Template C.O.R.E para Prompts Eficientes

1. Estructura Básica

C: [Stack/Framework + Archivos disponibles]
O: [Resultado específico esperado]
R: [Límites/Exclusiones]
E: [Formato de salida] (opcional)

Ejemplo Compacto

C: React 18 + TS, #components/UserList.tsx abierto
O: Hook useDebounce<T>(value, delay) genérico
R: Sin deps externas, <200 líneas
E: Export default, JSDoc incluido

---

2. Guía de Compresión

Tabla de Referencia Rápida

Elemento	Formato Largo ❌	Formato Compacto ✅
Contexto	"Estoy trabajando en Angular 17 con TypeScript"	C: Angular 17 + TS
Objetivo	"Necesito crear un servicio que obtenga usuarios"	O: UserService GET /users
Restricción	"No uses librerías externas excepto RxJS"	R: Solo RxJS
Ejemplo	"Debe retornar un Observable con array de usuarios"	E: Observable<User[]>

Abreviaturas Comunes

Frontend:

- React → R | Angular → Ng | Vue → V
- TypeScript → TS | JavaScript → JS
- Component → Cmp | Service → Svc | Hook → Hk

Backend:

- CRUD → Create/Read/Update/Delete
- API → Endpoint REST

- Auth → Autenticación
- Repo → Repository
- DTO → Data Transfer Object
- DI → Dependency Injection

General:

- DB → Database | ORM → Object-Relational Mapping
- Async → Asíncrono | Sync → Síncrono
- Req → Request | Res → Response
- Err → Error | Val → Validation

---

3. Plantillas por Escenario

Backend - API Endpoint

C: .NET 8 WebAPI + EF Core
O: ProductsController CRUD
R: Auth JWT, ValidationAttribute, sin ServiceLayer
E: IActionResult, ApiResponse<T>

Frontend - Componente

C: Next.js 14 App Router + Shadcn
O: ProductCard cmp: img, title, price, addToCart btn
R: Server Component, sin useState
E: Tailwind, TypeScript strict

Base de Datos

C: PostgreSQL 15
O: Schema ecommerce: users, products, orders (1:N)
R: UUID pk, timestamps auto, indexes en FKs
E: SQL migration script

Testing

C: Jest + React Testing Library

O: Tests UserForm: render, validation, submit
R: Mock fetch, AAA pattern
E: Coverage >80%

---

4. Técnicas Avanzadas de Compresión

Usar Símbolos

→ (implica/retorna)
| (separador)
+ (y/además)
? (opcional)
! (requerido)

Ejemplo:

O: validate(email: string) → boolean | null
  regex! + trim → return true/false/null

Referenciar Archivos

❌ "Aquí está el código de UserService: [200 líneas]"
✅ "C: #UserService.cs abierto"

Pilas de Decisiones

R: CSS → Tailwind > inline styles
  State → Zustand > Context > prop drilling
  Fetch → axios SI, else fetch nativo

---

5. Checklist de Optimización

Antes de enviar tu prompt, verifica:

- [ ] ¿Usé abreviaturas estándar?
- [ ] ¿Eliminé palabras como "por favor", "necesito que"?
- [ ] ¿Referencié archivos en vez de copiar código?
- [ ] ¿Separé en múltiples prompts si >1 tarea?
- [ ] ¿El objetivo es específico y medible?

- [ ] ¿Las restricciones son claras y necesarias?

---

6. Comparación: Antes vs Después

❌ Prompt Inflado (180 tokens)

Hola, necesito tu ayuda para crear un componente en React. Estoy trabajando con React 18 y TypeScript. El componente debe ser un formulario de login que tenga dos campos: email y password. Debe validar que el email sea válido y que la contraseña tenga al menos 8 caracteres. Cuando el usuario haga submit, debe llamar a una API y mostrar un mensaje de error si falla. Por favor usa React Hook Form y Zod para las validaciones. El diseño debe ser con Tailwind CSS.

✅ Prompt Optimizado (32 tokens)

C: React 18 + TS + Tailwind
O: LoginForm: email!, password(min8)
R: React Hook Form + Zod validation
E: POST /auth/login → show error toast on fail

Ahorro: 82% menos tokens

---

7. Template en Blanco para Copiar

C:
O:
R:
E:

---

8. Ejemplos Reales

Ejemplo 1: Migración de Código

C: Legacy jQuery, target: React 18
O: Refactor #dashboard.js → TSX components

R: Sin class components, Hooks only
E: Atomic design, #src/components/

Ejemplo 2: Bug Fix

C: Node.js Express API + MongoDB
O: Fix: /users endpoint 500 error on empty query
R: No cambiar schema, mantener middleware chain
E: Try-catch + logger, return 200 + []

Ejemplo 3: Feature Nueva

C: Flutter 3.16 + Riverpod
O: DarkModeToggle widget → persist SharedPrefs
R: Sin paquetes externos extras, AnimatedTheme
E: Switch Material, callback onChanged

---

9. Anti-Patrones a Evitar

❌ No Hacer	✅ Hacer
"Haz lo mejor posible"	Especifica métricas: "Response <200ms"
"Bonito y moderno"	"Tailwind, glassmorphism, dark mode"
Copiar todo el código	Referenciar: `#file.ts líneas 20-45`
Múltiples tareas	1 prompt = 1 responsabilidad
Contexto redundante	Solo info que cambia el resultado

---

10. Tips Pro

1. **Versiona tus prompts:** Guarda variaciones que funcionaron
2. **Mide tokens:** Usa herramientas como `tiktoken` (OpenAI)
3. **Itera:** Refina el prompt si la respuesta no es óptima
4. **Contexto mínimo viable:** Incluye solo lo esencial
5. **Sé específico en formatos:** JSON schema, tipos TS, etc.

Nota: Este template se optimiza para LLMs técnicos (GPT-4, Claude, etc.) que entienden jerga de desarrollo.