

Meta-Prompt: Generador de Prompts Optimizados

Importante: Formatos de Prompt

El meta-prompt puede generar prompts en **dos formatos**:

1. Formato C.O.R.E Estructurado

C: [contexto]
O: [objetivo]
R: [restricciones]
E: [ejemplo/salida]

Ventajas: Muy claro, fácil de escanear, ideal para prompts complejos

2. Formato Natural Compacto

[Acción] [Stack]: [Detalles clave]. [Restricciones importantes]

Ventajas: Más fluido para GitHub Copilot, menos formal, igualmente efectivo

Recomendación:

- Usa **C.O.R.E** para tareas complejas (>50 tokens, múltiples restricciones)
- Usa **Natural** para tareas simples y medianas (<50 tokens)
- GitHub Copilot funciona bien con ambos

Plantilla para Copiar y Pegar

Genera un prompt optimizado para GitHub Copilot sobre: [TU TEMA]

Requisitos:

- Máximo 50 tokens
- Formato: [C.O.R.E estructurado / Natural compacto]
- Incluye: detalles técnicos específicos (EF Core, migrations, etc.)
- Sin palabras de relleno

Target: [GitHub Copilot / ChatGPT / Claude]

Genera ambas versiones: C.O.R.E y Natural

Ejemplo de Uso

❌ Pregunta Normal (ineficiente)

Ayúdame a crear un prompt para dockerizar un backend en C#

✅ Usando el Meta-Prompt

Genera un prompt optimizado para GitHub Copilot sobre:
dockerizar backend C# .NET 8 Web API con SQL Server y EF Core

Requisitos:

- Máximo 50 tokens
- Formato: ambas versiones (C.O.R.E y Natural)
- Incluye: EF migrations, connection strings, health checks, volumes
- Target: GitHub Copilot

Genera ambas versiones

🎯 Resultado Esperado

C: .NET 8 WebAPI + EF Core + PostgreSQL
O: Dockerfile multi-stage + docker-compose.yml
R: Alpine base, HTTPS dev certs, health check endpoint
E: Build <2min, img <150MB, expose :5000+:5001

Variaciones del Meta-Prompt

Para Tareas Complejas

Genera un prompt optimizado formato C.O.R.E para: [TEMA]

Contexto adicional:

- Stack actual: [tu stack]
- Archivos relevantes: [lista]
- Pain point: [problema específico]

Requisitos:

- Máximo 70 tokens (tarea compleja)

- Incluir restricciones técnicas específicas
- Formato GitHub Copilot

Output esperado: Prompt listo para copiar/pegar

Para Refactoring

Genera prompt C.O.R.E optimizado para refactorizar: [COMPONENTE/FUNCIÓN]

De: [estado actual]

A: [estado deseado]

Máximo 40 tokens | Target: GitHub Copilot

Para Debugging

Genera prompt C.O.R.E para debug: [ERROR/PROBLEMA]

Síntomas: [descripción breve]

Stack: [tecnologías]

Máximo 30 tokens | Enfoque: root cause + fix

Ejemplos Reales con Resultados

Ejemplo 1: Dockerizar Backend C#

Input al Meta-Prompt:

Genera un prompt optimizado formato C.O.R.E para GitHub Copilot sobre:
dockerizar backend C# .NET 8 Web API con SQL Server y EF Core

Requisitos:

- Máximo 50 tokens
- Target: GitHub Copilot

Output Generado (Formato C.O.R.E - 35 tokens):

C: .NET 8 WebAPI + EF Core + SQL Server
O: Dockerfile multi-stage + docker-compose
R: EF migrations startup, env vars, health checks, volumes
E: Production-ready, networking, restore/build/publish optimizado

Output Alternativo (Formato Natural - 50 tokens):

Crea Dockerfile multi-stage .NET 8 Web API + docker-compose con SQL Server. Incluye: EF Core migrations al iniciar, connection string configurable, health checks, networking, volúmenes persistentes. Production-ready con restore/build/publish optimizados.

Output Ultra-Conciso (45 tokens):

Dockerfile .NET 8 API + docker-compose SQL Server: EF migrations startup, env connection string, health checks, volumes persistentes, multi-stage build optimizado para producción con networking configurado.

Ejemplo 2: Test Unitario

Input al Meta-Prompt:

Genera un prompt optimizado formato C.O.R.E para:
crear tests unitarios para servicio de autenticación JWT

Requisitos:

- Máximo 40 tokens
- Target: GitHub Copilot

Output Generado:

C: xUnit + Moq, #AuthService.cs
O: Tests: GenerateToken, ValidateToken, RefreshToken
R: AAA pattern, mock IUserRepository
E: Coverage >90%, async tests

Ejemplo 3: Componente React

Input al Meta-Prompt:

Genera un prompt optimizado formato C.O.R.E para:
crear formulario de registro con validación

- Requisitos:
- Máximo 45 tokens
 - Stack: React 18 + TypeScript
 - Target: GitHub Copilot

Output Generado:

C: React 18 + TS + Tailwind
O: RegisterForm: email, pwd, confirmPwd, terms checkbox
R: Zod validation, React Hook Form, no external UI libs
E: POST /auth/register, loading state, error toast

Template Ultra-Rápido (Una Línea)

Para cuando tienes prisa:

Prompt C.O.R.E <50 tokens para Copilot: [TU TAREA]

Ejemplo:

Prompt C.O.R.E <50 tokens para Copilot: API REST CRUD productos Node.js + MongoDB

Resultado:

C: Node 20 + Express + Mongoose
O: ProductsRouter CRUD: GET/POST/PUT/DELETE /api/products
R: async/await, error middleware, Joi validation
E: JSON responses, status codes REST standard

Casos de Uso por Tecnología

Backend

Tarea	Meta-Prompt Rápido
API REST	Prompt C.O.R.E <50 tokens: API [recurso] [framework]

Tarea	Meta-Prompt Rápido
Middleware	Prompt C.O.R.E <40 tokens: Middleware [función] [stack]
ORM Query	Prompt C.O.R.E <35 tokens: Query [entidad] [operación] [ORM]
Auth	Prompt C.O.R.E <45 tokens: Auth [método] [tecnología]



Frontend

Tarea	Meta-Prompt Rápido
Componente	Prompt C.O.R.E <45 tokens: Component [nombre] [framework]
Hook	Prompt C.O.R.E <35 tokens: Hook [funcionalidad] React
Form	Prompt C.O.R.E <50 tokens: Form [campos] [validación]
State	Prompt C.O.R.E <40 tokens: State mgmt [store] [framework]

DevOps

Tarea	Meta-Prompt Rápido
Docker	Prompt C.O.R.E <50 tokens: Dockerfile [app] [base img]
CI/CD	Prompt C.O.R.E <55 tokens: Pipeline [plataforma] [pasos]
Config	Prompt C.O.R.E <40 tokens: Config [servicio] [ambiente]

Tips para Mejores Resultados

1. Sé específico con dependencias críticas:
-  "con base de datos"
 -  "con SQL Server + EF Core"
 - Incluye: ORMs, connection pools, migrations, seeds
2. Menciona configuración de infraestructura:
- Connection strings configurables (env vars)
 - Health checks y endpoints de monitoreo
 - Volúmenes persistentes para datos
 - Networking entre servicios
3. Ajusta el límite de tokens según complejidad:
- Simple (CRUD básico): 30-40 tokens

- Media (API con DB y auth): 45-55 tokens
- Compleja (microservicio con mensajería): 60-70 tokens

4. Especifica optimizaciones:

- Multi-stage builds
- Image size targets (<150MB)
- Build cache strategies
- Production-ready configurations

5. Incluye restricciones críticas:

- Performance: <200ms response
- Seguridad: sanitize input, parameterized queries
- Compatibilidad: Node 18+ o .NET 8+

6. Menciona archivos abiertos si es relevante:

- #UserController.cs abierto
- Modifica #app.module.ts

7. Elige el formato adecuado:

- **C.O.R.E:** Tareas complejas, múltiples tecnologías
- **Natural:** Tareas directas, GitHub Copilot friendly

Flujo de Trabajo Completo

1. Identifica tu tarea
 - ↓
 2. Usa el meta-prompt para generar prompt optimizado
 - ↓
 3. Copia el prompt generado
 - ↓
 4. Pégalo en GitHub Copilot Chat
 - ↓
 5. Recibe código optimizado
 - ↓
 6. Refina si es necesario con prompts de seguimiento ultra-cortos
-

Meta-Prompt Maestro (Copia Todo Esto)

markdown

Eres un experto en crear prompts ultra-optimizados para IA de código.

Tarea: Genera un prompt formato C.O.R.E para: [DESCRIBE TU TAREA]

Especificaciones:

- Stack/Framework: [especifica]
- Máximo tokens: [30-70 según complejidad]
- Target: GitHub Copilot
- Formato estricto:
 - C: [contexto compacto]
 - O: [objetivo específico]
 - R: [restricciones clave]
 - E: [formato de salida]

Reglas de optimización:

- Usa abreviaturas: TS, API, CRUD, DTO, Auth, Repo, Svc, Cmp
- Sin verbos innecesarios: "crear", "hacer", "necesito"
- Símbolos: → (retorna), | (separador), + (y), ? (opcional), ! (requerido)
- Referencias: #archivo.ext en vez de copiar código
- Números específicos: <200ms, >90% coverage, :5000 port

Output: Solo el prompt C.O.R.E, sin explicaciones adicionales.

Atajos de Teclado Sugeridos

Si usas mucho este flujo, crea snippets:

VS Code snippet:

json

```
{
  "Meta-Prompt CORE": {
    "prefix": "mpc",
    "body": [
      "Genera prompt C.O.R.E <50 tokens para Copilot: $1",
      "$0"
    ]
  }
}
```

Alias de terminal:

```
bash

# En tu .bashrc o .zshrc
alias gpc='echo "Genera prompt C.O.R.E <50 tokens para Copilot:"'
```

Checklist Final

Antes de enviar tu meta-prompt, verifica:

- ☐ ¿Especifiqué el límite de tokens?
- ☐ ¿Mencioné el target (Copilot/ChatGPT/Claude)?
- ☐ ¿Incluí el stack/framework principal?
- ☐ ¿Es clara la tarea a resolver?
- ☐ ¿Agregué restricciones técnicas críticas?

Pro Tip: Guarda tus mejores meta-prompts generados en un repo de GitHub como documentación viva de tu equipo.