

# Documentació Pràctica TechShop

## 1. Introducció

El projecte **TechShop** consisteix en el desenvolupament d'una aplicació web de comerç electrònic senzilla, en la qual un usuari pot consultar productes, afegir-los al carretó i completar una comanda.

L'objectiu del projecte és posar en pràctica els coneixements sobre arquitectura MVC (Model–Vista–Controlador), persistència de dades amb SQLite i validació d'informació al client i al servidor.

L'aplicació s'ha desenvolupat amb:

- **Flask** com a framework backend en Python.
- **SQLAlchemy** com a ORM per interactuar amb la base de dades SQLite.
- **HTML, CSS i JavaScript** per a la capa de presentació.
- **Blueprints de Flask** per modularitzar les rutes.

## 2. Arquitectura i estructura del projecte

Hem aplicat una arquitectura **en tres capes** (dades, lògica de negoci i presentació) per mantenir el codi separat i ben organitzat.

Aquesta estructura segueix el patró **MVC**, on cada capa té una funció concreta:

- **Model:** gestiona les dades i la connexió amb la base de dades.
- **Vista:** gestiona la interfície d'usuari (fitxers HTML i recursos estàtics).
- **Controlador:** gestiona la lògica de les rutes i la comunicació entre la vista i el model.

L'estructura del projecte és la següent:

PRACTICA\_TECHSHOP/

```
|
|
| — database/
|   | — db_init.py      → Crea la base de dades i inicialitza SQLAlchemy.
|   | — db_seed.py      → Insereix dades de prova (productes inicials).
|   |
|   | — instance/
|     | — techshop.db    → Fitxer físic de la base de dades SQLite.
|     |
|     | — models/
|       | — product_model.py → Model Product (id, name, price, stock).
|       | — user_model.py   → Model User (id, username, password_hash, email).
|       | — order_model.py  → Model Order (id, total, user_id, created_at).
|       | — order_item_model.py → Model OrderItem (order_id, product_id, quantity).
|       |
|       | — routes/
|         | — routes.py     → Conté totes les rutes de l'aplicació (checkout, cart, etc.).
|         |
|         | — services/
|           | — cart_service.py → Lògica per gestionar el carretó i validar estoc.
|           | — order_service.py → Crea comandes i actualitza l'estoc.
|           | — product_service.py → Funcions auxiliars per obtenir o actualitzar productes.
|           |
|           | — static/
|             | — css/style.css → Full d'estils principal.
```

```

|   └─ js/main.js      → Validacions i comportament del frontend.
|
|   └─ templates/
|       └─ index.html   → Pàgina principal.
|       └─ checkout.html → Formulari de registre i compra.
|       └─ products.html → Llista de productes.
|       └─ cart.html     → Vista del carretó de compra.
|   └─ order_succes.html → Missatge final de confirmació.
|
|   └─ app.py           → Punt d'entrada de l'aplicació Flask.
└─ requirements.txt     → Llista de dependències necessàries.

```

Aquesta organització facilita entendre el flux complet de l'aplicació i permet que cada part del sistema es pugui mantenir o modificar sense afectar la resta.

Destaquem la carpeta services on

### **cart\_service.py**

Aquest mòdul gestiona totes les operacions del **carretó de compra**.

És responsable de comprovar les quantitats, validar l'estoc i controlar el límit màxim de productes.

Les funcions principals són:

- `add_to_cart(cart, product, quantity)`  
Afegeix un producte al carretó si la quantitat és vàlida.  
Comprova que el valor sigui positiu i que no superi les **5 unitats per producte**.  
En cas contrari, genera un missatge d'error o una excepció.
- `remove_from_cart(cart, product_id)`  
Elimina un producte del carretó.  
No depèn de cap codi HTML ni de cap accés directe a la base de dades, mantenint la lògica totalment separada.
- `validate_stock(product, quantity)`  
Verifica que hi hagi prou unitats disponibles abans d'afegir-les.  
Si l'estoc és insuficient, mostra un missatge informatiu a l'usuari.

Aquest servei permet garantir que totes les operacions del carretó siguin coherents i segures.

### **order\_service.py**

Aquest mòdul gestiona la creació de **comandes (orders)** i les seves línies de detall (**order\_items**).

S'encarrega de calcular el total de la compra, desar les dades i actualitzar l'estoc dels productes corresponents.

Les funcions principals són:

- `create_order(cart, user_id)`  
Crea una nova comanda associada a l'usuari actual.  
Calcula el total sumant *preu* × *quantitat* de cada producte.  
A més, actualitza automàticament l'estoc restant i desa les línies de comanda a la taula OrderItem.

Aquesta funció és clau per mantenir la coherència entre les taules Order, OrderItem i Product.

### **product\_service.py**

Aquest servei conté funcions d'accés i gestió de productes.

- `get_all_products()`  
Retorna tots els productes disponibles a la base de dades.  
Aquesta funció és utilitzada per la ruta /products.
- `get_product_by_id(product_id)`  
Busca un producte pel seu identificador únic (ID).
- `update_stock(product_id, quantity)`  
Actualitza l'estoc restant d'un producte després d'una compra.  
Inclou validacions per evitar que es restin més unitats de les disponibles.

Aquest mòdul permet tenir un punt únic de gestió per totes les operacions relacionades amb productes, facilitant el manteniment i la reutilització del codi.

### **user\_service.py**

Aquest servei és responsable de la **gestió dels usuaris**, incloent el registre, la validació i la seguretat de les contrasenyes.

- `create_user(username, password, email)`  
Registra un nou usuari després de comprovar que no existeix un nom d'usuari o correu duplicat.  
Les contrasenyes es guarden xifrades amb `generate_password_hash` per garantir la seguretat.
- `validate_user(username, password)`  
Verifica si les credencials proporcionades coincideixen amb un usuari existent.  
Utilitza `check_password_hash` per comparar de manera segura la contrasenya introduïda amb la guardada a la base de dades.

Aquest servei permet gestionar el registre i inici de sessió sense barrejar la lògica d'autenticació amb les rutes o el frontend.

### 3. Base de dades

S'ha utilitzat **SQLite** com a sistema de base de dades relacional.

El fitxer `techshop.db` s'ubica dins la carpeta `instance/` i es genera automàticament quan s'executa l'aplicació per primera vegada mitjançant el mòdul `db_init.py`.

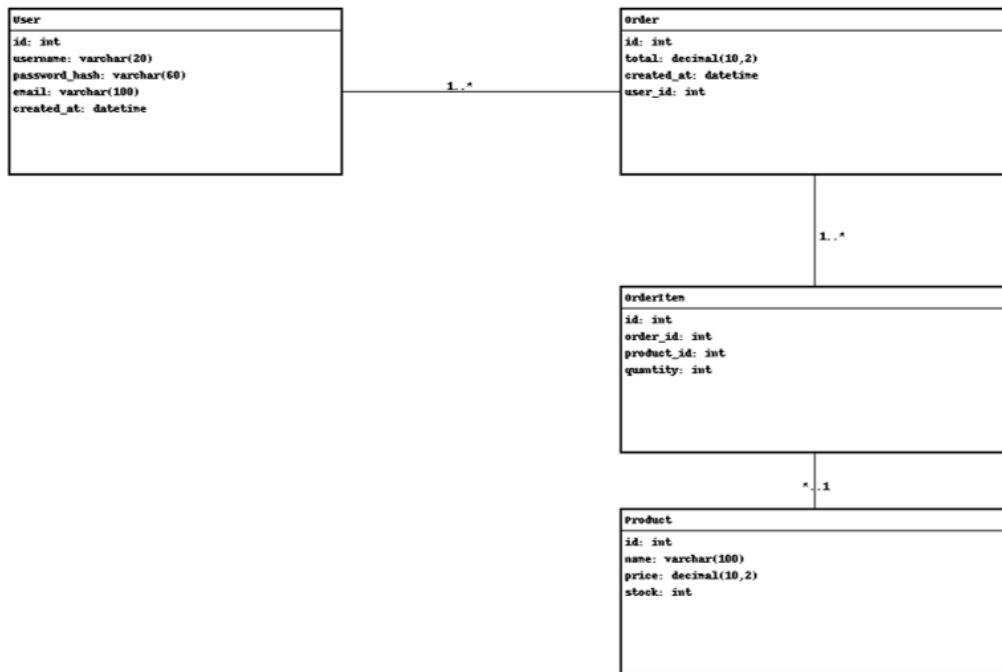
Taules principals:

- **Product:** conté la informació dels productes (nom, preu i estoc disponible).
- **User:** guarda les dades dels usuaris (nom, correu i contrasenya xifrada).
- **Order:** registra cada comanda feta per un usuari.
- **OrderItem:** enllaça els productes amb les comandes i en guarda la quantitat.

Les relacions són les següents:

- Un **usuari** pot tenir moltes **comandes** (1:N).
- Una **comanda** pot tenir molts **productes** associats (1:N).

- Cada OrderItem està vinculat a un únic Product i una única Order.



DB Browser for SQLite - C:\Users\aripa\MASTER\Practica\_TechShop\instance\techshop.db

Archivo Editar Ver Herramientas Ayuda

Nueva base de datos Abrir base de datos Guardar cambios Revertir cambios Deshacer Abrir proyecto Guardar proyecto Anexar base de datos Cerrar base de datos

Estructura Hoja de datos Editar pragmas Ejecutar SQL

Crear tabla Crear índice Modificar tabla Borrar tabla Imprimir Refrescar

Nombre	Tipo	Esquema
▼ Tablas (4)		
▼ order		CREATE TABLE "order" ( id INTEGER NOT NULL, total FLOAT NOT NULL, created_at DATETIME, user_id INTEGER NOT NULL, PRIMARY KEY (id), FOREIGN KEY(user_id) REFERENCES user (id) )
id	INTEGER	"id" INTEGER NOT NULL
total	FLOAT	"total" FLOAT NOT NULL
created_at	DATETIME	"created_at" DATETIME
user_id	INTEGER	"user_id" INTEGER NOT NULL
▼ order_item		CREATE TABLE order_item ( id INTEGER NOT NULL, order_id INTEGER NOT NULL, product_id INTEGER NOT NULL, quantity INTEGER NOT NULL, PRIMARY KEY (id), FOREIGN KEY(order_id) REFERENCES "order" (id), FOREIGN KEY(product_id) REFERENCES product (id) )
id	INTEGER	"id" INTEGER NOT NULL
order_id	INTEGER	"order_id" INTEGER NOT NULL
product_id	INTEGER	"product_id" INTEGER NOT NULL
quantity	INTEGER	"quantity" INTEGER NOT NULL
▼ product		CREATE TABLE product ( id INTEGER NOT NULL, name VARCHAR(100) NOT NULL, price NUMERIC(10, 2) NOT NULL, stock INTEGER NOT NULL, PRIMARY KEY (id) )
id	INTEGER	"id" INTEGER NOT NULL
name	VARCHAR(100)	"name" VARCHAR(100) NOT NULL
price	NUMERIC(10, 2)	"price" NUMERIC(10, 2) NOT NULL
stock	INTEGER	"stock" INTEGER NOT NULL
▼ user		CREATE TABLE user ( id INTEGER NOT NULL, username VARCHAR(20) NOT NULL, password_hash VARCHAR(60) NOT NULL, email VARCHAR(100) NOT NULL, created_at DATETIME, PRIMARY KEY (id), UNIQUE (username) )
id	INTEGER	"id" INTEGER NOT NULL
username	VARCHAR(20)	"username" VARCHAR(20) NOT NULL
password_hash	VARCHAR(60)	"password_hash" VARCHAR(60) NOT NULL
email	VARCHAR(100)	"email" VARCHAR(100) NOT NULL
created_at	DATETIME	"created_at" DATETIME
Índices (0)		
Vistas (0)		
Disparadores (0)		

## 4. Procés de desenvolupament

El desenvolupament del projecte s'ha realitzat seguint diverses fases ordenades:

### 4.1. Disseny inicial

Abans de començar a programar, es va definir l'arquitectura general i es va establir la separació de carpetes per garantir una estructura clara i modular.

A més, es van crear diagrames per visualitzar la relació entre entitats i el flux d'usuari (checkout → productes → carretó → compra finalitzada).

### 4.2. Creació dels models

Es van definir els models amb SQLAlchemy especificant tipus de dades, claus primàries i foranes, i les relacions entre ells.

Aquesta capa és responsable de comunicar-se amb la base de dades sense utilitzar sentències SQL directes.

### 4.3. Implementació dels serveis

A la carpeta services/ s'ha concentrat tota la lògica de negoci.

Aquí s'ha implementat, per exemple, el control de quantitat màxima per producte (5 unitats), la validació d'estoc abans d'afegir productes al carretó i la creació automàtica de les ordres i línies de comanda quan l'usuari finalitza la compra.

### 4.4. Rutes i controladors

Les rutes de Flask (checkout, products, cart, finish\_checkout) s'han definit dins un **Blueprint**, seguint el patró MVC.

Cada ruta invoca la lògica dels serveis i després mostra la plantilla HTML corresponent.

### 4.5. Interfície d'usuari

S'han creat diverses plantilles HTML amb un estil coherent i senzill.

La pàgina de **checkout** incorpora validacions HTML5 (required, minlength, maxlength, pattern) i validacions addicionals amb JavaScript per comprovar la força de la contrasenya i la coincidència entre contrasenya i confirmació.

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/checkout'. The page has a dark teal header with the title 'Formulari de Checkout' and a subtitle 'Introdueix les teves dades per continuar la compra'. The main content area is light gray and contains a white form with the following fields:

- Nom d'usuari:** A text input field containing 'emily.johnson@techco'. Below it, a small gray box says 'Nom d'usuari sense espais (4-20 caràcters)'.
- Contrasenya:** A password input field with a red error message 'X Massa curta' below it.
- Confirma la contrasenya:** A text input field for repeating the password. Below it, a small gray box says 'Les contrasenyes han de coincidir'.
- Correu electrònic:** A text input field containing 'emily.johnson@techcorp.co'. Below it, a small gray box says 'Introdueix un correu vàlid'.
- Adreça d'enviament:** A text input field with a placeholder 'Ex: Carrer de la Pau, 23, Bc'. Below it, a small gray box says 'Introdueix una adreça completa'.

At the bottom of the form is a green button with a star icon and the text 'Continuar a la botiga'. The footer of the page is dark teal and contains the text '© 2025 TechShop - Desenvolupat per Ariadna Pascual i Hugo Torres'.

#### 4.6. Validacions

S'ha implementat un doble sistema de validació:

##### Al client (frontend):

- Validació dels camps amb atributs HTML i JavaScript.
- Expressió regular per validar el correu electrònic.
- Confirmació visual de la contrasenya.

##### Al servidor (backend):

- Validació de dades rebudes abans de desar-les.
- Xifratge de contrasenya amb `generate_password_hash()`.
- Comprovació de l'estoc abans de crear una comanda.
- Control d'errors amb missatges mitjançant `flash()`.



## 5. Resultats i proves

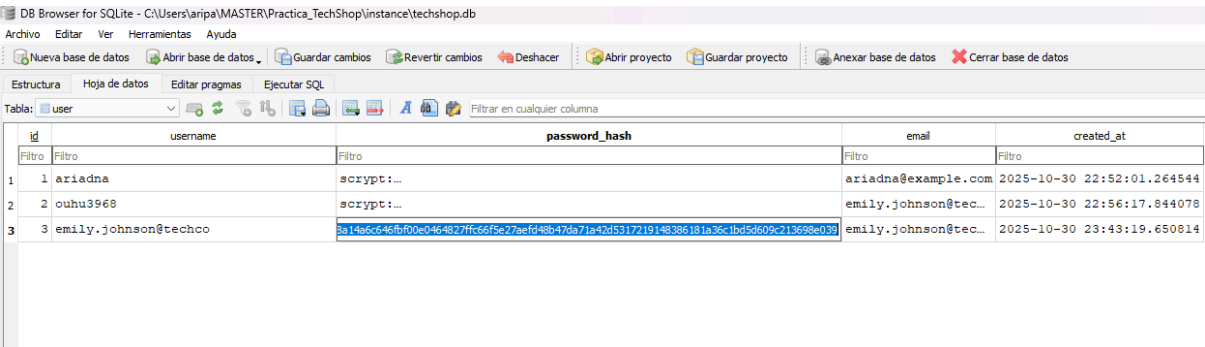
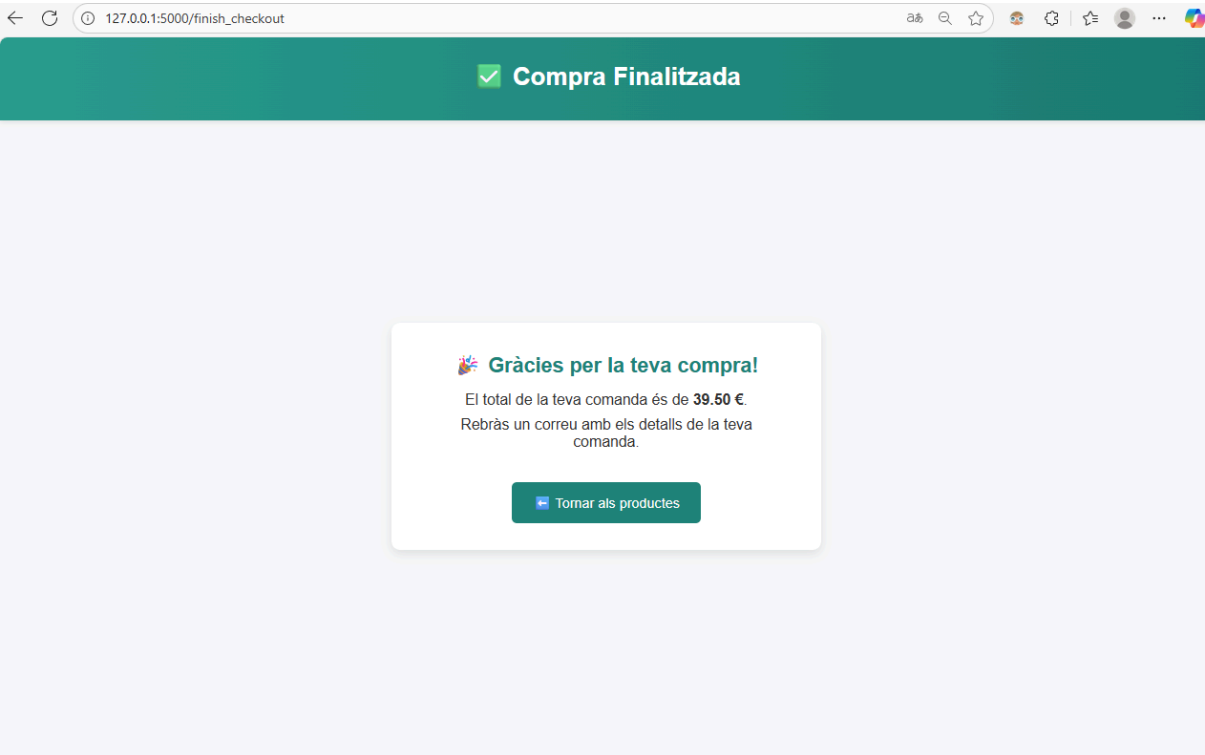
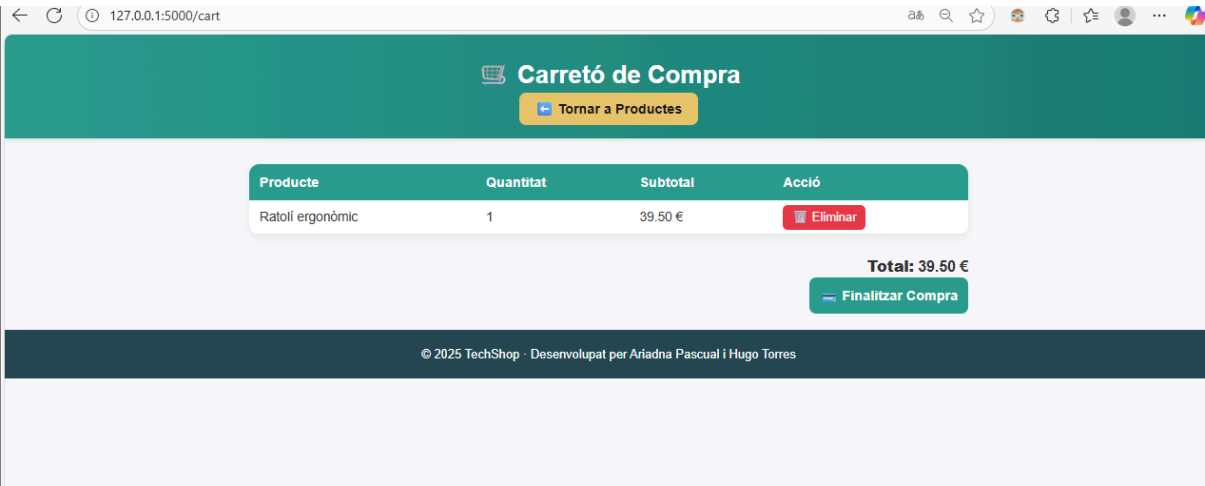
Un cop finalitzat el desenvolupament, s'ha comprovat el correcte funcionament del flux complet de compra:

1. L'usuari accedeix al formulari de **checkout**.
2. Si ja existeix a la base de dades, s'inicia la sessió; si no, es crea un nou registre.
3. Accedeix a la pàgina de **productes** i pot afegir articles al carretó (màxim 5 per producte).
4. A la pàgina **cart**, pot revisar la compra, veure els subtotals i el total.
5. En prémer "Finalitzar compra", es crea un registre a la taula **Order** i els corresponents **OrderItem**.
6. L'estoc dels productes es redueix i el carretó es buida automàticament.
7. Finalment, es mostra una pàgina de confirmació amb el total de la compra.

## PRODUCTS



## CART



DB Browser for SQLite - C:\Users\aripa\MASTER\Practica\_TechShop\instance\techshop.db

Archivo Editar Ver Herramientas Ayuda

Nueva base de datos Abrir base de datos Guardar cambios Revertir cambios

Estructura Hoja de datos Editar pragmas Ejecutar SQL

Tabla: product

	id	name	price	stock
	Filtro	Filtro	Filtro	Filtro
1	1	Auriculars Bluetooth	29.99	20
2	2	Teclat mecànic RGB	79.99	13
3	3	Ratolí ergonòmic	39.5	7
4	4	Monitor FullHD	199.99	8
5	5	Portàtil Lenovo	999	5

DB Browser for SQLite - C:\Users\aripa\MASTER\Practica\_TechShop\instance\techshop.db

Archivo Editar Ver Herramientas Ayuda

Nueva base de datos Abrir base de datos Guardar cambios Revertir cambios Deshacer

Estructura Hoja de datos Editar pragmas Ejecutar SQL

Tabla: order\_item

	id	order_id	product_id	quantity
	Filtro	Filtro	Filtro	Filtro
1	1	1	3	2
2	2	2	2	1
3	3	3	2	1
4	4	4	3	1

DB Browser for SQLite - C:\Users\aripa\MASTER\Practica\_TechShop\instance\techshop.db

Archivo Editar Ver Herramientas Ayuda

Nueva base de datos Abrir base de datos Guardar cambios Revertir cambios De

Estructura Hoja de datos Editar pragmas Ejecutar SQL

Tabla: order

	id	total	created_at	user_id
	Filtro	Filtro	Filtro	Filtro
1	1	79.0	2025-10-31 ...	2
2	2	79.99	2025-10-31 ...	2
3	3	79.99	2025-10-31 ...	2
4	4	39.5	2025-10-31 ...	3

## 6. Ús de la intel·ligència artificial (IA)

Durant la realització del projecte, hem fet ús d'eines d'intel·ligència artificial, com ChatGPT, de manera responsable i supervisada.

L'objectiu principal ha estat millorar el nostre procés d'aprenentatge, obtenir suport tècnic puntual i optimitzar algunes parts del desenvolupament del projecte TechShop.

### Aplicacions concretes

Al llarg del projecte, hem utilitzat la IA com a recurs complementari en diversos moments del desenvolupament:

Hem generat esbossos de codi inicials per als models i serveis, seguint les regles del patró MVC i assegurant la separació entre capes.

Hem revisat fragments de codi per detectar possibles errors o millorar aspectes de seguretat i eficiència.

Hem rebut suggeriments per millorar l'organització del projecte, especialment en la modularització i estructuració de carpetes.

Hem fet servir la IA com a suport en la redacció de la documentació tècnica, ajudant-nos a redactar textos més clars i ben estructurats.

Hem demanat ajuda per depurar errors relacionats amb la gestió de sessions, la persistència de dades i l'ús de SQLAlchemy.

### Regles establertes a la IA

Per garantir que la col·laboració amb la IA fos coherent amb els objectius educatius i tècnics del projecte, vam establir una sèrie de directrius clares que la IA havia de respectar en tot moment:

No barrejar mai codi HTML amb lògica de negoci o consultes SQL.

Centralitzar tots els accessos a la base de dades a través de la capa de models o serveis.

Limitar les unitats màximes del carretó a cinc per producte.

Validar sempre les dades rebudes abans de processar-les o guardar-les a la base de dades.

Xifrar les contrasenyes i evitar emmagatzemar informació sensible en text pla.

### Reflexió sobre l'ús de la IA

Considerem que la IA ha estat una eina de suport molt útil per reforçar la nostra comprensió de l'arquitectura MVC i millorar la qualitat del codi.

Tot i això, hem mantingut una actitud crítica i analítica davant de totes les seves respostes: cada proposta o fragment de codi suggerit ha estat revisat, comprès i validat abans d'aplicar-lo.

Aquesta experiència ens ha permès comprovar que la IA pot ser un recurs pedagògic molt potent, sempre que s'utilitzi amb responsabilitat i com a eina de suport, mai com a substitut del raonament propi o del coneixement tècnic.

## Exemples de consultes fetes a la IA

A continuació presentem alguns exemples reals de prompts o consultes que hem fet servir durant el desenvolupament del projecte.

Aquests exemples demostren com hem aprofitat la IA per complementar el nostre treball i millorar la qualitat del codi:

“Com puc estructurar un projecte Flask seguint el patró MVC amb carpetes separades per models, serveis i rutes?”

→ Aquesta consulta ens va ajudar a definir l'estructura inicial del projecte i crear una arquitectura clara i modular.

“Crea un model SQLAlchemy per a una taula OrderItem amb relacions cap a Order i Product.”

→ Ens va servir com a punt de partida per redactar correctament els models i entendre com funcionen les claus foranes en SQLAlchemy.

“Per què no es guarden les dades a la taula order després de fer checkout a Flask?”

→ Aquesta pregunta ens va permetre depurar un error relacionat amb la sessió i amb la falta de `db.session.commit()`.

“Com puc limitar a 5 unitats per producte dins del carretó abans de crear la comanda?”

→ Amb aquesta consulta vam obtenir idees per implementar la validació de quantitats dins del `cart_service.py`.

“Ajuda'm a redactar la memòria del projecte TechShop explicant les capes del patró MVC sense utilitzar codi.”

→ Aquesta consulta ens va ajudar a millorar la redacció de la documentació final i donar-li un to formal i acadèmic.

## Exemple1

Per què no es guarden les dades a la taula `order` després de fer `checkout` a Flask?


Les dades poden no guardar-se a la taula `order` després de fer `checkout` a Flask per diversos motius habituals, tots relacionats amb la manera com funciona la sessió i la persistència de dades amb SQLAlchemy.

A continuació t'explico les causes més freqüents i com resoldre-les:

### ♦ 1. No s'està fent `db.session.commit()`

Quan crees una nova comanda amb:

```
python
db.session.add(new_order)
```

 Copiar código

aquesta només s'afegeix a la sessió temporal.

Si no s'executa després:

```
python
db.session.commit()
```

 Copiar código

els canvis no s'escriuen realment a la base de dades.


Això és un error molt comú i provoca que, tot i semblar que s'ha creat l'ordre, no es desitja res al fitxer `.db`.

### ♦ 2. El `user_id` no existeix o no està guardat a la sessió

Si l'usuari no s'ha registrat correctament o no s'ha guardat el seu ID a la sessió (`session['user_id']`), l'ordre no pot associar-se a cap usuari i no es crea cap registre.

Solució:

```
python
session['user_id'] = new_user.id
```

 Copiar código

### ♦ 3. La base de dades es recrea cada cop qu executa l'app

Si el codi conté línies com:

## Exemple2

Crea un model SQLAlchemy per a una taula OrderItem amb relacions cap a Order i Product.

python

 Copiar código

```
from . import db

class OrderItem(db.Model):
    __tablename__ = 'order_item'

    id = db.Column(db.Integer, primary_key=True)
    order_id = db.Column(db.Integer, db.ForeignKey('order.id'), nullable=False)
    product_id = db.Column(db.Integer, db.ForeignKey('product.id'), nullable=False)
    quantity = db.Column(db.Integer, nullable=False)
```

     ...



## 7. Conclusions

El desenvolupament del projecte TechShop ens ha permès posar en pràctica tots els coneixements adquirits sobre programació web amb Python i Flask.

S'han aplicat correctament els principis de l'arquitectura MVC, la separació de responsabilitats i la persistència de dades amb SQLite.

L'aplicació final és funcional, manté una bona organització del codi i compleix tots els requisits sol·licitats.

A més, l'ús controlat de la intel·ligència artificial ha contribuït a millorar el resultat final i a aprofundir en la comprensió de les bones pràctiques de desenvolupament.