

# **Apuntes y notas personales de Vibe Coding**

---

Apuntes generados automáticamente

## Introducción {#introduccion}

### Objetivos de la sesión {#objetivos-sesion}

Este curso tiene como propósito familiarizar a los docentes con un conjunto de herramientas de programación para crear y gestionar contenido educativo de forma eficiente, aprovechando la inteligencia artificial.

Al finalizar la sesión, el docente será capaz de:

- 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
  - Tratamiento de legislación
- Ver ejemplos aplicados:
  - Generación de apuntes
  - Diagramas con Mermaid
  - Figuras con Python
- Conocer opciones para presentaciones:
  - Reveal.js
  - Canvas y alternativas

---

## Mapa general de herramientas y conceptos {#mapa-general}

Antes de entrar en detalles técnicos, es fundamental entender el ecosistema completo y cómo cada pieza encaja en el flujo de trabajo.

## Herramientas principales {#herramientas-principales}

Herramienta	Descripción
GitHub	Servicio en la nube para alojar proyectos (repositorios) y colaborar; funciona como un “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 y modificar contenido.
Modelos	Motores de IA (OpenAI, Claude, Gemini). Cada uno tiene costes, límites y sesgos.
Tokens	Presupuesto de uso de IA. Los PDFs consumen más tokens que texto plano.
Markdown	Formato de texto plano ideal para apuntes. Permite tablas, imágenes y fórmulas.
Mermaid	Lenguaje de diagramas en texto que se renderiza a gráfico.
Python	Lenguaje útil para generar figuras y gráficas mediante código (ej. Matplotlib).

---

## Ecosistema y licencias {#ecosistema-licencias}

- GitHub y VS Code pertenecen al ecosistema Microsoft
- Copilot está muy integrado en VS Code
- La licencia educativa de GitHub:
  - Habilita mejores capacidades y modelos
- Importante:
  - La licencia ayuda, pero no es imprescindible
  - Se puede trabajar con modelos gratuitos

- Recomendación:
  - Filtrar tutoriales por “último año” (estas herramientas cambian rápido)
- 

## Alternativas y herramientas complementarias {#alternativas-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.

### Herramientas de generación de webs

Generan estructura de ficheros. Pueden pedir cuenta Google/GitHub. Suelen tener límites de créditos.

### Antigravity

IDE tipo VS Code de Google (“fork”). Integra bien Gemini, cómodo para web y con buena gestión de tokens.

### Cursor

IDE pionero en integrar IA. Muy potente, normalmente de pago (no se menciona licencia educativa).

### Perplexity

Útil para búsquedas con fuentes. En ciertos periodos ofreció beneficios educativos.

Nota: afirmación a verificar, los programas cambian.

### NotebookLM

Muy recomendable para trabajar con muchos documentos (ej. legislación), resumir y generar Markdown. Útil como paso previo.

---

## Apuntes y notas de teoría (Bloques 1 a 12) {#teoria}

### Bloque 1: Licencia educativa GitHub {#bloque-1}

La obtención de la licencia educativa puede presentar dificultades.

Esta sección recopila los problemas más comunes y sus soluciones.

## Requisitos {#b1-requisitos}

- Tener cuenta de GitHub creada
- Añadir y verificar un correo institucional/educativo ([.edu](#) o equivalente)
- Solicitar el beneficio en “Education benefits”

Nota: algunos correos (ej. Gmail) no autentican como docente; se recomienda añadir varios correos verificados.

---

## Errores típicos observados {#b1-errores}

- Denegaciones repetidas (casos de hasta 4 intentos)
  - Verificación sensible a:
  - Calidad de cámara
  - Documento usado
  - Coincidencia exacta del nombre
- 

## Checklist de verificación {#b1-checklist}

### Cuenta y correos

- Añadir correos alternativos en [Settings → Emails](#)
- Verificar todos los correos
- Tener claro el correo primario

### Nombre coherente

- Perfil público: nombre completo real
- “Billing / Payment info”: mismo nombre exacto
- (no hace falta tarjeta; solo el nombre)

### Documento y cámara

- Probar documentos distintos
- Usar mejor cámara (móvil o buena webcam)

### Tiempos

- La verificación puede tardar hasta ~72 horas

### Reintentos

- Tras un fallo, el sistema se vuelve más exigente
- 

### Estrategias si no funciona {#b1-estrategias}

- Empezar igualmente con modelos gratuitos
  - Usar alternativas según tarea:
  - Búsqueda: Perplexity o Gemini
  - Web rápida: Canvas o Vercel V0
  - Apoyarse en la comunidad
- 

## Bloque 2: Diferencia entre GitHub y VS Code {#bloque-2}

Son herramientas distintas que se complementan:

- GitHub: “como Google Drive” para proyectos
- Git: mecanismo de versionado (historial)
- VS Code: editor/IDE local

### Símil compuesto:

- Git ≈ Word (control de versiones)
  - GitHub ≈ nube donde sincronizas y compartes
- 

## Bloque 3: Instalación y configuración de VS Code (Windows) {#bloque-3}

### **Pasos de instalación {#b3-pasos}**

- Descargar VS Code desde la web oficial
  - Elegir instalador adecuado (x64 en la mayoría)
  - Ejecutar instalador y seguir pasos estándar
- 

### **Opciones recomendadas: "Open with Code" {#b3-open}**

Permite abrir una carpeta con VS Code desde el botón derecho del explorador.

Ventaja: abrir directamente la carpeta del proyecto.

---

### **Configuración útil: Auto Save {#b3-autosave}**

- Abrir ajustes: ícono de engranaje → Settings
  - Buscar “Auto Save”
  - Activar modo recomendado: `afterDelay`
- 

## **Bloque 4: Copilot / Chat en VS Code {#bloque-4}**

### **Autenticación {#b4-autenticacion}**

- VS Code solicita iniciar sesión
  - Elegir “Continue with GitHub”
  - Autorizar en la web
  - Volver a VS Code: cuenta conectada
- 

### **Sincronización {#b4-sincronizacion}**

- Sincroniza configuración y preferencias

- Limitación: el historial del chat no siempre se sincroniza
  - → alternativa: guardar conversaciones en `.md`
- 

### Menciones en el chat {#b4-menciones}

- `@workspace`: preguntar sobre el proyecto
  - `@vscode`: aprender a usar el IDE
- 

### Modelos y API keys {#b4-modelos}

- Modelos incluidos con GitHub Education:
- Funcionan sin pedir nada
- Modelos externos:
- Piden API key
- No son necesarios

Criterio didáctico:

Si pide API key, no es imprescindible. No tener API key no es un fallo.

---

## Bloque 5: Estructura de trabajo con Markdown {#bloque-5}

### Crear README.md {#b5-readme}

- Abrir una carpeta de trabajo en VS Code
  - Crear `README.md` como punto de entrada
  - Trabajar siempre con texto plano
- 

### Ventajas de Markdown {#b5-ventajas}

- Ligero y portable
  - Ideal para apuntes y documentación
  - Funciona bien con IA
  - Permite tablas, imágenes y fórmulas LaTeX
- 

### Vista previa de Markdown (muy importante) {#b5-vista}

- Abrir vista previa:
  - **Ctrl + Shift + V**
  - Abrir vista previa lateral:
  - **Ctrl + K, luego V**
  - Alternar foco editor / preview:
  - **Ctrl + 1, Ctrl + 2**
- 

### Advertencia sobre extensiones {#b5-advertencia}

Windows suele ocultar `.md` y `.txt`. Conviene enseñar a mostrarlas.

---

### Organización de pantalla recomendada {#b5-organizacion}

Zona	Contenido
Izquierda	Explorador de archivos
Centro izquierda	Editor Markdown
Centro derecha	Vista previa
Derecha	Chat de IA

---

## Configuración adicional {#b5-config}

- Extensión: **Markdown All in One**
  - Zoom:  
`Ctrl + + / Ctrl + -`
  - Tema:  
`Ctrl + Shift + P` → Color Theme
  - VS Code recuerda la disposición y archivos abiertos
- 

## Bloque 6: Instrucciones y agentes {#bloque-6}

### Qué son {#b6-que}

Archivos con reglas persistentes para guiar a la IA (`agents.md`, `copilot-instructions.md`).

### Para qué sirven {#b6-para}

- Evitar repetir criterios
  - Fijar idioma, tono y rigor
  - Definir estructura de salida
  - Exigir crítica y verificación
- 

### Buenas prácticas {#b6-buenas}

- Especificar idioma
  - Pedir rigor pedagógico
  - Señalar sesgos comunes:
  - Capitalización inglesa
  - Simplificación excesiva
-

### **Reglas globales vs por chat {#b6-reglas}**

- Globales: siempre activas
  - Por chat: sesiones separadas
- 

### **Patrón recomendado {#b6-patron}**

- Chat A: generación
  - Chat B: crítica
  - Chat C: adaptación
- 

## **Bloque 7: Estrategia de prompting {#bloque-7}**

### **Diseño vs iteración {#b7-diseno}**

Mejor diseñar bien objetivos y criterios antes de ejecutar.

La iteración sobre borradores es menos efectiva.

---

### **Contrarian prompting {#b7-contrarian}**

- Generar versión 1
  - Pasarla a otro modelo
  - Pedir crítica dura
- 

### **Sesgos por nivel educativo {#b7-sesgos}**

- Tienden a simplificar
- Pedir 1–2 cursos por encima y luego adaptar

---

## Riesgo de acumulación de errores {#b7-riesgo}

Un error inicial se amplifica.

Revisar y corregir por fases.

---

## Bloque 8: Legislación educativa {#bloque-8}

■■ Alto riesgo de inventar normas si no se aportan documentos.

### Flujo recomendado {#b8-flujo}

- Descargar PDFs oficiales
  - Convertir a TXT o Markdown
  - Guardar en el proyecto
- 

### Por qué TXT consume menos tokens {#b8-txt}

PDF requiere extracción de texto; TXT se procesa directamente.

---

### Estructura de carpetas {#b8-estructura}

```
legislacion/
    ■■■ normas.md
    ■■■ decretos.txt
protocolos/
```

## Bloque 9: Generación de apuntes y recursos {#bloque-9}

### Mermaid para diagramas {#b9-mermaid}

Mermaid permite describir diagramas en texto y renderizarlos como gráficos:

- Diagramas de flujo
- Diagramas de clases
- Diagramas de secuencia

#### Ventaja clave:

La IA trabaja mejor con **estructuras textuales** que con píxeles. El diagrama es reproducible, editable y versionable.

---

### Python para generar figuras {#b9-python}

En lugar de generar imágenes directamente con modelos gráficos, se recomienda generar **código que dibuje las figuras**.

#### Flujo recomendado:

- Crear un script Python (por ejemplo `generar_figuras.py`)
- Instalar dependencias con `pip`:
  - `numpy`
  - `matplotlib`
- Ejecutar el script para generar archivos `.png`
- Insertar las imágenes en Markdown enlazando los `.png`

#### Ventaja didáctica:

Modificar código es mejor que editar imágenes.

El código contiene la **estructura y la intención**; el PNG es solo el resultado.

---

### Estrategia para matemáticas, diagramas y gráficas {#b9-estrategia}

Cuando se incluyen matemáticas o gráficas hay **tres niveles posibles**:

Nivel	Herramienta	Uso
Nivel 1	LaTeX embebido	Expresiones matemáticas, ecuaciones, demostraciones breves

Nivel 2	Mermaid	Flujos, relaciones, procesos, estructuras lógicas
Nivel 3	Python + Matplotlib	Funciones matemáticas, datos reales o simulados

---

## Publicar en GitHub desde VS Code {#b9-publicar}

VS Code ofrece un flujo directo de “**Publish to GitHub**” si estás autenticado.

### Advertencia importante:

- Git y GitHub son herramientas potentes
  - Cuando surge un conflicto serio, incluso perfiles técnicos se frustran
  - La IA puede ayudar, pero existe complejidad real
  - Revisar siempre antes de aceptar acciones automáticas
- 

## Bloque 10: Reutilización del contenido en múltiples formatos {#bloque-10}

El núcleo del enfoque es que el contenido se escribe **una sola vez** y se reutiliza.

Formato	Uso
Apuntes base	Fuente única del conocimiento en Markdown
Web (Docusaurus)	Consulta navegable para alumnado y actualización continua
Presentaciones	Reveal.js: una idea por diapositiva
PDF	Generado desde web o presentación

---

## Bloque 11: Web de apuntes con Docusaurus {#bloque-11}

Docusaurus resuelve el problema:

“Tengo muchos apuntes y quiero que se lean como un sitio web ordenado.”

### Qué hace Docusaurus {#b11-que}

- Toma archivos Markdown
  - Crea un menú lateral automático
  - Genera una web estática
  - Permite búsqueda y navegación
- 

### Por qué es adecuado para apuntes {#b11-por-que}

- No requiere base de datos
  - No hay que programar páginas una a una
  - Se actualiza con solo modificar un archivo de texto
- 

### Publicación y actualización automática {#b11-publicacion}

**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 {#b11-valor}

- El alumnado ve siempre la versión vigente
  - El docente corrige una vez y se refleja en todo
-

## Bloque 12: Presentaciones {#bloque-12}

### Opciones tradicionales {#b12-opciones}

Para presentaciones visuales tradicionales:

- Canva
  - Gamma
  - NotebookLM (exporta a Google Slides)
- 

### Reveal.js: presentaciones como web {#b12-reveal}

Reveal.js permite crear presentaciones como web:

- Pantalla completa en navegador
- Navegación por diapositivas
- Jerarquía (derecha / abajo para subniveles)
- Código, transiciones y elementos web

#### Ventaja para trabajar con IA:

Al estar basado en texto/HTML/Markdown, la IA lo manipula mejor que un lienzo gráfico.

---

### Exportación a PDF desde Reveal.js {#b12-pdf}

- Abrir la presentación en el navegador
- Activar modo impresión
- Guardar como PDF

#### Advertencias habituales:

- Revisar tamaño de página
- Comprobar saltos de diapositiva
- Verificar fuentes

---

## Preguntas frecuentes {#preguntas-frecuentes}

### Problemas frecuentes y soluciones (FAQ) {#faq}

Problema	Solución
La licencia educativa falla	Revisar nombre exacto, cámara, documento, tiempos y reintentos
La IA inventa legislación	Exigir documentos oficiales en el contexto
El contenido sale demasiado simple	Pedir nivel 1–2 cursos por encima
Cambios en Git se complican	Empezar por flujos simples, revisar antes de aceptar
La IA mezcla contextos	Separar chats o agentes por propósito

---

## Checklist {#checklist}

### Checklist final de trabajo recomendado {#checklist-final}

- Crear cuenta de GitHub y verificar correos
- Solicitar licencia educativa (si aplica)
- Instalar VS Code y activar “Open with Code”
- Activar Auto Save
- Autenticar VS Code con GitHub
- Crear carpeta de proyecto
- Crear `README.md`
- Crear fichero de instrucciones
- Preparar legislación en TXT o Markdown
- Generar apuntes por temas
- Usar Mermaid para diagramas
- Usar Python para figuras
- Elegir formato final (web, presentación, PDF)

---

## Glosario y términos {#glosario-terminos}

### Glosario de términos y siglas {#glosario}

Término	Definición
Git	Sistema de control de versiones
GitHub	Plataforma en la nube para repositorios Git
Repositorio	Proyecto con historial Git
IDE	Entorno de desarrollo
VS Code	Visual Studio Code
Copilot / Chat	Asistente de IA integrado en VS Code
Modelo	Motor de IA (GPT, Claude, Gemini)
Tokens	Unidad de consumo de IA
Markdown	Formato de texto plano
LaTeX	Sintaxis para fórmulas matemáticas
Mermaid	Lenguaje de diagramas en texto
Python	Lenguaje para scripts y figuras
PIP	Gestor de paquetes de Python
Dependencias	Librerías externas reutilizables
Fork	Variante de un proyecto
Canvas	Interfaz visual desde chat

---

## Ejercicio guiado {#ejercicio}

### Ejercicio guiado: Creación completa de una web de apuntes {#ejercicio-guiado}

#### Objetivo del ejercicio {#ej-objetivo}

Crear desde cero un proyecto educativo que:

- Use Markdown como fuente única
- Genere automáticamente:
- Web con Docusaurus (GitHub Pages)
- Presentaciones con Reveal.js
- Un PDF con todos los apuntes
- Se despliegue automáticamente al hacer cambios
- Sea creado y configurado **por IA desde VS Code**

El docente **no escribe código a mano**: dirige, revisa y valida.

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
- Tenga su **estructura, configuración y scripts creados por la IA desde VS Code**, no a mano

**Rol del docente:**

El docente **no escribe código, no crea carpetas manualmente y no configura GitHub Actions a mano**.

Dirige el proceso con instrucciones claras, revisa y valida.

---

## Requisitos previos del ejercicio {#ej-requisitos}

Necesario:

- VS Code instalado
- Cuenta de GitHub activa
- (preferiblemente con licencia educativa)
- VS Code iniciado sesión con GitHub

- Copilot / Chat disponible en VS Code
- 

## Fases del ejercicio {#ej-fases}

El ejercicio se estructura en fases progresivas:

- Crear carpeta vacía
  - Diseñar el proyecto con IA
  - Crear estructura automática
  - Generar apuntes en Markdown
  - Crear figuras con Python
  - Adaptar apuntes a Docusaurus
  - Crear presentaciones Reveal.js
  - Generar PDF
  - Configurar GitHub Pages con IA
  - Publicar el repositorio
  - Exportar presentaciones a PDF
- 

## Fase 1: Crear el proyecto base (carpeta vacía) {#ej-f1}

En el sistema operativo:

- Crear una carpeta vacía
- Ejemplo: `proyecto-apuntes-web`
- Abrir la 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 {#ej-f2}

- Abrir el **chat de IA** en VS Code

### Primer prompt (diseño global)

En la caja del chat de VS Code vamos a utilizar el modo Plan (no Agent) con la IA más potente (3X). Es mejor emplear más tokens en tener un proyecto bien definido y después utilizar el modo Agente con otra IA que sea menos potente para ejecutarlo.

#### 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 general del proyecto
- Propuesta de estructura de carpetas
- Tecnologías necesarias

#### Rol del docente:

- Lee la propuesta
- Valida que encaja con sus objetivos
- Pide ajustes si es necesario

---

## Fase 3: Creación automática de la estructura del proyecto {#ej-f3}

### Segundo prompt (ejecución)

Ahora es cuando escogemos el modo Agent con un modelo de IA que tenga un menor consumo (1X)

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 el 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

### Estructura resultante aproximada

```
apuntes/
docs/
presentacion/
recursos/
scripts/
package.json
docusaurus.config.js
sidebars.js
README.md
```

---

### Fase 4: Crear los apuntes (fuente única) {#ej-f4}

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 el archivo: `01_tema.md`

Este proceso se repite para cada tema.

## Importante

Estos archivos son la **fuente única** del proyecto:

- La web
- Las presentaciones
- El PDF

Todo se genera a partir de estos Markdown.

---

## Fase 5: Generación de figuras y gráficas con Python {#ej-f5}

### Prompt tipo

Genera un script en Python que cree las figuras necesarias para los apuntes:  
gráficas matemáticas y 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
- Genera archivos `.png`

### Después:

Las imágenes se enlazan desde los archivos `.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 {#ej-f6}

### 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 y adapta los archivos Markdown
  - Añade el frontmatter requerido
  - Configura la navegación lateral
- 

### Fase 7: Crear las presentaciones con Reveal.js {#ej-f7}

#### 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 la carpeta `presentacion/`
- Crea un `index.html` con enlaces a todas las presentaciones

#### Las presentaciones:

- Se visualizan en el navegador
  - Se integran en la web
  - Se pueden exportar a PDF
- 

### Fase 8: Generación del PDF {#ej-f8}

#### 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 equivalentes
  - Integra la generación del PDF en el flujo de build
- El PDF se genera automáticamente.
- 

### Fase 9: Configurar GitHub Pages con IA {#ej-f9}

#### Prompt tipo

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 y Python
- Ejecuta la build de la web, el PDF y las presentaciones
- Publica el resultado en GitHub Pages

El docente no escribe YAML, solo revisa.

---

### Fase 10: Publicar el repositorio {#ej-f10}

Desde VS Code:

- Botón "**Publish to GitHub**"

o mediante prompt:

Publica este proyecto en un repositorio nuevo en mi cuenta de GitHub.

#### Después:

- Ir a GitHub → Settings → Pages
  - Seleccionar GitHub Actions
  - La web aparece en la URL del repositorio
  - Cada cambio se despliega automáticamente
- 

### Fase 11: Exportar presentaciones a PDF {#ej-f11}

Para cada presentación Reveal.js:

- Abrir la presentación en el navegador
- Activar modo impresión
- Exportar a PDF

No se necesita software adicional.

---

### Resultado esperado {#ejresultado}

Un proyecto que:

- ■ Se edita en Markdown
- ■ Se publica como web
- ■ Genera presentaciones
- ■ Genera PDF
- ■ Se mantiene automáticamente
- ■ Puede reutilizarse cada curso