

# Guía de Importación a Otra Cuenta DeepAgent - EscalaFin MVP

---

## Descripción General

---

Esta guía detalla el proceso completo para importar el proyecto EscalaFin MVP a otra cuenta de DeepAgent, incluyendo transferencia de archivos, configuración de base de datos, y preservación de todas las funcionalidades.

## Antes de Comenzar

---

### Prerrequisitos

- **Cuenta DeepAgent destino** activa y funcional
- **Proyecto EscalaFin MVP** completamente funcional en cuenta origen
- **Acceso a base de datos** de producción (opcional)
- **Credenciales de servicios externos** (Openpay, AWS S3, etc.)

### Inventario del Sistema

Antes de migrar, verifique que tiene:

- ✓ Código fuente completo
- ✓ Base de datos con datos
- ✓ Archivos subidos (local o S3)
- ✓ Variables de entorno
- ✓ Configuraciones de APIs externas
- ✓ Documentación actualizada

## Preparación de la Exportación

---

### 1. Crear Checkpoint Final

En la cuenta DeepAgent original:

```
# Asegúrese de que todo está funcionando
npm test
npm run build

# Crear checkpoint con descripción clara
"Sistema completo listo para migración - v2.1.0"
```

## 2. Exportar Código Fuente

### Opción A: Via Git (Recomendado)

```
# 1. Crear repositorio en GitHub/GitLab
git init
git add .
git commit -m "EscalaFin MVP - Sistema completo v2.1.0"
git branch -M main
git remote add origin https://github.com/tu-usuario/escalafin-mvp-exportado.git
git push -u origin main

# 2. Crear tags para versiones
git tag -a v2.1.0 -m "Version completa con almacenamiento dual"
git push origin v2.1.0
```

### Opción B: Descarga Directa

1. Use el botón “Files” en DeepAgent
2. Descargue todo el proyecto como ZIP
3. Mantenga la estructura de carpetas intacta

## 3. Backup de Base de Datos

```
-- Conectarse a la base de datos
psql $DATABASE_URL

-- Crear backup completo
pg_dump $DATABASE_URL > escalafin_backup_$(date +%Y%m%d).sql

-- Verificar backup
wc -l escalafin_backup_$(date +%Y%m%d).sql
```

## 4. Inventario de Variables de Entorno

Documente todas las variables:

```
# === VARIABLES DE ENTORNO REQUERIDAS ===

# Base de Datos
DATABASE_URL=postgresql://...

# NextAuth
NEXTAUTH_SECRET=...
NEXTAUTH_URL=...

# Node Environment
NODE_ENV=production
NEXTAUTH_DEBUG=false

# Openpay
OPENPAY_MERCHANT_ID=...
OPENPAY_PRIVATE_KEY=...
OPENPAY_BASE_URL=...

# Almacenamiento
STORAGE_TYPE=s3|local
AWS_BUCKET_NAME=...
AWS_REGION=...
AWS_FOLDER_PREFIX=...
AWS_ACCESS_KEY_ID=...
AWS_SECRET_ACCESS_KEY=...

# EvolutionAPI (opcional)
EVOLUTION_API_BASE_URL=...
EVOLUTION_API_KEY=...
EVOLUTION_INSTANCE_NAME=...
```

## 5. Backup de Archivos

### Si usa S3:

```
# Listar todos los archivos
aws s3 ls s3://tu-bucket/escalafin-mvp/ --recursive

# Crear backup local (opcional)
aws s3 sync s3://tu-bucket/escalafin-mvp/ ./backup_files/
```

### Si usa almacenamiento local:

```
# Crear archivo comprimido
tar -czf escalafin_files_backup.tar.gz /ruta/a/uploads/

# Verificar contenido
tar -tzf escalafin_files_backup.tar.gz | head -20
```



## Proceso de Importación

### 1. Configurar Cuenta Destino

En la nueva cuenta DeepAgent:

## A. Inicializar Proyecto

```
# Opción 1: Desde repositorio Git
git clone https://github.com/tu-usuario/escalafin-mvp-exportado.git
cd escalafin-mvp-exportado

# Opción 2: Subir archivos ZIP
# - Usar interfaz DeepAgent para subir proyecto
# - Mantener estructura de carpetas
```

## B. Configurar Base de Datos

```
# 1. Inicializar nueva base de datos
# (DeepAgent proporcionará nueva DATABASE_URL)

# 2. Si quiere datos existentes:
# Restaurar desde backup
psql $NEW_DATABASE_URL < escalafin_backup_20250921.sql

# 3. Si quiere base limpia:
cd app
npx prisma db push
npx prisma db seed
```

## 2. Configurar Variables de Entorno

En DeepAgent, configurar todas las variables:

```
# === VARIABLES NUEVAS ===
DATABASE_URL=[nueva_url_proporcionada_por_deepagent]
NEXTAUTH_URL=[nueva_url_de_la_app]

# === VARIABLES REUTILIZADAS ===
NEXTAUTH_SECRET=[el_mismo_secreto]
OPENPAY_MERCHANT_ID=[mismo_merchant]
OPENPAY_PRIVATE_KEY=[misma_key]
# ... resto de variables idénticas
```

## 3. Configurar Almacenamiento

### Opción A: Reutilizar S3 Existente

```
# Mantener la misma configuración S3
STORAGE_TYPE=s3
AWS_BUCKET_NAME=[mismo_bucket]
AWS_ACCESS_KEY_ID=[mismas_credenciales]
AWS_SECRET_ACCESS_KEY=[mismas_credenciales]
```

## Opción B: Crear Nuevo S3 Bucket

```
# Crear nuevo bucket
aws s3 mb s3://escalafin-nueva-cuenta

# Copiar archivos del bucket anterior
aws s3 sync s3://escalafin-original/ s3://escalafin-nueva-cuenta/

# Actualizar variables
STORAGE_TYPE=s3
AWS_BUCKET_NAME=escalafin-nueva-cuenta
```

## Opción C: Migrar a Almacenamiento Local

```
STORAGE_TYPE=local
LOCAL_UPLOAD_DIR=/home/ubuntu/escalafin_mvp/uploads
```

## 4. Migrar Archivos (Si Necesario)

### Script de Migración de Archivos

```
// scripts/migrate-files.js
const { PrismaClient } = require('@prisma/client')
const fs = require('fs')
const path = require('path')

const prisma = new PrismaClient()

async function migrateFiles() {
  console.log('🔄 Iniciando migración de archivos...')

  // Obtener todos los archivos de la base de datos
  const files = await prisma.file.findMany({
    orderBy: { createdAt: 'desc' }
  })

  console.log(`📁 Encontrados ${files.length} archivos`)

  let migrated = 0
  let errors = 0

  for (const file of files) {
    try {
      // Verificar si el archivo existe
      const oldPath = file.filePath
      const newPath = oldPath // o modificar según necesidad

      // Aquí implementar lógica específica de migración
      // Ejemplo: copiar de S3 viejo a S3 nuevo

      console.log(`✅ Migrado: ${file.originalName}`)
      migrated++

    } catch (error) {
      console.error(`❌ Error con ${file.originalName}:`, error.message)
      errors++
    }
  }

  console.log(`\n📊 Resumen de migración:`)
  console.log(`✅ Exitosos: ${migrated}`)
  console.log(`❌ Errores: ${errors}`)
  console.log(`📁 Total: ${files.length}`)
}

migrateFiles()
  .catch(console.error)
  .finally(() => prisma.$disconnect())
```

## 5. Verificación Post-Migración

### A. Tests Funcionales

```
# En la nueva cuenta DeepAgent

# 1. Verificar build
npm run build

# 2. Verificar conexión a DB
npx prisma db pull

# 3. Verificar variables de entorno
node -e "console.log(process.env.DATABASE_URL ? '✅ DB OK' : '❌ DB MISSING')"
```

### B. Tests de Funcionalidad

```
// scripts/verify-migration.js
const { PrismaClient } = require('@prisma/client')

const prisma = new PrismaClient()

async function verifyMigration() {
  try {
    // Verificar conexión DB
    await prisma.$connect()
    console.log('✅ Conexión a base de datos OK')

    // Verificar tablas principales
    const userCount = await prisma.user.count()
    const clientCount = await prisma.client.count()
    const loanCount = await prisma.loan.count()
    const fileCount = await prisma.file.count()

    console.log(`📊 Datos encontrados:`)
    console.log(` 👤 Usuarios: ${userCount}`)
    console.log(` 📁 Clientes: ${clientCount}`)
    console.log(` 💰 Préstamos: ${loanCount}`)
    console.log(` 📁 Archivos: ${fileCount}`)

    // Verificar usuario admin
    const adminUser = await prisma.user.findFirst({
      where: { role: 'ADMIN' }
    })

    if (adminUser) {
      console.log('✅ Usuario admin encontrado: ${adminUser.email}')
    } else {
      console.log('⚠️ No se encontró usuario admin')
    }
  } catch (error) {
    console.error('❌ Error en verificación:', error.message)
  } finally {
    await prisma.$disconnect()
  }
}

verifyMigration()
```

## Configuraciones Específicas

### 1. Actualizar URLs y Dominios

#### NextAuth Configuration

```
// app/lib/auth.ts
const authOptions: NextAuthOptions = {
  // Actualizar URL base
  pages: {
    signIn: '/auth/login',
  },
  callbacks: {
    async redirect({ url, baseUrl }) {
      // Usar nueva URL base
      if (url.startsWith('/')) return `${baseUrl}${url}`
      else if (new URL(url).origin === baseUrl) return url
      return baseUrl
    },
  },
}
```

#### Webhooks URLs

```
// Actualizar URLs de webhooks
const webhookUrls = {
  openpay: `${process.env.NEXTAUTH_URL}/api/webhooks/openpay`,
  evolutionapi: `${process.env.NEXTAUTH_URL}/api/webhooks/evolution-api`,
}
```

### 2. Configurar Servicios Externos

#### Openpay Webhooks

```
# Actualizar webhooks en Openpay dashboard
# Nuevo endpoint: https://nueva-url.com/api/webhooks/openpay
```

#### EvolutionAPI

```
# Actualizar webhook URL en EvolutionAPI
# Nuevo endpoint: https://nueva-url.com/api/webhooks/evolution-api
```



### 3. SSL y Seguridad

```
// app/middleware.js - Actualizar para nueva URL
const allowedOrigins = [
  'https://nueva-url-deepagent.com',
  'https://dominio-personalizado.com',
]

export function middleware(request) {
  const origin = request.headers.get('origin')

  if (origin && !allowedOrigins.includes(origin)) {
    return new Response('Forbidden', { status: 403 })
  }

  // ... resto del middleware
}
```



### Checklist Final



### Verificación Completa

Después de la migración, verificar:

```
# 1. Aplicación accesible
curl -I https://nueva-url.com
# Esperado: HTTP 200 o 307 (redirect a login)

# 2. Login administrativo
# Navegador: ir a /auth/login
# Usuario: admin creado en seed

# 3. Base de datos funcional
# Panel: ir a /admin/dashboard
# Verificar métricas

# 4. Subida de archivos
# Panel: ir a /admin/files
# Subir archivo de prueba

# 5. Configuración de almacenamiento
# Panel: ir a /admin/storage
# Probar conectividad

# 6. APIs funcionando
curl https://nueva-url.com/api/health
# Esperado: {"status":"healthy"}
```



### Tests de Integración

```
# Test completo del flujo
# 1. Crear cliente
# 2. Crear préstamo
# 3. Procesar pago
# 4. Subir documento
# 5. Generar reporte
```

## Solución de Problemas Comunes

### 1. Error de Base de Datos

```
# Problema: Conexión rechazada
# Solución: Verificar DATABASE_URL
echo $DATABASE_URL

# Problema: Tabla no existe
# Solución: Aplicar migraciones
npx prisma db push
```

### 2. Error de Archivos

```
# Problema: Archivos no se cargan
# Verificar storage configuration
node -e "console.log(process.env.STORAGE_TYPE)"

# Test S3 connectivity
aws s3 ls s3://tu-bucket-nombre
```

### 3. Error de Servicios Externos

```
# Problema: Openpay no funciona
# Verificar keys
node -e "console.log(process.env.OPENPAY_MERCHANT_ID?.slice(0,5) + '...')"

# Test API
curl -H "Content-Type: application/json" \
  -u "${OPENPAY_MERCHANT_ID}:${OPENPAY_PRIVATE_KEY}" \
  "${OPENPAY_BASE_URL}/${OPENPAY_MERCHANT_ID}/customers"
```

### 4. Error de Next.js

```
# Limpiar cache
rm -rf app/.next
cd app && npm run build

# Verificar variables de entorno
npm run build 2>&1 | grep -i "env"
```

## Post-Migración y Soporte





### 1. Documentar Cambios

Crear documento con:

- Nueva URL de la aplicación
- Credenciales de usuario admin
- URLs de webhooks actualizadas
- Cambios en configuración

## 2. Comunicar a Stakeholders

Notificar cambios a:

-  **Usuarios administradores**
-  **Equipo de soporte**
-  **Proveedores de servicios** (si aplica)
-  **Auditores** (si corresponde)

## 3. Monitoreo Inicial

Durante las primeras 48 horas:

- Monitor uptime y performance
- Revisar logs de errores
- Verificar funcionalidad crítica
- Confirmar backups automáticos

## 4. Plan de Rollback

En caso de problemas críticos:






1. **Conservar cuenta original** por 30 días
2. **Backups verificados** antes de migración
3. **Procedimiento de reversa** documentado
4. **Contactos de emergencia** actualizados



## Resumen de Migración







### Lo que se Mantiene

-  **Código fuente completo**
-  **Base de datos y datos**
-  **Configuración de funcionalidades**
-  **APIs y integraciones**
-  **Sistema de archivos** (con migración)







### Lo que se Actualiza

-  **URLs de aplicación**
-  **Variables de entorno de DeepAgent**
-  **URLs de webhooks**
-  **Configuración de dominio**



### Consideraciones Importantes

-  **Tiempo de inactividad** durante migración
-  **Actualizar webhooks** en servicios externos
-  **Verificar todos los endpoints**
-  **Probar funcionalidad completa**

**Tiempo Estimado de Migración:** 2-4 horas

**Complejidad:** Media-Alta

**Riesgo:** Medio (con preparación adecuada)

**Versión de Guía:** 2.1.0

**Última actualización:** Septiembre 2025

**Compatibilidad:** DeepAgent 2024+, EscalaFin MVP 2.1.0+