

**Manual de Técnico**

**Por:** Angel Perez

[Introducción 4](#_Toc197467528)

[Objetivo del manual 4](#_Toc197467529)

[Público al que está dirigido 4](#_Toc197467530)

[Breve descripción del sistema 4](#_Toc197467531)

[Requisitos para el uso del sistema 5](#_Toc197467532)

[Estructura del proyecto 6](#_Toc197467533)

[Tecnologías implementadas en el proyecto: 7](#_Toc197467534)

[Diagrama EER 9](#_Toc197467535)

[Descripción de tablas principales 10](#_Toc197467536)

[Descripción de las aplicaciones 15](#_Toc197467537)

[Mantenimiento 16](#_Toc197467538)

[Errores comunes y cómo resolverlos 17](#_Toc197467539)

[Variables de entorno (.env) 18](#_Toc197467540)

[EndPoints del sistema: 21](#_Toc197467541)

Introducción

Este manual proporciona una guía clara y concisa para comprender y utilizar el sistema PuntoXpress. Está diseñado para apoyar a los desarrolladores en la correcta utilización del sistema, abarcando desde su propósito hasta una breve hasta buenas prácticas y solución de problemas.

Objetivo del manual

El objetivo de este manual es proporcionar la información necesaria para el uso correcto y eficiente del sistema PuntoXpress, incluyendo su instalación, configuración básica, descripción general de su funcionamiento etc.

Público al que está dirigido

Este manual está dirigido a futuros desarrolladores de PuntoXpress, especialmente aquellos involucrados en tareas de gestión y programación y desarrollo del sistema.

Breve descripción del sistema

PuntoXpress es un sistema de gestión desarrollado para un cualquier comercio. Permite administrar productos, ventas, inventario y usuarios. Está diseñado para facilitar las operaciones diarias, mejorar la organización interna y brindar herramientas efectivas para la toma de decisiones. Incluye roles específicos como administrador, gerente de inventario y vendedor, permitiendo un control segmentado y seguro de las funcionalidades.

Requisitos para el uso del sistema

Para utilizar correctamente la plataforma de PuntoXpress, es importante cumplir con los siguientes requisitos técnicos y de acceso:

**Requisitos técnicos:**

**Navegador actualizado:** Se recomienda usar Google Chrome, Mozilla Firefox o Microsoft Edge.

**Conexión a internet estable**: Para garantizar una carga correcta de las funciones.

**Resolución mínima recomendada:** 1280x720 (HD). Funciona mejor en pantallas medianas o grandes.

**Requisitos de acceso:**

Tener una cuenta activa en el sistema.

Contar con un correo electrónico válido para recibir notificaciones y restablecer contraseña.

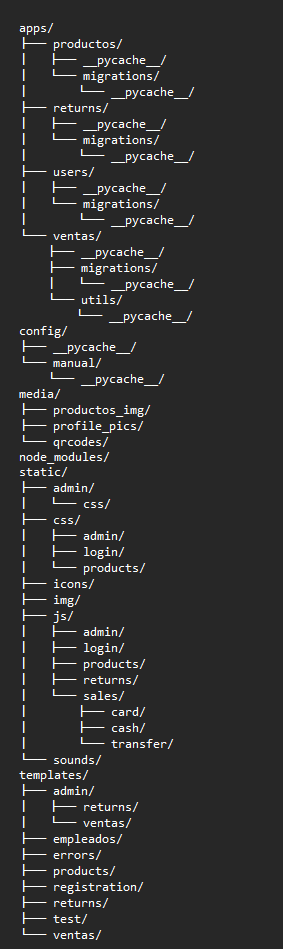
Haber recibido el enlace de activación en caso de ser nuevo usuario (solo en primer ingreso).

**Compatibilidad:**

Compatible con computadoras de escritorio, laptops y tabletas.

Funcional en dispositivos móviles, aunque se recomienda usar en pantalla más amplia para mejor experiencia.

Estructura del proyecto



Tecnologías implementadas en el proyecto:

**Front-end:**

* **Tecnologías base:**
* HTML
* CSS
* JavaScript
* **Librerías y herramientas:**
* TailwindCss
* FlowBite
* Google Fonts
* Font-Awesome
* Heroicons
* MerakiUI
* Wickedblocks
* Django Templates
* UnDraw
* Django-Jazmín

**Back-end:**

* **Tecnologías base:**
* Django (Python)
* MySQL
* **Librerías y herramientas:**
* Hunter.io (API para verificar los correos electrónicos)
* Stripe (API para los pagos con tarjeta.)
* Chart.js (API para los gráficos.)
* Gmail (API para enviar emails.)
* Html5QrCode (API para manejar los códigos QRs)

**Instalación y Configuración**

**1. Clonar el repositorio**

Abre una terminal y ejecuta los siguientes comandos:

git clone https://github.com/tuusuario/tu-repositorio.git

cd tu-repositorio

**2. Crear y activar el entorno virtual**

En sistemas Unix/macOS:

python -m venv venv

source venv/bin/activate

En Windows:

python -m venv venv

venv\Scripts\activate

**3. Instalar las dependencias**

Ejecuta:

pip install -r requirements.txt

**4. Configurar la base de datos y aplicar migraciones**

Edita el archivo settings.py con los datos de conexión a tu base de datos y luego ejecuta:

python manage.py migrate

**5. Crear un superusuario para el panel administrativo**

python manage.py createsuperuser

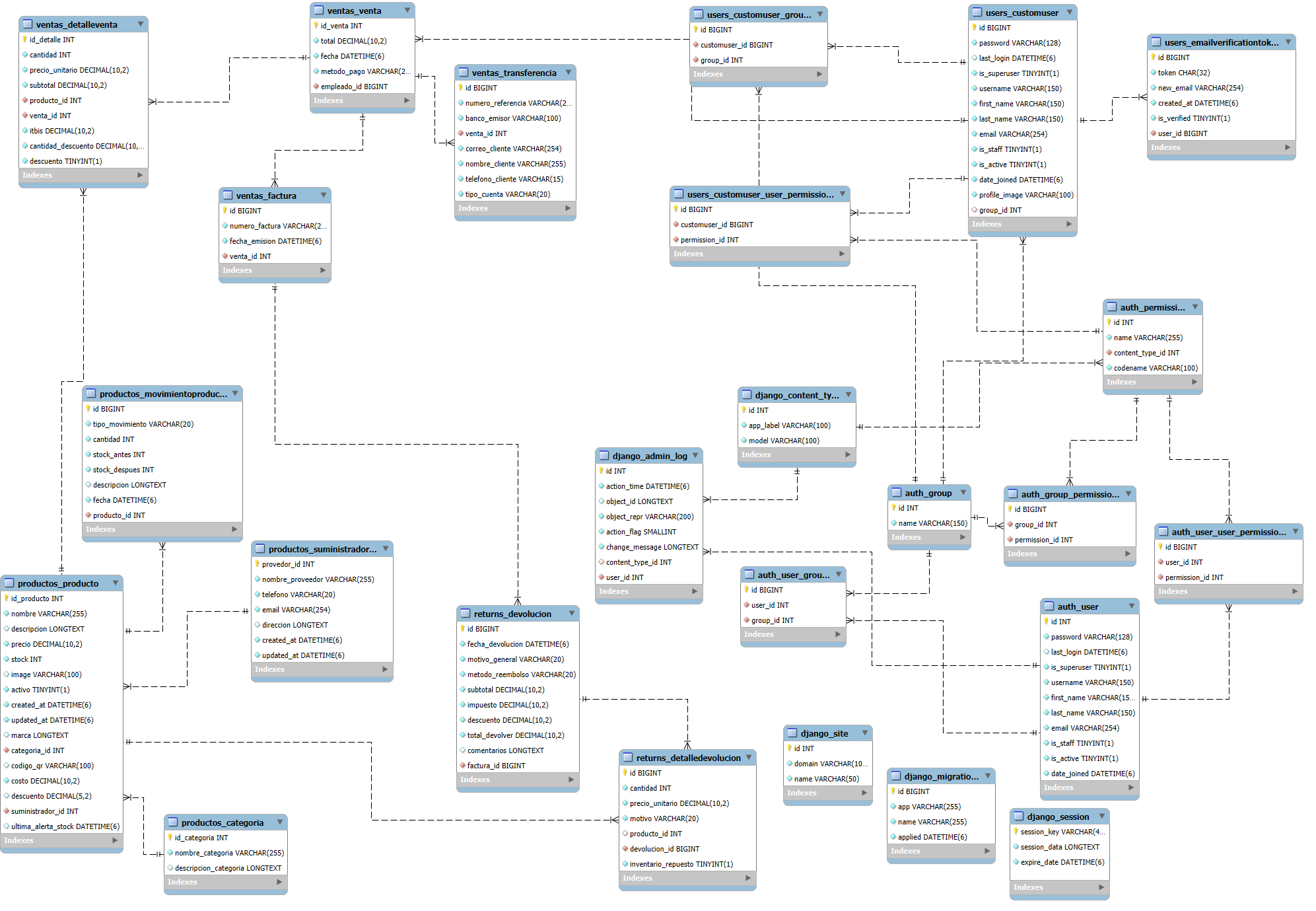
**6. Ejecutar el servidor de desarrollo**

python manage.py runserver

**7. Acceder a la aplicación**

Abre tu navegador y visita:  
[http://127.0.0.1:8000/](http://127.0.0.1:8000/" \t "_new)

Diagrama EER **(**Diagrama Entidad-Relación Extendido**)**



Descripción de tablas principales

En este apartado describiremos todas las tablas excepto por las de Django ya que son tablas que vienen con cualquier proyecto de Django y son generales.

users\_customuser está tabla hereda de auth\_users solo agregamos el campo de profile\_image y pusimos el campo email en unique para que sea único.

**users\_ emailverificationtoken**

esta tabla es usada al momento de cambiar un email creándose un token único para verificar l email.

Los campos son:

* **user:** Es una relación mucho a uno con la tabla users\_customuser.
* token: Es un campo de tipo **UUID (Identificador Universal Único)** es un tipo de dato que genera aleatoriamente un id único
* **new\_email:** Es un campo que almacena el nuevo correo electrónico.
* **created\_at:** Es un campo de tipo date que almacena la fecha en la que se creo ese token
* **is\_verified:** Es un campo tipo booleano que se usa para decir si el correo nuevo ya esta verificado.

**ventas\_venta**

esta tabla es usada para almacenar las ventas tiene campos como:

* **id\_venta:** es el id que se usa para identificar una venta.
* **Empleado:** Es una relación mucho a uno con la tabla users\_customuser.
* **Total:** guarda el total de dinero de la venta
* **Fecha:** almacena la fecha en la que se hizo la venta.
* **metodo\_pago:** almacena el método de pago con el que se hizo la venta.

**ventas\_transferencia**  
Esta tabla almacena la información relacionada con una transferencia realizada como método de pago de una venta. Los campos son:

* **venta**: Relación de *muchos a uno* con la tabla **Venta**.
* **numero\_referencia**: Es el número de referencia de la transferencia.
* **banco\_emisor**: Es el nombre del banco que emite la transferencia.
* **nombre\_cliente**: Es el nombre del cliente que realizó la transferencia.
* **telefono\_cliente**: Es el teléfono del cliente.
* **correo\_cliente**: Es el correo electrónico del cliente.
* **tipo\_cuenta**: Indica el tipo de cuenta desde la cual se hizo la transferencia (Cuenta Corriente o Cuenta de Ahorros).

**Ventas\_detalleVenta**  
Esta tabla almacena los detalles de cada producto vendido en una venta. Los campos son:

* **id\_detalle**: Es el identificador único del detalle de la venta.
* **venta**: Es una relación de clave foránea con la tabla **Venta**.
* **producto**: Es una relación de clave foránea con la tabla **Producto** (desde el archivo productos.models).
* **cantidad**: Es la cantidad de unidades del producto vendido.
* **precio\_unitario**: Es el precio unitario del producto en el momento de la venta.
* **subtotal**: Es el subtotal de la venta (cantidad \* precio\_unitario).
* **itbis**: Es el monto del ITBIS calculado como el 18% del subtotal.
* **descuento**: Indica si se aplicó un descuento a este detalle de venta.
* **cantidad\_descuento**: Es el monto del descuento aplicado en este detalle de venta.

**Ventas\_Factura**  
Esta tabla almacena la factura generada para cada venta. Los campos son:

* **venta**: Relación de *uno a uno* con la tabla **Venta**.
* **numero\_factura**: Es el número único de la factura (por ejemplo, F-000001) \*las ventas que tengan el método de pago en efectivo tienen el sufijo EF si son por transferencia TR y si son con tarjeta T\*.
* **fecha\_emision**: Es la fecha y hora en que se generó la factura.

**Productos\_Categoria**  
Esta tabla almacena las categorías de productos. Los campos son:

* **id\_categoria**: Identificador único de la categoría.
* **nombre\_categoria**: Nombre de la categoría, que debe ser único.
* **descripcion\_categoria**: Descripción de la categoría (opcional).

**Productos\_Suministradores**  
Esta tabla almacena la información de los proveedores de productos. Los campos son:

* **proveedor\_id**: Identificador único del proveedor.
* **nombre\_proveedor**: Nombre del proveedor, único.
* **telefono**: Teléfono de contacto del proveedor.
* **email**: Correo electrónico del proveedor, único.
* **direccion**: Dirección del proveedor (opcional).
* **created\_at**: Fecha y hora en que se creó el registro.
* **updated\_at**: Fecha y hora de la última actualización del registro.

**Productos\_Producto**  
Esta tabla almacena los productos. Los campos son:

* **id\_producto**: Identificador único del producto.
* **nombre**: Nombre del producto.
* **descripcion**: Descripción del producto (opcional).
* **precio**: Precio del producto.
* **costo**: Costo del producto.
* **descuento**: Descuento aplicado al producto (opcional).
* **stock**: Cantidad disponible del producto en inventario.
* **image**: Imagen del producto (por defecto se usa una imagen sin foto).
* **activo**: Indica si el producto está activo o no.
* **created\_at**: Fecha y hora en que se creó el producto.
* **marca**: Marca del producto (opcional).
* **updated\_at**: Fecha y hora de la última actualización del producto.
* **categoria**: Relación de *muchos a uno* con la tabla **Categoria**.
* **suministrador**: Relación de *muchos a uno* con la tabla **Suministradores**.
* **codigo\_qr**: Imagen del código QR generado para el producto (opcional).
* **ultima\_alerta\_stock**: Fecha y hora en la que se generó la última alerta de stock bajo (opcional).

**Productos\_MovimientoProducto**  
Esta tabla almacena los movimientos de stock de productos. Los campos son:

* **producto**: Relación de *muchos a uno* con la tabla **Producto**.
* **tipo\_movimiento**: Tipo de movimiento (Entrada, Salida, Actualización, Creación).
* **cantidad**: Cantidad de producto en el movimiento.
* **stock\_antes**: Stock del producto antes del movimiento.
* **stock\_despues**: Stock del producto después del movimiento.
* **descripcion**: Descripción del movimiento (opcional).
* **fecha**: Fecha y hora en la que ocurrió el movimiento.

**Returns\_Devolucion**  
Esta tabla almacena la información de las devoluciones realizadas por los clientes. Los campos son:

* **factura**: Relación de *muchos a uno* con la tabla **Factura**.
* **fecha\_devolucion**: Fecha y hora en la que se registró la devolución.
* **motivo\_general**: Motivo principal de la devolución, con opciones como *Producto defectuoso*, *Producto dañado*, o *Otro motivo*.
* **metodo\_reembolso**: Método utilizado para el reembolso, con opciones como *Método original de pago*, *Crédito en tienda*, *Transferencia bancaria*, o *Efectivo*.
* **subtotal**: Subtotal de los productos devueltos.
* **impuesto**: Importe del impuesto aplicado a los productos devueltos (por ejemplo, ITBIS).
* **descuento**: Descuento aplicado a la devolución.
* **total\_devolver**: Total que se va a devolver al cliente, calculado automáticamente sumando el subtotal, el impuesto y restando el descuento.
* **comentarios**: Comentarios adicionales sobre la devolución (opcional).

**Returns\_DetalleDevolucion**  
Esta tabla almacena los detalles de los productos que fueron devueltos en una devolución. Los campos son:

* **devolucion**: Relación de *muchos a uno* con la tabla **Devolucion**.
* **producto**: Relación de *muchos a uno* con la tabla **Producto**.
* **cantidad**: Cantidad de unidades devueltas del producto.
* **precio\_unitario**: Precio unitario del producto devuelto.
* **inventario\_repuesto**: Indica si el inventario se ha repuesto debido a la devolución del producto.
* **motivo**: Motivo específico para la devolución de este producto, con opciones como *Producto defectuoso*, *Producto dañado*, o *Otro motivo*.

Descripción de las aplicaciones

**App de Productos:** Esta aplicación gestiona toda la información relacionada con los productos que se venden. Incluye detalles como el nombre, descripción, precio, stock y la categoría del producto. Además, permite realizar movimientos de inventario, como entradas y salidas, y genera códigos QR para cada producto. La app también gestiona las imágenes de los productos, lo que permite una mejor visualización en las plataformas de venta.

**App de Ventas:** La aplicación de ventas es responsable de registrar las transacciones de compra. Gestiona las facturas generadas para cada venta, el detalle de los productos comprados, las cantidades y el total. Además, permite el seguimiento de los pagos realizados, descuentos aplicados y los impuestos correspondientes a cada transacción. Esta app facilita el control de las ventas realizadas y la facturación de las mismas.

**App de Devoluciones:** Esta aplicación gestiona las devoluciones de productos realizadas por los clientes. Permite registrar cada devolución, asociándola a una factura y especificando los motivos de la devolución. Además, calcula automáticamente el total a devolver, considerando el subtotal, impuestos y descuentos. La app también lleva un control detallado de los productos devueltos y la reposición de inventario, lo que facilita la gestión de las existencias.

**App de Usuarios:** La aplicación de usuarios maneja toda la gestión de cuentas de clientes y administradores. Permite registrar, autenticar y administrar los datos de los usuarios, como nombre, correo electrónico y contraseña. También gestiona las preferencias y notificaciones de los usuarios, y ofrece una capa de seguridad mediante la verificación de cuentas. Esta app es esencial para garantizar que solo los usuarios autorizados puedan acceder a las funciones del sistema y realizar transacciones, como ventas o devoluciones.

**Interrelación entre las Apps:** Las aplicaciones de productos, ventas, devoluciones y usuarios están interconectadas para crear un sistema completo de gestión. Los usuarios pueden realizar compras, y si es necesario, gestionar devoluciones asociadas a las facturas generadas. Los productos y las existencias se actualizan automáticamente en función de las ventas y devoluciones, lo que asegura un control eficiente del inventario y una administración clara de todas las transacciones.

Mantenimiento

**Mantenimiento Preventivo y Actualización de Dependencias:**

* **Base de Datos:** Para asegurar que el sistema funcione de manera óptima, debemos realizar un mantenimiento regular de la base de datos. Esto incluye la optimización de las consultas, la actualización de índices y la depuración de registros innecesarios. Por ejemplo, podríamos realizar tareas como la compactación de tablas, especialmente en las aplicaciones de productos y ventas, para asegurarte de que las transacciones y búsquedas sean rápidas. Además, es importante mantener las versiones de los sistemas de gestión de bases de datos (como PostgreSQL o MySQL) actualizadas.
* **Dependencias del Proyecto:** Con cada nueva versión de dependencias como Django o librerías adicionales, debes verificar que no haya actualizaciones disponibles que corrijan errores de seguridad o mejoren el rendimiento. Para hacerlo, ejecuta comandos como pip list --outdated y actualiza las dependencias utilizando pip install --upgrade para mantener el sistema a la vanguardia.

**Seguridad y Cumplimiento**:

* **Actualización de Seguridad**: Asegúrate de que todas las dependencias de tu sistema estén actualizadas y no contengan vulnerabilidades conocidas. Puedes realizar esto regularmente utilizando herramientas como **Snyk** o **Dependabot** para recibir notificaciones sobre vulnerabilidades.
* **Ley de Protección de Datos Personales**. Esto implica implementar medidas como el cifrado de la base de datos, auditorías de acceso y la obtención del consentimiento explícito de los usuarios para procesar sus datos.

**Backup y Recuperación**:

* Realiza copias de seguridad regulares de las bases de datos y archivos estáticos/media. Si se pierde información o se presentan fallos en el sistema, una copia de seguridad confiable permitirá restaurar el servicio rápidamente. Además, es fundamental probar las estrategias de recuperación ante desastres (disaster recovery) para asegurarte de que los datos se puedan recuperar de manera eficiente en caso de un fallo crítico.

Errores comunes y cómo resolverlos

A continuación, se detallan los errores más comunes que los usuarios pueden encontrar al interactuar con el sistema **PuntoXpress** y las soluciones correspondientes:

1. **Error 404: Página no encontrada**
   * **Descripción**: Este error ocurre cuando el recurso solicitado no se puede encontrar en el servidor.
   * **Solución**: Verifique que la URL ingresada sea correcta. Si es una URL interna, asegúrese de que la ruta del recurso esté configurada adecuadamente en el sistema.
2. **El servidor no responde (runserver caído)**
   * **Descripción**: Este error ocurre cuando el servidor de desarrollo (runserver) se cae y deja de funcionar temporalmente.
   * **Solución**: Reinicie el servidor ejecutando el comando python manage.py runserver en la terminal. Si el problema persiste, revise los logs del servidor para detectar errores más específicos y proceder con la corrección.
3. **Clave de API vencida**
   * **Descripción**: Este error ocurre cuando la clave de API utilizada en el sistema ha expirado.
   * **Solución**: La solución es renovar la clave de API. Acceda al portal del proveedor de la API y obtenga una nueva clave, luego actualice el sistema con la nueva clave.
4. **Error 500: Error interno del servidor**
   * **Descripción**: Este error indica que el servidor encontró una condición inesperada que impide que se cumpla la solicitud.
   * **Solución**: Revise los logs del servidor para encontrar el detalle del error. Puede estar relacionado con problemas en el código, la base de datos o el entorno del servidor.

Variables de entorno (.env)

**SECRET\_KEY**

* **Qué es:** Es una cadena secreta utilizada por Django para varios propósitos de seguridad, como la firma de cookies y la protección contra ataques.
* **Razón de uso:** Se usa para garantizar la seguridad de las sesiones de usuario y las cookies dentro de la aplicación.

**DEBUG**

* **Qué es:** Indica si la aplicación está en modo de desarrollo o en producción. Si está en True, la aplicación proporciona más información detallada en caso de errores.
* **Razón de uso:** Se usa para facilitar el desarrollo, ya que muestra mensajes de error detallados. En producción, debe estar en False para evitar mostrar información sensible.

**ALLOWED\_HOSTS**

* **Qué es:** Es una lista de direcciones de host que el servidor puede servir. Impide que la aplicación acepte solicitudes de hosts no autorizados.
* **Razón de uso:** Se utiliza para proteger tu aplicación de ataques como el envenenamiento de DNS.

**DB\_ENGINE**

* **Qué es**: Define el motor de la base de datos que usará la aplicación, como django.db.backends.postgresql, django.db.backends.mysql, etc.
* **Razón de uso:** Indica qué sistema de base de datos utilizará la aplicación.

**DB\_NAME**

* **Qué es:** El nombre de la base de datos que utiliza la aplicación.
* **Razón de uso:** Se usa para especificar qué base de datos se debe conectar.

**DB\_USER**

* **Qué es:** El nombre de usuario de la base de datos.
* **Razón de uso:** Es necesario para autenticar la conexión con la base de datos.

**DB\_PASSWORD**

* **Qué es:** La contraseña del usuario de la base de datos.
* **Razón de uso:** Se usa para autenticar al usuario que accederá a la base de datos.

**DB\_HOST**

* **Qué es:** La dirección del host de la base de datos (generalmente localhost o la IP de un servidor remoto).
* **Razón de uso**: Indica el lugar donde está alojada la base de datos.

**DB\_PORT**

* **Qué es:** El puerto en el que la base de datos escucha las solicitudes de conexión.
* **Razón de uso:** Especifica el puerto de conexión de la base de datos.

**EMAIL\_BACKEND**

* **Qué es:** Defineel backend que Django usará para enviar correos electrónicos (por ejemplo, django.core.mail.backends.smtp.EmailBackend).
* **Razón de uso:** Se usa para configurar cómo se enviarán los correos electrónicos (por ejemplo, utilizando un servidor SMTP).

**EMAIL\_HOST**

* **Qué es:** El servidor SMTP que se utilizará para enviar los correos electrónicos**.**
* **Razón de uso:** Se usa para definir el servidor que enviará los correos.

**EMAIL\_PORT**

* **Qué es:** El puerto del servidor SMTP.
* **Razón de uso:** Se usa para indicar qué puerto utilizará el servidor SMTP (normalmente 25, 465 o 587).

**EMAIL\_USE\_TLS**

* **Qué es:** Un valor booleano que indica si debe utilizarse TLS (seguridad) para la conexión SMTP.
* **Razón de uso:** Se usa para asegurar la transmisión de correos electrónicos cifrando la conexión.

**EMAIL\_HOST\_USER**

* **Qué es:** El nombre de usuario utilizado para autenticar el acceso al servidor SMTP.
* **Razón de uso:** Es necesario para autenticar la cuenta del servidor SMTP.

**EMAIL\_HOST\_PASSWORD**

* **Qué es:** La contraseña de la cuenta de correo del servidor SMTP.
* **Razón de uso:** Se usa para autenticar la cuenta en el servidor SMTP.

**DEFAULT\_FROM\_EMAIL**

* **Qué es:** La dirección de correo electrónico predeterminada desde la cual se enviarán los correos electrónicos.
* **Razón de uso:** Se utiliza para establecer la dirección del remitente en los correos electrónicos enviados.

**HUNTER\_API\_KEY**

* **Qué es:** La clave de API de Hunter.io, un servicio para encontrar correos electrónicos y verificar su validez.
* **Razón de uso:** Se usa para autenticar las solicitudes a la API de Hunter.io, generalmente para la verificación de correos electrónicos.

**STRIPE\_PUBLIC\_KEY**

* **Qué es:** La clave pública de Stripe, un servicio de pagos en línea.
* **Razón de uso:** Se usa para integrar Stripe en la aplicación y realizar pagos a través de su API.

**STRIPE\_SECRET\_KEY**

* **Qué es:** La clave secreta de Stripe**.**
* **Razón de uso:** Se usa para autenticar las solicitudes realizadas al servidor de Stripe, generalmente para procesar pagos.

**GOOGLE\_API\_KEY**

* **Qué es:** La clave de API de Google para acceder a sus servicios, como Google Maps, Google Cloud, etc.
* **Razón de uso:** Se usa para autenticar las solicitudes a los servicios de Google y habilitar características como mapas o geolocalización en la aplicación.

EndPoints del sistema:

**Endpoints de Usuario y Autenticación**

* **/accounts/login/**  
  Vista: django.contrib.auth.views.LoginView  
  Nombre de Vista: login
* **/accounts/logout/**  
  Vista: django.contrib.auth.views.LogoutView  
  Nombre de Vista: logout
* **/accounts/password\_change/**  
  Vista: django.contrib.auth.views.PasswordChangeView  
  Nombre de Vista: password\_change
* **/accounts/password\_change/done/**  
  Vista: django.contrib.auth.views.PasswordChangeDoneView  
  Nombre de Vista: password\_change\_done
* **/accounts/password\_reset/**  
  Vista: django.contrib.auth.views.PasswordResetView  
  Nombre de Vista: password\_reset
* **/accounts/password\_reset/done/**  
  Vista: django.contrib.auth.views.PasswordResetDoneView  
  Nombre de Vista: password\_reset\_done
* **/accounts/reset/<uidb64>/<token>/**  
  Vista: django.contrib.auth.views.PasswordResetConfirmView  
  Nombre de Vista: password\_reset\_confirm
* **/accounts/reset/done/**  
  Vista: django.contrib.auth.views.PasswordResetCompleteView  
  Nombre de Vista: password\_reset\_complete
* **/accounts/verify-email/uuid:token/**  
  Vista: apps.users.views.verify\_email  
  Nombre de Vista: verify\_email

**Endpoints Administrativos**

* **/admin/**  
  Vista: django.contrib.admin.sites.index  
  Nombre de Vista: admin:index
* **/admin/<app\_label>/**  
  Vista: django.contrib.admin.sites.app\_index  
  Nombre de Vista: admin:app\_list
* **/admin/login/**  
  Vista: django.contrib.admin.sites.login  
  Nombre de Vista: admin:login
* **/admin/logout/**  
  Vista: django.contrib.admin.sites.logout  
  Nombre de Vista: admin:logout

**Sección de Productos**

* **/admin/productos/categoria/**  
  Vista: django.contrib.admin.options.changelist\_view  
  Nombre de Vista: admin:productos\_categoria\_changelist
* **/admin/productos/categoria/path:object\_id/change/**  
  Vista: django.contrib.admin.options.change\_view  
  Nombre de Vista: admin:productos\_categoria\_change
* **/admin/productos/categoria/add/**  
  Vista: django.contrib.admin.options.add\_view  
  Nombre de Vista: admin:productos\_categoria\_add
* **/admin/productos/producto/**  
  Vista: django.contrib.admin.options.changelist\_view  
  Nombre de Vista: admin:productos\_producto\_changelist
* **/admin/productos/producto/path:object\_id/change/**  
  Vista: django.contrib.admin.options.change\_view  
  Nombre de Vista: admin:productos\_producto\_change
* **/admin/productos/producto/add/**  
  Vista: django.contrib.admin.options.add\_view  
  Nombre de Vista: admin:productos\_producto\_add

**Sección de Ventas y Devoluciones**

* **/admin/ventas/venta/**  
  Vista: apps.ventas.admin.changelist\_view  
  Nombre de Vista: admin:ventas\_venta\_changelist
* **/admin/returns/devolucion/**  
  Vista: apps.returns.admin.changelist\_view  
  Nombre de Vista: admin:returns\_devolucion\_changelist

**Endpoints de Empleados**

* **/empleados/perfil/**  
  Vista: apps.users.views.profile  
  Nombre de Vista: profile
* **/empleados/perfil/cambiar-imagen/**  
  Vista: apps.users.views.update\_profile\_picture  
  Nombre de Vista: update\_profile\_picture
* **/empleados/procesar-transferencia/**  
  Vista: apps.ventas.views.process\_transfer  
  Nombre de Vista: procesar\_transferencia
* **/empleados/procesar-venta/**  
  Vista: apps.ventas.views.process\_sale  
  Nombre de Vista: process\_sale
* **/empleados/procesar\_devolucion/**  
  Vista: apps.returns.views.process\_return\_view  
  Nombre de Vista: process\_return
* **/empleados/productos/**  
  Vista: apps.productos.views.product\_view  
  Nombre de Vista: products

**Endpoints de Reportes y Datos**

* **/api/chartdata/**  
  Vista: apps.ventas.views.chart\_data  
  Nombre de Vista: chart\_data
* **/devoluciones\_por\_periodo/**  
  Vista: apps.returns.views.devoluciones\_por\_periodo  
  Nombre de Vista: devoluciones\_por\_periodo
* **/ventas\_por\_categoria\_data/**  
  Vista: apps.ventas.views.sales\_by\_categories  
  Nombre de Vista: ventas\_por\_categoria\_data

**Endpoints de Archivos y Descargas**

* **/download-return-report/**  
  Vista: apps.returns.views.return\_pdf  
  Nombre de Vista: return\_pdf
* **/download-sale-report/**  
  Vista: apps.ventas.views.sale\_pdf  
  Nombre de Vista: sale\_pdf
* **/empleados/devoluciones/int:devolucion\_id/pdf/**  
  Vista: apps.returns.views.download\_return\_receipt\_pdf  
  Nombre de Vista: descargar\_comprobante\_devolucion\_pdf
* **/empleados/factura/int:sale\_id/pdf/**  
  Vista: apps.ventas.views.descargar\_factura\_pdf  
  Nombre de Vista: descargar\_factura\_pdf

**Endpoints de Otros Servicios**

* **/chat/**  
  Vista: config.manual.support\_bot.chat  
  Nombre de Vista: chat
* **/problem/**  
  Vista: apps.users.views.technical\_problem\_view  
  Nombre de Vista: technical\_problem

**Endpoints de Archivos Estáticos**

* **/media/<path>**  
  Vista: django.views.static.serve

**Nombre del sistema**

PuntoXpress

**Versión**

1.0

**Fecha**

06 de mayo de 2025

**Contacto de Soporte:**

+1 (849) 284-2781

**Licencia:**

MIT License

**Responsable o desarrollador**

Angel Alexander Pérez Martínez