Scripts Útiles Adaptados de CitaPlanner

✅ Archivos Implementados

1. `emergency-start.sh` 🚨

Propósito: Bypass completo de checks para debug rápido en producción

```
./emergency-start.sh
```

Cuándo usar:
- Error en migraciones de Prisma bloqueando el inicio
- Problemas de conectividad con la base de datos
- Debug rápido sin esperar validaciones

Advertencia: Omite migraciones y seed. Solo para troubleshooting.

---

2. `start-improved.sh` ⚡

Propósito: Script de inicio mejorado con mejor detección de errores

```
./start-improved.sh
```

Mejoras vs start.sh original:
- ✅ Detección automática del comando Prisma disponible
- ✅ Logging detallado de cada paso
- ✅ Verificación de conexión a base de datos
- ✅ Manejo robusto de errores
- ✅ Verificación de existencia de server.js

Cuándo usar: Como reemplazo del start.sh actual en producción

---

3. `verify-build.sh` 🔍

Propósito: Verificación completa del build antes de deploy

```
cd app
../verify-build.sh
```

Verificaciones:
- ✅ Archivos esenciales (package.json, next.config.js, schema.prisma)
- ✅ node_modules instalado correctamente
- ✅ Prisma Client generado
- ✅ Build standalone de Next.js
- ✅ server.js en ubicación correcta

Cuándo usar: Antes de hacer commit/push para validar el build

---

4. `docker-compose.easypanel.yml` 🐳

Propósito: Configuración específica para EasyPanel

```
docker-compose -f docker-compose.easypanel.yml up
```

Características:
- Variables de entorno completas
- Healthcheck configurado
- Network isolation
- Restart policy

Cuándo usar: Deploy en EasyPanel en lugar de docker-compose.yml genérico

---

🚀 Integración en Dockerfile

Para usar `start-improved.sh` en el Dockerfile:

```
# En el stage runner
COPY --chown=nextjs:nodejs start-improved.sh ./
RUN chmod +x start-improved.sh

CMD ["./start-improved.sh"]
```

📋 Comparación Scripts de Inicio

Feature	start.sh	start-improved.sh	emergency-start.sh
Detección Prisma CLI	Básica	Automática	No aplica
Logging	Simple	Detallado	Mínimo
Verificación DB	✅	✅	❌
Migraciones	✅	✅	❌
Seed automático	✅	✅	❌
Error handling	Básico	Robusto	Bypass
Verificación server.js	❌	✅	❌
Uso recomendado	Local dev	Producción	Debug

🔄 Workflow Recomendado

Desarrollo Local

```
# 1. Build
cd app && yarn build

# 2. Verificar
cd .. && ./verify-build.sh

# 3. Test local
cd app && node .next/standalone/app/server.js
```

Deploy a Producción

```
# 1. Verificar build
./verify-build.sh

# 2. Commit y push
git add .
git commit -m "..."
git push

# 3. En EasyPanel: usar docker-compose.easypanel.yml
```

Debug en Producción

```
# Si hay problemas de inicio, usar emergency-start
CMD ["./emergency-start.sh"]
```

🎯 Próximos Pasos

1. **Actualizar Dockerfile** para usar `start-improved.sh`
2. **Configurar EasyPanel** con `docker-compose.easypanel.yml`
3. **Agregar** `verify-build.sh` a CI/CD pipeline
4. **Documentar** `emergency-start.sh` en runbook de operaciones

📚 Referencias

- Código original: https://github.com/qhosting/citaplanner
- Adaptaciones específicas para Node 22 + Yarn
- Optimizado para Next.js standalone output