

**PROYECTO INTEGRADOR**

CODING

WITH AI

Contenido

[Proyecto integrador 3](#_Toc199414453)

[Descripción general: 3](#_Toc199414454)

[Estructura: 4](#_Toc199414455)

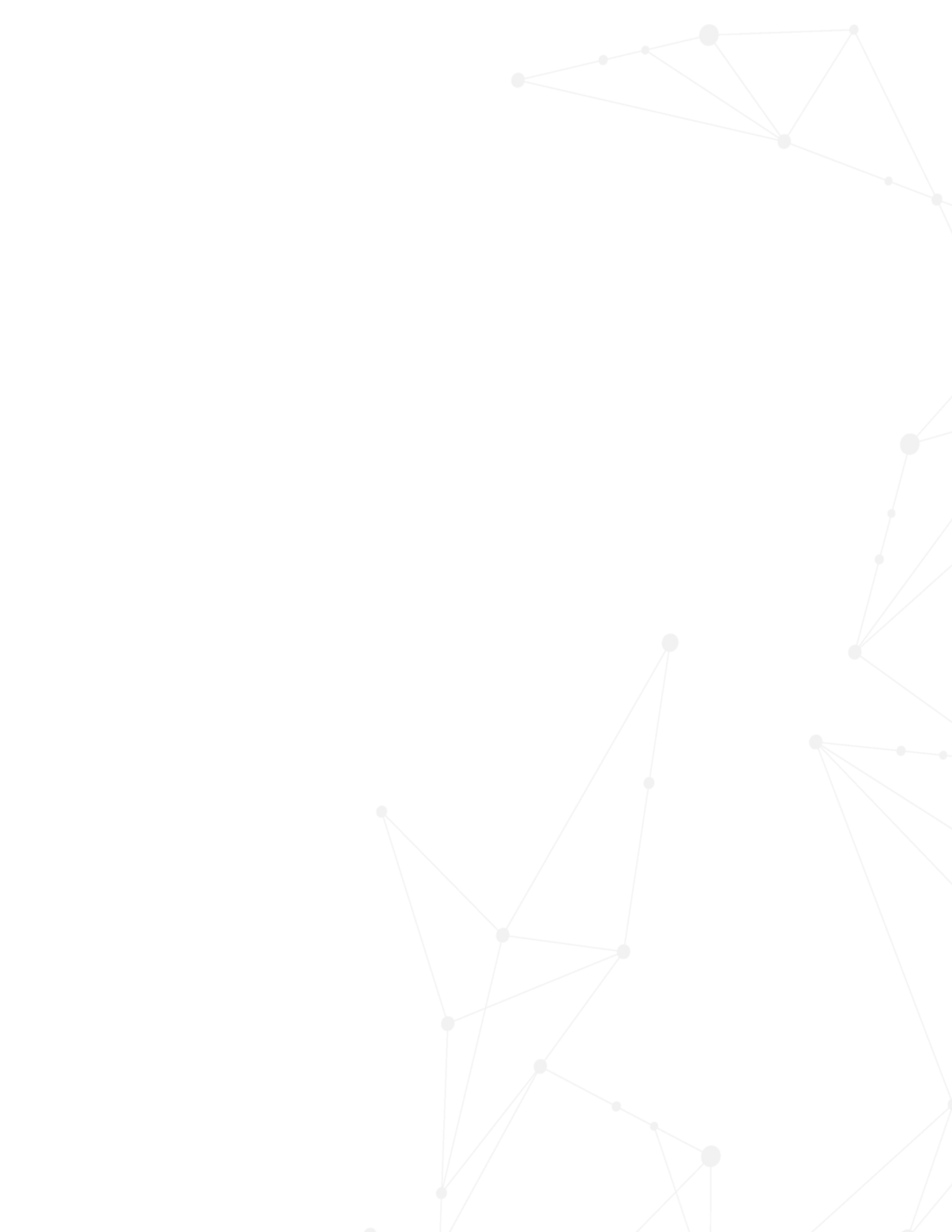
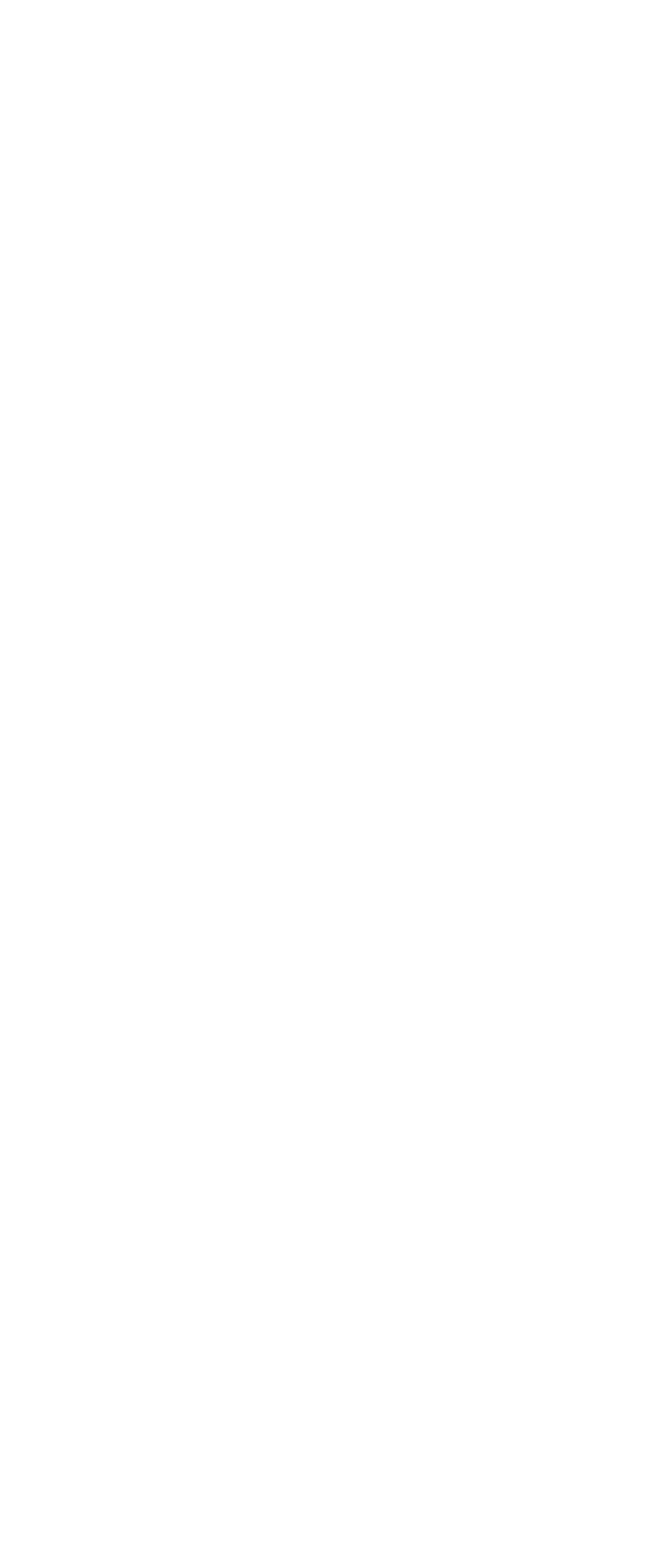
[Público objetivo 4](#_Toc199414456)

[Entidades principales: 4](#_Toc199414457)

[Tipos de relaciones en el diseño 5](#_Toc199414458)

[Estructura: 9](#_Toc199414459)

[Estructura: 18](#_Toc199414460)



# Proyecto integrador

Desarrollar una aplicación móvil (web) que integre los conocimientos adquiridos en cada módulo del curso. El proyecto final será una recopilación de los diferentes aprendizajes adquiridos durante el curso y los casos prácticos.

# Descripción general:

Consolidar el desarrollo de una aplicación de tipo e-commerce que refleje un entendimiento profundo de los temas del curso, incluyendo diseño y desarrollo de base de datos, desarrollo backend y frontend impulsado por IA, integración de modelos de IA, y prácticas de seguridad y optimización de

**Coding with AI:** Proyector integrador **01**

# Estructura:

**Planificación y diseño:**

Tomando como referencia lo aprendido en el Módulo 1, definir la idea de la aplicación, su propósito principal.

**GreenHub**

**Marketplace de productos de segunda mano, reutilizados y upcycled**, con un enfoque en la economía circular.

**Propósito principal:**

* Reducir el desperdicio masivo de recursos fomentando la compra-venta de productos usados en buen estado.
* Ofrecer una alternativa económica y ecológica al consumo tradicional.

## Público objetivo

Consumidores eco-conscientes (18-50 años) que buscan reducir su huella ambiental sin gastar mucho (ej.: millennials y Gen Z preocupados por el fast fashion o la obsolescencia programada).

Vendedores independientes: Tiendas de thrift, artesanos de upcycling, dueños de talleres de reparación (ej.: electrónica o muebles).

Empresas con excedentes: Marcas que quieran vender stock no vendido o devoluciones en lugar de destruirlo.

Diseñar un esquema inicial para la base de datos y la arquitectura general de la aplicación usando la IA para generar recomendaciones y/o sugerencias.

### **Entidades principales:**

| **Tabla** | **Descripción** |
| --- | --- |
| users | Almacena los datos de los usuarios |
| products | Información de productos en venta |
| orders | Compras realizadas por los usuarios |
| reviews | Opiniones y puntuaciones de productos |
| categories | Categorías para clasificar los productos |

 Un usuario puede vender **muchos productos** (1 a N).

 Un usuario puede hacer **muchas órdenes**.

 Un producto puede tener **muchas reviews**.

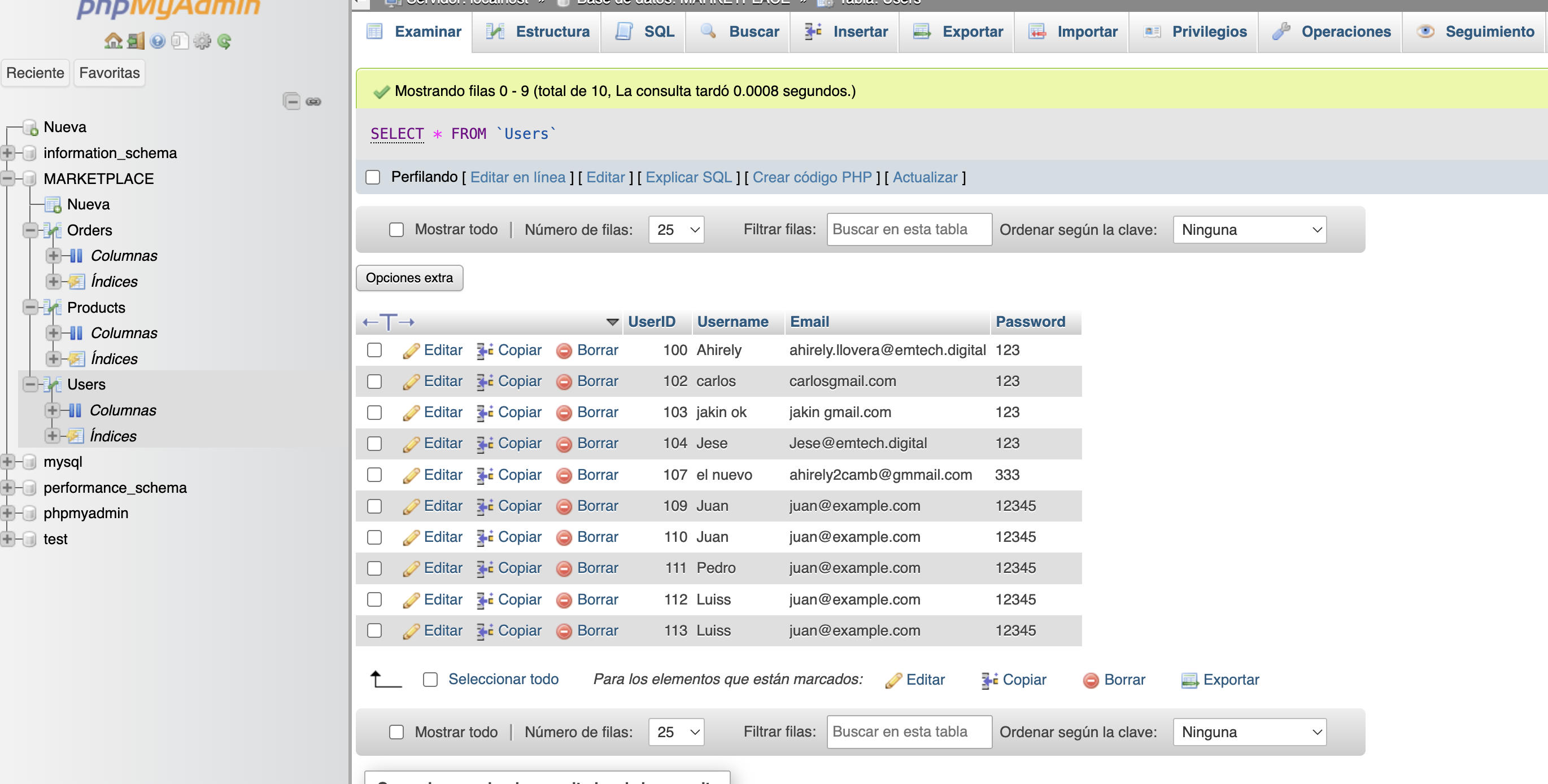
 Un producto **pertenece a una categoría**.

### Tipos de relaciones en el diseño

|  |  |  |
| --- | --- | --- |
| **Relación** | **Tipo** | **Comentario** |
| productos.id\_vendedor → usuarios.id\_usuario | Uno a muchos | Un usuario puede tener muchos productos |
| productos.id\_categoria → categorias.id\_categoria | Uno a muchos | Una categoría puede tener muchos productos |
| pedidos.id\_comprador → usuarios.id\_usuario | Uno a muchos | Un usuario puede hacer muchos pedidos |
| pedidos.id\_producto → productos.id\_producto | Uno a muchos | Un producto puede estar en muchos pedidos |
| reseñas.id\_producto → productos.id\_producto | Uno a muchos | Un producto puede tener muchas reseñas |
| reseñas.id\_usuario → usuarios.id\_usuario | Uno a muchos | Un usuario puede dejar muchas reseñas |

Desarrollo de la base de datos:

Aplicando los conocimientos del Módulo 3, se deberá implementar y configurar la base de datos del proyecto, incluyendo tablas y relaciones necesarias. Se puede tomar como base y/o referencia, la base de datos utilizada en el Caso Práctico 3.



Ejemplo, desde SQL.

-- Tabla de usuarios: contiene compradores y vendedores

CREATE TABLE usuarios (

id\_usuario INT AUTO\_INCREMENT PRIMARY KEY,

nombre\_usuario VARCHAR(50) UNIQUE NOT NULL,

correo VARCHAR(100) UNIQUE NOT NULL,

contraseña VARCHAR(100) NOT NULL

);

-- Tabla de categorías: cada producto puede pertenecer a una categoría

CREATE TABLE categorias (

id\_categoria INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(50) UNIQUE NOT NULL

);

-- Tabla de productos: cada producto tiene un vendedor y una posible categoría

CREATE TABLE productos (

id\_producto INT AUTO\_INCREMENT PRIMARY KEY,

-- RELACIÓN: muchos productos pueden pertenecer a un usuario vendedor

id\_vendedor INT NOT NULL,

FOREIGN KEY (id\_vendedor) REFERENCES usuarios(id\_usuario),

-- RELACIÓN: muchos productos pueden pertenecer a una categoría

id\_categoria INT,

FOREIGN KEY (id\_categoria) REFERENCES categorias(id\_categoria),

nombre VARCHAR(100) NOT NULL,

descripcion TEXT,

precio DECIMAL(10, 2) NOT NULL

);

-- Tabla de pedidos: cada pedido está hecho por un comprador y está asociado a un producto

CREATE TABLE pedidos (

id\_pedido INT AUTO\_INCREMENT PRIMARY KEY,

-- RELACIÓN: muchos pedidos pueden ser hechos por el mismo comprador

id\_comprador INT NOT NULL,

FOREIGN KEY (id\_comprador) REFERENCES usuarios(id\_usuario),

-- RELACIÓN: muchos pedidos pueden referirse al mismo producto

id\_producto INT NOT NULL,

FOREIGN KEY (id\_producto) REFERENCES productos(id\_producto),

cantidad INT NOT NULL,

fecha\_pedido DATE NOT NULL

);

* Desarrollo del Backend:

Basado en el Módulo 4, será necesario desarrollar la lógica backend de la

* aplicación, incluyendo la creación de servicios REST API principalmente en Python o JavaScript.
* Será importante garantizar que se integre el backend con la base de datos para poder realizar las distintas operaciones necesarias que se trabajaron en el Caso Práctico 4:
  1. Altas: crear nuevos registros (usuarios, productos, pedidos).
  2. Bajas: eliminar registros existentes.
  3. Modificaciones: actualizar datos de registros existentes.

**Coding with AI:** Proyector integrador **02**

# Estructura:



Pantalla azul con letras blancas

El contenido generado por IA puede ser incorrecto.

**Pantalla de un video juego

El contenido generado por IA puede ser incorrecto.**

**Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.**

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

Desarrollo del frontend:

* Siguiendo el Módulo 5, diseñar y desarrollar el frontend de la aplicación, ya sea móvil o web, utilizando Kivy o tecnologías similares.
* *class* ProductPanel(*BoxLayout*):
* *def* \_\_init\_\_(*self*, \*\**kwargs*):
* *super*().\_\_init\_\_(*orientation*='vertical', \*\*kwargs)
* self.add\_widget(Label(*text*='Nombre del Producto:'))
* self.product\_name = TextInput(*multiline*=False)
* self.add\_widget(self.product\_name)
* self.add\_widget(Label(*text*='Descripción:'))
* self.product\_description = TextInput(*multiline*=False)
* self.add\_widget(self.product\_description)
* self.add\_widget(Label(*text*='Precio:'))
* self.product\_price = TextInput(*multiline*=False)
* self.add\_widget(self.product\_price)
* self.add\_product\_button = Button(*text*='Guardar Producto')
* self.add\_product\_button.bind(*on\_press*=self.save\_product)
* self.add\_widget(self.add\_product\_button)
* self.view\_products\_button = Button(*text*='Ver Productos en Consola')
* self.view\_products\_button.bind(*on\_press*=self.show\_products)
* self.add\_widget(self.view\_products\_button)
* self.product\_list\_label = Label(*text*='Estado: Esperando acción...')
* self.add\_widget(self.product\_list\_label)

* Implementar funcionalidades clave como el inicio de sesión y la visualización de productos.

Use la librería sqlite3, pero si quisiera manejar los datos desde un servidor, debo isa SQLAlchemy + PyMySQL. Tiene interfaz grafica Kivy. Contiene Machine Leraning, pandas y scikit-learn.

Inicio de sesión

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

* Se deberán utilizar como mínimo las tres interfaces trabajadas en el Caso Práctico 5, donde la primera consiste en crear un inicio de sesión con OAUTH de Google,  
    
   la segunda se basa en crear un panel donde se registran los productos  
    
  La tercera servirá para presentar los productos para una compra y adición al carrito.  
    
  Integración de modelos de IA:

Registro de productos

Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Chatbot

Texto

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Carrito de producto

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Test

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Optimización y seguridad:

Aplicar las técnicas aprendidas en los Módulos 8 y 9 para optimizar el código usando los asistentes trabajados en el curso y asegurar que la aplicación esté protegida contra vulnerabilidades básicas y comunes.

Del Modulo **8.Monitoreo**…  
# Importamos las librerías necesarias

import requests # Para hacer solicitudes HTTP a una URL

import time # Para medir el tiempo y pausar el monitoreo

def get\_response\_time(url):

"""

Realiza una solicitud HTTP GET a la URL proporcionada y calcula

cuánto tiempo tardó en obtener la respuesta.

"""

# requests.get(url) envía una solicitud GET, HTTP GET a la URL

response = requests.get(url)

# Obtenemos el tiempo de respuesta en segundos desde que se envió hasta que se recibió

# .elapsed es un objeto timedelta, y .total\_seconds() lo convierte a segundos flotantes

response\_time = response.elapsed.total\_seconds()

# Retornamos el tiempo de respuesta

return response\_time

def monitor\_performance(url, threshold):

"""

Verifica el rendimiento de una URL comparando el tiempo de respuesta

con un umbral definido por el usuario.

"""

# Llama a la función que obtiene el tiempo de respuesta actual

response\_time = get\_response\_time(url)

# Compara el tiempo de respuesta con el umbral definido

if response\_time > threshold:

# Si se excede el umbral, se imprime una alerta con la información relevante

print(

"ALERTA: El tiempo de respuesta de la URL '{}' es de {:.2f} segundos. "

"Esto supera el umbral de {:.2f} segundos.".format(url, response\_time, threshold)

)

else:

# Si está dentro del rango normal, se muestra un mensaje informativo

print(

"OK: El tiempo de respuesta de '{}' es {:.2f} segundos, dentro del umbral.".format(url, response\_time)

)

if \_\_name\_\_ == "\_\_main\_\_":

"""

Bloque principal del programa: define la URL a monitorear y el umbral

y ejecuta el monitoreo en un bucle continuo.

"""

# Definimos la URL de la aplicación web a monitorear

#url = "https://www.google.com"

url = "https://www.example.com"

# Definimos el umbral de tiempo de respuesta permitido (en segundos)

threshold = 2.0

# Inicia un bucle infinito que monitorea la URL cada segundo

while True:

# Llama a la función que evalúa el rendimiento

monitor\_performance(url, threshold)

# Espera 1 segundo antes de volver a comprobar

* time.sleep(1.0)

Del modulo 9. Modelo de monitoreo y validación de datos. **03**

# Estructura:



Primero se planifico el proyecto, se obtuvo un carrito de productos. Se hizo el diseño de la base de datos y se creó. Posteriormente se hizo el código del backend, usando apis, luego se realizó el frontend usando la tecnología de Kivy. También se Outh para el inicio de sesión y un chatbot para la asistencia técnica del carrito. Por ultimo se hicieron pruebas de monitoreo y test de su funcionalidad y para la seguridad.



**Entregables a considerar:**

Aplicación móvil o web completamente funcional en una carpeta .ZIP con todos los assets y códigos necesarios. Documentación técnica del desarrollo del proyecto.

**Coding with AI:** Proyector integrador **04**