Inventory Service Use Case

The goal of the exercise is to understand the product needs and translate that to a SQL schema and create a simple API that will help us implement features quickly.

You should create a GitHub repository with the proper code to run an API, it will be helpful if you include a docker to wake up the server and DB as well.

You can use any backend stack or DB combination you want (we currently use node js (with Typescript) and Mysql, we plan to move to Rust later on)

The project should contain:

- Database schema definition (you can use an ORM schema or a raw SQL script)
- A functional API with endpoints that query a database
- Commands to build the db and run the code

Considerations

Having a **well designed SQL data schema** is a big plus, we would like to have a schema that can scale and is not a pain in the ass to maintain later to add new features or extract analytics from it.

Query reading performance using **indexes** is a big plus.

We do not expect you to implement any cache or user auth but the design should have that in mind.

We don't expect you to implement testing.

Read all the documentation before starting designing since you will need as much context as possible to make decisions.

Glossary definition

User: a person performing actions. There's two levels of users, admins and counters.
Product: A collection of *Subproducts*, they have a price, name and category, a *Product* is composed of many *Subproducts*, you may need more than one *Subproduct* to complete a whole *Product*.
Subproduct: It has multiple *Barcodes to* identify it.
Barcode scanner: A numeric string value that correlates to a *Subproduct*.
CountPlan: *Users* can create multiple *CountPlans*, *CountPlans* will have many users that will be notified when the plan starts. *CountPlans* have an *User* as owner. Plans will have a repetition schedule (you can store this as you may need), and *CountPlans* will create *CountExecutions* periodically based on this repetition schedule.
CountExecution: *CountExecutions* are started by *CountPlan* periodically. They have a status (on going, started), and they will hold many *UserProductCounts*.
UserProductCounts: An *User* can add many counted *CountedSubproducts* to a *CountExecution, this* will consist of a *Subproduct* counted with a quantity attached to it, an *User* can count multiple times the same *Subproduct* with different quantities each time.
Example:
An *User* counts x9 *Subproducts* on a *CountExecution.*

- x2 *Subproducts* A
- x3 Subproducts B
- x4 *Subprocut* A

Total *Subproducts* will be x6 A's and x3 B's.

Notes

On *Product* pricing: A *Product* price is only completed when all the *Subproducts* are found. So, if you have a product with a $1 price that requires 2 *Subproducts* A and 1 *Subproducts* B but you find 17 A's and 9 B's the total price will be $8 since two of the *Products* are not completed.

API definition

1. **Endpoint that builds a *CountPlan* with a weekly schedule.**
2. **Endpoint that builds a *CountPlan* with a every 2nd Monday monthly schedule. (optional)**
3. **Endpoint to add *Products*.**
4. **Endpoint that checks current *CountPlan* and starts *CountExecution* if necessary based on schedule.**
5. **Add *UserProductCounts* to *CountExecution*, a *User* will count *Subproducts*.**
6. **End a *CountExecution* changing its status to end (no more *UserProductCounts* can be added).**

Notes

- All operations are for admin roles only. Adding *UserProductCounts* to *CountExecution* is allowed to counter and admin roles.
- All following result examples will always use the following *UserProductCounts* data.

An *User* counts x9 *Subproducts* on a *CountExecution.*

1. x3 *Subproduct* A
2. x4 *Subproduct B*
3. x4 *Subproduct* A
4. x1 *Subproduct* C
5. x2 *Subproduct* D

Total *Subproducts* will be x7 A, x4 B, x1 C and x2 D.

- *Product* AABB
  - Has a price of $1
  - Has Furniture category
  - Needs x2 *Subproduct* A and x1 *Subproduct* B
- *Product* CC
  - Has a price of $2
  - Has Furniture category
  - Needs x1 *Subproduct* C
- *Product* DD
  - Has a price of $3
  - Has Food category
  - Needs x1 *Subproduct* D

7. **Extract from a *CountExecution* the pricing per *Product* based on the counted quantity:**

Example result:

- *Product* AABB total price is $3
- *Product* CC total price is $2
- *Product* DD total price is $6

8. **Extract from a *CountExecution* the total pricing of all *Product* based on the counted quantity:**

Example result:

- Total *CountExecution* total price will be $11

9. **Extract from a *CountExecution* the pricing of *Product* by category:**

Example result:

- Total Food is $5
- Total Furniture is $6