Planning

This week the students will be building a **backend API for a supermarket system**. They will practice:

- Setting up their own SQL database.
- Using **Drizzle ORM** (preferred) or Sequelize.
- Validating request bodies with **Zod**.
- Building CRUD operations for **Customers, Inventory, and Sales**.
- Applying **Basic Auth → JWT** for security.

👉 Students are beginners. Move slowly, emphasize **WHY** we do each step, and check their understanding often with **Postman testing checkpoints**.

Day 1 – Environment Setup + Fundamentals

Goal: Everyone has their own DB, ORM configured, and validation tool ready.

1. **Database Setup**
   - Guide them through creating their own MySQL database (local or cloud: e.g., Railway).
   - Show them how to make a `.env` file with DB credentials.
   - Practice connecting manually (via MySQL client or CLI).
   - Explain **why** each table is needed (Customers, Inventory, Sales).
2. **Preferred ORM (Drizzle o el que se te haga más fácil jaja)**
   - Show how schemas are defined
   - Basic setup in a project (students have never connected to a DB using JS)
3. **Zod for Validation**
   - Install `zod`.

- Show how to validate a simple object:

```
const productSchema = z.object({
  name: z.string().min(1),
  price: z.number().positive(),
  stock_quantity: z.number().int().nonnegative()
});
```

- Practice validating JSON in a fake request handler.
- **Checkpoint in Postman**: send invalid payloads and see the validation errors.

Days 2, 3, 4 – Guided Project (Supermarket API)

Day 2 – Authentication + Customers

- Implement `/auth/login` (Basic Auth hardcoded (no users tables yet) → returns JWT).
  - Walk through why we secure APIs.
  - Explain how JWT works.
- Create `customersRouter` with CRUD endpoints.
- Apply Zod validation to `POST` and `PUT` customer requests.
- Postman checkpoint: try accessing `/customers` **without** a JWT → fail. Then with JWT → success.

Day 3 – Inventory

- Create `inventoryRouter` with CRUD endpoints.
- Add Zod validation for product fields.
- Explain “business logic” like preventing negative stock.
- Practice updating stock with `PATCH` or an `/adjust-stock` route.

- Postman checkpoint: try inserting bad data (e.g., `price: -10`) and confirm validation catches it.

Day 4 – Sales

- Create `salesRouter`.
- Show how to **compute total_price = quantity × price** on the server.
- Teach transaction-style logic: when a sale is created → stock decreases.
- Add validation (e.g., prevent sale if `quantity > stock`).
- Optional reporting route: `/sales/summary`.
- Postman checkpoint: create a sale, then check that inventory stock is updated.

✅ Suggested Teaching Approach

- **Model → Route → Postman Test** cycle

  Each time a table/feature is introduced, follow this pattern:

  1. Define the schema/model in Drizzle.
  2. Create a route in Express.
  3. Test it live in Postman.

- **Visualize Relationships**

  Keep showing the ER diagram (Customers ↔ Sales ↔ Inventory). Beginners often get lost in relationships unless they see it visually.

- **Checkpoints & Questions**

  After finishing each router, pause and have students:

  - Add a record.
  - Retrieve it.
  - Break it (send invalid payload). Teach them how to debug (porque no saben mucho).
  - Fix it.
