📋 GUÍA COMPLETA PARA GITHUB - ESCALAFIN MVP

Versión: 2.1.0
Fecha: Septiembre 22, 2025

🚀 PREPARACIÓN PARA GITHUB

PASO 1: Crear .gitignore Completo

```
# Dependencies
node_modules/
.pnp
.pnp.js
yarn-error.log*

# Production
.next/
out/
build/
dist/

# Environment variables - CRÍTICO: NO subir credenciales
.env
.env.local
.env.development.local
.env.test.local
.env.production.local

# Database
*.db
*.db-journal
/prisma/dev.db*

# Debug
npm-debug.log*
yarn-debug.log*
yarn-error.log*
.pnpm-debug.log*
lerna-debug.log*

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

# Coverage directory used by tools like istanbul
coverage/
*.lcov

# nyc test coverage
.nyc_output

# Dependency directories
jspm_packages/

# TypeScript cache
*.tsbuildinfo

# Optional npm cache directory
.npm

# Optional eslint cache
.eslintcache

# Microbundle cache
.rpt2_cache/
.rts2_cache_cjs/
.rts2_cache_es/
.rts2_cache_umd/
```

```
# Optional REPL history
.node_repl_history

# Output of 'npm pack'
*.tgz

# Yarn Integrity file
.yarn-integrity

# parcel-bundler cache (https://parceljs.org/)
.cache
.parcel-cache

# Next.js build output
.next
out

# Nuxt.js build / generate output
.nuxt
dist

# Storybook build outputs
.out
.storybook-out

# Temporary folders
tmp/
temp/

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

# Logs
logs
*.log

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

# Editor directories and files
.vscode/
!.vscode/extensions.json
.idea/
*.suo
*.ntvs*
*.njsproj
*.sln
*.sw?

# Local env files
.env*.local
```

```
# Vercel
.vercel

# Prisma
prisma/*.db
prisma/*.db-journal

# Uploads - Archivos locales NO van a GitHub
/app/public/uploads/*
!/app/public/uploads/.gitkeep

# AWS y configs sensibles
.aws/
aws-exports.js

# Certificates
*.pem
*.key
*.crt
*.csr

# Backup files
*.backup
*.bak
*.tmp

# Test coverage
coverage/
.coverage/

# Playwright
/test-results/
/playwright-report/
/playwright/.cache/

# MacOS
*.DS_Store

# Windows
Thumbs.db
ehthumbs.db
Desktop.ini

# Linux
*~
```

PASO 2: Crear README.md Profesional

```
# 🏦 EscalaFin MVP - Sistema de Gestión de Préstamos

[![Next.js](https://img.shields.io/badge/Next.js-14-black?logo=next.js)](https://
nextjs.org/)
[![PostgreSQL](https://img.shields.io/badge/PostgreSQL-16-blue?logo=postgresql)](ht-
tps://postgresql.org/)
[![TypeScript](https://img.shields.io/badge/TypeScript-5.2-blue?logo=typescript)](ht-
tps://typescriptlang.org/)
[![Tailwind CSS](https://img.shields.io/badge/Tailwind-3.3-06B6D4?logo=tailwindcss)]
(https://tailwindcss.com/)

## 📖 Descripción

EscalaFin es una plataforma completa para la gestión de préstamos y créditos, desar-
rollada con tecnologías modernas y diseñada para instituciones financieras pequeñas y
medianas.

## ✨ Características Principales

- 🏦 **CRM Completo** - Gestión integral de clientes
- 💰 **Sistema de Préstamos** - Con amortización automática
- 💳 **Pagos Online** - Integración con Openpay
- 📱 **WhatsApp** - Notificaciones automáticas
- 📊 **Analytics** - Dashboard ejecutivo con KPIs
- 👥 **Multi-rol** - Admin, Asesor, Cliente
- ☁️ **Cloud Storage** - AWS S3 integrado
- 📱 **Móvil** - Módulo de cobranza en efectivo

## 🛠️ Tecnologías

### Frontend
- **Next.js 14** - React Framework con App Router
- **TypeScript** - Tipado estático
- **Tailwind CSS** - Estilos utilitarios
- **Shadcn/ui** - Componentes UI modernos
- **Recharts** - Gráficos y analytics

### Backend
- **Next.js API Routes** - API integrada
- **PostgreSQL** - Base de datos principal
- **Prisma ORM** - Object-Relational Mapping
- **NextAuth.js** - Autenticación segura

### Integraciones
- **Openpay** - Procesamiento de pagos
- **EvolutionAPI** - WhatsApp Business
- **AWS S3** - Almacenamiento de archivos

## 🚀 Instalación

### Prerrequisitos
- Node.js >= 18
- PostgreSQL >= 13
- Yarn (recomendado)

### Configuración Local

1. **Clonar el repositorio**
   ```bash
   git clone https://github.com/tu-usuario/escalafin-mvp.git
   cd escalafin-mvp/app
   ```
```

2. **Instalar dependencias**
   ```bash
   yarn install
   ```

3. **Configurar variables de entorno**
   ```bash
   cp .env.example .env
   # Editar .env con tus configuraciones
   ```

4. **Configurar base de datos**
   ```bash
   yarn prisma generate
   yarn prisma db push
   yarn prisma db seed
   ```

5. **Iniciar desarrollo**
   ```bash
   yarn dev
   ```

🔧 Variables de Entorno

Crear archivo `.env` con las siguientes variables:

```env
# Base de Datos
DATABASE_URL="postgresql://usuario:pass@localhost:5432/escalafin"

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

# Openpay
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 (Opcional)
AWS_ACCESS_KEY_ID="tu_access_key"
AWS_SECRET_ACCESS_KEY="tu_secret_key"
AWS_BUCKET_NAME="tu_bucket"

# WhatsApp (Opcional)
EVOLUTION_API_URL="https://tu-api.com"
EVOLUTION_API_TOKEN="tu_token"
```

👥 Cuentas de Prueba

```
Admin: admin@escalafin.com / admin123
Asesor: asesor@escalafin.com / asesor123
Cliente: cliente@escalafin.com / cliente123
```

📊 Estructura del Proyecto

```
app/
├── api/                    # API Routes
├── admin/                  # Dashboard Admin
├── asesor/                 # Dashboard Asesor
├── cliente/                # Dashboard Cliente
├── components/             # Componentes React
├── lib/                    # Utilidades
├── prisma/                 # Configuración BD
└── public/                 # Archivos estáticos
```

🛡️ Seguridad

- Autenticación con NextAuth.js
- Protección de rutas por middleware
- Validación de entrada con Zod
- Hash de contraseñas con bcrypt
- Control de acceso por roles

📱 API Endpoints

Principales

- `GET/POST /api/clients` - Gestión de clientes
- `GET/POST /api/loans` - Gestión de préstamos
- `POST /api/payments/openpay` - Procesar pagos
- `GET /api/reports/*` - Reportes y analytics

🚀 Deploy

Vercel (Recomendado)

1. Conectar repositorio en Vercel
2. Configurar variables de entorno
3. Deploy automático desde main

Otras Plataformas

- Railway
- Netlify
- DigitalOcean App Platform
- AWS Amplify

🧪 Testing

```
# Build de producción
yarn build

# Linting
yarn lint

# Verificar tipos TypeScript
yarn type-check
```

📈 Roadmap

- [ ] PWA para móviles
- [ ] Modo offline
- [ ] Geolocalización
- [ ] Push notifications
- [ ] API pública
- [ ] Multi-idioma

🤝 Contribución

1. Fork del proyecto
2. Crear feature branch (`git checkout -b feature/nueva-caracteristica`)
3. Commit cambios (`git commit -m 'Agregar nueva característica'`)
4. Push branch (`git push origin feature/nueva-caracteristica`)
5. Crear Pull Request

📄 Licencia

Este proyecto está bajo la Licencia MIT - ver el archivo LICENSE (LICENSE) para detalles.

📞 Contacto

- Proyecto: EscalaFin MVP
- Versión: 2.1.0
- Estado: Producción Ready

Desarrollado con ❤️ para instituciones financieras modernas

````
### **PASO 3: Crear Package.json Scripts Adicionales**

```json
{
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "type-check": "tsc --noEmit",
    "db:generate": "prisma generate",
    "db:push": "prisma db push",
    "db:seed": "prisma db seed",
    "db:studio": "prisma studio",
    "db:migrate": "prisma migrate dev",
    "clean": "rm -rf .next node_modules",
    "reinstall": "yarn clean && yarn install"
  }
}
````

PASO 4: Configurar GitHub Actions (CI/CD)

Crear `.github/workflows/ci.yml`:
````

```yaml
name: CI/CD Pipeline

on:
  push:
    branches: [ main, develop ]
  pull_request:
    branches: [ main ]

jobs:
  build:
    runs-on: ubuntu-latest

    services:
      postgres:
        image: postgres:13
        env:
          POSTGRES_PASSWORD: postgres
          POSTGRES_DB: escalafin_test
        options: >-
          --health-cmd pg_isready
          --health-interval 10s
          --health-timeout 5s
          --health-retries 5
        ports:
          - 5432:5432

    steps:
    - name: Checkout code
      uses: actions/checkout@v4

    - name: Setup Node.js
      uses: actions/setup-node@v3
      with:
        node-version: '18'
        cache: 'yarn'

    - name: Install dependencies
      run: yarn install --frozen-lockfile

    - name: Generate Prisma Client
      run: yarn prisma generate
      env:
        DATABASE_URL: postgresql://postgres:postgres@localhost:5432/escalafin_test

    - name: Run TypeScript check
      run: yarn type-check

    - name: Run linting
      run: yarn lint

    - name: Build application
      run: yarn build
      env:
        DATABASE_URL: postgresql://postgres:postgres@localhost:5432/escalafin_test
        NEXTAUTH_SECRET: test-secret-for-ci
        NEXTAUTH_URL: http://localhost:3000

    - name: Run database migrations
      run: yarn prisma db push
      env:
        DATABASE_URL: postgresql://postgres:postgres@localhost:5432/escalafin_test
```

PASO 5: Crear SECURITY.md

```
# Política de Seguridad

## Versiones Soportadas

| Versión | Soporte |
| ------- | ------- |
| 2.1.x   | ✅      |
| 2.0.x   | ✅      |
| < 2.0   | ❌      |

## Reportar Vulnerabilidades

Si descubres una vulnerabilidad de seguridad, por favor:

1. **NO** abras un issue público
2. Envía un email a: security@escalafin.com
3. Incluye detalles técnicos de la vulnerabilidad
4. Proporciona pasos para reproducir

### Proceso

1. Reconocemos la recepción en 24 horas
2. Investigación inicial en 72 horas
3. Parche de seguridad en 7 días para vulnerabilidades críticas
4. Disclosure coordinado después del parche

## Medidas de Seguridad Implementadas

- Autenticación con NextAuth.js
- Hash de contraseñas con bcrypt
- Protección CSRF integrada
- Validación de entrada con Zod
- Control de acceso basado en roles
- Variables de entorno para secretos
```

PASO 6: Crear CONTRIBUTING.md

```
# Guía de Contribución

## Cómo Contribuir

### Configuración del Entorno

1. Fork del repositorio
2. Clona tu fork localmente
3. Instala dependencias: `yarn install`
4. Configura base de datos local
5. Ejecuta `yarn dev` para desarrollo

### Estándares de Código

#### TypeScript
- Tipado estricto habilitado
- Interfaces para todos los objetos complejos
- No usar `any` excepto casos justificados

#### Styling
- Tailwind CSS para estilos
- Componentes reutilizables en `/components`
- Diseño responsivo mobile-first

#### Git Workflow
- Ramas descriptivas: `feature/nueva-funcionalidad`
- Commits atómicos con mensajes claros
- Rebase antes de merge para historial limpio

### Process de Pull Request

1. **Pre-PR Checklist**
   - [ ] Código compila sin errores
   - [ ] Linting pasa (`yarn lint`)
   - [ ] TypeScript check pasa (`yarn type-check`)
   - [ ] Build de producción exitoso (`yarn build`)

2. **PR Template**
   ```
   ## Descripción
   Breve descripción de los cambios

   ## Tipo de Cambio
   - [ ] Bug fix
   - [ ] Nueva funcionalidad
   - [ ] Breaking change
   - [ ] Documentación

   ## Testing
   - [ ] Unit tests incluidos
   - [ ] Tested manualmente
   - [ ] Browser testing realizado

   ## Screenshots (si aplica)
   ```

### Estructura de Commits
```

tipo(alcance): descripción breve

Descripción más detallada si es necesario

Fixes #123

```
**Tipos válidos:**
- `feat`: nueva funcionalidad
- `fix`: corrección de bug
- `docs`: cambios en documentación
- `style`: formateo, punto y coma faltante
- `refactor`: código refactorizado
- `perf`: mejoras de performance
- `test`: agregar tests
- `chore`: tareas de mantenimiento

## Reportar Bugs

Usar el template de issue para bugs:
- Descripción clara del problema
- Pasos para reproducir
- Comportamiento esperado vs actual
- Screenshots si es relevante
- Información del navegador/OS

## Solicitar Funcionalidades

- Explicar el caso de uso
- Proponer implementación si es posible
- Considerar breaking changes
- Evaluar impacto en performance
```

🚀 COMANDOS PARA SUBIR A GITHUB

Inicialización del Repositorio

```
# Navegar al directorio del proyecto
cd /home/ubuntu/escalafin_mvp

# Inicializar Git (si no está inicializado)
git init

# Configurar usuario (reemplazar con tus datos)
git config user.name "Tu Nombre"
git config user.email "tu-email@ejemplo.com"

# Agregar todos los archivos
git add .

# Primer commit
git commit -m "feat: initial commit - EscalaFin MVP v2.1.0

- Sistema completo de gestión de préstamos
- Integración con Openpay para pagos
- NotificacionesWhatsApp con EvolutionAPI
- Sistema de archivos AWS S3
- Multi-rol: Admin, Asesor, Cliente
- Analytics y reportes completos
- Módulo móvil de cobranza

Funcionalidades completas:
- CRM de clientes con info financiera
- Workflow de solicitudes de crédito
- Calculadora automática de amortización
- Dashboard ejecutivo con KPIs
- Sistema de auditoría completo
- Almacenamiento dual local/cloud"

# Crear rama main si no existe
git branch -M main
```

Conectar con GitHub Remote

```
# Agregar repositorio remoto (reemplazar URL)
git remote add origin https://github.com/tu-usuario/escalafin-mvp.git

# Verificar remote
git remote -v

# Subir código inicial
git push -u origin main
```

Configurar Ramas de Desarrollo

```
# Crear rama develop para desarrollo activo
git checkout -b develop
git push -u origin develop

# Crear ramas de feature ejemplo
git checkout -b feature/mejoras-dashboard
git push -u origin feature/mejoras-dashboard
```

📋 ESTRUCTURA RECOMENDADA DE RELEASES

Tags de Versión

```
# Crear tag para versión actual
git tag -a v2.1.0 -m "Release v2.1.0 - Sistema Completo

Nuevas funcionalidades:
- ✅ CRM completo de clientes
- ✅ Sistema de préstamos con amortización
- ✅ Integración Openpay
- ✅ Notificaciones WhatsApp
- ✅ Sistema de archivos AWS S3
- ✅ Analytics y reportes
- ✅ Módulo móvil de cobranza
- ✅ Multi-rol y autenticación
- ✅ Sistema de auditoría

Tecnologías:
- Next.js 14 + TypeScript
- PostgreSQL + Prisma ORM
- Tailwind CSS + Shadcn UI
- NextAuth.js

Estado: ✅ Producción Ready"

# Subir tag
git push origin v2.1.0
```

Release Notes Template

```
# 🚀 Release v2.1.0 - Sistema Completo

**Fecha:** Septiembre 22, 2025
**Estado:** ✅ Producción Ready

## 🎯 Resumen
Esta versión marca la completitud del MVP de EscalaFin con todas las funcionalidades
básicas implementadas y probadas para uso en producción.

## ✨ Nuevas Funcionalidades

### 🏦 CRM de Clientes
- Gestión completa de información personal y financiera
- Asignación automática de asesores
- Filtros y búsquedas avanzadas
- Migración de datos legacy

### 💰 Sistema de Préstamos
- 5 tipos de préstamo soportados
- Calculadora automática de amortización
- Workflow de estados completo
- Tabla de pagos programados

### 💳 Procesamiento de Pagos
- Integración completa con Openpay
- Múltiples métodos de pago
- Webhooks automáticos
- Reconciliación de transacciones

### 📱 WhatsApp Business
- Integración con EvolutionAPI
- Notificaciones automáticas post-pago
- Configuración por cliente
- Templates personalizables

### 📊 Analytics y Reportes
- Dashboard ejecutivo con KPIs
- Gráficos interactivos
- Exportación PDF/Excel/CSV
- Métricas en tiempo real

### 👥 Sistema Multi-Rol
- 3 roles: Admin, Asesor, Cliente
- Dashboards personalizados por rol
- Control de acceso granular
- Middleware de protección

### ☁️ Sistema de Archivos
- Almacenamiento dual Local/AWS S3
- Panel de configuración admin
- Upload drag & drop
- URLs firmadas para seguridad

### 📱 Módulo Móvil
- Cobranza en efectivo para asesores
- Registro manual de pagos
- Sincronización automática
- Preparado para geolocalización

## 🔧 Mejoras Técnicas
- Build de producción optimizado
- TypeScript strict mode
```

```
- Validación completa con Zod
- Sistema de auditoría implementado
- Performance optimizada
- Hydration errors solucionados

## 📊 Métricas del Proyecto
- **31 API endpoints** funcionales
- **40+ páginas** implementadas
- **150+ archivos TypeScript**
- **45+ componentes React**
- **10 módulos completos**

## 🚨 Breaking Changes
Ninguno - Primera versión de producción

## 🐛 Bug Fixes
- Corregidos errores de build de producción
- Solucionados problemas de hidratación
- Arregladas sesiones NextAuth
- Optimizada sincronización de BD

## 📈 Performance
- Build time: ~45 segundos
- First Contentful Paint mejorado
- JavaScript bundle optimizado
- Database queries optimizadas

## 🔐 Seguridad
- Hash seguro de contraseñas
- Validación de entrada robusta
- Control de acceso implementado
- Variables sensibles protegidas

## 📋 Checklist de Deploy
- [ ] Variables de entorno configuradas
- [ ] Base de datos migrada
- [ ] Servicios externos conectados
- [ ] SSL configurado
- [ ] Monitoring habilitado

## 🔗 Enlaces Útiles
- [Documentación Completa](./docs/)
- [Guía de Instalación](./INSTALLATION.md)
- [API Documentation](./API.md)
- [Guía de Contribución](./CONTRIBUTING.md)

## 🤝 Contribuidores
- Desarrollador Principal: DeepAgent
- Testing: Equipo QA
- Diseño UX/UI: Equipo Design

---

**¡EscalaFin MVP listo para transformar la gestión de préstamos! 🎉**
```

🔒 CONFIGURACIÓN DE SEGURIDAD GITHUB

Branch Protection Rules

En GitHub → Settings → Branches:

```
{
  "required_status_checks": {
    "strict": true,
    "contexts": ["ci/github-actions"]
  },
  "enforce_admins": true,
  "required_pull_request_reviews": {
    "required_approving_review_count": 1,
    "dismiss_stale_reviews": true
  },
  "restrictions": null
}
```

Secretos del Repositorio

En GitHub → Settings → Secrets:

```
DATABASE_URL (para CI)
NEXTAUTH_SECRET (para CI)
OPENPAY_MERCHANT_ID (para deploy)
OPENPAY_PRIVATE_KEY (para deploy)
AWS_ACCESS_KEY_ID (para deploy)
AWS_SECRET_ACCESS_KEY (para deploy)
```

📊 TEMPLATE DE ISSUES

Bug Report (.github/ISSUE_TEMPLATE/bug_report.md)

```
---
name: Bug Report
about: Crear reporte para ayudar a mejorar EscalaFin
title: '[BUG] '
labels: bug
assignees: ''
---

**Describir el bug**
Descripción clara y concisa del problema.

**Para Reproducir**
Pasos para reproducir:
1. Ir a '...'
2. Hacer click en '....'
3. Scroll hasta '....'
4. Ver error

**Comportamiento Esperado**
Descripción clara de lo que esperabas que pasara.

**Screenshots**
Si aplica, agregar screenshots para explicar el problema.

**Desktop (completar):**
 - OS: [ej. iOS]
 - Browser [ej. chrome, safari]
 - Version [ej. 22]

**Información Adicional**
Agregar contexto adicional sobre el problema.
```

Feature Request (.github/ISSUE_TEMPLATE/feature_request.md)

```
---
name: Feature Request
about: Sugerir una nueva funcionalidad para EscalaFin
title: '[FEATURE] '
labels: enhancement
assignees: ''
---

**Tu feature request está relacionado a un problema?**
Descripción clara del problema. Ej. Siempre me frustra cuando [...]

**Describe la solución que te gustaría**
Descripción clara y concisa de lo que quieres que pase.

**Describe alternativas que consideraste**
Descripción de soluciones alternativas que consideraste.

**Contexto adicional**
Agregar contexto, screenshots sobre el feature request.
```

🎯 PRÓXIMOS PASOS DESPUÉS DE GITHUB

1. Configurar Deploy Automático

- Conectar Vercel/Netlify con GitHub
- Variables de entorno en plataforma
- Deploy branches: main → producción, develop → staging

2. Configurar Monitoreo

- Sentry para error tracking
- Google Analytics para métricas
- Uptime monitoring para disponibilidad

3. Documentación API

- Swagger/OpenAPI docs
- Postman collections
- SDK para desarrolladores

4. Testing Avanzado

- Jest para unit tests
- Cypress para E2E testing
- Lighthouse para performance

✅ CHECKLIST FINAL GITHUB

- [ ] Repositorio creado en GitHub
- [ ] .gitignore configurado correctamente
- [ ] README.md profesional creado
- [ ] Variables sensibles NO subidas
- [ ] Código inicial commiteado
- [ ] Branches principales creadas (main, develop)
- [ ] GitHub Actions configurado
- [ ] Issue templates creados
- [ ] Security policy establecida
- [ ] Branch protection habilitada
- [ ] Release v2.1.0 taggeada
- [ ] Deploy automático configurado

🎉 ¡EscalaFin MVP listo en GitHub para colaboración y deploy! 🚀