📘 Guía Oficial de Requisitos del Proyecto BAZAR

1. Objetivo de la guía **2**
2. Requisitos del frontend orientado a clientes **2**
2.1 Funcionalidades principales 2
2.2 Navegación y comportamiento 3
3. Tabla categorias **3**
3.1 Campos 3
3.2 Reglas de negocio 3
3.3 Relación con productos 4
3.4 Rutas recomendadas 4
4. Tabla productos **4**
4.1 Campos 4
4.2 Reglas de negocio 4
4.3 Búsqueda y filtros 5
4.4 Productos recomendados 5
5. Tabla clientes **5**
5.1 Campos 5
5.2 Reglas de negocio 6
5.3 Login 6
5.4 Perfil 6
6. Tabla pedidos **7**
6.1 Campos 7
6.2 Reglas de negocio 7
7. Tabla pedidos_productos (relación N–M) **7**
7.1 Campos 7
7.2 Reglas de negocio 8
8. Reglas del carrito y del proceso de pedido **8**
8.1 Carrito sin login 8
8.2 Finalizar pedido 8
8.3 Creación del pedido 8
9. Rutas recomendadas del backend **9**
9.1 Autenticación 9
9.2 Categorías 9
9.3 Productos 9
9.4 Pedidos (usuario autenticado) 9
10. Tecnologías utilizadas **10**
10.1 Backend 10
10.2 Base de datos 10
10.3 Frontend 10
10.4 Autenticación 11

10.5 Errores y validación 11
11. Checklist final del backend **11**

1. Objetivo de la guía

Este documento define los requisitos funcionales y técnicos necesarios para el correcto desarrollo del proyecto **Bazar**, que incluye un backend en Node.js + Express y un frontend en React.
La guía establece:

- reglas de negocio,
- estructura de datos,
- modelos,
- controladores,
- rutas API,
- tecnologías empleadas,
- y el comportamiento esperado del cliente.

Su función principal es asegurar que toda la aplicación sigue una arquitectura coherente, consistente y mantenible.

---

2. Requisitos del frontend orientado a clientes

El frontend está orientado exclusivamente a usuarios clientes. El proyecto **no** incluye panel de administración.

2.1 Funcionalidades principales

El usuario cliente debe poder:

- visualizar productos,
- buscar por nombre,
- filtrar por categoría,

- acceder a un detalle de producto,
- añadir productos al carrito,
- finalizar un pedido únicamente si ha iniciado sesión.

2.2 Navegación y comportamiento

- Si el usuario ha iniciado sesión:
    - se muestra su nombre en la barra superior,
    - aparece un botón de cierre de sesión.
- Si el usuario no ha iniciado sesión:
    - se muestran los botones de acceso y registro.

---

3. Tabla categorias

3.1 Campos

Campo	Tipo	Obligatorio	Descripción
id	INT (PK)	Sí	Identificador
nombre	VARCHAR(100)	Sí	Único
slug	VARCHAR(120)	No	Para URLs limpias
descripcion	TEXT	No	Información descriptiva
activo	TINYINT(1) DEFAULT 1	No	Ocultar sin eliminar

3.2 Reglas de negocio

- El nombre es obligatorio y único.
- Categorías inactivas no aparecen en filtros, pero se mantienen para productos ya vinculados.

- No se permite eliminar categorías con productos asociados; deben reasignarse previamente.

3.3 Relación con productos

- `productos.categoria_id` es FK hacia `categorias.id`.
- Debe existir un índice sobre `categoria_id`.

3.4 Rutas recomendadas

- GET `/api/categorias`
- GET `/api/categorias/:id` (opcional)

---

4. Tabla productos

4.1 Campos

Campo	Tipo	Obligatorio	Descripción
id	INT (PK)	Sí	Identificador
nombre	VARCHAR(150)	Sí	Nombre visible
descripcion	TEXT	No	Detalle
precio	DECIMAL(10,2)	Sí	Precio final
stock	INT	Sí	Si = 0 → agotado
imagen_url	VARCHAR(255)	Sí	URL principal
categoria_id	INT (FK)	Sí	Relación
destacado	TINYINT(1) DEFAULT 0	No	Mostrar en inicio
activo	TINYINT(1) DEFAULT 1	Sí	Soft delete

4.2 Reglas de negocio

- Es obligatorio indicar nombre, precio y categoría.

- `precio > 0`; `stock >= 0`.
- Si `stock = 0`, no se permite añadir al carrito.
- Solo se listan productos activos.

4.3 Búsqueda y filtros

- Búsqueda por nombre: `GET /api/productos?nombre=camiseta`
- Filtros por categoría: `GET /api/productos?categoria_id=2`
- Paginación: `page`, `limit`
- Ordenación: `sort=precio_asc`, `precio_desc`, `nombre_asc`, etc.

4.4 Productos recomendados

- En `/api/productos/:id` deben devolverse tres productos recomendados.
- Se prioriza la misma categoría.

---

5. Tabla clientes

5.1 Campos

Campo	Tipo	Obligatorio	Descripción
id	INT (PK)	Sí	Identificador
nombre	VARCHAR(100)	Sí	Visible en navbar
email	VARCHAR(150) UNIQUE	Sí	Guardado en minúsculas
password_hash	VARCHAR(255)	Sí	Contraseña cifrada
domicilio	TEXT	No	Se añade al hacer pedido

telefono	INT	No	Se añade al hacer pedido
fecha_registro	DATETIME	Sí	`CURRENT_TIMESTAMP`
rol	ENUM('cliente','admin')	Sí	Por defecto “cliente”
activo	TINYINT(1) DEFAULT 1	Sí	Control de acceso

5.2 Reglas de negocio

- Es obligatorio proporcionar nombre, email y contraseña.
- Email único y normalizado.
- Contraseña mínima definida en controlador.
- Nunca se retorna `password_hash`.

5.3 Login

1. Buscar cliente por email.
2. Verificar contraseña mediante bcrypt.
3. Comprobar que el usuario está activo.
4. Generar token JWT con id, nombre, email y rol.
5. Responder con token + datos.

5.4 Perfil

- Acceso protegido por token.
- Solo permite consulta (perfil + pedidos).

---

6. Tabla pedidos

6.1 Campos

Campo	Tipo	Obligatorio	Descripción
id	INT (PK)	Sí	Identificador
cliente_id	INT (FK)	Sí	Cliente dueño
fecha	DATETIME	Sí	Timestamp
estado	ENUM	Sí	`carrito`, `pendiente`, `pagado`, `enviado`, `cancelado`
total	DECIMAL(10,2)	Sí	`DEFAULT 0.00`

6.2 Reglas de negocio

- `cliente_id` no puede ser NULL.
- Un cliente solo puede tener un pedido en estado **carrito**.
- El total siempre se recalcula en backend.
- Estado cambia según acciones del proceso de compra.

---

7. Tabla pedidos_productos (relación N–M)

7.1 Campos

Campo	Tipo	Obligatorio	Descripción
id	INT (PK)	Sí	Identificador
pedido_id	INT (FK)	Sí	Pedido
producto_id	INT (FK)	Sí	Producto
cantidad	INT	Sí	Mayor que 0
precio_unitario	DECIMAL(10,2)	Sí	Precio histórico

7.2 Reglas de negocio

- No se permiten líneas incompletas.
- `cantidad` debe ser menor o igual al stock.
- Al confirmar un pedido se descuenta stock.
- Si un usuario añade varias veces el mismo producto, se actualiza la cantidad (no se duplican líneas).
- Subtotal = cantidad × precio_unitario (no se guarda en BD).

---

8. Reglas del carrito y del proceso de pedido

8.1 Carrito sin login

El usuario no registrado puede:

- navegar,
- añadir productos al carrito,
- modificar cantidades,
- ver totales.

8.2 Finalizar pedido

- Solo es posible si el usuario está logueado.
- Si no lo está, se redirige a Login/Registro.

8.3 Creación del pedido

Una vez el usuario inicia sesión:

- se genera el pedido real en MySQL,
- se enlaza a su `cliente_id`,

- se reconstruye a partir del carrito.

---

9. Rutas recomendadas del backend

9.1 Autenticación

- POST `/api/auth/register`
- POST `/api/auth/login`
- GET `/api/mi-perfil`
- PUT `/api/mi-perfil`

9.2 Categorías

- GET `/api/categorias`
- GET `/api/categorias/:id`

9.3 Productos

- GET `/api/productos`
- GET `/api/productos/:id`
- GET `/api/productos/destacados` (opcional)

9.4 Pedidos (usuario autenticado)

- GET `/api/mis-pedidos`
- GET `/api/mis-pedidos/:id`
- POST `/api/mis-pedidos`

- POST `/api/mis-pedidos/:id/lineas`
- PUT `/api/mis-pedidos/:id/lineas/:lineaId`
- DELETE `/api/mis-pedidos/:id/lineas/:lineaId`
- POST `/api/mis-pedidos/:id/confirmar`

---

10. Tecnologías utilizadas

10.1 Backend

- Node.js
- Express.js
- bcrypt
- jsonwebtoken
- mysql2
- dotenv
- cors
- nodemon

10.2 Base de datos

- MySQL 8 / MariaDB
- phpMyAdmin
- XAMPP

10.3 Frontend

- React + Vite
- React Router

- Fetch (sin librería externa)
- Servicio con Api.js donde se centralizan las peticiones la back.
- Context API / Zustand
- Creación de layout y componentes según funcionalidad.

10.4 Autenticación

- JWT tipo Bearer Token
- Almacenamiento del token en LocalStorage
- Middleware de validación

10.5 Errores y validación

- Middleware centralizado
- Validaciones en controladores

---

11. Checklist final del backend

- Todas las tablas creadas con sus FK.
- `total` con DEFAULT 0.00 y siempre recalculado.
- `nombre` de categoría con UNIQUE.
- Control correcto de productos activos y stock.
- Carrito disponible sin login.
- Confirmación de pedido únicamente con token válido.
- Control de stock al confirmar pedido.
- Subtotales calculados sin almacenarse en BD.

- Validaciones en controlador.
- Arquitectura MVC limpia.
