Continuation of the front-end exercise on the e-commerce site (Database)

This exercise can be done individually or in groups. If some prefer to work on their personal projects, that's fine, but you'll need to send me a report at the end of the day detailing what was done.

We previously worked on a front-end exercise using an API that returned articles, users, and other information for building an e-commerce site.

The issue with this API is that it cannot be modified: it's impossible to create our own users, edit a cart, or add products...

Our next step is to create our own API to replace this test API.
But before doing so, in order to follow best practices, we'll need to create a schema.

1. Database Schema

Use the Merise method or any other method to create at least one schema (Conceptual Data Model - CDM) and ideally a second one (Logical Data Model - LDM) for your API's database.

To ease the transition from your front-end to this new API, it is recommended to keep the same column names as the properties returned by the existing API. (If the API uses "price" for an article, keep "price" as the column name rather than using "prix", for example.)

You'll likely need the following tables:

- users
- articles
- carts
- ...and probably others

Be sure to clearly indicate the cardinalities (relationships) between tables.

Summary

✅ You must submit a CDM schema (on paper or using a digital tool)
▨ If possible, also add an LDM schema
🔁 You can use tools like dbdiagram.io, MySQL Workbench, or even Draw.io

💡 Example of a relationship:
A cart contains multiple articles → N-N relationship → pivot table: `cart_items`

The pivot table will link articles to carts.
If we place the article ID in the cart, a cart could only hold one article.
If we place the cart ID in the article, an article could only be linked to one cart.
So, we create a pivot table with at least two fields: the article ID and the cart ID, so that each entry represents a cart-article relationship.

🧩 Before moving on, take 5 minutes to present your schema(s) to someone else.
This will help identify any missing elements or inconsistencies (e.g., incorrect relationships, missing tables).

2. Creating the Database

Once your schemas are complete, create the database.
You can do this using:

- an interface like PHPMyAdmin,
- the command line,
- or by creating an SQL file to import.

Once your database is created, be sure to export it to keep a file you can reuse on other machines, or to recover from mistakes or deletions.