

GUÍA COMPLETA DE IMPORTACIÓN - ESCALAFIN MVP 2025

Versión: 2.1.0 Final










Fecha: Septiembre 22, 2025

Estado:  PRODUCCIÓN READY

RESUMEN EJECUTIVO

EscalaFin es un **Sistema Completo de Gestión de Préstamos y Créditos** desarrollado en Next.js 14 con integración de pagos (Openpay), notificaciones WhatsApp (EvolutionAPI) y sistema de archivos cloud (AWS S3).

Características Principales:

-  **CRM de Clientes** con información financiera completa
-  **Sistema de Préstamos** con amortización automática
-  **Pagos Online** integrados con Openpay
-  **WhatsApp** notificaciones automáticas post-pago
-  **Analytics** y reportes ejecutivos
-  **Multi-rol:** Admin, Asesor, Cliente
-  **Autenticación** segura con NextAuth
-  **Almacenamiento Cloud** dual (Local/AWS S3)
-  **Módulo Móvil** para cobranza en efectivo

INSTRUCCIONES DE IMPORTACIÓN

PASO 1: Configuración Inicial en DeepAgent

1.1 Crear Nuevo Proyecto

```
# En DeepAgent, crear directorio del proyecto
mkdir /home/ubuntu/escalafin_mvp
cd /home/ubuntu/escalafin_mvp
```

1.2 Clonar o Importar Código

Si tienes acceso al código fuente, cópialos al directorio. Si no, solicita los archivos del proyecto.

1.3 Navegar a la Carpeta de la App

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

PASO 2: Instalación de Dependencias

2.1 Verificar Node.js y Package Manager

```
node --version # Debe ser >= 18
yarn --version # Package manager preferido
```

2.2 Instalar Dependencias

```
# Instalar todas las dependencias
yarn install

# Si hay problemas, limpiar cache
yarn cache clean
rm -rf node_modules
yarn install
```

PASO 3: Configuración de Base de Datos

3.1 Inicializar PostgreSQL

En DeepAgent:

```
# Usar el tool initialize_postgres_db
```

3.2 Configurar Variables de Entorno

Crear archivo `.env` basado en `.env.example`:

```
# Base de Datos
DATABASE_URL="postgresql://usuario:contraseña@host:5432/escalafin"

# Autenticación
NEXTAUTH_SECRET="tu_secreto_super_seguro_de_32_caracteres_o_mas"
NEXTAUTH_URL="http://localhost:3000"

# Ambiente
NODE_ENV="development"
NEXTAUTH_DEBUG=false

# Openpay (Pagos)
OPENPAY_MERCHANT_ID="tu_merchant_id"
OPENPAY_PRIVATE_KEY="tu_private_key"
OPENPAY_PUBLIC_KEY="tu_public_key"
OPENPAY_BASE_URL="https://sandbox-api.openpay.mx/v1"

# AWS S3 (Archivos)
AWS_ACCESS_KEY_ID="tu_access_key"
AWS_SECRET_ACCESS_KEY="tu_secret_key"
AWS_BUCKET_NAME="tu_bucket_name"
AWS_REGION="us-east-1"
AWS_FOLDER_PREFIX="escalafin/"

# WhatsApp (EvolutionAPI)
EVOLUTION_API_URL="https://tu-evolution-api.com"
EVOLUTION_API_TOKEN="tu_token_evolution"
EVOLUTION_INSTANCE_NAME="escalafin"
```

3.3 Sincronizar Base de Datos

```
# Generar cliente Prisma
yarn prisma generate

# Crear tablas en base de datos
yarn prisma db push

# Ejecutar seed para datos de prueba
yarn prisma db seed
```

PASO 4: Configuración de Servicios Externos

4.1 Configurar Autenticación

```
# En DeepAgent, usar tool:
initialize_auth
```

4.2 Configurar Almacenamiento Cloud

```
# En DeepAgent, usar tool:
initialize_cloud_storage
```

4.3 Configurar APIs LLM (Opcional)

```
# En DeepAgent, usar tool:
initialize_llm_apis
```

PASO 5: Verificación y Testing

5.1 Verificar Compilación

```
# Test de build
yarn build

# Si hay errores, revisar logs y corregir
```

5.2 Iniciar Servidor de Desarrollo

```
yarn dev
```

5.3 Probar Funcionalidades

Abrir <http://localhost:3000> y probar:

- Login con cuentas de prueba
- Navegación en dashboards
- Funcionalidades principales

PASO 6: Testing Completo del Proyecto

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

CUENTAS DE PRUEBA

Credenciales por Defecto:

ADMINISTRADOR:

- Email: **admin**@escalafin.com
- Password: admin123
- Acceso: Dashboard completo, gestión usuarios, configuraciones

ASESOR:

- Email: asesor@escalafin.com
- Password: asesor123
- Acceso: Clientes asignados, préstamos, cobranza móvil

CLIENTE:

- Email: cliente@escalafin.com
 - Password: cliente123
 - Acceso: Sus préstamos, pagos, historial
-

ESTRUCTURA DEL PROYECTO

escalafin_mvp/	
app/	# Aplicación Next.js
api/	# API Routes (31 endpoints)
auth/	# Autenticación NextAuth
clients/	# Gestión de clientes
loans/	# Gestión de préstamos
payments/	# Sistema de pagos
credit-applications/	# Solicitudes de crédito
files/	# Sistema de archivos
reports/	# Reportes y analytics
admin/	# APIs administrativas
webhooks/	# Webhooks externos
admin/	# Dashboard Administrador
asesor/	# Dashboard Asesor
cliente/	# Dashboard Cliente
mobile/	# Módulo Móvil Cobranza
auth/	# Páginas de autenticación
components/	# Componentes React
ui/	# Componentes base (Shadcn)
forms/	# Formularios específicos
tables/	# Tablas de datos
auth/	# Componentes autenticación
payments/	# Componentes de pagos
files/	# Componentes archivos
notifications/	# Sistema notificaciones
analytics/	# Componentes analytics
lib/	# Utilidades y configuraciones
auth.ts	# Config NextAuth
db.ts	# Cliente Prisma
utils.ts	# Utilidades generales
aws-config.ts	# Configuración AWS S3
s3.ts	# Servicio S3
storage-service.ts	# Servicio almacenamiento dual
whatsapp-service.ts	# Servicio WhatsApp
openpay.ts	# Cliente Openpay
validations/	# Esquemas validación Zod
prisma/	# Configuración base de datos
schema.prisma	# Esquema principal
seed.ts	# Datos de prueba
migrations/	# Migraciones (si aplica)
public/	# Archivos estáticos
styles/	# Estilos globales
types/	# Definiciones TypeScript
middleware.ts	# Middleware Next.js
next.config.js	# Configuración Next.js
tailwind.config.js	# Configuración Tailwind
package.json	# Dependencias proyecto
.env.example	# Variables de entorno ejemplo
docs/	# Documentación del proyecto
README.md	# Información general

FUNCIONALIDADES IMPLEMENTADAS

MÓDULOS COMPLETOS:

1. Sistema de Autenticación

- **Multi-rol:** Admin, Asesor, Cliente
- **NextAuth.js** con base de datos
- **Middleware** protección de rutas
- **Sessions** seguras con cookies

2. CRM - Gestión de Clientes

- **CRUD completo** de clientes
- **Información completa:** Personal, financiera, laboral
- **Asignación** de asesores
- **Filtros avanzados** y búsquedas
- **Migración** de datos legacy

3. Sistema de Préstamos

- **Tipos múltiples:** Personal, Empresarial, Hipotecario, Auto, Educativo
- **Calculadora automática** de amortización
- **Tabla de pagos** programados detallada
- **Estados:** Activo, Pagado, Mora, Cancelado
- **Workflow completo** de gestión

4. Aplicaciones de Crédito

- **Workflow completo** de solicitudes
- **Estados:** Pendiente, Revisión, Aprobada, Rechazada
- **Evaluación automática** de scoring crediticio
- **Generación automática** de préstamo al aprobar
- **Comentarios** y seguimiento de revisión

5. Sistema de Pagos Online

- **Integración completa** con Openpay
- **Métodos múltiples:** Tarjeta, SPEI, tiendas
- **Webhooks automáticos** para reconciliación
- **Referencias únicas** de transacciones
- **Historial detallado** de movimientos

6. Pagos en Efectivo

- **Módulo móvil** para asesores de cobranza
- **Registro manual** de pagos efectivo
- **Generación automática** de recibos
- **Sincronización** con sistema principal
- **Geolocalización** (preparado para implementar)

7. Sistema de Archivos Cloud

- **Almacenamiento dual:** Local + AWS S3
- **Panel administrativo** de configuración
- **Upload drag & drop** con preview

- **Categorización automática** por tipo
- **Control acceso** por roles de usuario
- **URLs firmadas** para descarga segura

8. Notificaciones WhatsApp

- **Integración** con EvolutionAPI
- **Notificaciones automáticas** post-pago
- **Configuración por cliente** (opt-in/opt-out)
- **Templates personalizables** de mensajes
- **Test conectividad** en tiempo real
- **Webhook bidireccional** para mensajes

9. Analytics y Reportes

- **Dashboard ejecutivo** con KPIs principales
- **Gráficos interactivos** (Chart.js, Recharts)
- **Exportación múltiple:** PDF, Excel, CSV
- **Métricas tiempo real:** Préstamos, pagos, mora
- **Filtros avanzados** por período y estado
- **Series temporales** para análisis de tendencias

10. Sistema de Auditoría

- **Logging automático** de todas las acciones
- **Trazabilidad completa** por usuario
- **Exportación** de logs de auditoría
- **Estadísticas** de uso del sistema
- **Control acceso** a logs por rol

INTEGRACIONES EXTERNAS:

Openpay (Pagos)

- Procesamiento tarjetas crédito/débito
- SPEI (transferencias bancarias)
- Pagos en tiendas de conveniencia
- Webhooks para confirmación automática
- Modo sandbox y producción

EvolutionAPI (WhatsApp)

- Conexión a WhatsApp Business API
- Envío automático de mensajes
- Templates de notificación configurables
- Webhook para mensajes recibidos
- Panel de configuración admin

AWS S3 (Almacenamiento)

- Upload directo a cloud
- URLs firmadas para seguridad
- Configuración por buckets
- Fallback a almacenamiento local
- Panel de configuración dual



BASE DE DATOS - ESQUEMA PRINCIPAL

Tablas Principales:

users	-- Sistema de usuarios multi-rol
clients	-- CRM completo de clientes
credit_applications	-- Solicitudes de crédito
loans	-- Préstamos activos
amortization_schedule	-- Tabla de amortización
payments	-- Historial de pagos
files	-- Sistema de archivos
whatsapp_notification_settings	-- Config WhatsApp por cliente
system_config	-- Configuraciones del sistema
audit_logs	-- Logs de auditoría

Relaciones Clave:

- Cliente → Múltiples Solicitudes → Préstamos → Pagos
- Usuario → Cliente (1:1 opcional)
- Asesor → Múltiples Clientes asignados
- Préstamo → Tabla Amortización → Pagos Realizados

Datos de Prueba Incluidos:

- **7 usuarios** con diferentes roles
- **5 clientes** con información completa
- **3 préstamos** en diferentes estados
- **Tabla amortización** calculada automáticamente
- **Pagos realizados** con referencias reales



COMANDOS ESENCIALES

Desarrollo Local:

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

# Iniciar desarrollo
yarn dev                # http://localhost:3000

# Build de producción
yarn build              # Verificar que compila sin errores

# Servidor de producción
yarn start              # Después del build

# Base de datos
yarn prisma generate    # Generar cliente Prisma
yarn prisma db push     # Sincronizar esquema
yarn prisma db seed     # Poblar con datos de prueba
yarn prisma studio      # Explorador BD (localhost:5555)
```


Verificación de Estado:

```
# Test completo del build
yarn build && echo "✅ Build exitoso" || echo "❌ Build falló"

# Verificar APIs principales
curl http://localhost:3000/api/clients
curl http://localhost:3000/api/loans
curl http://localhost:3000/api/payments/transactions

# Verificar que Next.js compila sin errores TypeScript
yarn build 2>&1 | grep -i error || echo "✅ Sin errores TS"
```



SOLUCIÓN DE PROBLEMAS COMUNES

Error: Build Falla

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

Error: Base de Datos no Conecta

```
# Verificar variables de entorno
cat .env | grep DATABASE_URL

# Regenerar cliente Prisma
yarn prisma generate
yarn prisma db push
```

Error: Dependencias Faltantes

```
# Verificar package.json vs node_modules
yarn install --check-files

# Instalar dependencias específicas si faltan
yarn add @prisma/client
yarn add next-auth
```

Error: TypeScript

```
# Verificar configuración TypeScript
cat tsconfig.json

# Verificar tipos instalados
yarn add --dev @types/node @types/react
```

DESPLIEGUE A PRODUCCIÓN

Preparación para Deploy:

1. Variables de Entorno Producción

```
DATABASE_URL="postgresql://prod_user:prod_pass@prod_host:5432/escalafin_prod"
NEXTAUTH_URL="https://tu-dominio.com"
NEXTAUTH_SECRET="secreto_produccion_seguro_32_chars"
```

```
# Cambiar de sandbox a producción
OPENPAY_BASE_URL="https://api.openpay.mx/v1"
```

```
# Configurar S3 real si se usa
AWS_BUCKET_NAME="escalafin-prod-files"
```

2. Build de Producción

```
NODE_ENV=production yarn build
```

3. Plataformas Recomendadas

- **Vercel:** Deploy automático desde GitHub
 - **Netlify:** Integración con Git
 - **Railway:** Base de datos incluida
 - **DigitalOcean App Platform:** VPS escalable
 - **AWS:** Infraestructura completa
-



CONFIGURACIÓN GITHUB

Crear Repositorio en GitHub:

1. Crear .gitignore

```
# Dependencies
node_modules/
.pnp
.pnp.js

# Production build
.next/
out/
build/

# Environment variables
.env
.env*.local

# Debug logs
npm-debug.log*
yarn-debug.log*
yarn-error.log*

# Runtime data
pids
*.pid
*.seed
*.pid.lock

# Optional npm cache directory
.npm

# Optional REPL history
.node_repl_history

# Prisma
/prisma/migrations/dev.db*

# OS generated files
.DS_Store
.DS_Store?
.*
.Spotlight-V100
.Trashes
ehthumbs.db
Thumbs.db

# IDE
.vscode/
.idea/

# Local storage
/public/uploads/*
!/public/uploads/.gitkeep
```

2. Preparar para GitHub

```
cd /home/ubuntu/escalafin_mvp
git init
git add .
git commit -m "Initial commit - EscalaFin MVP v2.1.0"

# Conectar con repo remoto
git remote add origin https://github.com/tuusuario/escalafin-mvp.git
git branch -M main
git push -u origin main
```

3. Configurar GitHub Actions (Opcional)

Crear `.github/workflows/ci.yml` para CI/CD automático.



SEGURIDAD Y MEJORES PRÁCTICAS

Variables Sensibles:

- ☒ `.env` en `.gitignore`
- ☒ `NEXTAUTH_SECRET` generado automáticamente
- ☒ Passwords hasheados con `bcrypt`
- ☒ Validación input con `Zod`
- ☒ Middleware protección rutas
- ☒ Control acceso por roles

Recomendaciones Producción:

- Usar HTTPS obligatorio
- Configurar CORS apropiadamente
- Implementar rate limiting
- Configurar headers de seguridad
- Backup automático de BD
- Monitoreo de errores (Sentry)



ROADMAP DE DESARROLLO

Próximas Funcionalidades Sugeridas:

Prioridad ALTA:

- ☐ **PWA** para móviles (app instalable)
- ☐ **Funcionalidad offline** con sincronización
- ☐ **Geolocalización** para rutas de cobranza
- ☐ **Push notifications** nativas
- ☐ **Dashboard ejecutivo** avanzado

Prioridad MEDIA:

- ☐ **Sistema de backup** automatizado

- ☐ **Monitoreo** y alertas de sistema
- ☐ **API pública** con documentación
- ☐ **Integración** con más pasarelas de pago
- ☐ **Reportes avanzados** con BI

Prioridad BAJA:

- ☐ **Multi-idioma** (i18n)
- ☐ **Temas personalizables**
- ☐ **Integración CRM** externa
- ☐ **Módulo de marketing**
- ☐ **Sistema de referidos**

SOPORTE Y DOCUMENTACIÓN

Documentación Técnica Disponible:

- `DEEPAGENT_CONTINUATION_GUIDE_FINAL.md` - Guía continuación técnica
- `SCHEMA.md` - Esquema completo base de datos
- `NEXT_STEPS.md` - Roadmap de desarrollo
- `FASE_2B_COMPLETADA.md` - Funcionalidades implementadas
- `analisis_funcionalidad.md` - Estado detallado módulos

Logs y Debugging:

```
# Ver logs de aplicación
yarn dev | tee app.log

# Ver logs de base de datos
yarn prisma studio

# Verificar estructura BD
\d+ en psql para ver tablas
```

Recursos Externos:

- [Next.js Docs](https://nextjs.org/docs) (https://nextjs.org/docs)
- [Prisma Docs](https://www.prisma.io/docs) (https://www.prisma.io/docs)
- [NextAuth.js Docs](https://next-auth.js.org) (https://next-auth.js.org)
- [Openpay API](https://www.openpay.mx/docs/) (https://www.openpay.mx/docs/)
- [EvolutionAPI Docs](https://doc.evolution-api.com) (https://doc.evolution-api.com)

CHECKLIST DE IMPORTACIÓN

Configuración Inicial:

- ☐ Proyecto creado en DeepAgent
- ☐ Código fuente importado/clonado
- ☐ Dependencias instaladas (`yarn install`)

- ☐ Variables de entorno configuradas (`.env`)

Base de Datos:

- ☐ PostgreSQL inicializado
- ☐ Cliente Prisma generado (`yarn prisma generate`)
- ☐ Esquema sincronizado (`yarn prisma db push`)
- ☐ Datos de prueba cargados (`yarn prisma db seed`)

Servicios:

- ☐ Autenticación configurada (NextAuth)
- ☐ Almacenamiento cloud configurado (S3 opcional)
- ☐ APIs externas configuradas (Openpay, WhatsApp)

Testing:

- ☐ Build exitoso (`yarn build`)
- ☐ Servidor de desarrollo funcional (`yarn dev`)
- ☐ Login con cuentas de prueba exitoso
- ☐ Navegación dashboards funcional
- ☐ APIs respondiendo correctamente

Finalización:

- ☐ Repositorio GitHub creado
- ☐ Código subido a GitHub
- ☐ Documentación actualizada
- ☐ Deploy a producción (opcional)

ESTADO FINAL

EscalaFin MVP está COMPLETO y LISTO para:

- Uso inmediato en desarrollo
- Deploy a producción
- Extensión con nuevas funcionalidades
- Escalamiento para más usuarios

El sistema incluye TODO lo necesario para:

- Gestión completa de préstamos
- Procesamiento de pagos online
- Notificaciones automáticas WhatsApp
- Sistema de archivos empresarial
- Analytics y reportes ejecutivos
- Módulo móvil de cobranza
- Multi-tenancy por roles

INFORMACIÓN DE CONTACTO

Proyecto: EscalaFin MVP v2.1.0

Creado: Septiembre 2025

Tecnología: Next.js 14 + PostgreSQL + Integrations

Status:  PRODUCCIÓN READY

Para soporte técnico, revisar documentación incluida en el proyecto.

¡Importación exitosa garantizada! 