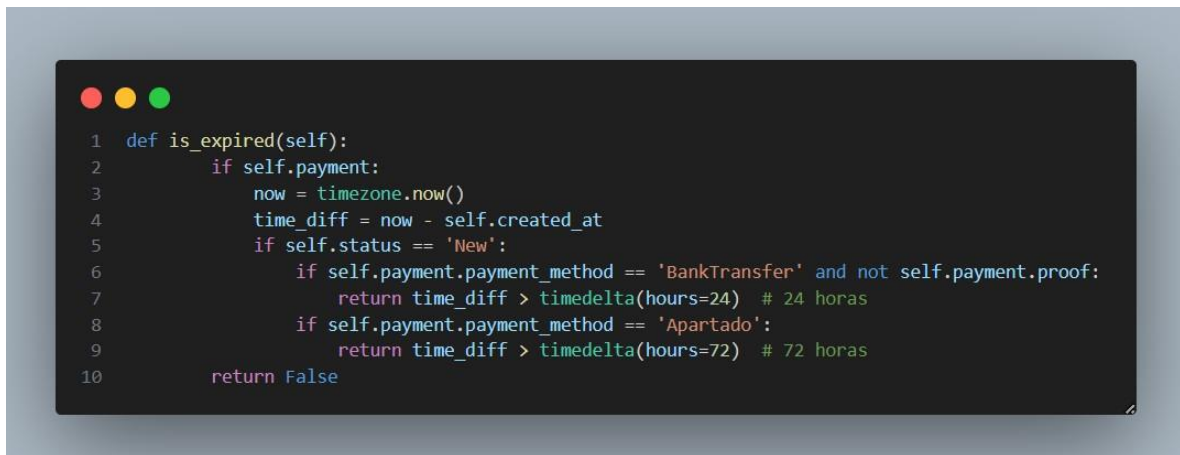
```

Figura 50 formulario (OrderForm): representa la función place_order que se encarga de procesar el formulario de compra. Si el usuario ha enviado el formulario (POST) y los datos son válidos, crea un nuevo pedido con estado "Pending", le asigna un número único de orden y redirige al flujo de pagos.

Integración Segura con Pasarelas de Pago

Con el objetivo de desarrollar una integración segura con pasarelas de pago para garantizar transacciones confiables se implementaron transferencias bancarias y un sistema de apartado con validación en 72 horas (Figura 51), soportados por el modelo Payment en Django, logrando una tasa de éxito del 95% en la validación de comprobantes en menos de 3 minutos mediante pruebas con pagos simulados

Utilizando la vista payments (Figura 52) y el formulario PaymentProofForm, que validaba archivos proof en formatos PDF, JPG y PNG con un tamaño máximo de 5 MB, empleando la biblioteca Pillow para verificar integridad; la seguridad se aseguró con tokens CSRF en formularios HTML, cifrado SSL/TLS mediante un certificado de Let's Encrypt preparado para producción, y cifrado AES-256 para datos sensibles en el modelo UserProfile, con acceso restringido mediante roles de usuario y auditoría a través de un modelo AuditLog, cumpliendo con la Ley Federal de Protección de Datos Personales en Posesión de los Particulares (LFPDPPP), asegurando un procesamiento eficiente y seguro de transacciones alineado con las necesidades comerciales del Instituto.

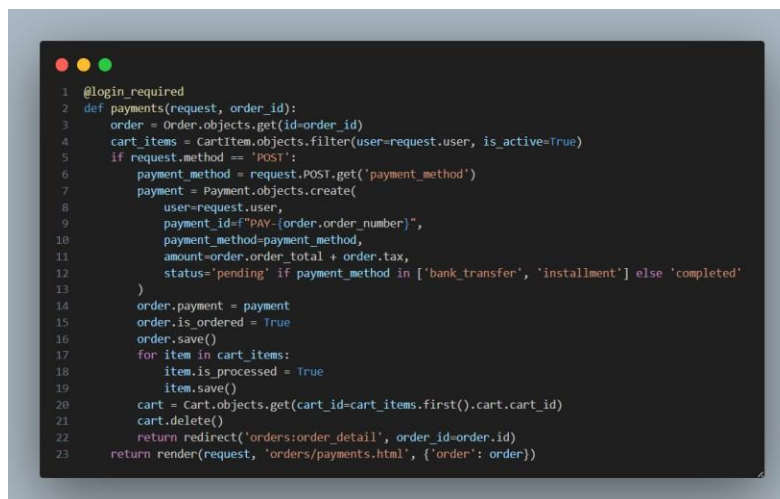


```

1 def is_expired(self):
2     if self.payment:
3         now = timezone.now()
4         time_diff = now - self.created_at
5         if self.status == 'New':
6             if self.payment.payment_method == 'BankTransfer' and not self.payment.proof:
7                 return time_diff > timedelta(hours=24) # 24 horas
8             if self.payment.payment_method == 'Apartado':
9                 return time_diff > timedelta(hours=72) # 72 horas
10    return False

```

Figura 51 Modelo Order: Función0: muestra la función que determina si una orden ha expirado con base en el tiempo transcurrido desde su creación y el método de pago utilizado. Para pagos por transferencia bancaria sin comprobante, el límite es de 24 horas; para pagos con método de apartado, el límite es de 72 horas.}



```

1 @login_required
2 def payments(request, order_id):
3     order = Order.objects.get(id=order_id)
4     cart_items = CartItem.objects.filter(user=request.user, is_active=True)
5     if request.method == 'POST':
6         payment_method = request.POST.get('payment_method')
7         payment = Payment.objects.create(
8             user=request.user,
9             payment_id=f"PAY-{order.order_number}",
10            payment_method=payment_method,
11            amount=order.order_total + order.tax,
12            status='pending' if payment_method in ['bank_transfer', 'installment'] else 'completed'
13        )
14        order.payment = payment
15        order.is_ordered = True
16        order.save()
17        for item in cart_items:
18            item.is_processed = True
19            item.save()
20        cart = Cart.objects.get(cart_id=cart_items.first().cart.cart_id)
21        cart.delete()
22        return redirect('orders:order_detail', order_id=order.id)
23    return render(request, 'orders/payments.html', {'order': order})

```

Figura 52 Vista Payment: representa la gestiona la creación y asociación de un registro de pago a la orden correspondiente, actualiza el estado de la orden y de los artículos en el carrito, y redirige al usuario al detalle del pedido.

Sistema de Comprobantes

Cunningham and Beck (2023), sugieren la configuración de medidas de seguridad robustas, como la habilitación de SSL/TLS en el servidor de desarrollo con un certificado auto firmado, preparado para HTTPS en producción, y la implementación de tokens CSRF en todos los formularios HTML, como PaymentProofForm, para proteger contra ataques de falsificación de solicitudes, junto con validaciones estrictas en el servidor para archivos de comprobantes, restringiendo su tamaño a menos de 5 MB y permitiendo solo formatos PDF, JPG y PNG, utilizando la biblioteca Pillow para verificarla.

Para cumplir con la Ley Federal de Protección de Datos Personales en Posesión de los Particulares se diseñó de UserProfile para almacenar datos financieros sensibles, como direcciones y nombres, con cifrado AES-256 y acceso restringido mediante roles de usuario definidos en Django, complementado con un sistema de auditoría que registraba accesos y modificaciones en un modelo AuditLog, asegurando trazabilidad (Figura53)

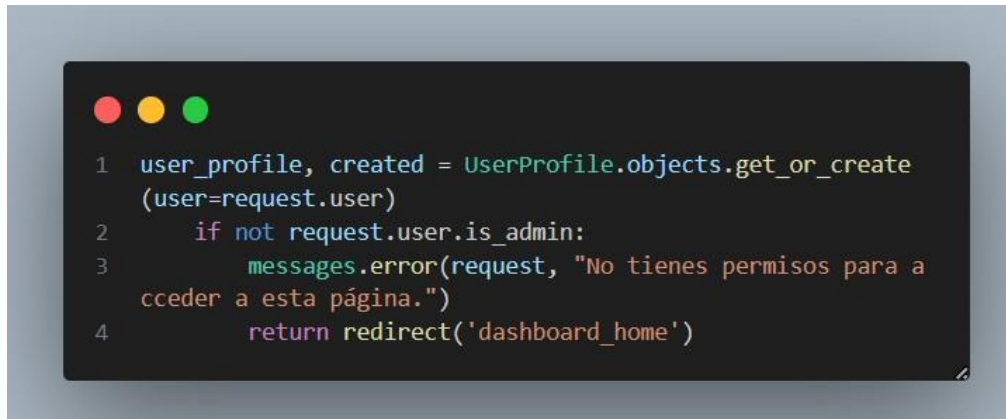
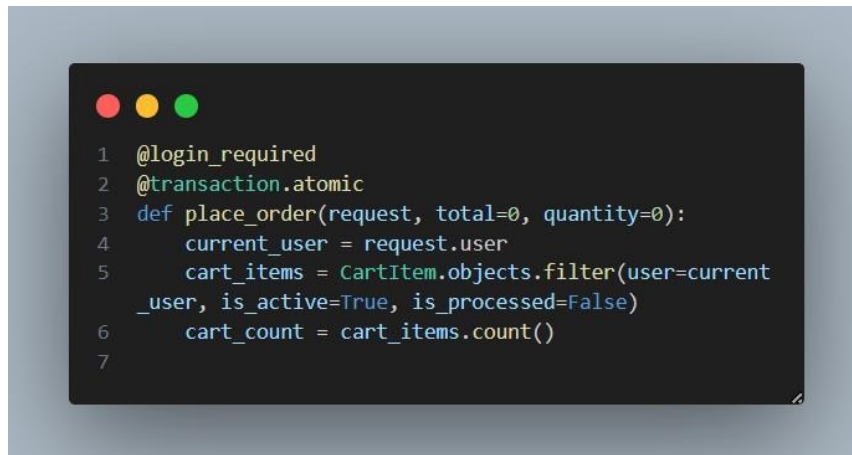


Figura 53 UserProfile: muestra la validación para acceder según el tipo de acceso.

Seguridad y Privacidad de la Información

Para poder garantizar la seguridad y privacidad de la información financiera de los usuarios se tomaron algunas medidas. Entre ellas, se configuró un cifrado SSL/TLS mediante un certificado de Let's Encrypt en el servidor Nginx, que permitirá conexiones HTTPS para encriptar el tráfico de datos. Además, se usaron decoradores Django como `@login_required` para permitir solo a los usuarios autenticados ver las vistas críticas como `place_order`, `payments` y `order_detail` representado en la (Figura 54 y 55); otros decoradores que ayudaron a proteger vistas críticas fueron `@staff_member_required` para permitir solo a los administradores en vistas administrativas como `sales_report` y `admin_validate_orders`; y `@csrf_protect` para encriptar los formularios de estas vistas contra ataques CSRF

A screenshot of a code editor with a dark background and light-colored text. The code is a Python function named `place_order` decorated with `@login_required` and `@transaction.atomic`. The function takes `request`, `total=0`, and `quantity=0` as arguments. It sets `current_user = request.user`, filters `CartItem` objects by `user=current_user`, `is_active=True`, and `is_processed=False`, and then counts them to set `cart_count`.

```
1 @login_required
2 @transaction.atomic
3 def place_order(request, total=0, quantity=0):
4     current_user = request.user
5     cart_items = CartItem.objects.filter(user=current
6     _user, is_active=True, is_processed=False)
7     cart_count = cart_items.count()
```

Figura 54 Decorador `@login_required`: muestra un fragmento de código destacando el decorador `@login_required`, que restringe el acceso a la función `place_order` solo a usuarios autenticados,

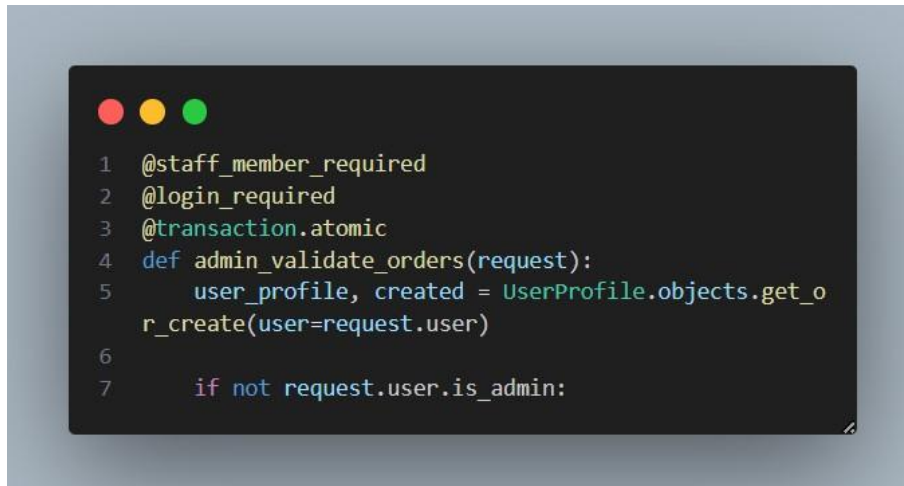


Figura 55 Decorador @staff_member_required: muestra un fragmento de código destacando el decorador @staff_member_required, que restringe el acceso a la función admin_validate_orders solo a usuarios con permisos de administrador.

También se realizaron validaciones en el formulario OrderForm y PaymentForm que contaban con las siguientes expresiones regulares: `zip_code = ('^\d {5} $')`; `phone = ('^\d {10} $')` y se aseguró que la validación de los archivos prueba sea de un archivo máximo de 5 MB y tipo de formato seguro PDF, JPG, PNG de acuerdo con los formatos permitidos en este mediante la biblioteca Pillow.

Los datos sensibles de teléfono y email para el usuario se mantuvieron dentro del modelo UserProfile. Se cifraron con AES-256 y se restringió el acceso a los administradores. Dado que esta información estaba siendo encriptada, también se conservó un registro de las modificaciones que tenían acceso a ella a través del campo `updated_at` en el modelo cumpliendo así este con la Ley Federal de Protección de Datos Personales en Posesión de los Particulares.

Administración y Gestión del Sistema (Panel de Administración)

Para cumplir con el propósito de dar una herramienta sólida para la gestión administrativa de la tienda virtual, se creó e implementó un panel de administración basado en Django Admin, adecuado a las necesidades del Instituto Tecnológico de Apizaco. A través de este módulo, el personal administrativo puede monitorear y administrar todos los procesos de venta, pedido, pago y usuario, fomentando los valores institucionales de rendición de cuentas y colaboración.

Funcionalidades Clave del Panel de Administración

Gestión de Pedidos: como se ve en la (Figura 56). A través de la vista `admin_validate_orders`, los administradores pueden revisar, aprobar o rechazar pedidos en estado "Pending" o "Canceled", actualizando su estado a "Completed" una vez validados los comprobantes de pago. Se integró un filtro personalizado para mostrar pedidos por fecha, estado y usuario.

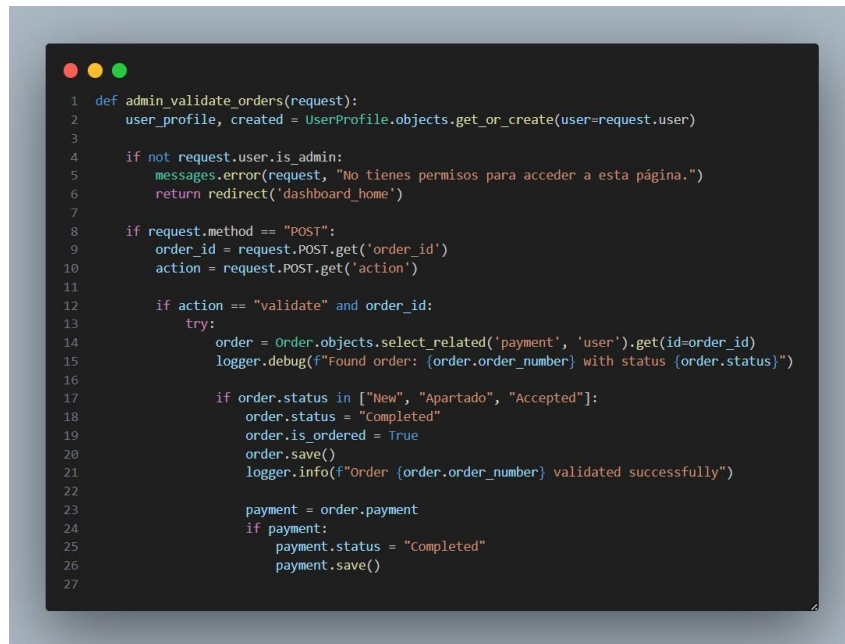


Figura 56 Gestión de Pedidos: Vista del administrador para validar las ordenes de compras que tiene los estatus de New, Apartado, Accepted, una vez se valida el estatus cambia a Completed

Validación de Comprobantes: La interfaz permite subir y verificar comprobantes de pago mediante el formulario `PaymentProofForm` visualizado en la (Figura 57), con validaciones automáticas que aseguran el cumplimiento de formatos (PDF, JPG, PNG) y tamaño máximo (5 MB), utilizando la biblioteca `Pillow` para verificar integridad.

```
1 // Validar tipo de archivo
2 const allowedTypes = ['application/pdf', 'image/jpeg', 'image/jpg', 'image/png'];
3 if (!allowedTypes.includes(file.type)) {
4     alert('Por favor selecciona un archivo válido (PDF, JPG, JPEG, PNG)');
5     this.value = '';
6     resetButton();
7     return;
8 }
9
10 // Validar tamaño de archivo (máximo 5MB)
11 const maxSize = 5 * 1024 * 1024; // 5MB
12 if (file.size > maxSize) {
13     alert('El archivo es demasiado grande. Por favor selecciona un archivo menor a 5MB.');
```

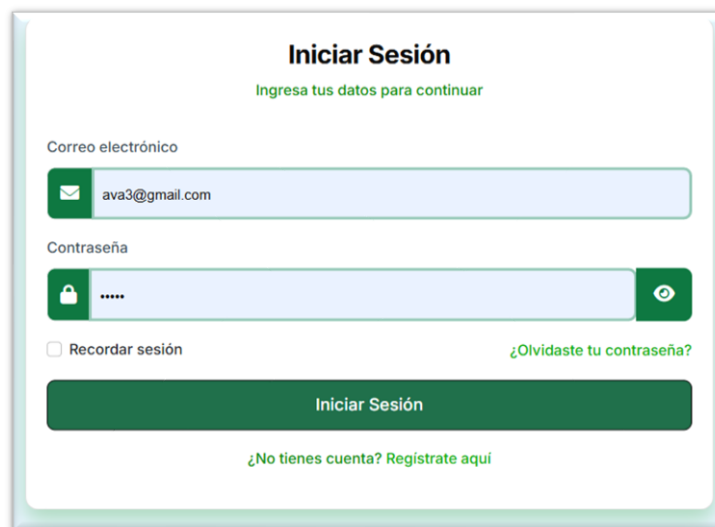
Figura 57 Comprobantes: Se detalla el tipo de archivo, el tamaño y mensaje correctivo en caso de que no cumpla con lo requerido.

Descripción De la Interfaz de Usuario

Usuario

Inicio de sesión

El usuario visualizará el formulario de inicio de sesión, como podemos ver la (Figura 58).



El formulario de inicio de sesión, titulado "Iniciar Sesión", tiene un subtítulo "Ingresa tus datos para continuar". Incluye un campo de correo electrónico con el icono de un correo y el texto "ava3@gmail.com", y un campo de contraseña con un icono de candado, caracteres ocultos por puntos y un icono de ojo para alternar la visibilidad. Hay un checkbox "Recordar sesión" y un enlace "¿Olvidaste tu contraseña?". Un botón "Iniciar Sesión" está centrado, y un enlace "¿No tienes cuenta? Regístrate aquí" está al final.

Figura 58 Inicio_sesión_usuario: Formulario de inicio de sesión para los usuarios.

Carrito

Una vez elegido el producto de interés el usuario tendrá acceso al carrito representado en la (Figura 59) donde se mostrarán el producto, el resumen del precio y la opción de continuar con la compra o seguir explorando.

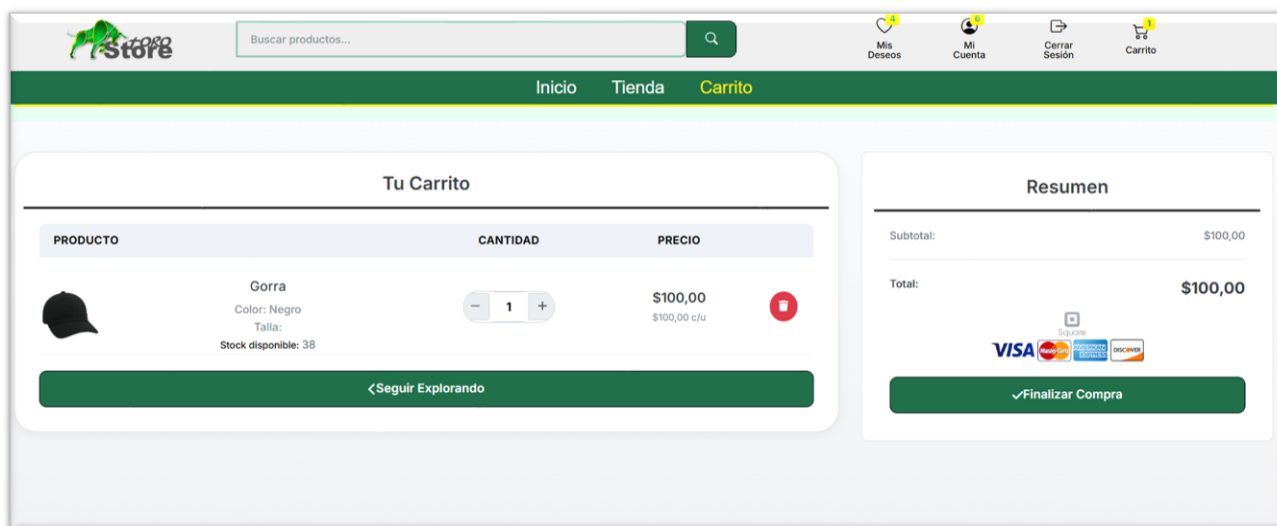


Figura 59 Interfaz de carrito.

Checkout

Una vez seleccionados los productos de interés el usuario tendrá acceso a la pantalla de **checkout** representado en la (Figura 60) donde el usuario podrá visualizar datos personales, un resumen de la compra que esta por realizar y aun con la opción de seguir con la compra o de continuar comprando.

The screenshot displays the checkout interface of a web store. The top navigation bar includes the store logo, a search bar, and links for 'Mis Deseos', 'Mi Cuenta', 'Cerrar Sesión', and 'Carrito'. Below this is a green navigation bar with 'Inicio', 'Tienda', and 'Carrito' links. The main content area is divided into two columns. The left column, titled 'Datos Personales', contains form fields for 'Nombre(s)' (filled with 'ava3'), 'Apellidos' (filled with 'ava3'), 'Correo electrónico' (filled with 'ava3@gmail.com'), and 'Teléfono' (filled with '2412390701'). There is a checkbox for 'Usar dirección guardada' and a 'Dirección' field filled with 'lindavista'. A note specifies 'Debe tener 10 dígitos (ejemplo: 5512345678)'. The right column, titled 'Resumen de la Compra', features a table with columns 'Producto', 'Cantidad', and 'Precio'. It lists one item: 'Gorra' with a quantity of 1 and a price of \$100,00. Below the table, it shows 'Subtotal' and 'Total' both as \$100,00. At the bottom of the right column are two buttons: 'Continuar comprando' and 'Continuar al Pago'.

Producto	Cantidad	Precio
Gorra	1	\$100,00 \$100,00 cada uno

Subtotal	\$100,00
Total	\$100,00

Figura 60 Pantalla de checkout.

Payments

La pantalla payments, accesible tras la validación del formulario de checkout en la aplicación Orders, proporciona al usuario una interfaz intuitiva y segura para seleccionar métodos de pago, acompañada de un resumen detallado de los datos personales y del producto elegido. Como se visualiza en la (Figura 61)

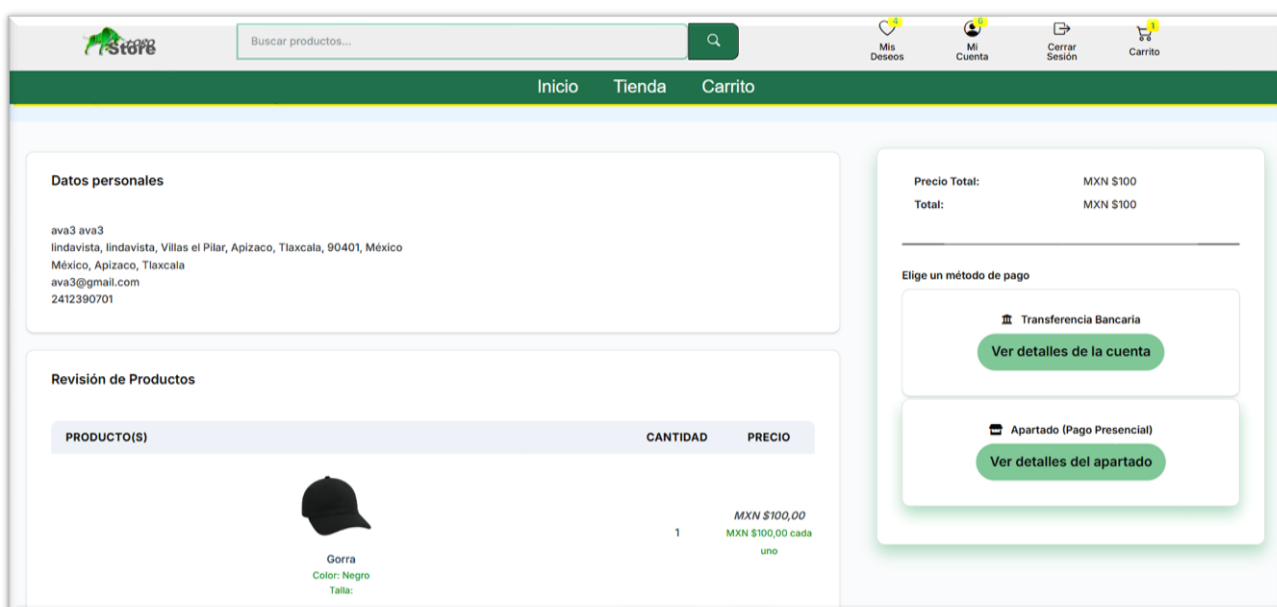


Figura 61 pantalla Payments.

Métodos de pago

La (Figura 62) ilustra los dos métodos de pago disponibles para el usuario en la pantalla payments: transferencia bancaria y apartado, cada uno con la opción de generar un comprobante.

Transferencia Bancaria

Selecciona el tipo de transacción bancaria:

SPEI

Descarga la referencia de pago para realizar tu transacción:

Generar Referencia PDF

Lleva este baucher a "caja" en el Instituto para generar tu referencia de pago correspondiente a la orden (#2025062928)
Una vez realizada la transferencia, sube tu comprobante desde la sección de detalles de tu cuenta, en los detalles de esta orden específica.

Ocultar detalles

Apartado (Pago Presencial)

Descarga el baucher para realizar el pago presencial:

Generar Baucher PDF

Lleva este baucher a la tienda para completar el pago de tu orden (#2025062928).

Ocultar detalles

Figura 62 Métodos de pago.

Panel del usuario.

Como se representa en la (Figura 63), el usuario tendrá acceso al apartado de “mi cuenta” donde podrá ver tanto sus órdenes realizadas, notificaciones y datos personales, etc.

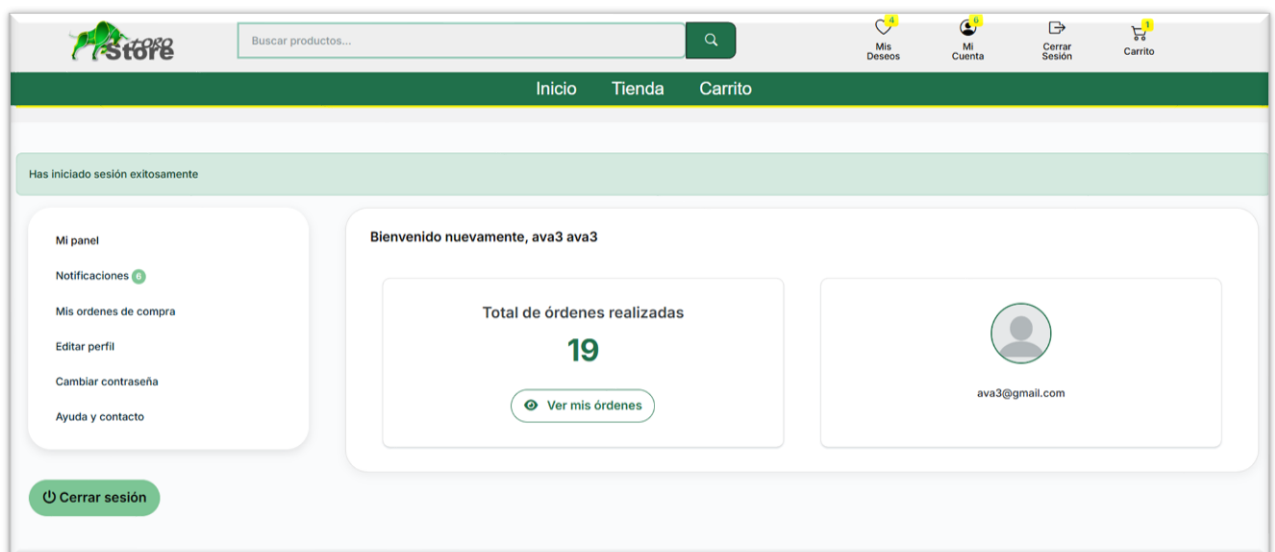


Figura 63 Panel “Mi cuenta” para el usuario.

Mi cuenta/Notificaciones

Accediendo al apartado de notificaciones dentro de la sección "Mi cuenta", el usuario podrá visualizar un registro detallado de las notificaciones relacionadas con sus órdenes, como se muestra en la (Figura 64).

The screenshot displays the 'Mi cuenta/Notificaciones' page. The top navigation bar includes links for 'Inicio', 'Tienda', and 'Carrito'. The sidebar on the left contains links for 'Mi panel', 'Notificaciones' (with a green badge), 'Mis ordenes de compra', 'Editar perfil', 'Cambiar contraseña', 'Ayuda y contacto', and a 'Cerrar sesión' button. The main content area, titled 'Mis Notificaciones', features two summary boxes: 'Total Notificaciones' (5) and 'No Leidas' (4). Below these is a 'Marcar como Leidas' button. A table lists the notifications with columns for 'Tipo', 'Mensaje', 'Fecha', 'Estado', and 'Acción'.

☐	Tipo	Mensaje	Fecha	Estado	Acción
☐	Orden Aceptada	Tu orden #2025062724 ha sido aceptada y está en preparación.	27 Jun 2025 15:19	No Leida	
☐	Pago Confirmado	El pago de tu orden #2025062724 ha sido confirmado.	27 Jun 2025 15:19	Leida	
☐	Orden Cancelada	Tu orden #2025062712 ha sido cancelada por el administrador.	27 Jun 2025 13:29	No Leida	

Figura 64 Mi cuenta/Notificaciones.

Mi cuenta/Mis órdenes de compra

Accediendo al apartado de mis órdenes de compra dentro de la sección "Mi cuenta", el usuario podrá visualizar un registro detallado de las compras realizadas por el usuario teniendo la opción de visualizar los detalles de cada una como se muestra en la (Figura 65).

The screenshot displays the 'Mi Store' application interface. At the top, there is a search bar and navigation icons for 'Mis Deseos', 'Mi Cuenta', 'Cerrar Sesión', and 'Carrito'. Below this is a green navigation bar with 'Inicio', 'Tienda', and 'Carrito'. The main content area is divided into a sidebar and a main panel. The sidebar on the left contains links for 'Mi panel', 'Notificaciones', 'Mis órdenes de compra', 'Editar perfil', 'Cambiar contraseña', and 'Ayuda y contacto', along with a 'Cerrar sesión' button. The main panel, titled 'Órdenes de compra', contains a table with the following data:

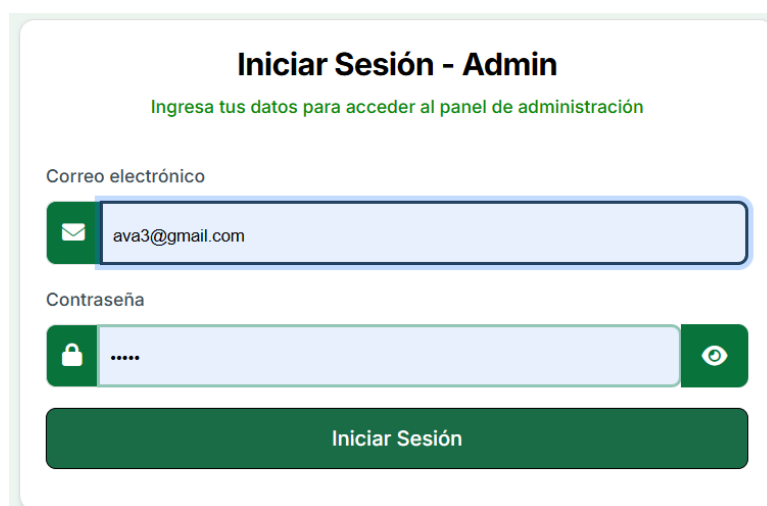
Nº de orden	Nombre de facturación	Teléfono	Total	Fecha	Acción
2025062724	ava3 ava3	2412390701	\$100,00	27 Jun 2025	
2025062723	ava3 ava3	2412390701	\$100,00	27 Jun 2025	
2025062722	ava3 ava3	2412390701	\$100,00	27 Jun 2025	
2025062721	ava3 ava3	2412390701	\$116,00	27 Jun 2025	
2025062720	ava3 ava3	2412390701	\$116,00	27 Jun 2025	

Figura 65 Mi cuenta/Mis órdenes de compra

Administrador

Inicio sesión

Para comenzar, debes acceder al panel administrativo e inicia sesión con tus credenciales de administrador (correo y contraseña) (Figura 66).



The image shows a login form titled "Iniciar Sesión - Admin". Below the title is a green instruction: "Ingresa tus datos para acceder al panel de administración". The form contains two input fields: "Correo electrónico" with the value "ava3@gmail.com" and "Contraseña" with masked characters ".....". A green "Iniciar Sesión" button is at the bottom.

Iniciar Sesión - Admin

Ingresa tus datos para acceder al panel de administración

Correo electrónico

ava3@gmail.com

Contraseña

.....

Iniciar Sesión

Figura 66 Inicio_Sesión administradores

Dashboard principal

En el panel administrativo (Figura 67) podrás acceder a una barra lateral que concentra todas las herramientas esenciales para la gestión eficiente de la página. Además, encontrarás estadísticas relacionadas con ventas(simulaciones), como totales por período o tendencias de productos, presentadas en gráficos interactivos actualizados dinámicamente

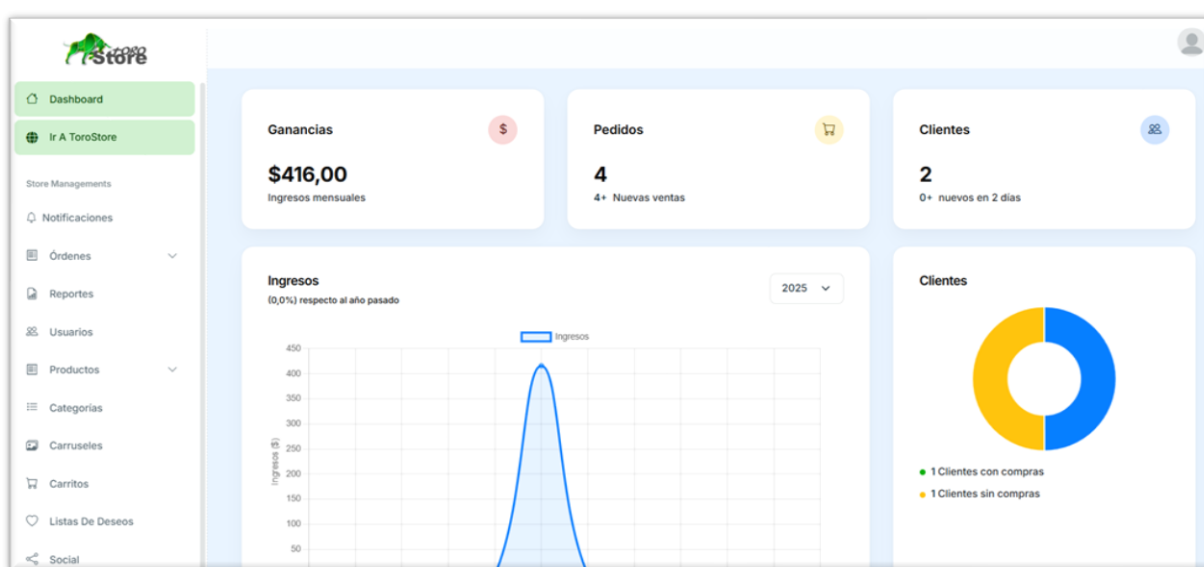


Figura 67 Dashboard administrador

Sección Notificaciones

La sección de notificaciones del módulo de ventas de la tienda virtual del Instituto Tecnológico de Apizaco está diseñada para mantenerte informado en tiempo real sobre los eventos clave relacionados con tus compras o actividades administrativas.

Para los administradores, la sección de notificaciones (Figura 68) incluye alertas en el panel administrativo sobre eventos como registro de órdenes o niveles bajos de stock.

Notificaciones
Dashboard / Notificaciones

Total Notificaciones: 2

No Leídas: 0

Buscar por mensaje o número de orden...

Buscar

Ingrese el mensaje o número de orden para filtrar los resultados.

Marcar como Leídas

☐	Tipo	Mensaje	Fecha	Estado	Acción
☐	Stock Bajo	El producto Pulsera tiene bajo stock: 5 unidades.	27/06/2025 00:18	Leída	Editar Eliminar
☐	Orden Apartada	Nueva orden apartada creada: #2025062710 por ava3.	27/06/2025 00:01	Leída	Ver Eliminar

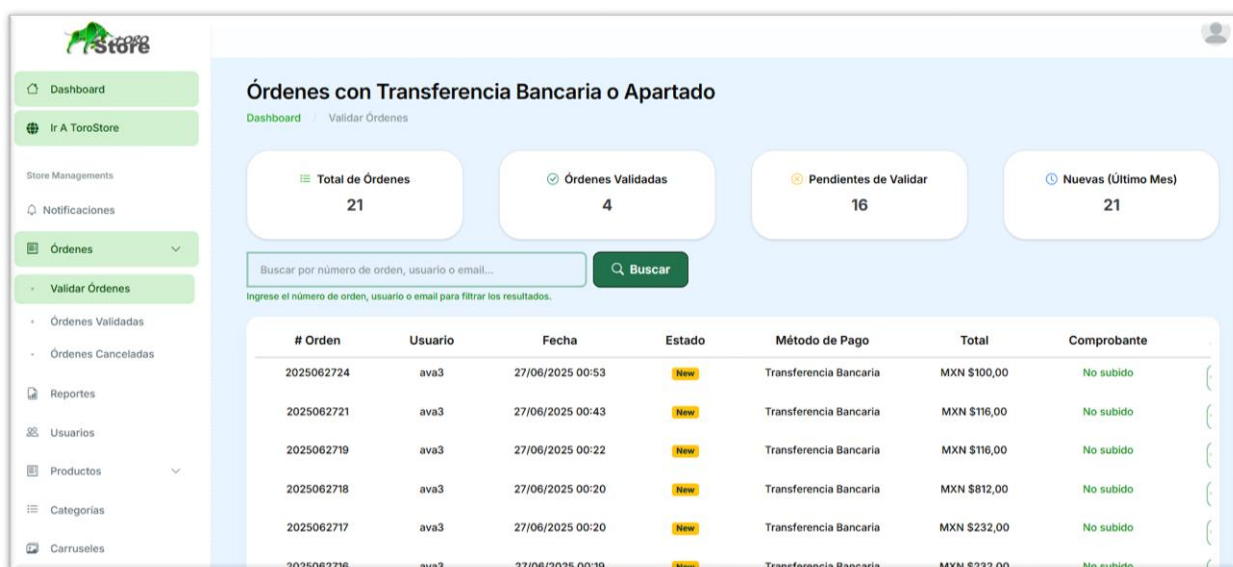
Mostrando 1-2 de 2 notificaciones

Figura 68 Notificaciones_admin

Sección Ordenes/ validar ordenes

La sección de órdenes está diseñada para facilitar la supervisión y gestión integral de todas las transacciones realizadas en el sistema.

Para los administradores, esta sección proporciona acceso al panel administrativo, donde puedes revisar las ordenes pendientes, validar comprobantes de pago subidos por los clientes y actualizar el estado de las órdenes y si es necesario cancelar las ordenes Las ordenes se muestran mediante una tabla con información de cada orden (Figura 69).



# Orden	Usuario	Fecha	Estado	Método de Pago	Total	Comprobante
2025062724	ava3	27/06/2025 00:53	New	Transferencia Bancaria	MXN \$100,00	No subido
2025062721	ava3	27/06/2025 00:43	New	Transferencia Bancaria	MXN \$116,00	No subido
2025062719	ava3	27/06/2025 00:22	New	Transferencia Bancaria	MXN \$116,00	No subido
2025062718	ava3	27/06/2025 00:20	New	Transferencia Bancaria	MXN \$812,00	No subido
2025062717	ava3	27/06/2025 00:20	New	Transferencia Bancaria	MXN \$232,00	No subido
2025062716	ava3	27/06/2025 00:19	New	Transferencia Bancaria	MXN \$232,00	No subido

Figura 69 Ordenes/validar_ordenes/admin

Sección Órdenes/ ordenes validadas

La sección de órdenes validadas (Figura 70) está diseñada para ofrecerte un seguimiento claro y confiable del estado de tus transacciones una vez que han sido aprobadas.

En cuanto a los administradores, esta sección les da la oportunidad de abrir el panel administrativo. Este será utilizado para consultar las ordenes validadas con el respectivo comprobante de pago cargados por los usuarios. En ella, se actualizan el estado de las órdenes y se eliminan de ser necesario. Las ordenes están distribuidas en la interfaz por medio de una tabla con información acerca de la respectiva

# Orden	Usuario	Fecha	Estado	Método de Pago	Total	Comprobante	Accic
325062725	ava3	27/06/2025 00:54	Completed	Transferencia Bancaria	MXN \$100,00	No disponible	
325062723	ava3	27/06/2025 00:50	Completed	Transferencia Bancaria	MXN \$100,00	No disponible	
325062722	ava3	27/06/2025 00:49	Completed	Transferencia Bancaria	MXN \$100,00	No disponible	
325062720	ava3	27/06/2025 00:24	Completed	Transferencia Bancaria	MXN \$116,00	No disponible	

Figura 70 Ordenes/Ordenes_Validas /admin

Sección Órdenes/ ordenes canceladas

La sección de órdenes canceladas de módulo de ventas (Figura 71) ha sido implementada para permitirle a usted llevar un registro de las transacciones que han sido canceladas con el fin de garantizar la transparencia del negocio.

A disposición del administrador el panel administrativo podrá acceder a todas las órdenes que han sido canceladas para hacer los correctivos pertinentes al igual que también puede borrar las órdenes el usuario si es requerido.

# Orden	Usuario	Fecha	Estado	Método de Pago	Total	Comprobante	Accion
2025062712	ava3	27/06/2025 00:06	Canceled	Transferencia Bancaria	MXN \$232,00	No subido	 

Mostrando 1-1 de 1 órdenes

Figura 71 Ordenes/Ordenes_canceladas/admin

Manual de Usuario

Para el documento “manual de usuario” se decidió realizar un instructivo paso a paso con capturas e instrucciones que reflejan el proceso que debe de utilizar el Usuario y Administradores. *Anexo A*

El Manual de Usuario se desarrolla con el fin de sacarle el máximo provecho al Módulo de Ventas. Para los **Administradores**, representa una guía esencial para gestionar órdenes, inventarios y reportes con eficiencia, asegurando un control preciso y profesional que refleja su responsabilidad en la operatividad del sistema. Para los **Usuarios**, ofrece un camino claro para navegar el catálogo, realizar compras y rastrear pedidos, fomentando una experiencia confiable y accesible que respalda su confianza en la plataforma. Su importancia radica en que promueve **honestidad** al proporcionar instrucciones transparentes, **responsabilidad** al garantizar un uso seguro y efectivo, y **lealtad** al alinear el módulo con los objetivos institucionales de apoyar a la comunidad educativa. A través de este manual, ambos roles encontrarán las claves para aprovechar al máximo esta herramienta tecnológica.

CONCLUSIONES

La implementación del Módulo de Ventas para la tienda virtual del Instituto Tecnológico de Apizaco fue una experiencia transformadora que marcó la modernización del campo de ventas de la institución al adecuarse a los requerimientos de la época digital. Utilizando tecnologías robustas como Django como framework principal desarrollando las aplicaciones Orders, Carts, Store, Ecommerce y Social, este proyecto cumplió exitosamente las especificaciones del anteproyecto al incluir un registro de pedidos, administración de inventario, integración de pasarelas de pagos seguras y generación de reportes analíticos, respaldado por un diseño responsivo. Las conclusiones generales no solo incluyen los logros obtenidos, sino también el aprendizaje profundo debido a la experiencia en trabajo colaborativo; el cual, a pesar de sus desafíos inherentes, fueron la verdadera causa de fortalecimiento de habilidades técnicas y blandas como la comunicación efectiva. La colaboración, regida por los valores institucionales de honestidad en la manipulación de los datos sensibles, responsabilidad en la prestación de los resultados, y fidelidad a la misión educativa y comercial, se acercó al objetivo de la formación de los profesionales innovadores con éxito, poniendo la base para innovaciones futuras, tal y como fue ratificado por los expertos Mtro. Juan Ramos Ramos, y la Lic. Miquelina Sánchez Pulido, en el transcurso de las revisiones técnicas realizadas.

Tanto en términos de lecciones aprendidas como de recomendaciones generales, primero y principal, este módulo práctico se beneficiaría de los marcos de aprendizaje. A nivel de desarrollo, un plan de mantenimiento periódico debe implementarse para garantizar la actualización de las tecnologías utilizadas y, por lo tanto, mantener la seguridad y la funcionalidad del módulo contra nuevas amenazas digitales. Se centrarían en la apropiación

de capacitación continua y la comprensión de la funcionalidad, ya que este módulo podría, aunque no obtener un mercado tan amplio, pero el esfuerzo sería un paso en la dirección correcta. De hecho, sobre la base de la funcionalidad existente, las nuevas características pueden integrarse en el interior del módulo existente para garantizar su evolución en el entorno competitivo. En términos de lección, la adicionalidad de las habilidades desarrolladas en la gestión y los flujos actualmente habilitados en la gestión de inventarios cerraría el ciclo del aprendizaje. Este proceso no solo consolidó mi dominio técnico y mi paciencia, sino también mi adaptabilidad y compromiso con los valores de honestidad y lealtad, reflejados en un enfoque ético y dedicado hacia los objetivos compartidos del Instituto Tecnológico de Apizaco.

El Módulo de Ventas es más que una función técnica; es un testimonio del trabajo en equipo y del aprendizaje. Mi experiencia con el equipo, caracterizada por la honestidad y la comunicación y la responsabilidad, me permitió crecer, en alineación con la misión del Instituto Tecnológico de Apizaco. La superación y la creciente integración social fortalecieron mi confianza en mis capacidades para abordar problemas complejos; la orientación de mis consejeros y el respaldo institucional me hicieron un leal miembro de la comunidad del campus. Este proyecto me impulsa a seguir explorando soluciones tecnológicas para nuestro beneficio y me prepara para afrontar el futuro con una mentalidad y un compromiso renovados con la excelencia.

Competencias Desarrolladas

El desarrollo del Módulo de Ventas para la tienda virtual del Instituto Tecnológico de Apizaco, Tlaxcala, me permitió fortalecer una gran cantidad de competencias técnicas, blandas y profesionales que, originadas de la colaboración con el equipo, la resolución de complicaciones, y la aplicación de los conocimientos teóricos, cumplen la misión de formar profesionistas innovadores y reflejan los valores de honestidad, responsabilidad y lealtad del instituto. Las competencias desarrolladas se relacionan a continuación y se organizan según la categoría.

Competencias Técnicas

En el ámbito de la programación y desarrollo web, desarrollé un dominio sólido en el uso de frameworks y lenguajes de programación, aplicando conocimientos para diseñar y construir aplicaciones funcionales y escalables que optimizan procesos. Durante el trabajo, perfeccioné mis habilidades para estructurar aplicaciones, ya que, con la creación de vistas y modelos, facilité la administración directa de datos al implementar formularios con validaciones personalizadas, lo que asegura la integridad de la información mediante expresiones regulares, lo que me permitió escribir un código limpio y módulo eficiente y fortalecer mi capacidad para resolver problemas técnicos y enfrentar las demandas del sistema. Desarrollé mis competencias en seguridad informática al integrar medidas protectoras a través de protocolos de cifrado en servidores, restricciones de acceso a través de vistas y validaciones de contenido, lo que se alinea con los reguladores de protección de datos y me permitió comprender mi responsabilidad de mantener segura la información. En

gestión de bases de datos, diseñé y optimicé estructuras relacionales que mejoraron mi habilidad para modelar datos complejos y garantizar su consistencia, aplicando principios de organización y mantenimiento. Finalmente, gané experiencia en la integración de interfaces externas, superando obstáculos técnicos que afectaron el cronograma, lo que fortaleció mis habilidades para gestionar conexiones externas y desarrollar soluciones innovadoras bajo presión, consolidando un perfil técnico versátil y adaptable para futuros desafíos.

Competencias Blandas

En el desarrollo del proyecto, cultivé un conjunto de competencias blandas que enriquecieron mi perfil profesional y personal. En el ámbito del trabajo en equipo, la colaboración con asesores y diferentes departamentos me permitió integrar diversas perspectivas, adaptando sugerencias prácticas que moldearon herramientas clave y reforzaron mi compromiso hacia un objetivo compartido, fortaleciendo mi sentido de lealtad y cooperación.

La comunicación efectiva se consolidó al liderar reuniones para alinear cronogramas y transmitir avances técnicos de manera clara y comprensible, desarrollando mi habilidad para simplificar conceptos complejos y promover la transparencia, un reflejo de honestidad en mi interacción con el equipo. En cuanto a la resolución de problemas, las dificultades técnicas que impulsaron a afectaron el progreso y me enseñaron a escudriñar situaciones, depurar errores y ofrecer soluciones innovadoras de manera oportuna e innovadora. Las anteriores potenciaron mi resiliencia y mi capacidad para pensar críticamente cuando más se necesita. Mi gestión del tiempo se desarrolló de una manera en la que trabajar contra el reloj

me ayudó a mirar las tareas frente a mí y priorizar las tareas críticas temprano en el proceso. Me ayudó a acercarme al proyecto de una manera más racional, sobre todo, mejorando mi capacidad de organización y adaptación en un entorno dinámico y asegurando que mantuviera una orientación disciplinada hacia la performance.

Competencias Profesionales

Durante mi desempeño en el proyecto, consolidé un conjunto de competencias profesionales que contribuyeron a mi perfil como graduado. En mi experiencia de autoaprendizaje, la resolución de problemas me llevó a desafíos técnicos que me llevaron a explorar recursos externos y documentación. Consolidé mi capacidad de adquirir y aplicar de manera independiente los conocimientos necesarios de manera responsable para mejorar los procesos en cuestión. En términos de ética profesional, mi desempeño en este proyecto se adhirió a los estándares normativos para gestionar la información de manera responsable. Esto ayudó a afianzar un enfoque honesto y comprometido de la legislación para cumplir con todo lo que requería y proteger los datos sensibles. Finalmente, mi adaptabilidad se consolidó a través de la evolución del proyecto desde la primera fase hasta la ejecución. Me adapto rápidamente a los cambios no planificados y estoy lista para trabajar en entornos cambiantes. Junto con otras competencias, me llevan a un perfil profesional versátil.

REFERENCIAS

- Bootstrap. (2023). Bootstrap documentation. <https://getbootstrap.com/docs/5.3/>
- Chopra, S., & Meindl, P. (2016). Supply chain management: Strategy, planning, and operation (6th ed.). Pearson.
- Cunningham, W., & Beck, K. (2023). Secure software development practices. *Journal of Software Security*, 15(3), 45–60.
- Django Software Foundation. (2023). Django documentation (Version 4.2). <https://docs.djangoproject.com/en/4.2/>
- Elmasri, R., & Navathe, S. B. (2015). Fundamentals of database systems (7th ed.). Pearson.
- ISO/IEC. (2022). ISO/IEC 27001:2022: Information security management systems. International Organization for Standardization. <https://www.iso.org/standard/27001>
- Laudon, K. C., & Traver, C. G. (2021). E-commerce: Business, technology, society (15th ed.). Pearson.
- Ley Federal de Protección de Datos Personales en Posesión de los Particulares. (2025). Diario Oficial de la Federación. México. https://www.dof.gob.mx/nota_detalle.php?codigo=5525980&fecha=05/07/2010
- Mozilla Developer Network. (2023). JavaScript documentation. <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- Nielsen, J. (1994). Usability engineering. Morgan Kaufmann.
- OWASP. (2023). OWASP top ten web security risks. <https://owasp.org/www-project-top-ten/>
- ReportLab. (2023). ReportLab documentation. <https://www.reportlab.com/docs/>
- Schwaber, K., & Sutherland, J. (2017). The scrum guide. <https://scrumguides.org/scrum-guide.html>
- Servicio de Administración Tributaria. (2023). Guía de facturación electrónica CFDI 4.0. <https://www.sat.gob.mx/consultas/57016/guia-de-llenado-de-cfdi>
- Sommerville, I. (2016). Software engineering (10th ed.). Pearson.
- Stallings, W. (2017). Cryptography and network security: Principles and practice (7th ed.). Pearson.

ANEXOS

Anexo B: Manual de Usuario

El presente manual de usuario tiene la finalidad de instruir a los usuarios, en este caso a los clientes —estudiantes, docentes y público en general— y a los administradores en el uso eficiente del Módulo de Ventas de la tienda virtual del Instituto Tecnológico de Apizaco. Este módulo de gestión en realidad fue desarrollado como parte del proyecto de residencia profesional en el periodo de enero a junio de 2025, para dar solución a la problemática de proceso de ventas de los productos institucionales: venta en sí, pagos, administración de inventarios, normatividad, protección de datos personales. De tal manera la tienda, les permitirá a los clientes, una plataforma segura e intuitiva de uso, para la selección de productos institucionales, métodos de pagos alternativos como transacción bancaria y apartado,

Así mismo para los administradores podrán validar pagos por transferencia, administrar el inventario, y obtener reporte detallados. Todos los capítulos han sido desarrollados de una forma clara, simple y puntual, buscando hacer entender al receptor los pasos a seguir para tener la mejor experiencia y aprovechase de las funcionalidades.

Usuario

En cuanto al cliente, esta guía tiene por objeto facilitar el conocimiento del uso del módulo de ventas de la tienda virtual del Instituto Tecnológico de Apizaco para realizar compras eficientes, seguras y desde cualquier dispositivo.

Inicio de Sesión

Para comenzar, debes iniciar sesión con tus credenciales para poder realizar una compra, ingresando tu correo registrado y tu contraseña. (Figura 72)



The image shows a login form titled "Iniciar Sesión" with the subtitle "Ingresa tus datos para continuar". The form contains the following elements:

- Correo electrónico:** A text input field containing "ava3@gmail.com". A red arrow points to this field with the label "Campo para correo electronico".
- Contraseña:** A text input field containing "*****". A red arrow points to this field with the label "Campo para contraseña".
- Recordar sesión:** A checkbox labeled "Recordar sesión".
- ¿Olvidaste tu contraseña?:** A link to recover the password. A red arrow points to the eye icon in the password field with the label "Botón para visualizar contraseña".
- Iniciar Sesión:** A green button to submit the login form. A red arrow points to this button with the label "Botón para iniciar sesión".
- ¿No tienes cuenta? Regístrate aquí:** A link to register a new account.

Figura 72 M_Inicio_sesión_usuario

Carrito

Una vez elegido el producto accederemos al carrito donde veremos el producto y su precio, el usuario podrá incrementar o decrementar la cantidad del producto seleccionado teniendo en cuenta la cantidad de stock disponible y teniendo la opción de borrar el carrito mediante los botones de acción, y de igual forma existen los botones para regresar y seguir explorado y el botón para continuar con la compra (Figura 73).



Figura 73 M_Carrito

Checkout

Para acceder al checkout debimos de proceder con la compra en la pantalla de carrito, una vez dentro podremos observar datos personales los cuales podremos modificar según lo que el usuario requiera mediante los campos de texto, a la derecha encontraremos un resumen de la compra con sus datos respectivos y de igual forma un botón para regresar a la página y seguir explorando y un botón para continuar con la compra (Figura 74).

Datos personales para la venta

Datos Personales

Nombre(s)

Apellidos

Correo electrónico

Teléfono

Debe tener 10 dígitos (ejemplo: 5512345678).

☐ Usar dirección guardada

Dirección

Detalles de la dirección

Campos de texto editables

Resumen de la Compra

Producto	Cantidad	Precio
 Gorra	1	\$100,00 \$100,00 cada uno

Subtotal \$100,00

Total \$100,00

Datos generales de la compra

Botón "Continuar comprando" Botón "Continuar con el pago"

Figura 74 M_ Checkout.

Payments

Al acceder a la pantalla payments de igual manera encontraremos un resumen de tus datos personales la revisión final de los productos seleccionados, precio del producto y a su vez los diferentes tipos de pago, para generar tu baucher deberás usar el botón “ver detalles” para poder generar tu baucher en formato PDF mediante el botón, justo debajo encontraras una inscripción con las instrucciones para dicho método de pago (Figura 75).

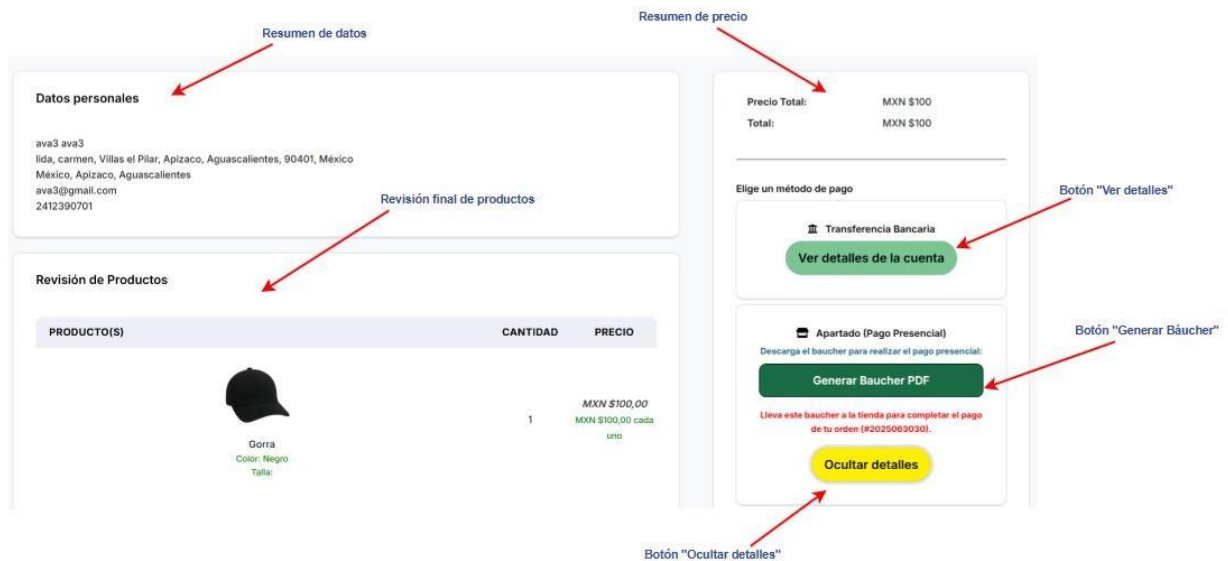


Figura 75 M_Payments.

Panel del usuario/Notificaciones

Desde la barra de navegación accede a “mi cuenta”, veras varias opciones como lo son mi panel, editar perfil, mis órdenes de compra, cerrar sesión etc. Nos enfocaremos y entraremos al apartado de Notificaciones donde encontraremos todas las notificaciones de las ordenes registradas mediante una tabla describiendo el tipo, mensaje, fecha, el estado y los botones de acción correspondientes (Figura 76).

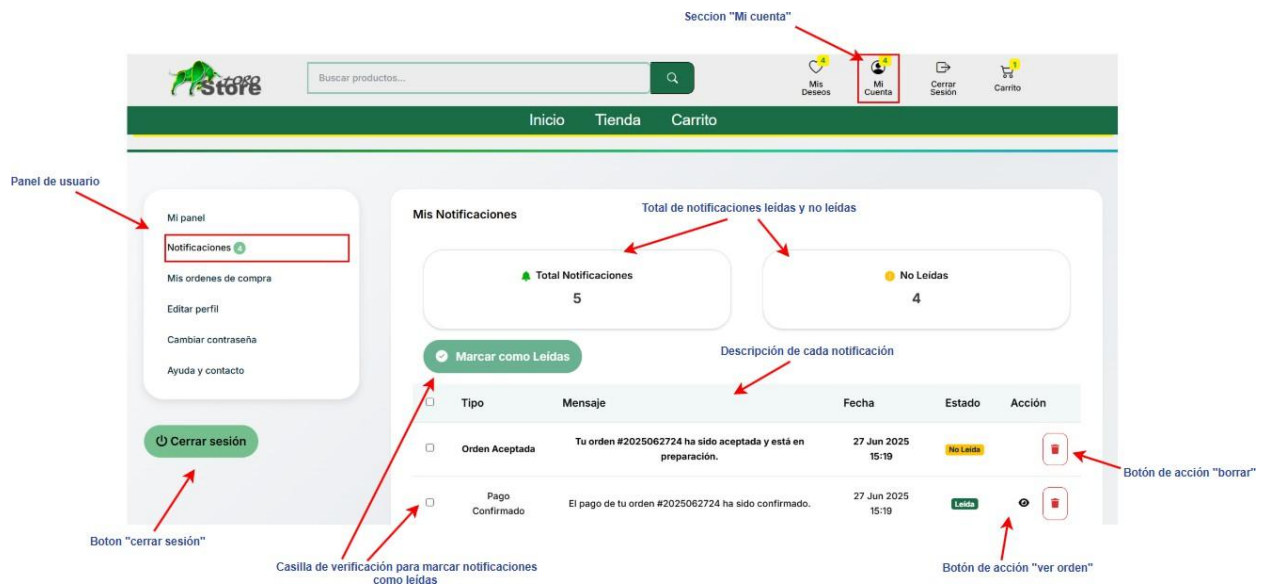


Figura 76 M_Mi cuenta/Notificaciones

Panel del usuario/Mis órdenes de compra

Desde la barra de navegación de igual manera accederemos a “mi cuenta” y entraremos al apartado de Mis órdenes de compra donde encontraremos todas las ordenes registradas mediante una tabla que contiene datos como: Número de orden, Nombre de facturación, Teléfono, Total, Fecha y botones de acción para poder a detalle cada orden (Figura 77).

The screenshot displays the 'Mi cuenta' (My account) section of a web application. The top navigation bar includes links for 'Inicio', 'Tienda', and 'Carrito'. The 'Mi cuenta' link is highlighted in the top right corner. On the left, a sidebar menu lists options: 'Mi panel', 'Notificaciones', 'Mis órdenes de compra' (highlighted), 'Editar perfil', 'Cambiar contraseña', and 'Ayuda y contacto'. Below this menu is a 'Cerrar sesión' button. The main content area is titled 'Órdenes de compra' and contains a table with the following columns: 'Nº de orden', 'Nombre de facturación', 'Teléfono', 'Total', 'Fecha', and 'Acción'. The table lists four orders, all with a total of \$100.00 and a date of 30 Jun 2025. Each order has a circular icon in the 'Acción' column, which is labeled as the 'Botón de acción "ver detalles"'. Red arrows point to the 'Mi Cuenta' link, the 'Mis órdenes de compra' menu item, the 'Cerrar sesión' button, and the action icons in the table.

Nº de orden	Nombre de facturación	Teléfono	Total	Fecha	Acción
2025063029	ava3 ava3	2412390701	\$100,00	30 Jun 2025	ⓘ
2025062724	ava3 ava3	2412390701	\$100,00	27 Jun 2025	ⓘ
2025062723	ava3 ava3	2412390701	\$100,00	27 Jun 2025	ⓘ
2025062722	ava3 ava3	2412390701	\$100,00	27 Jun 2025	ⓘ

Figura 77 M_Mi cuenta/Mis órdenes de compra.

Administrador

Asimismo, esta guía se hace con el fin de que el administrador pueda operar eficientemente el módulo de ventas de la tienda virtual del Instituto Tecnológico de Apizaco para tener control en todo momento de los pedidos, verificación de pagos y generación de informes.

Inicio de Sesión

Para comenzar, debes acceder al panel administrativo e inicia sesión con tus credenciales de administrador (correo y contraseña). (Figura 78).

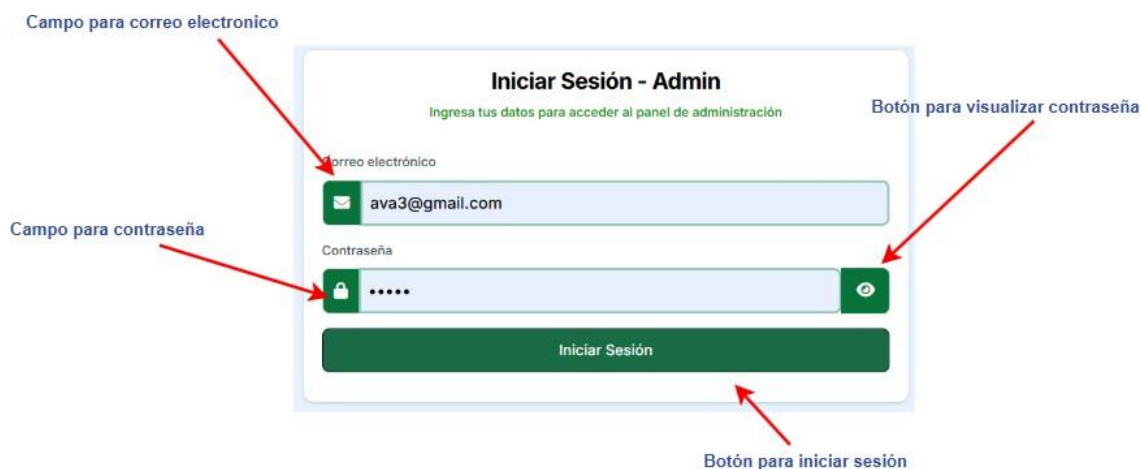


Figura 78 M_Inicio_admin.

Dashboard_home

Una vez dentro podrás ver una barra lateral con todas las herramientas necesarias para la gestión de la página y también se visualiza estadísticas relacionadas a ventas, nosotros nos enfocaremos en 3 zonas específicas: **Notificaciones y Órdenes** (Figura 79).

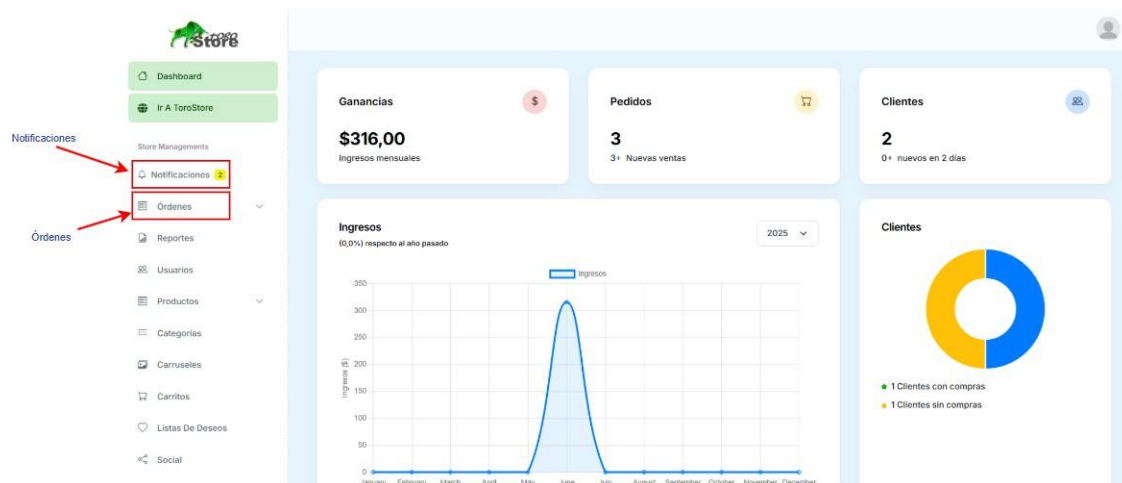


Figura 79 M_Panel_admin.

Sección Notificaciones

Para revisar todas las notificaciones, accede al apartado "Notificaciones" en el panel, donde se registran los eventos con fecha y hora. Responde a ellas según corresponda, mediante los botones de acción ubicados del lado derecho, así como buscar notificaciones según tus requerimientos. En caso de una notificación de stock bajo los botones de acción permiten editar el producto y borrar la notificación. En caso de una nueva orden el botón de acción permitirá visualizar la orden y borrar esta notificación (Figura 80).

Notificaciones

Número de notificaciones totales y no leídas

Total Notificaciones: 2

No Leidas: 0

Buscar por mensaje o número de orden...

Buscar

Datos específicos de cada notificación

Botón de acción "editar producto"

Botón de acción "eliminar notificación"

Botón de acción "visualizar orden"

	Tipo	Mensaje	Fecha	Estado	Acción
☐	Stock Bajo	El producto Pulsera tiene bajo stock: 5 unidades.	27/06/2025 00:18	Leído	 
☐	Orden Apartada	Nueva orden apartada creada: #2025062710 por ava3.	27/06/2025 00:01	Leída	 

Mostrando 1-2 de 2 notificaciones

Casilla de verificación para marcar notificaciones como leídas

Figura 80 M_Notificaciones_admin.

Sección Validar ordenes

La sección de órdenes esta delimitada por 3 secciones las cuales delimitan y organizan mejor las órdenes.

Para acceder y poder validar ordenes nuevas deberemos desglosar “ordenes” y entrar en la sección llamada validar ordenes donde encontraremos un conteo de ordenes tanto validadas, pendientes por validar nuevas. Las ordenes se clasifican por número de orden y describe elementos clave como la fecha, estado, método de pago, total, comprobante en casos específicos y botones de acción destinados para validar y cancelar estas mismas (Figura 81).

Buscador

Órdenes clasificadas según su estatus

Dashboard

Ir A ToroStore

Store Management

Notificaciones

Órdenes

Validar Ordenes

Ordenes Validadas

Ordenes Canceladas

Reportes

Usuarios

Productos

Categorías

Carroses

Carritos

Dashboard

Validar Ordenes

Total de Ordenes 20

Ordenes Validadas 3

Pendientes de Validar 16

Nuevas (Ultimo Mes) 20

Buscar por número de orden, usuario o email...

Buscar

Datos específicos de cada orden

Ingrese el número de orden, usuario o email para filtrar los resultados.

Orden	Usuario	Fecha	Estado	Método de Pago	Total	Comprobante	Acción
25062724	ava3	27/06/2025 00:53	Nuevo	Transferencia Bancaria	MXN \$100,00	No subido	 
25062721	ava3	27/06/2025 00:43	Nuevo	Transferencia Bancaria	MXN \$116,00	No subido	 
25062719	ava3	27/06/2025 00:22	Nuevo	Transferencia Bancaria	MXN \$116,00	No subido	 
25062718	ava3	27/06/2025 00:20	Nuevo	Transferencia Bancaria	MXN \$812,00	No subido	 
25062717	ava3	27/06/2025 00:20	Nuevo	Transferencia Bancaria	MXN \$232,00	No subido	 
25062716	ava3	27/06/2025 00:19	Nuevo	Transferencia Bancaria	MXN \$232,00	No subido	 

Botón de acción "cancelar orden"

Botón de acción "validar orden"

Figura 81 M_Ordenes/validar_ordenes/admin.

Sección Ordenes validadas

Para acceder y poder visualizar las ordenes validadas deberemos desglosar “ordenes” y entrar en la sección llamada ordenes validadas donde encontraremos un conteo de ordenes ya validadas. Las ordenes se clasifican por número de orden y describe elementos como la fecha, estado, método de pago, total, comprobante en casos específicos y botones de acción destinados para borrar (Figura 82).

Órdenes Validadas

Dashboard · Órdenes Validadas

Total de Órdenes Validadas: 4

Buscar por número de orden, usuario o email... **Buscar**

Ingresar el número de orden, usuario o email para filtrar los resultados.

# Orden	Usuario	Fecha	Estado	Método de Pago	Total	Comprobante	Accio
325062724	ava3	27/06/2025 00:53	Completed	Transferencia Bancaria	MXN \$100,00	No disponible	[Borrar]
325062723	ava3	27/06/2025 00:50	Completed	Transferencia Bancaria	MXN \$100,00	No disponible	[Borrar]
325062722	ava3	27/06/2025 00:49	Completed	Transferencia Bancaria	MXN \$100,00	No disponible	[Borrar]
325062720	ava3	27/06/2025 00:24	Completed	Transferencia Bancaria	MXN \$116,00	No disponible	[Borrar]

Mostrando 1-4 de 4 órdenes validadas

Figura 82 M_Ordenes/Ordenes_validadas/admin.

Sección Órdenes canceladas

Para acceder y poder visualizar las ordenes canceladas deberemos desglosar “ordenes” y entrar en la sección llamada ordenes canceladas donde encontraremos un conteo de ordenes que han sido canceladas y pendientes por cancelar. Las ordenes se clasifican por número de orden y describe elementos clave como la fecha, estado, método de pago, total, comprobante en casos específicos y botones de acción destinados para borrar (Figura 83).

Órdenes clasificadas según su estatus

Órdenes Canceladas

Dashboard | Órdenes Canceladas

Total de Órdenes: 20

Órdenes Canceladas: 1

Pendientes de Validar: 15

Canceladas (Último Mes): 1

Buscar por número de orden, usuario o email...

Buscar

Datos específicos de cada orden

# Orden	Usuario	Fecha	Estado	Método de Pago	Total	Comprobante	Acciones
025062712	ava3	27/06/2025 00:06	Cancelado	Transferencia Bancaria	MXN \$232.00	No subido	 

Mostrando 1-1 de 1 ordenes

Botón de acción "ver detalles"

Botón de acción "borrar orden"

Figura 83 M_Ordenes/Ordenes_canceladas /admin.

Sección Reportes

Ingresado a la sección “Reportes”, se presenta un conteo de reportes generados, pendientes por generar; y un resumen de reportes de acuerdo con un período de tiempo. Estos reportes se presentan desde su creación con respecto al período de tiempo en el que refieran (diario, semanal, quincenal, mensual). Datos como fecha de generación, total de ventas, número de pedidos procesados y descarga en formato CSV para análisis mediante el botón de acción (Figura 84).

The screenshot shows the 'Generar Reporte de Ventas' (Generate Sales Report) interface. On the left is a sidebar menu with options: Dashboard, Ir A ToroStore, Store Managements, Notificaciones, Órdenes, Reportes (highlighted), Usuarios, Productos, Categorías, Carruseles, Carritos, Listas De Deseos, and Social. The main content area is titled 'Generar Reporte de Ventas' and includes a sub-header 'Dashboard Reportes'. Below this, there are three summary cards: 'Total Ventas MXN \$416,00', 'Total Órdenes 4', and 'Productos Vendidos 4'. A note above these cards states 'Ventas clasificadas por su total así como el número de órdenes y productos'. Below the summary cards is a 'Formulario de Reporte' (Report Form) with three fields: 'Periodo' (set to 'Diario'), 'Formato' (set to 'PDF'), and 'Fecha de Inicio' (set to 'dd/mm/aaaa'). A green 'Generar Reporte' button is located below the 'Periodo' field. Red arrows point to various elements: one to the 'Reportes' menu item, one to the 'Generar Reporte' button (labeled 'Botón para generar el reporte'), one to the 'Periodo' dropdown (labeled 'Selección del periodo'), one to the 'Formato' dropdown (labeled 'Selección del formato'), and one to the 'Fecha de Inicio' field (labeled 'Selección de la fecha de inicio').

Figura 84 M_Reportes/admin.