

🎯 Plan de Reorganización y Corrección - QuantPay Chain MVP

Fecha: 5 de Noviembre de 2025

Repositorio: [francoMengarelli/quantpaychain-mvpro](#)

Objetivo: Corregir commits con emails no verificados y optimizar el repositorio para despliegue en Vercel

📋 Resumen Ejecutivo

Este documento proporciona un plan de acción paso a paso para:

1. Corregir el commit más reciente con email problemático
2. Limpiar el historial de commits (opcional pero recomendado)
3. Asegurar que todos los futuros commits usen el email verificado
4. Optimizar el repositorio para despliegue exitoso en Vercel

🎯 Objetivos del Plan

Objetivo	Prioridad	Estado
Corregir commit HEAD con email problemático	🔴 CRÍTICA	Pendiente
Verificar configuración de Git local	🔴 CRÍTICA	Pendiente
Decidir estrategia de limpieza de historial	🟡 ALTA	Pendiente
Crear backup completo del repositorio	🟡 ALTA	Pendiente
Ejecutar correcciones necesarias	🟡 ALTA	Pendiente
Probar despliegue en Vercel	🟡 MEDIA	Pendiente
Documentar cambios realizados	🟡 MEDIA	Pendiente



ACCIÓN INMEDIATA - Corrección del Commit HEAD

Problema Identificado

```
Commit: 6a4fd3c151436d6b054977d3f633873357429aa4
Autor: QuantPay Chain Bot <quantpaychain@example.com>
Fecha: 2025-11-05T16:48:20+00:00
Rama: main (HEAD)
```

Este commit es el más reciente en la rama `main` y utiliza un email no verificado que **causará problemas en Vercel**.

⚡ Solución Rápida (Opción Recomendada)

Método: Revert + Nuevo Commit Limpio

Esta es la opción más segura y no requiere force push.

Paso 1: Crear backup de seguridad

```
cd /home/ubuntu/github_repos/quantpaychain-mvpro

# Crear backup completo
git bundle create ~/quantpaychain-mvpro-backup-$(date +%Y%m%d-%H%M%S).bundle --all

# Verificar el backup
git bundle verify ~/quantpaychain-mvpro-backup-*.bundle
```

Paso 2: Revisar los cambios del commit problemático

```
# Ver qué cambió en el commit problemático
git show 6a4fd3c --stat

# Ver el diff completo
git show 6a4fd3c
```

Paso 3: Hacer revert del commit problemático

```
# Asegurarse de estar en main
git checkout main

# Verificar configuración de git (IMPORTANTE)
git config user.name "Franco Mengarelli"
git config user.email "fmengarelli@gmail.com"

# Hacer revert del commit problemático
git revert 6a4fd3c --no-edit

# Esto crea un nuevo commit que deshace los cambios
```

Paso 4: Reaplicar los cambios correctamente

```
# Cherry-pick los cambios pero con autoría correcta
git cherry-pick 6a4fd3c --no-commit

# Revisar los cambios
git status
git diff --cached

# Hacer commit con email correcto
git commit -m "fix: Resolve deployment configuration and enhance QPC v2 Core discoverability

- Remove incompatible i18n configuration from next.config.js
- Update vercel.json deployment settings
- Add libsodium-wrappers type declarations

(Reapplied changes from 6a4fd3c with verified email)"
```

Paso 5: Verificar y push

```
# Verificar el nuevo commit
git log -1 --format="%H|%an|%ae|%s"

# Debería mostrar: Franco Mengarelli <fmengarelli@gmail.com>

# Push a remote
git push origin main
```

Ventajas de este método:

- No requiere force push
- Mantiene historial completo
- Seguro para colaboradores
- Fácil de revertir si algo sale mal

Desventajas:

- Añade 2 commits adicionales al historial (revert + reapply)
- El commit problemático sigue en el historial (pero no en HEAD)

Soluciones Alternativas

Opción B: Amend del Commit (Requiere Force Push)

 **ADVERTENCIA:** Solo usar si eres el único trabajando en el repositorio o si coordinas con todo el equipo.

```

# Paso 1: Backup
git bundle create ~/backup-before-amend.bundle --all

# Paso 2: Reset al commit anterior
git reset --soft HEAD~1

# Paso 3: Verificar configuración
git config user.name "Franco Mengarelli"
git config user.email "fmengarelli@gmail.com"

# Paso 4: Hacer nuevo commit con mismos cambios
git commit -m "fix: Resolve deployment configuration and enhance QPC v2 Core discoverability

- Remove incompatible i18n configuration from next.config.js
- Update vercel.json deployment settings
- Add libsodium-wrappers type declarations"

# Paso 5: Force push (!CUIDADO!)
git push --force-with-lease origin main

```

Ventajas:

- Historial limpio
- Elimina completamente el commit problemático

Desventajas:

- Requiere force push
- Puede causar problemas si otros tienen clones del repo
- Más riesgoso

Opción C: Ignorar y Continuar

 **NO RECOMENDADO** pero mencionado para completitud.

Simplemente hacer un nuevo commit encima y esperar que Vercel no rechace el despliegue.

```

# Crear un commit vacío con email correcto para "limpiar" el HEAD
git commit --allow-empty -m "chore: Update commit authorship verification"

git push origin main

```

Limpieza Completa del Historial (Opcional)

¿Deberías limpiar el historial completo?

Criterio	Sí Limpiar	No Limpiar
Múltiples colaboradores	✗	✓
Repositorio privado y solo tú	✓	-
Problemas activos de despliegue	✓	-
Historial importante para auditoría	✗	✓
Tiempo disponible	✓	✗

Estrategia de Limpieza Completa

Si decides limpiar todo el historial, aquí está el plan:

Paso 1: Preparación y Backup

```
cd /home/ubuntu/github_repos/quantpaychain-mvpro

# Backup completo del repositorio actual
git bundle create ~/quantpaychain-mvpro-FULL-BACKUP-$(date +%Y%m%d-%H%M%S).bundle --all

# Crear copia de trabajo
cp -r /home/ubuntu/github_repos/quantpaychain-mvpro /home/ubuntu/github_repos/quant-paychain-mvpro-cleanup
cd /home/ubuntu/github_repos/quantpaychain-mvpro-cleanup
```

Paso 2: Crear archivo .mailmap

Este archivo mapea emails antiguos al email correcto:

```
cat > .mailmap << 'EOF'
# Map all bot emails to verified email
Franco Mengarelli <fmengarelli@gmail.com> QuantPay Chain Bot <quantpay-chain@example.com>
Franco Mengarelli <fmengarelli@gmail.com> QuantPay AI <ai@quantpaychain.com>
Franco Mengarelli <fmengarelli@gmail.com> Abacus AI Agent <agent@abacus.ai>
Franco Mengarelli <fmengarelli@gmail.com> QuantPay Chain Development Team <quant-pay@quantpaychain.org>
EOF

git add .mailmap
git commit -m "chore: Add mailmap for email consistency"
```

Paso 3: Reescribir historial con git filter-repo

⚠ ATENCIÓN: Esta operación es IRREVERSIBLE y cambiará todos los commits.

```

# Instalar git-filter-repo si no está instalado
pip3 install git-filter-repo

# Crear script de reescritura
cat > /tmp/fix-authors.sh << 'EOF'
#!/bin/bash

# Lista de emails problemáticos
PROBLEMATIC_EMAILS=(
    "quantpaychain@example.com"
    "ai@quantpaychain.com"
    "agent@abacus.ai"
    "quantpay@quantpaychain.org"
)

# Email y nombre correcto
CORRECT_NAME="Franco Mengarelli"
CORRECT_EMAIL="fmengarelli@gmail.com"

# Obtener email del commit actual
AUTHOR_EMAIL=$(git log -1 --pretty=format:'%ae')

# Verificar si es un email problemático
for email in "${PROBLEMATIC_EMAILS[@]}"; do
    if [ "$AUTHOR_EMAIL" = "$email" ]; then
        # Cambiar autor y committer
        export GIT_AUTHOR_NAME="$CORRECT_NAME"
        export GIT_AUTHOR_EMAIL="$CORRECT_EMAIL"
        export GIT_COMMITTER_NAME="$CORRECT_NAME"
        export GIT_COMMITTER_EMAIL="$CORRECT_EMAIL"
        break
    fi
done
EOF

chmod +x /tmp/fix-authors.sh

# Ejecutar reescritura (ESTO TOMA TIEMPO)
git filter-repo --commit-callback '
import sys
import os

# Emails problemáticos
problematic = [
    b"quantpaychain@example.com",
    b"ai@quantpaychain.com",
    b"agent@abacus.ai",
    b"quantpay@quantpaychain.org"
]

correct_name = b"Franco Mengarelli"
correct_email = b"fmengarelli@gmail.com"

if commit.author_email in problematic:
    commit.author_name = correct_name
    commit.author_email = correct_email

if commit.committer_email in problematic:
    commit.committer_name = correct_name
    commit.committer_email = correct_email
'

```

Paso 4: Verificar resultado

```
# Ver todos los commits con emails
git log --all --format="%H|%an|%ae|%s" | grep -v "fmengarelli@gmail.com"

# Si no hay resultados, ¡éxito! Todos los commits usan el email correcto

# Ver estadísticas
git log --all --format="%ae" | sort | uniq -c
```

Paso 5: Push forzado (PUNTO DE NO RETORNO)

```
# Agregar el remote nuevamente (git filter-repo lo elimina)
git remote add origin https://github.com/francoMengarelli/quantpaychain-mvpro.git

# Force push (!CUIDADO!)
git push --force --all origin

# Push tags también
git push --force --tags origin
```

Verificación de Configuración de Git

Antes de hacer cualquier commit futuro, verifica tu configuración:

Configuración Global (Para todos los repositorios)

```
# Ver configuración actual
git config --global user.name
git config --global user.email

# Si no es correcta, configurar:
git config --global user.name "Franco Mengarelli"
git config --global user.email "fmengarelli@gmail.com"

# Verificar
git config --global --list | grep user
```

Configuración Local (Solo para este repositorio)

```
cd /home/ubuntu/github_repos/quantpaychain-mvpro

# Ver configuración local
git config user.name
git config user.email

# Configurar si no es correcta
git config user.name "Franco Mengarelli"
git config user.email "fmengarelli@gmail.com"

# Verificar
git config --list | grep user
```

Script de Verificación Automática

Crear un script que verifique antes de cada commit:

```
cat > /home/ubuntu/github_repos/quantpaychain-mvpro/.git/hooks/pre-commit << 'EOF'
#!/bin/bash

# Verificar que el email configurado sea el correcto
CURRENT_EMAIL=$(git config user.email)
REQUIRED_EMAIL="fmengarelli@gmail.com"

if [ "$CURRENT_EMAIL" != "$REQUIRED_EMAIL" ]; then
    echo "✗ ERROR: Email incorrecto configurado"
    echo "  Actual: $CURRENT_EMAIL"
    echo "  Requerido: $REQUIRED_EMAIL"
    echo ""
    echo "Ejecuta:"
    echo "  git config user.email '$REQUIRED_EMAIL'"
    exit 1
fi

echo "✓ Email verificado: $CURRENT_EMAIL"
exit 0
EOF

chmod +x /home/ubuntu/github_repos/quantpaychain-mvpro/.git/hooks/pre-commit

# Probar el hook
/home/ubuntu/github_repos/quantpaychain-mvpro/.git/hooks/pre-commit
```



Asegurar Email Verificado en GitHub

Verificar en GitHub

1. Ir a: <https://github.com/settings/emails>
2. Verificar que `fmengarelli@gmail.com` está:
 - Listado
 - Verificado (con checkmark verde)
 - Marcado como “Primary” (opcional pero recomendado)

Configurar Email Privado (Opcional)

GitHub ofrece emails privados en formato: `ID+username@users.noreply.github.com`

Si prefieres usar email privado:

```
# Ejemplo (reemplaza con tu ID real):
git config user.email "213843293+francoMengarelli@users.noreply.github.com"
```

Habilitar Protección de Email

En GitHub Settings:

1. Ir a: <https://github.com/settings/emails>
2. Marcar: “Block command line pushes that expose my email”
3. Esto **bloqueará** automáticamente commits con emails no verificados

Plan de Despliegue en Vercel

Una vez corregidos los problemas de email:

Paso 1: Verificar Estado del Repositorio

```
cd /home/ubuntu/github_repos/quantpaychain-mvpro

# Verificar que HEAD tiene email correcto
git log -1 --format="%an <%ae>%n%"

# Debería mostrar: Franco Mengarelli <fmengarelli@gmail.com>
```

Paso 2: Verificar Configuración de Vercel

Revisar archivos de configuración:

```
# Ver vercel.json
cat vercel.json

# Ver next.config.js
cat quantpaychain-mvp/frontend/app/next.config.js

# Verificar package.json
cat quantpaychain-mvp/frontend/app/package.json
```

Paso 3: Test Local Antes de Despliegue

```
# Instalar dependencias
cd quantpaychain-mvp/frontend/app
npm install

# Build de producción
npm run build

# Si el build es exitoso, Vercel probablemente también lo será
```

Paso 4: Desplegar a Vercel

Opciones:

Opción A: Desde Vercel Dashboard

1. Ir a: <https://vercel.com/dashboard>
2. Import Project → GitHub
3. Seleccionar `francoMengarelli/quantpaychain-mvpro`
4. Configure build settings:
 - Framework Preset: Next.js
 - Root Directory: `quantpaychain-mvp/frontend/app`
 - Build Command: `npm run build`
 - Output Directory: `.next`

Opción B: Desde CLI

```
# Instalar Vercel CLI
npm i -g vercel

# Login
vercel login

# Deploy
cd /home/ubuntu/github_repos/quantpaychain-mvpro
vercel --prod
```

Paso 5: Verificar Despliegue

Después del despliegue:

1. Verificar que la página carga correctamente
 2. Revisar logs en Vercel Dashboard
 3. Probar funcionalidad principal
 4. Verificar que no hay errores de autenticación de GitHub
-



Checklist de Verificación Post-Corrección

Verificaciones de Git

- [] Configuración global de Git usa fmengarelli@gmail.com
- [] Configuración local del repo usa fmengarelli@gmail.com
- [] Commit HEAD tiene email verificado
- [] No hay archivos sin commit (git status clean)
- [] Pre-commit hook instalado y funcionando

Verificaciones de GitHub

- [] Email verificado en GitHub Settings
- [] Protección de email habilitada (opcional)
- [] Push exitoso al repositorio remoto
- [] GitHub muestra commit con email correcto

Verificaciones de Código

- [] npm install funciona sin errores
- [] npm run build completa exitosamente
- [] TypeScript compila sin errores
- [] No hay dependencias faltantes

Verificaciones de Vercel

- [] Repositorio conectado a Vercel
 - [] Build settings configurados correctamente
 - [] Variables de entorno configuradas (si necesario)
 - [] Despliegue exitoso sin errores
 - [] Sitio accesible en URL de Vercel
-

Workflow Futuro Recomendado

Para evitar futuros problemas:

1. Configuración de Desarrollo

```
# Crear alias útiles en ~/.bashrc o ~/.zshrc
echo 'alias git-check-email="git config user.email"' >> ~/.bashrc
echo 'alias git-set-franco="git config user.name \"Franco Mengarelli\" && git config
user.email \"fmengarelli@gmail.com\""' >> ~/.bashrc

source ~/.bashrc

# Ahora puedes usar:
# git-check-email -> Ver email actual
# git-set-franco -> Configurar email correcto
```

2. Template de Commit

Crear template para commits consistentes:

```
cat > ~/.gitmessage << 'EOF'
# <type>: <subject>
#
# <body>
#
# <footer>
#
# Types: feat, fix, docs, style, refactor, test, chore
# Subject: Max 50 chars, capitalized, no period
# Body: Wrap at 72 chars, explain WHAT and WHY
# Footer: Reference issues, breaking changes
EOF

git config --global commit.template ~/.gitmessage
```

3. Pre-push Hook

Verificar antes de push:

```

cat > /home/ubuntu/github_repos/quantpaychain-mvpro/.git/hooks/pre-push << 'EOF'
#!/bin/bash

echo "🔍 Verificando commits antes de push..."

# Verificar que no hay commits con emails problemáticos
BAD_EMAILS=$(git log @{}.. --format="%ae" | grep -E "(quantpaychain@example\.com|ai@quantpaychain\.com|agent@abacus\.ai|quantpay@quantpaychain\.org)")

if [ ! -z "$BAD_EMAILS" ]; then
    echo "❌ ERROR: Se detectaron commits con emails no verificados:"
    echo "$BAD_EMAILS"
    echo ""
    echo "Por favor corrige los commits antes de push."
    exit 1
fi

echo "✅ Todos los commits tienen email verificado"
exit 0
EOF

chmod +x /home/ubuntu/github_repos/quantpaychain-mvpro/.git/hooks/pre-push

```

4. Branch Protection en GitHub

Configurar en GitHub:

1. Settings → Branches → Add rule
 2. Branch name pattern: `main`
 3. Enable:
 - [x] Require pull request reviews before merging
 - [x] Require status checks to pass before merging
 - [x] Require conversation resolution before merging
 - [x] Include administrators
-

🎯 Decisión Recomendada

Para Acción Inmediata (HOY)

Recomendación: Usar **Opción A (Revert + Nuevo Commit)**

Razones:

- Más seguro
- No requiere force push
- Fácil de revertir si algo sale mal
- Mantiene historial de auditoría
- No afecta a otros colaboradores

Pasos:

1. Hacer backup (5 minutos)
2. Revert del commit problemático (2 minutos)
3. Cherry-pick con email correcto (5 minutos)
4. Verificar y push (2 minutos)
5. Probar despliegue en Vercel (10-15 minutos)

Tiempo total: ~30 minutos

Para Limpieza Completa (OPCIONAL - Esta semana)

Recomendación: Evaluar después de confirmar que el despliegue funciona

Si decides hacerlo:

1. Coordinar con cualquier colaborador
2. Hacer backup completo
3. Ejecutar git filter-repo en una copia
4. Verificar resultado exhaustivamente
5. Force push con -force-with-lease

Tiempo estimado: 2-3 horas (incluyendo verificaciones)



Comandos Rápidos de Referencia

Ver información de commit

```
# Ver último commit con detalles de autor
git log -1 --format="%H%n%an <%ae>%n%cn <%ce>%n%s"

# Ver todos los emails usados en el repositorio
git log --all --format="%ae" | sort | uniq -c | sort -rn

# Ver commits con email específico
git log --all --author="quantpaychain@example.com" --oneline
```

Correcciones rápidas

```
# Cambiar email del último commit (si aún no has hecho push)
git commit --amend --author="Franco Mengarelli <fmengarelli@gmail.com>" --no-edit

# Revert del último commit
git revert HEAD

# Reset suave del último commit (mantiene cambios)
git reset --soft HEAD~1
```

Backups

```
# Crear bundle backup
git bundle create ~/backup-$(date +%Y%m%d-%H%M%S).bundle --all

# Verificar bundle
git bundle verify ~/backup-*(bundle

# Restaurar desde bundle (si necesario)
git clone ~/backup-YYYYMMDD-HHMMSS.bundle restored-repo
```

SOS Troubleshooting

Problema: “Permission denied” al hacer push

```
# Verificar remote
git remote -v

# Si usa HTTPS, asegúrate de tener token de GitHub configurado
# O cambiar a SSH:
git remote set-url origin git@github.com:francoMengarelli/quantpaychain-mvpro.git
```

Problema: “Updates were rejected because the tip of your current branch is behind”

```
# Pull primero
git pull --rebase origin main

# Resolver conflictos si hay
# Luego continuar
git rebase --continue

# Y finalmente push
git push origin main
```

Problema: Vercel aún rechaza el despliegue después de correcciones

```
# Verificar que el HEAD tiene email correcto
git log -1 --format="%ae"

# Si es correcto, puede ser otro problema. Revisar:
# 1. Logs de Vercel
# 2. Variables de entorno
# 3. Configuración de build

# Contactar soporte de Vercel si persiste
```

Problema: Accidentalmente hiciste push con email incorrecto

```
# Si acabas de hacer push (menos de 5 minutos)
git reset --hard HEAD~1
git push --force-with-lease origin main

# Si ya pasó tiempo, usa el método de revert del plan principal
```

📞 Recursos y Enlaces

Documentación Oficial

- **Git Filter-Repo:** <https://github.com/newren/git-filter-repo>
- **Git Mailmap:** <https://git-scm.com/docs/gitmailmap>
- **GitHub Email Settings:** <https://github.com/settings/emails>

- **Vercel Deployment:** <https://vercel.com/docs/deployments/overview>

Documentación Interna del Proyecto

- `GIT_EMAIL_FIX.md` - Guía previa de corrección de emails
- `VERCEL_BUILD_FIX.md` - Guía de corrección de builds de Vercel
- `VERCEL_DEPLOYMENT_GUIDE.md` - Guía de despliegue en Vercel
- `ANALISIS_COMMITS.md` - Este análisis completo de commits

Contactos y Soporte

- **GitHub Issues:** <https://github.com/francoMengarelli/quantpaychain-mvpro/issues>
- **Vercel Support:** <https://vercel.com/support>



Registro de Cambios del Plan

Fecha	Cambio	Autor
2025-11-05	Plan inicial creado	Sistema de análisis



Próximos Pasos INMEDIATOS

1. **[] AHORA:** Leer y entender este documento completo
2. **[] AHORA:** Decidir entre Opción A (revert) u Opción B (amend)
3. **[] HOY:** Ejecutar corrección del commit HEAD
4. **[] HOY:** Verificar que el fix funcionó
5. **[] HOY:** Intentar despliegue en Vercel
6. **[] Esta semana:** Decidir si hacer limpieza completa del historial
7. **[] Esta semana:** Implementar hooks y protecciones

¡IMPORTANTE! Antes de ejecutar cualquier comando que modifique el historial:

1. Hacer backup completo
2. Verificar que tienes el backup
3. Entender qué hace cada comando
4. Tener plan de rollback

En caso de duda, siempre es mejor pedir ayuda que arriesgarse a perder trabajo.

Documento creado: 2025-11-05

Última actualización: 2025-11-05

Autor: Sistema de análisis y planificación QuantPay Chain

Estado: Listo para ejecución