🛡️ Protocolo Anti-Pendejadas: Guía de Seguridad para Desarrollo con Claude Code

📋 Resumen Ejecutivo

Este documento establece un protocolo de seguridad para trabajar con Claude Code en aplicaciones Azure, diseñado para prevenir errores críticos como la sobrescritura de requirements.txt con 200+ paquetes innecesarios.

🚨 Reglas de Oro

❌ Lo que NUNCA debe hacer Claude Code:

- Tocar `requirements.txt` (solo el desarrollador lo modifica)
- Modificar archivos `.env`
- Cambiar configuraciones principales (`settings.py`, `config.py`)
- Alterar archivos de base de datos
- Ejecutar `pip freeze > requirements.txt`

✅ Lo que SÍ puede hacer Claude Code:

- Modificar archivos de aplicación (`.py`, `.html`, `.css`, `.js`)
- Crear nuevos archivos de funcionalidades
- Sugerir instalación de paquetes (pero no instalarlos)

🔒 Sistema de Backup Obligatorio

Antes de CUALQUIER intervención de Claude Code:

bash

```
# Crear punto de restauración
git add .
git commit -m "Backup antes de Claude - [descripción del cambio solicitado]"
```

En caso de emergencia:

bash

```
# Regresar al estado anterior
git reset --hard HEAD~1
```

🛠️ Workflow de Seguridad

1. Preparación (Desarrollador)

- [ ] Crear backup: `git commit -am "Backup antes de Claude"`
- [ ] Crear rama de trabajo: `git checkout -b feature/[nombre-cambio]`
- [ ] Verificar rama actual: `git branch`

2. Intervención de Claude Code

- [ ] Claude Code realiza cambios
- [ ] Claude Code DEBE reportar todos los archivos modificados
- [ ] Claude Code DEBE explicar cada cambio realizado

3. Revisión Obligatoria (Desarrollador)

bash

```
# Ver archivos modificados
git status
git diff --name-only

# Revisar cambios específicos
git diff [nombre-archivo]
```

Checklist de Revisión:

- [ ] ¿Modificó archivos prohibidos? → **DETENER TODO**
- [ ] ¿Los cambios son solo los solicitados?
- [ ] ¿Entiendo todos los cambios realizados?
- [ ] ¿requirements.txt sigue teniendo solo los paquetes necesarios?

4. Commit Selectivo (Desarrollador)

bash

```
# NO hagas git add .
# Agrega archivos uno por uno:
git add archivo1.py
git add archivo2.html
git commit -m "Feature: [descripción específica]"
```

5. Testing Local

- [ ] Probar aplicación localmente
- [ ] Verificar todas las funcionalidades
- [ ] Confirmar que no hay errores en consola

6. Deploy Seguro

bash

```
# Solo después de testing exitoso:
git push origin feature/[nombre-cambio]

# Crear Pull Request o merge manual después de revisión final
```

🚫 Archivos Protegidos

Lista de archivos que Claude Code NO debe modificar:

```
requirements.txt
.env
.env.local
.env.production
settings.py
config.py
database.db
*.db
.gitignore (solo con permiso explícito)
Dockerfile (solo con permiso explícito)
```

📋 Protocolo para Nuevas Dependencias

Si Claude Code sugiere instalar un paquete:

1. **Claude Code SOLO sugiere**, NO instala
2. **Desarrollador decide** si instalar
3. **Desarrollador instala** manualmente: `pip install [paquete]`
4. **Desarrollador agrega** manualmente al requirements.txt:

[paquete]==1.2.3

5. **NUNCA usar** `pip freeze > requirements.txt`

🔄 Comandos de Emergencia

Si Claude Code la cagó:

bash

```
# Verificar daños
git status
git diff

# Restaurar archivo específico
git checkout HEAD -- [archivo-cagado]

# Restaurar todo al backup
git reset --hard HEAD~1

# Limpiar cambios no commiteados
git clean -fd
```

📊 Indicadores de Alerta

🚨 Detener inmediatamente si:

- requirements.txt cambia de 8 líneas a 200+

- Aparecen errores de importación nuevos
- La aplicación no arranca después de los cambios
- Claude Code modificó archivos no relacionados con la tarea

⚠️ Revisar cuidadosamente si:

- Se agregaron nuevos archivos de configuración
- Se modificaron imports en archivos principales
- Aparecen nuevas carpetas o estructuras

🎯 Responsabilidades

Claude Code:

- Realizar SOLO los cambios solicitados
- Reportar todos los archivos modificados
- Explicar la razón de cada cambio
- NUNCA tocar archivos protegidos sin permiso explícito

Desarrollador:

- Crear backups antes de cada sesión
- Revisar TODOS los cambios antes de commit
- Mantener control final sobre qué se sube al repositorio
- Tomar decisiones sobre instalación de dependencias

📝 Log de Incidentes

Registro de cagadas para aprender:

[Fecha] - [Descripción del problema] - [Causa] - [Solución aplicada]

Ejemplo:
2024-10-06 - requirements.txt con 200+ paquetes - pip freeze automático - git reset + requirements.txt manual

---

🚀 Comandos de Referencia Rápida

bash

```
# Workflow básico
git status                    # Ver estado actual
git diff --name-only          # Archivos modificados
git add archivo.py            # Agregar archivo específico
git commit -m "mensaje"       # Commit con mensaje
git checkout -b feature/nombre  # Nueva rama
git reset --hard HEAD~1       # Emergency reset

# Verificaciones de seguridad
git branch                    # ¿En qué rama estoy?
git remote -v                 # ¿A dónde voy a pushear?
git log -1                    # Último commit
wc -l requirements.txt        # Contar líneas en requirements
```

Versión: 1.0
Última actualización: Octubre 2024
Próxima revisión: Después del próximo incidente 😅