



Universidad Politécnica de Pachuca

Ingeniería en Software
9° Cuatrimestre

Punto de venta para papelería

Rello Yañez Felipe de Jesus
De La Cruz Valdez Angel Marco Polo
Cruz Moreno Fernando Jesús
Zuriaga Martinez Christian Arturo
Benitez Hernandez Angela

Arquitectura Orientada a Servicios
Jazmin Rodriguez Flores
25 de junio del 2023

Índice

Metodología	3
Introducción	4
Herramientas utilizadas	5
Gráfica de quemado	6
Tablero de GitHub	7
Historias de usuario	7
Modelo de datos	11
Listado de datos y servicios	12
Diagrama de paquetes	13
Backend	13
Frontend	13
Patrón de diseño	14
Patrón de diseño de enrutamiento (Router)	14
Pruebas unitarias	15
Pruebas Unitarias	15
Patrón Arquitectónico	15
Principio de Diseño	17
Principio abierto/cerrado	17
Servicio Proveedores (Fernando)	18
Servicio Productos (Angela)	30
Clientes mediante prototipo falso:	43
Servicio web funcionales:	45
Servicio Ventas (Christian)	52
Time recording log:	57
Autoevaluación	65

Metodología

Elegimos utilizar Scrum como metodología de desarrollo ya que promueve la colaboración y el aprendizaje a través de la experiencia. Dada la restricción de tiempo para completar el desarrollo de nuestro software, Scrum se ajusta a nuestras necesidades. Sus características nos permiten distribuir las responsabilidades entre los miembros del equipo, asignando tareas de acuerdo a sus habilidades y conocimientos particulares. Esto optimiza el uso de sus capacidades individuales y acelera la finalización de las actividades.

Se harán reuniones diarias propuestas por el Scrum porque brindan un espacio para abordar dudas y superar obstáculos. Estos encuentros no solo resuelven problemas, sino que también fomentan la adquisición de nuevas habilidades y conocimientos en áreas desconocidas.

El rol crucial del Scrum Master radica en la asignación y supervisión de tareas, tomando la responsabilidad global del equipo y asegurando la correcta orientación del proyecto.

Introducción

En un mundo cada vez más dinámico y tecnológico, la eficiencia en los procesos comerciales se vuelve esencial para el éxito de cualquier negocio. En este contexto, presentamos nuestro proyecto: el Sistema de Punto de Venta para Papelería. Este sistema ha sido concebido con el objetivo principal de agilizar y optimizar tanto los procesos de venta como los registros y el control interno en el entorno de una papelería.

La gestión de una papelería conlleva una serie de desafíos en cuanto a la administración eficiente de productos, ventas, interacción con proveedores y atención a clientes. Los procesos manuales y los sistemas fragmentados pueden dar lugar a errores, pérdida de tiempo y oportunidades, así como a dificultades en el mantenimiento de un registro preciso y completo de las transacciones. Es en este contexto que surge la necesidad de un enfoque más integral y tecnológico.

El Sistema de Punto de Venta para Papelería representa una solución completa y unificada para enfrentar estos desafíos. No solo facilita el proceso de ventas, sino que también brinda herramientas avanzadas para el registro y el control de las mismas. Además, el sistema abarca la gestión de proveedores y productos, garantizando un flujo constante de suministros y una amplia gama de opciones para los clientes.

Una característica fundamental de este proyecto es su capacidad para ofrecer una experiencia de usuario intuitiva y accesible. Con una interfaz amigable y funcionalidades diseñadas para optimizar la interacción con el sistema, tanto los empleados de la papelería como los clientes se beneficiarán de una plataforma que simplifica los procesos y mejora la calidad del servicio.

En resumen, el Sistema de Punto de Venta para Papelería es una respuesta a la necesidad imperante de modernización en el ámbito comercial. A través de su implementación, se espera lograr una mayor eficiencia en la gestión de ventas, un control más preciso de inventario, un seguimiento efectivo de proveedores y una mejora general en la experiencia tanto de los clientes como de los empleados. Este proyecto surge como una propuesta en la búsqueda constante de la excelencia operativa y la satisfacción del cliente en el sector de las papelerías.

Herramientas utilizadas

Herramientas utilizadas en el proyecto:

Frontend:

- **HTML:** Utilizamos HTML para estructurar y definir el contenido de nuestras páginas web.
- **CSS:** Empleamos CSS para dar estilo y diseño a nuestras páginas, logrando una presentación visual atractiva y coherente.
- **JavaScript:** Integrando JavaScript, dotamos a nuestras páginas de interactividad y dinamismo, mejorando la experiencia del usuario.

Backend:

- **PHP:** Nuestro backend está desarrollado en PHP, un lenguaje de programación ampliamente utilizado para la creación de aplicaciones web dinámicas y funcionales.

Base de Datos:

- **MySQL:** Utilizamos MySQL como sistema de gestión de bases de datos, permitiéndonos almacenar, organizar y recuperar datos de manera eficiente y segura.

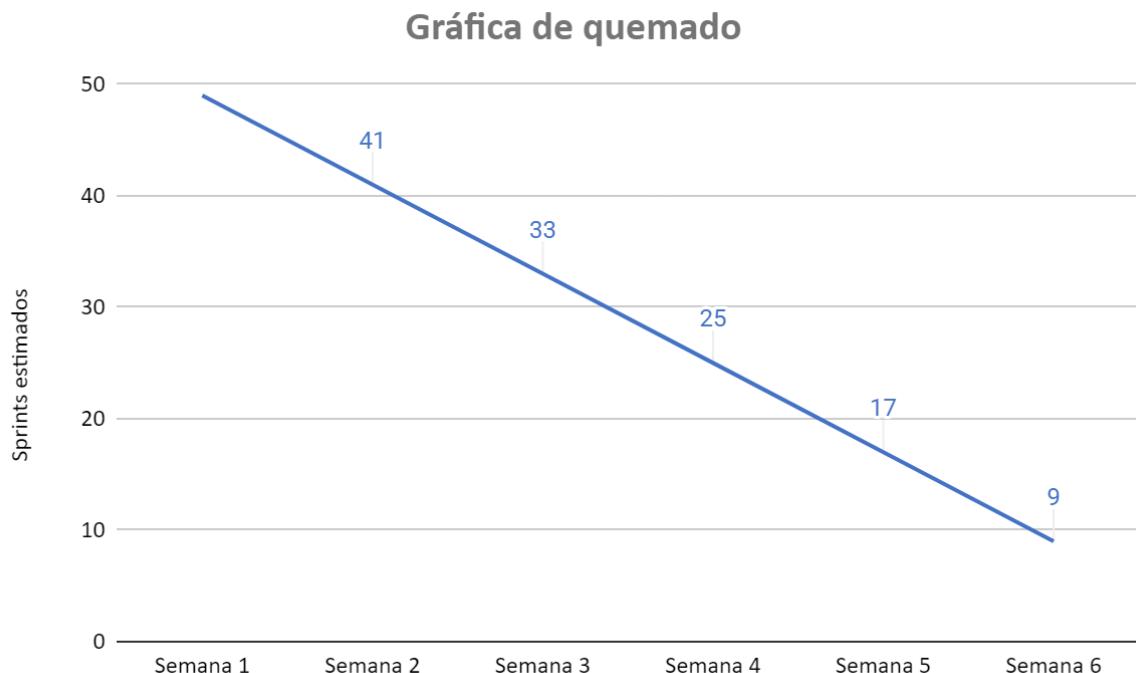
Servidor:

- **Apache:** Nuestro proyecto está alojado en un servidor Apache, que actúa como servidor web para entregar nuestras páginas a los usuarios finales de manera rápida y confiable.

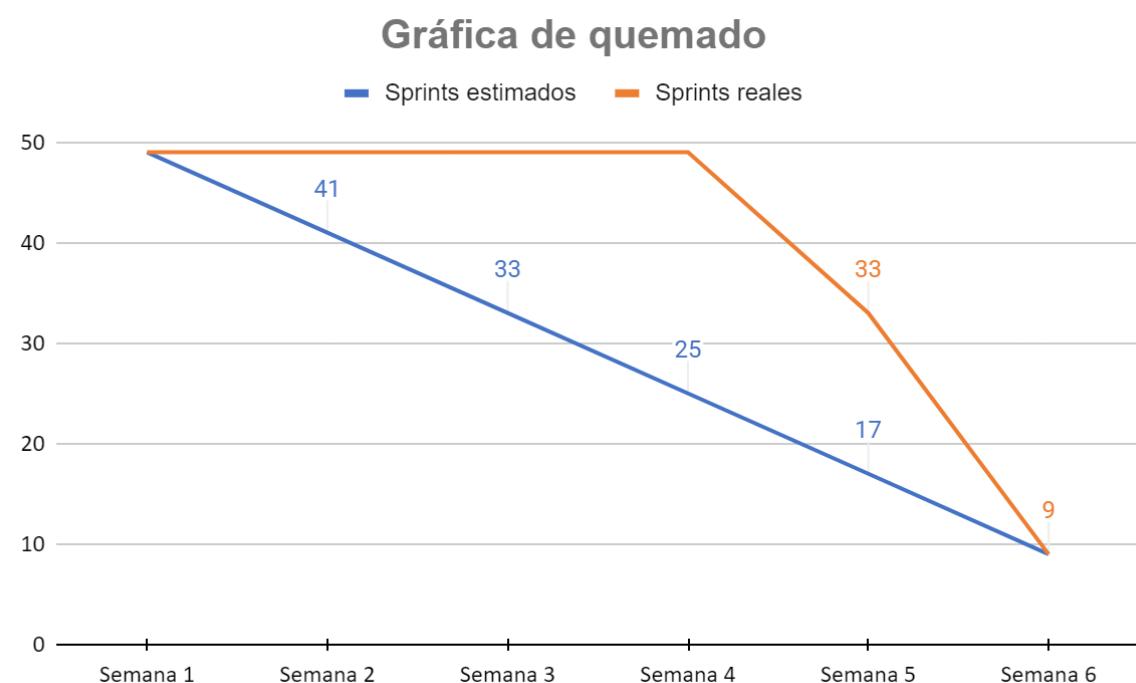
Esta combinación de herramientas nos permite desarrollar una aplicación web completa y eficaz, desde la estructura y el diseño hasta la funcionalidad y la gestión de datos.

Gráfica de quemado

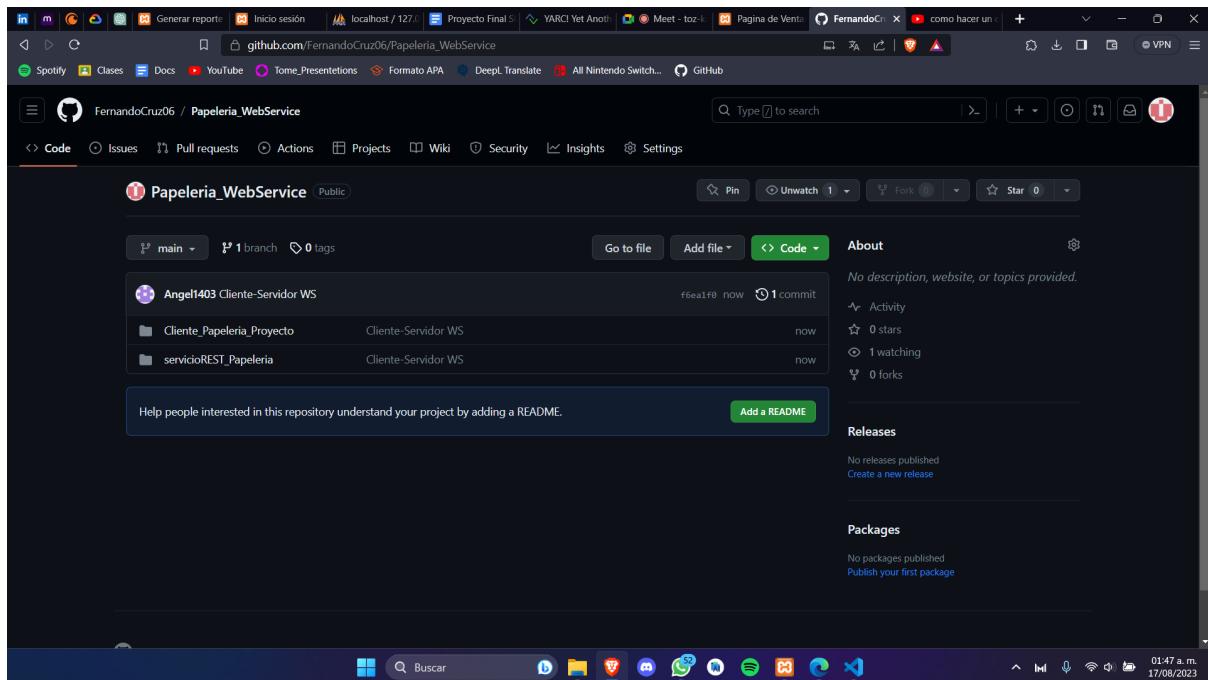
Gráfica estimada:



Gráfica real resultante:



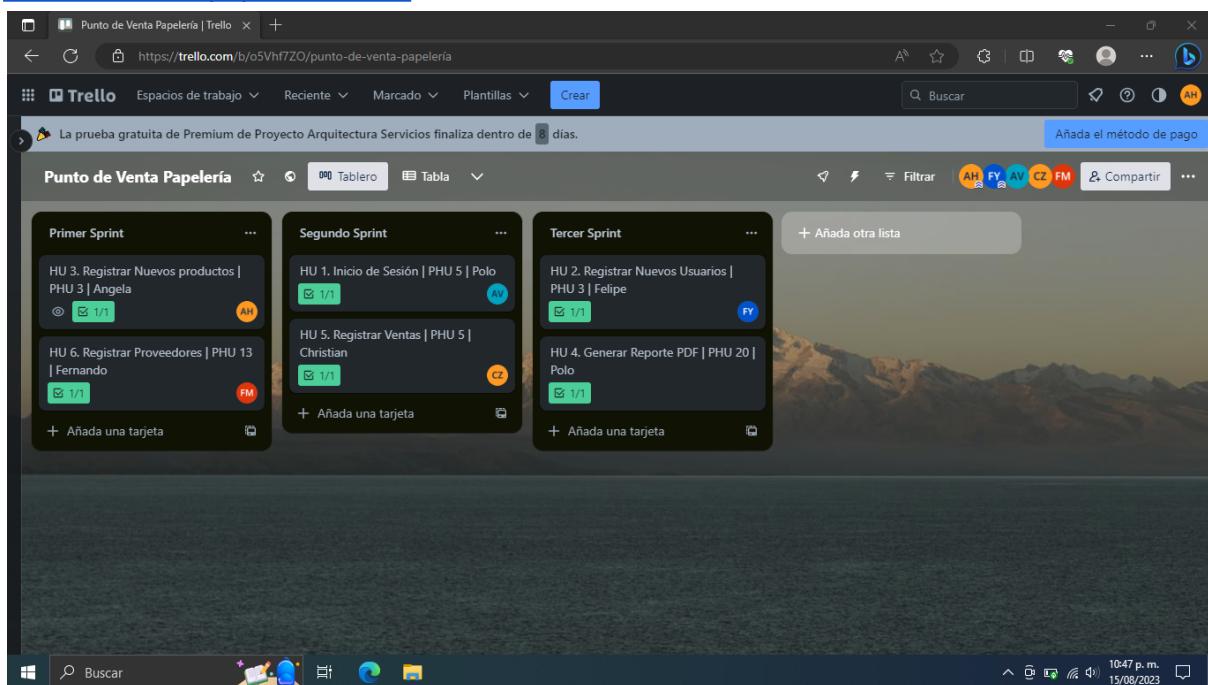
Tablero de GitHub



Historias de usuario

Enlace a tablero Trello:

[Punto de venta papelería Trello](https://trello.com/b/o5Vh7ZO/punto-de-venta-papeleria)



Historia de Usuario #	1	
Título	Inicio de Sesión.	
Descripción	Como usuario de la página web quiero acceder a la página para ingresar al sitio.	
Criterios de aceptación	<ol style="list-style-type: none"> 1. La página de inicio debe mostrar de manera prominente la opción "Iniciar sesión" en la página principal. 2. Al seleccionar la opción "Iniciar sesión", el usuario debe ser redirigido a una página de inicio de sesión dedicada. 3. La página de inicio de sesión debe proporcionar campos claros y visibles para que el usuario ingrese su nombre de usuario y contraseña. 4. Al proporcionar credenciales válidas y al hacer clic en el botón "Iniciar sesión", el usuario debe ser autenticado exitosamente y dirigido a la página principal del sitio web. 5. En caso de ingresar credenciales incorrectas y hacer clic en el botón "Iniciar sesión", el sistema debe mostrar un mensaje de error claro indicando la naturaleza del error. 6. En caso de tener campos vacíos se deberá mostrar una advertencia 7. Al escribir el correo este deberá tener un formato válido, de lo contrario se mostrará una advertencia 	
Encargado	Angel Marco Polo De La Cruz Valdez	
Puntos de historia		5

Historia de Usuario #	2	
Título	Registrar Usuarios.	
Descripción	Como dueño de la página web quiero registrar nuevos usuarios para garantizar el acceso a la página web y hacer uso de sus servicios	
Criterios de aceptación	<ol style="list-style-type: none"> 1. Debe existir un formulario de registro accesible desde la página principal. 2. El formulario debe requerir campos como correo electrónico, contraseña y rol. 3. Al enviar el formulario, se debe validar la información y crear una nueva cuenta de usuario si los datos son válidos. 4. El sistema debe mostrar una alerta de confirmación de registro. 5. Los datos de usuario registrados deben almacenarse de forma segura en la base de datos. 6. El sistema debe contener 1 botón para eliminar datos y otro para modificar. 7. Al modificar los datos el sistema debe mostrar un aviso de datos modificados. 	
Encargado	Felipe Rello Yañez	
Puntos de historia		3

Historia de Usuario #	3	
Título	Registrar Productos.	
Descripción	Como dueño de la página quiero registrar productos para llevar un control del inventario	
Criterios de aceptación	<ol style="list-style-type: none"> 1. Debe haber una sección o interfaz dedicada para registrar nuevos productos. 2. Los campos necesarios para el registro deben incluir nombre del producto, costo, cantidad e id del proveedor. 3. Al registrar un producto, el sistema debe actualizar automáticamente el inventario. 4. Se debe proporcionar la opción de editar o eliminar productos registrados. 5. Los datos de los productos deben almacenarse de manera organizada y accesible. 	
Encargado	Angela Benitez Hernandez	
	Puntos de historia	3

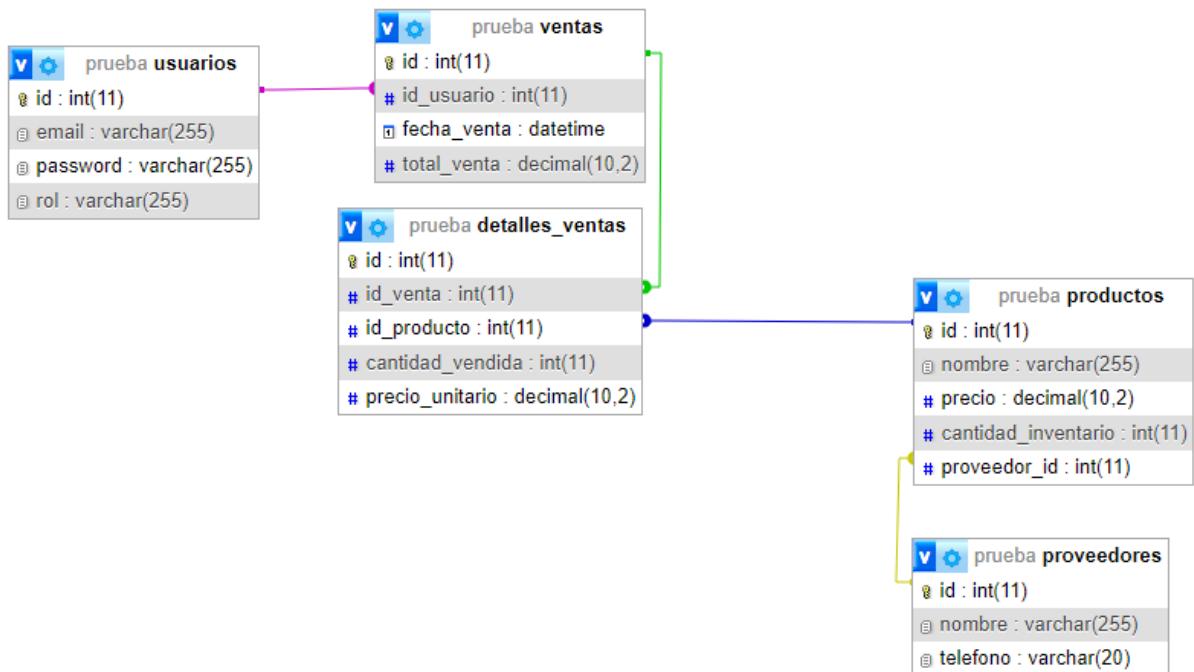
Historia de Usuario #	4	
Título	Generar Reportes PDF.	
Descripción	Como dueño de la página quiero generar un reporte pdf para llevar un control de las ventas en la página	
Criterios de aceptación	<ol style="list-style-type: none"> 1. Debe existir una función para generar un reporte PDF de ventas. 2. El reporte debe contener detalles de las ventas realizadas incluyendo fechas y montos. 3. Se deben proporcionar opciones para filtrar el número de venta. 4. El reporte PDF debe poder descargarse e imprimirse desde la página web. 5. El diseño y formato del reporte deben ser legibles. 	
Encargado	Angel Marco Polo De La Cruz Valdez	
	Puntos de historia	20

Historia de Usuario #	5	
Título	Registrar Ventas	
Descripción	Como usuario normal quiero realizar ventas de productos para llevar un control de los ingresos de las ventas realizadas	
Criterios de aceptación	<ol style="list-style-type: none"> 1. Funcionalidad de Ventas: La funcionalidad de registro de ventas debe permitir al usuario normal ingresar los detalles de cada venta, como productos vendidos, cantidad, precio unitario y fecha de la venta. 2. Interfaz Intuitiva: La interfaz de registro de ventas debe ser intuitiva y fácil de usar, con campos claros y opciones descriptivas para que el usuario pueda ingresar la información de manera precisa. 3. Registro Completo: El sistema debe permitir al usuario registrar todas las ventas realizadas, sin limitaciones en la cantidad de registros. 	
Encargado	Zuriaga Martínez Christian Arturo	Puntos de historia
		5

Historia de Usuario #	6	
Título	Registrar Proveedores	
Descripción	Como dueño de la página, quiero tener un control de proveedores para tener un control de los proveedores para llevar un seguimiento óptimo sobre las compras que se realizan a los proveedores y el inventario.	
Criterios de aceptación	<ol style="list-style-type: none"> 1. Gestión de Proveedores: Debe existir una funcionalidad que permita al dueño de la página registrar y gestionar información detallada de los proveedores, como nombre de la empresa, información de contacto y dirección. 2. Registro de Compras: El sistema debe permitir al dueño registrar las compras realizadas a los proveedores, indicando los productos adquiridos, cantidades, precios unitarios y fechas de compra. 3. Seguimiento de Compras: El sistema debe mantener un historial completo de todas las compras realizadas a los proveedores, incluyendo los detalles de los productos comprados, fechas y montos totales. 4. Relación con Inventario: Las compras registradas deben actualizarse automáticamente en el inventario de productos, aumentando las existencias disponibles. 	
Encargado	Fernando Cruz Moreno	Puntos de historia
		13

Total de puntos: 49/6=8

Modelo de datos



La representación del modelo de datos propuesto para nuestro sistema de punto de venta representa cómo se organizará y almacenará la información relacionada con la papelería en una base de datos. Este modelo se conforma por diferentes tablas interconectadas que cubren y abarcan los diferentes aspectos del funcionamiento de la papelería, desde la administración de usuarios hasta el seguimiento de las ventas y los productos disponibles.

La tabla de usuarios registra la información sobre las personas que interactúan con el sistema. Los usuarios pueden iniciar sesión en el sistema para realizar tareas específicas y acceder a la información relevante. La tabla de proveedores almacena la información sobre las fuentes de suministros de los productos de la papelería y se relaciona con la tabla de los productos donde también se observa la información sobre el nombre del producto, su descripción, el precio unitario entre otros. Esta tabla permite un seguimiento preciso del inventario y ayuda a mantener actualizada la información sobre los productos disponibles para la venta.

La tabla de ventas registra las transacciones realizadas en la papelería. Cada registro incluye un ID único de venta, la ID del usuario que realizó la venta, la fecha y el total de la venta. Esta información permite un seguimiento cronológico de las ventas y facilita el cálculo de ingresos. Además de que se relaciona con la tabla de detalles de ventas en los cuales se muestran los detalles específicos de cada venta como el ID de la venta relacionada, la ID del producto vendido, la cantidad vendida, el precio unitario al momento de la venta y el subtotal calculado.

Esta estructura de datos permite un desglose detallado de cada producto, venta, o proveedor lo que ayuda en el análisis y la generación de informes sobre el inventario y organización de la papelería para llevar un control de mejor calidad.

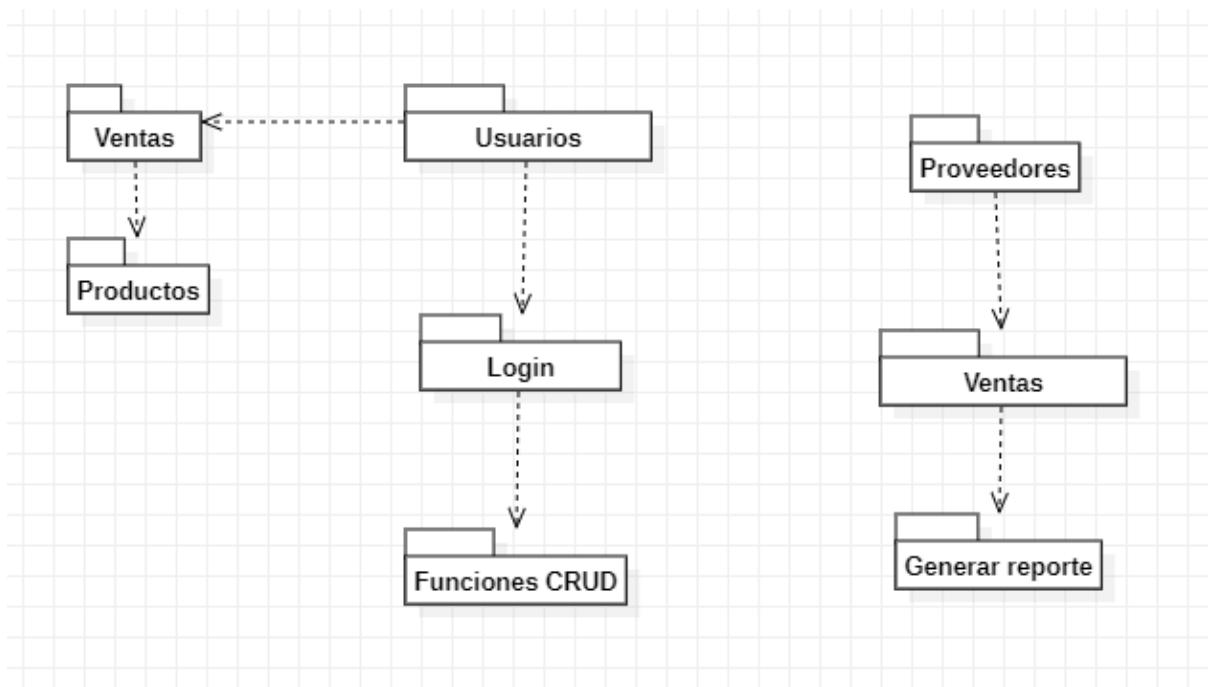
Listado de datos y servicios

Entidades	Funcionalidad	Métodos
Acceder a la página web:	Iniciar Sesión	GET
Registrar usuario	Obtener todos los usuarios	GET
	Crear nuevo usuario	POST
	Editar usuario	PUT
	Eliminar usuario	DELETE
Registro articulos	Obtener todos los articulos	GET
	Crear nuevo producto	POST
	Editar producto	PUT
	Eliminar producto	DELETE
Ventas	Obtener todas las ventas	GET
	Crear nueva venta	POST
	Eliminar venta	DELETE
Proveedores	Obtener todos los proveedores	GET
	Crear nuevo proveedor	POST
	Editar proveedor	PUT
	Eliminar proveedor	DELETE
Reportes	Crear nuevo reporte	POST

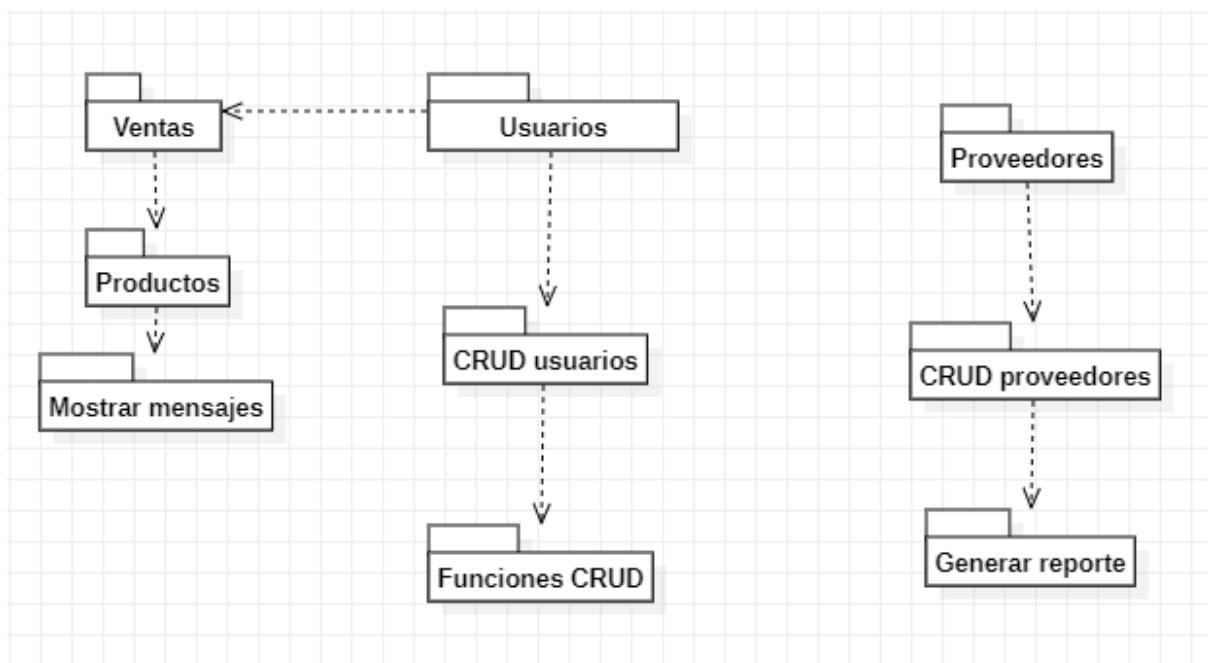
Diagrama de paquetes

El siguiente diagrama de paquetes muestra la relación que existe entre las diferentes clases.

Backend



Frontend



Patrón de diseño

Patrón de diseño de enrutamiento (Router)

En el proyecto de desarrollo de la aplicación web punto de venta para una papelería, se ha empleado el patrón de diseño de enrutamiento de manera eficaz. El patrón de enrutamiento ha sido utilizado para crear una estructura de navegación coherente y funcional que permite a los usuarios acceder de manera intuitiva a las distintas funcionalidades de la aplicación.

En este contexto, el patrón de enrutamiento se ha implementado para gestionar las diferentes secciones de la aplicación de manera organizada. Por ejemplo, cuando un usuario accede a la aplicación, el enrutador dirige al usuario a la página principal del inventario, donde puede ver y gestionar los productos disponibles en la papelería. Además, cuando el usuario desea registrar una venta, el patrón de enrutamiento garantiza que sea dirigido a la sección correspondiente para ingresar los detalles de la venta.

La aplicación también hace uso del enrutamiento para permitir la generación de reportes PDF. Cuando un usuario solicita un informe específico, el enrutador se encarga de dirigirlo a la página adecuada, donde se generará el reporte en formato PDF. Esta característica demuestra cómo el patrón de enrutamiento se ha implementado para manejar diferentes acciones y procesos dentro de la aplicación.

En resumen, el patrón de diseño de enrutamiento se ha utilizado en este proyecto para crear una estructura de navegación lógica y coherente que facilita la interacción del usuario con las diversas funcionalidades de la aplicación, como la gestión de inventario, el registro de ventas y la generación de informes PDF.

Pruebas unitarias

 Pruebas Unitarias

Patrón Arquitectónico

Patrón Arquitectónico: Una sola instancia de la aplicación tenant-aware

La elección de una sola instancia de la aplicación tenant-aware es la mejor opción para el proyecto de punto de venta de una papelería por las siguientes razones:

1. *Gestión eficiente de múltiples sucursales:* Si la papelería tiene varias sucursales, una sola instancia tenant-aware permitirá gestionar todas ellas de manera centralizada. Cada sucursal puede ser tratada como un inquilino independiente, lo que facilita la administración y el seguimiento de las ventas, inventario y otros datos de forma individualizada.
2. Menor costo y complejidad de infraestructura: Como la papelería es un negocio con un solo tipo de usuario y no se necesitan personalizaciones altamente complejas para cada sucursal, tener una sola instancia de la aplicación reduce los costos de infraestructura y simplifica el mantenimiento y la administración.
3. *Facilidad de actualización y mejoras:* Al tener una única base de código compartida por todas las sucursales, las actualizaciones y mejoras se aplican de manera consistente a todos los puntos de venta. Esto garantiza que cada sucursal cuente con las mismas características y mejoras, lo que conduce a una experiencia uniforme para los usuarios.
4. *Coherencia en la experiencia del usuario:* Al compartir la misma instancia de la aplicación, todos los empleados de las diferentes sucursales trabajarán con la misma interfaz y flujos de trabajo, lo que facilita la formación y reduce la curva de aprendizaje para el personal.
5. *Escalabilidad para el crecimiento futuro:* A medida que la papelería crezca y haya más sucursales, la arquitectura tenant-aware se adaptará fácilmente a la adición de nuevos inquilinos, evitando la necesidad de implementar una nueva instancia para cada sucursal y manteniendo una gestión centralizada.
6. *Seguridad y aislamiento de datos:* Al ser una sola instancia tenant-aware, la seguridad y el aislamiento de datos entre las sucursales pueden ser gestionados

cuidadosamente para garantizar que la información de una sucursal no sea accesible para otras.

7. *Flexibilidad para futuras expansiones*: Si la papelería decide diversificarse o expandir sus operaciones a otro tipo de negocio relacionado en el futuro, una arquitectura tenant-aware facilitará la incorporación de nuevos inquilinos para cada unidad de negocio.

En conclusión, para un punto de venta de una papelería con múltiples sucursales y un solo tipo de usuario, la arquitectura de una sola instancia tenant-aware proporciona una solución rentable, escalable y de fácil administración. Permite unificar y centralizar la gestión, manteniendo al mismo tiempo la coherencia en la experiencia del usuario y la seguridad de los datos entre las diferentes sucursales. Esta opción brinda una base sólida para el crecimiento futuro y es adecuada para satisfacer las necesidades del proyecto de punto de venta de la papelería de manera eficiente y eficaz.

Principio de Diseño

Principio abierto/cerrado

Es uno de los principios fundamentales de diseño de software propuesto por Bertrand Meyer. Que establece lo siguiente: "Una entidad de software clase, módulo o función debe estar abierta para su extensión pero cerrada para su modificación."

Este principio se aplicará para fomentar un diseño flexible y escalable del sistema.

1. Módulo de Productos:

- Cerrado: El módulo de productos debe estar cerrado para modificaciones. Una vez que está diseñado y funcionando correctamente, no se deben realizar cambios que alteren su comportamiento actual.

- Abierto: Debe estar abierto para extensiones, lo que significa que se debe permitir la incorporación de nuevos tipos de productos o características adicionales sin tener que modificar el código existente del módulo de productos. Esto se puede lograr mediante el uso de interfaces o clases abstractas que definen un contrato que otras clases implementen para agregar nuevos tipos de productos.

2. Módulo de Ventas:

- Cerrado: Una vez que el módulo de ventas está desarrollado y funcionando correctamente, debe ser cerrado para modificaciones.

- Abierto: Debe ser abierto para extensiones para permitir la incorporación de nuevas funcionalidades relacionadas con las ventas, como métodos de pago adicionales o promociones especiales.

Gestión de Alquiler de Productos:

Este enfoque será relevante cuando el cliente busque incorporar el alquiler de computadoras en su empresa. En esta instancia, el sistema proporcionará la capacidad de registrar intervalos de tiempo de manera fraccionada, adaptándose a las necesidades específicas.

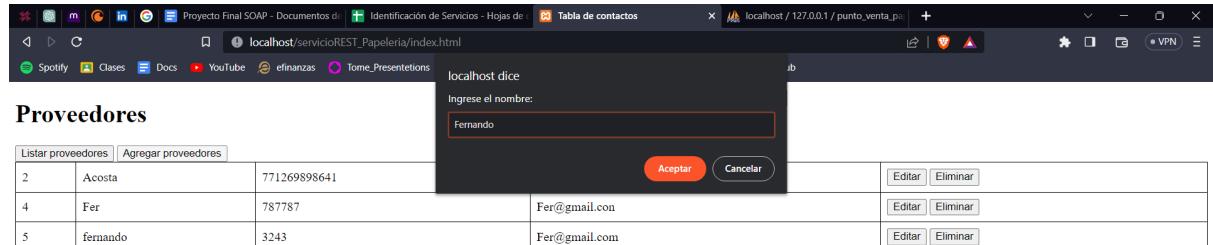
Proceso de Cobro:

Una vez que el administrador haya registrado el alquiler como producto, el sistema procederá a efectuar un cobro preciso al momento de la finalización.

Servicio Proveedores (Fernando)

POST

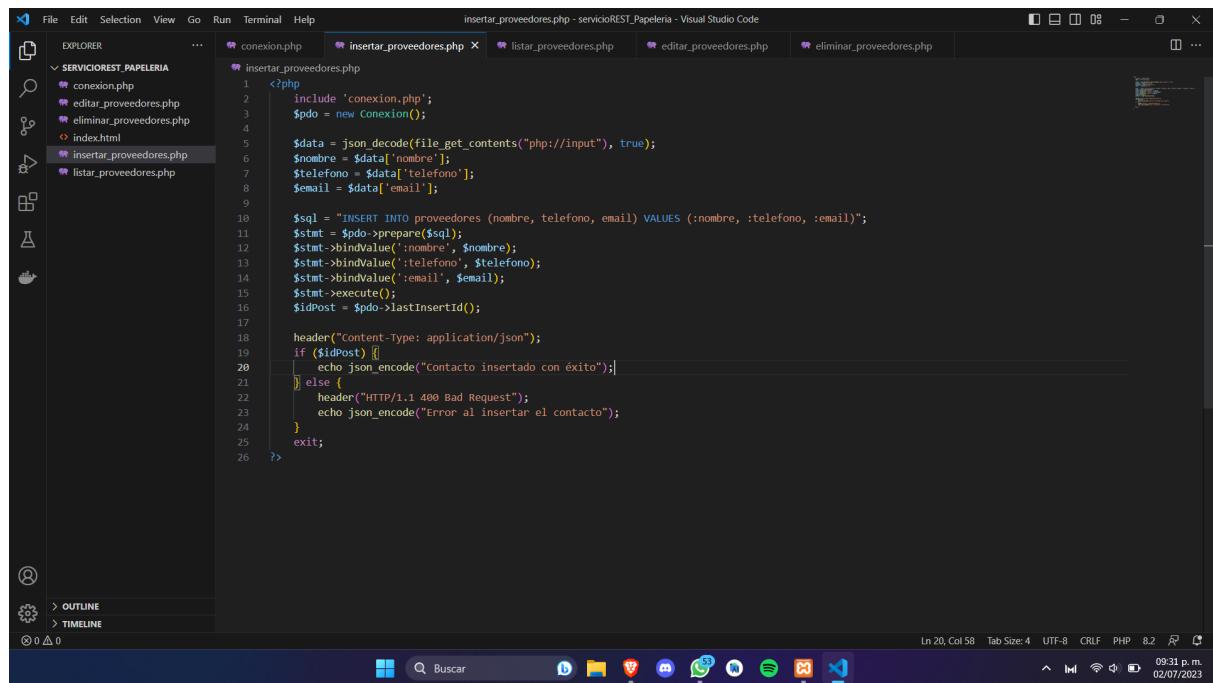
Interfaz



2	Acosta	771269898641		Editar	Eliminar
4	Fer	787787	Fer@gmail.com	Editar	Eliminar
5	fernando	3243	Fer@gmail.com	Editar	Eliminar



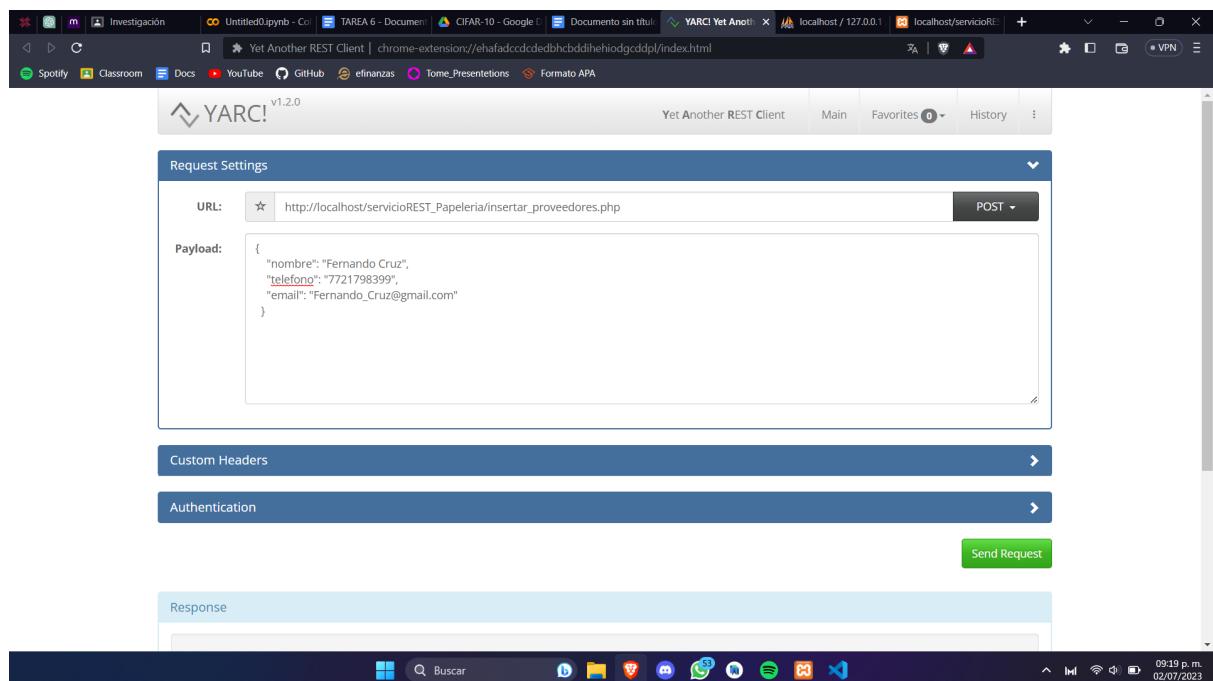
Código



```
File Edit Selection View Go Run Terminal Help insertar_proveedores.php - servicioREST_Papelaria - Visual Studio Code
EXPLORER SERVICIOREST_PAPELERIA
conexion.php insertar_proveedores.php listar_proveedores.php editar_proveedores.php eliminar_proveedores.php
index.html
insertar_proveedores.php
listar_proveedores.php

1 <?php
2     include 'conexion.php';
3     $pdo = new Conexion();
4
5     $data = json_decode(file_get_contents("php://input"), true);
6     $nombre = $data['nombre'];
7     $telefono = $data['telefono'];
8     $email = $data['email'];
9
10    $sql = "INSERT INTO proveedores (nombre, telefono, email) VALUES (:nombre, :telefono, :email)";
11    $stmt = $pdo->prepare($sql);
12    $stmt->bindValue(':nombre', $nombre);
13    $stmt->bindValue(':telefono', $telefono);
14    $stmt->bindValue(':email', $email);
15    $stmt->execute();
16    $idPost = $pdo->lastInsertId();
17
18    header("Content-Type: application/json");
19    if ($idPost) {
20        echo json_encode("Contacto insertado con éxito");
21    } else {
22        header("HTTP/1.1 400 Bad Request");
23        echo json_encode("Error al insertar el contacto");
24    }
25    exit;
26 ?>
```

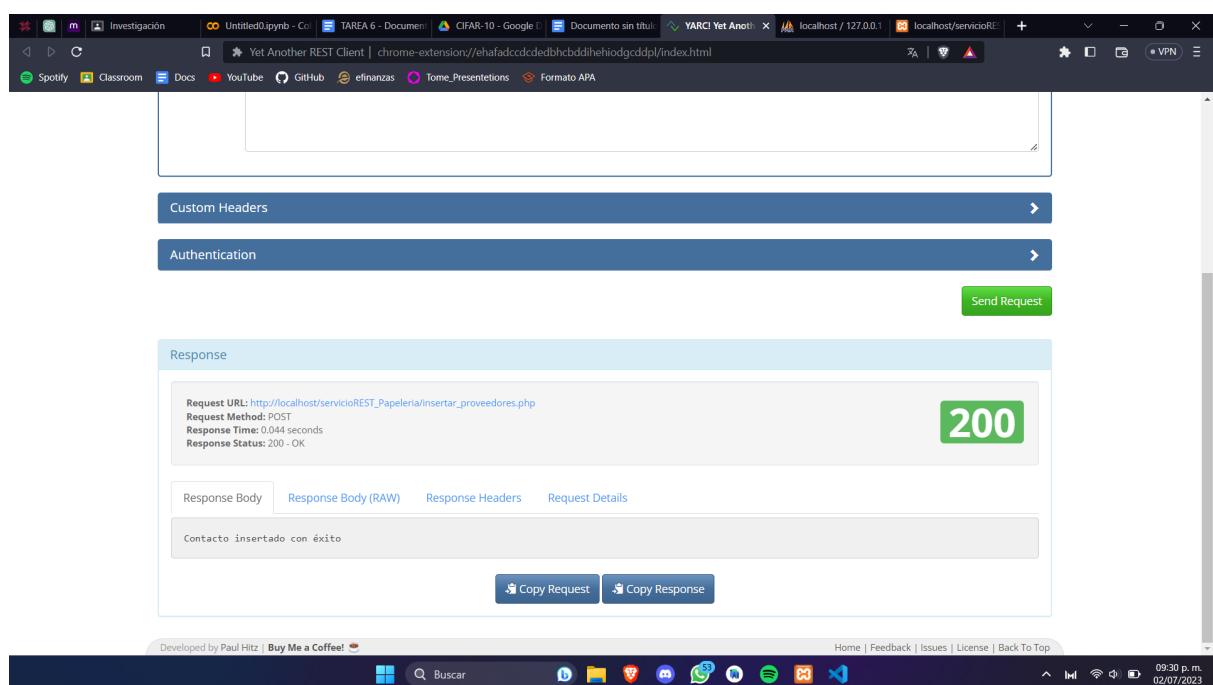
Funcionamiento



The screenshot shows the YARCI REST Client interface. In the 'Request Settings' section, the URL is set to `http://localhost/servicioREST_Papelaria/insertar_proveedores.php` and the method is set to POST. The 'Payload' field contains the following JSON data:

```
{
  "nombre": "Fernando Cruz",
  "telefono": "7721798399",
  "email": "Fernando_Cruz@gmail.com"
}
```

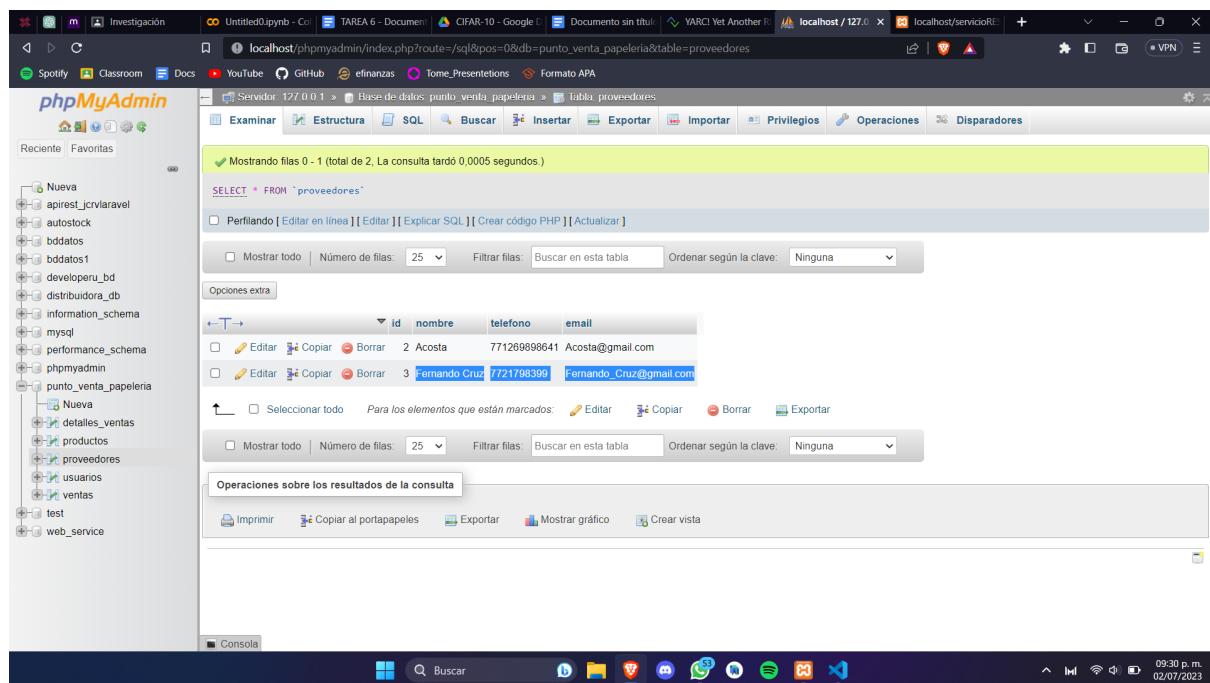
Below the request settings, there are sections for 'Custom Headers' and 'Authentication'. A green 'Send Request' button is located to the right of the payload field. The 'Response' section is currently empty. The browser's status bar at the bottom shows the date and time as 02/07/2023 at 09:19 p.m.



The screenshot shows the YARCI REST Client interface after a successful request. The 'Response' section displays the following details:

- Request URL: `http://localhost/servicioREST_Papelaria/insertar_proveedores.php`
- Request Method: POST
- Response Time: 0.044 seconds
- Response Status: 200 - OK

A large green '200' is prominently displayed. Below the details, the response body shows the message: "Contacto insertado con éxito". There are buttons for 'Copy Request' and 'Copy Response'. The browser's status bar at the bottom shows the date and time as 02/07/2023 at 09:30 p.m.



Showing results 0 - 1 (total 2). The query took 0.0005 seconds.

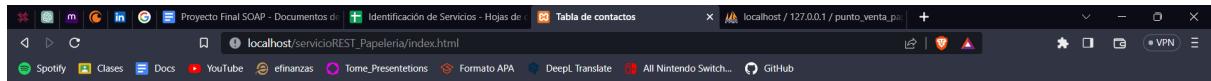
	Id	nombre	telefono	email
<input type="checkbox"/>	2	Acosta	771269898641	Acosta@gmail.com
<input type="checkbox"/>	3	Fernando Cruz	7721798399	Fernando_Cruz@gmail.com

Operations on the query results:

- [Print](#)
- [Copy to clipboard](#)
- [Export](#)
- [Show graphic](#)
- [Create view](#)

GET

Interfaz

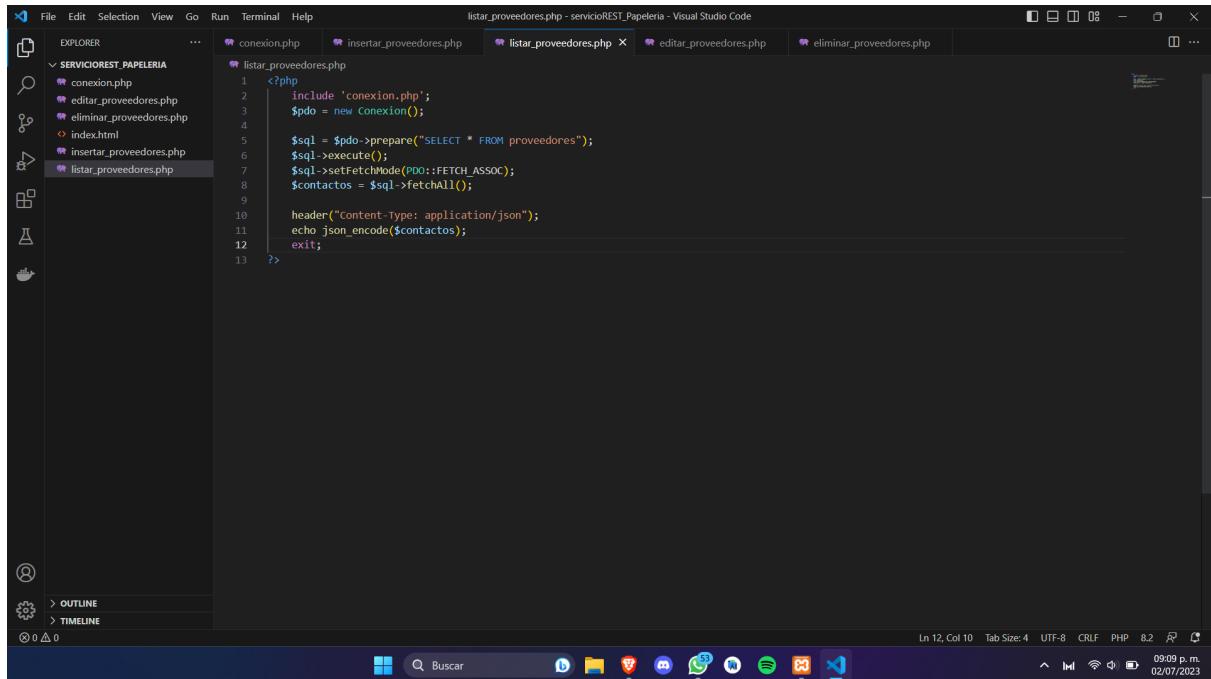


Proveedores

Listar proveedores			
2	Acosta	771269898641	Acosta@gmail.com
4	Fer	787787	Fer@gmail.com
5	fernando	3243	Fer@gmail.com

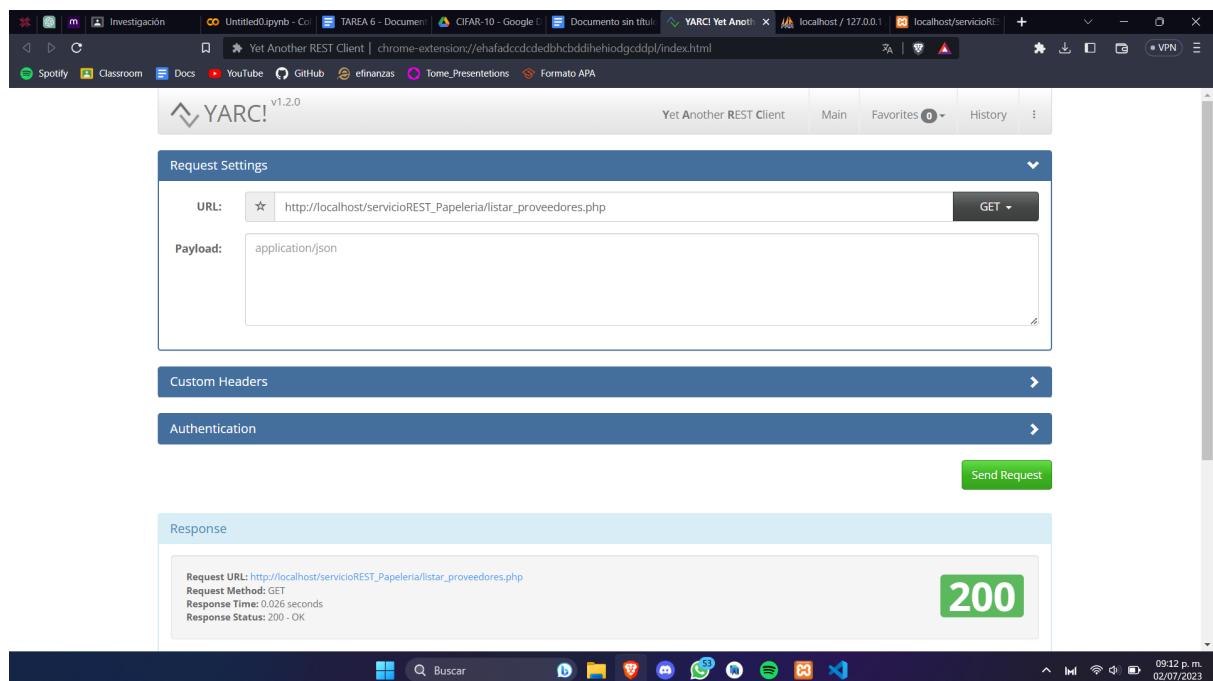


Código

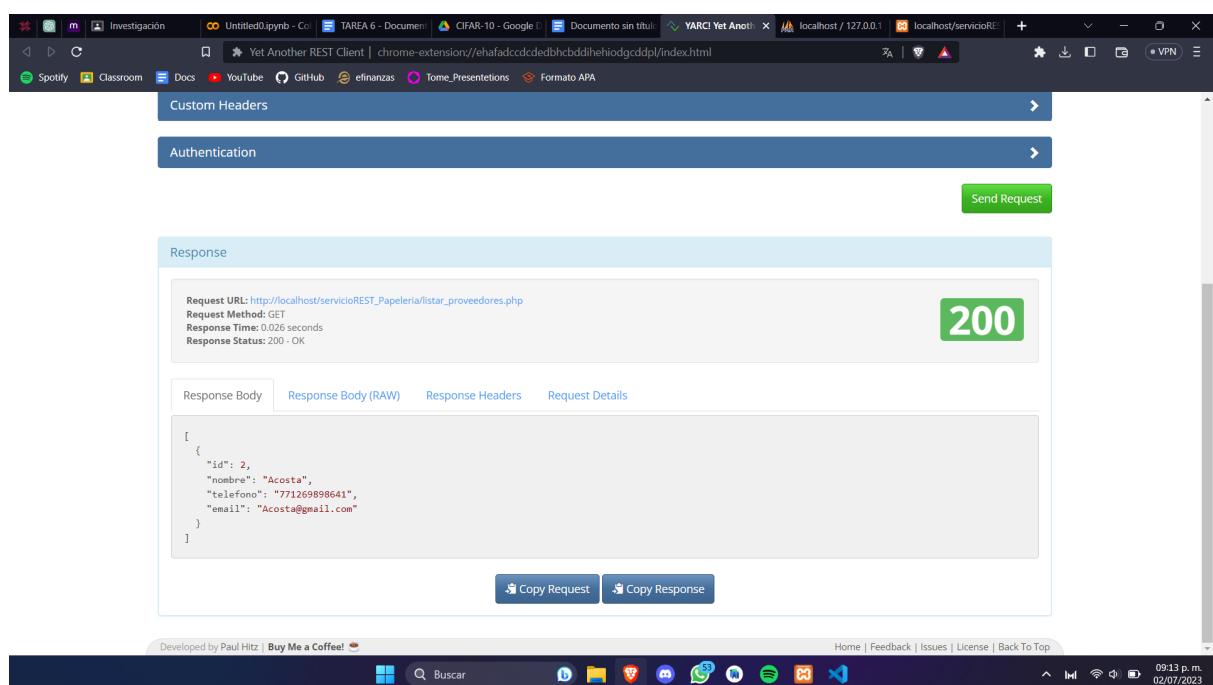


```
<?php
include 'conexion.php';
$pdo = new Conexion();
$sql = $pdo->prepare("SELECT * FROM proveedores");
$sql->execute();
$sql->setFetchMode(PDO::FETCH_ASSOC);
$contactos = $sql->fetchAll();
header("Content-Type: application/json");
echo json_encode($contactos);
exit;
?>
```

Funcionamiento



The screenshot shows the YARCI REST Client interface. In the Request Settings section, the URL is set to `http://localhost/servicioREST_Papeleria/listar_proveedores.php` and the method is `GET`. The response section shows a green **200** status with the following details: Request URL: `http://localhost/servicioREST_Papeleria/listar_proveedores.php`, Request Method: `GET`, Response Time: 0.026 seconds, and Response Status: 200 - OK.



The screenshot shows the YARCI REST Client interface. In the Request Settings section, the URL is set to `http://localhost/servicioREST_Papeleria/listar_proveedores.php` and the method is `GET`. The response section shows a green **200** status with the following details: Request URL: `http://localhost/servicioREST_Papeleria/listar_proveedores.php`, Request Method: `GET`, Response Time: 0.026 seconds, and Response Status: 200 - OK. The Response Body tab is selected, displaying the following JSON array:

```
[{"id": 2, "nombre": "Acosta", "telefono": "771269898641", "email": "Acosta@gmail.com"}]
```

At the bottom of the interface, there are `Copy Request` and `Copy Response` buttons.

localhost / 127.0.0.1 > Base de datos: punto_venta_papelaria > Tabla: proveedores

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Disparadores

Mostrando filas 0 - 0 (total de 1. La consulta tardó 0,0156 segundos.)

SELECT * FROM `proveedores`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla

Opciones extra

	id	nombre	telefono	email
	2	Acosta	771269898641	Acosta@gmail.com

Seleccionar todo Para los elementos que están marcados: Editar Copiar Borrar Editar Copiar Borrar Exportar

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla

Operaciones sobre los resultados de la consulta

Imprimir Copiar al portapapeles Exportar Mostrar gráfico Crear vista

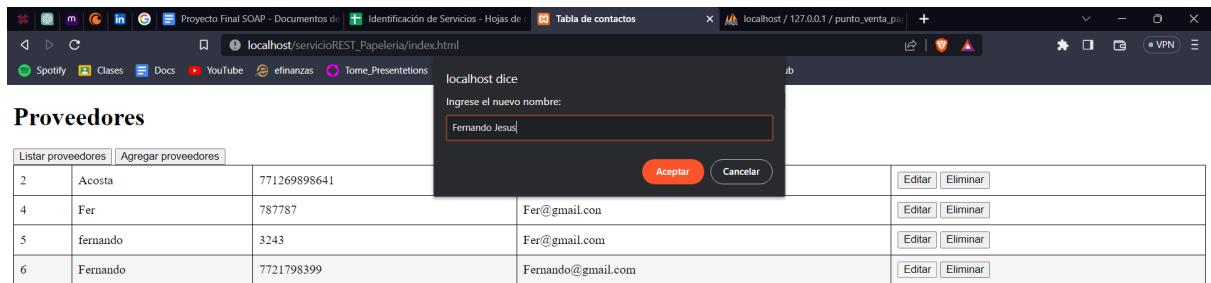
Consola

Buscar

09:13 p.m. 02/07/2023

PUT

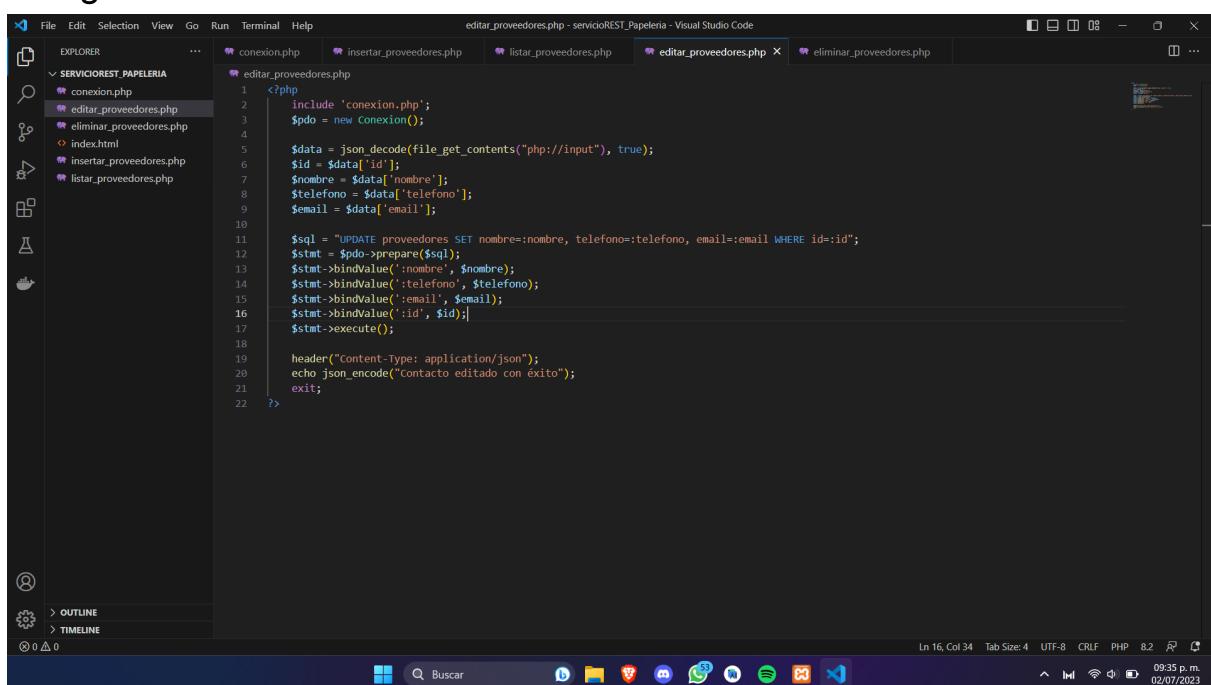
Interfaz



The screenshot shows a web browser with multiple tabs open. The active tab is 'localhost/servicioREST_Papelaria/index.html'. A modal window is displayed in the center, titled 'localhost dice' (localhost says). It contains the text 'Ingrese el nuevo nombre:' (Enter the new name:) and a text input field with the value 'Fernando Jesus'. Below the input field are two buttons: 'Aceptar' (Accept) and 'Cancelar' (Cancel). In the background, there is a table titled 'Proveedores' (Suppliers) with the following data:

	Nombre	Telefono	Email	Actions
2	Acosta	771269898641		<button>Editar</button> <button>Eliminar</button>
4	Fer	787787	Fer@gmail.com	<button>Editar</button> <button>Eliminar</button>
5	fernando	3243	Fer@gmail.com	<button>Editar</button> <button>Eliminar</button>
6	Fernando	7721798399	Fernando@gmail.com	<button>Editar</button> <button>Eliminar</button>

Código



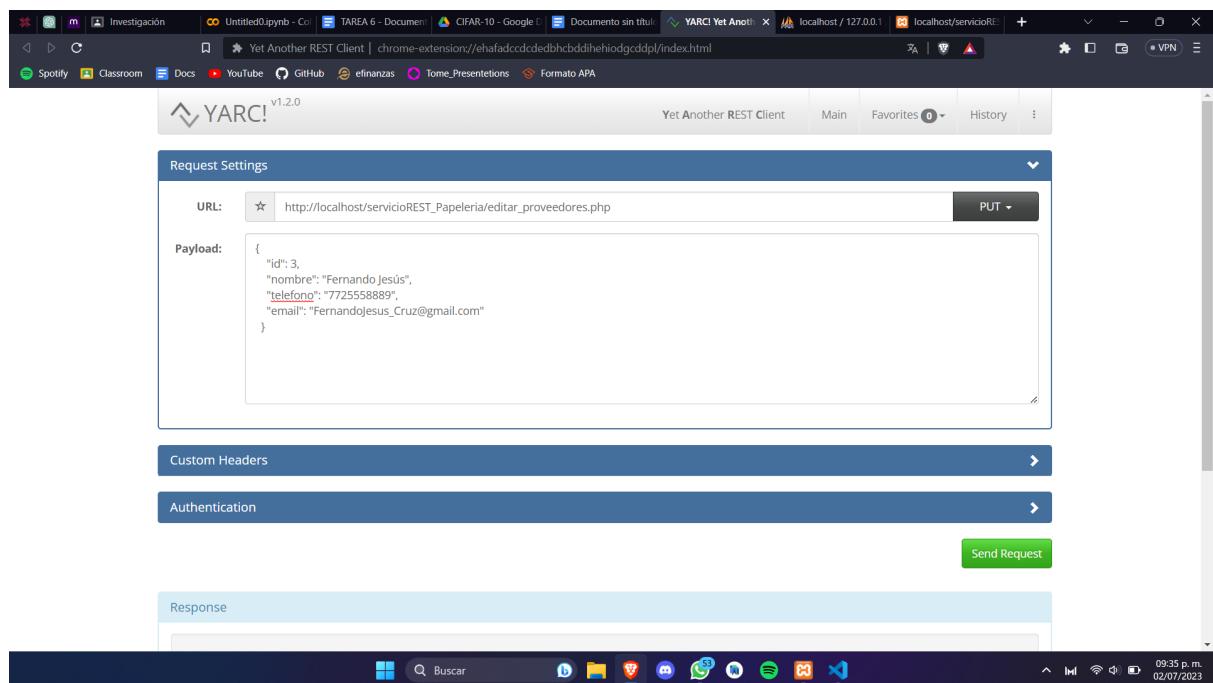
The screenshot shows the Visual Studio Code interface with the file 'editar_proveedores.php' open. The code is a PHP script for updating supplier information. It includes error handling, JSON decoding of input data, and preparation and execution of an UPDATE query. The code is as follows:

```
<?php
include 'conexion.php';
$pdo = new Conexion();
$data = json_decode(file_get_contents("php://input"), true);
$id = $data['id'];
$nombre = $data['nombre'];
$telefono = $data['telefono'];
$email = $data['email'];

$sql = "UPDATE proveedores SET nombre=:nombre, telefono=:telefono, email=:email WHERE id=:id";
$stmt = $pdo->prepare($sql);
$stmt->bindParam(':nombre', $nombre);
$stmt->bindParam(':telefono', $telefono);
$stmt->bindParam(':email', $email);
$stmt->bindParam(':id', $id);
$stmt->execute();

header("Content-Type: application/json");
echo json_encode("Contacto editado con éxito");
exit;
?>
```

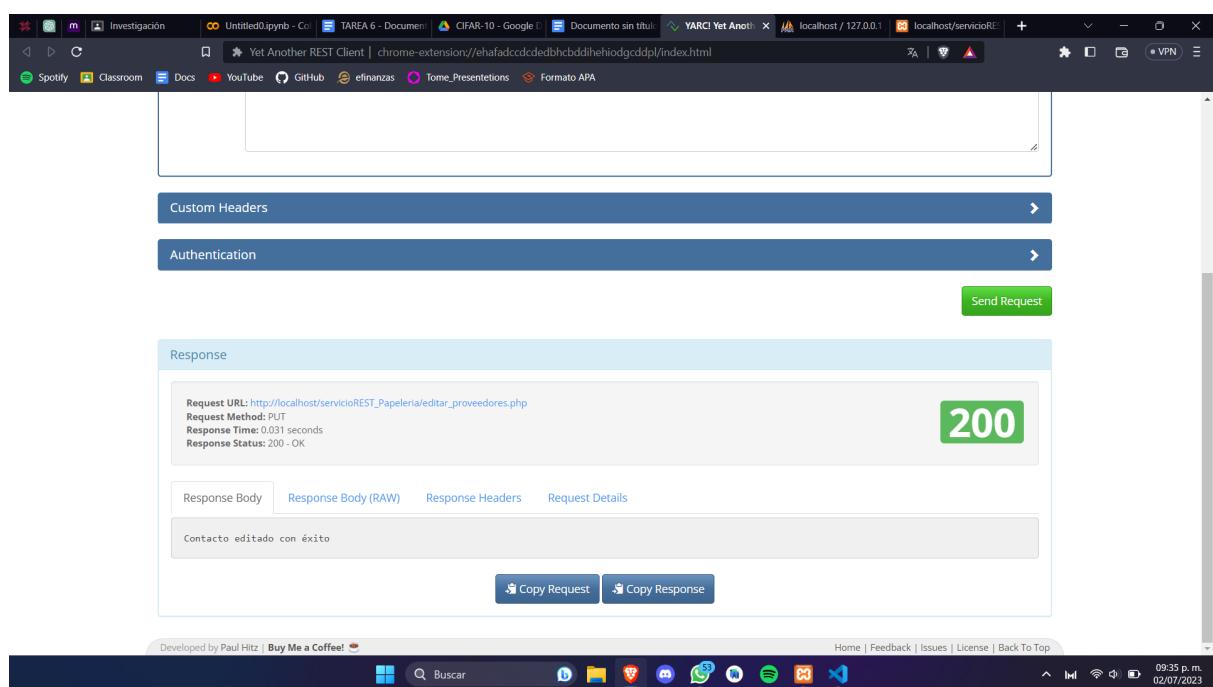
Funcionamiento



The screenshot shows the YARCI REST Client interface. In the 'Request Settings' section, the URL is set to `http://localhost/servicioREST_Papelaria/editar_proveedores.php` and the method is set to `PUT`. The 'Payload' field contains the following JSON object:

```
{ "id": 3, "nombre": "Fernando Jesús", "telefono": "7725558889", "email": "Fernandojesus_Cruz@gmail.com" }
```

Below the request settings, there are sections for 'Custom Headers' and 'Authentication'. A green 'Send Request' button is located to the right of the payload field. The 'Response' section is currently empty. The system status bar at the bottom shows the date and time as 09:35 p.m. 02/07/2023.

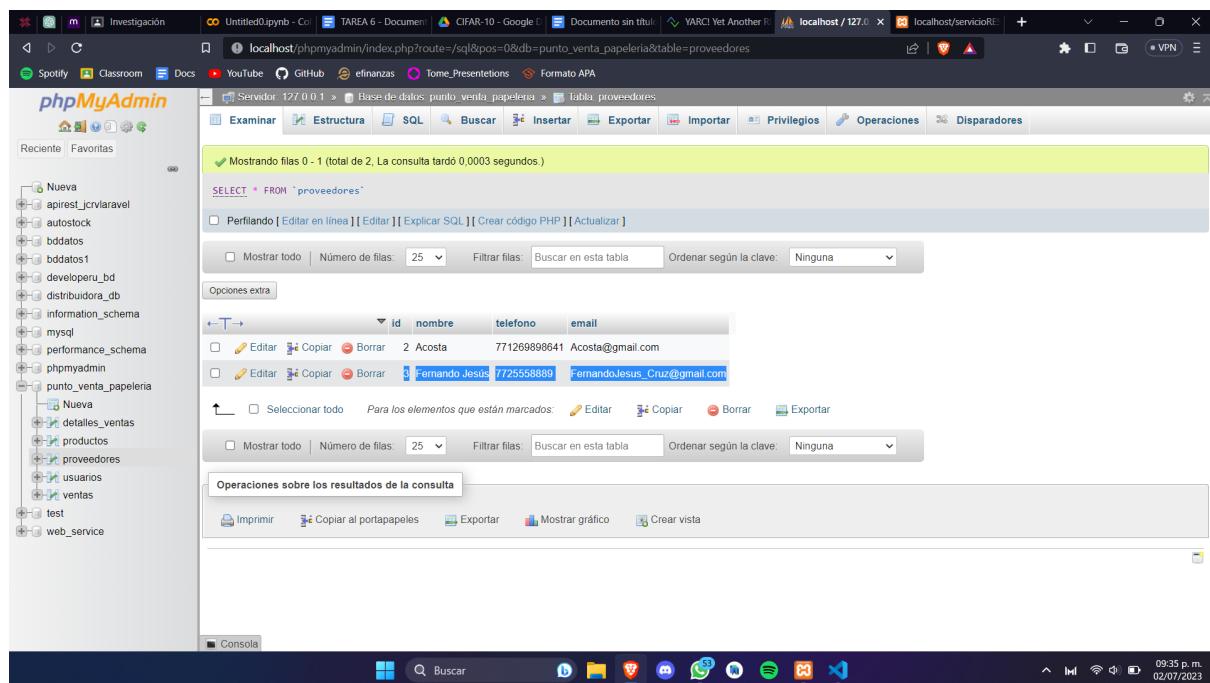


The screenshot shows the YARCI REST Client interface after a successful request. The 'Response' section displays the following details:

- Request URL: http://localhost/servicioREST_Papelaria/editar_proveedores.php
- Request Method: PUT
- Response Time: 0.031 seconds
- Response Status: 200 - OK

The response body is a green box containing the message: **200**
Contacto editado con éxito

Below the response body, there are buttons for 'Copy Request' and 'Copy Response'. The system status bar at the bottom shows the date and time as 09:35 p.m. 02/07/2023.



The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists databases and tables. The main area shows the 'proveedores' table in the 'punto_venta_papelaria' database. The table has columns: id, nombre, telefono, and email. There are two rows of data:

	id	nombre	telefono	email
	2	Acosta	771269898641	Acosta@gmail.com
	3	Fernando Jesus	7725558899	Fernando.Jesus_Cruz@gmail.com

Below the table, there are buttons for 'Imprimir', 'Copiar al portapapeles', 'Exportar', 'Mostrar gráfico', and 'Crear vista'.

DELETE

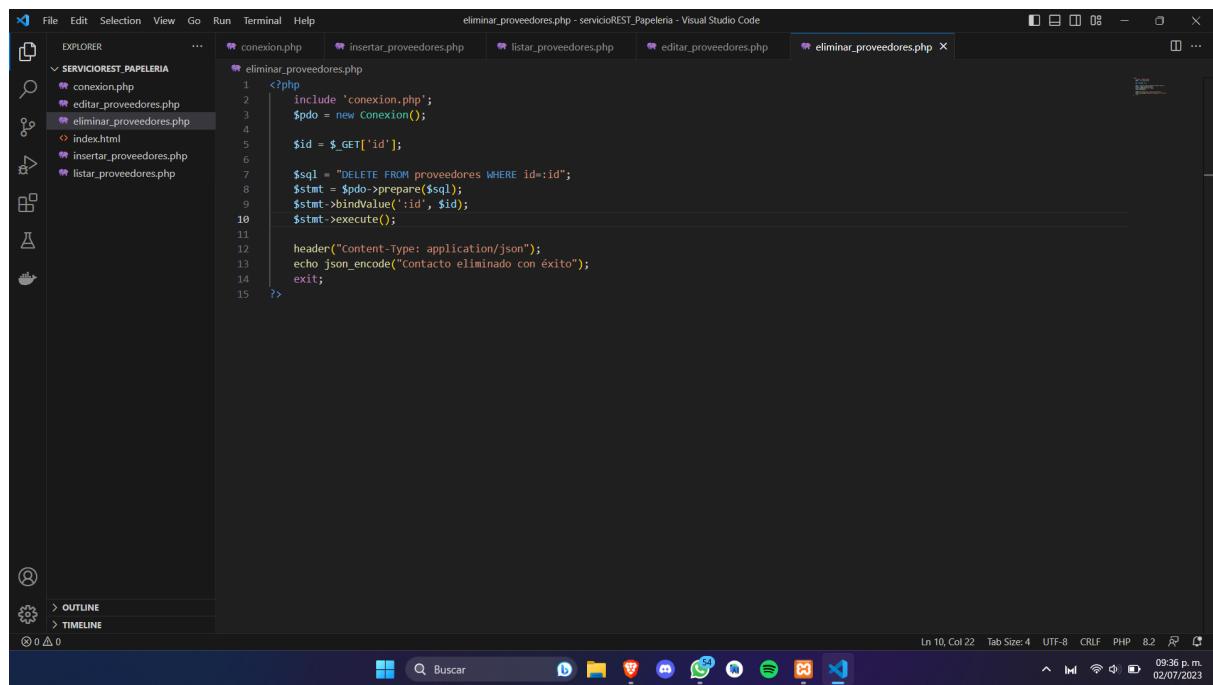
Interfaz



A screenshot of a web browser window. The title bar shows "localhost/servicioREST_Papelaria/index.html". A confirmation dialog box is overlaid on the page, asking "localhost dice ¿Estás seguro de que quieres eliminar este contacto?". Below the dialog, a table titled "Proveedores" is displayed, showing a list of suppliers with columns for ID, Name, Phone, Email, Edit, and Delete buttons.

	Nombre	Telefono	Correo	Editar	Eliminar
2	Acosta	771269898641	Acosta@gmail.com	<button>Editar</button>	<button>Eliminar</button>
4	Fer	787787	Fer@gmail.com	<button>Editar</button>	<button>Eliminar</button>
5	fernando	3243	Fer@gmail.com	<button>Editar</button>	<button>Eliminar</button>
6	Fernando Jesus	777	Jesus@gmail.com	<button>Editar</button>	<button>Eliminar</button>

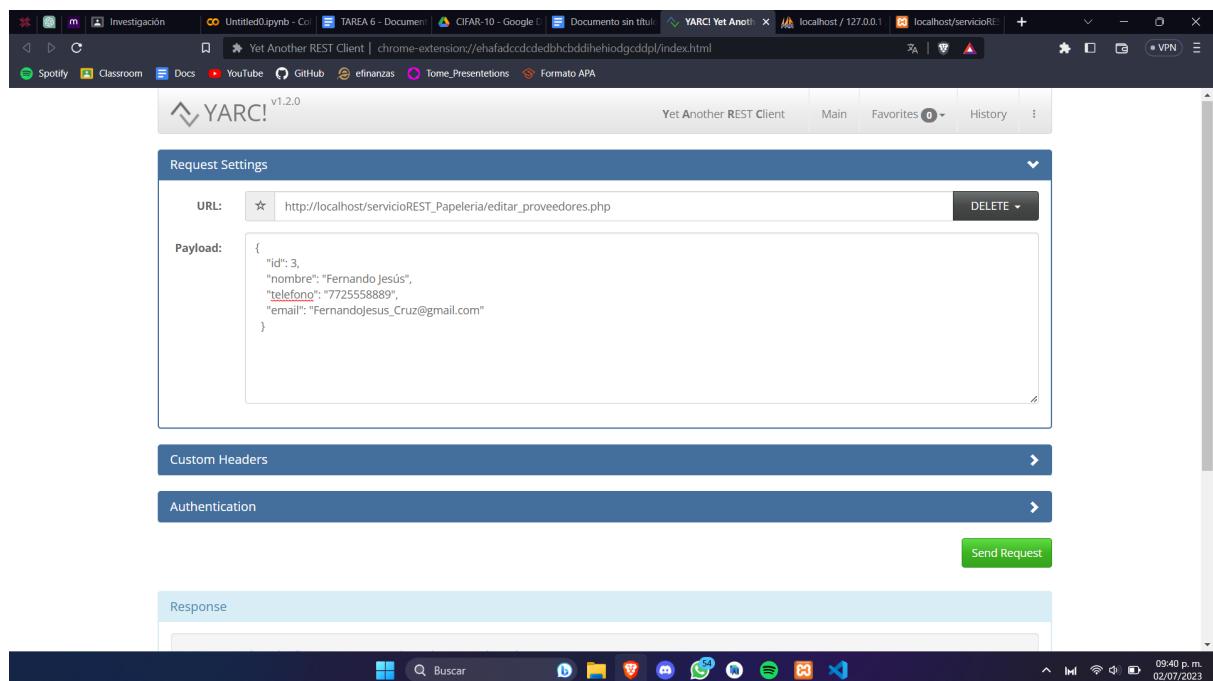
Código



A screenshot of the Visual Studio Code interface. The left sidebar shows a file tree with a folder named "SERVICIOREST_PAPELERIA" containing several PHP files: "conexion.php", "eliminar_proveedores.php", "index.html", "insertar_proveedores.php", and "listar_proveedores.php". The main editor tab is "eliminar_proveedores.php", which contains the following PHP code:

```
<?php
    include 'conexion.php';
    $pdo = new Conexion();
    $id = $_GET['id'];
    $sql = "DELETE FROM proveedores WHERE id=:id";
    $stmt = $pdo->prepare($sql);
    $stmt->bindParam(':id', $id);
    $stmt->execute();
    header("Content-Type: application/json");
    echo json_encode("Contacto eliminado con éxito");
    exit;
?>
```

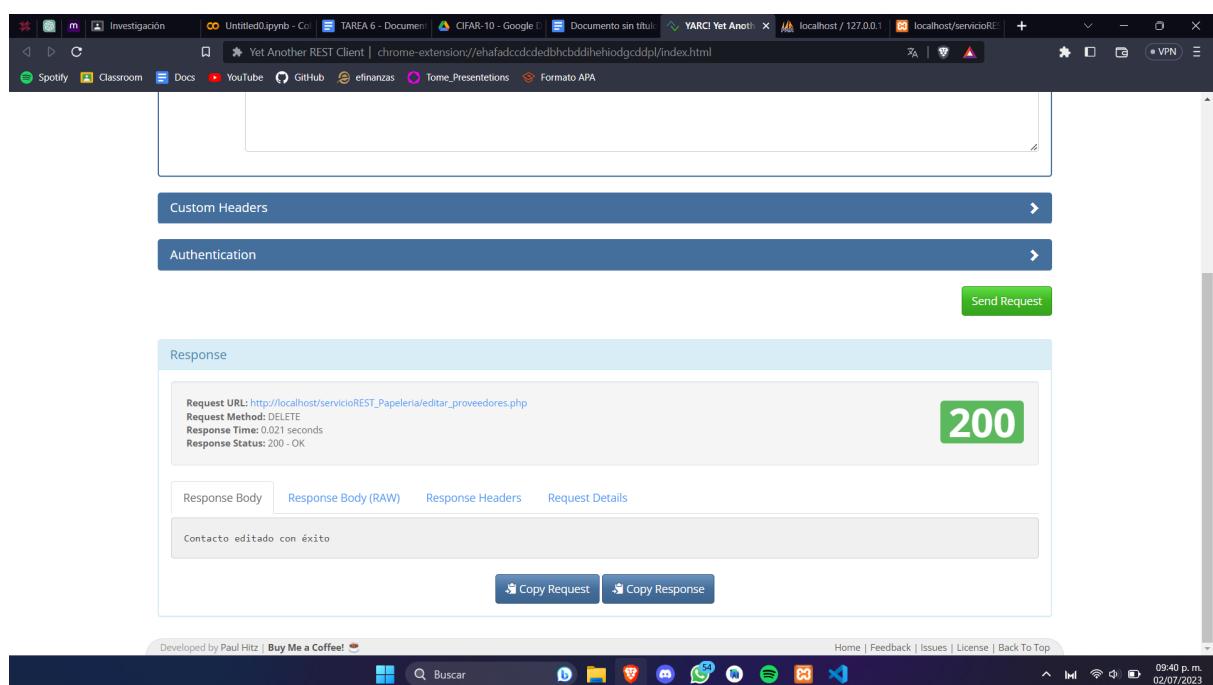
Funcionamiento



The screenshot shows the YARCI REST Client interface. In the 'Request Settings' section, the URL is set to `http://localhost/servicioREST_Papeleria/editar_proveedores.php`. The 'Payload' field contains the following JSON data:

```
{ "id": 3, "nombre": "Fernando Jesús", "telefono": "772558889", "email": "Fernandojesus_Cruz@gmail.com" }
```

Below the request settings, there are sections for 'Custom Headers' and 'Authentication'. A green 'Send Request' button is located to the right of the payload field. The 'Response' section is currently empty. The system status bar at the bottom shows the date and time as 02/07/2023 at 09:40 p.m.



The screenshot shows the YARCI REST Client interface after the request has been sent. The 'Response' section now displays the following information:

Request URL: `http://localhost/servicioREST_Papeleria/editar_proveedores.php`
Request Method: DELETE
Response Time: 0.021 seconds
Response Status: 200 - OK

A large green '200' indicates a successful response. Below this, a message box shows the text: 'Contacto editado con éxito'. At the bottom of the response section are buttons for 'Copy Request' and 'Copy Response'. The system status bar at the bottom shows the date and time as 02/07/2023 at 09:40 p.m.

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists databases: Nueva, apirest_laravel, autostock, bddatos, bddatos1, developeru_db, distribuidora_db, information_schema, mysql, performance_schema, phpmyadmin, punto_venta_papelaria, test, and web_service. The 'punto_venta_papelaria' database is selected. The main area shows the 'proveedores' table with the following data:

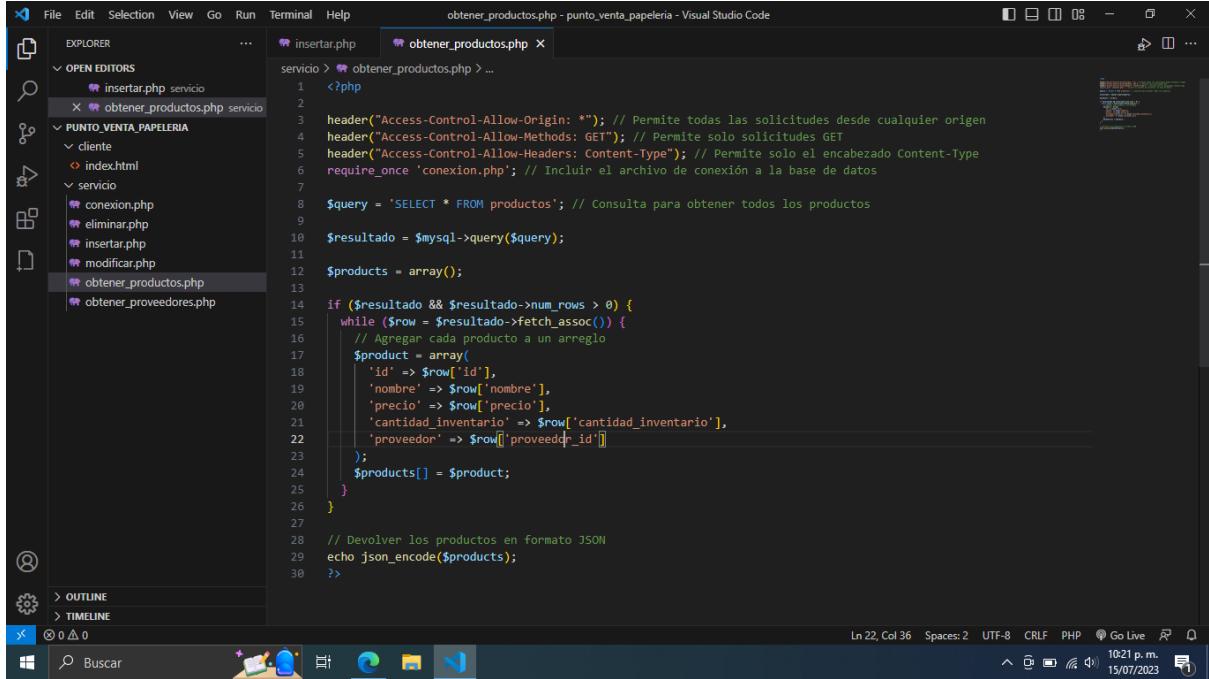
	id	nombre	telefono	email
<input checked="" type="checkbox"/>	2	Acosta	771269898641	Acosta@gmail.com

Below the table, there are buttons for 'Imprimir', 'Copiar al portapapeles', 'Exportar', 'Mostrar gráfico', and 'Crear vista'.

Servicio Productos (Angela)

Método GET (Productos)

Código



```
<?php
header("Access-Control-Allow-Origin: *"); // Permite todas las solicitudes desde cualquier origen
header("Access-Control-Allow-Methods: GET"); // Permite solo solicitudes GET
header("Access-Control-Allow-Headers: Content-Type"); // Permite solo el encabezado Content-Type
require_once 'conexion.php'; // Incluir el archivo de conexión a la base de datos

$query = 'SELECT * FROM productos'; // Consulta para obtener todos los productos

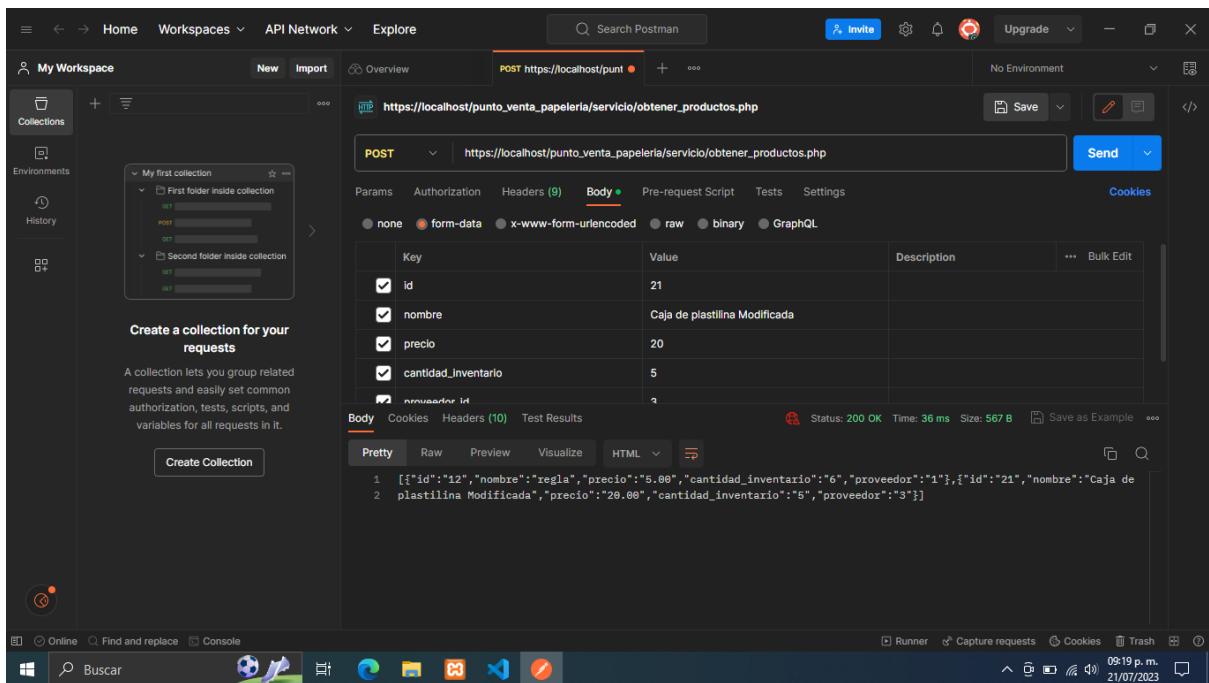
$resultado = $mysql->query($query);

$products = array();

if ($resultado && $resultado->num_rows > 0) {
    while ($row = $resultado->fetch_assoc()) {
        // Agregar cada producto a un arreglo
        $product = array(
            'id' => $row['id'],
            'nombre' => $row['nombre'],
            'precio' => $row['precio'],
            'cantidad_inventario' => $row['cantidad_inventario'],
            'proveedor' => $row['proveedor_id']
        );
        $products[] = $product;
    }
}

// Devolver los productos en formato JSON
echo json_encode($products);
?>
```

Funcionamiento Postman



POST https://localhost/punto_venta_papelaria/servicio/obtener_productos.php

Key	Value	Description
id	21	
nombre	Caja de plastilina Modificada	
precio	20	
cantidad_inventario	5	
proveedor_id	2	

1. [{"id": "21", "nombre": "Caja de plastilina Modificada", "precio": "20.00", "cantidad_inventario": "5", "proveedor": "2"}, {"id": "22", "nombre": "Regla", "precio": "5.00", "cantidad_inventario": "6", "proveedor": "1"}]

Funcionamiento Cliente

Nombre	Costo	Cantidad	Proveedor	Acciones
Tijeras	50.00	16	1	<button>Editar</button> <button>Eliminar</button>
regla	5.00	6	1	<button>Editar</button> <button>Eliminar</button>
Lápices de colores	50.00	26	1	<button>Editar</button> <button>Eliminar</button>
Goma	2.50	12	1	<button>Editar</button> <button>Eliminar</button>

Método GET (Proveedores)

Código

```
header("Access-Control-Allow-Origin: *"); // Permite todas las solicitudes desde cualquier origen
header("Access-Control-Allow-Methods: GET"); // Permite solo solicitudes GET
header("Access-Control-Allow-Headers: Content-Type"); // Permite solo el encabezado Content-Type
require_once 'conexion.php'; // Incluir el archivo de conexión a la base de datos

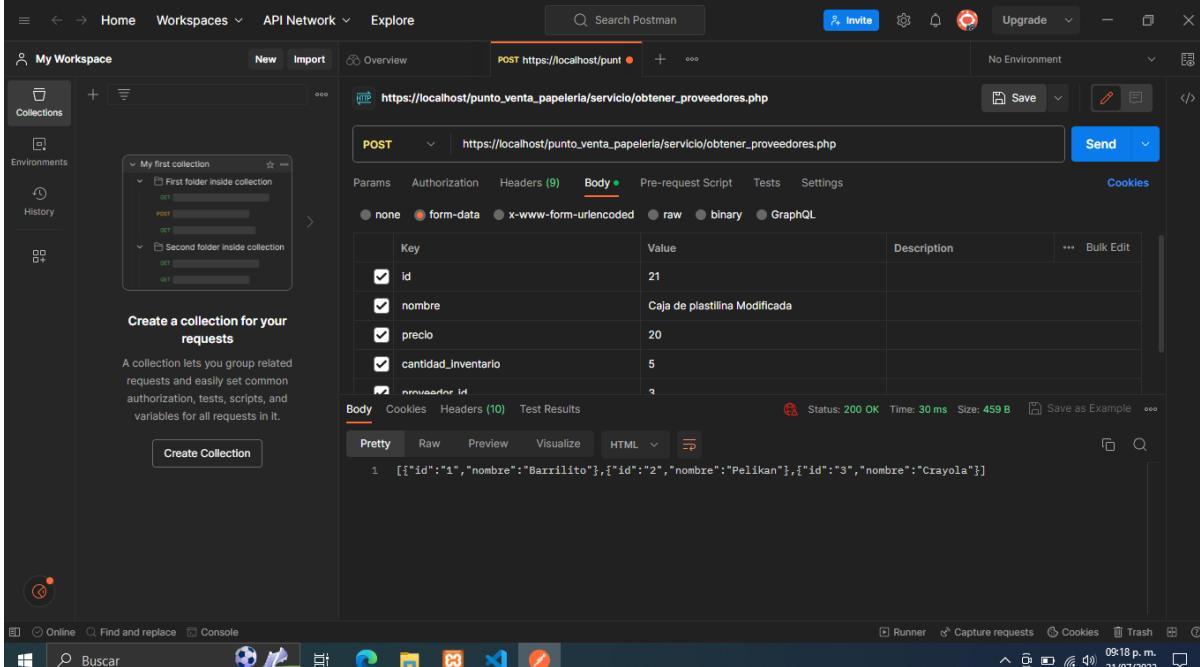
$query = 'SELECT * FROM proveedores'; // Consulta para obtener todos los productos

$resultado = $mysql->query($query);
$products = array();

if ($resultado && $resultado->num_rows > 0) {
    while ($row = $resultado->fetch_assoc()) {
        // Agregar cada producto a un arreglo
        $product = array(
            'id' => $row['id'],
            'nombre' => $row['nombre']
        );
        $products[] = $product;
    }
}

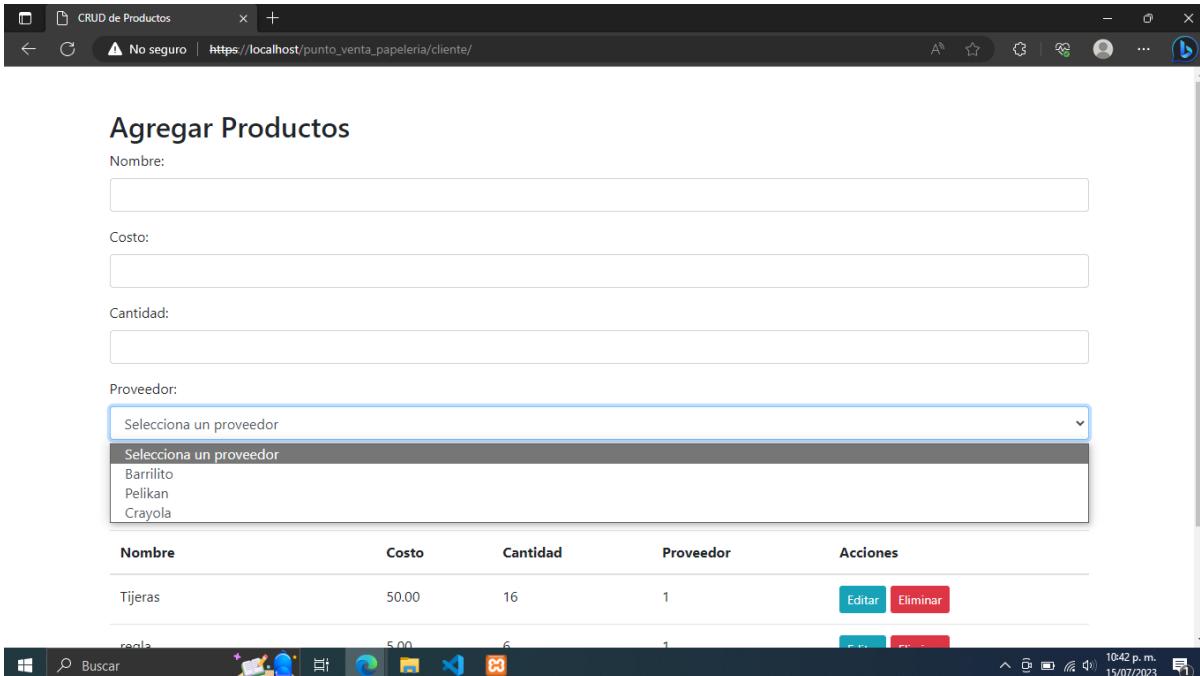
// Devolver los productos en formato JSON
echo json_encode($products);
?>
```

Funcionamiento Postman



The screenshot shows the Postman application interface. The left sidebar displays 'My Workspace' with 'My first collection' containing 'First folder inside collection' and 'Second folder inside collection', each with a 'GET' request. A 'Create a collection for your requests' section is present. The main area shows a POST request to 'https://localhost/punto_venta_papelaria/servicio/obtener_proveedores.php'. The 'Body' tab is selected, showing a JSON structure with fields: id (21), nombre (Caja de plastilina Modificada), precio (20), cantidad_inventario (5), and proveedor_id (4). The response status is 200 OK, time is 30 ms, and size is 459 B. The response body is a JSON array: [{"id": "1", "nombre": "Barrilito"}, {"id": "2", "nombre": "Pelikan"}, {"id": "3", "nombre": "Crayola"}]. The bottom status bar shows the date and time: 09:18 p. m. 21/07/2023.

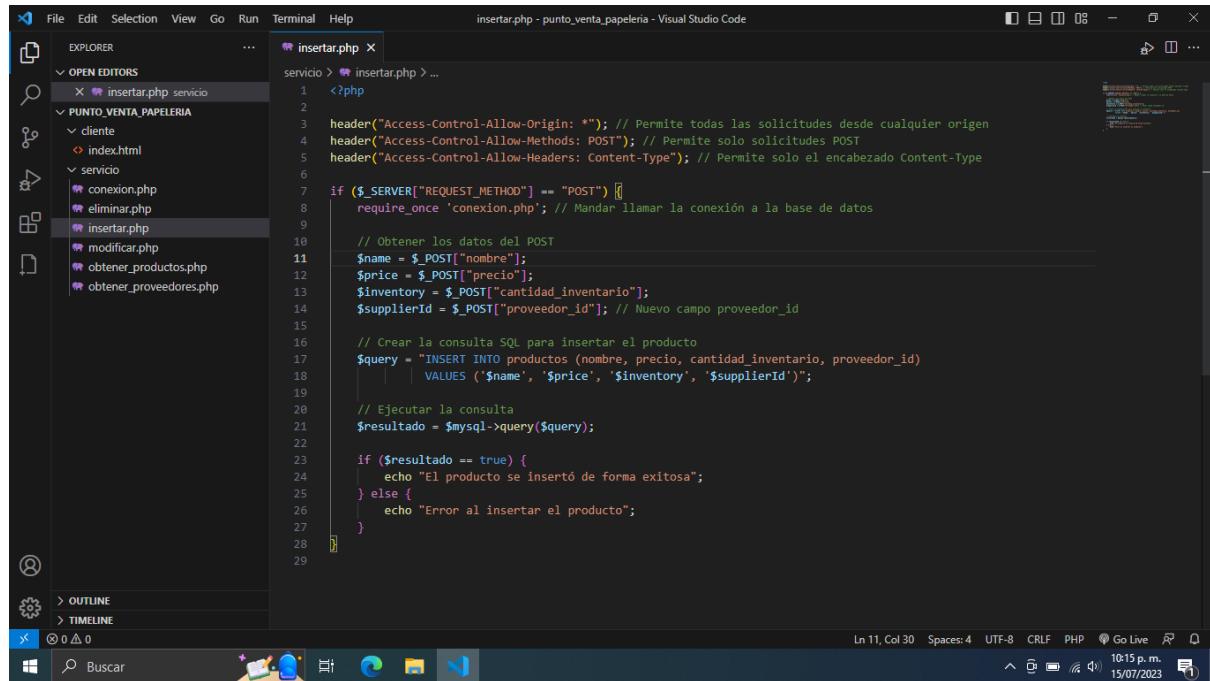
Funcionamiento Cliente



The screenshot shows a web browser window titled 'CRUD de Productos'. The address bar shows 'No seguro | https://localhost/punto_venta_papelaria/cliente/'. The main content is a form titled 'Agregar Productos' with fields for 'Nombre', 'Costo', 'Cantidad', and 'Proveedor'. The 'Proveedor' field is a dropdown menu with options: 'Selecciona un proveedor', 'Barrilito', 'Pelikan', and 'Crayola'. Below the form is a table with columns: Nombre, Costo, Cantidad, Proveedor, and Acciones. The table contains one row: 'Tijeras' with Costo 50.00, Cantidad 16, and Proveedor 1. There are 'Editar' and 'Eliminar' buttons in the 'Acciones' column. The bottom status bar shows the date and time: 10:42 p. m. 15/07/2023.

Método POST

Código



```
<?php
header("Access-Control-Allow-Origin: *"); // Permite todas las solicitudes desde cualquier origen
header("Access-Control-Allow-Methods: POST"); // Permite solo solicitudes POST
header("Access-Control-Allow-Headers: Content-Type"); // Permite solo el encabezado Content-Type

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    require_once 'conexion.php'; // Mandar llamar la conexión a la base de datos

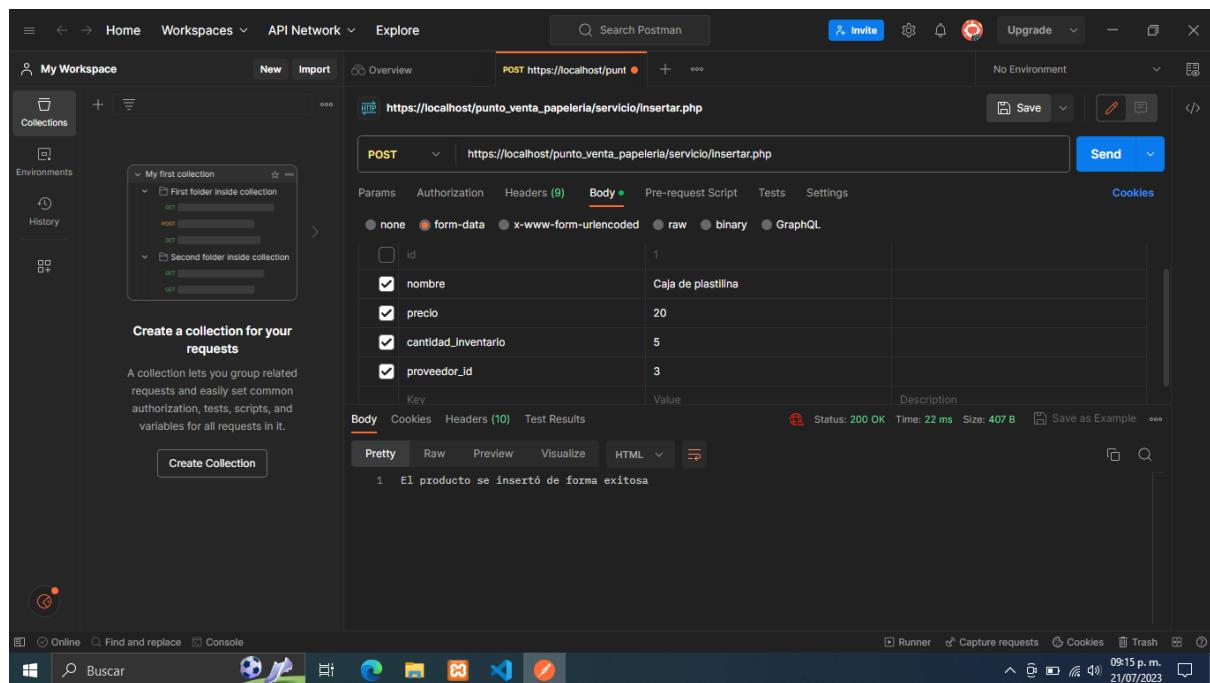
    // Obtener los datos del POST
    $name = $_POST["nombre"];
    $price = $_POST["precio"];
    $inventory = $_POST["cantidad_inventario"];
    $supplierId = $_POST["proveedor_id"]; // Nuevo campo proveedor_id

    // Crear la consulta SQL para insertar el producto
    $query = "INSERT INTO productos (nombre, precio, cantidad_inventario, proveedor_id)
              VALUES ('$name', '$price', '$inventory', '$supplierId')";

    // Ejecutar la consulta
    $resultado = $mysql->query($query);

    if ($resultado == true) {
        echo "El producto se insertó de forma exitosa";
    } else {
        echo "Error al insertar el producto";
    }
}
```

Funcionamiento Postman

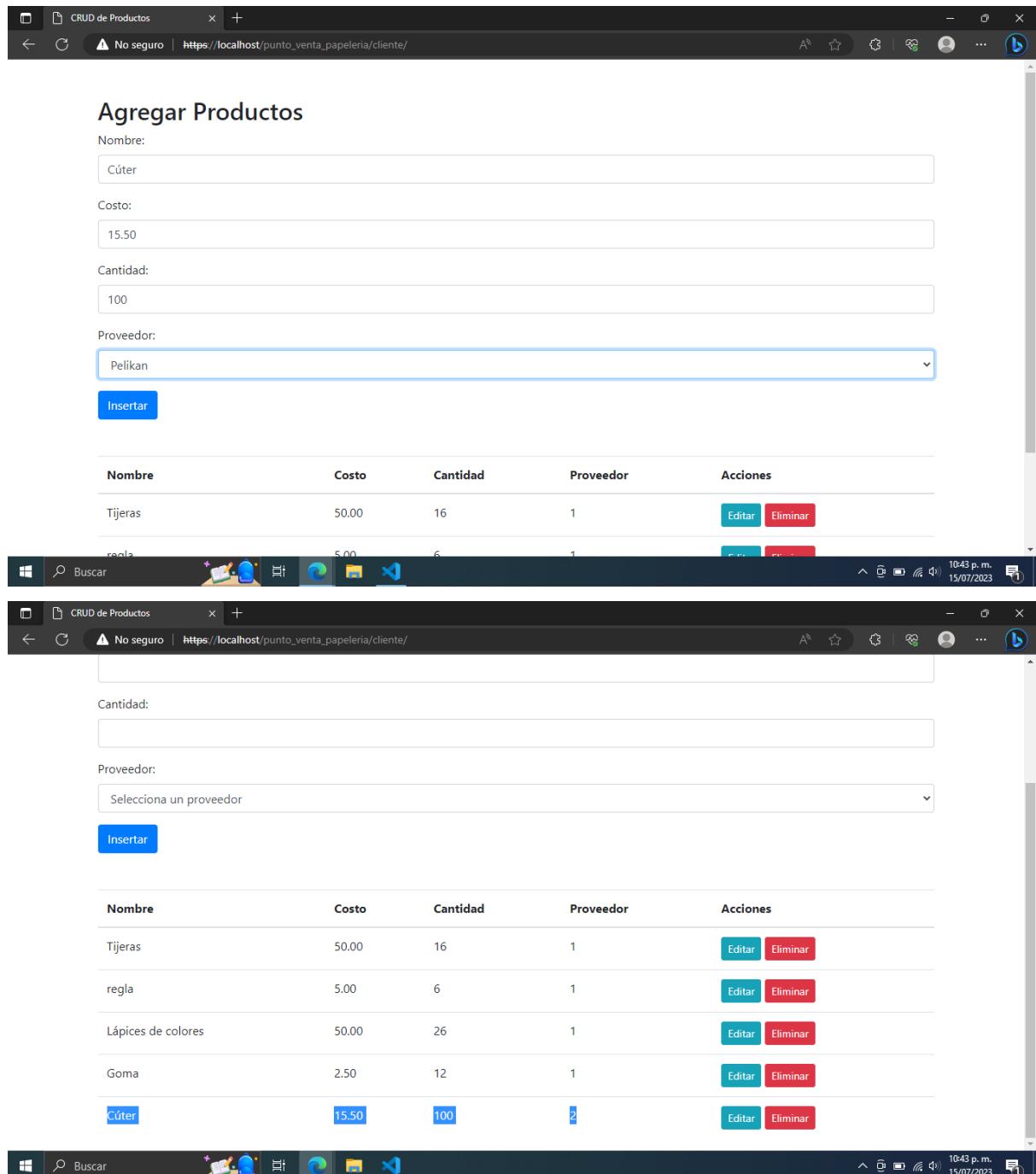


POST https://localhost/punto_venta_papelaria/servicio/insertar.php

Key	Value	Description
id	1	
nombre	Caja de plastilina	
precio	20	
cantidad_inventario	5	
proveedor_id	3	

1. El producto se insertó de forma exitosa

Funcionamiento Cliente

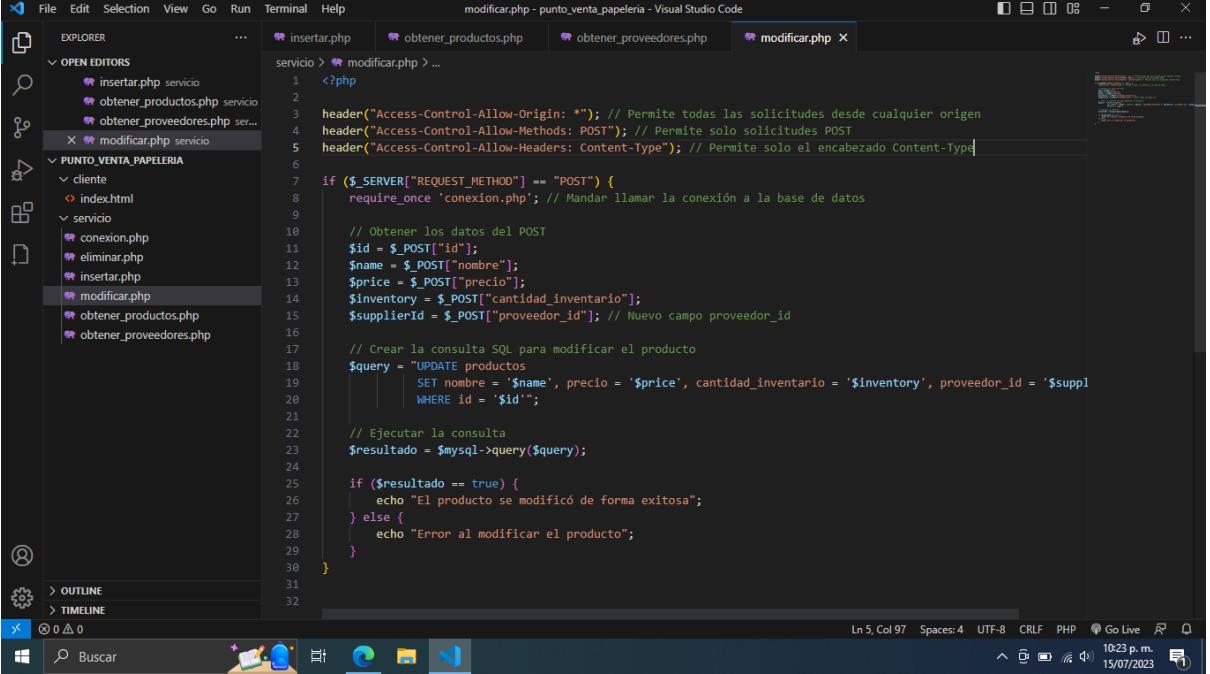


The screenshots illustrate the client-side functionality for adding products. The first screenshot shows the 'Agregar Productos' form with fields for Nombre (Cúter), Costo (15.50), Cantidad (100), and Proveedor (Pelikan). The second screenshot shows the same form but with the Proveedor field empty. The third screenshot shows a table of products with columns: Nombre, Costo, Cantidad, Proveedor, and Acciones (Editar, Eliminar). The table lists four products: Tijeras, regla, Lápices de colores, and Goma. The last row is for the product 'Cúter' with values 15.50, 100, and 2.

Nombre	Costo	Cantidad	Proveedor	Acciones
Tijeras	50.00	16	1	<button>Editar</button> <button>Eliminar</button>
regla	5.00	6	1	<button>Editar</button> <button>Eliminar</button>
Lápices de colores	50.00	26	1	<button>Editar</button> <button>Eliminar</button>
Goma	2.50	12	1	<button>Editar</button> <button>Eliminar</button>
Cúter	15.50	100	2	<button>Editar</button> <button>Eliminar</button>

Método PUT

Código



```
<?php
header("Access-Control-Allow-Origin: *"); // Permite todas las solicitudes desde cualquier origen
header("Access-Control-Allow-Methods: POST"); // Permite solo solicitudes POST
header("Access-Control-Allow-Headers: Content-Type"); // Permite solo el encabezado Content-Type

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    require_once 'conexion.php'; // Mandar llamar la conexión a la base de datos

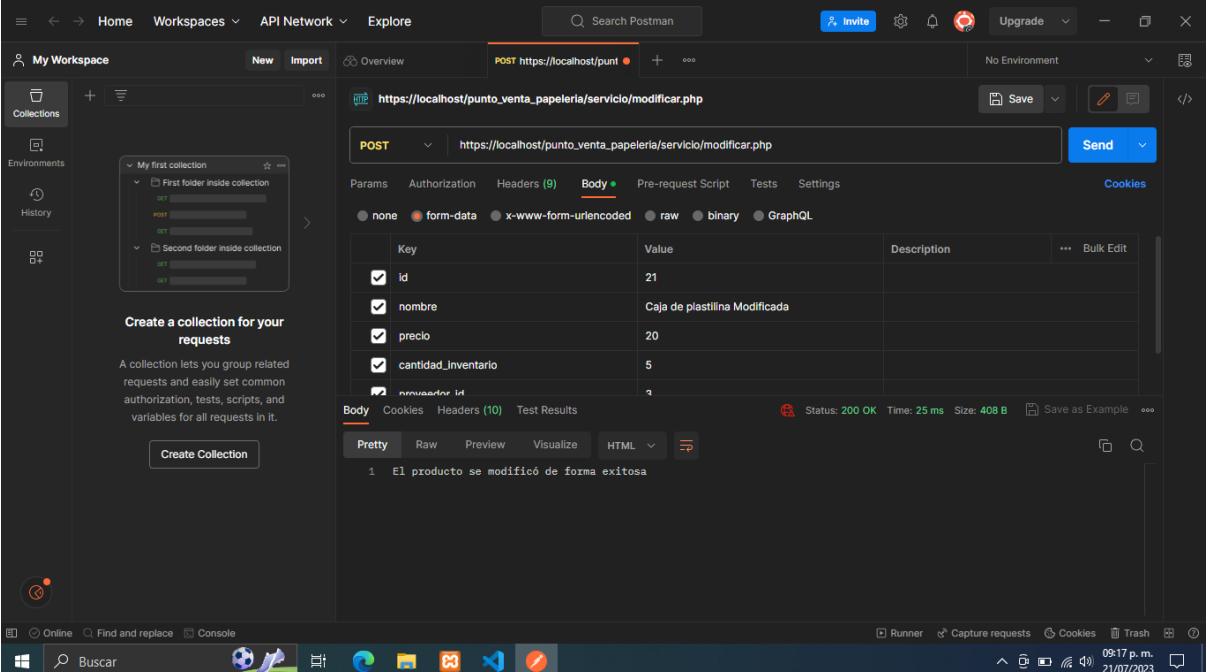
    // Obtener los datos del POST
    $id = $_POST["id"];
    $name = $_POST["nombre"];
    $price = $_POST["precio"];
    $inventory = $_POST["cantidad_inventario"];
    $supplierId = $_POST["proveedor_id"]; // Nuevo campo proveedor_id

    // Crear la consulta SQL para modificar el producto
    $query = "UPDATE productos
              SET nombre = '$name', precio = '$price', cantidad_inventario = '$inventory', proveedor_id = '$supplierId'
              WHERE id = '$id'";

    // Ejecutar la consulta
    $resultado = $mysql->query($query);

    if ($resultado == true) {
        echo "El producto se modificó de forma exitosa";
    } else {
        echo "Error al modificar el producto";
    }
}
```

Funcionamiento Postman

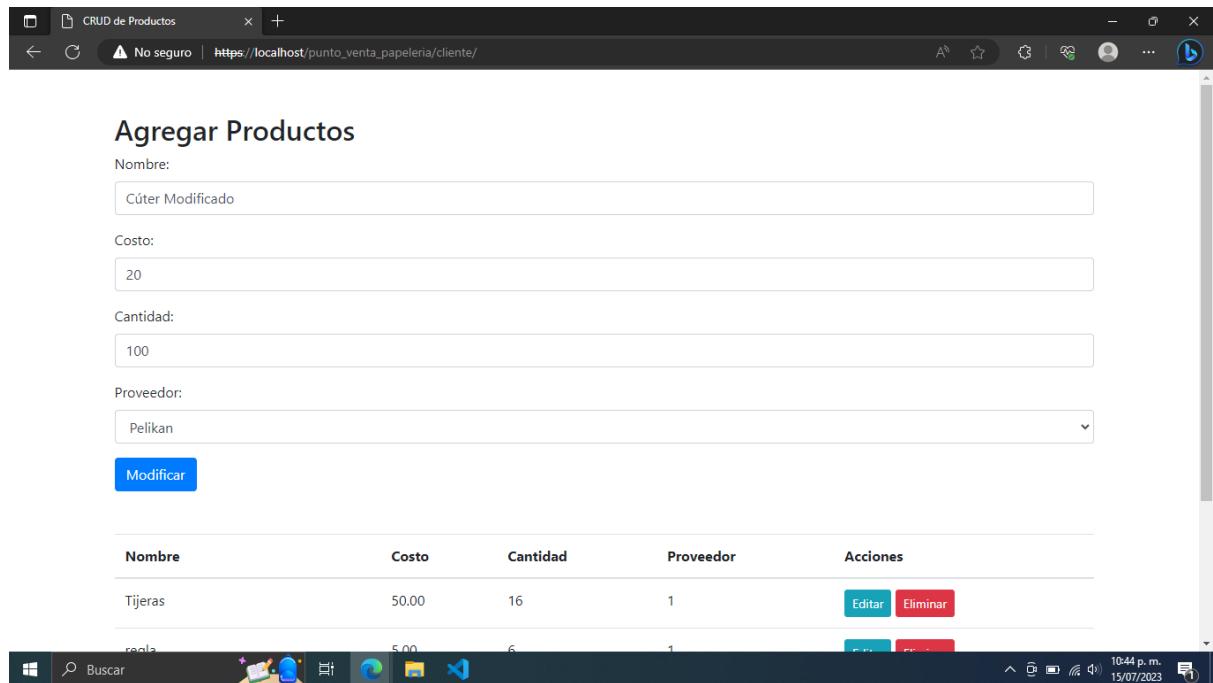


POST https://localhost/punto_venta_paqueria/servicio/modificar.php

Key	Value	Description
<input checked="" type="checkbox"/> id	21	
<input checked="" type="checkbox"/> nombre	Caja de pistachos Modificada	
<input checked="" type="checkbox"/> precio	20	
<input checked="" type="checkbox"/> cantidad_inventario	5	
<input checked="" type="checkbox"/> proveedor_id	1	

1 El producto se modificó de forma exitosa

Funcionamiento Cliente



Agregar Productos

Nombre:

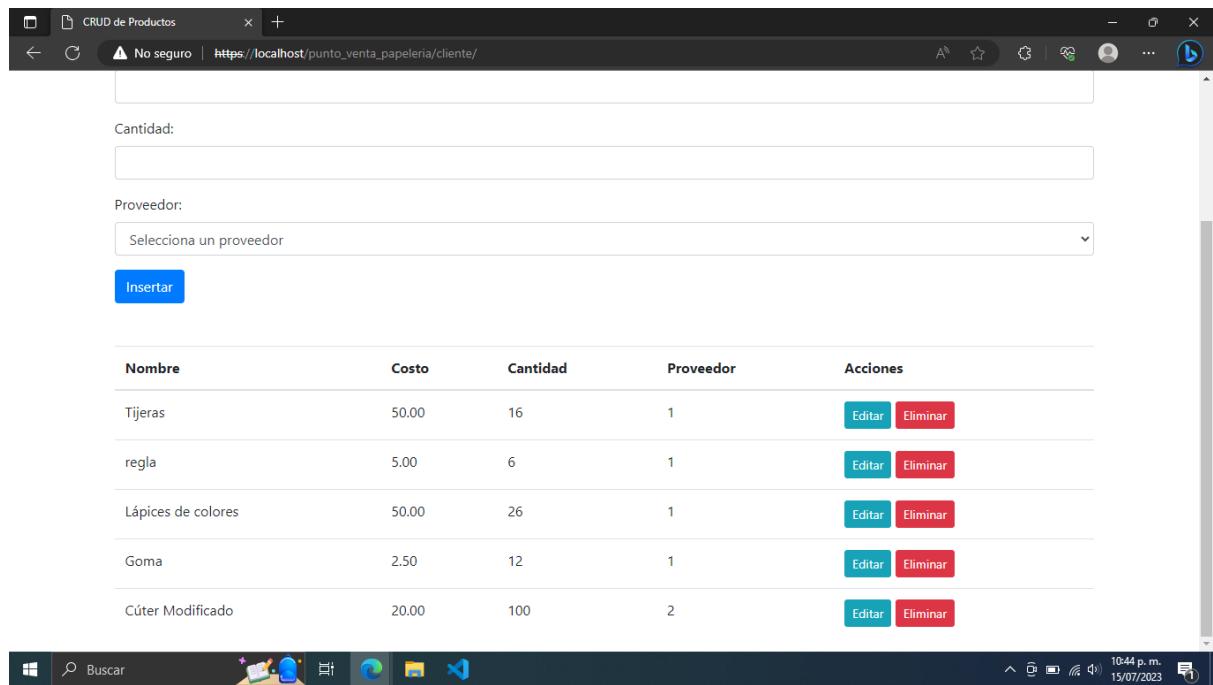
Costo:

Cantidad:

Proveedor:

Modificar

Nombre	Costo	Cantidad	Proveedor	Acciones
Tijeras	50.00	16	1	Editar Eliminar
regla	5.00	6	1	Editar Eliminar



Cantidad:

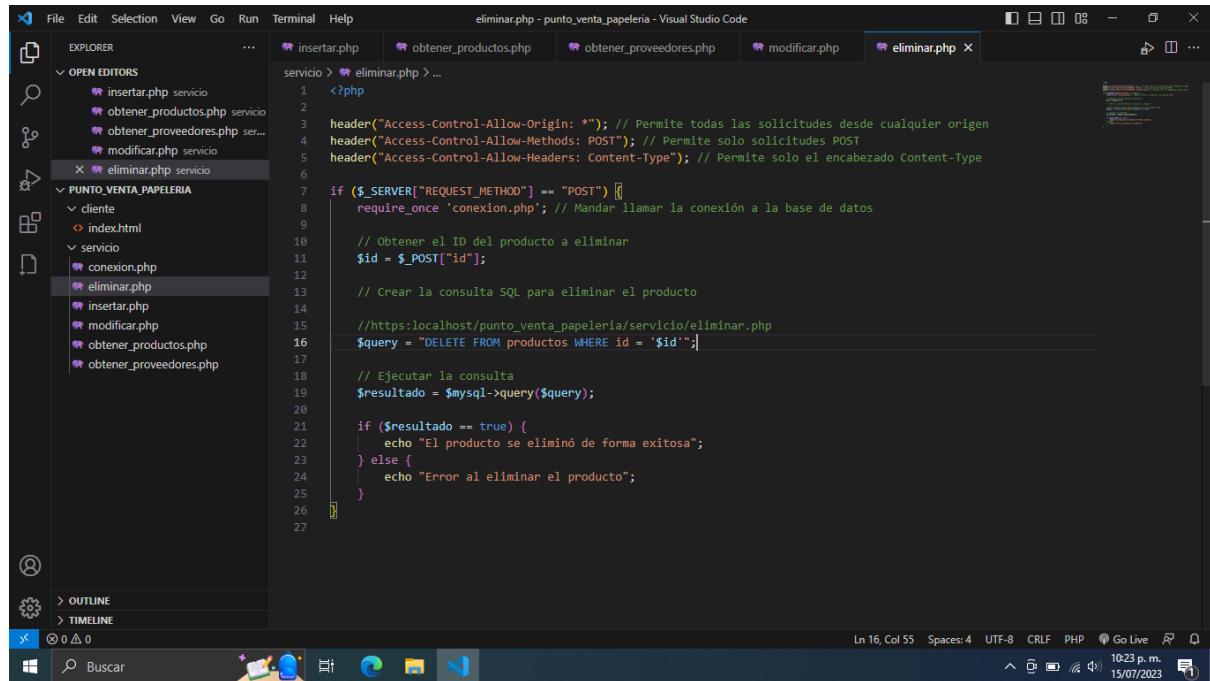
Proveedor:

Insertar

Nombre	Costo	Cantidad	Proveedor	Acciones
Tijeras	50.00	16	1	Editar Eliminar
regla	5.00	6	1	Editar Eliminar
Lápices de colores	50.00	26	1	Editar Eliminar
Goma	2.50	12	1	Editar Eliminar
Cúter Modificado	20.00	100	2	Editar Eliminar

Método DELETE

Código

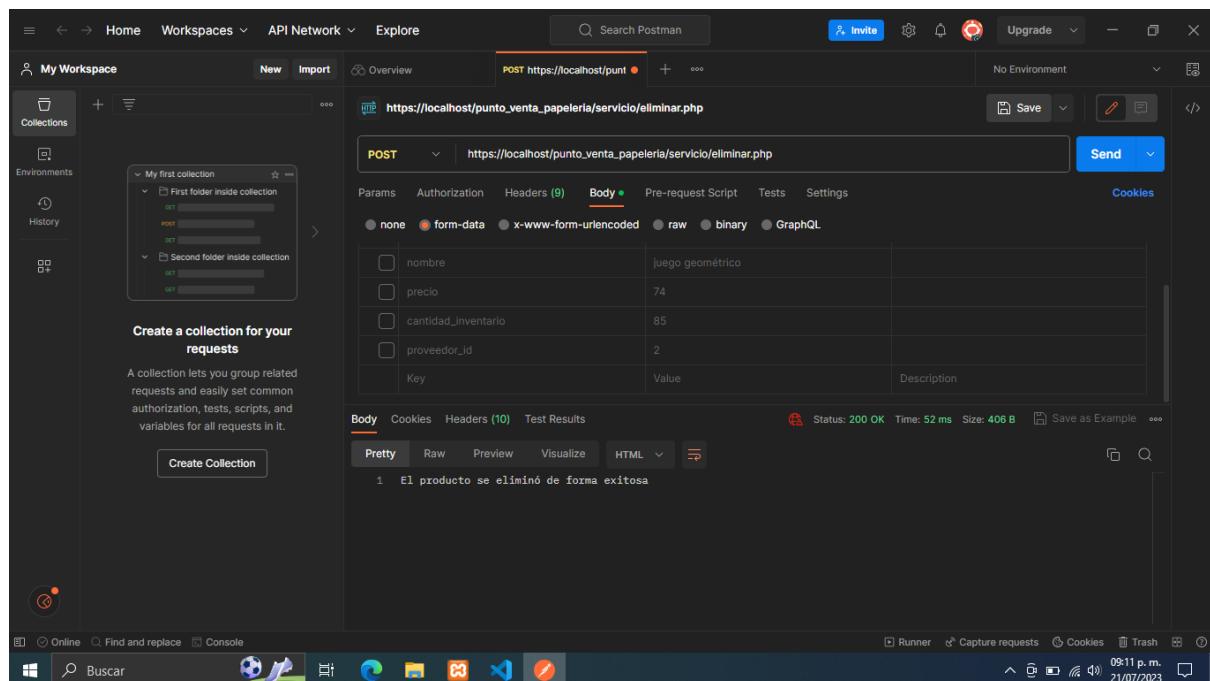


```
File Edit Selection View Go Run Terminal Help eliminar.php - punto_venta_paquete - Visual Studio Code

OPEN EDITORS
  insertar.php servicio
  obtener_productos.php servicio
  obtener_proveedores.php servicio
  modificar.php servicio
  eliminar.php servicio
  servicio > eliminar.php ...
  1 <?php
  2
  3 header("Access-Control-Allow-Origin: *"); // Permite todas las solicitudes desde cualquier origen
  4 header("Access-Control-Allow-Methods: POST"); // Permite solo solicitudes POST
  5 header("Access-Control-Allow-Headers: Content-Type"); // Permite solo el encabezado Content-Type
  6
  7 if ($_SERVER["REQUEST_METHOD"] == "POST") {
  8     require_once 'conexion.php'; // Mandar llamar la conexión a la base de datos
  9
 10    // Obtener el ID del producto a eliminar
 11    $id = $_POST["id"];
 12
 13    // Crear la consulta SQL para eliminar el producto
 14
 15    //https://localhost/punto_venta_paquete/servicio/eliminar.php
 16    $query = "DELETE FROM productos WHERE id = '$id'";
 17
 18    // Ejecutar la consulta
 19    $resultado = $mysql->query($query);
 20
 21    if ($resultado == true) {
 22        echo "El producto se eliminó de forma exitosa";
 23    } else {
 24        echo "Error al eliminar el producto";
 25    }
 26
 27 }

Ln 16, Col 55  Spaces:4  UTF-8  CRLF  PHP  Go Live  10:23 p.m.  15/07/2023
```

Funcionamiento Postman

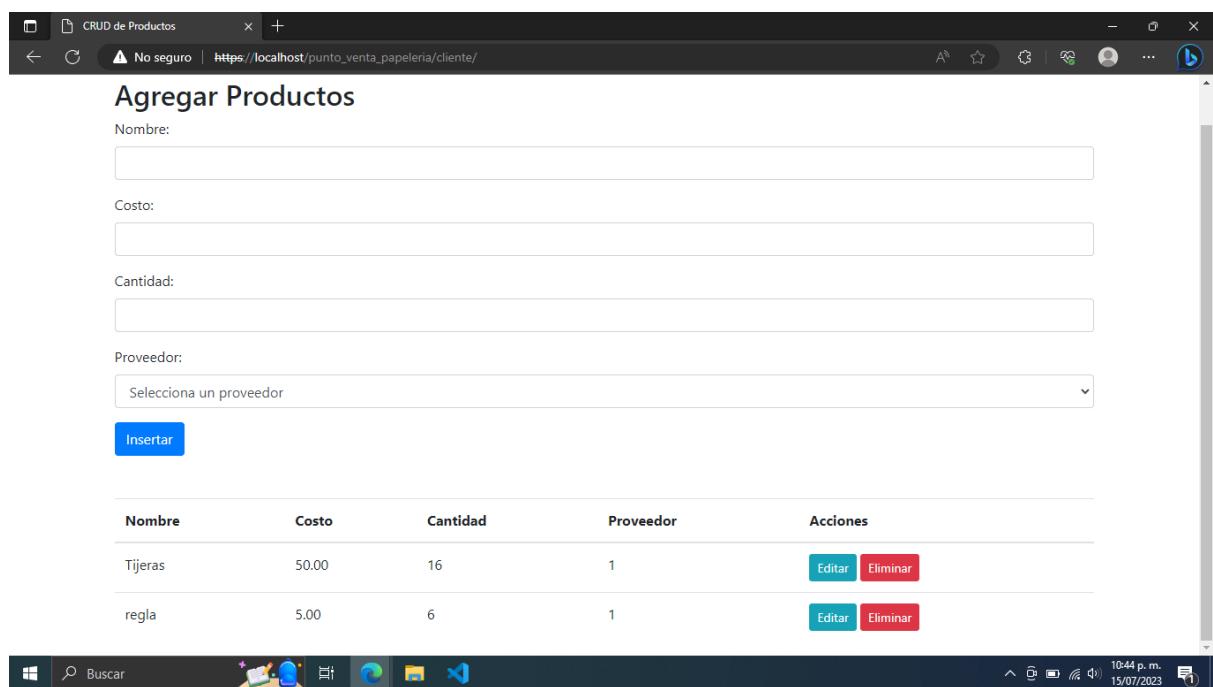


POST https://localhost/punto_venta_paquete/servicio/eliminar.php

Key	Value	Description
nombre	juego geométrico	
precio	74	
cantidad_inventario	85	
proveedor_id	2	

1. El producto se eliminó de forma exitosa

Funcionamiento Cliente



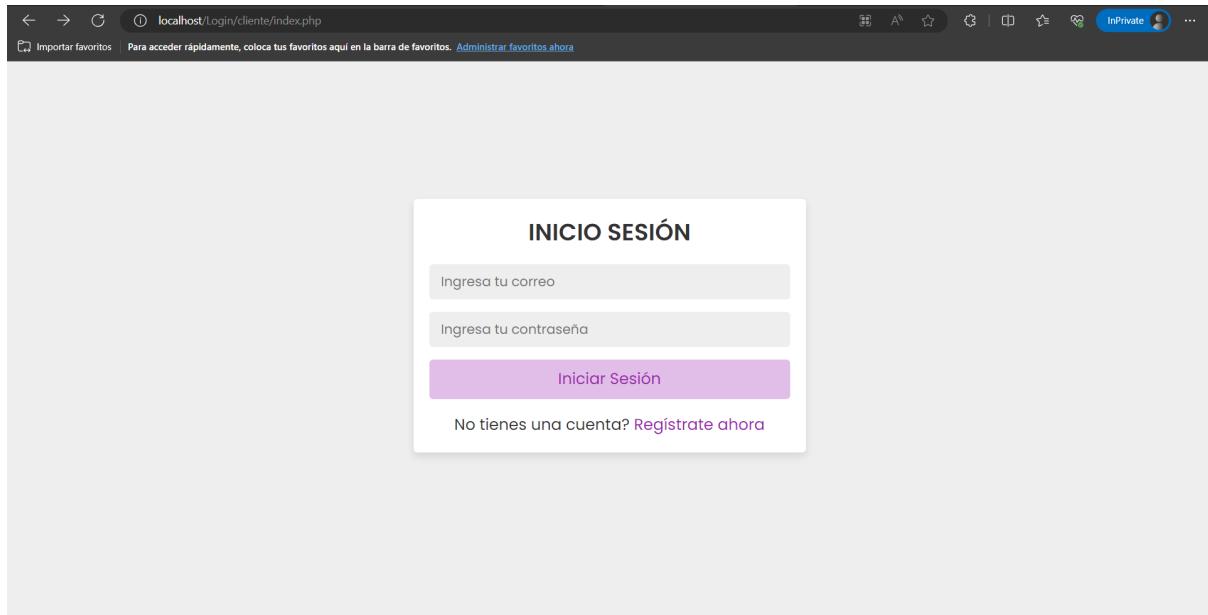
The screenshot shows a web browser window with the title "CRUD de Productos". The address bar indicates the URL is https://localhost/punto_venta_papeleria/cliente/. The page displays a form titled "Agregar Productos" for adding new products. The form fields are: "Nombre" (Name), "Costo" (Cost), "Cantidad" (Quantity), and "Proveedor" (Supplier). The "Proveedor" field is a dropdown menu with the placeholder "Selecciona un proveedor". Below the form is a table showing existing products:

Nombre	Costo	Cantidad	Proveedor	Acciones
Tijeras	50.00	16	1	Editar Eliminar
regla	5.00	6	1	Editar Eliminar

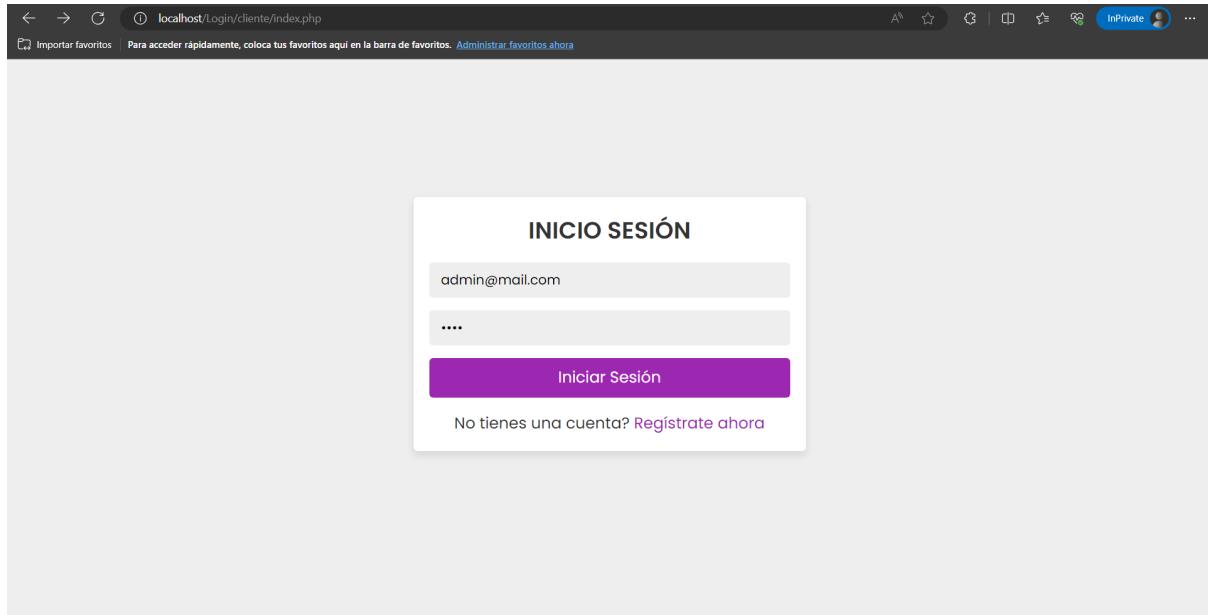
At the bottom of the browser window, the taskbar shows various pinned icons and the system tray displays the date and time as 15/07/2023 at 10:44 p.m.

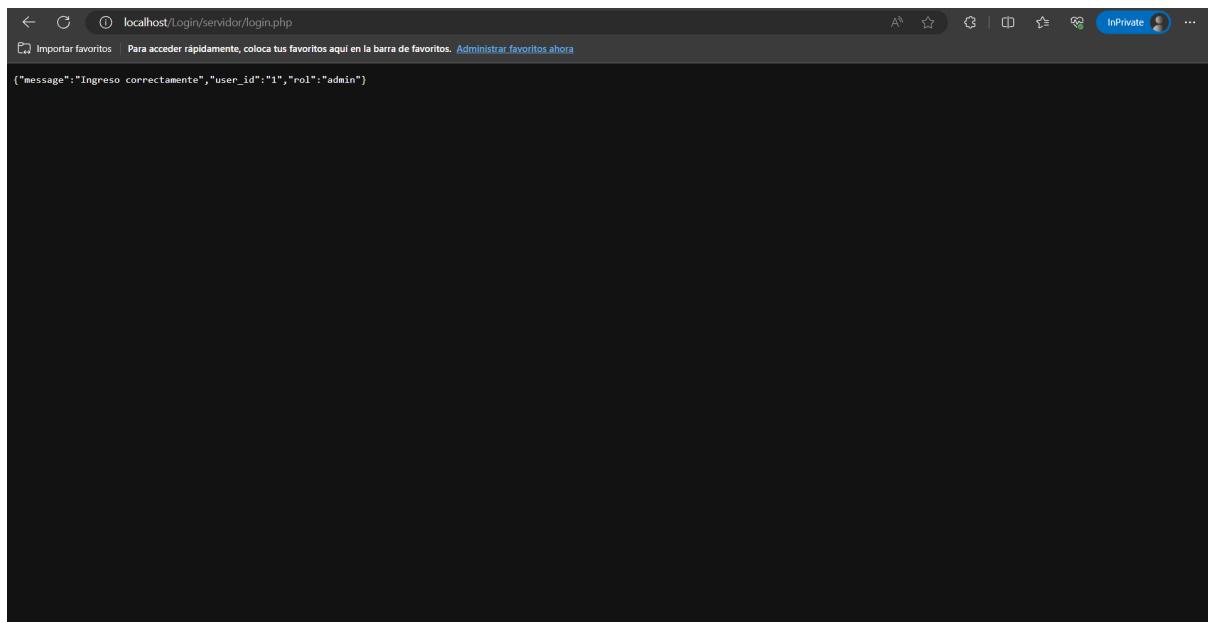
Servicio Inicio de Sesión (Angel)

Interfaz

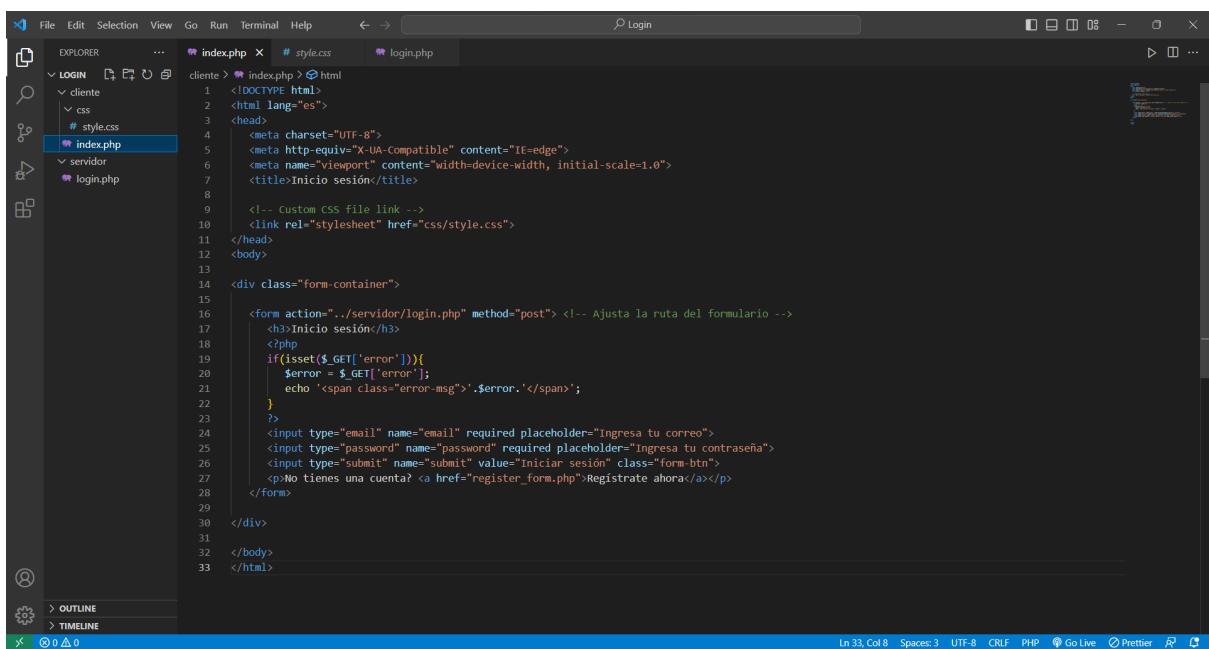


Funcionamiento

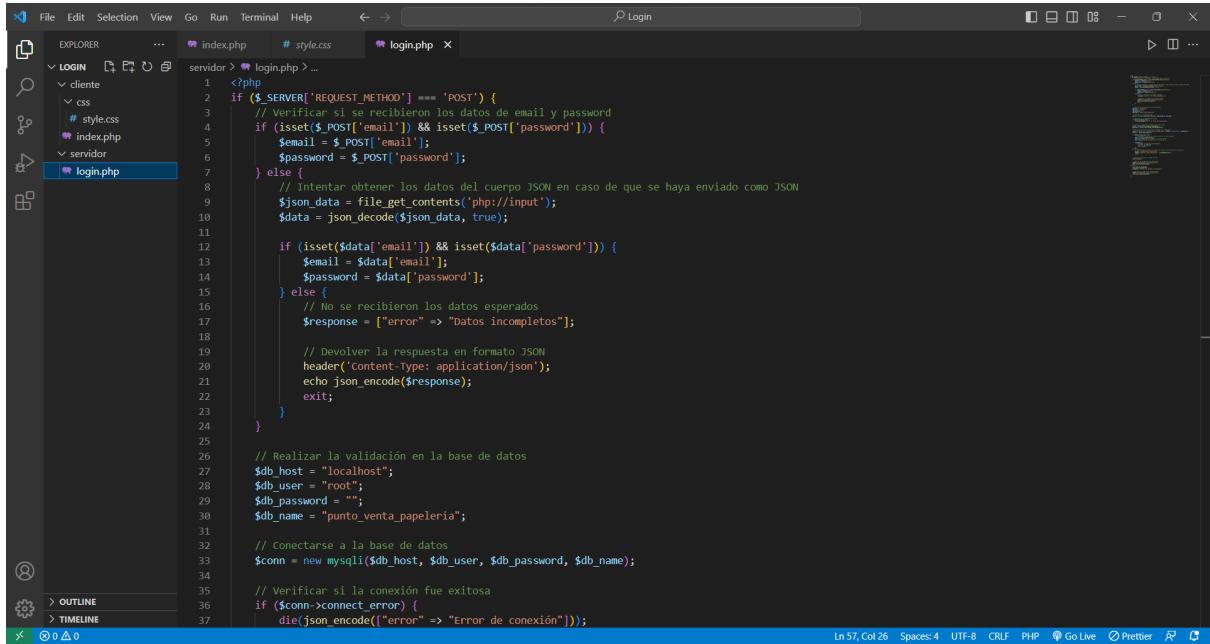




Código



```
index.php
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Inicio sesión</title>
8
9      <!-- Custom CSS file link -->
10     <link rel="stylesheet" href="css/style.css">
11
12 </head>
13
14 <div class="form-container">
15
16     <form action="../servidor/login.php" method="post"> <!-- Ajusta la ruta del formulario -->
17         <h3>Inicio sesión</h3>
18         <?php
19             if(isset($_GET['error'])){
20                 $error = $_GET['error'];
21                 echo '<span class="error-msg">'.$error.'</span>';
22             }
23         ?>
24         <input type="email" name="email" required placeholder="Ingresa tu correo">
25         <input type="password" name="password" required placeholder="Ingresa tu contraseña">
26         <input type="submit" name="submit" value="Iniciar sesión" class="form-btn">
27         <p>No tienes una cuenta? <a href="register_form.php">Regístrate ahora</a></p>
28     </form>
29
30 </div>
31
32 </body>
33 </html>
```

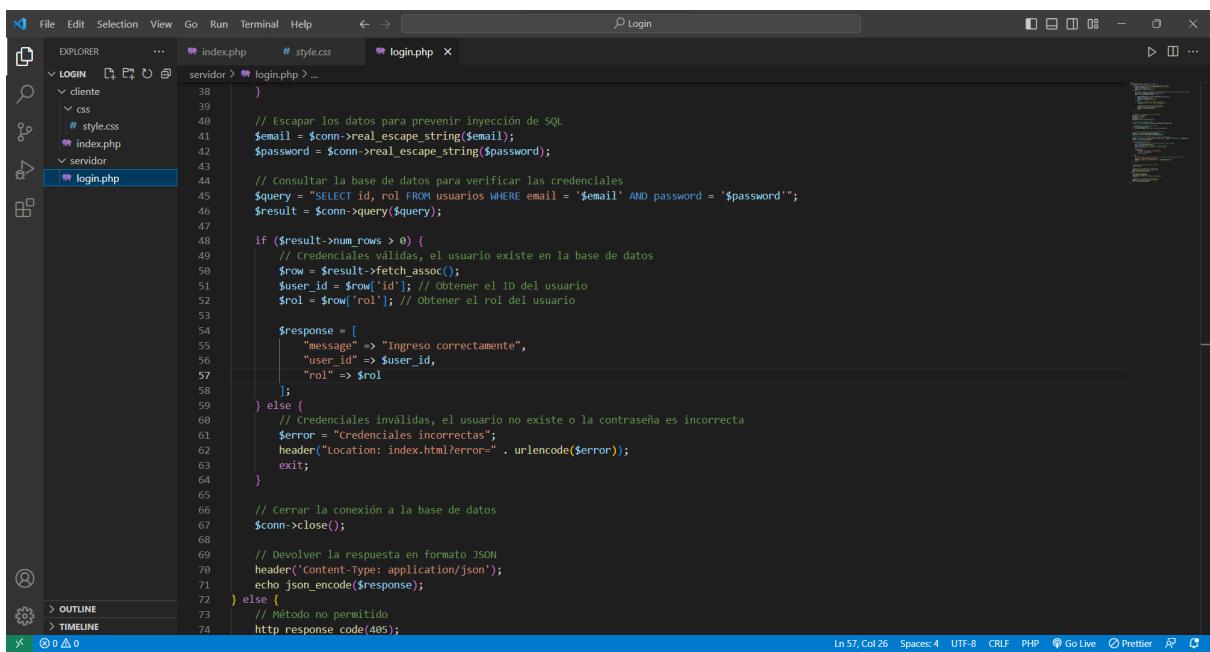


File Edit Selection View Go Run Terminal Help Login

EXPLORER ... index.php # style.css login.php

```
servidor > login.php > ...
1 <?php
2 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
3     // Verificar si se recibieron los datos de email y password
4     if (isset($_POST['email']) && isset($_POST['password'])) {
5         $email = $_POST['email'];
6         $password = $_POST['password'];
7     } else {
8         // Intentar obtener los datos del cuerpo JSON en caso de que se haya enviado como JSON
9         $json_data = file_get_contents('php://input');
10        $data = json_decode($json_data, true);
11
12        if (isset($data['email']) && isset($data['password'])) {
13            $email = $data['email'];
14            $password = $data['password'];
15        } else {
16            // No se recibieron los datos esperados
17            $response = ['error' => "Datos incompletos"];
18
19            // Devolver la respuesta en formato JSON
20            header('Content-type: application/json');
21            echo json_encode($response);
22            exit;
23        }
24    }
25
26    // Realizar la validación en la base de datos
27    $db_host = "localhost";
28    $db_user = "root";
29    $db_password = "";
30    $db_name = "punto_venta_papeleria";
31
32    // Conectarse a la base de datos
33    $conn = new mysqli($db_host, $db_user, $db_password, $db_name);
34
35    // Verificar si la conexión fue exitosa
36    if ($conn->connect_error) {
37        die(json_encode(['error' => "Error de conexión"]));
38    }
39
40    // Escapar los datos para prevenir inyección de SQL
41    $email = $conn->real_escape_string($email);
42    $password = $conn->real_escape_string($password);
43
44    // Consultar la base de datos para verificar las credenciales
45    $query = "SELECT id, rol FROM usuarios WHERE email = '$email' AND password = '$password'";
46    $result = $conn->query($query);
47
48    if ($result->num_rows > 0) {
49        // Credenciales válidas, el usuario existe en la base de datos
50        $row = $result->fetch_assoc();
51        $user_id = $row['id']; // Obtener el ID del usuario
52        $rol = $row['rol']; // Obtener el rol del usuario
53
54        $response = [
55            "message" => "Ingreso correctamente",
56            "user_id" => $user_id,
57            "rol" => $rol
58        ];
59    } else {
60        // Credenciales inválidas, el usuario no existe o la contraseña es incorrecta
61        $error = "Credenciales incorrectas";
62        header("location: index.html?error=" . urlencode($error));
63        exit;
64    }
65
66    // Cerrar la conexión a la base de datos
67    $conn->close();
68
69    // Devolver la respuesta en formato JSON
70    header('Content-type: application/json');
71    echo json_encode($response);
72} else {
73    // Método no permitido
74    http_response_code(405);
75}
```

Ln 57, Col 26 Spaces: 4 UTF-8 CRLF PHP Go Live Prettier

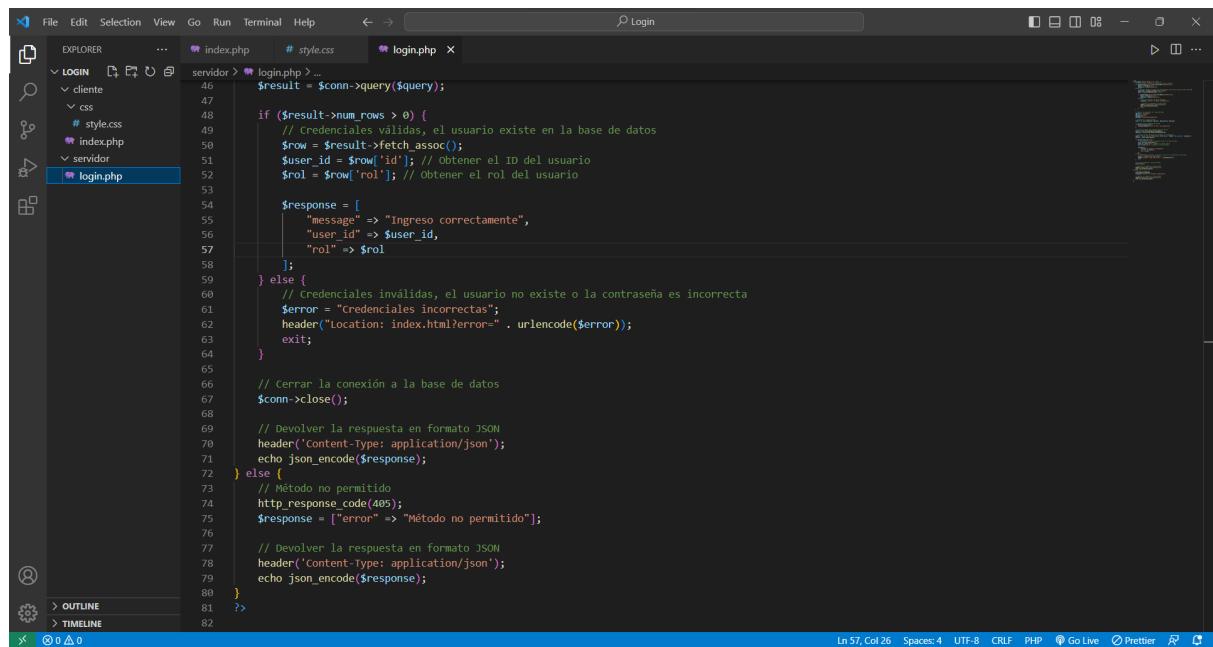


File Edit Selection View Go Run Terminal Help Login

EXPLORER ... index.php # style.css login.php

```
servidor > login.php > ...
38 }
39
40 // Escapar los datos para prevenir inyección de SQL
41 $email = $conn->real_escape_string($email);
42 $password = $conn->real_escape_string($password);
43
44 // Consultar la base de datos para verificar las credenciales
45 $query = "SELECT id, rol FROM usuarios WHERE email = '$email' AND password = '$password'";
46 $result = $conn->query($query);
47
48 if ($result->num_rows > 0) {
49     // Credenciales válidas, el usuario existe en la base de datos
50     $row = $result->fetch_assoc();
51     $user_id = $row['id']; // Obtener el ID del usuario
52     $rol = $row['rol']; // Obtener el rol del usuario
53
54     $response = [
55         "message" => "Ingreso correctamente",
56         "user_id" => $user_id,
57         "rol" => $rol
58     ];
59 } else {
60     // Credenciales inválidas, el usuario no existe o la contraseña es incorrecta
61     $error = "Credenciales incorrectas";
62     header("location: index.html?error=" . urlencode($error));
63     exit;
64 }
65
66 // Cerrar la conexión a la base de datos
67 $conn->close();
68
69 // Devolver la respuesta en formato JSON
70 header('Content-type: application/json');
71 echo json_encode($response);
72} else {
73    // Método no permitido
74    http_response_code(405);
75}
```

Ln 57, Col 26 Spaces: 4 UTF-8 CRLF PHP Go Live Prettier



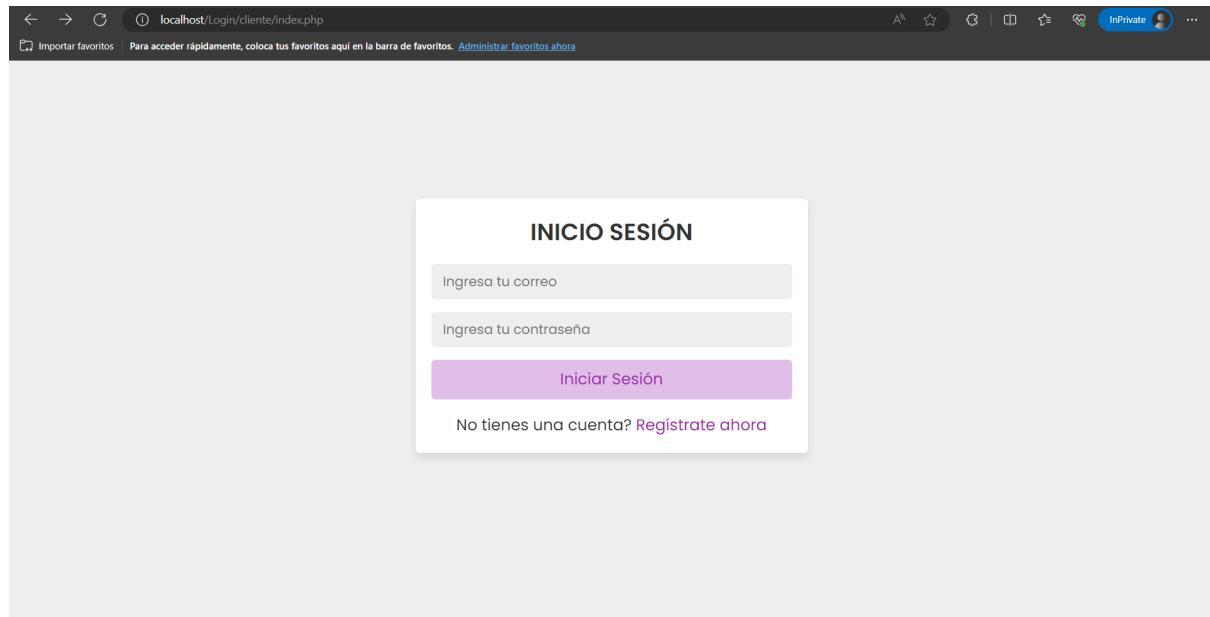
A screenshot of a code editor interface, likely Visual Studio Code, showing a PHP file named `login.php` in the `LOGIN` folder. The code is a login script that checks user credentials against a database and returns a JSON response. The editor includes a sidebar with file navigation, a central code editor with syntax highlighting, and a status bar at the bottom.

```
46     $result = $conn->query($query);
47
48     if ($result->num_rows > 0) {
49         // Credenciales válidas, el usuario existe en la base de datos
50         $row = $result->fetch_assoc();
51         $user_id = $row['id']; // Obtener el ID del usuario
52         $rol = $row['rol']; // Obtener el rol del usuario
53
54         $response = [
55             "message" -> "Ingreso correctamente",
56             "user_id" -> $user_id,
57             "rol" -> $rol
58         ];
59     } else {
56         // Credenciales inválidas, el usuario no existe o la contraseña es incorrecta
57         $error = "Credenciales incorrectas";
58         header("Location: index.html?error=" . urlencode($error));
59         exit;
60     }
61
62     // Cerrar la conexión a la base de datos
63     $conn->close();
64
65     // Devolver la respuesta en formato JSON
66     header('Content-Type: application/json');
67     echo json_encode($response);
68 } else {
69     // Método no permitido
70     http_response_code(405);
71     $response = ["error" -> "Método no permitido"];
72
73     // Devolver la respuesta en formato JSON
74     header('Content-type: application/json');
75     echo json_encode($response);
76 }
77
78
79
80
81
82
```

Ln 57, Col 26 Spaces: 4 UTF-8 CRLF PHP Go Live Prettier

Clientes mediante prototipo falso:

Iniciar sesión (Ángel Marco Polo De la Cruz Valdez)



Registro de usuarios (Felipe de Jesus Rello Yañez)

Registro de Usuarios

Nombre:

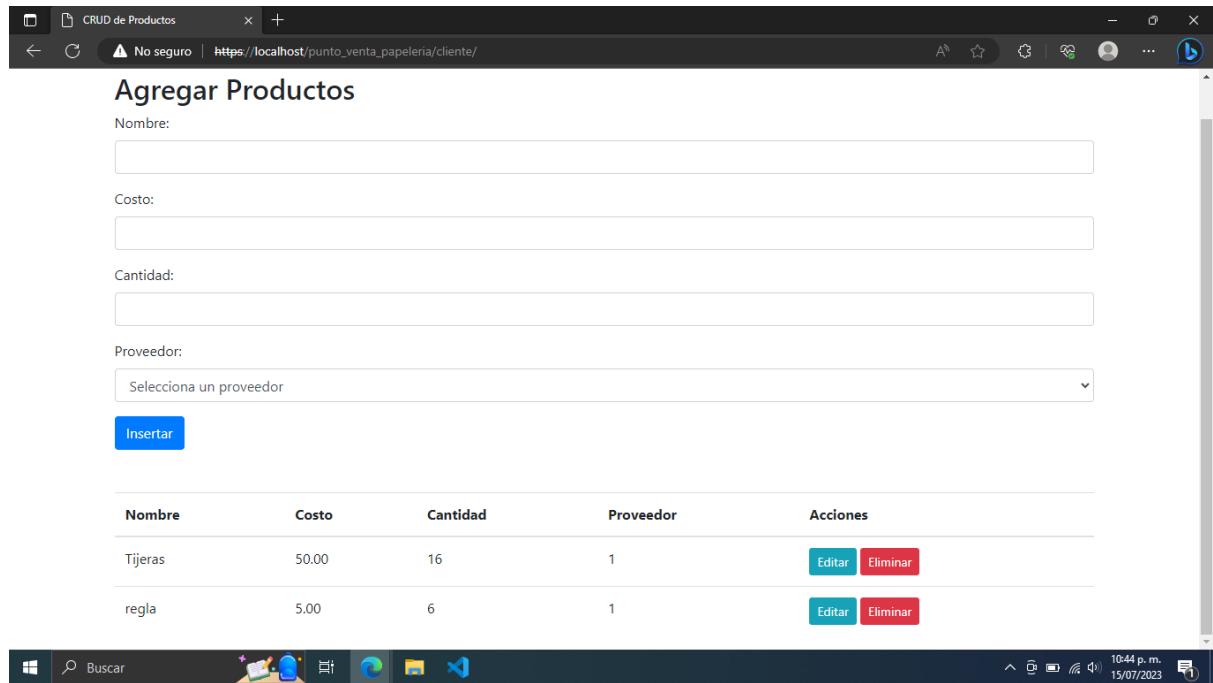
Contraseña:

Rol:

Registrar Usuario

ID	Nombre	Contraseña	Rol	Acciones
1	Usuario1	123456	Administrador	Editar Eliminar
2	Usuario2	abcdef	Usuario	Editar Eliminar

Registro de productos (Angela Benitez Hernandez)



Agregar Productos

Nombre:

Costo:

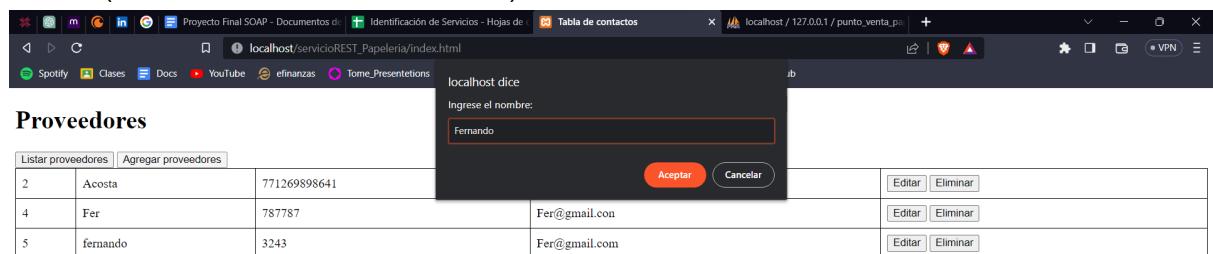
Cantidad:

Proveedor:

Insertar

Nombre	Costo	Cantidad	Proveedor	Acciones
Tijeras	50.00	16	1	Editar Eliminar
regla	5.00	6	1	Editar Eliminar

Ventas (Fernando Jesús Cruz Moreno)



Proveedores

localhost dice
Ingrese el nombre:

Aceptar **Cancelar**

2	Acosta	771269898641		Editar Eliminar
4	Fer	787787	Fer@gmail.com	Editar Eliminar
5	fernando	3243	Fer@gmail.com	Editar Eliminar

Proveedores (Christian Arturo Zuriaga Martinez)



localhost:8080/Pape/ventas.php

Aplicaciones material para videos Circuit design Fanta... #1 Solidity Tutorial...

Página de Ventas

Producto: Cantidad: Generar Venta

Tabla de Ventas

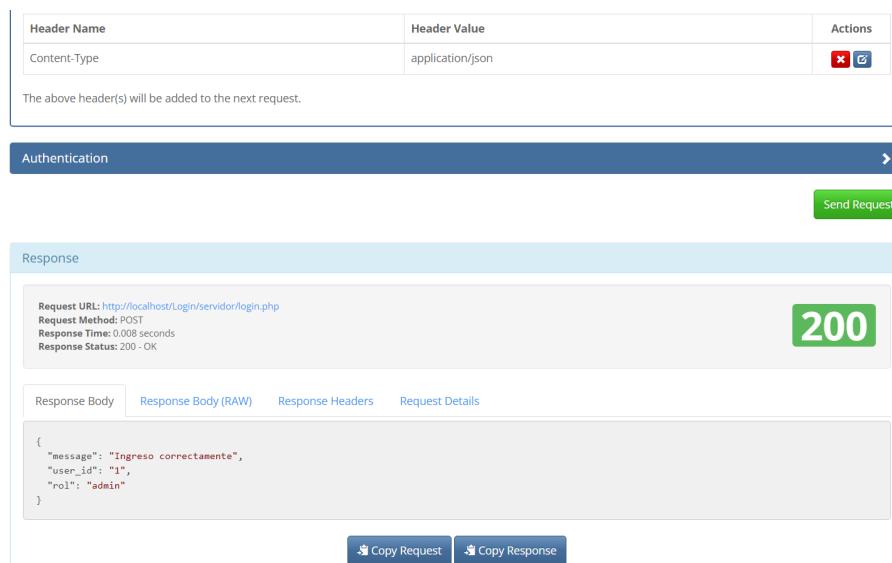
ID Venta	Fecha Venta	Total Venta
----------	-------------	-------------

14	2023-07-20 00:00:00	5.00
15	2023-07-20 00:00:00	50.00

Servicio web funcionales:

Servicio - Login

- Post



Header Name Header Value Actions

Content-Type	application/json		
--------------	------------------	--	--

The above header(s) will be added to the next request.

Authentication >

Send Request

Response

Request URL: <http://localhost/Login/servidor/login.php>
Request Method: POST
Response Time: 0.008 seconds
Response Status: 200 - OK

200

Response Body Response Body (RAW) Response Headers Request Details

```
{ "message": "Ingreso correctamente", "user_id": "1", "rol": "admin" }
```

Servicio - Registrar usuarios

- Get

The screenshot shows a REST API response interface. At the top right is a green button labeled "Send Request". Below it is a "Response" section with a "200" status indicator. The response body is a JSON array containing one user object:

```
[
  {
    "id": "1",
    "nombre": "Felipe",
    "password": "12345678",
    "rol": "admin"
  }
]
```

Below the response body are two buttons: "Copy Request" and "Copy Response".

- Post

The screenshot shows a "Request Settings" interface. The URL is set to "http://localhost/mi_servicio_rest/" and the method is "POST". The "Payload" field contains the following JSON object:

```
{
  "nombre": "Nombre del Usuario",
  "password": "Contraseña del Usuario",
  "rol": "Rol del Usuario"
}
```

The screenshot shows a "Response" section with a "200" status indicator. The response body is a JSON object with a single key "message":

```
{
  "message": "Usuario agregado correctamente"
}
```

- Put

Request Settings

URL: PUT

Payload:

```
{  
  "id": 1,  
  "nombre": "Nuevo Nombre del Usuario",  
  "password": "Nueva Contraseña del Usuario",  
  "rol": "Nuevo Rol del Usuario"  
}
```

Response

Request URL: http://localhost/mi_servicio_rest/
Request Method: PUT
Response Time: 0.024 seconds
Response Status: 200 - OK

200

Response Body Response Body (RAW) Response Preview Response Headers Request Details

```
{  
  "message": "Usuario actualizado correctamente"  
}
```

- Delete

Request Settings

URL: DELETE

Payload:

```
application/json
```

Response

Request URL: http://localhost/mi_servicio_rest/?id=1
Request Method: DELETE
Response Time: 0.026 seconds
Response Status: 200 - OK

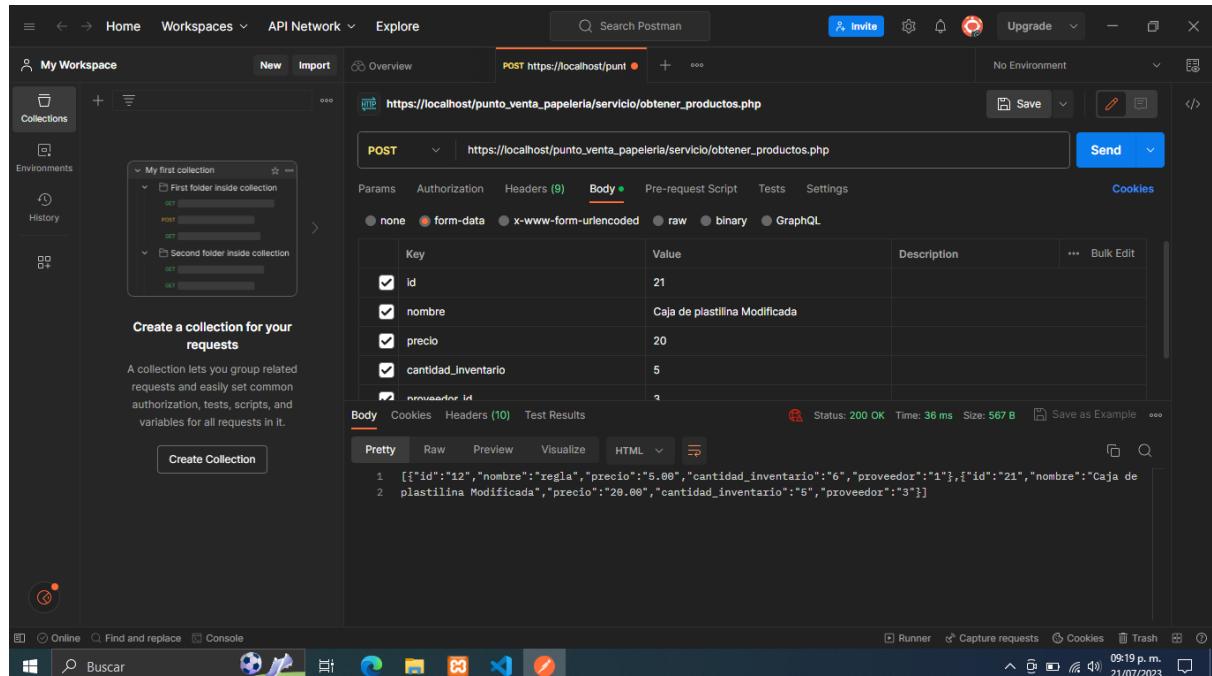
200

Response Body Response Body (RAW) Response Preview Response Headers Request Details

```
{  
  "message": "Usuario eliminado correctamente"  
}
```

Servicio - Registrar productos

- Get



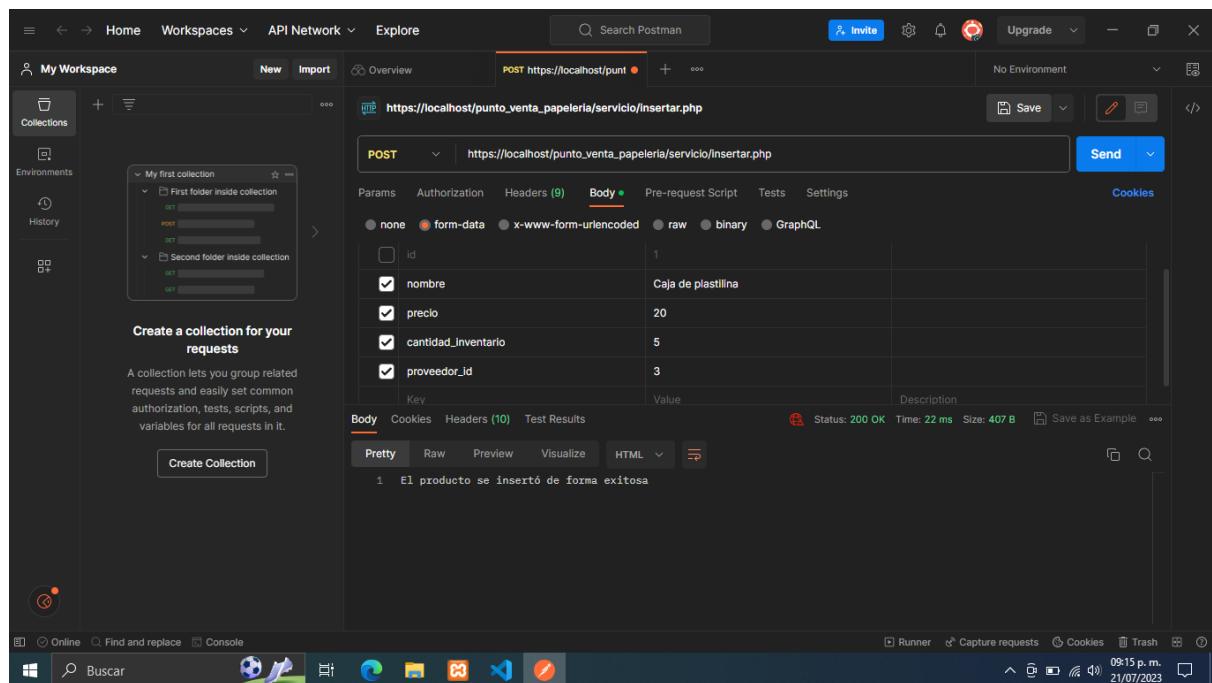
POST https://localhost/punto_venta_papelaria/servicio/obtener_productos.php

Key	Value	Description
id	21	
nombre	Caja de plastilina Modificada	
precio	20	
cantidad_inventario	5	

Body Cookies Headers (10) Test Results

1. [{"id": "12", "nombre": "regla", "precio": "5.00", "cantidad_inventario": "6", "proveedor": "1"}, {"id": "21", "nombre": "Caja de plastilina Modificada", "precio": "20.00", "cantidad_inventario": "5", "proveedor": "3"}]

- Post



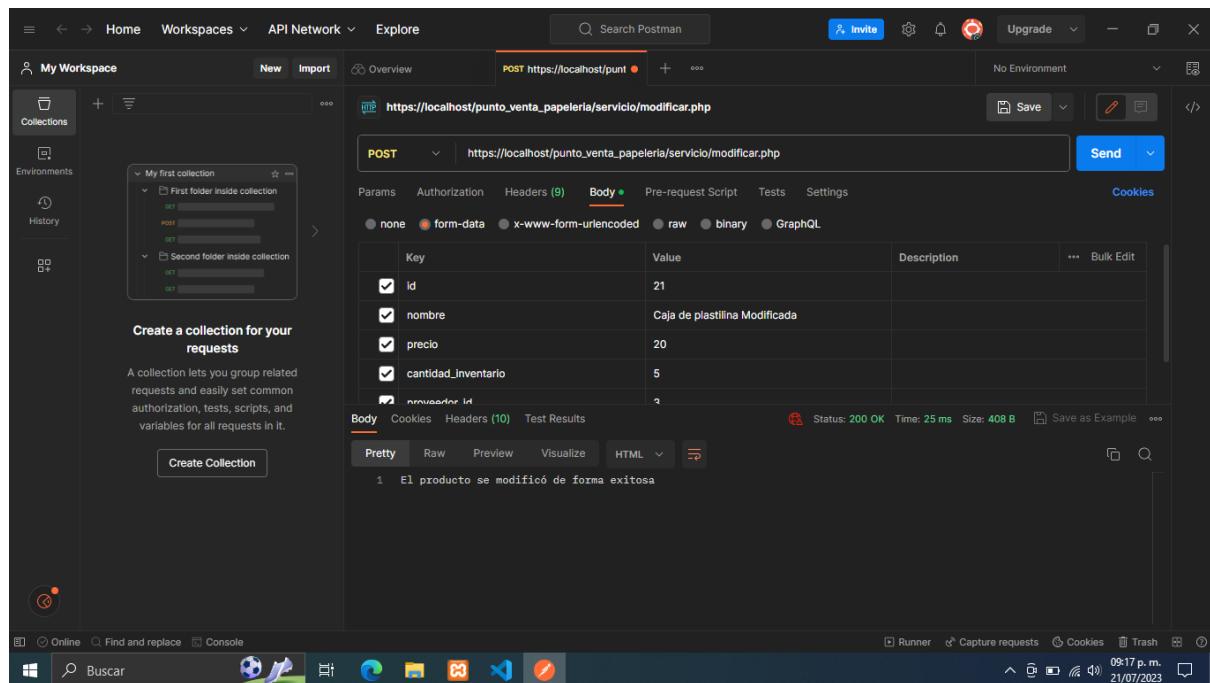
POST https://localhost/punto_venta_papelaria/servicio/insertar.php

Key	Value	Description
id	1	
nombre	Caja de plastilina	
precio	20	
cantidad_inventario	5	
proveedor_id	3	

Body Cookies Headers (10) Test Results

1. El producto se insertó de forma exitosa

- Put



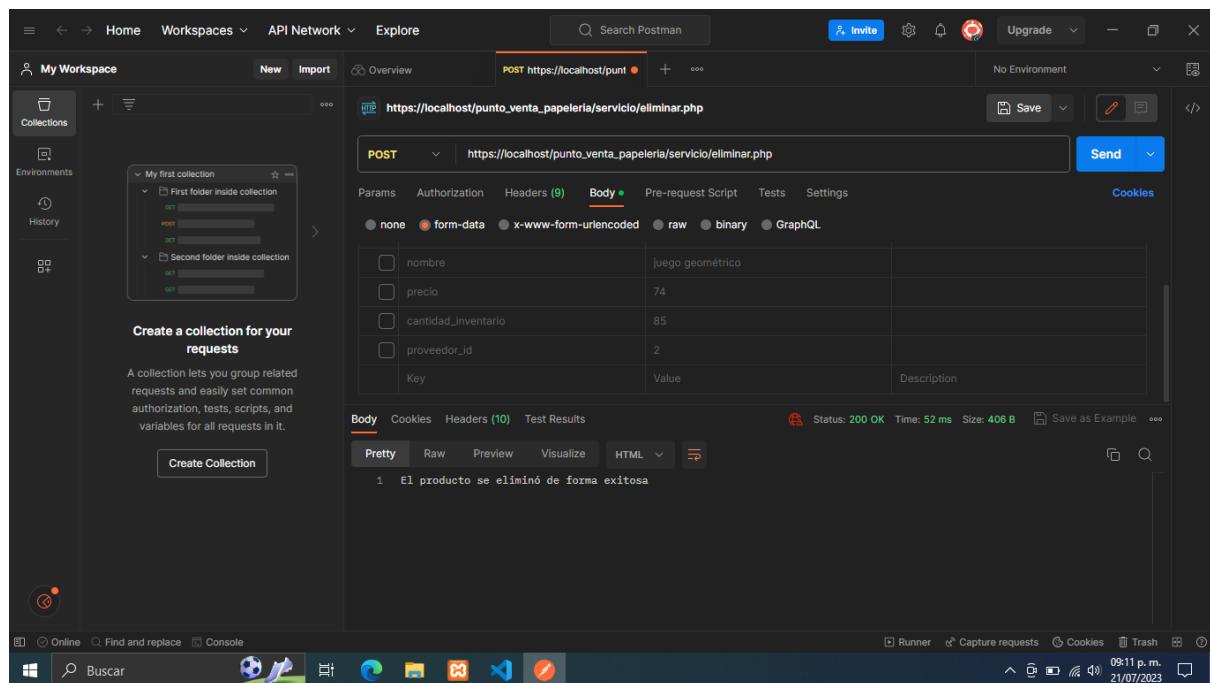
POST https://localhost/punto_venta_papelaria/servicio/modificar.php

Key	Value	Description	...
<input checked="" type="checkbox"/> id	21		
<input checked="" type="checkbox"/> nombre	Caja de plastilina Modificada		
<input checked="" type="checkbox"/> precio	20		
<input checked="" type="checkbox"/> cantidad_inventario	5		
<input checked="" type="checkbox"/> proveedor_id			

Body Cookies Headers (10) Test Results

1 El producto se modificó de forma exitosa.

- Delete



POST https://localhost/punto_venta_papelaria/servicio/eliminar.php

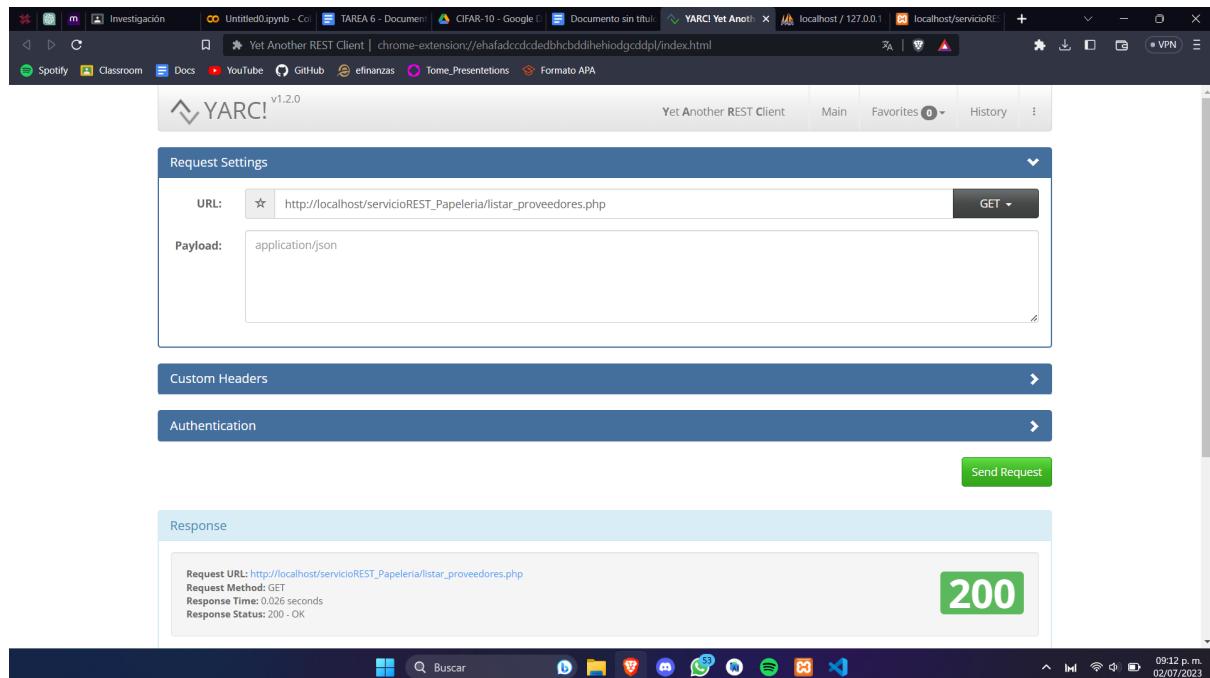
Key	Value	Description
<input type="checkbox"/> nombre	juego geométrico	
<input type="checkbox"/> precio	74	
<input type="checkbox"/> cantidad_inventario	85	
<input type="checkbox"/> proveedor_id	2	
Key	Value	Description

Body Cookies Headers (10) Test Results

1 El producto se eliminó de forma exitosa.

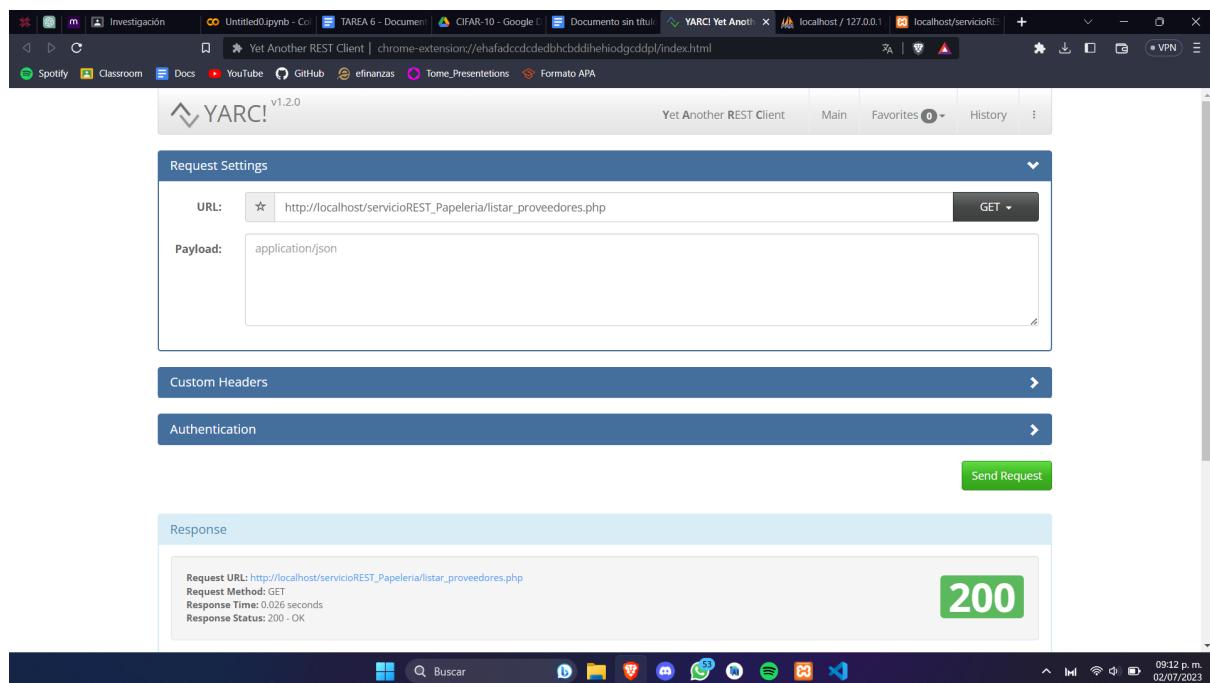
Servicio - Proveedores

- Get



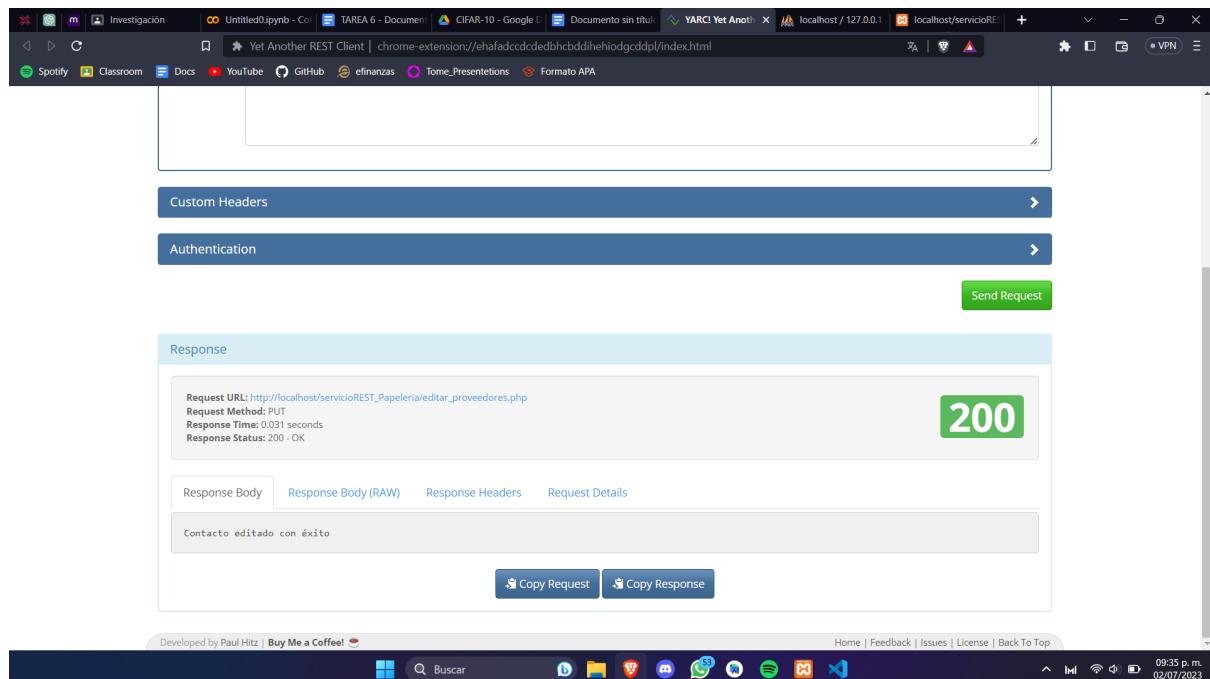
The screenshot shows the YARC! REST Client interface. In the 'Request Settings' section, the URL is set to `http://localhost/servicioREST_Papelaria/listar_proveedores.php` and the method is set to 'GET'. The 'Payload' field contains `application/json`. Below the request settings, there are sections for 'Custom Headers' and 'Authentication'. A green 'Send Request' button is located to the right of the request settings. The 'Response' section displays the following details: Request URL: `http://localhost/servicioREST_Papelaria/listar_proveedores.php`, Request Method: GET, Response Time: 0.026 seconds, and Response Status: 200 - OK. To the right of the status text is a large green '200' box. At the bottom of the screen, the Windows taskbar shows the date and time as 02/07/2023 09:12 p.m.

- Post



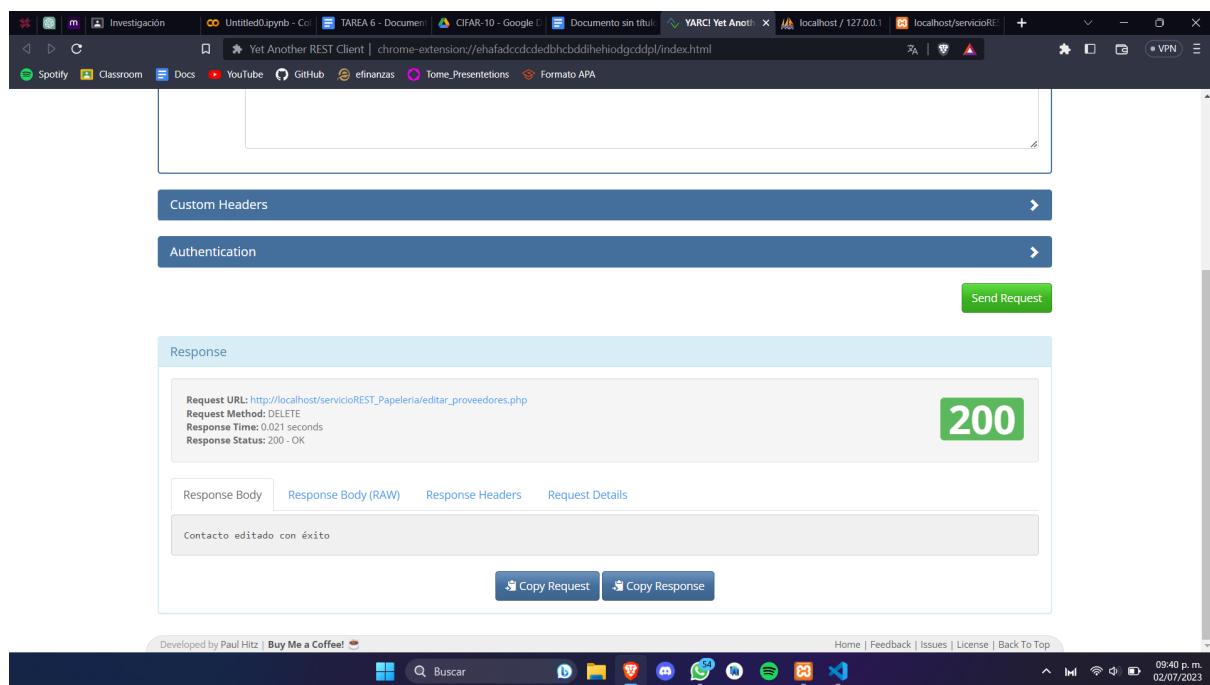
The screenshot shows the YARC! REST Client interface. In the 'Request Settings' section, the URL is set to `http://localhost/servicioREST_Papelaria/listar_proveedores.php` and the method is set to 'POST'. The 'Payload' field contains `application/json`. Below the request settings, there are sections for 'Custom Headers' and 'Authentication'. A green 'Send Request' button is located to the right of the request settings. The 'Response' section displays the following details: Request URL: `http://localhost/servicioREST_Papelaria/listar_proveedores.php`, Request Method: GET, Response Time: 0.026 seconds, and Response Status: 200 - OK. To the right of the status text is a large green '200' box. At the bottom of the screen, the Windows taskbar shows the date and time as 02/07/2023 09:12 p.m.

- Put



The screenshot shows the Yet Another REST Client interface. The request URL is `http://localhost/servicioREST_Papelaria/editar_proveedores.php`, the method is PUT, and the response status is 200 OK. The response body contains the message "Contacto editado con éxito".

- Delete



The screenshot shows the Yet Another REST Client interface. The request URL is `http://localhost/servicioREST_Papelaria/editar_proveedores.php`, the method is DELETE, and the response status is 200 OK. The response body contains the message "Contacto editado con éxito".

Servicio Ventas (Christian)

Interfaz

Página de Ventas

Producto:

Seleccióna un producto

Cantidad:

Tabla de Ventas

ID Venta	Fecha Venta	Total Venta	Acciones
1	2023-07-07 12:00:00	70.00	<input type="button" value="Eliminar"/>

Venta Registrada

Producto:

Caja Colores

Cantidad:

Tabla de Ventas

ID Venta	Fecha Venta	Total Venta	Acciones
1	2023-07-07 12:00:00	70.00	<input type="button" value="Eliminar"/>
60	2023-08-18 00:00:00	120.00	<input type="button" value="Eliminar"/>

Post

Código

```
16 // Procesar la generación de la venta
17 if (isset($_POST['generar_venta'])) {
18     $productoId = $_POST['producto'];
19     $cantidad = $_POST['cantidad'];
20
21     // Validar que se haya seleccionado un producto y una cantidad
22     if (!empty($productoId) && !empty($cantidad)) {
23         // Obtener la información del producto de la base de datos por su ID
24         $query = "SELECT * FROM productos WHERE id = $productoId";
25         $result = mysqli_query($con, $query);
26         $row = mysqli_fetch_assoc($result);
27
28         if ($row) {
29             $productoNombre = $row['nombre'];
30             $precio = $row['precio'];
31             $cantidadInventario = $row['cantidad_inventario'];
32
33             if ($cantidad <= $cantidadInventario) {
34                 $subtotal = $precio * $cantidad;
35
36                 // Insertar los datos en la tabla ventas
37                 $fechaVenta = date('Y-m-d'); // Fecha actual
38                 $query = "INSERT INTO ventas (fecha_venta, total_venta) VALUES ('$fechaVenta', $subtotal)";
39                 mysqli_query($con, $query);
40
41                 // Obtener el ID de la última venta insertada
42                 $idVenta = mysqli_insert_id($con);
43
44                 // Actualizar la cantidad_inventario en la tabla productos
45                 $nuevaCantidadInventario = $cantidadInventario - $cantidad;
46                 $query = "UPDATE productos SET cantidad_inventario = $nuevaCantidadInventario WHERE id = $productoId";
47                 mysqli_query($con, $query);
48             }
49         }
50     }
51 }
```

Funcionamiento

Response

Request URL: <http://localhost:8080/PAPE/ventas.php>
Request Method: POST
Response Time: 0.026 seconds
Response Status: 200 - OK

200

Response Body Response Body (RAW) Response Preview Response Headers Request Details

```
<!-- HTML para la página de ventas -->
<!DOCTYPE html>
<html>
<head>
    <title>Página de Ventas</title>
</head>
<body>
    <h1>Página de Ventas</h1>

```

Mostrando filas 0 - 2 (total de 3, La consulta tardó 0,0201 segundos.)

SELECT * FROM `detalles_ventas`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: ↑

Opciones extra

	id	id_venta	id_producto	cantidad_vendida	precio_unitario
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	1	14	1	1	5.00
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	2	15	1	10	5.00
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	3	16	1	13	5.00

↑ Seleccionar todo Para los elementos que están marcados: Editar Copiar Borrar E

Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: ↑

Operaciones sobre los resultados de la consulta

Put

Response

Request URL: <http://localhost:8080/PAPE/ventas.php>
 Request Method: PUT
 Response Time: 0.028 seconds
 Response Status: 200 - OK

200

Response Body | Response Body (RAW) | Response Preview | Response Headers | Request Details

```

<!-- HTML para la página de ventas -->
<!DOCTYPE html>
<html>
<head>
  <title>Página de Ventas</title>
</head>
<body>
  <h1>Página de Ventas</h1>

  <form method="post" action="">
    <label for="producto">Producto:</label>
    <select name="producto" id="producto" required>
      <option value="">Selecciona un producto</option>
      <option value="1">Plumas</option>
      <option value="2">Cuadernos</option>
  
```

phpMyAdmin

Servidor: 127.0.0.1 » Base de datos: punto_venta_papeleria » Tabla: productos

Examinar Estructura SQL Buscar Insertar Exportar Importar

Mostrando filas 0 - 1 (total de 2, La consulta tardó 0,0005 segundos.)

SELECT * FROM `productos`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla Ordenar según la

Opciones extra

← → id nombre precio cantidad_inventario proveedor_id

	Editar	Copiar	Borrar	id	nombre	precio	cantidad_inventario	proveedor_id
<input type="checkbox"/>				1	Plumas	5.00	6	123
<input type="checkbox"/>				2	Cuadernos	20.00	40	123

↑ Seleccionar todo Para los elementos que están marcados: Editar Copiar Borrar

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla Ordenar según la

Operaciones sobre los resultados de la consulta

Get

Send Request

Response

Request URL: <http://localhost:8080/PAPE/ventas.php>
 Request Method: GET
 Response Time: 0.075 seconds
 Response Status: 200 - OK

200

Response Body Response Body (RAW) Response Preview Response Headers Request Details

```

<!-- HTML para la página de ventas -->
<!DOCTYPE html>
<html>
<head>
  <title>Página de Ventas</title>
</head>
<body>
  <h1>Página de Ventas</h1>

  <form method="post" action="">
    <label for="producto">Producto:</label>
    <select name="producto" id="producto" required>
      <option value="">Selecciona un producto</option>
    </select>
  </form>

```

Delete

Response

Request URL: <http://localhost:8080/PAPE/ventas.php>
Request Method: DELETE
Response Time: 0.027 seconds
Response Status: 200 - OK

200

Response Body Response Body (RAW) Response Preview Response Headers Request Details

```
<!-- HTML para la página de ventas -->
<!DOCTYPE html>
<html>
<head>
<title>Página de Ventas</title>
</head>
<body>
<h1>Página de Ventas</h1>

<form method="post" action="">
<label for="producto">Producto:</label>
<select name="producto" id="producto" required>
<option value="">Selecciona un producto</option>
<option value="1">Plumas</option>
<option value="2">Cuadernos</option>
</select>
//ehafadcccdedbhcbddihediogcdpl/index.html tidad:</label>
```

Time recording log:

Ángel Marco Polo De la Cruz Valdez

Fecha	Inicio	Fin	Tiempo interrupción	Tiempo efectivo	Fase	Comentario
21/06/2023	8:00 a.m.	10:00 a.m.	30 minutos	90 minutos	Planeación	Se realizaron historias de usuario.
22/06/2023	8:00 a.m.	9:00 a.m.	15 minutos	45 minutos	Planeación	Se identificaron servicios y funcionalidades.
26/06/2023	10:00 a.m.	11:00 p.m.	10 minutos	50 minutos	Design	Se realizaron casos de prueba del servicio Inicio de sesión.
28/06/2023	8:00 a.m.	9:00 a.m.	20 minutos	40 minutos	Codificación	Se realizaron servicios Post para el módulo Inicio de sesión.
28/06/2023	9:30 a.m.	10:00 a.m.	10 minutos	20 minutos	Test	Se realizó un test a los servicios para comprobar su funcionalidad.
08/07/2023	8:40 a.m.	9:40 a.m.	10 minutos	50 minutos	Codificación	Se hicieron correcciones y mejoras en el módulo Inicio de sesión.
09/07/2023	12:00 a.m.	12:30 a.m.	3 minutos	27 minutos	Design	Se realizó el llenado del formato Time Recording Log recopilando todos los avances hasta el momento.

21/07/2023	8:04 p.m.	9:35 p.m.	25 minutos	66 minutos	Codificación	Se realizó un prototipo con funcionalidad de la interfaz para el Inicio de sesión.
09/08/2023	11:50 a.m.	2:50 p.m.	20 minutos	160 minutos	Codificación	Se realizó un prototipo con funcionalidad del módulo de Reportes PDF.
15/08/2023	8:30 p.m.	10:30 p.m.	10 minutos	110 minutos	Codificación	Se separó la parte del servidor y el cliente dentro del módulo del Login.
15/08/2023	10:30 p.m.	11:30 p.m.	15 minutos	45 minutos	Test	Se hizo test a los cambios del cliente y servidor en el Login.

Felipe de Jesus Rello Yañez

Fecha	Inicio	Fin	Tiempo interrupción	Tiempo efectivo	Fase	Comentario
21/06/2023	8:00 am	10:00 am	30 minutos	90 minutos	Planeación	Realicé las historias de usuario.
22/06/2023	8:00 am	9:00 am	15 minutos	45 minutos	Planeación	Se identificaron servicios y funcionalidades.
26/06/2023	10:00 am	11:00 pm	10 minutos	50 minutos	Design	Se realizaron casos de prueba del servicio de

						registro de usuarios.
28/06/2023	8:00 am	9:00 am	20 minutos	40 minutos	Codificación	Se realizaron los servicios POST, GET, PUT y DELETE para el módulo de registro de usuarios
28/06/2023	9:30am	10:00 am	10 minutos	20 minutos	Test	Se realizó un test a los servicios para comprobar su funcionalidad
09/07/2023	10:30 pm	11:00 pm	0 minutos	30 minutos	Design	Se realizó el formato Time Recording Log para registrar el avance del desarrollo del programa
09/07/2023	12:00 am	12:30 am	3 minutos	27 minutos	Design	Se realizó el llenado del formato Time Recording Log recopilando todos los avances hasta el momento
21/07/2023	10:00 pm	10:30 pm	5 minutos	25 minutos	Codificación	Se realizó un prototipo no funcional de la interfaz para el registro de usuarios
15/08/2023	9:00 p.m	11:00 p.m	15 minutos	105 minutos	Codificación	Se realizaron correcciones a la parte de registrar usuario para separar el cliente del servidor
16/08/2023	12:00 p.m	12:30 a.m	5 minutos	25 minutos	Test	Se comprobó el funcionamiento del módulo para los métodos GET, POST, PUT y DELETE

Angela Benitez Hernandez

Fecha	Inicio	Fin	Tiempo interrupción	Tiempo efectivo	Fase	Comentario
21/06/2023	8:00am	10:00am	20 minutos	100 minutos	Design	Se realizaron las historias de usuario.
22/06/2023	7:30am	9:00am	20 minutos	70 minutos	Design	Se identificaron servicios, funcionalidades y el diseño de la base de datos en MySql.
26/06/2023	10:20 am	10:55 am	0 minutos	35 minutos	Design	Se realizaron casos de prueba del módulo 3: Agregar productos.
28/06/2023	8:20 am	11:00 am	13 minutos	87 minutos	Design	Se realizaron los servicios POST, PUT y DELETE para el módulo 3: Añadir productos.
29/06/2023	7:20 am	8:50 am	20 minutos	70 minutos	Code	Se realizó los servicios GET Productos y GET proveedores para el módulo 3: Añadir productos
30/06/2023	21:40 pm	22:30 pm	15 minutos	45 minutos	Test	Se realizó el consumo de los servicios del módulo 3(Añadir Productos) por medio de PostMan.
03/07/2023	10:25am	10:55 am	0 minutos	30 minutos	Design	Se realizó el formato Time Recording Log
05/07/2023	8:30 am	9:50 am	20 minutos	60 minutos	Design	Creación del Cliente para el

						servicio
06/07/2023	7:30 am	8:55 am	20 minutos	65 minutos	Design	Creación del Cliente para el servicio
13/07/2023	7:20 am	8:55 am	20 minutos	75 minutos	Design	Seguimiento de la creación del cliente y llenado del formato Time Recording Log desde 21/06/2023 hasta 13/07/2023

Fecha	Inicio	Fin	Tiempo interrupción	Tiempo efectivo	Fase	Comentario
21/06/2023	8:00am	10:00am	20 minutos	100 minutos	Design	Se realizaron las historias de usuario.
22/06/2023	8:00am	9:00am	7 minutos	53 minutos	Design	Se identificaron servicios, funcionalidades y el diseño de la base de datos en MySql.
26/06/2023	10:00am	11:00pm	13 minutos	47 minutos	Design	Se realizaron casos de prueba del módulo 6: Control de proveedores.
28/06/2023	8:00am	9:30am	0 minutos	90 minutos	Code	Se realizaron los servicios POST para el módulo 6: Control de proveedores.
28/06/2023	9:30am	10:00am	0 minutos	30 minutos	Test	Se realizó el consumo de los servicios del módulo 6(Control de proveedores) por medio de Yet Another REST Client.
03/07/2023	10:35am	11:00am	8 minutos	17 minutos	Design	Se realizó el formato Time Recording Log
05/07/2023	8:20am	9:35am	16 minutos	59 minutos	Design	Se hizo el llenado del formato Time Recording Log desde 21/06/2023 hasta 05/07/2023
06/07/2023	8:00am	9:30am	20 minutos	70 minutos	Code	Se realizó los servicios GET para el modulo 6: Control de proveedores.
12/07/2023	9:30am	10:00am	0 minutos	30 minutos	Test	Se realizó el consumo de los servicios del modulo 6(Control de proveedores) por medio de Yet Another REST Client.

13/07/2023	8:00am	9:30am	10 minutos	80 minutos	Code	Se realizo los servicios PUT para el modulo 6: Control de proveedores.
19/07/2023	9:30am	10:00am	0 minutos	30 minutos	Test	Se realizo el consumo de los servicios del modulo 6(Control de proveedores) por medio de Yet Another REST Client.
20/07/2023	8:00am	9:30am	0 minutos	90 minutos	Code	Se realizo los servicios DELETE para el modulo 6: Control de proveedores.
26/07/2023	9:30am	10:00am	0 minutos	30 minutos	Test	Se realizo el consumo de los servicios del modulo 6(Control de proveedores) por medio de Yet Another REST Client.
09/08/2023	8:00am	9:30am	0 minutos	90 minutos	Design	Se realizo el diseño de la interfaz del cliente del Modulo 6(Control de proveedores).
10/08/2023	7:00am	8:30am	0 minutos	90 minutos	Design	Se realizo el diseño de la interfaz de menu principal del cliente.
16/08/2023	8:00am	9:30am	0 minutos	90 minutos	Code	Se programo la interfaz del cliente del Modulo 6(Control de proveedores).
16/08/2023	8:00pm	9:30pm	0 minutos	90 minutos	Code	Se programo la interfaz de menu principal del cliente.
16/08/2023	9:30pm	11:30pm	0 minutos	120 minutos	Code	Se unieron todos los modulos funcionales.
16/08/2023	12:00am	1:00am	0 minutos	60 minutos	Test	Se realizaron pruebas de funcionalidad a todos los modulos.

Christian Arturo Zuriaga Martinez

Fecha	Inicio	Fin	Tiempo interrupción	Tiempo efectivo	Fase	Comentario
21/06/2023	8:00am	10:00am	20 minutos	100 minutos	Design	Se realizaron las historias de usuario para el servicio ventas.
22/06/2023	8:00am	9:00am	7 minutos	53 minutos	Design	Se identificaron servicios, funcionalidades y el diseño de la base de datos en MySql.
26/06/2023	10:00am	11:00pm	13 minutos	47 minutos	Design	Se realizaron casos de prueba del módulo 5.
28/06/2023	8:00am	9:30am	0 minutos	120 minutos	Code	Se realizaron los servicios POST, GET, PUT y DELETE para el módulo 5..
28/06/2023	9:30am	10:00am	0 minutos	30 minutos	Test	Se realizó el consumo de los servicios del módulo 5 por medio de Yet Another REST Client.
03/07/2023	10:30am	11:00am	8 minutos	17 minutos	Design	Se realizó el formato Time Recording Log
05/07/2023	8:20am	9:35am	16 minutos	59 minutos	Design	Se hizo el llenado del formato Time Recording Log.

Tiempo total

Nombre	Tiempo total individual
Christian Arturo Zuriaga Martinez	7,1
Fernando Jesús Cruz Moreno	21,1
Angela Benitez Hernandez	10,61666667
Felipe de Jesus Rello Yañez	7.6166666
Ángel Marco Polo De la Cruz Valdez	11,7166667
Total de horas	58,14999997

Autoevaluación

Ángel Marco Polo De la Cruz Valdez

Aspecto de Autoevaluación	Excelente (5)	Muy Bueno (4)	Bueno (3)	Satisfactorio (2)	Puede Mejorar (1)
Trabajo en Equipo	X				
Adopción de Herramientas y Tecnologías	X				
Eficiencia Organizativa		X			
Comunicación Clara		X			
Calidad en el desempeño de mi rol	X				
SUMA TOTAL: 23/25 = 9.2					

Felipe de Jesus Rello Yañez

Aspecto de Autoevaluación	Excelente (5)	Muy Bueno (4)	Bueno (3)	Satisfactorio (2)	Puede Mejorar (1)
Trabajo en Equipo	X				
Adopción de Herramientas y Tecnologías		X			
Eficiencia Organizativa	X				
Comunicación Clara	X				
Calidad en el desempeño de mi rol			X		
SUMA TOTAL: 22/25 = 8.8					

Angela Benitez Hernandez

Aspecto de Autoevaluación	Excelente (5)	Muy Bueno (4)	Bueno (3)	Satisfactorio (2)	Puede Mejorar (1)
Trabajo en Equipo	X				
Adopción de Herramientas y Tecnologías	X				
Eficiencia Organizativa		X			
Comunicación Clara		X			
Calidad en el desempeño de mi rol	X				
SUMA TOTAL: 23/25 = 9.2					

Fernando Jesús Cruz Moreno

Aspecto de Autoevaluación	Excelente (5)	Muy Bueno (4)	Bueno (3)	Satisfactorio (2)	Puede Mejorar (1)
Trabajo en Equipo	X				
Adopción de Herramientas y Tecnologías	X				
Eficiencia Organizativa	X				
Comunicación Clara	X				
Calidad en el desempeño de mi rol	X				
SUMA TOTAL: /25 = 10					

Christian Arturo Zuriaga Martinez

Aspecto de Autoevaluación	Excelente (5)	Muy Bueno (4)	Bueno (3)	Satisfactorio (2)	Puede Mejorar (1)
Trabajo en Equipo		x			
Adopción de Herramientas y Tecnologías		x			
Eficiencia Organizativa		x			
Comunicación Clara		x			
Calidad en el desempeño de mi rol		x			
SUMA TOTAL: /25 = 8					