



Gestión de Dependencias - VertexERP

Fecha de actualización: 24 de Octubre, 2025
Versión del proyecto: v4.0.0



Lockfile y Versiones Exactas

Este proyecto utiliza **yarn.lock** para garantizar que todas las instalaciones usen las mismas versiones exactas de dependencias, tanto en desarrollo como en producción.



Estado Actual:

- **yarn.lock:** Creado y versionado (12,300+ líneas)
- **node_modules:** Ignorado en Git (como debe ser)
- **package.json:** Versionado con dependencias fijas



Versiones Principales de Dependencias

Framework y Core:

Paquete	Versión	Propósito
next	14.2.28	Framework Next.js
react	18.2.0	Librería React
react-dom	18.2.0	React DOM
typescript	5.2.2	TypeScript compiler

Base de Datos y ORM:

Paquete	Versión	Propósito
@prisma/client	6.7.0	Prisma Client
prisma	6.7.0	Prisma CLI

Autenticación:

Paquete	Versión	Propósito
next-auth	4.24.11	NextAuth.js
@next-auth/prisma-adapter	1.0.7	Adaptador Prisma
bcryptjs	2.4.3	Hash de contraseñas
jsonwebtoken	9.0.2	JWT tokens

UI y Estilos:

Paquete	Versión	Propósito
tailwindcss	3.3.3	Framework CSS
@radix-ui/react-*	1.x.x - 2.x.x	Componentes UI
lucide-react	0.446.0	Iconos
framer-motion	10.18.0	Animaciones



Instalación de Dependencias

Desarrollo Local:

```
# Navegar al directorio de la aplicación
cd sistema_erp_completo/app

# Instalar todas las dependencias usando las versiones exactas del lockfile
yarn install --immutable

# Generar Prisma Client
yarn prisma generate

# Ejecutar migraciones (si hay base de datos configurada)
yarn prisma migrate dev
```

Producción (Docker):

El `Dockerfile` ya está configurado para usar `yarn install --immutable` que garantiza:

- ☒ Usa exactamente las versiones del `yarn.lock`
- ☒ Falla si `yarn.lock` no está sincronizado con `package.json`
- ☒ No modifica el `yarn.lock` durante la instalación

```
# En el Dockerfile
RUN yarn install --immutable --network-timeout 300000
```

Producción (Manual):

```
cd /ruta/al/proyecto/app

# Instalar dependencias (modo producción)
yarn install --immutable --production=false

# Generar Prisma Client
yarn prisma generate

# Ejecutar migraciones
yarn prisma migrate deploy

# Build de producción
yarn build

# Iniciar aplicación
yarn start
```



Actualización de Dependencias



Importante:

NO actualices dependencias sin probar exhaustivamente. Las versiones actuales están probadas y funcionan correctamente juntas.

Si necesitas actualizar:

```
# Ver dependencias desactualizadas
yarn outdated

# Actualizar una dependencia específica (con cuidado)
yarn upgrade <paquete>@<versión>

# Actualizar yarn.lock después de cambios en package.json
yarn install

# SIEMPRE probar después de actualizar
yarn build
yarn prisma generate
```



Docker y Producción

Dockerfile Optimizado:

El Dockerfile usa una estrategia multi-stage que:

1. Stage de dependencias:

- Copia solo package.json y yarn.lock
- Ejecuta `yarn install --immutable`
- Cachea node_modules para builds rápidos

2. Stage de build:

- Copia el código fuente
- Genera Prisma Client
- Ejecuta el build de Next.js

3. Stage de producción:

- Usa imagen slim de Node.js
- Copia solo los archivos necesarios
- Configuración optimizada para producción



Gestión de Cache de Yarn

Cache Local:

Yarn cachea los paquetes descargados para instalaciones más rápidas:

```
# Ver ubicación del cache
yarn cache dir

# Limpiar cache (si hay problemas)
yarn cache clean

# Reinstalar desde cero
rm -rf node_modules .yarn/cache
yarn install --immutable
```



Verificación de Integridad

Verificar que yarn.lock está sincronizado:

```
# Esto debe pasar sin errores
yarn install --immutable

# Si falla, actualiza el lockfile
yarn install
```

Auditoría de seguridad:

```
# Auditar dependencias
yarn npm audit

# Ver detalles de vulnerabilidades
yarn npm audit --recursive
```



Estadísticas del Proyecto

Tamaño de Dependencias:

- **node_modules:** ~1.2 GB (desarrollo completo)
- **node_modules:** ~400 MB (solo producción)
- **yarn.lock:** ~434 KB
- **Total de paquetes:** ~1,146 paquetes

Tiempo de Instalación:

- **Primera instalación:** ~60-120 segundos (según conexión)
 - **Con cache:** ~15-30 segundos
 - **Docker build (con cache):** ~45-90 segundos
-



Compatibilidad

Node.js:

- **Versión requerida:** $\geq 18.17.0$
- **Versión recomendada:** 20.x LTS
- **Versión probada:** 20.6.2

Yarn:

- **Versión requerida:** $\geq 4.0.0$
- **Versión actual:** 4.9.4

Sistemas Operativos:

- ✓ Linux (Ubuntu, Debian, CentOS, etc.)
- ✓ macOS
- ✓ Windows (con WSL2 recomendado)

Navegadores (Frontend):

- ✓ Chrome/Edge 90+
 - ✓ Firefox 88+
 - ✓ Safari 14+
 - ✓ Mobile (iOS Safari 14+, Chrome Android 90+)
-



Mejores Prácticas

✓ DO (Hacer):






1. **Siempre usar** `yarn install --immutable` **en CI/CD**
2. **Versionar yarn.lock en Git**
3. **Hacer commit de package.json y yarn.lock juntos**
4. **Probar después de actualizar dependencias**
5. **Documentar cambios importantes en CHANGELOG**

DON'T (No Hacer):

1. **NO** ignorar `yarn.lock` en `.gitignore`
 2. **NO** usar `npm install` (siempre usar `yarn`)
 3. **NO** editar `yarn.lock` manualmente
 4. **NO** hacer commit solo de `package.json` sin `yarn.lock`
 5. **NO** actualizar todas las dependencias a la vez
-

Siguientes Pasos

Para deployment en producción:

1.  Verificar que `yarn.lock` esté en el repositorio
 2.  Configurar variables de entorno
 3.  Ejecutar migraciones de base de datos
 4.  Build y deploy usando Docker
 5.  Monitorear logs y errores
-

VertexERP v4.0.0 - Gestión de Dependencias

Última actualización: 24 de Octubre, 2025