MecaniSoft - Manuales

Autor: Chumbes Espinoza Marco Antonio

Contenidos:

- manual_instalacion.md
- manual_docker.md

manual_instalacion.md

Autor: Chumbes Espinoza Marco Antonio

MecaniSoft - Manual de Instalación Rápida

Autor: CHUMBES ESPINOZA MARCO ANTONIO

Este documento acompaña el ZIP que se comparte. Siga los pasos en orden para levantar **MecaniSoft** desde cero. Todo lo que no se mencione es opcional.

Guía exprés (copiar y pegar)

```
# 1) Instalar dependencias del proyecto
npm install
# 2) Configurar la base de datos (crea DB, aplica migraciones y semillas)
npx prisma migrate reset --force
npm run seed
npx tsx scripts/create-admin-user.ts
npx tsx scripts/seed-sample-data.ts   # opcional: carga datos demo completos
# 3) Levantar el servidor de desarrollo
npm run dev
```

Luego abra `http://localhost:3000` e ingrese con `admin.pruebas` / `Admin123!`.

1. Requisitos previos

1. **Node.js 20 LTS** (incluye `npm`). Descárguelo de nodejs.org e instale con las opciones por defecto.
2. **Git** (opcional). Solo se necesita si desea clonar; para el ZIP no es necesario.
3. **PostgreSQL 16** o compatible.
   - Descarga desde [postgresql.org/download](https://www.postgresql.org/download).
   - Durante la instalación tome nota de usuario (`postgres`), contraseña y puerto (`5432`).
   - Cree la base vacía `taller_mecanico` con pgAdmin o en consola:
     ```powershell
     createdb -U postgres -h localhost -p 5432 taller_mecanico
     ```
     Ajuste los parámetros si cambió usuario o puerto.
4. **Docker Desktop** solo es necesario si prefiere usar contenedores (ver sección 10).

2. Obtener el proyecto

1. Descomprima el archivo ZIP en cualquier carpeta (ej. `C:\MecaniSoft`).
2. Abra PowerShell y ejecute `cd C:\MecaniSoft` para ubicarse en la raíz del proyecto.

3. Instalar dependencias

Ejecute una sola vez:

```
npm install
```

Nota: este comando también ejecuta `prisma generate` automáticamente gracias al `postinstall` del proyecto.

4. Configurar variables de entorno

1. Copie `.env.example` a `.env` (si el archivo de ejemplo no está, créelo a mano).
2. Actualice como mínimo estas claves:

```
DATABASE_URL="postgresql://USUARIO:CONTRASENA@localhost:5432/taller_mecanico?schema=public"
```

```
NEXTAUTH_SECRET="clave-larga-aleatoria"
NEXTAUTH_URL="http://localhost:3000"
```

Reemplace `USUARIO` y `CONTRASENA` por los datos configurados en PostgreSQL.

5. Preparar la base de datos

En la misma terminal, ejecute en orden:

```
npx prisma migrate reset --force   # limpia la base y aplica todas las migraciones
npm run seed                       # inserta catálogos, permisos y parámetros mínimos
npx tsx scripts/create-admin-user.ts
npx tsx scripts/seed-sample-data.ts # opcional: genera datos realistas para probar
```

Puede omitir el último comando si desea comenzar con una base limpia.

6. Levantar el servidor

```
npm run dev
```

Abra `http://localhost:3000` en el navegador. El servidor queda activo hasta que cierre la terminal (Ctrl+C).

7. Credenciales sugeridas

Los scripts crean usuarios listos para probar:

Rol	Usuario	Contraseña
Administrador	`admin.pruebas`	`Admin123!`
Recepción	`recepcion.lucia`	`Taller123!`
Mecánico senior	`mecanico.jorge`	`Taller123!`
Diagnóstico	`mecanico.sofia`	`Taller123!`

Puede crear más usuarios desde el módulo de administración utilizando la misma interfaz.

8. Verificación y pruebas rápidas

1. Inicie sesión como `admin.pruebas`.
2. Revise el dashboard y abra los módulos de Órdenes, Inventario y Reportes para confirmar que los datos demo se cargaron.
3. Si algo luce desactualizado, vuelva a ejecutar `npx tsx scripts/seed-sample-data.ts`.

9. Problemas frecuentes

- **No conecta a PostgreSQL:** confirme que el servicio está iniciado, que la base `taller_mecanico` existe y que `DATABASE_URL` tiene usuario/contraseña correctos.
- **Puerto 3000 ocupado:** cierre la aplicación que use ese puerto o ejecute `setx PORT 4000` antes de `npm run dev` y luego ingrese a `http://localhost:4000`.
- **Migraciones fallidas:** ejecute de nuevo `npx prisma migrate reset --force`. Si la DB quedó en un estado incorrecto, elimínela y créela otra vez.
- **Dependencias nativas no compilan:** asegúrese de estar en Node.js 20 LTS, reinstale (`npm install --force`) y vuelva a probar.

10. Alternativa con Docker

Para evitar instalar Node.js y PostgreSQL localmente:

1. Instale Docker Desktop y asegúrese de que esté activo.
2. Desde la carpeta del proyecto ejecute:
   ```powershell
   docker compose up --build
   ```
3. Esto levanta dos contenedores (`web` y `db`). La aplicación queda disponible en `http://localhost:3000` una vez que `npm run start` comience a servir.
4. Inicialice la base por única vez (solo la primera vez que levanta los contenedores):
   ```powershell
   docker compose exec web npm run seed
   docker compose exec web npx tsx scripts/create-admin-user.ts
   ```
5. Opcional: cargue datos demo completos dentro del contenedor `web` con:
   ```powershell
   docker compose exec web npx tsx scripts/seed-sample-data.ts
   ```
6. Detenga todo con `docker compose down`. Use `docker compose down -v` si desea borrar la base persistida.

Las credenciales y la `DATABASE_URL` usadas por Docker están definidas en `docker-compose.yml`; modifíquelas si migrará a un entorno público.

Redis opcional

El proyecto incluye `docker-compose.redis.yml` para lanzar un Redis 7 usado por tareas de cola/cache. Si necesita probar esas funcionalidades, ejecútelo aparte:

```
docker compose -f docker-compose.redis.yml up -d
```

Puede detenerlo con `docker compose -f docker-compose.redis.yml down` cuando termine.# Manual de Puesta en Marcha

Este manual explica cómo levantar el sistema **MecaniSoft** en una máquina que no tiene las herramientas instaladas. Sigue cada sección en orden.

Acceso demo pública: https://sistema-mecanisoft-z2td.vercel.app/

1. Programas necesarios

1. **Node.js 20 LTS** (incluye `npm`). Descarga desde https://nodejs.org e instala con las opciones por defecto.
2. **Git** (opcional pero recomendado para clonar repositorios). Disponible en https://git-scm.com.
3. **PostgreSQL 16** o compatible.
   - Descarga: https://www.postgresql.org/download
   - Durante la instalación anota:
     - Usuario administrador (por defecto `postgres`).
     - Contraseña asignada al usuario `postgres`.
     - Puerto (por defecto `5432`).
   - Al finalizar, crea la base de datos vacía `taller_mecanico` usando **pgAdmin** o la línea de comandos:
     ```bash
     createdb -U postgres taller_mecanico
     ```
     Ajusta el usuario y puerto si utilizas otros valores.
4. **Docker Desktop** (opcional, solo si quieres levantar todo con contenedores). Descarga desde

https://www.docker.com/products/docker-desktop e instala siguiendo el asistente.

2. Preparar el proyecto

1. **Descomprimir** el código fuente en la carpeta de tu preferencia.
2. Abrir una terminal (PowerShell en Windows) dentro de la carpeta del proyecto.
3. Instalar dependencias:
   ```powershell
   npm install
   ```

3. Configurar variables de entorno

1. Copiar el archivo de ejemplo `.env.example` (si existe) o crear `.env` en la raíz.
2. Asegúrate de definir la cadena de conexión de la base de datos:
   ```env
   DATABASE_URL="postgresql://USUARIO:CONTRASENA@localhost:5432/taller_mecanico?schema=public"
   NEXTAUTH_SECRET="clave-larga-aleatoria"
   NEXTAUTH_URL="http://localhost:3000"
   ```
   Sustituye `USUARIO` y `CONTRASENA` por los datos reales del servidor PostgreSQL.

4. Inicializar la base de datos

En la misma terminal, ejecuta los siguientes comandos en orden:

1. Crear esquemas y datos base:
   ```powershell
   npx prisma migrate reset --force
   npm run seed
   ```
2. Registrar usuario administrador estándar:
   ```powershell
   npx tsx scripts/create-admin-user.ts
   ```
3. Poblar datos de demostración coherentes para todos los módulos (clientes, trabajadores, inventario, órdenes, etc.):
   ```powershell
   npx tsx scripts/seed-sample-data.ts
   ```

   Si deseas una base vacía, omite el último comando.

5. Levantar el servidor de desarrollo

1. Ejecuta:
   ```powershell
   npm run dev
   ```
2. Abre http://localhost:3000 en un navegador.

6. Credenciales de acceso

Los scripts anteriores generan los siguientes usuarios de prueba:

Rol	Usuario	Contraseña
Administrador	`admin.pruebas`	`Admin123!`
Recepción	`recepcion.lucia`	`Taller123!`
Mecánico senior	`mecanico.jorge`	`Taller123!`
Diagnóstico	`mecanico.sofia`	`Taller123!`

El sistema obliga a cambiar la contraseña si una cuenta tiene clave temporal; estos usuarios ya tienen credenciales definitivas.

7. Verificación rápida

1. Inicia sesión como `admin.pruebas`.
2. Revisa el dashboard, el módulo de órdenes y el inventario para confirmar que los datos de prueba aparecen.
   - También puedes verificar la instancia desplegada en Vercel: https://sistema-mecanisoft-z2td.vercel.app/
3. Si necesitas recalcular datos, puedes relanzar el script de demo (`npx tsx scripts/seed-sample-data.ts`).

8. Problemas comunes

- **Error de conexión a la base de datos**: verifica la cadena `DATABASE_URL`, el puerto y que PostgreSQL esté en ejecución.
- **Puerto 3000 en uso**: cierra otras aplicaciones que lo utilicen o exporta `PORT=XXXX` antes de `npm run dev` para usar otro puerto.
- **Migraciones fallan**: asegúrate de que la base `taller_mecanico` esté vacía o ejecuta nuevamente `npx prisma migrate reset --force`.

9. Opción alternativa: levantar con Docker

Si prefieres evitar instalaciones locales, puedes usar los contenedores incluidos (`Dockerfile`, `.dockerignore` y `docker-compose.yml`).

1. Instala **Docker Desktop** (ver punto 1.4) y asegúrate de que esté en ejecución.
2. En la raíz del proyecto, ejecuta:
   ```powershell
   docker compose up --build
   ```
   Esto construirá la imagen de la aplicación, descargará PostgreSQL 16 y levantará ambos servicios.
3. El servicio `web` corre en `http://localhost:3000` y el servicio de base de datos en el puerto `5432` (expuesto solo para pruebas locales).
4. El `docker-compose.yml` ejecuta automáticamente `npx prisma migrate deploy`, `npm run seed` y `npm run start` dentro del contenedor web. Si deseas cargar también los datos de demostración, ejecuta:
   ```powershell
   docker compose exec web npx tsx scripts/seed-sample-data.ts
   ```
5. Para detener los contenedores, usa `docker compose down`. Si quieres borrar los datos persistidos de PostgreSQL, añade `-v` al comando (`docker compose down -v`).

Nota: la cadena `DATABASE_URL` y las contraseñas por defecto están definidas en `docker-compose.yml`. Ajusta valores sensibles antes de desplegar en un entorno público.

manual_docker.md

Autor: Chumbes Espinoza Marco Antonio

Manual de Uso con Docker

Autor: CHUMBES ESPINOZA MARCO ANTONIO

Este documento explica cómo levantar y administrar el sistema **MecaniSoft** utilizando contenedores Docker. Está pensado para personas sin experiencia previa; se asume que ya tienes el código fuente del proyecto.

1. Requisitos

1. **Docker Desktop** instalado y en ejecución.
   - Descarga: https://www.docker.com/products/docker-desktop
   - En Windows y macOS sigue el asistente de instalación y reinicia si te lo solicita.
2. **Terminal** (PowerShell en Windows, Terminal en macOS/Linux).
3. Espacio en disco suficiente (aprox. 2 GB) para imágenes y datos persistentes de PostgreSQL.

2. Archivos relevantes

- `Dockerfile`: define la imagen de la aplicación Next.js.
- `.dockerignore`: evita que archivos innecesarios se copien a la imagen.
- `docker-compose.yml`: orquesta los servicios `web` (app) y `db` (PostgreSQL).

Todos estos archivos se encuentran en la raíz del proyecto.

3. Variables y credenciales por defecto

`docker-compose.yml` expone las siguientes configuraciones:

- **Base de datos**
  - Usuario: `mecanisoft`
  - Contraseña: `mecanisoft123`
  - Base de datos: `taller_mecanico`
  - Puerto: `5432`
- **Aplicación**
  - `NEXTAUTH_URL`: `http://localhost:3000`
  - `NEXTAUTH_SECRET`: `change-me` (cámbiala antes de producción)
  - `DATABASE_URL`: apunta al contenedor `db` usando las credenciales anteriores.

Puedes modificar estos valores directamente en `docker-compose.yml` antes de construir.

4. Instalación usando el Docker incluido en este repositorio

Este repositorio ya incluye un `Dockerfile` y `docker-compose.yml` preparados para levantar la aplicación y una base PostgreSQL lista para desarrollo. Sigue estos pasos desde PowerShell (Windows) o tu terminal preferida.

1. Clona o sitúate en la carpeta del proyecto:

```
cd C:\ruta\a\proyecto\sistema-mecanico
```

2. Construye y levanta los contenedores en primer plano (útil para ver logs la primera vez):

```
docker compose up --build
```

3. Comportamiento automático del contenedor `web` (definido en `docker-compose.yml`):
   - Ejecuta `npx prisma migrate deploy` para aplicar migraciones.

- Corre `npm run seed` para poblar roles/permiso/catálogos básicos.
- Inicia el servidor con `npm run start`.

4. Si quieres ejecutar el proceso en segundo plano:

```
docker compose up -d --build
```

5. Crear usuario administrador de pruebas (si no está creado):
- El seed principal crea `admin` por defecto. Si prefieres `admin.pruebas` ejecuta dentro del contenedor:

```
docker compose exec web npx tsx scripts/create-admin-user.ts
```

6. Cargar dataset de demostración (opcional):

```
docker compose exec web npx tsx scripts/seed-sample-data.ts
```

7. Ejecutar migraciones manualmente (si necesitas forzar una re-aplicación):

```
docker compose exec web npx prisma migrate deploy
```

8. Ver logs en vivo (útil para depurar problemas de despliegue/migraciones):

```
docker compose logs -f web
```

9. Parar y borrar (sin borrar volúmenes):

```
docker compose down
```

10. Borrar datos persistentes (volumen de PostgreSQL):

```
docker compose down -v
```

Consejos y notas:
- Si necesitas cambiar credenciales o puertos, edita `docker-compose.yml` antes de `up --build`.
- Si al construir falla `npm install` dentro de la imagen, elimina `node_modules` locales y vuelve a intentar. En Windows, a veces `npm ci` dentro del contenedor reduce problemas de permisos.
- El contenedor web ejecuta migraciones al inicio; si ves errores en logs relacionados con Prisma, copia el error exacto y revisa `prisma/migrations`.

4. Primer arranque

1. Abre una terminal en la carpeta del proyecto.
2. Ejecuta:
   ```powershell
   docker compose up --build
   ```
   - `--build` fuerza la construcción de la imagen cada vez que cambian las dependencias o el código.
   - El proceso tarda unos minutos la primera vez (descarga Node.js, PostgreSQL y compila la app).
3. Cuando veas los mensajes "Listening on port 3000" (o similares), la app estará lista en `http://localhost:3000`.

El contenedor `web` ejecuta automáticamente:
1. `npx prisma migrate deploy`
2. `npm run seed`
3. `npm run start`

Para cargar datos de demostración adicionales, ejecuta en otra terminal:

```
docker compose exec web npx tsx scripts/seed-sample-data.ts
```

5. Comandos útiles

Acción	Comando
Arrancar en segundo plano	`docker compose up -d`
Ver logs de ambos servicios	`docker compose logs -f`
Ver logs solo de la app	`docker compose logs -f web`
Ejecutar migraciones manualmente	`docker compose exec web npx prisma migrate deploy`
Abrir una shell dentro del contenedor	`docker compose exec web sh`
Detener los contenedores	`docker compose down`
Detener y borrar volúmenes (datos DB)	`docker compose down -v`
Reconstruir la imagen sin usar cache	`docker compose build --no-cache`

6. Persistencia de datos

- El servicio `db` monta un volumen llamado `postgres-data` para guardar los datos de PostgreSQL fuera del contenedor. De este modo, los datos se conservan aunque detengas `docker compose`.
- Para limpiar todo (incluido el contenido del volumen), usa `docker compose down -v`.

7. Personalizaciones comunes

1. **Cambiar puertos**
   - Edita `docker-compose.yml` y reemplaza `3000:3000` o `5432:5432` por los puertos deseados (`HOST:CONTENEDOR`).
2. **Cambiar credenciales**
   - Modifica las variables `POSTGRES_USER`, `POSTGRES_PASSWORD` y actualiza la `DATABASE_URL` para coincidir.
3. **Salto de `npm run seed`**
   - Si quieres arrancar con una base vacía, elimina `npm run seed` del comando del servicio `web` y ejecuta manualmente `docker compose exec web npx prisma migrate deploy` o el script que prefieras.

8. Actualización de la aplicación

1. Detén los contenedores: `docker compose down`.
2. Actualiza el código fuente (copiando archivos o usando Git).
3. Reconstruye y levanta nuevamente: `docker compose up --build`.

9. Resolución de problemas

- **La app no inicia y se apaga el contenedor `web`**: revisa los logs con `docker compose logs web` para ver el error exacto. Las causas comunes son migraciones fallidas o credenciales de DB incorrectas.
- **Puerto ocupado**: otro proceso está usando el puerto (3000 o 5432). Modifica el puerto host en `docker-compose.yml` o libera el puerto ocupado.
- **Fallo de compilación al construir**: asegúrate de que el directorio contenga `package-lock.json` actualizado. Si cambiaste dependencias, corre `npm install` localmente antes de

construir.
- **Necesitas reiniciar solo PostgreSQL**: `docker compose restart db`.

10. Limpieza

Si ya no necesitas la infraestructura Docker:

```
docker compose down -v
docker image prune -f
```

Esto elimina contenedores, volúmenes y limpia imágenes sin usar.

Con estos pasos puedes administrar MecaniSoft mediante Docker sin depender de instalaciones locales adicionales.