

PROGRAMACIÓN DE SERVICIOS WEB

UNIDAD 5

1 SERVICIOS WEB

- ▶ Un **servicio web** es un conjunto de protocolos y estándares que permiten la comunicación entre aplicaciones a través de una red (generalmente Internet). Su propósito principal es permitir que diferentes sistemas, desarrollados en lenguajes o plataformas distintas, puedan interactuar y compartir datos. Se alojan en un servidor y son utilizados por clientes.
- ▶ Características:
 - ▶ **Interoperabilidad:** Pueden comunicarse entre aplicaciones escritas en diferentes lenguajes o plataformas.
 - ▶ **Uso de protocolos estándar:** Generalmente utilizan HTTP, SOAP, o REST.
 - ▶ **Estandarización:** Siguen formatos comunes como XML, JSON o YAML para intercambiar datos.
 - ▶ **Modularidad:** Permiten acceder a funciones o recursos específicos sin necesidad de exponer todo el sistema.
- ▶ Al utilizar servicios web, el servidor ofrece un punto de acceso a la información que quiere compartir (endpoint). De esta forma controla y facilita el acceso a la misma por parte de otras aplicaciones.
- ▶ Los clientes del servicio, por su parte, no necesitan conocer la estructura interna de almacenamiento. En lugar de tener que programar un mecanismo para localizar la información, tienen un punto de acceso directo a lo que les interesa.
- ▶ Existen diversas formas de implementar servicios web: SOAP, API REST, etc...
 - ▶ SOAP (Simple Object Access Protocol):
 - ▶ Basado en XML.
 - ▶ Protocolo más estricto con una estructura definida.
 - ▶ Requiere WSDL (Web Services Description Language) para describir el servicio.
 - ▶ Más utilizado en sistemas empresariales o donde se necesita seguridad avanzada.
 - ▶ REST (Representational State Transfer)
 - ▶ Basado en HTTP y principios arquitectónicos.
 - ▶ Usa métodos HTTP (GET, POST, PUT, DELETE).
 - ▶ Más ligero y fácil de implementar que SOAP.
 - ▶ Los datos se suelen intercambiar en JSON o XML.
 - ▶ Ideal para APIs públicas y aplicaciones modernas.

2 API REST

- ▶ Una API (Application Programming Interface) es un conjunto de funciones y procedimientos que conectan aplicaciones para que puedan intercambiar información entre sí de forma estándar.
- ▶ Una de las características fundamentales de las API es que son **Stateless**, lo que quiere decir que las peticiones se hacen y desaparecen, no hay usuarios logueados ni datos que se quedan almacenados.
- ▶ La información que devuelve la API al cliente se puede entregar a un cliente en prácticamente cualquier formato, por ejemplo, JavaScript Object Notation (JSON), HTML, XML, Python, PHP o texto sin formato. JSON es popular porque es legible tanto por personas como por máquinas, y es independiente del lenguaje de programación.
- ▶ Cada acción de una API REST se denomina **ENDPOINT** y usar uno de los métodos estándar de HTTP
 - ▶ GET (obtener datos)
 - ▶ POST (crear datos)
 - ▶ PUT (actualizar datos)
 - ▶ DELETE (eliminar datos).

Método	Endpoint	Descripción
GET	/api/users	Obtiene una lista de usuarios.
GET	/api/users/1	Obtiene los datos del usuario con ID 1.
POST	/api/users	Crea un nuevo usuario.
PUT	/api/users/1	Actualiza los datos del usuario con ID 1.
DELETE	/api/users/1	Elimina al usuario con ID 1.

3 ESTADOS Y RESPUESTAS

- ▶ En peticiones GET, se devuelven objetos o listas de objetos. Laravel los transforma a JSON de forma automática, aunque se pueden definir *recursos* para personalizar la transformación. El código de estado de estas respuestas es 200.
- ▶ Se pueden personalizar las respuestas y los códigos de estado, aquí tienes ejemplos:

```
return response()->json('Error al crear la nota',500);  
return response()->noContent(); //204  
return abort(401,'Error ...'); //Forbidden  
return abort(403);  
return abort(404);  
return abort(500,'Error ...'); //Error  
  
//Devolver más de un dato  
return response()->json(['mensaje' => 'Tarea Creada', 'tarea' => $tarea], 201);
```

- Listing and getting resources: 200 (OK).
- Creating resources: 201 (Created).
- Updating resources: 200 (OK).
- Deleting resources: 204 (No Content).
- Need to be authenticated to access resources: 401 (Forbidden).
- Unauthorized access to resources: 403 (Unauthorized).
- Missing resources: 404 (Not Found).
- Something went wrong: 500 (Internal Server Error).

4 RECURSOS DE API

- ▶ Laravel transforma de forma automática un Modelo a JSON como respuesta de la api.
- ▶ Los recursos permiten personalizar la transformación de un Modelo a JSON.
- ▶ Se almacenan en [app/Http/Resources](#).

▶ Creación

- ▶ Recurso individual. Transforma un sólo objeto del modelo: `php artisan make:resource UserResource`
- ▶ Recurso colectivo. Transformación una colección de objetos del modelo: `php artisan make:resource User --collection`

▶ Uso

- ▶ Individual: En método show. `return new UserResource(User::findOrFail($id));`
- ▶ Colección: En método index. `return UserResource::collection(User::all());`

▶ Código de recursos:

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'email' => $this->email,
        'posts' => PostResource::collection($this->posts),
        'created_at' => $this->created_at,
        'updated_at' => $this->updated_at,
    ];
}
```

Individual

```
public function toArray(Request $request): array
{
    return [
        'data' => $this->collection,
        'links' => [
            'self' => 'link-value',
        ],
    ];
}
```

Colectivo

No es necesario crear y definir los dos tipos de recursos.

Vamos a definir solamente los individuales cuando sea necesario, por ejemplo, si no queremos que el json tenga todos los atributos del modelo o cuando haya que modificarlos.

Es posible incluir recursos de objetos relacionados.

5 CLIENTES API REST

► Frontend

- Axios
- Ajax

► Backend

► Thunder Client: Extensión de Visual Studio Code

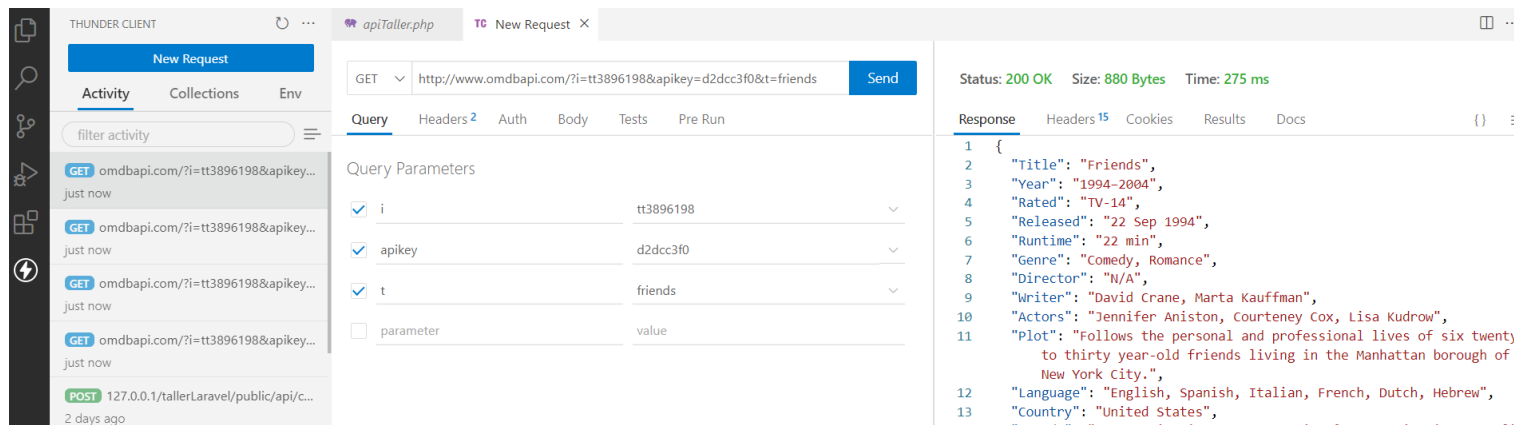
► Postman

► SpaceX API: Información sobre lanzamientos, cohetes y misiones de SpaceX.Base

- <https://api.spacexdata.com/v4>

► EndPoints:

- /launches (lanzamientos)
- /rockets (cohetes)
- /crew (tripulación)



6 AXIOS

- ▶ **Axios** es una librería de JavaScript que se utiliza para realizar solicitudes HTTP desde el html o Node.js de forma sencilla. Axios convierte automáticamente los datos enviados y recibidos a/desde JSON.

- ▶ Usaremos Axios, haciendo referencia a su CDN

```
<script  
src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
```

- ▶ Puedes descargar la librería

npm install axios

- ▶ En Axios, los estados o códigos de estado HTTP se manejan a través de la respuesta que devuelve la solicitud. Puedes acceder al código de estado en la propiedad status del objeto de respuesta (**response.status o error.response.status**).
- ▶ Si la respuesta tiene un código de estado **2xx (éxito)**, se ejecuta el bloque **then**. Si ocurre un error (código de estado **4xx o 5xx**), se ejecuta el bloque **catch**.

- ▶ Petición de servicio:

```
axios.[get/post/put/delete](apiUrl, [datos])  
  
//then: Códigos de estados 2XX  
  
.then(response => {  
    const datosR = response.data;  
    const stado = response.status;  
  
})  
  
//catch: Códigos de estados 4XX o 500  
  
.catch(error => {  
    console.error('Error:', error);  
    alert(`Error ${error.response.status}:  
        ${error.response.data}`);  
  
});
```

7 API REST EN LARAVEL - TAREAS

- ▶ Crearemos una API REST que permita gestionar una lista de tareas.
- ▶ Los Endpoints de esta API serán:
 - ▶ Obtener todas las tareas
 - ▶ Obtener información de una tarea concreta
 - ▶ Crear tarea
 - ▶ Modificar tarea
 - ▶ Borrar tarea
- ▶ Crear proyecto
 - `composer create-project laravel/laravel APItareas`**
- ▶ Configurar conexión con BD en .env

7.1 MODELOS, MIGRACIONES Y CONTROLADORES

- Crear Modelo/Migración/Controlador para la api según las necesidades:

- **Crea modelo, migración y controlador API:**

php artisan make:model NombreModelo -m --api

- **Crea modelo y controlador API si la bd ya existe:**

php artisan make:model NombreModelo --api

- **Crea controlador API si el modelo y la bd ya están creados**

php artisan make:controller NombreControlador --resource

- El controlador incluye un método para cada función de la API

```
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('tareas', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
            $table->enum('prioridad',['Alta','Media','Baja'])->default('Media');
            $table->date('fecha');
            $table->time('hora');
            $table->string('descripción');
            $table->boolean('finalizada')->default(false);
        });
    }
}
```

```
class TareaController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        //
    }

    /**
     * Display the specified resource.
     */
    public function show(Tarea $tarea)
    {
        //
    }

    /**
     * Update the specified resource in storage.
     */
    public function update(Request $request, Tarea $tarea)
    {
        //
    }

    /**
     * Remove the specified resource from storage.
     */
    public function destroy(Tarea $tarea)
    {
        //
    }
}
```

7.2 RUTAS DE LA API

- Definir todas las rutas de la API

```
//Define las rutas para los métodos de la API (CRUD) sin verificar el token csrf
Route::apiResource('tareas',TareaController::class)->withoutMiddleware([VerifyCsrfToken::class]);
```

- Desactivamos la verificación de token csrf porque estamos definiendo las rutas de la api en web.php. Si lo hiciéramos en api.php no sería necesario.

- Ver rutas creadas: **php artisan route:list**

GET HEAD	tareas	tareas.index	>	TareaController@index
POST	tareas	tareas.store	>	TareaController@store
GET HEAD	tareas/{tarea}	tareas.show	>	TareaController@show
PUT PATCH	tareas/{tarea}	tareas.update	>	TareaController@update
DELETE	tareas/{tarea}	tareas.destroy	>	TareaController@destroy

7.3 MÉTODOS DE LA API

```
/**
 * Display a listing of the resource.
 */
public function index()
{
    // Recuperar todas las tareas
    $tareas = Tarea::all();
    return $tareas;
}

/**
 * Display the specified resource.
 */
public function show(Tarea $tarea)
{
    try {
        if ($tarea::find($tarea->id)) {
            return $tarea;
        } else {
            return response()->json('Error', 500);
        }
    } catch (\Throwable $th) {
        return response()->json('Excepción' . $th->getMessage(), 500);
    }
}

public function update(Request $request, Tarea $tarea)
{
    try {
        if (isset($request->descripcion) and $request->descripcion != $tarea->descripcion) {
            $tarea->descripcion = $request->descripcion;
        }
        if (isset($request->fecha) and $request->fecha != $tarea->fecha) {
            $tarea->fecha = $request->fecha;
        }
        if (isset($request->hora) and $request->hora != $tarea->hora) {
            $tarea->hora = $request->hora;
        }
        if (isset($request->prioridad) and $request->prioridad != $tarea->prioridad) {
            $tarea->prioridad = $request->prioridad;
        }
        if (isset($request->finalizada) and $request->finalizada != $tarea->finalizada) {
            $tarea->finalizada = $request->finalizada;
        }

        if ($tarea->save()) {
            return response()->json(['mensaje' => 'Tarea Modificada', 'tarea' => $tarea], 200);
        } else {
            return response()->json('Error al modificar la tarea', 500);
        }
    } catch (\Throwable $th) {
        return response()->json('Excepción' . $th->getMessage(), 500);
    }
}
```

```
/**
 * Store a newly created resource in storage.
 */
public function store(Request $request)
{
    //Validaciones
    $request->validate([
        'prioridad' => 'required|in:Alta,Media,Baja',
        'fecha' => 'required',
        'hora' => 'required',
        'descripcion' => 'required',
    ]);
    try {
        //Crear objeto Tarea
        $tarea = new Tarea();
        $tarea->prioridad = $request->prioridad;
        $tarea->fecha = $request->fecha;
        $tarea->hora = $request->hora;
        $tarea->descripcion = $request->descripcion;
        if ($tarea->save()) {
            return response()->json(['mensaje' => 'Tarea Creada', 'tarea' => $tarea], 201);
        } else {
            return response()->json('Error al crear la tarea', 500);
        }
    } catch (\Throwable $th) {
        return response()->json('Excepción' . $th->getMessage(), 500);
    }
}
```

```
/**
 * Remove the specified resource from storage.
 */
public function destroy(Tarea $tarea)
{
    try {
        if ($tarea->delete($tarea->id)) {
            return response()->json(204);
        } else {
            return response()->json('Error', 500);
        }
    } catch (\Throwable $th) {
        return response()->json('Excepción' . $th->getMessage(), 500);
    }
}
```

7.5 CLIENTE WEB API TAREAS

- **Axios** es una librería de JavaScript que se utiliza para realizar solicitudes HTTP desde el html o Node.js de forma sencilla. Axios convierte automáticamente los datos enviados y recibidos a/desde JSON.
- Usaremos Axios, haciendo referencia a su CDN
- Definir URL base de la API

```
<!-- Incluir Axios desde un CDN -->  
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
```

Gestión de Tareas

dd/mm/aaaa --:-- Descripción

Tareas

Id	Fecha	Hora	Prioridad	Descripción	Finalizada	Acciones
12	2025-01-08	10:20:00	Alta	Clase 2º DAW	0	<input type="button" value="Finalizar"/> <input type="button" value="Borrar"/>
13	2025-01-05	20:11:00	Media	Cabalgata de Reyes	0	<input type="button" value="Finalizar"/> <input type="button" value="Borrar"/>

```
<body>  
  <div class="container">  
    <h1>Gestión de Tareas</h1>  
  
    <!-- Formulario para crear o editar tareas -->  
    <form id="fTareas">  
      <input type="date" id="fecha" placeholder="Fecha" required />  
      <input type="time" id="hora" placeholder="Hora" required />  
      <select name="prioridad" id="prioridad">  
        <option>Alta</option>  
        <option selected="selected">Media</option>  
        <option>Baja</option>  
      </select>  
      <input id="descripcion" placeholder="Descripción" required />  
      <button type="button" class="btn btn-primary" onclick="crearTarea()">Crear</button>  
    </form>  
    <p></p>  
    <!-- Lista de tareas -->  
    <div>  
      <h2>Tareas</h2>  
      <table id="tTareas" class="table">  
        </table>  
    </div>  
  </div>  
  <!-- CONSUMIR API -->  
  <script>  
    // URL base de la API  
    const apiUrl = 'http://localhost:2425/APItareas/public/tareas';  
  </script>
```

7.6 MOSTRAR TAREAS

- ▶ Se realiza una petición por get a la URL base de la API: **axios.get(URL)**
- ▶ Cuando se obtenga la respuesta se crean una fila en la tabla para cada tarea. Los datos de respuestas están en **response.data**.

```
<script>
  // URL base de la API
  const apiUrl = 'http://localhost/2425/APItareas/public/tareas';

  // Función para cargar las tareas
  function cargarTareas() {
    axios.get(apiUrl)
      .then(response => {
        const tareas = response.data; //Axios convierte JSON en una lista de objetos
        const tabla = document.getElementById('tTareas');
        tabla.innerHTML = '<tr><td>Id</td><td>Fecha</td><td>Hora</td><td>Prioridad</td><td>Descripción</td><td>Finalizada</td><td>Acciones</td></tr>';
        tareas.forEach(tarea => {
          const fila = document.createElement('tr');
          fila.innerHTML = `<td>${tarea.id}</td>
            <td>${tarea.fecha}</td>
            <td>${tarea.hora}</td>
            <td>${tarea.prioridad}</td>
            <td>${tarea.descripcion}</td>
            <td>${tarea.finalizada}</td>
            <td><button class="btn btn-outline-primary" onclick="finalizarTarea(${tarea.id})">Finalizar</button>
            <button class="btn btn-outline-primary" onclick="borrarTarea(${tarea.id})">Borrar</button></td>`;
          tabla.appendChild(fila);
        });
      })
      .catch(error => console.error('Error al cargar las tareas:', error));
  }
}
```

7.7 CREAR TAREA

- ▶ Se crea un objeto con los datos de la tarea a crear.
- ▶ Se realiza una petición por POST a la URL base de la API, pasando los datos :
axios.post(URL,datosTarea)
- ▶ Cuando se obtenga la respuesta se recupera el id de la tarea creada, se recarga la lista de tarea y se borran los datos del formulario

```
// Función para crear o actualizar una tarea
function crearTarea() {
  const datosTarea = {
    prioridad: document.getElementById('prioridad').value,
    fecha: document.getElementById('fecha').value,
    hora: document.getElementById('hora').value,
    descripcion: document.getElementById('descripcion').value
  };
  // Si no hay ID, creamos una nueva tarea
  axios.post(apiUrl, datosTarea)
    .then(response => {
      const datosR = response.data;
      alert(`Tarea con id ${datosR.tarea.id} creada correctamente`);
      cargarTareas(); // Recargar la lista de tareas
      limpiarForm(); // Limpiar el formulario
    })
    .catch(error => {
      console.error('Error al crear la tarea:', error);
      alert(`Error ${error.response.status}:${error.response.data}`);
    });
}
```

7.8 MODIFICAR TAREA

```
// Función para finalizar una tarea
function finalizarTarea(id) {
  const datos = {
    finalizada: true
  };
  axios.put(`${apiUrl}/${id}`, datos)
    .then(response => {
      alert('Tarea finalizada correctamente');
      cargarTareas(); // Recargar la lista de tareas
    })
    .catch(error => {
      console.error('Error al crear la tarea:', error);
      alert(`Error ${error.response.status}:${error.response.data}`);
    });
}
```

7.9 BORRAR TAREA

```
// Función para eliminar una tarea
function borrarTarea(id) {
  if (confirm('¿Estás seguro de que deseas eliminar esta tarea?')) {
    axios.delete(`${apiUrl}/${id}`)
      .then(response => {
        alert('Tarea eliminada');
        cargarTareas(); // Recargar la lista de tareas
      })
      .catch(error => {
        console.error('Error al crear la tarea:', error);
        alert(`Error ${error.response.status}:${error.response.data}`);
      });
  }
}
```


8 TOKENS DE AUTENTICACIÓN

- ▶ Aunque las APIs son Stateless, a veces es necesario autenticar y autorizar las solicitudes. Para ello se utilizan **tokens de autenticación**, como **JSON Web Tokens (JWT)** o **Personal Access Tokens**.
- ▶ Los tokens permiten autenticar usuarios de manera **Stateless** al incluir toda la información necesaria en la solicitud. El servidor valida el token en cada solicitud sin guardar el estado del cliente.
- ▶ Un **token de autenticación** es una cadena única generada para un usuario después de autenticarse. Este token se incluye en cada solicitud que el cliente realiza a la API. El servidor valida el token para autorizar al usuario.
- ▶ Tokens en Laravel
 - ▶ **Laravel Sanctum:** es una solución ligera para la autenticación basada en tokens. Se utiliza principalmente para APIs REST y Single Page Applications (SPA).
 - ▶ **Laravel Passport:** es una solución más completa para autenticación OAuth2. Se utiliza para aplicaciones que necesitan autorización avanzada.
- ▶ El modelo User debe implementar HasApiTokens

```
use Illuminate\Foundation\Auth\User as Authenticatable;  
use Illuminate\Notifications\Notifiable;  
use Laravel\Sanctum\HasApiTokens;  
  
class User extends Authenticatable  
{  
    use HasApiTokens, Notifiable;  
}
```

9 API REST EN LARAVEL - TIENDA

- ▶ Crearemos una API REST que permita trabajar con la tienda creada en el tema anterior.
- ▶ Los Endpoints de esta API serán:
 - ▶ Registro
 - ▶ Logueo
 - ▶ Obtener todos los productos, se debe validar token de sesión.
 - ▶ Obtener información de un producto, se debe validar token de sesión.
 - ▶ Obtener todos los pedidos de un cliente, se debe validar token de sesión.
 - ▶ Obtener información de un pedido, se debe validar token de sesión.
 - ▶ Crear pedido, se debe validar token de sesión.
 - ▶ Modificar pedido, se debe validar token de sesión.
- ▶ Crear proyecto
 - `composer create-project laravel/laravel APItienda`
- ▶ Configurar conexión con BD tienda en .env
- ▶ Crear el fichero para rutas de la API e instala Laravel Sanctum para la gestión de tokens
 - `php artisan install:api`
- ▶ Ejecuta las migraciones: Solamente se debe crear la tabla **personal_access_tokens** para gestionar los tokens de autenticación.

9.1 MODELOS, MIGRACIONES Y CONTROLADORES

- ▶ Crear Modelo y Controlador para los productos.
- ▶ Crear Modelo y Controlador para los pedidos.
- ▶ Crear controlador para realizar las acciones relacionadas con el login.
- ▶ Crea las relaciones en los modelos.

9.2 CREACIÓN DE RUTAS DE LA API

- Añadir rutas para la api en **routes/api.php**:

```
//Rutas sin token de autenticación
Route::post('/login', [LoginController::class, 'login']); //Login
Route::post('/registro', [LoginController::class, 'registro']); //Registro

//Rutas en las que hay que enviar token de autenticación
Route::post('/logout', [LoginController::class, 'logout'])->middleware('auth:sanctum'); //Cerrar sesión
Route::get('/productos', [ProductoController::class, 'index'])->middleware('auth:sanctum'); //Recupera todos los productos
Route::get('/pedidos', [PedidoController::class, 'index'])->middleware('auth:sanctum'); //Recuperar pedidos de usuario logueado
Route::post('/pedidos', [PedidoController::class, 'store'])->middleware('auth:sanctum'); //Crea pedido de usuario logueado
```

- Ver rutas de la api: **php artisan route:list**

```
POST      api/login ..... LoginController@login
POST      api/logout ..... LoginController@logout
GET|HEAD  api/pedidos ..... PedidoController@index
POST      api/pedidos ..... PedidoController@store
GET|HEAD  api/productos ..... ProductoController@index
POST      api/registro ..... LoginController@registro
```

9.3 MÉTODOS CONTROLADOR DE LA API

LoginController

```
// Registrar USuario
public function registro(Request $request)
{
    $request->validate([
        'nombre' => 'required',
        'email' => 'required|unique:App\Models\User,email',
        'ps' => 'required|min:3|max:10',
        'ps2' => 'required|min:3|max:10|same:ps'
    ]);

    try {
        $us = new User();
        $us->name = $request->nombre;
        $us->email = $request->email;
        $us->password = Hash::make($request->ps);
        if ($us->save()) {
            return response()->json('Usuario registrado correctamente', 201);
        } else {
            return response()->json('Error al crear el usuario ', 500);
        }
    } catch (\Throwable $th) {
        return response()->json('Excepción ' . $th->getMessage(), 500);
    }
}

// Cierre de sesión
public function logout(Request $request)
{
    try {
        $request->user()->tokens()->delete();
        return response()->json('Cierre de sesión exitoso', 200);
    } catch (\Throwable $th) {
        return response()->json('Excepción ' . $th->getMessage(), 500);
    }
}
```

```
//Validar Usuario
public function login(Request $request)
{
    $request->validate([
        'email' => 'required',
        'ps' => 'required'
    ]);
    try {
        //Crear array con us y ps
        $credenciales = ['email' => $request->email, 'password' => $request->ps];
        //Validación de credenciales
        if (Auth::attempt($credenciales)) {
            //Obtenemos el usuario
            $us = User::find(Auth::user()->id);
            //Generar token de autenticación
            $token = $us->createToken('auth_token')->plainTextToken;
            return response()->json([
                'message' => 'Inicio de sesión exitoso',
                'access_token' => $token,
                'nombre' => $us->name,
                'token_type' => 'Bearer', //Token proviene de cliente autenticado cuando se use
            ]);
        } else {
            return response()->json('Login incorrecto ', 401);
        }
    } catch (\Throwable $th) {
        return response()->json('Excepción ' . $th->getMessage(), 500);
    }
}
```

9.4 MÉTODOS CONTROLADOR DE LA API

ProductoController

```
class ProductoController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        try {
            //Recuperar productos
            $productos = Producto::all();
            return $productos;
        } catch (\Throwable $th) {
            return response()->json('Excepción ' . $th->getMessage(), 500);
        }
    }
}
```

9.5 MÉTODOS CONTROLADOR DE LA API

PedidoController

```
/**
 * Display a listing of the resource.
 */
public function index()
{
    //
    try {
        $pedidos = Pedido::where('user_id', Auth::user()->id)->orderBy('id', 'desc')->get();
        return PedidoResource::collection($pedidos);
    } catch (\Throwable $th) {
        return response()->json('Excepción ' . $th->getMessage(), 500);
    }
}
```

```
public function store(Request $request)
{
    //Comprobar que hay producto
    $request->validate([
        'producto' => 'required'
    ]);
    try {
        //Crear el pedido
        DB::transaction(function () use ($request) {
            //Obtener Producto
            $p = Producto::find($request->producto);

            //Comprobar si hay stock
            if ($p==null or $p->stock < 1) {
                throw new Exception('Error, no hay stock para el producto');
            }
            $pe = new Pedido();
            $pe->user_id = Auth::user()->id;
            $pe->producto_id = $p->id;
            $pe->cantidad = 1;
            $pe->precioU = $p->precio;
            if ($pe->save()) {
                //Modificar stock del producto
                $p->stock -= 1;
                $p->save();
            }
            return response()->json('Pedido'. $pe->id.' Creado', 201);
        });
    } catch (\Throwable $th) {
        return response()->json('Excepción, '.$th->getMessage(), 500);
    }
}
```

9.6 CLIENTE WEB API TIENDA - LOGIN

Tienda Online

Email

Contraseña

Login Registrarse

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Tienda</title>
  <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfspd3yD65VohhpuuCOMLASJC" crossorigin="anonymous">
</head>
<body>
  <div class="container">
    <h1>Tienda Online</h1>
    <form action="" method="post" class="row">
      <div class="row justify-content-md-center">
        <div class="col col-lg-3">
          <label for="email" class="form-label">Email</label><br/>
          <input type="email" name='email' id="email" class="form-control"/>
        </div>
      </div>
      <div class="row justify-content-md-center">
        <div class="col col-lg-3">
          <label for="ps" class="form-label">Contraseña</label><br/>
          <input type="password" name='ps' id="ps" class="form-control"/>
        </div>
      </div>
      <div class="row justify-content-md-center">
        <div class="col col-lg-3">
          <button type="button" name="login" class="btn btn-outline-secondary" onclick="loguear()">Login</button>
          <a href="registro.html" class="btn btn-outline-secondary">Registrarse</a>
        </div>
      </div>
    </form>
  </div>
```

```
<!-- CONSUMIR API-->
<script>
  // URL base de la API
  const apiUrl = 'http://localhost/2425/APItienda/public/api/login';

  // Función para cargar las tareas
  function loguear() {
    const datos = {
      email: document.getElementById('email').value,
      ps: document.getElementById('ps').value
    };
    axios.post(apiUrl, datos)
      .then(response => {
        // Recupera y muestra el token
        const token = response.data.access_token;
        alert('Login exitoso. Token: ' + token);

        // Guarda el token en el almacenamiento local (opcional)
        localStorage.setItem('token', token);
        // Guarda el nombre en el almacenamiento local (opcional)
        localStorage.setItem('nombre', response.data.nombre);
        // Redirigir a index.html
        window.location.href = 'index.html';
      })
      .catch(error => {
        console.error('Error al validar usuario:', error);
        const datosR = error.response.data;
        alert(`Error ${error.response.status}:${datosR}`);
      });
  }
</script>
```


9.7 CLIENTE WEB API TIENDA - REGISTRO

Registro de Usuarios

Nombre

Email

Contraseña

Confirmar Contraseña

```
<!-- CONSUMIR API -->
<script>
  // URL base de la API
  const apiUrl = 'http://localhost:2425/APItienda/public/api/registro';

  // Función para cargar las tareas
  function registrar() {
    const datos = {
      nombre: document.getElementById('nombre').value,
      email: document.getElementById('email').value,
      ps: document.getElementById('ps').value,
      ps2: document.getElementById('ps2').value
    };
    axios.post(apiUrl, datos)
      .then(response => {
        alert('Registro correcto');
        // Redirigir a login.html
        window.location.href = 'login.html';
      })
      .catch(error => {
        console.error('Error al registrar el usuario:', error);
        const datosR = error.response.data;
        alert(`Error ${error.response.status}: ${datosR}`);
      });
  }
</script>
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Tienda</title>
  <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfSpd3yD65VohhpucComLASjC" crossorigin="anonymous">
</head>

<body>
  <div class="container">
    <h1>Registro de Usuarios</h1>
    <form method="post" class="row g-3">
      <div class="row-md-3">
        <label for="nombre" class="form-label">Nombre</label>
        <input type="text" id="nombre" placeholder="Nombre" class="form-control" />
      </div>
      <div class="row-md-3">
        <label for="email" class="form-label">Email</label>
        <input type="email" id="email" placeholder="correo@gmail.com" class="form-control" />
      </div>
      <div class="row-md-3">
        <label for="ps" class="form-label">Contraseña</label>
        <input type="password" id="ps" class="form-control" />
      </div>
      <div class="row-md-3">
        <label for="ps2" class="form-label">Confirmar Contraseña</label>
        <input type="password" id="ps2" class="form-control" />
      </div>
      <div class="row-md-3">
        <button type="button" name="crearU" value="crearU" onclick="registrar()"
          class="btn btn-outline-secondary">Crear</button>
        <a href="login.html" class="btn btn-outline-secondary">Volver</a>
      </div>
    </form>
  </div>
```

```
<!-- CONSUMIR API -->
<script>
  // URL base de la API
  const apiUrl = 'http://localhost:2425/APItienda/public/api/registro';

  // Función para cargar las tareas
  function registrar() {
    const datos = {
      nombre: document.getElementById('nombre').value,
      email: document.getElementById('email').value,
      ps: document.getElementById('ps').value,
      ps2: document.getElementById('ps2').value
    };
    axios.post(apiUrl, datos)
      .then(response => {
        alert('Registro correcto');
        // Redirigir a login.html
        window.location.href = 'login.html';
      })
      .catch(error => {
        console.error('Error al registrar el usuario:', error);
        const datosR = error.response.data;
        alert(`Error ${error.response.status}: ${datosR}`);
      });
  }
</script>
```

9.8 CLIENTE WEB API TIENDA - INDEX I




```
<body>
  <div class="container">
    <h1>Tienda Online</h1>
    <div class="row">
      <h3 class="col-2" id="nombre"></h3>
      <a class="col-1 btn btn-outline-secondary" onclick="salir()">Salir</a>
    </div>
    <div class="container">
      <div class="row">
        <div class="col">
          <h3>Productos</h3>
          <table id="tProductos" class="table">
            </table>
        </div>
        <div class="col">
          <h3>Pedidos</h3>
          <table id="tCompras" class="table">
            </table>
        </div>
      </div>
    </div>
  </div>
</div>
```

Tienda Online

Cliente:rosa

Salir

Productos

Id	Nombre	Precio	Stock	Imagen	Acciones
1	Ratón inalámbrico	10.2	20		Comprar
2	Módulo RAM 32GB	100.4	5		Comprar
3	Disco duro SSD 2TB	80.54	0		Comprar

Pedidos

Id	Fecha	Producto	Precio	Cantidad	Total	Imagen
39	2024-12-31T16:56:20.000000Z	Ratón inalámbrico	10.2	1	10.2	

```
<script>
  //Obtener el token
  const token = localStorage.getItem('token');

  if (token == null) {
    // Redirigir a login.html
    window.location.href = 'login.html';
  } else {
    //Obtener el token
    const token = localStorage.getItem('token');
    axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;
    //Obtener nombre usuario
    const nombre = localStorage.getItem('nombre');
    document.getElementById('nombre').innerHTML=`Cliente:${nombre}`;
    cargarProductos();
    cargarPedidos();
  }

  // Función para cargar las tareas
  function salir() {
    // URL base de la API
    const apiUrl = 'http://localhost:2425/APItienda/public/api/logout';
    axios.post(apiUrl).then(response => {
      // Elimina el token en el almacenamiento local (opcional)
      localStorage.removeItem('token');
      // Redirigir a login.html
      window.location.href = 'login.html';
    })
    .catch(error => {
      console.error('Error al desconectar:', error);
      const datosR = error.response.data;
      alert(`Error ${error.response.status}:${datosR}`);
    });
  }
}
```

9.9 CLIENTE WEB API TIENDA - INDEX II

```
// Función para cargar los productos
function cargarProductos() {
  // URL base de la API
  const apiUrl = 'http://localhost:2425/APItienda/public/api/productos';
  axios.get(apiUrl)
    .then(response => {
      const productos = response.data; //Axios convierte JSON en una lista de objetos
      const tabla = document.getElementById('tProductos');
      tabla.innerHTML = '<tr><td>Id</td><td>Nombre</td><td>Precio</td><td>Stock</td><td>Imagen</td><td>Acciones</td></tr>';
      productos.forEach(producto => {
        const fila = document.createElement('tr');
        fila.innerHTML = `<td>${producto.id}</td>
        <td>${producto.nombre}</td>
        <td>${producto.precio}</td>
        <td>${producto.stock}</td>
        <td></td>
        <td><button class="btn btn-outline-primary" onclick="comprar(${producto.id})">Comprar</button></td>`;
        tabla.appendChild(fila);
      });
    })
    .catch(error => console.error('Error al cargar los productos:', error));
}
```

9.10 CLIENTE WEB API TIENDA - INDEX III

```
// Función para cargar las compras realizadas por el usuario
function cargarPedidos() {
  // URL base de la API
  const apiUrl = 'http://localhost/2425/APItienda/public/api/pedidos';
  axios.get(apiUrl)
    .then(response => {
      const pedidos = response.data.data; //Axios convierte JSON en una lista de objetos
      const tabla = document.getElementById('tCompras');
      tabla.innerHTML = '<tr><td>Id</td><td>Fecha</td><td>Producto</td><td>Precio</td><td>Cantidad</td><td>Total</td><td>Imagen</td></tr>';
      pedidos.forEach(pedido => {
        const fila = document.createElement('tr');
        fila.innerHTML = `<td>${pedido.id}</td>
          <td>${pedido.fecha}</td>
          <td>${pedido.producto}</td>
          <td>${pedido.precioU}</td>
          <td>${pedido.cantidad}</td>
          <td>${pedido.cantidad * pedido.precioU}</td>
          <td></td>`;
        tabla.appendChild(fila);
      });
    })
    .catch(error => console.error('Error al cargar los productos:', error));
}
```

9.11 CLIENTE WEB API TIENDA - INDEX IV

```
// Función para realizar una compra
function comprar(idP) {
  // URL base de la API
  const apiUrl = 'http://localhost/2425/APItienda/public/api/pedidos';
  const datos = {
    producto: idP
  };
  axios.post(apiUrl, datos)
    .then(response => {
      alert('Compra realizada');
      cargarProductos();
      cargarPedidos();
    })
    .catch(error => {
      const datosR = error.response.data;
      console.error('Error al crear pedido:', error);
      alert(`${datosR}`);
    });
}
```

9.12 RECUSO PEDIDOS

- Personalizar el JSON de pedido para que se devuelva el nombre del producto en lugar del id, la fecha y la imagen.
- **php artisan make:resource PedidoResource**

```
class PedidoResource extends JsonResource
{
    /**
     * Transform the resource into an array.
     *
     * @return array<string, mixed>
     */
    public function toArray(Request $request): array
    {
        //return parent::toArray($request);
        return [
            'id' => $this->id,
            'fecha' => $this->created_at,
            'producto' => $this->producto->nombre,
            'precioU' => $this->precioU,
            'cantidad' => $this->cantidad,
            'imagen' => $this->producto->imagen
        ];
    }
}
```

```
class PedidoController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        //
        try {
            $pedidos = Pedido::where('user_id', Auth::user()->id)->orderBy('id', 'desc')->get();
            return PedidoResource::collection($pedidos);
        } catch (\Throwable $th) {
            return response()->json('Excepción ' . $th->getMessage(), 500);
        }
    }
}
```