🔄 GUÍA DE MIGRACIÓN DEEPAGENT - ESCALAFIN MVP

Versión de Migración: v2.1.0
Fecha: Septiembre 22, 2025
Objetivo: Transferir proyecto completo a nueva cuenta DeepAgent

---

🎯 PROPÓSITO DE ESTA GUÍA

Esta guía permite a **cualquier usuario de DeepAgent** importar y continuar el desarrollo de EscalaFin MVP desde cero, sin perder funcionalidad ni configuraciones.

📋 PRE-REQUISITOS

En la Nueva Cuenta DeepAgent:

- ✅ Acceso activo a DeepAgent
- ✅ Permisos para crear proyectos
- ✅ Capacidad para instalar dependencias
- ✅ Acceso a herramientas de base de datos

Archivos Necesarios:

- ✅ Código fuente completo del proyecto
- ✅ Documentación técnica
- ✅ Variables de entorno ejemplo
- ✅ Esquema de base de datos

🚀 PROCESO DE MIGRACIÓN PASO A PASO

ETAPA 1: PREPARACIÓN DEL ENTORNO

Paso 1.1: Crear Directorio del Proyecto

```
# En DeepAgent, ejecutar:
mkdir /home/ubuntu/escalafin_mvp
cd /home/ubuntu/escalafin_mvp
```

Paso 1.2: Verificar Herramientas Disponibles

```
# Verificar Node.js
node --version  # Debe ser >= 18

# Verificar Yarn
yarn --version  # Package manager principal

# Verificar Git
git --version   # Control de versiones
```

ETAPA 2: IMPORTACIÓN DEL CÓDIGO

Paso 2.1: Obtener el Código Fuente

Opción A: Desde GitHub (Si está disponible)

```
git clone https://github.com/usuario/escalafin-mvp.git .
```

Opción B: Importación Manual

Si el código se proporciona como archivos:

1. Subir archivos al directorio `/home/ubuntu/escalafin_mvp`
2. Extraer si están comprimidos
3. Verificar estructura de carpetas

Paso 2.2: Verificar Estructura del Proyecto

```
# Verificar que existe la estructura esperada
ls -la
# Debe mostrar: app/, docs/, *.md files

cd app
ls -la
# Debe mostrar: api/, components/, lib/, prisma/, etc.
```

ETAPA 3: CONFIGURACIÓN DE DEPENDENCIAS

Paso 3.1: Navegar a la App

```
cd /home/ubuntu/escalafin_mvp/app
```

Paso 3.2: Instalar Dependencias

```
# Limpiar cache si es necesario
rm -rf node_modules yarn.lock

# Instalar todas las dependencias
yarn install

# Si hay errores, intentar:
yarn install --network-timeout 100000
```

Paso 3.3: Verificar Instalación

```
# Verificar que las dependencias críticas están instaladas
yarn list @prisma/client
yarn list next
yarn list react
yarn list next-auth
```

ETAPA 4: CONFIGURACIÓN DE BASE DE DATOS

Paso 4.1: Inicializar PostgreSQL

```
# En DeepAgent, usar el tool:
# initialize_postgres_db con project_path: /home/ubuntu/escalafin_mvp
```

Paso 4.2: Configurar Variables de Entorno

```
# Copiar archivo ejemplo
cp .env.example .env

# Editar variables (usar herramientas DeepAgent o manualmente)
# Asegurar que DATABASE_URL apunte a la BD creada
```

Variables Críticas a Configurar:

```
DATABASE_URL="postgresql://user:pass@host:5432/escalafin"
NEXTAUTH_SECRET="generar_secreto_32_caracteres_minimo"
NEXTAUTH_URL="http://localhost:3000"
NODE_ENV="development"
```

Paso 4.3: Generar y Sincronizar Base de Datos

```
# Generar cliente Prisma
yarn prisma generate

# Sincronizar esquema con BD
yarn prisma db push

# Poblar con datos de prueba
yarn prisma db seed
```

ETAPA 5: CONFIGURACIÓN DE SERVICIOS

Paso 5.1: Configurar Autenticación

```
# En DeepAgent, usar tool:
# initialize_auth con project_path: /home/ubuntu/escalafin_mvp
```

Paso 5.2: Configurar Almacenamiento (Opcional)

```
# En DeepAgent, usar tool:
# initialize_cloud_storage con project_path: /home/ubuntu/escalafin_mvp
```

Paso 5.3: Configurar APIs LLM (Opcional)

```
# En DeepAgent, usar tool:
# initialize_llm_apis con project_path: /home/ubuntu/escalafin_mvp
```

ETAPA 6: VERIFICACIÓN Y TESTING

Paso 6.1: Verificar Compilación

```
# Test de build completo
yarn build

# Si hay errores, revisar logs y corregir
# Errores comunes y soluciones al final de esta guía
```

Paso 6.2: Testing Completo del Proyecto

```
# En DeepAgent, usar tool:
# test_nextjs_project con project_path: /home/ubuntu/escalafin_mvp
```

Paso 6.3: Iniciar Servidor de Desarrollo

```
# Iniciar en modo desarrollo
yarn dev

# Debe mostrar: Ready - started server on 0.0.0.0:3000
```

Paso 6.4: Probar Funcionalidades Básicas

```
# Abrir en navegador: http://localhost:3000
# Probar login con cuentas de prueba:
# admin@escalafin.com / admin123
# asesor@escalafin.com / asesor123
# cliente@escalafin.com / cliente123
```

🔧 CONFIGURACIONES ESPECÍFICAS POR SERVICIO

Openpay (Pagos Online)

Obtener Credenciales:

1. Registrarse en https://openpay.mx
2. Crear cuenta sandbox
3. Obtener: merchant_id, private_key, public_key

Configurar en .env:

```
OPENPAY_MERCHANT_ID="tu_merchant_id_sandbox"
OPENPAY_PRIVATE_KEY="sk_xxxxxxxxxxxx"
OPENPAY_PUBLIC_KEY="pk_xxxxxxxxxxxx"
OPENPAY_BASE_URL="https://sandbox-api.openpay.mx/v1"
```

WhatsApp (EvolutionAPI)

Obtener Instancia:

1. Contratar servicio EvolutionAPI
2. Crear instancia WhatsApp Business
3. Obtener: URL base, token, nombre instancia

Configurar en .env:

```
EVOLUTION_API_URL="https://tu-instancia.evolution-api.com"
EVOLUTION_API_TOKEN="tu_token_aqui"
EVOLUTION_INSTANCE_NAME="escalafin"
```

AWS S3 (Almacenamiento Cloud)

Configurar Bucket:

1. Crear cuenta AWS
2. Crear bucket S3
3. Configurar IAM user con permisos

Configurar en .env:

```
AWS_ACCESS_KEY_ID="AKIA..."
AWS_SECRET_ACCESS_KEY="tu_secret_key"
AWS_BUCKET_NAME="escalafin-files"
AWS_REGION="us-east-1"
AWS_FOLDER_PREFIX="escalafin/"
```

🚨 PROBLEMAS COMUNES Y SOLUCIONES

Error: Build Falla

Síntoma:

```
yarn build
# Error: Type error or compilation failed
```

Solución:

```
# Limpiar y reinstalar
rm -rf .next node_modules yarn.lock
yarn install
yarn prisma generate
yarn build
```

Error: Base de Datos no Conecta

Síntoma:

```
# Error: Can't reach database server
```

Solución:

```
# Verificar DATABASE_URL en .env
echo $DATABASE_URL

# Regenerar cliente Prisma
yarn prisma generate

# Test conexión
yarn prisma db push
```

Error: NextAuth Session

Síntoma:

```
# Error: NEXTAUTH_SECRET missing
```

Solución:

```
# Generar secreto seguro
openssl rand -base64 32

# Agregar a .env:
NEXTAUTH_SECRET="secreto_generado_aqui"
```

Error: Dependencias Faltantes

Síntoma:

```
# Module not found errors
```

Solución:

```
# Instalar dependencias específicas
yarn add @prisma/client
yarn add next-auth
yarn add @aws-sdk/client-s3

# O reinstalar todo
yarn install --check-files
```

Error: Puerto en Uso

Síntoma:

```
# Port 3000 is already in use
```

Solución:

```
# Usar puerto alternativo
yarn dev --port 3001

# O encontrar y matar proceso
lsof -ti:3000 | xargs kill
```

🔄 VERIFICACIÓN DE MIGRACIÓN EXITOSA

Checklist de Funcionalidades:

✅ Autenticación

- [ ] Login page carga correctamente
- [ ] Login con admin@escalafin.com funciona
- [ ] Dashboard admin es accesible
- [ ] Logout funciona correctamente

✅ Base de Datos

- [ ] Prisma cliente genera sin errores
- [ ] Tablas se crean correctamente
- [ ] Seed data se carga exitosamente
- [ ] Queries funcionan desde la app

✅ APIs

- [ ] GET /api/clients devuelve datos
- [ ] GET /api/loans devuelve préstamos
- [ ] APIs protegidas requieren autenticación
- [ ] Webhooks endpoints responden

✅ Frontend

- [ ] Páginas cargan sin errores 404
- [ ] Componentes renderizan correctamente
- [ ] Navegación entre dashboards funciona
- [ ] Formularios se envían exitosamente

✅ Servicios Externos

- [ ] Openpay test endpoint funciona (opcional)
- [ ] WhatsApp config page carga (opcional)
- [ ] File upload funciona (local como mínimo)

📊 VALIDACIÓN FINAL

Comando de Verificación Completa:

```
cd /home/ubuntu/escalafin_mvp/app

# 1. Build exitoso
echo "🏗️  Testing build..."
yarn build && echo "✅ Build OK" || echo "❌ Build FAIL"

# 2. Servidor inicia
echo "🚀 Testing server start..."
timeout 10s yarn start && echo "✅ Server OK" || echo "❌ Server FAIL"

# 3. Database conecta
echo "💾 Testing database..."
yarn prisma db push && echo "✅ DB OK" || echo "❌ DB FAIL"

# 4. APIs responden
echo "🔌 Testing APIs..."
curl -s http://localhost:3000/api/clients > /dev/null && echo "✅ API OK" || echo "❌
API FAIL"

echo "🎉 Migration validation complete!"
```

Resultado Esperado:

```
🏗️  Testing build...
✅ Build OK
🚀 Testing server start...
✅ Server OK
💾 Testing database...
✅ DB OK
🔌 Testing APIs...
✅ API OK
🎉 Migration validation complete!
```

📈 PRÓXIMOS PASOS DESPUÉS DE MIGRACIÓN

Desarrollo Inmediato:

1. **Revisar documentación** en /docs para entender funcionalidades
2. **Probar todas las características** con las cuentas de prueba
3. **Configurar servicios externos** según necesidades del negocio
4. **Planificar siguientes desarrollos** según roadmap

Configuración Producción:

1. **Configurar variables de entorno** de producción
2. **Configurar servicios externos** en modo producción
3. **Deploy en plataforma** elegida (Vercel, Railway, etc.)
4. **Configurar monitoreo** y backup de datos

Desarrollo Avanzado:

1. **Implementar PWA** para aplicación móvil
2. **Agregar geolocalización** para cobranza
3. **Integrar más pasarelas** de pago
4. **Desarrollar API pública** documentada

🆘 SOPORTE POST-MIGRACIÓN

Documentación Disponible:

- `GUIA_COMPLETA_IMPORTACION_2025.md` - Guía general de importación
- `DEEPAGENT_CONTINUATION_GUIDE_FINAL.md` - Guía técnica continuación
- `SCHEMA.md` - Esquema completo base de datos
- `GITHUB_SETUP_COMPLETO.md` - Configuración GitHub
- `analisis_funcionalidad.md` - Estado detallado módulos

Logs y Debugging:

```
# Ver logs de aplicación
yarn dev 2>&1 | tee migration.log

# Ver logs específicos de errores
yarn build 2>&1 | grep -i error

# Ver estado de base de datos
yarn prisma studio  # Abre en http://localhost:5555
```

Comandos de Diagnóstico:

```
# Verificar estructura de archivos
find . -name "*.tsx" -o -name "*.ts" | head -20

# Verificar dependencias instaladas
yarn list --depth=0

# Verificar variables de entorno
env | grep -E "(DATABASE|NEXTAUTH|OPENPAY)"

# Verificar puertos disponibles
netstat -tlnp | grep :3000
```

✅ CHECKLIST FINAL DE MIGRACIÓN

Pre-Migración:

- [ ] Nueva cuenta DeepAgent activa
- [ ] Código fuente disponible
- [ ] Documentación revisada

Migración:

- [ ] Proyecto creado en directorio correcto
- [ ] Dependencias instaladas exitosamente
- [ ] Base de datos configurada y poblada
- [ ] Variables de entorno configuradas
- [ ] Servicios externos conectados (opcional)

Verificación:

- [ ] Build de producción exitoso
- [ ] Servidor de desarrollo funcional
- [ ] Login con cuentas de prueba exitoso
- [ ] Navegación dashboards funcional
- [ ] APIs respondiendo correctamente
- [ ] Testing completo pasado

Post-Migración:

- [ ] Funcionalidades principales probadas
- [ ] Documentación técnica revisada
- [ ] Plan de desarrollo definido
- [ ] Servicios externos configurados (si aplica)

🏆 CONFIRMACIÓN DE MIGRACIÓN EXITOSA

Criterios de Éxito:

1. ✅ **Build exitoso:** `yarn build` completa sin errores
2. ✅ **Servidor funcional:** `yarn dev` inicia correctamente
3. ✅ **Login funcionando:** Todas las cuentas de prueba accesibles
4. ✅ **Base de datos:** Datos de prueba cargados y accesibles
5. ✅ **APIs operativas:** Endpoints principales responden
6. ✅ **Navegación:** Dashboards cargan sin errores 404/500

Mensaje de Confirmación:

```
🎉 ¡MIGRACIÓN EXITOSA! 🎉

EscalaFin MVP ha sido migrado exitosamente a tu nueva cuenta DeepAgent.

✅ Sistema completamente funcional
✅ Base de datos poblada con datos de prueba
✅ Todas las funcionalidades operativas
✅ APIs respondiendo correctamente
✅ Autenticación funcionando
✅ Dashboards multi-rol accesibles

🚀 ¡Listo para continuar el desarrollo!

Próximos pasos:
1. Revisar documentación técnica
2. Probar todas las funcionalidades
3. Configurar servicios externos según necesidad
4. Planificar siguientes desarrollos

📚 Documentación disponible en /docs
🔧 Cuentas de prueba listas para usar
💻 Desarrollo listo para continuar
```

🎯 ¡Migración completa y exitosa a nueva cuenta DeepAgent! 🚀

Fecha de Creación: Septiembre 22, 2025
Versión Migración: v2.1.0
Status: ✅ LISTO PARA USO INMEDIATO