# MANUAL TÉCNICO DEL SISTEMA

## ÍNDICE

* 1. Introducción
* 2. Objetivos
* 3. Requisitos del Sistema
* 4. Herramientas utilizadas
* 5. Especificación de Requerimientos
* 6. Modelado del Sistema
* 6.1 Diagrama de Casos de Uso
* 6.2 Historias de Usuario
* 6.3 Modelo Entidad-Relación (MER) y Diccionario de Datos
* 8. Instalación y Configuración
* 8.1 Instalación en Servidor Local
* 8.2 Instalación en Servidor Web
* 8.3 Requisitos de Hardware y Software
* 8.4 Configuración del sistema
* 8.5 Despliegue del sistema (Diagrama de Deploy)
* 9. Funcionalidades del sistema
* 10. Resolución de Problemas

## 1. Introducción

El presente manual técnico documenta el desarrollo e implementación del sistema de información “Consejería Distrital de Mujeres Indígenas (CDMI)”, una plataforma web orientada a facilitar la venta de productos artesanales, la gestión de donaciones y la organización de eventos con enfoque social y cultural.

El propósito de este documento es ofrecer una guía detallada y estructurada para los responsables técnicos del sistema, incluyendo desarrolladores, personal de soporte y administradores, que permita comprender el funcionamiento interno del sistema, así como su instalación, configuración, mantenimiento y administración.

Este sistema fue diseñado con el objetivo general de **desarrollar una solución tecnológica orientada a la web que permita al cliente gestionar la venta de productos, donaciones y eventos**, promoviendo la visibilización y valorización del trabajo de las mujeres indígenas.

Los objetivos específicos que guiaron el desarrollo del sistema son:

* **Gestionar la venta de productos artesanales** elaborados por mujeres indígenas, ofreciendo una amplia variedad de artesanías auténticas que reflejan su cultura y tradiciones.
* **Facilitar donaciones monetarias** destinadas a mejorar las condiciones de trabajo de estas mujeres, brindándoles acceso a recursos, capacitación y herramientas que fortalezcan sus habilidades y bienestar.
* **Generar reconocimiento a través de eventos culturales**, que permitan visibilizar su trabajo, concientizar al público sobre su esfuerzo y ofrecer promociones u objetos exclusivos de cada evento.

Este manual se dirige a usuarios con conocimientos técnicos que requieran comprender la estructura del sistema, desde el entorno de desarrollo hasta la interacción entre sus módulos principales. A lo largo del documento se presentan las tecnologías empleadas, la arquitectura general, la configuración del entorno de ejecución, y las instrucciones para el despliegue y mantenimiento de la plataforma.

## 2. Objetivos

### Objetivo General

Desarrollar un sistema de información orientado a la web con el fin de permitir al cliente la gestión de ventas de productos, recepción de donaciones y organización de eventos, promoviendo la cultura indígena y el empoderamiento económico de las mujeres artesanas.

### Objetivos Específicos

1. **Gestionar la venta de productos artesanales** elaborados por mujeres indígenas, ofreciendo una plataforma que permita exhibir una amplia variedad de artesanías auténticas que reflejen su cultura, identidad y tradiciones.
2. **Facilitar la recepción de donaciones monetarias**, destinadas a mejorar las condiciones laborales de las mujeres indígenas, proporcionándoles acceso a recursos, capacitación y herramientas para fortalecer sus habilidades artesanales y su bienestar económico.
3. **Promover el reconocimiento y la concientización** mediante la implementación de eventos culturales, que den a conocer el trabajo, el esfuerzo y la riqueza cultural de las mujeres indígenas, además de brindar al público la oportunidad de participar en promociones y acceder a productos exclusivos.

## 3. Requisitos del Sistema

#### 1. Requisitos Funcionales

Son las funcionalidades que el sistema debe cumplir para satisfacer las necesidades del usuario.

1. El sistema debe permitir el registro y autenticación de usuarios (clientes y administradores).
2. El sistema debe permitir a los usuarios visualizar productos artesanales disponibles para la venta.
3. El sistema debe permitir a los clientes agregar productos a un carrito de compras y realizar pedidos.
4. El sistema debe gestionar la recepción de donaciones mediante un formulario seguro.
5. El sistema debe permitir al administrador registrar, actualizar y eliminar productos, eventos y donaciones.
6. El sistema debe permitir a los usuarios consultar eventos disponibles, su descripción, fechas y productos asociados.
7. El sistema debe generar un historial de compras y donaciones por usuario.
8. El sistema debe permitir que el usuario visualice y edite su perfil.
9. El sistema debe enviar notificaciones o alertas relacionadas con eventos y promociones.
10. El sistema debe generar gráficas e informes para la visualización de estadísticas de ventas, eventos y donaciones (para el administrador).

#### 2. Requisitos No Funcionales

Son características que describen cómo debe comportarse el sistema más allá de sus funciones específicas.

1. El sistema debe estar desarrollado como una aplicación web con backend en Node.js o Laravel y frontend en Flutter Web.
2. El sistema debe utilizar una base de datos MySQL para el almacenamiento de la información.
3. El sistema debe ser accesible desde navegadores web modernos (Chrome, Firefox, Edge).
4. El sistema debe contar con autenticación segura (con manejo de tokens o sesiones).
5. El tiempo de respuesta del sistema no debe superar los 3 segundos en operaciones comunes.
6. El sistema debe estar disponible al menos el 99% del tiempo (alta disponibilidad).
7. El sistema debe permitir la escalabilidad futura, tanto en funcionalidades como en usuarios.
8. El sistema debe tener una interfaz amigable, intuitiva y adaptada a los lineamientos gráficos del proyecto (colores vino tinto, blanco, imágenes indígenas, logo).
9. La información sensible del usuario debe estar protegida mediante cifrado y validaciones.
10. El sistema debe cumplir con principios de usabilidad y accesibilidad para facilitar su uso por parte de la comunidad objetivo.

## 4. Herramientas utilizadas

A continuación se describen las principales herramientas, lenguajes y tecnologías empleadas en el desarrollo del sistema de información "Consejería Distrital de Mujeres Indígenas (CDMI)":

#### Frontend (Aplicación Web y App en React)

* **React.js** Biblioteca JavaScript para la construcción de interfaces de usuario dinámicas y componentes reutilizables.
* **JavaScript (ES6+)** Lenguaje de programación utilizado para la lógica del frontend y manipulación de datos.
* **HTML5 y CSS3** Lenguajes estándar para la estructura y el diseño visual de las interfaces.
* **React Router** Librería para la gestión de rutas y navegación entre vistas dentro de la aplicación React.
* **Axios** Cliente HTTP utilizado para realizar peticiones a la API REST del backend.
* **Bootstrap y/o Tailwind CSS** Frameworks de diseño y estilos para construir interfaces responsivas y visualmente coherentes.

#### Backend (API RESTful)

* **Node.js** Entorno de ejecución de JavaScript en el servidor.
* **Express.js** Framework minimalista de Node.js para la creación de APIs RESTful.
* **MySQL** Sistema de gestión de bases de datos relacional utilizado para almacenar información de usuarios, productos, eventos, ventas y donaciones.
* **Sequelize o Knex.js** (según lo que hayas usado)  
   ORM para la interacción entre Node.js y la base de datos MySQL.
* **JWT (JSON Web Tokens)** Herramienta para la autenticación y autorización segura de usuarios en el sistema.

#### Entorno de Desarrollo

* **Visual Studio Code** Editor de código fuente utilizado para el desarrollo tanto del frontend como del backend.
* **Postman / Thunder Client** Herramientas para pruebas y validación de endpoints de la API.
* **Git y GitHub** Sistema de control de versiones y repositorio remoto para la colaboración y respaldo del código fuente.
* **XAMPP o MySQL Workbench** Herramientas para la gestión local de la base de datos durante el desarrollo.

## 5. Especificación de Requerimientos

#### 1. Requerimientos Funcionales

| **Código** | **Requerimiento** | **Descripción** |
| --- | --- | --- |
| RF-01 | Registro de usuarios | El sistema debe permitir que nuevos usuarios se registren como cliente o administrador. |
| RF-02 | Inicio de sesión | El sistema debe permitir el inicio de sesión con validación de credenciales. |
| RF-03 | Gestión de productos | El administrador puede crear, editar, eliminar y listar productos artesanales. |
| RF-04 | Visualización de productos | El cliente puede visualizar una galería de productos disponibles para la compra. |
| RF-05 | Carrito de compras | El cliente puede agregar productos al carrito y proceder con la compra. |
| RF-06 | Gestión de eventos | El administrador puede registrar eventos culturales con fecha, hora, descripción y productos asociados. |
| RF-07 | Visualización de eventos | Los usuarios pueden consultar eventos disponibles y su información detallada. |
| RF-08 | Donaciones | Los usuarios pueden realizar donaciones mediante un formulario y monto definido. |
| RF-09 | Historial de compras y donaciones | Los usuarios pueden ver su historial de compras y donaciones realizadas. |
| RF-10 | Gestión de perfil | Los usuarios pueden consultar, editar o eliminar la información de su perfil. |
| RF-11 | Gráficas estadísticas | El administrador puede visualizar gráficas de ventas, donaciones y asistencia a eventos. |

#### 2. Requerimientos No Funcionales

| **Código** | **Requerimiento** | **Descripción** |
| --- | --- | --- |
| RNF-01 | Seguridad | El sistema debe cifrar contraseñas y proteger datos sensibles. Se utiliza JWT para autenticación. |
| RNF-02 | Usabilidad | La interfaz debe ser intuitiva y accesible para los diferentes perfiles de usuario. |
| RNF-03 | Rendimiento | Las respuestas del sistema no deben superar los 3 segundos en operaciones estándar. |
| RNF-04 | Compatibilidad | La aplicación debe ser accesible desde navegadores modernos y dispositivos móviles. |
| RNF-05 | Mantenibilidad | El código debe estar modularizado y documentado para facilitar futuras modificaciones. |
| RNF-06 | Escalabilidad | El sistema debe permitir agregar nuevos módulos sin afectar el funcionamiento existente. |
| RNF-07 | Disponibilidad | El sistema debe estar disponible al menos el 99% del tiempo, excepto en mantenimientos programados. |

#### 3. Requerimientos de Usuarios

| **Código** | **Requerimiento** | **Descripción** |
| --- | --- | --- |
| RU-01 | Rol de Cliente | Puede registrarse, iniciar sesión, ver productos, donar, participar en eventos y gestionar su perfil. |
| RU-02 | Rol de Administrador | Puede gestionar productos, eventos, usuarios, donaciones y visualizar estadísticas. |

## 6. Modelado del Sistema

## 6.1 Diagrama de Casos de Uso

**ADMINISTRADOR**

| Registro/Crear cuenta | | | |
| --- | --- | --- | --- |
| Administrador | | Sistema | |
| 1 | Ingresa la información | 2 | Registrar acciones realizadas |
| 3 | Crea la cuenta | 4 | Gestionar y revisar esos registros |
|  |  | 5 | Guarda la información del registro |

| Iniciar sesión | | | |
| --- | --- | --- | --- |
| Administrador | | Sistema | |
| 1 | Ingresa correo y contraseña | 2 | Verifica la información |
| 3 | Ingresa | 4 | Vista de la página |

**USUARIO.**

| Crear cuenta | | | |
| --- | --- | --- | --- |
| Cliente | | Sistema | |
| 1 | Ingresa la información | 2 | Verifica la información |
| 3 | Crea la cuenta | 4 | Guarda la información |

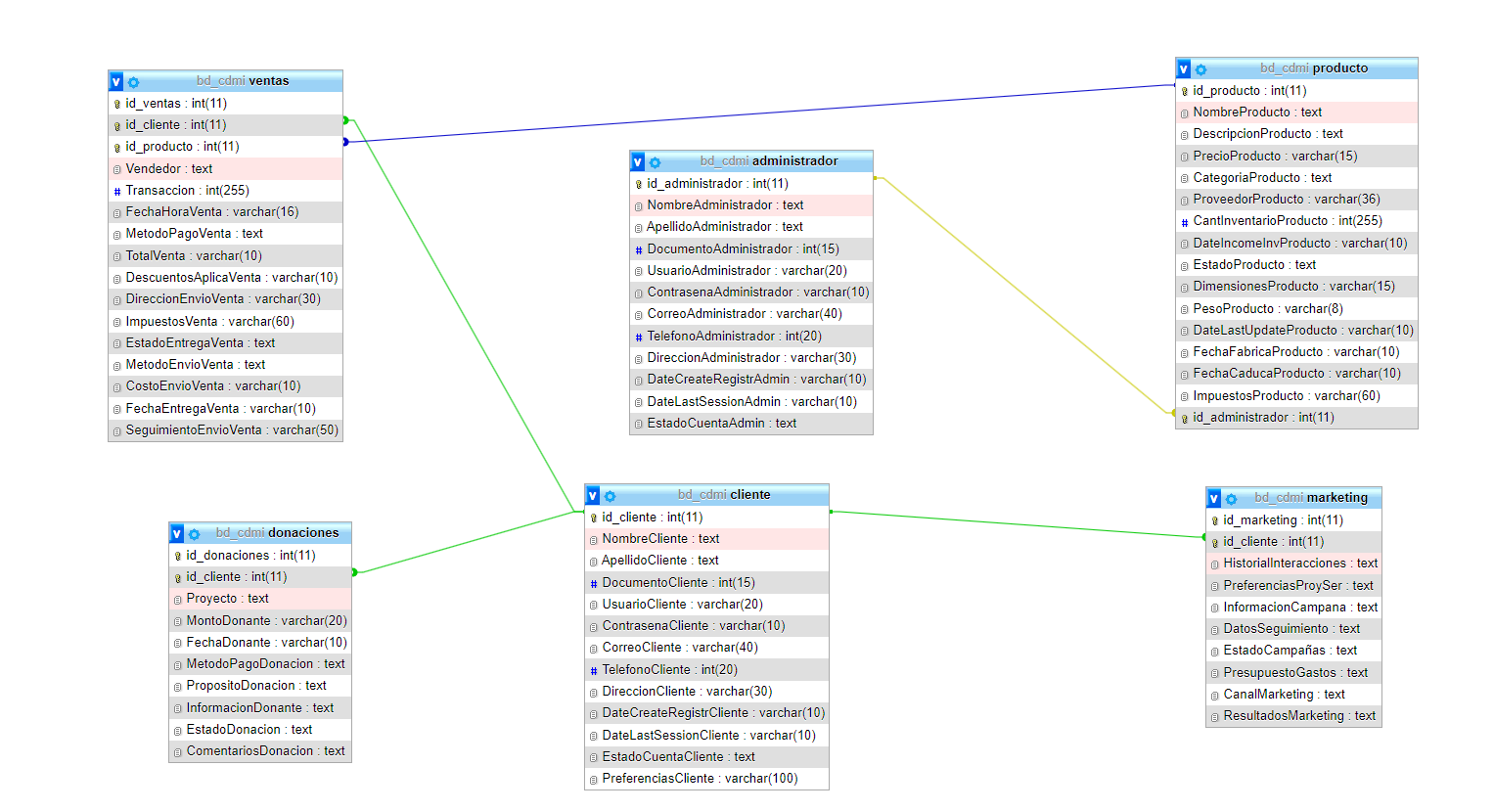
| Login | | | |
| --- | --- | --- | --- |
| Cliente | | Sistema | |
| 1 | Ingresa correo y contraseña | 2 | Valida la información |
| 3 | Ingresa | 4 | Vista de funciones |

[Caso de uso extendido.docx](https://docs.google.com/document/d/14LVOkPGaCkeIXrPJb5wT1hcnMce9fO1H/edit?usp=sharing&ouid=104222312880139260030&rtpof=true&sd=true)

## 6.2 Historias de Usuario

## 6.3 Modelo Entidad-Relación (MER) y Diccionario de Datos

MER:



[6. MODELO ENTIDAD RELACION CON NOTACION CROWS FOOT](https://drive.google.com/drive/folders/1T6mZNP9eEaK5pN5B9Vk5ryBB9kTZmJCm?usp=drive_link)

Diccionario de datos:

| PRODUCTOS | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| COLUMN NAME | DATATYPE | LENTGH | RELATION | | | MANDATORY | RESTRICTION | DESCRIPTION | UPDATE |
| PK | NN | FK |
| IdProducto | INT | 6 | X |  |  | YES | Registered value | ID único del producto | Unspecified |
| NombreProducto | TEXT | 30 |  | X |  | YES | NO | Nombre del producto | Unspecified |
| DescripcionProducto | TEXT | 30 |  | X |  | YES | NO | Descripción del producto | Unspecified |
| PrecioProducto | VARCHAR | 15 |  | X |  | YES | NO | Precio del producto | Unspecified |
| CategoriaProducto | TEXT | 20 |  | X |  | NO | NO | Categoría a la que pertenece el producto | Unspecified |
| ProveedorProducto | VARCHAR | 36 |  | X |  | NO | NO | Proveedor o fabricante del producto | Unspecified |
| CantInventarioProducto | INT | 9000 |  | X |  | NO | NO | Cantidad del producto en el inventario | Unspecified |
| DateIncomeInvProducto | VARCHAR | 10 |  | X |  | NO | NO | Fecha de ingreso del producto al inventario | Unspecified |
| EstadoProducto | TEXT | 10 |  | X |  | YES | Activo, Inactivo, Bloqueado | Estado del producto | Unspecified |
| DimensionesProducto | VARCHAR | 15 |  | X |  | NO | Measurement units | Dimensiones del producto (ancho, alto, largo) | Unspecified |
| PesoProducto | VARCHAR | 8 |  | X |  | NO | Measurement units | Peso del producto | Unspecified |
| DateLastUpdateProducto | VARCHAR | 10 |  | X |  | NO | NO | Fecha de última actualización del producto | Unspecified |
| FechaFabricaProducto | VARCHAR | 10 |  | X |  | YES | NO | Fecha de fabricación del producto | Unspecified |
| FechaCaducaProducto | VARCHAR | 10 |  | X |  | YES | NO | Fecha de caducidad del producto | Unspecified |
| ImpuestosProducto | VARCHAR | 60 |  | X |  | NO | NO | Impuestos aplicables al producto | Unspecified |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| VENTAS | | | | | | | | | |
| COLUMN NAME | DATATYPE | LENTGH | RELATION | | | MANDATORY | RESTRICTION | DESCRIPTION | UPDATE |
| PK | NN | FK |
| IdVenta | INT | 15 | X |  |  | YES | NO | ID único de la venta | Unspecified |
| IdCliente | INT | 6 |  |  | X | YES | NO | ID único del cliente | Unspecified |
| IdProducto | INT | 6 |  |  | X | YES | Registered value | ID único del producto | Unspecified |
| Vendedor | INT | 6 |  | x |  | YES | Registered value | ID único del vendedor | Unspecified |
| Transaccion | INT | 22 |  | x |  | YES | NO | ID único de la transacción | Unspecified |
| FechaHoraVenta | VARCHAR | 16 |  | X |  | YES | NO | Proveedor o fabricante del producto | Unspecified |
| MetodoPagoVenta | TEXT | 10 |  | X |  | YES | NO | Cantidad del producto en el inventario | Unspecified |
| TotalVenta | VARCHAR | 10 |  | X |  | YES | NO | Fecha de ingreso del producto al inventario | Unspecified |
| DescuentosAplicaVenta | VARCHAR | 10 |  | X |  | YES | NO | Estado del producto | Unspecified |
| DireccionEnvioVenta | VARCHAR | 30 |  | X |  | NO | NO | Dimensiones del producto (ancho, alto, largo) | Unspecified |
| ImpuestosVenta | VARCHAR | 60 |  | X |  | NO | NO | Peso del producto | Unspecified |
| EstadoEntregaVenta | TEXT | 10 |  | X |  | YES | Activo, Inactivo, Bloqueado | Fecha de última actualización del producto | Unspecified |
| MetodoEnvioVenta | TEXT | 17 |  | X |  | NO | NO | Fecha de fabricación del producto | Unspecified |
| CostoEnvioVenta | VARCHAR | 10 |  | X |  | NO | NO | Fecha de caducidad del producto | Unspecified |
| FechaEntregaVenta | VARCHAR | 10 |  | X |  | YES | NO | Impuestos aplicables al producto | Unspecified |
| SeguimientoEnvioVenta | VARCHAR | 50 |  | X |  | NO | NO |  | Unspecified |

[DICCIONARIO DE DATOS BASE DE DATOS CDMI.xlsx](https://docs.google.com/spreadsheets/d/1Gj8sKvs_yCBzoKNx5LNyWUCjUCgHIB8o/edit?usp=sharing&ouid=104222312880139260030&rtpof=true&sd=true)

## 7. Instalación y Configuración

## 7.1 Instalación en Servidor Local

Esta sección describe los pasos necesarios para instalar y ejecutar el sistema CDMI en un entorno local para desarrollo o pruebas.

#### 📦 1. Requisitos Previos

* Node.js v14 o superior
* npm o yarn
* MySQL Server (XAMPP, MySQL Workbench o independiente)
* Git (opcional, para clonar repositorios)
* Editor de código (Visual Studio Code recomendado)
* Navegador web moderno (Chrome, Firefox)

#### ⚙️ 2. Instalación del Backend (API REST)

**Clonar o descargar el proyecto backend** bash  
CopiarEditar  
git clone https://github.com/usuario/cdmi-backend.git

cd cdmi-backend

**Instalar dependencias** bash  
CopiarEditar  
npm install

**Configurar archivo .env** Crea un archivo .env en la raíz del proyecto con las variables necesarias:  
  
 env  
CopiarEditar  
PORT=3000

DB\_HOST=localhost

DB\_USER=root

DB\_PASSWORD=tu\_clave\_mysql

DB\_NAME=cdmi\_bd

JWT\_SECRET=clave\_secreta

1. **Crear la base de datos** Importa el archivo cdmi\_bd.sql en tu gestor MySQL (phpMyAdmin o MySQL Workbench).

**Iniciar el servidor** bash  
CopiarEditar  
npm start

1. La API estará disponible en http://localhost:3000

#### 🌐 3. Instalación del Frontend (React - Web/App)

**Clonar o descargar el proyecto frontend** bash  
CopiarEditar  
git clone https://github.com/usuario/cdmi-frontend.git

cd cdmi-frontend

**Instalar dependencias** bash  
CopiarEditar  
npm install

**Configurar archivo .env** Crea un archivo .env en la raíz del frontend:  
  
 env  
CopiarEditar  
REACT\_APP\_API\_URL=http://localhost:3000

**Iniciar el servidor de desarrollo** bash  
CopiarEditar  
npm start

1. La aplicación estará disponible en http://localhost:3000 o http://localhost:5173 (según el puerto usado).

#### Notas Adicionales

* Asegúrate de que el puerto del backend no esté ocupado por otro servicio.
* Si usas XAMPP, asegúrate de que Apache y MySQL estén activos.
* Puedes usar herramientas como **Thunder Client** o **Postman** para probar los endpoints de la API.

## 8.2 Instalación en Servidor Web

Esta sección describe el procedimiento necesario para desplegar el sistema CDMI en un servidor web (Linux), con el fin de permitir su acceso desde internet o una red interna.

#### 📋 Requisitos del Servidor

* Sistema operativo: Linux (Ubuntu 20.04+ recomendado)
* Acceso SSH al servidor (usuario root o con permisos sudo)
* Servidor web: Nginx o Apache
* Node.js v14 o superior
* npm o yarn
* MySQL Server
* Certificado SSL (Let’s Encrypt recomendado)
* PM2 (para mantener el backend activo)
* Git (opcional)

#### ⚙️ 1. Instalación del Backend (API en Node.js)

**Conectarse al servidor:** bash  
CopiarEditar  
ssh usuario@tu\_dominio\_o\_ip

**Clonar el proyecto o subirlo al servidor:** bash  
CopiarEditar  
git clone https://github.com/usuario/cdmi-backend.git

cd cdmi-backend

**Instalar dependencias:** bash  
CopiarEditar  
npm install

**Configurar variables de entorno:** Crear el archivo .env con los datos reales de producción:  
  
 env  
CopiarEditar  
PORT=3000

DB\_HOST=localhost

DB\_USER=usuario\_mysql

DB\_PASSWORD=clave\_segura

DB\_NAME=cdmi\_bd

JWT\_SECRET=clave\_super\_segura

**Importar base de datos:** Importar el archivo SQL (cdmi\_bd.sql) con:  
  
 bash  
CopiarEditar  
mysql -u usuario\_mysql -p cdmi\_bd < ruta/cdmi\_bd.sql

**Iniciar el backend con PM2 (para producción):** bash  
CopiarEditar  
npm install -g pm2

pm2 start index.js --name cdmi-api

pm2 save

pm2 startup

**Configurar proxy inverso con Nginx (ejemplo):** nginx  
CopiarEditar  
server {

listen 80;

server\_name tu\_dominio.com;

location /api/ {

proxy\_pass http://localhost:3000/;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

}

#### 🌐 2. Instalación del Frontend (React)

**Subir el proyecto React:** bash  
CopiarEditar  
git clone https://github.com/usuario/cdmi-frontend.git

cd cdmi-frontend

**Configurar el entorno:** Crear .env.production:  
  
 env  
CopiarEditar  
REACT\_APP\_API\_URL=https://tu\_dominio.com/api

**Construir la aplicación:** bash  
CopiarEditar  
npm install

npm run build

**Subir la carpeta build/ a un servidor web:** Si usas Nginx:  
  
 nginx  
CopiarEditar  
server {

listen 80;

server\_name tu\_dominio.com;

root /var/www/html/cdmi-frontend/build;

index index.html index.htm;

location / {

try\_files $uri /index.html;

}

}

#### 🔐 3. (Opcional) Configurar HTTPS con Let's Encrypt

**Instalar Certbot:** bash  
CopiarEditar  
sudo apt install certbot python3-certbot-nginx

**Obtener certificado SSL:** bash  
CopiarEditar  
sudo certbot --nginx -d tu\_dominio.com

#### ✅ Verificación Final

* Accede a: https://tu\_dominio.com
* La app React se debe cargar correctamente.
* Las peticiones a /api deben estar funcionando.

## 8.3 Requisitos de Hardware y Software

Esta sección detalla las especificaciones mínimas y recomendadas necesarias para el correcto funcionamiento del sistema CDMI, tanto en entornos de desarrollo como de producción.

#### 🖥️ Servidor Web / Producción

| **Recurso** | **Requisito Mínimo** | **Recomendado** |
| --- | --- | --- |
| **Procesador (CPU)** | 2 núcleos (Intel/AMD 64 bits) | 4 núcleos o más |
| **Memoria RAM** | 2 GB | 4 GB o más |
| **Disco Duro** | 10 GB de espacio libre | SSD con al menos 20 GB libres |
| **Sistema Operativo** | Ubuntu 20.04+, Debian 10+, CentOS 7+ | Ubuntu Server 22.04 LTS |
| **Base de Datos** | MySQL 5.7 o superior | MySQL 8.0 |
| **Servidor Web** | Nginx o Apache | Nginx |
| **Node.js** | Versión 14 o superior | Versión LTS actual (18 o superior) |
| **Otros** | PM2, Certbot, Git, Firewall configurado | Docker (opcional para contenerización) |

#### 👨‍💻 Estación de Trabajo / Desarrollo

| **Recurso** | **Requisito Mínimo** | **Recomendado** |
| --- | --- | --- |
| **Procesador (CPU)** | Intel Core i3 / AMD equivalente | Intel Core i5 o superior |
| **Memoria RAM** | 4 GB | 8 GB o más |
| **Disco Duro** | 5 GB de espacio libre | SSD con al menos 10 GB libres |
| **Sistema Operativo** | Windows 10, macOS 10.13+, Ubuntu 20.04+ | Windows 11, macOS Ventura, Ubuntu 22.04+ |
| **Node.js** | Versión 14 o superior | Versión LTS actual |
| **Gestor de Paquetes** | npm o yarn | npm |
| **Base de Datos** | MySQL (XAMPP, WAMP, MySQL Workbench) | MySQL Workbench + phpMyAdmin |
| **Navegador** | Google Chrome, Firefox | Google Chrome actualizado |
| **Editor de Código** | Visual Studio Code | Visual Studio Code con extensiones útiles |

#### 🧰 Software Complementario

* **Thunder Client o Postman** (para probar la API)
* **Git** (para control de versiones)
* **Insomnia** (opcional, para pruebas de API)
* **Docker** (opcional, para contenedores)
* **Certbot** (para certificados SSL en producción)

## 8.4 Configuración del sistema

Esta sección describe los pasos necesarios para configurar correctamente el sistema CDMI una vez instalado en el servidor, tanto en el entorno backend como frontend. Asegura que todas las rutas, servicios y conexiones estén funcionando según lo esperado.

#### 1. 📁 Estructura General del Proyecto

**Backend (Node.js):**

pgsql

CopiarEditar

cdmi-backend/

├── controllers/

├── middlewares/

├── models/

├── routes/

├── .env

├── index.js

└── package.json

**Frontend (React):**

pgsql

CopiarEditar

cdmi-frontend/

├── public/

├── src/

│ ├── components/

│ ├── pages/

│ ├── services/

│ ├── App.js

│ └── index.js

├── .env

└── package.json

#### 2. 🌐 Configuración del Backend (API)

**Archivo .env:** Dentro de cdmi-backend/ crear el archivo .env con los siguientes datos:  
  
 env  
CopiarEditar  
PORT=3000

DB\_HOST=localhost

DB\_USER=tu\_usuario\_mysql

DB\_PASSWORD=tu\_contraseña

DB\_NAME=cdmi\_bd

JWT\_SECRET=clave\_secreta\_segura

1. **Base de datos MySQL:**
   * Verifica que la base de datos cdmi\_bd esté correctamente importada.
   * Asegúrate de que las tablas estén normalizadas y relacionadas correctamente.

**Ejecución del servidor:** bash  
CopiarEditar  
npm install

npm start

**PM2 en producción:** bash  
CopiarEditar  
pm2 start index.js --name cdmi-api

pm2 save

#### 3. 🌍 Configuración del Frontend (React)

**Archivo .env:** Dentro de cdmi-frontend/ crear el archivo .env o .env.production con:  
  
 env  
CopiarEditar  
REACT\_APP\_API\_URL=https://tu\_dominio.com/api

1. **Ajustar endpoints en servicios (src/services):** Asegúrate de que todas las peticiones fetch o axios usen la variable de entorno REACT\_APP\_API\_URL.

**Build de producción:** bash  
CopiarEditar  
npm run build

1. **Servir con Nginx o Apache:**
   * Subir la carpeta build/ al servidor.
   * Configurar el servidor web para redirigir todas las rutas a index.html (Single Page App).

#### 4. 🔐 Configuración de Seguridad

* **CORS** en backend configurado para aceptar solicitudes desde el frontend.
* **Protección de rutas** usando JWT en el backend.
* **Middleware de autenticación** para rutas protegidas.
* **HTTPS** con Let's Encrypt (en producción).

#### 5. 📡 Verificación del Sistema

* Probar conexión entre frontend y backend.
* Acceder con un usuario registrado y verificar JWT.
* Realizar operaciones CRUD (productos, eventos, donaciones).
* Probar funcionalidades de roles: administrador y cliente.

## 8.5 Despliegue del sistema (Diagrama de Deploy)

El despliegue del sistema CDMI se realiza bajo una arquitectura cliente-servidor, donde el frontend en React consume los servicios proporcionados por una API REST desarrollada en Node.js. La persistencia de datos se maneja a través de una base de datos MySQL. El sistema puede ser desplegado tanto en servidores locales como en servidores web (hosting o VPS).

#### 🧱 Arquitectura General del Despliegue

**Componentes involucrados:**

1. **Cliente (Navegador Web)**
   * Accede a la interfaz desarrollada en React.
   * Envía solicitudes HTTP (GET, POST, PUT, DELETE) a la API.
2. **Frontend (ReactJS)**
   * Desarrollado como SPA (Single Page Application).
   * Compilado y desplegado en un servidor web (Nginx o Apache).
3. **Backend (Node.js + Express)**
   * Expone rutas para autenticación, productos, eventos, donaciones, etc.
   * Valida tokens JWT y roles de usuario.
   * Comunica con la base de datos MySQL.
4. **Base de Datos (MySQL)**
   * Guarda información de usuarios, productos, eventos, donaciones, ventas, etc.

#### 📊 Diagrama de Despliegue (lógico)

scss

CopiarEditar

┌───────────────────────────┐

│ Usuario final │

│ (Navegador Web) │

└────────────┬──────────────┘

│

▼

┌───────────────────────────┐

│ Servidor Web (Nginx) │

│ [Frontend - ReactJS] │

└────────────┬──────────────┘

│

Solicitudes HTTP (API)

│

▼

┌────────────────────────────────┐

│ Servidor Backend (Node.js) │

│ - API REST con Express │

│ - Middleware JWT │

└────────────┬───────────────────┘

│

Consultas SQL (CRUD)

│

▼

┌────────────────────────┐

│ MySQL Database │

│ (cdmi\_bd / tablas) │

└────────────────────────┘

#### 🌐 Ejemplo de Tecnologías Utilizadas

| **Componente** | **Tecnología** |
| --- | --- |
| Cliente | React (SPA) |
| Servidor Web | Nginx / Apache |
| Servidor Backend | Node.js + Express |
| Base de Datos | MySQL |
| Autenticación | JWT |
| Seguridad Web | HTTPS (Certbot) |

## 9. Funcionalidades del sistema

El sistema CDMI está diseñado para facilitar la gestión integral de la venta de productos artesanales, donaciones y eventos culturales, con un enfoque especial en promover y apoyar a las mujeres indígenas. A continuación, se describen las funcionalidades principales:

#### 1. Gestión de Productos Artesanales

* **Visualización de productos:** Los usuarios pueden navegar y visualizar una amplia variedad de artesanías auténticas elaboradas por mujeres indígenas, incluyendo imágenes, descripciones y precios.
* **Registro y administración de productos (Administrador):** Los administradores pueden crear, editar y eliminar productos, gestionando la información y stock disponible.
* **Carrito de compras:** Los clientes pueden agregar productos al carrito, modificar cantidades y proceder a la compra.
* **Procesamiento de ventas:** El sistema registra las ventas realizadas, generando un historial para cada usuario.

#### 2. Gestión de Donaciones

* **Donaciones monetarias:** Los usuarios pueden realizar donaciones económicas destinadas a mejorar las condiciones laborales de las mujeres indígenas.
* **Registro y seguimiento:** Las donaciones se registran en el sistema para mantener un control y generar reportes.

#### 3. Gestión de Eventos Culturales

* **Visualización de eventos:** Los usuarios pueden consultar eventos programados que promueven la cultura indígena y las artesanías.
* **Registro a eventos:** Los clientes pueden inscribirse en eventos para participar o recibir información.
* **Administración de eventos (Administrador):** Creación, edición y eliminación de eventos, incluyendo detalles como fecha, lugar y promociones exclusivas.

#### 4. Sistema de Usuarios

* **Registro y autenticación:** Usuarios y administradores pueden registrarse y acceder mediante un sistema seguro de inicio de sesión con autenticación JWT.
* **Roles y permisos:** El sistema distingue entre roles de cliente y administrador, permitiendo acciones específicas según el rol asignado.

#### 5. Seguridad y Protección

* **Autenticación con JWT:** Garantiza sesiones seguras y protección de rutas sensibles.
* **Validación de datos:** Formularios con validación para garantizar integridad y evitar errores.

#### 6. Reportes y Estadísticas (Opcional)

* Generación de reportes sobre ventas, donaciones y participación en eventos para la toma de decisiones.

## 10. Resolución de Problemas

Esta sección proporciona soluciones a problemas comunes que pueden surgir durante la instalación, configuración y uso del sistema CDMI.

#### 1. Problema: No se puede conectar a la base de datos MySQL

* **Síntomas:** El backend no inicia o lanza errores al intentar establecer conexión con la base de datos.
* **Posibles causas:**
  + Credenciales incorrectas en el archivo .env.
  + Servicio de MySQL no está corriendo.
  + Base de datos no creada o faltan tablas.
  + Firewall bloqueando la conexión.
* **Solución:**
  + Verificar usuario, contraseña, host y nombre de la base de datos en .env.
  + Asegurarse que el servicio MySQL esté activo (sudo service mysql status o equivalente).
  + Importar el archivo SQL con la estructura y datos necesarios.
  + Configurar reglas de firewall para permitir conexiones locales o remotas según sea necesario.

#### 2. Problema: El frontend no carga o muestra pantalla en blanco

* **Síntomas:** Al acceder a la URL del frontend, la página no carga o aparece en blanco.
* **Posibles causas:**
  + El build no fue generado o desplegado correctamente.
  + La ruta base (basename) está mal configurada.
  + Error en las variables de entorno, especialmente en la URL del API.
  + Problemas con la configuración del servidor web (Nginx, Apache).
* **Solución:**
  + Ejecutar npm run build en el proyecto React y subir la carpeta build correctamente.
  + Verificar configuración en el archivo .env y en el código fuente para usar correctamente REACT\_APP\_API\_URL.
  + Comprobar que el servidor web está configurado para servir archivos estáticos y redirigir todas las rutas a index.html.

#### 3. Problema: Error de autenticación o token inválido

* **Síntomas:** No se puede iniciar sesión o las rutas protegidas retornan error 401 Unauthorized.
* **Posibles causas:**
  + Clave secreta JWT diferente entre backend y generación de tokens.
  + Token expirado o mal formado.
  + Cabeceras HTTP mal configuradas al hacer peticiones desde frontend.
* **Solución:**
  + Confirmar que la variable JWT\_SECRET en .env es la misma en el backend.
  + Asegurarse que el frontend envía el token en el header Authorization: Bearer <token>.
  + Implementar refresco de token o re-login si el token está expirado.

#### 4. Problema: No se pueden registrar productos, eventos o donaciones

* **Síntomas:** Al intentar agregar o modificar datos desde la interfaz, la acción falla o no se refleja en la base de datos.
* **Posibles causas:**
  + Problemas de validación en el backend o frontend.
  + Error en las rutas API o métodos HTTP incorrectos.
  + Restricciones de permisos por roles.
* **Solución:**
  + Revisar logs del backend para detectar errores.
  + Verificar que el usuario tenga rol administrador para acciones restringidas.
  + Confirmar que los datos enviados cumplen con el formato esperado.

#### 5. Problema: Problemas con CORS

* **Síntomas:** El navegador bloquea solicitudes al backend con error relacionado a CORS.
* **Posibles causas:**
  + Falta de configuración adecuada en el backend para permitir peticiones desde el dominio del frontend.
* **Solución:**
  + Configurar middleware CORS en Express con el origen del frontend permitido, por ejemplo:

javascript  
CopiarEditar  
const cors = require('cors');

app.use(cors({

origin: 'http://localhost:3000' // o URL de producción

}));

#### 6. Problema: Servidor backend se cae o se reinicia constantemente

* **Síntomas:** El servidor Node.js se detiene inesperadamente o reinicia.
* **Posibles causas:**
  + Errores no capturados en el código.
  + Falta de manejo de excepciones.
  + Recursos del servidor insuficientes.
* **Solución:**
  + Revisar los logs para identificar el error.
  + Añadir manejo de errores con try/catch.
  + Utilizar un gestor de procesos como PM2 para reiniciar automáticamente el servidor.