THE AMERICAS CONSULTING

SOFTWARE · IA · STAFFING · CIBERSEGURIDAD

FORMACIÓN PARA INSTITUCIONES

FUNDAMENTOS DE PROGRAMACIÓN CON CLAUDE CODE

Un Curso Práctico para crear software con IA

32 horas | 8 módulos | Conceptos + Laboratorios | Principiante → Avanzado | Español

Fundamentos de Programación con Claude Code

Un Curso Práctico para crear software

DESCRIPCIÓN DEL CURSO

Audiencia objetivo	Desarrolladores principiantes y avanzados que quieren aprender a programar con IA
Formato	8 módulos, 32 horas en total. Conceptos + laboratorios.
Requisitos previos	Familiaridad básica con computadoras. Sin experiencia previa en IA ni programación.
Después del curso	Una PC con 4 u 8 GB de memoria y, optativo, una subscripción a Claude (U$S 20 mensuales)

PROYECTO FINAL

Se desarrolla a lo largo de las clases y es requisito para el certificado. Construye una configuración completa de Claude Code y una aplicación real:

1. Escribe un CLAUDE.md con las convenciones del proyecto
2. Configura los permisos (allow / ask / deny)
3. Crea la base de datos
4. Crea la aplicación web
5. Crea la aplicación móvil (optativo)
6. Crea al menos un hook (p. ej., formateo automático)
7. Crea una skill o subagente personalizado
8. Demuestra un flujo de trabajo completo: tarea → plan → implementar → revisar → listo

MÓDULO 0 Preparación — 4 horas

0.1 Conceptos generales de programación (optativo — 2 horas)

- Programas y datos
- Conceptos básicos de programación: sentencias, variables, flujos de control
- Datos: archivos y bases de datos
- Las superficies: Terminal, VS Code, Web, Escritorio
- Lenguajes y generaciones
- Uso básico de git

LABORATORIO

Crea un repositorio git.

0.2 Inteligencia Artificial — Conceptos básicos (2 horas)

- Origen de la Inteligencia Artificial

— Instrucciones versus Aprendizaje
— Fei Fei Lee
— Explosión: Big Data
— Modelos de Lenguaje: Qué son y cómo piensan
— LLMs y aplicaciones: ChatGPT, Claude, Gemini, Grok, modelos privados

LABORATORIO

Prueba los LLMs públicos.

MÓDULO 1 Qué es Claude Code y Cómo Piensa — 4 horas

1.1 Introducción (1 hora)

— Qué es Claude Code (y qué no es)
— En qué se diferencia de ChatGPT, Copilot y otras herramientas de IA
— Las superficies: Terminal, VS Code, JetBrains, Web, Escritorio

1.2 Instalación y Primer Uso (1 hora)

— Instalar en macOS / Linux / Windows (WSL)
— `claude` en tu primer proyecto
— Entender el contexto de inicio

LABORATORIO

Instala Claude Code, abre un proyecto real y pídele que explique la base de código. Luego sal de la sesión y usa `claude --resume <id>`.

1.3 El Bucle Agente (2 horas)

— Cómo Claude lee tu código → planifica → actúa → evalúa
— Herramientas que Claude usa internamente: Read, Edit, Bash, Grep, Glob, WebFetch
— Qué significa "agente" en la práctica — y por qué importa

LABORATORIO

Genera fragmentos de código de prueba con Claude.

MÓDULO 2 Construcción de Aplicaciones — 4 horas

2.1 Estructura de una aplicación de red

— Base de datos (SQL y NoSQL)
— Backend (concepto y desarrollo)
— Aplicación web (en browser)
— Aplicación móvil (Android o iOS)

LABORATORIO

Imagina tu aplicación y define la arquitectura.

2.2 Base de datos

— Tablas
— Normalización
— Índices

LABORATORIO

Crea la base de datos para tu aplicación. Sube tu código a git.

MÓDULO 3 Uso Diario Básico — 4 horas

3.1 Cómo Hablar con Claude Eficazmente (1 hora)

- Dar instrucciones claras y con alcance definido
- Cuándo ser específico vs. cuándo dejar que Claude explore
- Iterar sobre los resultados

3.2 Comandos de Barra Esenciales (1 hora)

- /plan — pensar antes de actuar
- /rewind — deshacer y crear puntos de control
- /compact — gestionar el contexto
- /cost — controlar el uso de tokens
- /clear, /resume, /diff

LABORATORIO

Crea la primera versión del código de tu aplicación.

3.3 Gestión del Contexto (1 hora)

- Qué es la ventana de contexto y por qué se llena
- /context — visualizar el uso
- Cuándo y cómo comprimir
- Mantener a Claude enfocado en sesiones largas

3.4 Atajos de Teclado y Navegación (1 hora)

- Enviar, interrumpir, entrada multilínea
- Búsqueda en el historial
- Tareas en segundo plano (Ctrl+B)
- Modo Vim

LABORATORIO

Usa Claude para encontrar un bug, corregirlo, revisar el diff y deshacer el cambio — todo desde el terminal.

MÓDULO 4 Hazlo Tuyo: Configuración y Memoria — 4 horas

4.1 CLAUDE.md — La Biblia de tu Proyecto (1 hora)

- Qué es y dónde vive (global / proyecto / local)
- Qué incluir: convenciones, comandos, reglas de "nunca hacer X"
- Mantenerlo por debajo de 500 líneas
- Escribir instrucciones efectivas

4.2 La Jerarquía de Configuración (1 hora)

- ~/.claude/settings.json (usuario) vs .claude/settings.json (proyecto) vs .local

— Selección de modelo, temas, comportamiento predeterminado
— Inyección de variables de entorno

LABORATORIO
Generar nuevamente el código del proyecto usando las convenciones.

4.3 Memoria Automática (1 hora)

— Qué recuerda Claude entre sesiones
— Ver y editar la memoria con /memory
— Los 4 tipos de memoria: user, feedback, project, reference

4.4 Permisos (1 hora)

— Cómo funciona el sistema de permisos (allow / ask / deny)
— Escribir reglas de permisos: comodines, coincidencias exactas, por herramienta
— Configurar un perfil seguro por defecto para el proyecto

LABORATORIO
Escribe un CLAUDE.md para un proyecto real. Configura reglas de permisos para que Claude pueda ejecutar pruebas pero no `rm`. Genera nuevamente el código.

MÓDULO 5 Automatización: Hooks, Skills y Subagentes — 4 horas

5.1 Hooks — Automatización Determinista

- Eventos del ciclo de vida: PreToolUse, PostToolUse, SessionStart, etc.
- Tipos de hooks: command, prompt, agent, http
- Ejemplos prácticos:
  - Formatear archivos automáticamente después de cada edición
  - Bloquear escrituras en archivos protegidos
  - Notificación de escritorio cuando Claude está esperando

5.2 Skills — Flujos de Trabajo Reutilizables

- Qué son las skills y cuándo usarlas
- Crear una skill: frontmatter + instrucciones
- Skills de referencia vs. skills de acción (invocadas con /nombre)
- Compartir skills con el equipo

5.3 Subagentes — Trabajadores Especializados

- Por qué subagentes: aislamiento de contexto + paralelismo
- Agentes integrados: Explore, Plan, General-purpose
- Crear subagentes personalizados (.claude/agents/)
- Configurar herramientas, modelo, memoria y maxTurns
- Patrones prácticos: investigación paralela, ejecutores de pruebas, revisores de seguridad

LABORATORIO

Crea un hook que ejecute el linter después de cada edición. Crea un subagente que revise el código en busca de problemas de seguridad.

MÓDULO 6 Conectando al Mundo: MCP e Integración con IDEs — 4 horas

6.1 Model Context Protocol (MCP)

- Qué es MCP: conectar Claude a herramientas externas
- Instalar servidores MCP (comando /mcp, .mcp.json)
- Integraciones MCP prácticas: GitHub, bases de datos, Slack, navegadores
- Alcance: global vs. proyecto vs. sesión

6.2 Automatización del Navegador con Chrome

- Habilitar la integración con Chrome
- Probar apps web, capturas de pantalla, automatización de formularios
- Depuración con acceso a la consola

6.3 Integraciones con IDEs

- VS Code: diffs en línea, @menciones, modo de plan
- JetBrains: PyCharm, IntelliJ, WebStorm
- Mover sesiones entre superficies

LABORATORIO

Conecta Claude a un servidor MCP de GitHub y pídele que resuma los PRs abiertos desde dentro de Claude Code.

MÓDULO 7 Flujos Avanzados y Escala de Equipo — 4 horas

7.1 Trabajo en Paralelo con Worktrees

- Worktrees de Git: trabajar en dos cosas a la vez
- Patrón `claude -w feature-x "implementar X"`
- Combinar worktrees + subagentes para máximo paralelismo

7.2 Integración con CI/CD

- GitHub Actions: revisión automática de PRs, corrección de lint, triaje de issues
- Integración con GitLab CI/CD
- Tareas programadas y basadas en la nube (/schedule)

7.3 El SDK de Agentes

- Cuándo usar el SDK vs. el CLI
- SDKs de Python y TypeScript/Node.js
- Construir agentes personalizados: definición de herramientas, sesiones, streaming
- Conectar a MCP desde código

7.4 Equipo y Empresa

- Compartir CLAUDE.md, skills y subagentes a través del control de versiones
- Configuración gestionada y políticas empresariales

— Proveedores de nube: AWS Bedrock, Google Vertex, Azure Foundry
— Monitoreo de uso y costos en el equipo

LABORATORIO

Escribe un script de Node.js usando el SDK de Anthropic que toma una ruta de archivo como entrada y devuelve una revisión de código.

Tarjeta de Referencia Rápida

NECESITO…	USAR
Pensar antes de actuar	`/plan`
Deshacer algo	`/rewind`
Liberar contexto	`/compact`
Ver el costo de tokens	`/cost`
Gestionar memoria	`/memory`
Gestionar permisos	`/permissions`
Agregar integraciones	`/mcp`
Flujo de trabajo reutilizable	`.claude/skills/`
Aislar una subtarea	`.claude/agents/`
Automatizar en eventos	`Hook en settings.json`