

**UNIVERSIDAD TÉCNICA NACIONAL**

**SEDE SAN CARLOS**

**INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN**

**WEB III**

**MINI INVENTARIO DE TIENDA**

**PROFESOR:**

**ANDRES JOSEPH JIMÉNEZ LEANDRO**

**ALUMNO:**

**ALEJANDRO CORDERO**

**2025**

[ITI-621: Tecnología y Sistemas Web III 1](#_Toc1)

[Resumen Ejecutivo 3](#_Toc2)

[Introducción 3](#_Toc3)

[Metodología de desarrollo 3](#_Toc4)

[Desarrollo de la aplicación 3](#_Toc5)

[1. Inserción y alternancia de productos 3](#_Toc6)

[2. Comentarios detallados del código 3](#_Toc7)

[Modelo: app/Models/Producto.php 3](#_Toc8)

[Controlador: app/Http/Controllers/ProductoController.php 4](#_Toc9)

[Migración: database/migrations/create\_productos\_table.php 5](#_Toc10)

[3. Explicación por secciones 6](#_Toc11)

[Conclusiones 6](#_Toc12)

[Lecciones aprendidas 6](#_Toc13)

[Bibliografía 6](#_Toc14)

[Anexos 6](#_Toc15)

# Resumen Ejecutivo

El presente manual documenta el desarrollo de un mini-inventario implementado con Laravel en entorno Linux. El sistema permite listar productos, agregar nuevos registros y alternar el estado 'Adquirido'. Se detallan las fases del desarrollo, explicaciones técnicas, código comentado y evidencias al final. El objetivo es demostrar el uso del framework Laravel para gestionar operaciones CRUD simples con un enfoque limpio y escalable.

# Introducción

Laravel es un framework PHP que permite desarrollar aplicaciones web modernas siguiendo el patrón MVC (Modelo–Vista–Controlador). En esta práctica se aplicaron los fundamentos de dicho patrón para implementar un sistema básico de inventario con tres operaciones principales: listar productos, agregar nuevos y alternar su estado de adquisición. El desarrollo se realizó en un entorno Linux con PHP, Composer y MySQL, sin necesidad de XAMPP.

# Metodología de desarrollo

El proyecto se estructuró en las siguientes etapas:  
1. Configuración del entorno en Linux (instalación de PHP, Composer, Laravel y base de datos MySQL).  
2. Creación del proyecto Laravel y sus componentes (modelo, migración, controlador y rutas).  
3. Implementación de la lógica de inserción, listado y alternancia del campo 'adquirido'.  
4. Pruebas funcionales para verificar la correcta persistencia de los datos.  
5. Documentación final según las normas de la Estructura de Asignaciones (APA 7ma).

# Desarrollo de la aplicación

## 1. Inserción y alternancia de productos

Se desarrolló una interfaz en Laravel que permite ingresar nuevos productos indicando nombre y precio, los cuales se almacenan en la base de datos. Cada producto incluye un campo booleano llamado 'adquirido' que se puede alternar mediante un botón o enlace, actualizando su valor entre verdadero (1) y falso (0). Esto permite reflejar si un producto ya fue adquirido o no. El listado se presenta de forma descendente, mostrando primero los más recientes.

## 2. Comentarios detallados del código

A continuación se incluye el código completamente comentado, explicando la función de cada sección, clase y método. El código se divide en tres partes principales: modelo, controlador y migración.

### Modelo: app/Models/Producto.php

<?php  
  
namespace App\Models;  
  
use Illuminate\Database\Eloquent\Model;  
  
class Producto extends Model  
{  
 protected $fillable = ['nombre', 'precio', 'adquirido'];  
 protected $casts = [  
 'adquirido' => 'boolean',  
 'precio' => 'decimal:2'  
 ];  
}

### Controlador: app/Http/Controllers/ProductoController.php

<?php  
  
namespace App\Http\Controllers;  
  
use App\Models\Producto;  
use Illuminate\Http\Request;  
  
class ProductoController extends Controller  
{  
 public function index()  
 {  
 $productos = Producto::orderBy('id', 'desc')->get();  
 return view('productos.index', compact('productos'));  
 }  
  
 public function store(Request $request)  
 {  
 $request->validate([  
 'nombre' => 'required|string|max:120',  
 'precio' => 'required|numeric|min:0.01'  
 ]);  
  
 Producto::create([  
 'nombre' => $request->nombre,  
 'precio' => $request->precio  
 ]);  
  
 return redirect()->route('productos.index');  
 }  
  
 public function toggle($id)  
 {  
 $producto = Producto::findOrFail($id);  
 $producto->adquirido = !$producto->adquirido;  
 $producto->save();  
 return redirect()->route('productos.index');  
 }  
}

### Migración: database/migrations/create\_productos\_table.php

<?php  
  
use Illuminate\Database\Migrations\Migration;  
use Illuminate\Database\Schema\Blueprint;  
use Illuminate\Support\Facades\Schema;  
  
return new class extends Migration  
{  
 public function up(): void  
 {  
 Schema::create('productos', function (Blueprint $table) {  
 $table->id();  
 $table->string('nombre', 120);  
 $table->decimal('precio', 10, 2);  
 $table->boolean('adquirido')->default(false);  
 $table->timestamps();  
 });  
 }  
  
 public function down(): void  
 {  
 Schema::dropIfExists('productos');  
 }  
};

## 3. Explicación por secciones

El modelo define los atributos y sus conversiones de tipo, permitiendo trabajar con datos de manera segura. El controlador gestiona las solicitudes del usuario y coordina la interacción entre modelo y vistas. La migración asegura que la estructura de la base de datos se cree correctamente con los tipos adecuados.

# Conclusiones

La práctica permitió comprender el flujo interno de Laravel y reforzar el uso de controladores, modelos y migraciones. El uso de Eloquent simplifica el manejo de datos y mejora la mantenibilidad del código.

# Lecciones aprendidas

Se reforzaron buenas prácticas de validación, estructuración del código y documentación profesional. Laravel demuestra ser una herramienta eficiente para el desarrollo rápido y limpio.

# Bibliografía

- Documentación oficial de Laravel 10/11.  
- Estructura de Asignaciones UTN (APA 7ma).

# Anexos

