:::tip Descargas

• [PDF de apuntes](../pdf/apuntes.pdf)
• [Presentacion Reveal.js](../reveal/)
::: Vibe Coding para Docentes **VS Code + GitHub + IA** *Apuntes operativos del curso* Objetivos de la sesión Este curso tiene como propósito familiarizar a los docentes con un conjunto de herramientas modernas para crear y gestionar contenido educativo de forma eficiente, aprovechando la inteligencia artificial. Entender el mapa de herramientas (GitHub, Git, VS Code, Copilot/Chat, modelos de IA) y cómo se relacionan Aprender a instalar y configurar VS Code en Windows para trabajar con proyectos Conectar VS Code con GitHub para habilitar Copilot/Chat y sincronización Crear estructura de trabajo basada en Markdown (README.md) para apuntes reutilizables Introducir el uso de instrucciones y agentes para guiar a la IA Comprender buenas prácticas: sesgos de modelos, riesgo de acumulación de errores, y tratamiento de legislación Ver ejemplos aplicados: generación de apuntes, diagramas con Mermaid, y figuras con Python Conocer opciones para presentaciones (Reveal.js) y alternativas canvas Mapa general de herramientas y conceptos Antes de entrar en detalles técnicos, es fundamental entender el ecosistema completo y cómo cada pieza encaja en el flujo de trabajo. Herramientas principales Herramienta Descripción ---- --- GitHub Servicio en la nube para alojar proyectos (repositorios) y colaborar; funciona como 'Drive' para proyectos, con despliegue automático de webs (GitHub Pages) Git Sistema/protocolo de control de versiones (historial de cambios). Similar al historial de un documento VS Code Editor/IDE local para trabajar con ficheros de texto (código, Markdown, etc.) Copilot/Chat Capa de IA integrada en VS Code que permite chatear, generar ficheros, modificar contenido, etc. Modelos Diferentes 'motores' de IA (OpenAI, Anthropic/Claude, Google/Gemini) disponibles según licencia. Cada uno tiene costes, límites y sesgos Tokens Presupuesto de uso de IA (consumo). Los PDFs consumen más tokens que texto plano Markdown Formato de texto plano para documentos, ideal para apuntes. Permite tablas, imágenes y fórmulas Mermaid Lenguaje de diagramas en texto que se renderiza a gráfico (diagramas de flujo, secuencias, etc.) Python Lenguaje útil para generar figuras/gráficas mediante código (ej. con Matplotlib), evitando edición manual de imágenes Ecosistema y licencias GitHub y VS Code pertenecen al ecosistema de Microsoft; Copilot está muy integrado en VS Code La licencia educativa de GitHub habilita acceso a mejores capacidades/modelos Importante: la licencia ayuda, pero no es imprescindible para empezar; se puede trabajar con modelos gratuitos Se recomienda filtrar tutoriales en internet por 'último año' porque estas herramientas cambian rápido Alternativas y herramientas complementarias Canvas (ChatGPT/Gemini/Claude) Permiten crear una web o contenido desde el chat y compartir rápido por URL. Muy accesible, menos control que VS Code. Herramienta que genera webs y estructura de ficheros. Puede pedir cuenta Google/GitHub. Suele tener límites de créditos. Antigravity IDE tipo VS Code de Google ('fork'). Integra bien Gemini, especialmente cómodo para web y con buena gestión de tokens. Cursor IDE pionero en integrar IA. Se considera muy bueno, pero normalmente es de pago (sin licencia educativa mencionada). Perplexity Útil para búsquedas con fuentes. En ciertos periodos ofreció beneficios educativos. Nota: afirmación a verificar, los programas cambian. NotebookLM Recomendado para trabajar con muchos documentos (ej. legislación), resumir y generar materiales en Markdown. Útil como paso previo. Bloque 1: Licencia educativa GitHub La obtención de la licencia educativa puede presentar dificultades. Esta sección recopila los problemas más comunes y sus soluciones. Requisitos Tener cuenta de GitHub creada Añadir/verificar un correo institucional/educativo (dominio tipo .edu o equivalente) Nota: con algunos correos (ej. Gmail) no autentican como docente; se recomienda añadir varios correos verificados Solicitar el beneficio en 'Education benefits' Errores típicos observados Denegaciones repetidas (casos de hasta 4 intentos) Verificación sensible a: calidad de cámara (webcam mala vs móvil/Mac mejor), documento usado, coincidencia exacta del nombre Checklist de verificación Cuenta y correos Añadir correos alternativos en 'Settings !’ Emails' Verificar todos los correos añadidos Tener claro cuál es el correo primario y cuáles están confirmados Nombre coherente Perfil público: nombre completo real 'Billing/Payment info': mismo nombre exacto Importante: no hace falta meter tarjeta; se refiere a la ficha de facturación con el nombre Documento y cámara Probar con un documento válido (se mencionan distintos documentos; uno 'en inglés' funcionó en un caso) Probar cámara mejor: móvil o equipo con buena webcam Tiempos Puede tardar hasta ~72 horas en verificación/actualización Reintentos Si falla la primera vez, el sistema se vuelve más 'quisquilloso' con nombres y coincidencias Estrategias si no funciona Empezar

igualmente con modelos gratuitos para aprender el flujo Usar alternativas según tarea: Para buscar información: Perplexity o Gemini; Para crear web rápida: Canvas o Vercel V0 Apoyarse en intercambio de experiencias en comunidad Bloque 2: Diferencia entre GitHub y VS Code Es fundamental entender que son herramientas distintas que se complementan: GitHub: 'como Google Drive' para proyectos. Aloja repositorios en la nube y permite historial/colaboración/despliegues Git: mecanismo de versionado (historial), similar a ver versiones de un documento VS Code: editor/IDE local (como un 'Word' local, pero para texto/código/Markdown) Símil compuesto: Git "H 'Word (control de versiones/historial)' GitHub "H 'nube donde sincronizas y compartes' Bloque 3: Instalación y configuración de VS Code (Windows) Pasos de instalación Descargar VS Code desde la web oficial Elegir instalador adecuado (arquitectura x64 en la mayoría de equipos) Ejecutar instalador y seguir pasos estándar Opciones recomendadas (Open with Code) Marcar opciones para integrar en el explorador de archivos: 'Open with Code' / abrir carpeta con VS Code desde botón derecho Ventaja: abrir una carpeta del proyecto directamente sin abrir VS Code y luego buscarla Configuración útil: Auto Save Cómo encontrarlo: Abrir ajustes: icono de engranaje !' Settings Buscar 'Auto Save' Activar autoguardado (se sugiere modo 'afterDelay') Bloque 4: Copilot/Chat en VS Code y sincronización con GitHub Cómo autenticar Al intentar usar el chat de IA, VS Code solicita iniciar sesión Elegir 'Continue' con GitHub Se abre una ventana web de GitHub para autorizar Tras autorizar, volver a VS Code: la cuenta aparece conectada Qué permite la sincronización Sincroniza configuración y preferencias del entorno entre equipos Limitación: el historial/conversaciones del chat no siempre se sincroniza. Alternativa: guardar en fichero y recuperar Menciones de '@' en el chat @workspace: preguntar sobre el contenido de la carpeta/proyecto abierto @vscode: preguntar sobre cómo usar VS Code (útil para aprender el IDE) Modelos disponibles y API keys En VS Code pueden aparecer varios modelos de IA. Dos casos distintos: Modelos incluidos con GitHub Education: funcionan sin pedir nada. Son los recomendados para empezar Modelos externos: piden API key. No son necesarios para el flujo básico Criterio didáctico: Si pide API key, no es imprescindible No tener API key no es un fallo No hay que 'perseguir modelos' Bloque 5: Estructura de trabajo con carpetas y Markdown Crear README.md Abrir una carpeta de trabajo ('workspace') en VS Code Crear un fichero README.md como punto de entrada del proyecto Importante: trabajar con texto plano Ventajas de Markdown Ligero, portable y fácil de versionar Adecuado para apuntes, documentación y webs basadas en contenido Evita formatos cerrados; funciona bien con IA Tablas, LaTeX, previsualización Markdown permite tablas que se renderizan en vista previa Permite fórmulas con LaTeX embebido (usar $...$ / $$...$$) Se trabaja con vista previa para ver el resultado renderizado Advertencia: muchos usuarios jóvenes no ven extensiones (.txt, .md) porque Windows las oculta; conviene conocerlas Organización de pantalla en VS Code Configuración recomendada con cuatro zonas visibles: Zona Contenido --- --- Izquierda Carpetas (explorador de archivos) Centro izquierda Editor de Markdown Centro derecha Vista previa (Ctrl + Shift + V) Derecha Chat de IA (View !' Chat) Configuración adicional Extensión recomendada: 'Markdown All in One' (autocompletado, atajos, mejor manejo de listas y tablas) Zoom de texto: Ctrl + o Ctrl - (o Settings !' Font Size) Tema visual: Ctrl + Shift + P !' 'Color Theme' Persistencia: VS Code recuerda la disposición, carpeta y archivos abiertos Bloque 6: Instrucciones/Agentes Qué son Ficheros que contienen reglas/instrucciones persistentes para que la IA las tenga en cuenta al trabajar en el proyecto (ej. agents.md / copilot-instructions.md). Para qué sirven Evitar repetir criterios en cada mensaje Fijar idioma (español/gallego), tono, rigor, estilo pedagógico Definir estructura de salida (apuntes + web + presentación) Incluir reglas de calidad: señalar limitaciones, ser crítico, proponer verificación Buenas prácticas Especificar idioma de trabajo y norma Pedir rigor y enfoque pedagógico sin perder precisión Pedir que sea crítico y señale errores o límites Señalar sesgos comunes: Títulos en 'Title Case' del inglés (capitalización incorrecta para español/gallego); Nivel educativo: tienden a simplificar; pedir 1-2 cursos por encima y luego adaptar Diferencia entre reglas globales vs por chat Reglas globales: fichero de instrucciones que se adjunta 'siempre' Reglas por chat/agente: sesiones separadas con objetivos distintos; útil para no contaminar trabajos Patrón recomendado Chat A (generación inicial): crear apuntes o estructura Chat B (contrarian prompting): criticar, detectar errores, lagunas Chat C (adaptación): convertir a web o presentación Ventaja: evita sesgos por contexto previo, cada chat tiene un rol claro Bloque 7: Estrategia de prompting aplicada a educación Diseño exhaustivo vs iterar sobre borrador Recomendación central: funciona mejor diseñar bien lo que se quiere (objetivos, criterios, contexto, fuentes) y luego pedir a la IA que lo ejecute. La alternativa común ('crea borrador y corrige') se considera menos efectiva. Cuanto más contexto y criterios se aporten, mejor el resultado. Contrarian prompting Técnica recomendada para detectar fallos: Generar una primera versión con un modelo Pasarla a otro modelo distinto y pedirle que 'destroce' el texto: detectar fallos, imprecisiones, mejoras

Motivo: los modelos tienen sesgo a 'darte la razón'; un segundo modelo criticando reduce ese sesgo Sesgos por nivel educativo Observación: tienden a dar materiales demasiado simples (posible sesgo por corpus/sistema educativo). Estrategia: pedir un nivel más alto (1-2 cursos por encima) y luego simplificar Riesgo de acumulación de errores Si una frase inicial tiene un error y no se corrige, la IA puede usarlo como premisa y el fallo se amplifica Buen hábito: revisar por pasos, corregir antes de seguir Bloque 8: Legislación educativa Advertencia: riesgo de inventar normas Alto riesgo de 'alucinación' en legislación si no se aportan documentos. Se insiste en revisar especialmente todo lo legislativo. Recomendación: descargar PDFs y convertir a TXT/Markdown Flujo recomendado: Descargar PDFs oficiales Convertir a texto plano (TXT o Markdown) Guardar en carpeta del proyecto para que forme parte del contexto Por qué TXT consume menos tokens que PDF Para la IA, leer PDF implica extraer texto (más costoso) Texto plano se procesa mejor, más rápido y con menos coste Propuesta de estructura de carpeta Carpeta legislacion/ con: Documentos convertidos a .txt o .md Resúmenes 'clave' generados y revisados Carpeta protocolos/ separada para no mezclar Bloque 9: Generación de apuntes y recursos Mermaid para diagramas Mermaid permite describir diagramas en texto y renderizarlos como gráficos: Diagramas de flujo Clases Secuencias Ventaja: la IA trabaja mejor con estructuras textuales que con píxeles Python para generar figuras En lugar de generar 'imágenes' directamente con un modelo de imagen, se propone generar código que dibuje figuras. Flujo recomendado: Crear un script Python (ej. generar_figuras.py) Instalar dependencias con pip (numpy, matplotlib) Ejecutar el script para generar .png Insertar las imágenes en Markdown con enlaces a los .png Ventaja: pasar código vs pasar PNG (píxeles). Si quieres modificar una figura, mejor modificar el código. El código contiene la estructura y la intención; el PNG es solo resultado. Estrategia para matemáticas, diagramas y gráficas Cuando se incluyen matemáticas o gráficas hay tres niveles posibles: Nivel Herramienta Uso --- --- --- Nivel 1 LaTeX embebido Expresiones matemáticas, ecuaciones, demostraciones breves. Se integran en el texto Nivel 2 Mermaid Flujos, relaciones, procesos, estructuras lógicas. Describen ideas, no datos Nivel 3 Python + Matplotlib Funciones matemáticas, distribuciones, comparativas, datos reales o simulados Publicar en GitHub desde VS Code VS Code ofrece botón/flujo de 'Publish to GitHub' si estás autenticado. Advertencia importante: Git/GitHub es potente, pero cuando surge un conflicto serio, incluso técnicos se frustran La IA puede ayudar, pero hay complejidad real Bloque 10: Reutilización del contenido en múltiples formatos El núcleo del enfoque es que el contenido se escribe una sola vez y se reutiliza para distintos fines. Formato Uso --- --- Apuntes base Fuente única del conocimiento en Markdown. Aquí se escribe y revisa todo Web (Docusaurus) Páginas navegables, índice lateral, enlaces entre temas. Para consulta, alumnado, actualización continua Presentaciones Reveal.js: una idea por diapositiva, estructura jerárquica. Para clases presenciales, exposiciones PDF Se genera a partir de web o presentación. Para impresión, entrega formal Bloque 11: Web de apuntes con Docusaurus Docusaurus resuelve un problema concreto: 'Tengo muchos apuntes y quiero que se lean como un sitio web ordenado.' Qué hace Docusaurus Toma archivos Markdown Crea un menú lateral automático Genera una web estática Permite búsqueda y navegación Por qué es adecuado para apuntes No requiere base de datos No hay que programar páginas una a una Se actualiza con solo cambiar un archivo de texto Publicación y actualización automática 'Desplegar' significa convertir archivos en una web accesible por URL. Flujo real: Se modifica un archivo Markdown Se sube el cambio a GitHub GitHub ejecuta un proceso automático La web se regenera sola Valor pedagógico: El alumnado ve siempre la versión vigente El docente corrige una vez y se refleja en todo Bloque 12: Presentaciones Opciones tradicionales Para presentaciones 'tradicionales' y visuales: Canva y Gamma funcionan bien NotebookLM puede generar y exportar a Google Slides para editar después Reveal.js: presentaciones como web Reveal.js permite hacer presentaciones como web: Pantalla completa en navegador Navegación por diapositivas Posibilidad de jerarquía (mover 'derecha' o 'abajo' para subniveles) Admite código, transiciones y elementos web Ventaja para trabajar con IA: Al ser estructura basada en texto/HTML/Markdown, la IA la manipula mejor que un lienzo gráfico Exportación a PDF desde Reveal.js Abrir la presentación en el navegador Activar modo impresión Guardar como PDF Advertencias habituales: Revisar tamaño de página Comprobar saltos de diapositiva Verificar fuentes Problemas frecuentes y soluciones (FAQ) Problema Solución --- --- La licencia educativa falla o la deniegan Revisar nombre exacto en perfil y 'billing', usar mejor cámara, probar documento alternativo, asumir tiempos de espera, reintentar La IA inventa datos concretos (especialmente legislación) Exigir documentos oficiales en el contexto; revisar y verificar; preparar material previo en herramientas de búsqueda/documentos El contenido sale demasiado simple Pedir nivel más alto (1-2 cursos por encima) y luego adaptar Cambios en Git se complican Empezar por flujos simples; aceptar que los conflictos existen; pedir ayuda o usar IA con cautela; revisar antes de

aprobar acciones La IA mezcla contextos de chats Separar sesiones (agentes/chats) por proyecto o propósito; usar instrucciones claras de qué debe ignorar Checklist final de trabajo recomendado Crear cuenta GitHub y añadir/verificar correos necesarios Solicitar licencia educativa (si aplica) y preparar reintentos con checklist Instalar VS Code en Windows y activar 'Open with Code' Activar Auto Save en VS Code Autenticar VS Code con GitHub para habilitar Copilot/Chat y sincronización Crear una carpeta de proyecto y abrirla en VS Code Crear README.md y empezar apuntes en Markdown Crear fichero de instrucciones con: idioma, rigor, estilo de títulos, necesidad de crítica/ limitaciones, estructura del proyecto Si hay legislación: descargar PDFs oficiales, convertir a TXT/ Markdown, guardar en legislacion/ Generar apuntes por temas con IA, revisando por pasos Para diagramas: usar Mermaid cuando convenga Para figuras/gráficas: generar con Python + librerías, guardar y enlazar desde .md Para presentaciones: elegir herramienta según objetivo (Canva/Gamma/ NotebookLM o Reveal.js) Glosario de términos y siglas Término Definición --- --- Git Sistema de control de versiones (historial de cambios) GitHub Plataforma en la nube para repositorios Git Repositorio Carpeta/proyecto con historial (Git) alojable en GitHub IDE Entorno de desarrollo (editor con herramientas); VS Code es un IDE/editor VS Code Visual Studio Code Copilot/Chat Asistente de IA integrado en VS Code (conexión a modelos) Modelo Motor de IA (familias GPT, Claude, Gemini) Tokens Unidad de consumo de IA (coste/limitaciones por texto procesado) Markdown Formato de texto plano para documentación LaTeX Sintaxis para fórmulas matemáticas Mermaid Lenguaje de diagramas en texto que se renderiza a gráfico Python Lenguaje de programación útil para scripts y generación de figuras/datos PIP Gestor de paquetes de Python para instalar librerías Dependencias Código de terceros reutilizable (ej. Matplotlib, NumPy) Fork Variante de un proyecto basada en otro (Antigravity como 'fork' de VS Code) Canvas Interfaz de creación/edición visual desde chat (ChatGPT/Gemini/Claude) Ejercicio guiado: Creación completa de una web de apuntes CREACIÓN COMPLETA DE UNA WEB DE APUNTES CON GITHUB PAGES, DOCUSAURUS, REVEAL.JS Y PDF USANDO IA DESDE VS CODE Objetivo del ejercicio Crear desde cero un proyecto educativo que: tenga apuntes en Markdown como fuente única genere automáticamente: una web pública con Docusaurus en GitHub Pages, presentaciones interactivas con Reveal.js (una por tema), un PDF con todos los apuntes se despliegue automáticamente al hacer cambios y cuya estructura, configuración y scripts sean creados por la IA desde VS Code, no a mano El docente no escribe código, no crea carpetas manualmente, no configura GitHub Actions a mano: dirige el proceso con instrucciones claras y revisa. Requisitos previos del ejercicio VS Code instalado Cuenta de GitHub activa (preferiblemente con licencia educativa) VS Code iniciado sesión con GitHub Copilot / Chat disponible en VS Code No es necesario: saber Node.js saber Python saber Git avanzado Fase 1: Crear el proyecto base (carpeta vacía) En el sistema operativo: crear una carpeta vacía (por ejemplo: proyecto-apuntes-web) Abrir esa carpeta con VS Code: botón derecho !’ "Open with Code" En este punto: no hay subcarpetas, no hay archivos, no hay configuración. Fase 2: Pedir a la IA que diseñe el proyecto Abrir el chat de IA en VS Code. Primer prompt (diseño global) Texto literal recomendado: "Quiero crear un proyecto educativo basado en Markdown que genere: una web de apuntes con Docusaurus publicada en GitHub Pages, presentaciones interactivas con Reveal.js (una por tema), un PDF con todos los apuntes. Diseña la estructura completa del proyecto, indicando carpetas, herramientas y flujo de trabajo. No crees aún los archivos: solo explícame el plan." La IA responde con: explicación del flujo, propuesta de estructura, tecnologías necesarias. El docente: lee, valida que encaja con lo que quiere, pide ajustes si es necesario. Fase 3: Creación automática de la estructura del proyecto Segundo prompt (ejecución) "Ahora crea la estructura completa del proyecto según lo explicado. Incluye: carpeta de apuntes en Markdown, carpeta para Docusaurus, carpeta de presentaciones Reveal.js, carpeta de recursos (imágenes y scripts), scripts para generar PDF, archivos de configuración necesarios. Crea también un README.md explicativo." La IA: crea carpetas, crea archivos, muestra los cambios propuestos en VS Code. El docente: revisa, acepta los cambios. En este momento ya aparece una estructura equivalente a: apuntes/ docs/ presentacion/ recursos/ scripts/ package.json docusaurus.config.js sidebars.js README.md Fase 4: Crear los apuntes (fuente única) En la carpeta apuntes/: PROMPT TIPO: "Crea los apuntes del Tema 1 en Markdown. Lenguaje claro, rigor conceptual, con ejemplos. Incluye: texto explicativo, tablas, fórmulas matemáticas con LaTeX si procede, indicaciones para añadir figuras." La IA crea 01_tema.md. Se repite el proceso para cada tema. IMPORTANTE: estos archivos son la fuente única: la web, las presentaciones, el PDF salen de aquí. Fase 5: Generación de figuras y gráficas con Python PROMPT TIPO: "Genera un script en Python que cree las figuras necesarias para los apuntes: gráficas matemáticas, esquemas simples. Usa matplotlib y guarda las imágenes en la carpeta recursos/imagenes. Incluye comentarios en el código." La IA: crea recursos/generar_figuras.py,

incluye importaciones, guarda PNGs. Luego: esas imágenes se enlazan desde los .md. VENTAJA DIDÁCTICA: si cambia el ejercicio, se regenera la figura no se edita una imagen a mano Fase 6: Adaptar los apuntes para Docusaurus PROMPT TIPO: "Adapta los apuntes de la carpeta apuntes/ para que se usen en Docusaurus. Crea versiones en docs/apuntes/ con el frontmatter necesario. Configura el sidebar con todos los temas." La IA: copia/adapta los Markdown, añade frontmatter, configura navegación. Fase 7: Crear las presentaciones con Reveal.js PROMPT TIPO: "Para cada tema, crea una presentación en Reveal.js a partir de los apuntes. Una presentación por tema. Incluye: una diapositiva por idea clave, navegación clara, índice general." La IA: crea archivos HTML en presentacion/, crea un index.html con enlaces a todas. Estas presentaciones: se ven en navegador, se integran en la web, se pueden imprimir a PDF. Fase 8: Generación del PDF PROMPT TIPO: "Crea un script que genere un PDF con todos los apuntes. Usa Pandoc y LaTeX. El PDF debe incluir: portada, índice, todos los temas en orden." La IA: crea scripts/build-pdf.py, o comandos npm equivalentes. El PDF se genera automáticamente en el flujo de build. Fase 9: Configurar GitHub Pages con IA PROMPT CLAVE (ESTE ERA EL QUE FALTABA EN LOS APUNTES): "Configura el proyecto para que se despliegue automáticamente en GitHub Pages usando GitHub Actions. Crea el workflow necesario en .github/workflows/deploy.yml. El despliegue debe ejecutarse en cada push a main." La IA: crea el workflow, configura Node, Python, ejecuta build web, PDF y presentaciones, publica en GitHub Pages. El docente: NO escribe YAML a mano, solo revisa. Fase 10: Publicar el repositorio En VS Code: botón "Publish to GitHub" o prompt: "Publica este proyecto en un repositorio nuevo en mi cuenta de GitHub." Después: ir a GitHub !’ Settings !’ Pages, seleccionar "GitHub Actions". Resultado: la web aparece en la URL del repositorio cada cambio actualiza automáticamente Fase 11: Exportar presentaciones a PDF Para cada presentación Reveal.js: abrir en navegador activar modo impresión exportar a PDF No se necesita software adicional. Resultado final del ejercicio Un proyecto que: se edita en Markdown se publica como web genera presentaciones genera PDF se mantiene solo y puede reutilizarse cada curso Exactamente el modelo del README que has compartido.