# Documentación de practica de vacante de desarrolladora: CaribeShop (Asp.net + Angular + Bootstrap)

## API (ASP.NET)

## Resumen

CaribeShop es una API RESTful desarrollada en ASP.NET Core que proporciona funcionalidades para la gestión de productos, autenticación de usuarios y manejo de un carrito de compras. La arquitectura utilizada es la arquitectura Onion, que sigue principios limpios con capas separadas para dominio, aplicación y presentación.

## Tecnologías Utilizadas

* ASP.NET Core Web API: Framework para construir la API REST.
* Entity Framework Core: ORM para acceso a la base de datos usando Code First.
* SQL Server: Base de datos relacional.
* JWT (JSON Web Tokens): Para autenticación y autorización basada en tokens.
* C#: Lenguaje de programación principal.
* Swagger / OpenAPI: Para documentación y pruebas interactivas de la API.
* Inyección de Dependencias: Para desacoplar servicios y repositorios.
* Password hashing (SHA256): Para almacenar contraseñas de forma segura.

## Estructura y Componentes

### 1. Entidades de Dominio

- User, Product, CartItem, OrderItem: Modelan los datos y relaciones principales.  
- Clases base con propiedades de auditoría para seguimiento (AuditableBaseEntities).

### 2. Servicios de Aplicación

- ProductService: Gestiona la lógica relacionada a productos, como obtener todos o uno específico.  
- CartService: Maneja las operaciones sobre el carrito de compras (añadir, actualizar, eliminar, limpiar).  
- AuthenticationService: Controla el registro, login y obtención de datos de usuario. Utiliza hashing de contraseñas y generación de tokens JWT.

### 3. Controladores API

- ProductsController: Expone endpoints para consultar productos (GET /api/products, GET /api/products/{id}).  
- CartController: Permite gestionar el carrito, con autenticación requerida, usando endpoints como añadir, actualizar, eliminar ítems.  
- AuthenticationController: Endpoints para registro, login y consulta de usuario. Genera y devuelve tokens JWT para sesiones autenticadas.

## Autenticación y Seguridad

- La API usa JWT para autenticar solicitudes.  
- Las contraseñas se almacenan con hash SHA256 para mayor seguridad.  
- Endpoints que modifican datos sensibles (como carrito) están protegidos con [Authorize] y requieren token válido.  
- Token generado incluye el nombre de usuario y tiene expiración configurada (por defecto 60 minutos).

## Principales Endpoints

|  |  |  |  |
| --- | --- | --- | --- |
| Método | Ruta | Descripción | Seguridad |
| GET | /api/products | Obtiene lista de todos los productos | Pública |
| GET | /api/products/{id} | Obtiene un producto específico por ID | Pública |
| POST | /register | Registra un nuevo usuario | Pública |
| POST | /login | Autentica usuario y retorna token JWT | Pública |
| GET | /user?username=... | Obtiene datos públicos de usuario | Pública |
| GET | /api/cart | Obtiene contenido del carrito | Autenticado |
| POST | /api/cart/add | Añade ítem al carrito | Autenticado |
| PUT | /api/cart/update | Actualiza cantidad ítem carrito | Autenticado |
| DELETE | /api/cart/remove | Elimina ítem del carrito | Autenticado |
| DELETE | /api/cart/clear | Vacía el carrito | Autenticado |

## Flujo Básico de Uso

1. Registro: El usuario envía sus datos al endpoint /register para crear una cuenta. La contraseña se encripta con SHA256 y se guarda en la base de datos.  
2. Login: El usuario envía usuario y contraseña a /login. Si son válidos, recibe un token JWT que debe enviar en el encabezado de autorización para acceder a recursos protegidos.  
3. Consulta de productos: El cliente consume /api/products para mostrar catálogo, y puede ver detalles con /api/products/{id}.  
4. Gestión del carrito: Usando el token JWT, el cliente realiza operaciones sobre el carrito a través de /api/cart (añadir, actualizar, eliminar, limpiar).

## Consideraciones Técnicas

- Inyección de Dependencias permite cambiar la implementación de repositorios o servicios sin afectar el controlador.  
- DTOs y ViewModels se usan para separar la lógica de negocio de la presentación y la API.  
- Auditoría: Las entidades heredan propiedades para rastrear creación y modificación.  
- Validaciones básicas se implementan en el servicio de autenticación para evitar usuarios duplicados.

**FRONTEND (ANGULAR)**

## Resumen

El frontend de CaribeShop es una aplicación web desarrollada en Angular que permite a los usuarios registrarse, iniciar sesión, explorar productos, administrar un carrito de compras y realizar un checkout con confirmación de pedido. La interfaz es moderna, reactiva y está diseñada para ofrecer una experiencia fluida y segura.

## Funcionalidades principales

* **Autenticación:** Registro y login de usuarios con validación básica y gestión de token JWT para mantener sesiones.
* **Gestión de Productos:** Listado de productos obtenidos desde la API backend, con visualización detallada por producto.
* **Carrito de Compras:** Agregar, disminuir cantidad, eliminar productos y limpiar el carrito, con actualización reactiva del contador de ítems.
* **Checkout:** Confirmación de pedido con ingreso manual de dirección y limpieza del carrito tras completar la compra.
* **Navegación:** Rutas protegidas con guardias de autenticación para restringir el acceso a usuarios no autenticados.

## ¿Cómo Funciona?

* **Rutas y Navegación:**  
  Se utiliza el sistema de rutas de Angular para manejar las distintas páginas (login, registro, productos, detalle, carrito, checkout). Algunas rutas están protegidas con un guardia authGuard que verifica si el usuario tiene un token válido.
* **Servicios y Comunicación con API:**  
  Los servicios (AuthService, ProductService, CartService) manejan la lógica para interactuar con la API RESTful backend y almacenar el estado local del carrito.
  + AuthService hace peticiones POST para registro y login, recibe y almacena token JWT en localStorage.
  + ProductService obtiene productos y detalles usando HTTP GET.
  + CartService mantiene un array local con los productos agregados, usa un BehaviorSubject para emitir el conteo actualizado del carrito a cualquier componente que lo necesite (por ejemplo, la barra de navegación).

Componentes:  
La aplicación está dividida en componentes independientes y reutilizables:

* + LoginComponent y RegisterComponent manejan autenticación y validación de formulario simple.
  + ProductListComponent muestra la lista de productos obtenidos de la API.
  + ProductDetailComponent muestra detalles del producto seleccionado y permite agregarlo al carrito.
  + CartComponent muestra los productos agregados, con opciones para aumentar, disminuir, eliminar y limpiar el carrito.
  + CheckoutComponent permite confirmar el pedido ingresando la dirección y vacía el carrito al confirmar.
* **Estado Reactivo:**  
  El contador del carrito en la barra de navegación se actualiza automáticamente mediante un observable (BehaviorSubject) que emite el total de ítems agregados.
* **Persistencia Local:**  
  El token JWT y el nombre de usuario se almacenan en localStorage para persistir la sesión entre recargas.

## Tecnologías Utilizadas

* **Angular (Standalone Components):**  
  Uso de Angular 15+ con componentes standalone para modularidad y simplicidad en la configuración.
* **TypeScript:**  
  Para tipado estático y desarrollo estructurado.
* **RxJS:**  
  Para manejo reactivo del estado del carrito y operaciones asíncronas.
* **Angular Router:**  
  Para navegación y protección de rutas con guards.
* **Angular Forms (Template-driven):**  
  Para captura y validación básica en formularios de login y registro.
* **HTTPClient (Angular):**  
  Para comunicación con la API backend usando peticiones HTTP.
* **LocalStorage:**  
  Para almacenar token JWT y mantener la sesión activa.

# DISENO (Bootstrap)

## Identidad visual y paleta de colores

La aplicación fue diseñada siguiendo una línea visual coherente con la identidad institucional del Banco de Ahorro y Crédito Caribe (BACC). Se respetaron las siguientes reglas de estilo:

* **Fondo principal:**  
  Se utiliza un tono de azul oscuro institucional (#003865) como fondo de todas las pantallas importantes (login, registro, checkout, detalles, etc.).
* **Color de texto:**  
  Se emplea un tono de **gris institucional** (#6c757d) para todos los textos, excepto los títulos o precios destacados.
* **Color de acento:**  
  Para botones principales y llamados a la acción (CTA), se utiliza btn-primary, personalizado con azul institucional si se desea expandir.
* **Color blanco:**  
  El blanco se reserva únicamente para fondos de tarjetas, formularios y componentes internos (por ejemplo, los <div class="card">), manteniendo un contraste adecuado con el fondo.

## Consistencia visual

* Se estandarizó el uso de **Bootstrap 5** para mantener un diseño limpio y responsive.
* Todos los formularios (login, registro, checkout) siguen una estructura visual centralizada, con max-width para limitar el ancho y sombra (shadow) para dar profundidad.
* Las tarjetas de productos y componentes internos mantienen bordes redondeados, sombras sutiles y márgenes consistentes.

## Diseño responsivo

* Todos los layouts fueron diseñados con clases como container, row, col-md-4, col-md-6, etc., lo que garantiza una experiencia fluida en **pantallas grandes y móviles**.
* Imágenes y tarjetas se ajustan con img-fluid y object-fit: cover para evitar distorsión.
* Formularios y botones usan clases como w-100, d-grid, y vh-100 para una correcta alineación vertical y horizontal sin importar el dispositivo.

## Diseño por vista

**Pantalla de Login y Registro**

* Fondo azul institucional con el formulario centrado y tarjeta blanca.
* Inputs con validación visual (is-valid, is-invalid).
* Botones accesibles y bien espaciados.

**Lista de productos**

* Cada producto se muestra en una tarjeta blanca con imagen, nombre, descripción y botón de acción.
* Imágenes recortadas uniformemente para mantener armonía visual.

**Carrito y Checkout**

* Tabla con botones para sumar/restar productos.
* Checkout con diseño limpio, donde el usuario introduce su dirección de forma manual.
* Botones grandes y accesibles para confirmar el pedido o vaciar el carrito.

**Detalle del producto**

* Imagen grande del producto a la izquierda, y descripción detallada a la derecha.
* Botón para “Agregar al carrito” y otro para volver a la lista de productos.

## Accesibilidad y experiencia de usuario

* Se utilizaron etiquetas label correctamente asociadas a sus respectivos campos input.
* Contraste suficiente entre texto y fondo para asegurar legibilidad.
* Botones e inputs de tamaño adecuado para dispositivos táctiles.