



# API RepaReg



# El software

## Introducción

### Dashboard de Sistema de Facturación

Bienvenidos a la documentación de la API para el Dashboard de Sistema de Facturación de RepaReg, una herramienta integral diseñada para empoderar a los proveedores de soluciones informáticas y a sus clientes. Este dashboard es la culminación de nuestro compromiso con la excelencia y la eficiencia, proporcionando una interfaz unificada para

la gestión y monitoreo de las operaciones de facturación y servicios relacionados.

En RepaReg, entendemos la importancia de una visualización clara y concisa de los datos que impulsa la toma de decisiones informadas. Por ello, nuestro dashboard presenta métricas clave de manera simplificada, permitiendo a los usuarios visualizar y contrastar diferentes aspectos de su negocio en tiempo real.

### **Características y Funcionalidades del Dashboard de Facturación:**

- **Facturación en Tiempo Real:** Emita facturas instantáneamente y optimice el flujo de trabajo con la automatización del envío y la exportación de comprobantes en diversos formatos.
- **Insights de Gestión:** Acceda a informes de gestión y a reportes detallados para impulsar una toma de decisiones estratégica y fundamentada en datos.
- **Control Total de Inventario:** Gestione su stock con herramientas avanzadas para el control y el movimiento de inventario.
- **Análisis de Ventas y Tendencias:** Mejore la visibilidad del rendimiento de su negocio con rankings de ventas detallados por producto, servicio y cliente, complementados con un historial completo de ventas.
- **Seguridad de Datos:** Centralice y resguarde su base de datos de clientes, proveedores y productos, con la capacidad de almacenar y exportar datos en la nube de forma segura.

Estas características están diseñadas para proporcionar una visión completa y control exhaustivo sobre todos los aspectos de la facturación y la gestión empresarial, asegurando que cada cliente pueda maximizar la eficiencia y la efectividad de sus operaciones comerciales a través de nuestro sistema.

La API de RepaReg ofrece una funcionalidad avanzada para el análisis profundo del funcionamiento de una empresa mediante el estudio de datos e informes de gestión. Este sistema facilita el acceso a análisis analíticos y reportes detallados que descubren tendencias, patrones y métricas claves de rendimiento, esenciales para una organización. Estos insights o conocimientos profundos capacitan a los tomadores de

decisiones para comprender mejor la situación actual de su empresa, anticipar escenarios futuros y tomar decisiones estratégicas basadas en datos sólidos y fiables.

En esencia, esta capacidad de la API es vital para convertir los datos brutos en información valiosa, la cual es crucial para optimizar la planificación, las operaciones y las estrategias de negocios.

Con el Dashboard de Sistema de Facturación de RepaReg, ofrecemos una solución completa que permite a los usuarios concentrarse en lo que mejor saben hacer: dirigir sus negocios con confianza y claridad.

## Visión General del Dashboard

El Tablero de Control es la interfaz central de nuestro sistema de gestión, diseñado para proporcionar una visión rápida y actualizada del estado del negocio. Presenta una serie de widgets e indicadores clave que permiten a los usuarios monitorear la actividad comercial en tiempo real:

- **Cantidad de Clientes:** Este panel muestra el número total de clientes registrados, ofreciendo una visión inmediata de la base de clientes.
- **Cantidad de Ventas Hoy:** Refleja las ventas realizadas durante el día actual, proporcionando información actualizada sobre las transacciones diarias.
- **Total Hoy:** Indica los ingresos generados en el día en curso, lo que permite a los usuarios evaluar la rentabilidad diaria de manera instantánea.
- **Ingresos en los Últimos 7 Días:** Representado mediante un gráfico de líneas, este elemento visual destaca los ingresos generados cada día durante la última semana, facilitando la identificación de tendencias y patrones.
- **Servicios en los Últimos 7 Días:** Un gráfico de barras muestra la cantidad de servicios proporcionados, permitiendo comparar rápidamente los niveles de actividad de diferentes días.

- **Desglose de Ventas por Categoría:** El gráfico circular descompone las ventas por categorías de productos como accesorios, componentes y software, dando una visión clara de qué segmentos están impulsando los ingresos.

Este dashboard es una herramienta esencial para la gestión y el seguimiento del rendimiento comercial, y está diseñado para ser intuitivo y fácil de interpretar.

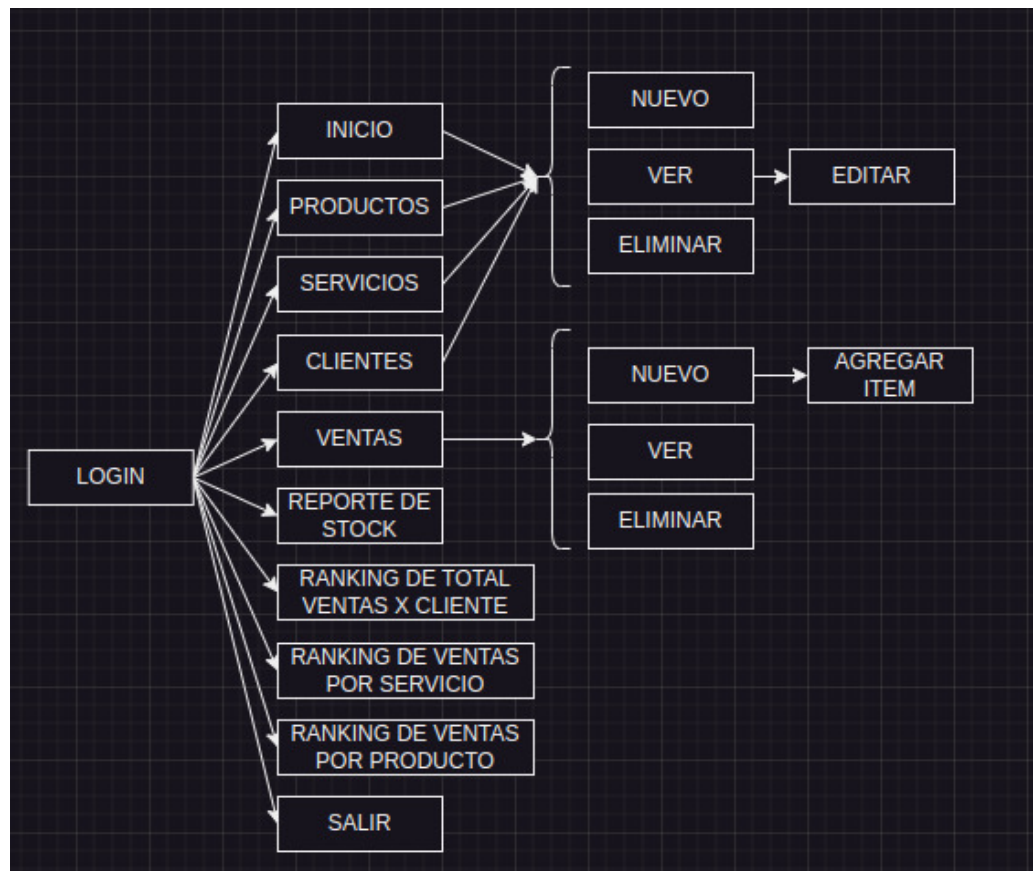


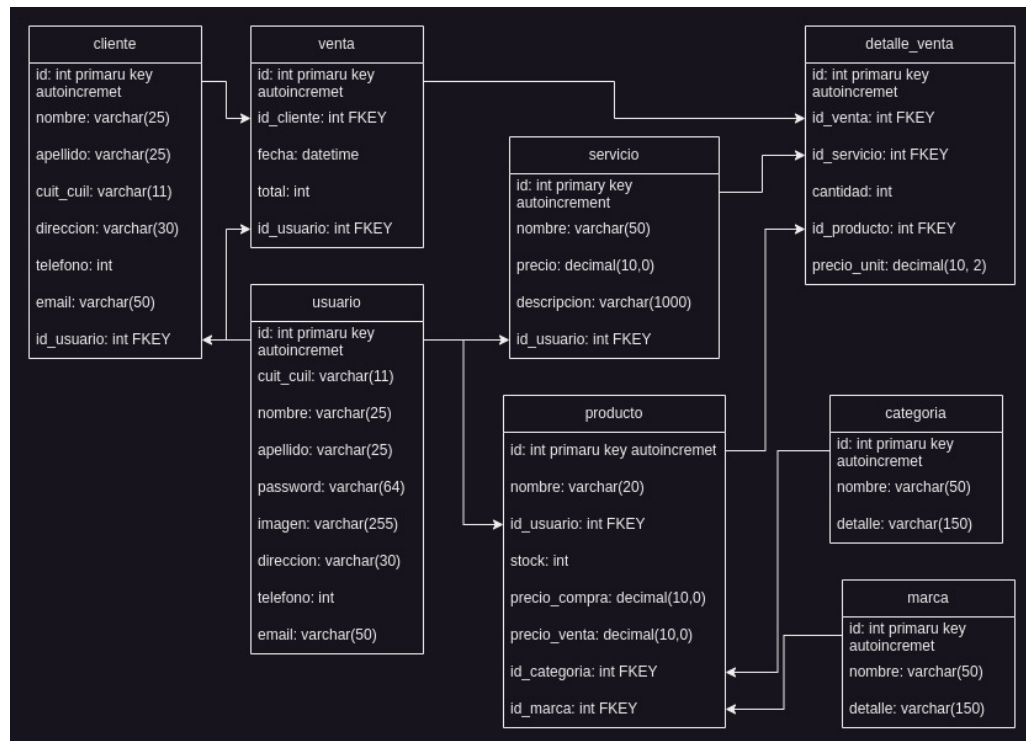
### Gestión de la Base de Datos:

- **Estructura de Datos Robusta:** Nuestra API utiliza una base de datos MySQL para manejar de manera eficiente y segura la información persistente. Esta base de datos está optimizada para garantizar la integridad, la velocidad y la accesibilidad de los datos.
- **Esquemas de Base de Datos:** La estructura de la base de datos está diseñada para soportar todas las operaciones críticas del sistema de facturación y del dashboard, incluyendo la gestión de usuarios, clientes, productos, servicios y facturas.
- **Seguridad y Rendimiento:** Implementamos las mejores prácticas en cuanto a seguridad de la base de datos, asegurando la protección de los datos sensibles. Además, nos centramos en el rendimiento para garantizar tiempos de respuesta rápidos y una experiencia de usuario fluida.

## Modelado UML (Lenguaje Unificado de Modelado):

- **Diagramas UML:** Para visualizar la estructura y el flujo del sistema, hemos desarrollado una serie de diagramas UML. Estos incluyen diagramas de clases, de secuencia, de actividades y de casos de uso, que proporcionan una representación clara de la arquitectura del sistema.





- **Relaciones y Dependencias:** Los diagramas UML detallan las relaciones entre las diferentes entidades del sistema, como usuarios, clientes, productos y facturas, y cómo interactúan entre sí a través de las diversas funcionalidades del sistema.
- **Documentación de Apoyo:** Estos diagramas son una herramienta esencial no solo para el desarrollo y mantenimiento del sistema, sino también para ayudar a los nuevos desarrolladores y partes interesadas a comprender rápidamente la lógica y el funcionamiento del sistema.

### Integración y Mantenimiento:

- **Actualizaciones Consistentes:** A medida que el sistema evoluciona, los diagramas UML y los esquemas de la base de datos se actualizan para reflejar cualquier cambio o mejora, asegurando que la documentación siempre esté alineada con la versión actual del sistema.
- **Colaboración y Transparencia:** Fomentamos una cultura de colaboración y transparencia, donde los diagramas UML y la estructura de la base de datos están disponibles para todos los miembros del equipo, facilitando un entendimiento común y cohesivo del sistema.



En esta sección, hemos proporcionado una visión general de cómo la API de RepaReg gestiona su base de datos y utiliza el modelado UML para asegurar una estructura clara y eficiente. Estos componentes son fundamentales para el éxito del sistema, brindando una base sólida sobre la cual construimos y escalamos nuestras soluciones.

---

### Desarrollo y Tecnología:

- Implementado con Python y Flask en el backend y una base de datos SQL para la persistencia de información.
- El frontend se construye con HTML, CSS y JavaScript, asegurando un diseño responsivo para una experiencia de usuario óptima en dispositivos móviles y de escritorio.
- Pruebas exhaustivas mediante clientes REST y herramientas de testing para garantizar la robustez y fiabilidad de la generación de facturas.

Este proyecto es el resultado de un esfuerzo colaborativo y está sujeto a un riguroso control de versiones y documentación en cada etapa del desarrollo, aplicando las mejores prácticas y el paradigma de programación orientada a objetos.

### Instalación y Descargas

Antes de comenzar a utilizar la API, es necesario completar el proceso de instalación y configurar su entorno de desarrollo. Siga los pasos a continuación para asegurarse de que todo esté configurado correctamente:

1. **Descarga del Software:** Visite nuestro <https://github.com/SOFT-cRONOS/RepaReg> para obtener la última versión del software. Asegúrese de seleccionar la versión que corresponda a su sistema operativo.
2. **Instrucciones de Instalación:** Consulte nuestra <https://github.com/SOFT-cRONOS/RepaReg/edit/cambios/README.md> para obtener instrucciones paso a paso sobre cómo configurar el software en su máquina local.
3. **Repositorio de Código:** Para aquellos que están interesados en contribuir al proyecto o simplemente quieren explorar el código,



visite nuestro <https://github.com/SOFT-cRONOS/RepaReg>.

Asegúrese de revisar los requisitos del sistema y las dependencias necesarias antes de la instalación para evitar cualquier problema. Si encuentra dificultades durante el proceso, comuníquese con nuestro soporte técnico.

## Convenciones para los Endpoints de la API

La URL base para enviar todas las solicitudes de API es <http://localhost:5200>. Se requiere HTTPS para todas las solicitudes de API.

La API de RepaReg sigue las convenciones RESTful , y las operaciones se realizan mediante solicitudes `GET` , `POST` , `PUT` , y `DELETE` en recursos de páginas y bases de datos. Los cuerpos de solicitud y respuesta están codificados como JSON.

### Estructura de los Endpoints:

HTTP method	Endpoint	Description
GET	<code>/static/&lt;filename&gt;</code>	Sirve archivos estáticos como imágenes o CSS.
GET	<code>/tablero</code>	Obtiene datos para el tablero de control.
GET	<code>/productos</code>	Lista todos los productos.
GET	<code>/productos/&lt;id_producto&gt;</code>	Obtiene detalles de un producto específico por ID.
POST	<code>/productos</code>	Crea un nuevo producto.
PUT	<code>/productos/&lt;id_producto&gt;</code>	Modifica un producto existente por ID.
DELETE	<code>/productos/&lt;id_producto&gt;</code>	Elimina un producto por ID.

GET	<b>/servicios</b>	Lista todos los servicios.
GET	/servicios/<id_servicio>	Obtiene detalles de un servicio específico por ID.
POST	/servicios	Crea un nuevo servicio.
PUT	/servicios/<id_servicio>	Modifica un servicio existente por ID.
DELETE	/servicios/<id_servicio>	Elimina un servicio por ID.
GET	/clientes	Lista todos los clientes.
GET	/clientes/<id_cliente>	Obtiene detalles de un cliente específico por ID.
POST	/clientes	Crea un nuevo cliente.
PUT	/clientes/<id_cliente>	Modifica un cliente existente por ID.
DELETE	/clientes/<id_cliente>	Elimina un cliente por ID.
POST	/usuarios/login	Autentica a un usuario.
GET	/usuarios	Obtiene datos de usuario.
GET	/ventas	Lista todas las ventas.
GET	/ventas/<id_venta>	Obtiene detalles de una venta específica por ID.
POST	/ventas	Crea una nueva venta.

DELETE	/ventas/<id_venta>	Elimina una venta por ID.
GET	/reportes/stock	Obtiene un reporte de stock.
GET	/reportes/total_compras_por_cliente	Obtiene reporte de compras totales por cliente.
GET	/reportes/reporte_total_ventas_x_producto	Obtiene reporte de ventas totales por producto.
GET	/reportes/reporte_total_ventas_x_servicio	Obtiene reporte de ventas totales por servicio.
GET	/reportes/<idreport>	Obtiene un reporte específico por su ID.

## Status Codes

Los códigos de respuesta HTTP se utilizan para indicar clases generales de éxito y error.

## Success Code

HTTP Status Quote	Description
200	Successfully processed request.
	Cliente actualizado exitosamente
	Producto eliminado exitosamente
	Servicio creado exitosamente
	Servicio actualizado exitosamente
	Servicio eliminado exitosamente
201	Cliente creado exitosamente
	Producto creado exitosamente
	Producto actualizado exitosamente

HTTP Status Quote	Description
	Venta creada exitosamente

## Error Codes

Las respuestas de error contienen más detalles sobre el error en el cuerpo de la respuesta, en las propiedades "código" y "mensaje".

HTTP Status Quote	code	message
400	Not Found	This request URL is not valid.
	Not Found	Cliente no encontrado
401	Unauthorized	The bearer token is not valid.
403	<b>FORBIDDEN</b>	Usuario o contraseña incorrectos
	<b>FORBIDDEN</b>	No tiene permiso para eliminar el cliente
	<b>FORBIDDEN</b>	El usuario no ha iniciado sesion
	<b>FORBIDDEN</b>	No tiene permiso para eliminar el producto
	<b>FORBIDDEN</b>	No tiene permiso para modificar el producto
	<b>FORBIDDEN</b>	No tiene permiso para modificar el servicio
	<b>FORBIDDEN</b>	No tiene permiso para eliminar el servicio

## Configuración y Uso de Endpoints con Token de Autenticación en Thunder Client

### Descripción:

Esta sección proporciona una guía detallada para configurar y utilizar endpoints de API mediante Thunder Client, con un enfoque específico en la autenticación mediante `authToken`. Aprenderás a configurar variables de entorno para almacenar tu token de forma segura, y a incorporar estas variables en tus solicitudes para realizar operaciones de prueba

eficaces. La guía cubre los pasos para realizar solicitudes POST y GET, así como consejos para la depuración y la interpretación de las respuestas de la API, asegurando que puedas manejar tanto la creación de recursos como la recuperación de datos de manera exitosa.

### 1. Configuración de Variables de Entorno:

- Abrir Thunder Client dentro de tu editor de código o plataforma de API.
- Ir a la sección de Variables de Entorno ( `Env` ).
- Crea una nueva variable de entorno, por ejemplo, `repareg_env` .
- Añade una variable, como `authToken` , y asigna tu token de autenticación como su valor.

### 2. Uso de la Variable de Entorno en las Solicitudes:

- Al Configurar una nueva solicitud, en lugar de agregar un encabezado de autorización, incluir el token en la URL o como un parámetro de consulta, dependiendo de cómo tu API espere recibir el token. Por ejemplo, tu URL podría ser algo como `/miRecurso?authToken={{authToken}}` , donde `{{authToken}}` es el nombre de la variable de entorno que has configurado previamente.

### 3. Realizar Solicitudes de Prueba:

- Para una solicitud POST (para crear un nuevo recurso), añadir un cuerpo de solicitud ( `body` ) en formato JSON con los datos a enviar.
- Para una solicitud GET (para obtener datos), asegúrese de que la URL está correctamente formada y que cualquier parámetro requerido está incluido.
- Asegúrate de que el método de solicitud es el correcto (GET para obtener datos, POST para enviar datos, PUT para actualizar, etc.).
- Enviar la solicitud y revisar la respuesta. Los códigos de estado HTTP te informarán si la solicitud fue exitosa (200 OK, 201 Created) o si hubo algún error (400 Bad Request, 403 Forbidden, etc.).




#### 4. Revisión y Depuración:









- Si recibes un error, revisa la configuración de tu solicitud y las variables de entorno para asegurarte de que todo esté correcto.
- Utilizar la consola o los registros de la herramienta para obtener más detalles sobre el error y cómo solucionarlo.

En caso de encontrar errores al realizar solicitudes a la API, es importante revisar y comparar la solicitud hecha con la especificación de los endpoints provistos en la documentación. A continuación, se detallan los formatos de entrada, los tipos de solicitud y la descripción de lo que cada endpoint realiza, lo que facilitará la identificación y corrección de posibles discrepancias o faltas en la solicitud enviada. Asimismo, se deben verificar las credenciales y parámetros utilizados, asegurándose de que coincidan con los requeridos por cada endpoint específico y su correspondiente descripción.

## Endpoints

### Endpoints

Aa Propiedad	Type	URL	Description
 <u>user login</u>	POST	<u>/usuarios/login</u>	Creates an access token for API authentication.
 <u>client</u>	GET	<u>/clientes?authToken={{token}}</u>	Lista todos los clientes.
 <u>client</u>	POST	<u>/clientes?authToken={{token}}</u>	Dar de alta un nuevo cliente
 <u>client by id</u>	GET	<u>/clientes/&lt;id_cliente&gt;?authToken={{token}}</u>	obtener cliente a traves de su id.
 <u>del_cliente</u>	DELETE	<u>/clientes/&lt;id_cliente&gt;?authToken={{token}}</u>	eliminar cliente por su id.
 <u>client by id</u>	PUT	<u>/clientes/&lt;id_cliente&gt;?authToken={{token}}</u>	
 <u>productos</u>	GET	<u>/productos?authToken={{token}}</u>	listado de todos los productos

Aa Propiedad	Type	URL	Description
 <u>product_by_id.</u>	GET	<u>/productos/&lt;id_producto&gt;?authToken={{token}}</u>	obtener producto por su id
 <u>product</u>	POST	<u>/productos?authToken={{token}}</u>	crear un nuevo producto
 <u>producto</u>	DELETE	<u>/productos/&lt;id_producto&gt;?authToken={{token}}</u>	eliminar producto por su id
 <u>product</u>	PUT	<u>/productos/&lt;id_producto&gt;?authToken={{token}}</u>	modificar producto por su id
 <u>user</u>	GET	<u>/usuarios?authToken={{token}}</u>	obtener datos de usuario logueado.
 <u>servicios</u>	GET	<u>/servicios?authToken={{token}}</u>	obtener lista de todos los servicios
 <u>serv_by_id.</u>	GET	<u>/servicios/&lt;id_servicio&gt;?authToken={{token}}</u>	
 <u>servicios</u>	POST	<u>/servicios?authToken={{token}}</u>	agregar nuevo servicio
Sin título			
 <u>service_by_id</u>	PUT	<u>/servicios/&lt;id_servicio&gt;?authToken={{token}}</u>	modificar servicio por su id
 <u>servi_by_id</u>	DELETE	<u>/servicios/&lt;id_servicio&gt;?authToken={{token}}</u>	eliminar servicio
Sin título			
 <u>ventas</u>	GET	<u>/ventas?authToken={{token}}</u>	obtener lista ventas
 <u>ventas_by_id.</u>	GET	<u>/ventas/&lt;id_venta&gt;?authToken={{token}}</u>	obtener venta por su id
 <u>ventas</u>	POST	<u>/ventas?authToken={{token}}</u>	crear una venta
 <u>ventas_id</u>	DELETE	<u>/ventas/&lt;id_venta&gt;?authToken={{token}}</u>	eliminar venta



# Proximas Mejoras a Implementar en Proyectos Futuros

## 1. Implementación de CRUD Completo para Usuarios:

- Ampliar la API para incluir operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para los usuarios, permitiendo una gestión integral a través de la API en lugar de depender directamente de la base de datos.

## 2. Rutas Específicas para la Gestión de Usuarios:

- Desarrollar y documentar rutas específicas para la manipulación de datos de usuarios, asegurando que cada acción tenga su endpoint correspondiente y que estos cumplan con los principios RESTful.

## 3. Login de Clientes:

- Brinda a los clientes la comodidad de acceder a sus historiales de compra y el estado de productos en stock a través de un portal personalizado.

## 4. Interfaz de Usuario Intuitiva:

- Implementar indicadores de progreso, como una animación de "cargando", para mejorar la retroalimentación visual mientras los datos se están recuperando o procesando, mejorando la experiencia de usuario al interactuar con la página.

## 5. Persistencia de Datos de Ventas:

- Modificar el flujo de trabajo de eliminación de ventas para que, en lugar de borrar los datos, se marquen como inactivos o se trasladen a una tabla de archivo. Esto permitiría mantener un registro histórico de todas las ventas para referencia futura o auditorías.

## 6. Alertas de Stock Mínimo:

- Recibir alertas automáticas sobre niveles mínimos de stock, permitiendo tomar decisiones informadas y oportunas para

reabastecer productos.

#### **7. Documentación Detallada:**

- Mejorar la documentación de la API proporcionando ejemplos de solicitudes y respuestas, y descripciones claras de cada endpoint para facilitar la integración y el uso por parte de los desarrolladores.

#### **8. Autenticación y Autorización Robustas:**

- Integrar un sistema de autenticación y autorización más robusto que permita un control más granular del acceso a la API, mejorando la seguridad y la gestión de usuarios.

#### **9. Exportación de Datos de Facturación:**

- Desarrollar una funcionalidad que permita a los usuarios exportar datos de facturación en diferentes formatos, como CSV, PDF o Excel. Esto facilitaría la contabilidad y la gestión financiera, permitiendo a los usuarios manipular y analizar los datos fuera de la plataforma según sea necesario.

#### **8. División y Estados de Pago:**

- Simplificar transacciones comerciales al ofrecer flexibilidad en los métodos y plazos de pago, adaptándose a las necesidades de los clientes y llevar un control de pagos pendientes.