Este el el contenido de mi archivo **~/.claude/CLAUDE.md** (también puede ser el contenido de **~/.codex/AGENTS.md** o **~/.config/opencode/AGENTS.md**):

````
# Código
Para todo mi código usa las siguientes reglas que son OBLIGATORIAS y tienen prioridad sobre cualquier otra
instrucción:

## Indentación
- Usa siempre una indentación de 2 espacios.
- Debes reemplazar cualquier tabulación por exactamente 2 espacios.

## Estructura
- No simplifiques la lógica al traducir código entre lenguajes.
- No simplifiques la lógica al modificar código escrito previamente por mí.
- No reinterpretes ni “optimices” mi forma de programar.
- Respecta estrictamente mi estructura original.

## Convenciones de nombres
- Variables: Las variables se deben nombrar con estilo CamelCase precedido de `v` minúscula (ej: `vContador`).
- Constantes: Las constantes se deben nombrar con estilo CamelCase precedido de `c` minúscula (ej: `cMaximo`).
- Arrays/Listas: Las listas o arrays se deben nombrar con estilo CamelCase precedido de `a` minúscula (ej:
`aUsuarios`).
- Diccionarios: Los diccionarios se deben nombrar con estilo CamelCase precedido de `d` minúscula (ej:
`dLibrosSciFi`).
- Métodos: Los métodos se deben nombrar con estilo CamelCase precedido de `m` minúscula (ej: `mEspecial`).
- Lista de diccionarios: Las listas de diccionarios se deben nombrar con estilo CamelCase precedido de `ld`
minúscula (ej: `ldEstoEsUnaListaDeDiccionarios`).
- Funciones: Las funciones se deben nombrar con estilo CamelCase precedido de `f` minúscula (ej: `fProcesarDatos`).
- Parámetros: Cuando se define una función, el nombre de los parámetros debe empezar por 'p' (ej: `pCiudad`)

#### Ejemplo

```python
def fCalcularTotal(pCantItems, pPrecioUnitario):
  cImpuesto = 0.21
  vSubtotal = pCantItems * pPrecioUnitario
  vTotal = vSubtotal * (1 + cImpuesto)
  return vTotal
aResultados = [1, 2, 3]
dConfiguracion = {"clave": "valor"}
```

## Bash
- Shebang obligatorio y exacto: `#!/bin/bash`
- Siempre deja una línea en blanco después del shebang.
- No usar `awk`, a menos que sea estrictamente necesario. Siempre dale preferencia al uso de `sed`.
- Prohibido usar EOF o here-documents en cualquier contexto.
- Los scripts deben ser reproducibles y no interactivos cuando sea posible.
- Siempre prefiere `< /dev/tty` antes que stdin , así el script nunca avanza sin la interacción real del usuario.
### Control de errores en bash
 - Debes usar siempre set -euo pipefail
 - Debes definir una función fCleanup y registrar trap fCleanup EXIT
 - Debes encapsular la lógica principal en una función fMain
 - Debes ejecutar fMain con if ! fMain; then ... fi
 - No debes usar bloques { ... } || { ... } como mecanismo principal de control de errores
#### Ejemplo de plantilla
```bash
#!/bin/bash

set -euo pipefail

fCleanup() {
  # limpieza
  :
}

trap fCleanup EXIT
````
````

```
fMain() {
  # lógica principal
  :
}

if ! fMain; then
  echo "error"
fi
```

Python
 - Pon un shebang siempre que el script se pueda ejecutar directamente.
 - El código del shebang debe ser siempre `#!/usr/bin/env -S PYTHONDONTWRITEBYTECODE=1 python3`.
 - Dejar una línea en blanco después del shebang.
 - Si un script escucha en un puerto, cerrar previamente cualquier socket abierto en ese mismo puerto.
 - Instalar paquetes con: `python3 -m pip install NombreDelPaquete --break-system-packages`.
Control de errores en Python
 - Usa try/except/finally siempre que se pueda usar.
Namespace packages implícitos
 - Crea código únicamente compatible con python 3.13 o superior, porque Python 3.13 soporta namespace packages implícitos. Eso lo quiero así porque no me gusta que entre los archivos del proyecto hayan archivos `__init__.py`

PHP
Contro de errores en PHP
- Usa try/catch/finally siempre que se pueda usar.

Apache
Cuando crees webs con Apache, usa estos templates de `.conf`:

VirtualHost HTTP (puerto 80)

```apache
<VirtualHost *:80>
  Redirect permanent / https://dominio.com/
  ServerName dominio.com
  ServerAlias www.dominio.com
  DocumentRoot /var/www/dominio.com
  <Directory "/var/www/dominio.com">
    Require all granted
    Options FollowSymLinks
    AllowOverride All
  </Directory>
  ServerAdmin admin@dominio.com
  ErrorLog  /var/www/dominio.com-logs/error.log
  CustomLog /var/www/dominio.com-logs/access.log combined
  <Directory "/var/www/dominio.com/__">
    AuthType Basic
    AuthName "Restricted Content"
    AuthUserFile /etc/apache2/.htpasswd
    Require valid-user
  </Directory>
  RewriteEngine on
  RewriteCond %{SERVER_NAME} =www.dominio.com [OR]
  RewriteCond %{SERVER_NAME} =dominio.com
  RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>
```

VirtualHost HTTPS (puerto 443)

```apache
<IfModule mod_ssl.c>
  <VirtualHost *:443>
    #Redirect permanent / http://dominio.com/
    RemoteIPProxyProtocol On
    ServerName dominio.com
    ServerAlias www.dominio.com
    DocumentRoot /var/www/dominio.com
```

```
    <Directory "/var/www/dominio.com">
      Require all granted
      Options FollowSymLinks
      AllowOverride All
    </Directory>
    ServerAdmin admin@dominio.com
    ErrorLog  /var/www/dominio.com-logs/error.log
    CustomLog /var/www/dominio.com-logs/access.log combined
    Include               /etc/letsencrypt/options-ssl-apache.conf
    SSLCertificateFile    /etc/letsencrypt/live/dominio.com/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/dominio.com/privkey.pem
  </VirtualHost>
</IfModule>
```

Puertos personalizados
Si necesitas escuchar en puertos distintos al 80 y al 443, aplica los siguientes cambios:

En el bloque HTTP, sustituye:

```apache
<VirtualHost *:80>
```
por:

```apache
Listen 11080
<VirtualHost *:11080>
```

En el bloque HTTPS, sustituye:

```apache
<IfModule mod_ssl.c>
  <VirtualHost *:443>
```

por:
```apache
<IfModule mod_ssl.c>
  Listen 11443
  <VirtualHost *:11443>
```
Archivos .md en la raíz de cada proyecto

Cada proyecto deberá tener los siguientes archivos .md en su correspondiente carpeta raíz:

 - README.md: con las explicaciones típicas que suelen estar en los README (para que la gente menos técnica entienda que funcionalidad tiene el proyecto, como se instala en Debian, como se ejecuta una vez instalado y como se interactúa con él).
 - CODE.md: con la explicación técnica detallada de todas las partes del código, para que la gente más técnica o los modelos de lenguaje puedan consultarla y entender el codebase sin tener que leer todo el código, a menos que sea estrictamente necesario).
 - MANUAL.md: será el manual de uso de la aplicación. Servirá al usuario para que entienda como se usa la aplicación, como se interactúa con las diferentes funcionalidades de la misma y toda la información que creas que será relevante para que una persona que no conoce como se opera con la app, pueda aprender a utilizarla y ponerse en marcha inmediatamente.

Cada vez que se hagan cambios en el código deben actualizarse esos tres archivos para reflejar lo modificado. Los tres archivos, tanto README-md como CLAUDE.md y MANUAL.md deben estar SIEMPRE actualizados.

Tildes en las palabras en español

Siempre que comentes código en español, pongas cadenas de texto que serán visibles en la app o popules los archivos README.md, CODE.md y MANUAL.md con palabras en español, asegúrate de que esas palabras lleven tilde donde deben llevarla.