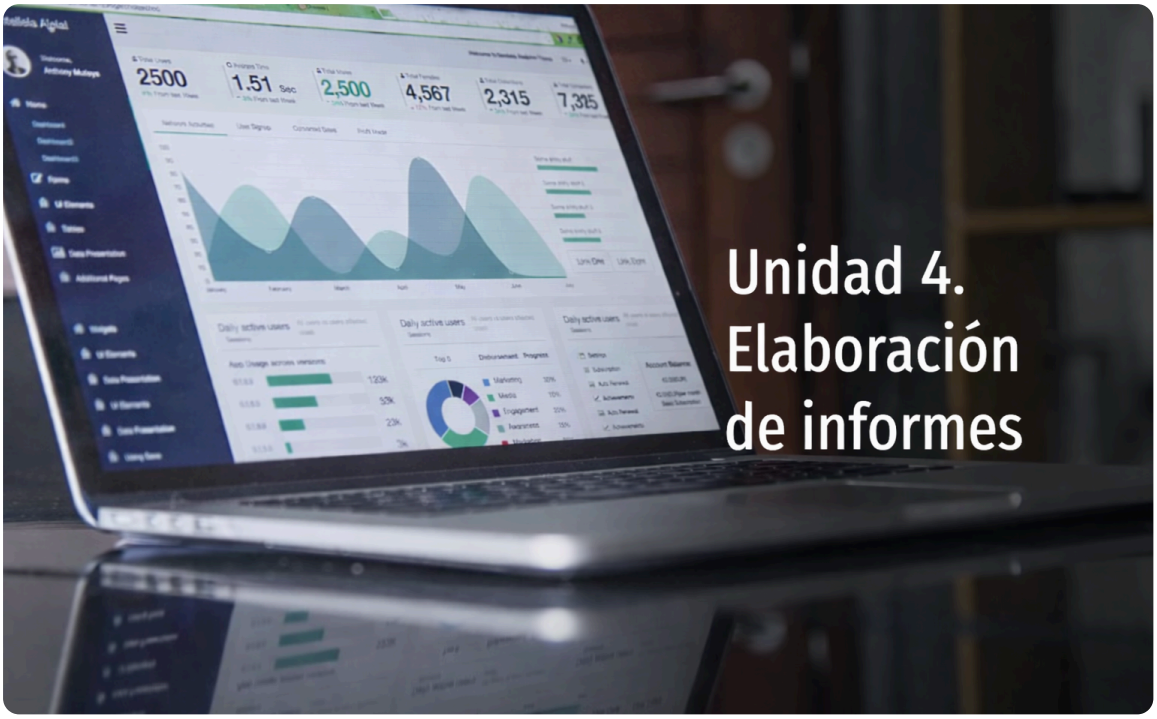

TEMA 4 - ELABORACIÓN DE INFORMES

Índice de contenidos

1. Introducción al diseño de informes.
2. Diseño de informes.
3. Elaboración de gráficos.
4. Desarrollo de aplicaciones con informes.

Objetivos de aprendizaje

1. Conocer el concepto de informe, sus diferentes tipos y las herramientas para su elaboración.
2. Elaborar informes con tablas de datos, indicadores y otros elementos.
3. Diseñar gráficos de diferentes tipos para incluirlos en un informe.
4. Desarrollar aplicaciones que incluyan informes.

Alternativas en la creación de nuevos componentes

La mayoría de las organizaciones necesitan información como base para los procesos de toma de decisiones. Una de las formas más habituales de ofrecer esa información es por medio de informes, que pueden estar integrados en aplicaciones o consultarse mediante herramientas especializadas.

En este apartado conoceremos los distintos tipos de informes que podemos encontrar, los orígenes de los datos de donde puede provenir la información y las herramientas disponibles para su elaboración.

1. Introducción al diseño de informes

En cualquier organización, desde una pequeña startup hasta una gran empresa tecnológica, **las decisiones se toman a partir de datos**. Ventas, usuarios activos, pedidos entregados, errores en una aplicación… todo genera información que hay que **entender y presentar de forma clara**.

El problema no suele ser la falta de datos, sino justo lo contrario: **hay demasiados**. Por eso, uno de los retos más importantes en el desarrollo de software es **transformar los datos en información útil** para las personas que toman decisiones.

Aquí es donde entran en juego los **informes**.

¿Qué es la inteligencia empresarial?

Todo lo relacionado con convertir datos en conocimiento para apoyar la toma de decisiones se engloba dentro de la **inteligencia empresarial o de negocios**, conocida como **Business Intelligence (BI)**.

Hoy en día, la BI no es algo exclusivo de grandes empresas. Muchas aplicaciones que usáis a diario incluyen BI sin que os deis cuenta:

- Paneles de estadísticas en apps móviles
- Gráficos de uso en plataformas web
- Informes automáticos enviados por correo
- Dashboards que se actualizan en tiempo real

Como futuros desarrolladores y desarrolladoras de aplicaciones, **vais a crear sistemas que no solo funcionen, sino que informen**.

El informe como parte de una aplicación

Un **informe** es un medio para presentar información relevante en un formato concreto y comprensible.

Puede ser:

- Un documento PDF
- Una tabla en una aplicación web
- Un gráfico interactivo
- Un panel visual con indicadores

En la práctica actual, los informes **suelen estar integrados dentro de las propias aplicaciones**, como ocurre en:

- Sistemas ERP
- CRM
- Aplicaciones web de gestión
- Plataformas de análisis de datos

También existen herramientas especializadas para consultar informes, tanto técnicas como orientadas a usuarios finales.

IMPORTANTE

El fichero `DI_U05_A01_01.html`, muestra un ejemplo de informe creado con la librería que utilizaremos durante la unidad. Conviene revisarlo para tener una idea clara del resultado final que se espera.

DI_U05_A01_01.html
Text File
8.3 MB

1.1. Tipos de informes

No todos los informes sirven para lo mismo ni para el mismo tipo de usuario. Según el grado de control que tenga la persona que los consulta, podemos distinguir varios tipos.

Informes predefinidos

Son informes con una **estructura fija**, diseñada previamente por quien desarrolla la aplicación.

El usuario no puede modificar su contenido ni su formato.

Suelen utilizarse cuando:

- La información es siempre la misma
- Se quiere asegurar un formato estándar
- El informe se imprime o se envía por correo

Ejemplo:

Un informe mensual con las ventas del último mes para cada agente comercial.

Informes configurables

Permiten al usuario **ajustar ciertos parámetros**, aunque la estructura general sigue estando definida por el desarrollador.

Normalmente el usuario puede:

- Cambiar fechas
- Aplicar filtros
- Seleccionar algún criterio de agrupación

Este tipo de informe es muy habitual en aplicaciones actuales, porque ofrece flexibilidad sin complicar demasiado la interfaz.

Ejemplo:

Un informe de ventas donde el usuario puede elegir el periodo de tiempo (una semana, un mes, un año…).

Informes personalizados

En este caso, el usuario tiene un **alto grado de control** sobre lo que quiere ver y cómo quiere verlo.

Puede decidir:

- Qué datos mostrar
- Qué métricas le interesan
- Cómo organizar la información
- Si prefiere tablas, gráficos o ambos

Este tipo de informes suelen crearse con herramientas pensadas para usuarios finales, aunque detrás haya mucho trabajo de desarrollo.

Ejemplo:

Un informe de ventas donde el usuario elige:

- El periodo de tiempo
- El comercial
- La métrica (número de ventas, importe total, media…)
- El formato de visualización

Cuadros de mando (dashboards)

Los **cuadros de mando** son un tipo especial de informe orientado a la **visualización rápida de indicadores clave**.

Su objetivo no es mostrar todos los datos, sino **resumir lo importante de un vistazo**:

- Gráficos
- Indicadores numéricos
- Comparativas
- Tendencias

Son muy habituales en puestos de responsabilidad y en aplicaciones modernas.

Ejemplo:

Un dashboard que muestre:

- Total de ventas del año
- Comparación con el año anterior

- Evolución de ventas en los últimos meses

1.2. Orígenes de datos

Cuando diseñamos un informe, una de las primeras preguntas que debemos hacernos es muy clara:

¿De dónde salen los datos que voy a mostrar?

Un informe no "inventa" información. Siempre se apoya en datos que provienen de algún origen, y como desarrolladores es fundamental **conocer esos orígenes y saber trabajar con ellos**.

Hoy en día, las herramientas de generación de informes permiten trabajar con datos procedentes de múltiples fuentes, algo muy habitual en aplicaciones modernas.

Ficheros de datos

Son uno de los orígenes más sencillos y comunes, sobre todo en proyectos pequeños o en fases iniciales.

Los formatos más habituales son:

- **CSV**: muy usado para exportaciones e intercambios de datos.
- **JSON**: muy frecuente en APIs y aplicaciones web.
- **XML**: menos común hoy en día, pero aún presente en algunos sistemas.
- Ficheros de hojas de cálculo (Excel, LibreOffice Calc).

Este tipo de datos es muy útil para:

- Pruebas
- Importaciones puntuales
- Prototipos
- Ejercicios prácticos en clase

Bases de datos

Es el origen más habitual en aplicaciones reales.

En la mayoría de empresas, los informes se generan a partir de **bases de datos de la organización**, donde se almacena la información del día a día.

Predominan las bases de datos relacionales, como:

- **MySQL**
- **SQL Server**
- **Oracle**
- **PostgreSQL** (muy presente en proyectos actuales)

También es cada vez más común encontrar bases de datos **NoSQL**, como:

- **MongoDB**
- **Firebase Firestore**
- **Cassandra**

Este tipo de origen es clave porque:

- Permite informes actualizados
- Facilita el filtrado y la agregación de datos
- Está directamente conectado con las aplicaciones que desarrolláis

Almacenes de datos (Data Warehouse)

Un **data warehouse** es un repositorio de datos pensado específicamente para el **análisis**, no para el uso diario de la aplicación.

Suele:

- Recoger datos de varias fuentes
- Organizar la información de forma histórica
- Facilitar consultas complejas

Es muy habitual en empresas medianas y grandes, donde se analizan datos a largo plazo: ventas, comportamiento de clientes, rendimiento de procesos, etc.

Big Data y analítica avanzada

En entornos con grandes volúmenes de datos, entramos en el terreno del **big data**.

Algunas herramientas actuales permiten generar informes a partir de:

- Plataformas distribuidas
- Procesamiento masivo de datos
- Fuentes en tiempo casi real

Ejemplos de tecnologías relacionadas:

- Apache Hadoop
- Apache Spark
- Sistemas de streaming de datos

Este tipo de origen es habitual en:

- Grandes plataformas web
- Aplicaciones con muchos usuarios
- Sistemas de monitorización y análisis avanzado

Datos en la nube

Hoy en día, casi todos estos orígenes pueden estar en la nube.

Las plataformas de nube pública ofrecen servicios para:

- Almacenar datos
- Gestionar bases de datos
- Analizar información
- Generar informes

Algunas de las más utilizadas son:

- Amazon AWS
- Microsoft Azure
- Google Cloud

1.3. Herramientas para la elaboración de informes

Una vez sabemos **de dónde vienen los datos**, toca decidir **con qué herramienta vamos a crear los informes**.

Esta decisión depende de:

- El tipo de aplicación
- El perfil de usuario
- El nivel de personalización necesario
- La tecnología que estemos usando

Librerías de informes

Una opción muy habitual en desarrollo es usar **librerías que generan informes directamente desde el código**.

Ventajas:

- Control total sobre el informe
- Integración directa con la aplicación
- Automatización completa

En esta unidad utilizaremos esta alternativa, trabajando con **Python** y una librería de generación de informes.

Herramientas visuales de diseño de informes

Otra opción es utilizar herramientas visuales que permiten diseñar informes sin necesidad de programar (o con muy poca programación).

Existen dos grandes tipos:

- Herramientas orientadas a desarrolladores
  - SAP Crystal Reports
  - TIBCO JasperReports
- Herramientas orientadas a usuarios finales
  - Microsoft Power BI
  - MicroStrategy
  - Tableau (muy extendida en la actualidad)

Estas herramientas permiten crear informes interactivos, paneles visuales y cuadros de mando de forma rápida.

Servicios de informes en la nube

Muchas plataformas cloud incluyen servicios específicos para análisis de datos y BI.

Algunos ejemplos actuales:

- Amazon QuickSight (AWS)
- Looker (Google Cloud)
- Power BI Service (Microsoft)

Estas soluciones permiten conectar directamente con bases de datos y servicios en la nube, facilitando la creación de informes sin necesidad de desplegar infraestructura propia.

2. Diseño de informes

En este apartado aprenderemos a crear **informes en formato HTML** utilizando **Datapane**, una librería de Python que permite generar informes visuales a partir de datos, incorporando elementos como tablas y, más adelante, gráficos.

Estos informes pueden:

- Guardarse como ficheros HTML locales.
- Abrirse en un navegador web.
- Compartirse o integrarse en aplicaciones mayores.

Desde el punto de vista de **Desarrollo de Interfaces**, Datapane nos permite centrarnos en:

- Cómo se presenta la información al usuario.
- Qué opciones de interacción tiene (ordenar, filtrar, explorar datos).
- La experiencia de uso de informes y paneles de datos.

2.1. ¿Qué es Datapane?

Datapane es una librería de Python que permite crear informes visuales de forma sencilla, a partir de datos estructurados.

Características principales:

- Genera informes en **HTML**.
- Trabaja directamente con **DataFrames de Pandas**.
- Incluye componentes visuales listos para usar (tablas, tablas interactivas, gráficos…).
- Está distribuida con licencia **Apache 2.0** (software libre).

Datapane permite:

- Guardar informes de forma local.
- Subirlos a su plataforma web (opción que veremos solo a nivel experimental).

> **IMPORTANTE**
>
> Durante el módulo trabajaremos principalmente con **informes locales**, ya que son más fáciles de integrar en proyectos y no requieren cuenta externa.

Para poder usar Datapane, debemos instalar la librería:

Una vez instalada, podremos importarla en nuestros programas como cualquier otra librería.

2.2. Preparación de los datos

El origen de los datos

En una aplicación real, los datos pueden venir de:

- Ficheros (CSV, JSON…)
- Bases de datos
- APIs
- Entradas del usuario

Sin embargo, **Datapane trabaja siempre con un único tipo de origen de datos**:

👉 **DataFrames de la librería Pandas**

Esto simplifica el diseño de los informes, ya que todos los componentes esperan datos con la misma estructura.

¿Qué es Pandas?

Pandas es una de las librerías más usadas en Python para trabajar con datos.

Su estructura principal es el **DataFrame**, que podemos entender como:

- Una tabla de filas y columnas.
- Similar a una tabla de una base de datos relacional.
- Muy cómoda para ordenar, filtrar y transformar datos.

💡 **¿Sabías que…?**

Muchas herramientas de análisis y visualización de datos en Python trabajan directamente con DataFrames, por lo que aprender Pandas es clave para este tipo de aplicaciones.

Cargar datos desde un fichero CSV

Veamos un ejemplo típico: crear un DataFrame a partir de un fichero CSV.

DI_U05_A02_02.csv
Text File
2.1 KB

```
import pandas as pd
import datapane as dp

fichero_csv = "DI_U05_A02_02.csv"
df = pd.read_csv(fichero_csv)
```

En este ejemplo:

- Importamos Pandas y Datapane.
- Leemos un fichero CSV.
- Guardamos los datos en un DataFrame llamado `df`.

El fichero CSV contiene:

- Nombre del comercial
- Mes
- Número de unidades vendidas
- Importe total de ventas

Este DataFrame será la base de nuestros informes.

2.3. Informes con tablas

La tabla como elemento de interfaz

En muchos informes, el elemento principal es la **tabla de datos**. Desde el punto de vista del usuario:

- Permite ver información detallada.
- Facilita comparaciones.
- Sirve como apoyo a la toma de decisiones.

Datapane ofrece **dos tipos de tablas**, con distinto nivel de interacción.

Tipos de tablas en Datapane

- **Table**

- Tabla estática.
- El usuario solo puede visualizar los datos.

- Todos los registros se muestran a la vez.

👉 Útil cuando:

- Queremos imprimir el informe.
- Vamos a exportarlo a PDF.
- No necesitamos interacción.

🔹 **DataTable**

- Tabla interactiva.
- Permite:
  - Ordenar por columnas.
  - Filtrar datos.
  - Desplazarse por filas.

👉 Más cercana a una **interfaz interactiva**, ideal para exploración de datos.

Creación de un informe con tablas

A partir del DataFrame anterior, podemos crear un informe con ambos tipos de tabla.

```
table = dp.Table(df)
data_table = dp.DataTable(df)

report = dp.Report(table, data_table)
report.save(path="DI_U05_A02_03.html", open=True)
```

Qué hace cada parte:

1. Se crea una tabla estática.
2. Se crea una tabla interactiva.
3. Se genera un informe que contiene ambas.
4. Se guarda el informe como HTML y se abre en el navegador.

Desde DI, esto equivale a **construir una interfaz de visualización sin programar HTML o JavaScript directamente**.

Ordenación y filtrado de datos

Ordenación

En las **DataTable**, el usuario puede:

- Ordenar por cualquier columna.
- Cambiar entre orden ascendente y descendente.

El icono junto al nombre de la columna indica:

- El tipo de dato.
- El estado de la ordenación.

Filtrado

El filtrado permite mostrar solo los datos que cumplen una condición:

- Por ejemplo, ventas mayores de una cantidad.
- Meses concretos.
- Un comercial específico.

Características del filtrado:

- Se pueden aplicar varios filtros a la vez.
- Los operadores dependen del tipo de dato.
- Los filtros activos se indican visualmente.

Para eliminar un filtro, se edita y se establece la condición **None**.

> **IMPORTANTE**
>
> La ordenación y el filtrado de Datapane actúan **sobre la visualización**.
>
> Si queremos que los datos ya estén filtrados u ordenados antes de generar el informe, debemos hacerlo con **Pandas**.

2.3. Informes con indicadores

¿Qué son los indicadores en un informe?

En muchos informes no interesa mostrar todos los datos en forma de tabla.

En su lugar, el usuario necesita **valores clave**, claros y rápidos de interpretar.

A estos valores los llamamos **indicadores**.

Ejemplos:

- Ventas totales del último mes.
- Número de clientes activos.
- Importe medio por operación.
- Diferencia respecto al mes anterior.

En visualización de datos, estos indicadores también se conocen como:

- **Grandes números**
- **Big numbers**
- **KPIs (Key Performance Indicators)**

📌 Desde el punto de vista del **Desarrollo de Interfaces**, un indicador:

- Resume información compleja.
- Reduce la carga cognitiva del usuario.
- Permite tomar decisiones de forma rápida.

Cálculo de valores agregados con Pandas

¿Por qué no calcula Datapane los indicadores?

Datapane **no calcula datos**, solo los muestra.

Por tanto, **los indicadores deben calcularse previamente** en el código, usando Pandas.

Ejemplo: cálculo de ventas por mes

Supongamos que tenemos un DataFrame con ventas mensuales.

Queremos calcular el total de unidades vendidas en noviembre y diciembre.

```
datos_diciembre = df[df['Mes'] == 'Diciembre']
unidades_diciembre = datos_diciembre['Unidades'].sum()

datos_noviembre = df[df['Mes'] == 'Noviembre']
unidades_noviembre = datos_noviembre['Unidades'].sum()
```

Qué ocurre en este código:

1. Se filtran las filas por mes.
2. Se selecciona la columna `Unidades`.
3. Se suman todos sus valores.

> 💡 **Idea clave**
>
> Primero seleccionamos los datos → después calculamos → por último los mostramos en el informe.

Funciones de agregación más usadas en Pandas

Pandas incluye muchas funciones para crear indicadores. Las más habituales son:

Función	Descripción
`sum()`	Suma todos los valores
`count()`	Cuenta el número de valores
`mean()`	Calcula la media
`min()`	Obtiene el valor mínimo
`max()`	Obtiene el valor máximo

Estas funciones son la base de la mayoría de indicadores que veremos en informes reales.

Creación de informes con indicadores (BigNumber)

El componente BigNumber

Datapane incluye el componente **BigNumber**, pensado para mostrar:

- Un **valor principal destacado**.
- Opcionalmente, una **comparación con un valor anterior**.

Es uno de los componentes más habituales en **paneles de control y cuadros de mando**, ya que permite interpretar un dato de forma rápida.

Parámetros principales de BigNumber

Al crear un `BigNumber` podemos indicar:

- `heading`: texto que describe el indicador.
- `value`: valor actual que se quiere destacar.
- `change`: diferencia respecto a un valor anterior (opcional).
- `is_upward_change`: indica si el cambio debe interpretarse como una mejora (`True`) o un empeoramiento (`False`).

⚠️ Si se usa `change`, es obligatorio indicar `is_upward_change`.

📌 **Importante**

El componente **no muestra una flecha como icono**.

El indicador visual de mejora o empeoramiento se representa mediante **el color del cambio**:

- Verde → mejora
- Rojo → empeora

Ejemplo completo de indicador

```
unidades = dp.BigNumber(
    heading='Unidades totales en diciembre',
    value=unidades_diciembre,
    change=unidades_diciembre - unidades_noviembre,
    is_upward_change=unidades_diciembre > unidades_noviembre
)

report = dp.Report(unidades)
report.save(path='DI_U05_A02_05.html', open=True)
```

Este indicador muestra:

- El total de unidades vendidas en diciembre.
- La diferencia respecto a noviembre.
- Un **indicador visual por color** que permite identificar si el resultado mejora o empeora.

2.4. Otros componentes de los informes

Además de tablas e indicadores, Datapane permite añadir otros elementos que mejoran la interfaz del informe.

Texto (Text)

Permite incluir texto explicativo en formato **Markdown**.

Usos habituales:

- Títulos.
- Descripciones.
- Interpretación de resultados.
- Instrucciones para el usuario.

```
1  texto = dp.Text("**Puedes descargar el fichero con los datos del
```

💡 Markdown es un lenguaje sencillo para dar formato a texto sin escribir HTML.

HTML

El componente HTML permite incluir bloques de código HTML directamente en el informe.

Características:

- Permite usar CSS para estilos.
- No permite JavaScript.
- Útil para personalizar títulos o cabeceras.

```
titulo = dp.HTML(
    '<p style="font-size:30px; text-align:center; color:#ffffff;
)
```

Multimedia (Media)

Permite incluir:

- Imágenes
- Vídeos
- Audio

```
imagen = dp.Media(file='DI_U05_A02_07.png')
```

Muy útil para:

- Logotipos.
- Gráficos exportados.
- Elementos visuales de apoyo.

Ficheros adjuntos (Attachment)

Permite adjuntar archivos descargables al informe:

- CSV
- PDF

- Hojas de cálculo

```
fichero = dp.Attachment(file='DI_U05_A02_02.csv')
```

Desde la interfaz, el usuario podrá descargar el fichero directamente.

Informe completo combinando componentes

Ejemplo de informe final con varios elementos:

```
report = dp.Report(
    imagen,
    titulo,
    unidades,
    texto,
    fichero
)

report.save(path='DI_U05_A02_08.html', open=True)
```

Este informe combina:

- Elementos visuales.
- Indicadores clave.
- Texto explicativo.
- Recursos descargables.

📌 Esto se acerca mucho a un **panel de información real** usado en empresas.

3. Elaboración de gráficos

El papel de los gráficos en una interfaz

Los gráficos son uno de los elementos más utilizados en informes y paneles de datos, ya que permiten:

- Visualizar grandes cantidades de información de un vistazo.
- Detectar tendencias y patrones.

- Comparar valores de forma rápida.

En muchos casos, un gráfico comunica mejor la información que una tabla, sobre todo cuando el usuario necesita **interpretar datos**, no solo leerlos.

Desde el punto de vista del diseño de interfaces (DI), un buen gráfico debe:

- Ser claro.
- Representar exactamente los datos que se quieren mostrar.
- Evitar elementos visuales que no aporten información.

Gráficos en Datapane

Datapane **no crea gráficos por sí mismo**.

Su función es **mostrar gráficos que ya han sido creados con otras librerías de Python**.

Para ello utiliza el componente `Plot`, que recibe un gráfico previamente generado.

El flujo habitual de trabajo es el siguiente:

```
Datos (DataFrame)
   ↓
Librería de gráficos (Matplotlib)
   ↓
dp.Plot → integración en Datapane
```

Datapane es compatible con varias librerías de gráficos, entre ellas:

- Matplotlib
- Plotly
- Altair
- Bokeh

En este módulo utilizaremos **Matplotlib**, ya que:

- Es la librería de gráficos más utilizada en Python.
- Se integra directamente con Pandas.
- Permite crear sin dificultad los gráficos habituales de un informe.

⚠ **IMPORTANTE**

Matplotlib no forma parte de la biblioteca estándar de Python, pero se considera el estándar de facto para la creación de gráficos. Se instala mediante el siguiente comando:

```
pip install matplotlib
```

Para comprobar que ha quedado correctamente instalado, crea un fichero de prueba en VS Code con el siguiente código y ejecútalo:

```
import matplotlib.pyplot as plt

# Datos de ejemplo
x = [1, 2, 3, 4, 5]
y = [2, 4, 3, 5, 4]

# Crear el gráfico
plt.plot(x, y)
plt.title("Prueba de Matplotlib")
plt.xlabel("Eje X")
plt.ylabel("Eje Y")

# Mostrar el gráfico
plt.show()
```

Si todo está correcto:

- Se abrirá una **ventana nueva** con un gráfico de líneas.
- No aparecerán mensajes de error en la terminal.

3.1. Gráficos de líneas

El gráfico de líneas se utiliza para:

- Representar la evolución de una variable.
- Mostrar cambios a lo largo del tiempo.
- Analizar tendencias.

Es habitual cuando una de las variables es:

- El tiempo (meses, años, días).

- Una secuencia ordenada.

Este tipo de gráfico ayuda al usuario a detectar picos, caídas y tendencias de forma inmediata.

Ejemplo: evolución mensual de ventas

A partir de un fichero CSV con datos de ventas, queremos representar:

- El total de unidades vendidas en cada mes.
- La evolución de esas ventas a lo largo del año.

El gráfico se integrará en un **informe Datapane**, que se generará como un archivo HTML.

El origen de los datos es el fichero:

```
DI_U05_A02_02.csv
```

```
import pandas as pd
import datapane as dp

# Cargar los datos desde el CSV
df = pd.read_csv("DI_U05_A02_02.csv")

# Agrupar los datos por mes y sumar las unidades vendidas
ventas_mes = df.groupby(["Mes"], sort=False).sum()

# Crear el gráfico de líneas (Matplotlib a través de Pandas)
grafico_matplotlib = ventas_mes.plot(y="Unidades")

# Adaptar el gráfico para Datapane
grafico_datapane = dp.Plot(grafico_matplotlib)

reporte = dp.Report(
    dp.Text("# Informe de ventas"),
    grafico_datapane
)

reporte.save("informe_ventas.html", open=True)
```

Qué ocurre en cada paso

1. **Importación de librerías**
   - `pandas` se utiliza para trabajar con los datos.
   - `datapane` permite generar el informe final.
   - **Nota:** Pandas se encarga de crear los gráficos usando Matplotlib por debajo, por lo que no es necesario importar Matplotlib de forma directa.
2. **Carga del CSV**

```
df = pd.read_csv("DI_U05_A02_02.csv")
```

   - Se leen los datos desde el fichero CSV.
   - El resultado es un DataFrame con la información de ventas.
3. **Agrupación de datos**

```
ventas_mes = df.groupby(["Mes"], sort=False).sum()
```

   - Se agrupan los registros por mes.
   - Se suman las columnas numéricas.
   - Se obtiene un nuevo DataFrame donde:
     - Cada fila representa un mes.
     - El mes pasa a ser el índice del DataFrame.
4. **Creación del gráfico de líneas**

```
grafico_matplotlib = ventas_mes.plot(y="Unidades")
```

   - Se crea un gráfico de líneas.
   - El eje horizontal representa los meses.
   - El eje vertical representa el total de unidades vendidas.
   - El gráfico se genera mediante Matplotlib, aunque se invoque desde Pandas.
5. **Adaptación del gráfico a Datapane**

```
grafico_datapane = dp.Plot(grafico_matplotlib)
```

   - El gráfico no se muestra todavía.

- Se prepara para poder insertarse dentro de un informe Datapane.

6. **Creación del informe**

```
reporte = dp.Report(
    dp.Text("# Informe de ventas"),
    grafico_datapane
)
```

- Se crea un informe Datapane.
- Se añade un texto de título.
- Se añade el gráfico de líneas.

7. **Generación del informe final**

```
reporte.save("informe_ventas.html", open=True)
```

- Se genera un archivo HTML con el informe.
- El informe se abre automáticamente en el navegador.

3.2. Gráficos de barras

Los gráficos de barras son adecuados cuando:

- Queremos comparar valores entre categorías.
- La variable del eje horizontal es discreta o cualitativa.

Ejemplos:

- Ventas por vendedor.
- Resultados por departamento.
- Importes por producto.

Ejemplo: ventas por vendedor

A partir de un fichero CSV con datos de ventas, queremos representar:

- El **importe total vendido por cada vendedor**.
- La comparación entre vendedores de forma visual.

El origen de los datos es el fichero:

```
DI_U05_A02_02.csv
```

El gráfico se integrará en un **informe Datapane**, que se generará como un archivo HTML.

```python
import pandas as pd
import datapane as dp

# Cargar los datos desde el CSV
df = pd.read_csv("DI_U05_A02_02.csv")

# Agrupar los datos por vendedor y sumar los importes
ventas_vendedor = df.groupby(["Nombre"]).sum()

# Crear el gráfico de barras (Pandas usa Matplotlib internamente
grafico_matplotlib = ventas_vendedor.plot.bar(y="Importe (€)")

# Adaptar el gráfico para Datapane
grafico_datapane = dp.Plot(grafico_matplotlib)

# Crear el informe Datapane
reporte = dp.Report(
    dp.Text("# Ventas por vendedor"),
    grafico_datapane
)

# Guardar y abrir el informe
reporte.save("informe_ventas_vendedor.html", open=True)
```

Qué ocurre en cada parte del programa

1. **Carga del CSV**
   - Se leen los datos desde el fichero `DI_U05_A02_02.csv`.
   - Los datos se almacenan en un DataFrame.
2. **Agrupación por vendedor**

```
ventas_vendedor = df.groupby(["Nombre"]).sum()
```

- Se agrupan los registros por nombre del vendedor.
- Se suman las columnas numéricas.
- El resultado es un nuevo DataFrame donde:
  - Cada fila representa un vendedor.
  - Los valores corresponden a sus totales de ventas.

3. **Creación del gráfico de barras**

```
grafico_matplotlib = ventas_vendedor.plot.bar(y="Importe (€)
```

- Cada barra representa a un vendedor.
- La altura de la barra indica el importe total vendido.
- El eje horizontal muestra las categorías (vendedores).
- El eje vertical muestra los valores numéricos.

4. **Adaptación a Datapane**

```
grafico_datapane = dp.Plot(grafico_matplotlib)
```

- El gráfico se prepara para poder incluirse en un informe.
- Todavía no se muestra en pantalla.

5. **Creación del informe**

```
reporte = dp.Report(
    dp.Text("# Ventas por vendedor"),
    grafico_datapane
)
```

- Se crea un informe Datapane.
- Se añade un título.
- Se inserta el gráfico de barras.

6. **Generación del informe final**

```
reporte.save("informe_ventas_vendedor.html", open=True)
```

- Se genera un archivo HTML con el informe.
- El informe se abre automáticamente en el navegador.

3.3. Gráficos de sectores (tarta)

Los gráficos de sectores se utilizan para:

- Mostrar cómo se reparte un total entre varias categorías.
- Representar porcentajes o proporciones.

Son útiles cuando:

- Hay **pocas categorías** (por ejemplo, 3 a 6).
- Queremos ver rápidamente **qué parte del total corresponde a cada categoría**.

⚠️ **Ojo**

Si hay demasiadas categorías, el gráfico se vuelve difícil de interpretar. En esos casos suele ser mejor un **gráfico de barras**.

Ejemplo: reparto de unidades vendidas

Queremos representar qué porcentaje del total de unidades corresponde a cada vendedor.

El resultado se integrará en un **informe Datapane** (archivo HTML).

```python
import pandas as pd
import datapane as dp

# Cargar los datos desde el CSV
df = pd.read_csv("DI_U05_A02_02.csv")

# Agrupar por vendedor y sumar (para obtener los totales)
ventas_vendedor = df.groupby(["Nombre"]).sum()

# Crear el gráfico de sectores (tarta)
grafico_matplotlib = ventas_vendedor.plot.pie(
    y="Unidades",
    legend=False,
    ylabel=""
)

# Adaptar el gráfico para Datapane
grafico_datapane = dp.Plot(grafico_matplotlib)

# Crear el informe Datapane
reporte = dp.Report(
    dp.Text("# Reparto de unidades vendidas por vendedor"),
    grafico_datapane
)

# Guardar y abrir el informe
reporte.save("informe_reparto_unidades.html", open=True)
```

Qué ocurre en cada parte del programa

1. **Carga del CSV**

```python
df = pd.read_csv("DI_U05_A02_02.csv")
```

Se leen los datos del fichero y se guardan en un DataFrame.

2. **Agrupación por vendedor**

```python
ventas_vendedor = df.groupby(["Nombre"]).sum()
```

- Se agrupan los registros por vendedor.
- Se suman las columnas numéricas.

- Se obtiene un DataFrame donde cada fila es un vendedor con sus totales.

3. **Creación del gráfico de sectores**

```
grafico_matplotlib = ventas_vendedor.plot.pie(
    y="Unidades",
    legend=False,
    ylabel=""
)
```

- Cada sector representa a un vendedor.
- El tamaño del sector indica su proporción sobre el total de unidades.
- `legend=False` evita mostrar la leyenda si no aporta.
- `ylabel=""` elimina una etiqueta que no tiene utilidad en este tipo de gráfico.

4. **Adaptación a Datapane**

```
grafico_datapane = dp.Plot(grafico_matplotlib)
```

El gráfico se prepara para insertarlo en un informe Datapane.

5. **Creación y guardado del informe**

```
reporte.save("informe_reparto_unidades.html", open=True)
```

Se genera el HTML y se abre en el navegador.

3.4. Integración de gráficos en informes Datapane

En Datapane, los gráficos no se muestran "en una ventana" como ocurre en otros entornos.

En su lugar, se insertan dentro de un **informe** (normalmente un HTML) junto con otros elementos como texto, tablas e indicadores.

El flujo típico es:

1. Cargar los datos en un DataFrame (`read_csv`).
2. Preparar los datos (agrupaciones, sumas…).

3. Crear el gráfico con `df.plot()`.
4. Adaptarlo con `dp.Plot`.
5. Construir el informe con `dp.Report`.
6. Guardar el informe (`save`) y abrirlo en el navegador.

Ejemplo completo: panel con gráfico + tabla (ventas mensuales)

En este ejemplo creamos un informe con:

- Un título.
- Una breve explicación.
- Un gráfico de líneas (unidades vendidas por mes).
- Una tabla con los datos agregados (para ver el detalle).

El origen de los datos es:

```
DI_U05_A02_02.csv
```

```python
import pandas as pd
import datapane as dp

# Cargar los datos desde el CSV
df = pd.read_csv("DI_U05_A02_02.csv")

# Preparar datos: unidades totales por mes
ventas_mes = df.groupby(["Mes"], sort=False).sum()

# Crear el gráfico (Pandas usa Matplotlib internamente)
grafico_matplotlib = ventas_mes.plot(y="Unidades")

# Adaptar gráfico a Datapane
bloque_grafico = dp.Plot(grafico_matplotlib)

# Crear una tabla con los datos agregados
tabla_resumen = dp.DataTable(ventas_mes)

# Construir el informe completo
reporte = dp.Report(
    dp.Text("# Panel de ventas"),
    dp.Text("Este panel muestra la evolución mensual de las unic
    bloque_grafico,
    dp.Text("## Detalle por mes"),
    tabla_resumen
)

# Guardar y abrir el informe
reporte.save("panel_ventas_mensual.html", open=True)
```

Qué aporta este diseño (en términos de DI)

- **El gráfico** sirve para interpretar rápidamente la tendencia (subidas/bajadas).
- **La tabla** permite consultar el dato exacto cuando el usuario necesita confirmar valores.
- El texto guía la lectura: qué se está viendo y por qué.

Un panel suele combinar "vista rápida" (gráfico) con "detalle" (tabla). Así el usuario decide rápido, pero tiene respaldo numérico.

Buenas prácticas de diseño de gráficos

Al diseñar gráficos para una interfaz:

- Evita gráficos innecesarios.
- No satures la pantalla con demasiados elementos.
- Usa el tipo de gráfico adecuado al dato.
- Prioriza claridad frente a estética.

4. Desarrollo de aplicaciones con informes

De informe a aplicación

Hasta ahora hemos aprendido a **diseñar informes** con Datapane.

El siguiente paso lógico es **integrarlos en una aplicación**, de forma que el usuario pueda acceder a ellos desde una interfaz gráfica.

Un aspecto clave es que Datapane genera los informes en **formato HTML**, lo que nos ofrece varias ventajas:

- Se pueden abrir en cualquier navegador.
- Se pueden incrustar dentro de aplicaciones.
- Se pueden desplegar en servidores web.

4.1. Integración de informes en una aplicación Qt

Cuando trabajamos con aplicaciones Qt (por ejemplo, usando PyQt o PySide), tenemos **dos formas principales** de integrar un informe generado con Datapane:

1. Abrir el informe en el navegador predeterminado del sistema.
2. Mostrar el informe dentro de la propia aplicación.

Cada opción tiene ventajas y se usa en contextos distintos.

Abrir el informe en el navegador predeterminado

Abrir el informe en el navegador es adecuado cuando:

- El informe es independiente de la aplicación.
- Queremos reutilizar el navegador del sistema.
- No necesitamos incrustar el contenido en la interfaz.

Desde el punto de vista del usuario:

- El informe se abre como una página web.
- La aplicación actúa como lanzador.

Uso de QDesktopServices

Qt proporciona la clase **QDesktopServices**, que permite interactuar con servicios del sistema, como el navegador web.

La idea es:

1. Obtener la ruta absoluta del fichero HTML.
2. Convertirla en un objeto `QUrl`.
3. Abrirla con el navegador predeterminado.

```
ruta_absoluta = QDir().absoluteFilePath("./DI_U05_A03_11.html")
QDesktopServices.openUrl(QUrl.fromLocalFile(ruta_absoluta))
```

💡 **Ojo con esto**

Siempre es recomendable trabajar con rutas absolutas para evitar errores si cambia el directorio de ejecución.

Esta opción:

- Simplifica la aplicación.
- Reduce carga en la interfaz.
- Es habitual en aplicaciones que generan informes externos (por ejemplo, informes financieros o académicos).

Incrustar el informe en la aplicación (QWebEngineView)

QWebEngineView es un componente de Qt que permite mostrar contenido web dentro de una ventana de la aplicación.

Características:

- Muestra HTML local o remoto.
- Está basado en el motor Chromium.
- Permite una integración completa del informe en la interfaz.

Esta opción es más adecuada cuando:

- El informe forma parte central de la aplicación.
- Queremos una experiencia integrada.
- El usuario no debe salir de la aplicación.

Es habitual en:

- Aplicaciones de análisis.
- Herramientas de gestión.
- Paneles de control.

Ejemplo

```
view = QWebEngineView()
view.load(QUrl.fromLocalFile(ruta_absoluta))
```

Qué ocurre aquí:

1. Se crea el componente visual.
2. Se carga el informe HTML.
3. El informe se muestra dentro de la aplicación.

⚠ IMPORTANTE

La carga del contenido es **asíncrona**, por lo que Qt ofrece señales para controlar el proceso si fuera necesario (por ejemplo, mostrar un indicador de carga).

Comparativa entre ambas opciones

Opción	Ventaja principal	Cuándo usarla
Navegador externo	Simplicidad	Informes independientes
QWebEngineView	Integración total	Aplicaciones con informes integrados

Desde DI, la elección depende del **diseño de la experiencia de usuario**.

4.2. Despliegue de informes en un servidor web

Al ser ficheros HTML, los informes de Datapane pueden:

- Subirse a un servidor web.
- Consultarse desde distintas aplicaciones.
- Accederse desde una intranet o red corporativa.

Esto permite separar:

- Generación del informe.
- Consulta del informe.

El despliegue en servidor es adecuado cuando:

- Los informes no se generan en tiempo real.
- Se programan con antelación (por ejemplo, informes mensuales).
- Muchos usuarios consultan el mismo informe.

Uso desde aplicaciones Qt

Los mecanismos vistos antes siguen siendo válidos:

- Abrir el informe en el navegador.
- Cargar la URL en un QWebEngineView.

La única diferencia es que la URL:

- Apunta a un servidor web.
- No a un fichero local.

Despliegue en la nube

Los servicios en la nube facilitan el despliegue de contenido web estático, como los informes de Datapane.

Ejemplos habituales:

- Azure: Static Web Apps
- AWS: Amplify

Servicio online de Datapane

Datapane ofrece su propia plataforma online (datapane.com) que permite:

- Subir informes directamente desde Python.
- Publicarlos sin configurar servidores.

Para ello se utiliza el método `upload` del componente `Report`.

⚠️ Esta opción es interesante para pruebas y prototipos, pero no sustituye el aprendizaje de integración y despliegue en aplicaciones reales.