

# MÓDULO 6

## Configuración profesional

Sandbox · Approvals · Red · Perfiles · Reglas

*Plan de Formación — OpenAI Codex para Desarrolladores*

Nivel: Intermedio-Avanzado · Duración estimada: 3–4 horas

Versión 1.0 — Febrero 2026

## Índice

## 1. Introducción

Codex es un agente que ejecuta comandos shell, escribe ficheros y potencialmente accede a la red. Sin controles adecuados, un prompt mal formulado o un repositorio malicioso podrían provocar daños reales: borrar ficheros, filtrar secretos o modificar configuraciones de producción.

La seguridad de Codex se basa en dos capas que trabajan juntas: el sandbox (qué puede hacer técnicamente el agente) y la política de aprobaciones (cuándo debe detenerse y pedir permiso). Este módulo enseña a configurar ambas capas de forma profesional, adaptada al contexto del equipo.

## 2. Objetivos de aprendizaje

#	Objetivo	Evidencia de logro
O1	Configurar sandbox y política de aprobaciones para un proyecto.	Config.toml funcional con sandbox workspace-write y aprobaciones.
O2	Entender los tres modos de sandbox y cuándo usar cada uno.	Identifica correctamente qué modo aplicar en 3 escenarios dados.
O3	Crear perfiles de configuración por tipo de tarea (dev, review, CI).	3 perfiles definidos en config.toml y probados.
O4	Aplicar execution policy rules para controlar comandos específicos.	Regla que bloquea comandos peligrosos verificada con codex execpolicy check.

## 3. Contenidos teóricos

### 3.1 Las dos capas de seguridad de Codex

Codex protege tu sistema mediante dos mecanismos independientes que se combinan:

Capa	Qué controla	Configuración
Sandbox mode	Qué puede hacer Codex técnicamente: dónde puede escribir y si puede acceder a la red.	sandbox_mode en config.toml
Approval policy	Cuándo Codex debe detenerse y pedir permiso antes de actuar.	approval_policy en config.toml
Execution policy rules	Qué comandos específicos puede ejecutar fuera del sandbox.	Ficheros .rules en .codex/rules/

### 3.2 Modos de sandbox

Modo	Descripción	Caso de uso	Red
read-only (defecto)	Solo lectura del filesystem. No puede escribir ni ejecutar comandos que modifiquen estado.	Exploración de código, onboarding, análisis sin riesgo.	Desactivada
workspace-write	Escrutura limitada al workspace (CWD). .git/ y .codex/ permanecen solo lectura. Red desactivada por defecto.	Desarrollo activo: editar código, ejecutar tests, crear ficheros.	Configurable
danger-full-access	Sin sandbox. Acceso total al filesystem y red. Extremadamente arriesgado.	Solo en contenedores CI aislados o entornos controlados.	Activada

#### ⚠ Red desactivada por defecto

Localmente, Codex ejecuta con **red desactivada** por defecto en todos los modos de sandbox. Para habilitar acceso a red en workspace-write, configura `network_access = true` en `[sandbox_workspace_write]`. Esto es una decisión consciente de seguridad.

#### 3.2.1 Configuración de workspace-write

```
# En .codex/config.toml (proyecto) o ~/.codex/config.toml (global)
sandbox_mode = "workspace-write"

[sandbox_workspace_write]
network_access = false          # defecto: sin red
exclude_slash_tmp = false        # permitir /tmp como escribible
exclude_tmpdir_env_var = false   # permitir $TMPDIR
writable_roots = ["/ruta/extra"] # raíces adicionales escribibles
```

#### 💡 --full-auto es un alias

**El flag `--full-auto` es equivalente a `--sandbox workspace-write --ask-for-approval on-request`. Es el modo recomendado para desarrollo activo.**

### 3.3 Políticas de aprobación (approval\_policy)

Política	Comportamiento	Riesgo	Caso de uso
untrusted	Solo comandos read-only conocidos se auto-ejecutan. Todo lo demás requiere aprobación.	Mínimo	Repos no confiables, auditoría estricta.
on-failure	Auto-ejecuta en sandbox. Solo pide aprobación si falla y necesita escalar.	Bajo	Desarrollo día a día (defecto recomendado).
on-request	El modelo decide cuándo pedir permiso.	Medio	Desarrollo activo con confianza moderada.
never	Nunca pide aprobación. Ejecuta todo automáticamente.	Alto	Solo CI automatizado en contenedores aislados.

#### 3.3.1 Ejemplo: política estricta para datos sensibles

```
# .codex/config.toml para proyecto con datos sensibles
approval_policy = "untrusted"
sandbox_mode = "read-only"

# Si necesitas escritura puntual, usa workspace-write
# con red deshabilitada y aprobaciones untrusted:
# sandbox_mode = "workspace-write"
# [sandbox_workspace_write]
# network_access = false
```

### 3.4 Execution policy rules (experimental)

Las reglas de ejecución controlan qué comandos específicos puede ejecutar Codex fuera del sandbox. Se definen en ficheros .rules en .codex/rules/ (proyecto) o ~/codex/rules/ (usuario).

#### 3.4.1 Formato de regla

```
# .codex/rules/project.rules

# Permitir git add sin pedir aprobación
prefix_rule(
    pattern = ["git", "add"],
    decision = "allow",
    match = ["git add src/"],
    not_match = ["git add . && rm -rf /"],
)

# Bloquear permanentemente rm -rf en la raíz
prefix_rule(
    pattern = ["rm", "-rf", "/"],
    decision = "forbidden",
    justification = "Eliminación recursiva de la raíz prohibida.",
    match = ["rm -rf /"],
)
```

```
# Pedir aprobación para docker commands
prefix_rule(
    pattern = ["docker"],
    decision = "prompt",
    justification = "Docker modifica contenedores del sistema.",
    match = ["docker run nginx"],
)
```

### 3.4.2 Decisiones disponibles

Decisión	Comportamiento	Prioridad
allow	Ejecutar sin pedir aprobación.	Baja
prompt	Pedir aprobación al usuario antes de ejecutar.	Media
forbidden	Bloquear permanentemente. No se puede aprobar.	Alta (gana siempre)

**Regla de conflicto:** Si múltiples reglas coinciden, Codex aplica la decisión más restrictiva:  
forbidden > prompt > allow.

### 3.4.3 Verificar reglas con codex execpolicy check

```
# Verificar cómo se aplican las reglas a un comando
codex execpolicy check --pretty \
--rules .codex/rules/project.rules \
-- git add src/main.py

# Output: JSON con la decisión final y reglas que coinciden
```

## 3.5 Capas de configuración y precedencia

Codex resuelve la configuración en orden de precedencia (de mayor a menor):

1. Flags de CLI (--sandbox, --model, -c clave=valor).
2. Config de proyecto: .codex/config.toml (desde la raíz del repo Git hasta el CWD, el más cercano gana). Solo si el proyecto está trusted.
3. Config de usuario: ~/.codex/config.toml (defaults personales).
4. Requirements enterprise (requirements.toml): constraints que el usuario NO puede sobreescribir.
5. Defaults compilados en el CLI.

### 3.5.1 Trusted projects

Codex solo carga ficheros .codex/ del repositorio (config.toml, skills, rules) si el proyecto está marcado como trusted. Esto previene que repositorios maliciosos inyecten configuraciones peligrosas. El trust se gestiona con:

```
# En ~/.codex/config.toml:
[[project_trust]]
path = "/ruta/al/repo"
trust = "trusted" # o "untrusted"
```

## 3.6 Perfiles de configuración

Los perfiles permiten cambiar de contexto rápidamente. Se definen en config.toml bajo [profiles]:

```
# ~/.codex/config.toml

[profiles.safe-readonly]
sandbox_mode = "read-only"
approval_policy = "untrusted"
# Para exploración, onboarding, auditoría

[profiles.dev-edit]
model = "gpt-5.3-codex"
sandbox_mode = "workspace-write"
approval_policy = "on-request"
model_reasoning_effort = "medium"
# Desarrollo día a día (equivalente a --full-auto)

[profiles.ci-fixer]
model = "gpt-5.1-codex-mini"
sandbox_mode = "workspace-write"
approval_policy = "on-request"
model_reasoning_effort = "low"
# Tareas mecánicas de CI: lint fixes, formato

[profiles.deep-review]
model = "gpt-5.3-codex"
approval_policy = "untrusted"
sandbox_mode = "read-only"
model_reasoning_effort = "high"
```

```
# Code review profundo sin riesgo de modificación
```

Usar un perfil:

```
# Desde CLI:  
codex --profile safe-readonly  
  
# O establecer perfil por defecto:  
# En config.toml (raíz):  
profile = "dev-edit"
```

## 3.7 Red y web search

Setting	Descripción	Valor por defecto
network_access	Acceso a red desde el sandbox. Solo aplica en workspace-write.	false
web_search = "disabled"	Sin búsqueda web.	(no es el default)
web_search = "cached"	Resultados de un índice pre-cacheado por OpenAI. Menor riesgo de prompt injection.	Default local
web_search = "live"	Búsqueda web en tiempo real.	Default con --yolo

**[i] Web search y seguridad**  
Los resultados de web search (incluso cached) deben tratarse como **no confiables**. Pueden contener prompt injection. En entornos con datos sensibles, desactivar con `web_search = "disabled"`.

## 3.8 Monitorización y auditoría (OTel)

Codex emite eventos estructurados via OpenTelemetry para auditar qué hizo el agente:

- `codex.conversation_starts` — modelo, sandbox/approval policy seleccionados.
- `codex.tool_decision` — comando aprobado/denegado, si la decisión fue de config o del usuario.
- `codex.tool_result` — duración, éxito/fallo, snippet de output.
- `codex.user_prompt` — longitud (contenido redactado salvo opt-in explícito).

```
# Habilitar OTel en config.toml
[otel]
environment = "prod"
exporter = "otlp-http"
log_user_prompt = false # redactar prompts por defecto

[otel.exporter_otlp_http]
endpoint = "https://collector.empresacom/v1/traces"
```

## 3.9 Enterprise: requirements.toml

En entornos enterprise, los administradores pueden definir constraints que los usuarios NO pueden sobreescribir:

```
# requirements.toml (distribuido por MDM)

# Políticas de aprobación permitidas
allowed_approval_policies = ["untrusted", "on-request"]
# Prohibe "never" (sin aprobaciones)

# Modos de sandbox permitidos
allowed_sandbox_modes = ["read-only", "workspace-write"]
# Prohibe "danger-full-access"

# Web search permitido
allowed_web_search = ["disabled", "cached"]
# Prohibe "live"
```

## 4. Buenas prácticas

### 4.1 Principio de mínimo privilegio

- Empezar siempre en read-only. Escalar a workspace-write solo cuando necesites editar.
- Red desactivada por defecto. Activar solo si el workflow lo requiere (ej. npm install).
- approval\_policy = "untrusted" para repos que no controlas.

### 4.2 Perfiles por tipo de tarea

- safe-readonly: exploración, onboarding, auditoría.
- dev-edit: desarrollo activo (workspace-write + on-request).
- ci-fixer: tareas mecánicas con modelo económico.
- deep-review: review profundo en read-only con reasoning alto.

### 4.3 Execution policy rules para equipos

- Definir reglas en .codex/rules/ del repo (versionadas con Git).
- Bloquear (forbidden) comandos destructivos: rm -rf /, DROP DATABASE, etc.
- Pedir aprobación (prompt) para Docker, cloud CLIs, Git push.
- Verificar reglas con codex execpolicy check antes de mergear.

### 4.4 Auditoría y post-mortem

- Habilitar OTel para registrar todas las acciones del agente.
- Revisar periódicamente eventos tool\_decision para detectar patrones inesperados.
- Las transcripciones se guardan en ~/.codex/sessions/ para análisis posterior.

## 5. Errores comunes y cómo evitarlos

### 5.1 Activar red sin auditoría

**Problema:** Habilitar network\_access = true sin monitorización. Codex podría hacer requests a URLs inesperadas o filtrar datos.

**Solución:** Si se necesita red, habilitar OTel y revisar eventos codex.api\_request y codex.tool\_result. En entornos sensibles, mantener red desactivada.

### 5.2 Usar danger-full-access en desarrollo local

**Problema:** Elimina todas las protecciones. Un prompt malicioso podría modificar ficheros del sistema, acceder a credenciales, etc.

**Solución:** Reservar danger-full-access exclusivamente para contenedores CI aislados. En local siempre usar workspace-write como máximo.

### 5.3 No configurar proyecto como trusted/untrusted

**Problema:** Si no configuras trust explícitamente, Codex puede pedir confirmación repetidamente o cargar .codex/ de un repo malicioso sin revisión.

**Solución:** Marcar explícitamente los repos en [[project\_trust]] de ~/.codex/config.toml.

### 5.4 Tabla de errores adicionales

Error	Consecuencia	Prevención
approval_policy = "never" en local	Codex ejecuta todo sin confirmación, incluidos comandos destructivos	Solo en CI aislado. En local mínimo on-request.
No versionar .codex/config.toml	Cada desarrollador configura distinto	Incluir .codex/config.toml en Git
No usar execution policy rules	Codex puede ejecutar cualquier comando dentro del sandbox	Definir reglas para comandos críticos
Ignorar transcripciones	Imposible hacer post-mortem de errores	Revisar ~/.codex/sessions/ periódicamente

## 6. Casos de uso reales

### 6.1 Equipo regulado (finanzas/salud)

**Escenario:** Equipo maneja datos sensibles de pacientes. Requiere máxima restricción y auditoría completa.

#### Configuración aplicada

```
# .codex/config.toml del repo
approval_policy = "untrusted"
sandbox_mode = "workspace-write"
web_search = "disabled"

[sandbox_workspace_write]
network_access = false

[otel]
environment = "prod"
exporter = "otlp-http"
log_user_prompt = false
```

```
# .codex/rules/security.rules
prefix_rule(
    pattern = ["curl"],
    decision = "forbidden",
    justification = "Red prohibida. No se permiten requests HTTP.",
    match = ["curl https://example.com"],
)
prefix_rule(
    pattern = ["pip", "install"],
    decision = "prompt",
    justification = "Instalación de paquetes requiere aprobación.",
    match = ["pip install requests"],
)
```

**Resultado:** Codex no puede acceder a la red, web search está desactivado, curl está bloqueado y pip install requiere aprobación. Toda actividad queda registrada en OTel.

### 6.2 Entorno CI automatizado

**Escenario:** GitHub Action que ejecuta Codex para auto-fix de tests rotos.

```
# En el workflow de GitHub Actions:
codex exec --full-auto --sandbox workspace-write \
    "Read the test failures, identify the minimal fix, \
     implement it, and run tests to verify."

# --full-auto = workspace-write + on-request
# codex exec en modo no interactivo
```

## 7. Laboratorio práctico (L6) — Política de aprobaciones y sandbox

### ⌚ Objetivo del laboratorio

Configurar un proyecto con límites estrictos (sandbox, approvals, execution rules) y **verificar que Codex los respeta** al pedirle tareas que requieren acciones sensibles.

### 7.1 Prerrequisitos

- Codex CLI instalado y autenticado.
- Python 3.10+ con venv.
- Git instalado.

### 7.2 Paso 1 — Crear proyecto con configuración segura

*Tiempo estimado: 5 minutos*

```
mkdir /tmp/codex-lab06 && cd /tmp/codex-lab06
git init

# Entorno virtual
python3 -m venv .venv
source .venv/bin/activate
pip install pytest flake8

cat > .gitignore << 'EOF'
.venv/
__pycache__/
*.pyc
EOF

# Código base
mkdir -p src tests

cat > src/app.py << 'PYEOF'
import os

def get_config():
    return {"debug": os.getenv("DEBUG", "false")}

def process(data):
    return [x * 2 for x in data if x > 0]
PYEOF

cat > tests/test_app.py << 'PYEOF'
from src.app import get_config, process

def test_config():
    config = get_config()
    assert "debug" in config
```

```

def test_process():
    assert process([1, -2, 3]) == [2, 6]
PYEOF

PYTHONPATH=. pytest tests/ -v
git add -A && git commit -m "chore: scaffold lab06"

```

## 7.3 Paso 2 — Configurar sandbox y aprobaciones

*Tiempo estimado: 5 minutos*

```

# Crear configuración de proyecto restrictiva
mkdir -p .codex

cat > .codex/config.toml << 'EOF'
# Política estricta para el laboratorio
approval_policy = "untrusted"
sandbox_mode = "workspace-write"
web_search = "disabled"

[sandbox_workspace_write]
network_access = false
EOF

git add .codex/ && git commit -m "config: política restrictiva"

```

## 7.4 Paso 3 — Crear execution policy rules

*Tiempo estimado: 5 minutos*

```

mkdir -p .codex/rules

cat > .codex/rules/lab.rules << 'EOF'
# Bloquear curl (sin acceso a red)
prefix_rule(
    pattern = ["curl"],
    decision = "forbidden",
    justification = "Red deshabilitada en este proyecto.",
    match = ["curl https://example.com"],
)

# Pedir aprobación para pip install
prefix_rule(
    pattern = ["pip", "install"],
    decision = "prompt",
    justification = "Instalar paquetes requiere aprobación.",
    match = ["pip install requests"],
)

# Permitir pytest sin aprobación
prefix_rule(

```

```

        pattern = ["pytest"],
        decision = "allow",
        match = ["pytest tests/ -v"],
)
EOF

git add .codex/rules/
git commit -m "rules: política de ejecución"

```

## Verificar las reglas (sin lanzar Codex)

```

# Verificar que curl está bloqueado
codex execpolicy check --pretty \
    --rules .codex/rules/lab.rules \
    -- curl https://example.com
# Esperado: decision = "forbidden"

# Verificar que pip install pide aprobación
codex execpolicy check --pretty \
    --rules .codex/rules/lab.rules \
    -- pip install requests
# Esperado: decision = "prompt"

# Verificar que pytest está permitido
codex execpolicy check --pretty \
    --rules .codex/rules/lab.rules \
    -- pytest tests/ -v
# Esperado: decision = "allow"

```

## 7.5 Paso 4 — Probar que Codex respeta los límites

*Tiempo estimado: 10 minutos*

```

cd /tmp/codex-lab06
source .venv/bin/activate
codex

```

### TEST 1: Pedir acción bloqueada (curl):

```

# Prompt en el TUI:
"Descarga el fichero https://example.com/data.json
usando curl y guárdalo en src/data.json."

# Observación esperada: Codex NO puede ejecutar curl
# (bloqueado por la regla forbidden)

```

### TEST 2: Pedir acción que requiere aprobación (pip install):

```

# Prompt en el TUI:
"Añade la dependencia 'requests' al proyecto
e importala en src/app.py."

```

```
# Observación esperada: Codex solicita aprobación
# antes de ejecutar pip install (política untrusted
# + regla prompt)
```

### TEST 3: Pedir acción permitida (tests):

```
# Prompt en el TUI:
"Añade un test para process() con una lista vacía
y ejecuta pytest." 

# Observación esperada: Codex puede ejecutar pytest
# sin pedir aprobación (regla allow)
```

## 7.6 Paso 5 — Crear perfiles y probar

*Tiempo estimado: 5 minutos*

```
# Añadir perfiles al config del proyecto
cat >> .codex/config.toml << 'EOF'

[profiles.review-only]
sandbox_mode = "read-only"
approval_policy = "untrusted"

[profiles.dev]
sandbox_mode = "workspace-write"
approval_policy = "on-request"
EOF

# Probar perfil review-only:
codex --profile review-only
# En este modo, Codex NO puede escribir ficheros.
# Prompt: "Explica cómo funciona src/app.py"
# → Funciona (solo lectura)
# Prompt: "Añade un test nuevo"
# → Falla (no puede escribir en read-only)
```

## 7.7 Resultado esperado

Verificación	Resultado esperado
curl bloqueado	Codex no puede ejecutar curl (forbidden)
pip install con aprobación	Codex pide permiso antes de instalar
pytest permitido	Codex ejecuta tests sin pedir aprobación
Perfil review-only	Codex puede leer pero no escribir
Perfil dev	Codex puede leer y escribir con aprobaciones
Web search desactivado	Codex no busca en la web

## 7.8 Limpieza (OBLIGATORIA)

### ⚠️ Limpieza de recursos

Ejecutar al finalizar:

```
deactivate
rm -rf /tmp/codex-lab06

ls /tmp/codex-lab06 2>/dev/null \
&& echo 'ERROR' || echo 'OK: limpio'
```

## 8. Resumen y siguientes pasos

### 8.1 Conceptos clave

Concepto	Detalle
Sandbox mode	read-only (defecto)   workspace-write   danger-full-access
Approval policy	untrusted   on-failure   on-request (defecto)   never
Red desactivada	network_access = false por defecto. Decisión consciente.
Execution policy rules	prefix_rule() con allow/prompt/forbidden en .codex/rules/
codex execpolicy check	Verificar reglas antes de usarlas
Trusted projects	Codex solo carga .codex/ de repos marcados como trusted
Perfiles	[profiles.nombre] en config.toml para cambiar contexto
OTel	Eventos estructurados para auditoría
requirements.toml	Constraints enterprise no sobreescribibles
--full-auto	Alias de workspace-write + on-request

### 8.2 Preparación para el Módulo 7

En el Módulo 7 abordaremos Codex Cloud y GitHub Action: cómo delegar tareas a entornos remotos y automatizar workflows. Para prepararse:

- Revisar: <https://developers.openai.com/codex/cloud/>
- Revisar: <https://developers.openai.com/codex/noninteractive/>
- Revisar: <https://developers.openai.com/codex/github-action/>

## 9. Referencias y recursos

- Security: <https://developers.openai.com/codex/security/>
- Config basics: <https://developers.openai.com/codex/config-basic/>
- Config advanced: <https://developers.openai.com/codex/config-advanced/>
- Config reference: <https://developers.openai.com/codex/config-reference/>
- Sample config: <https://developers.openai.com/codex/config-sample/>
- Execution policy rules: <https://developers.openai.com/codex/exec-policy/>
- Non-interactive mode: <https://developers.openai.com/codex/noninteractive/>
- Automations: <https://developers.openai.com/codex/app/automations/>