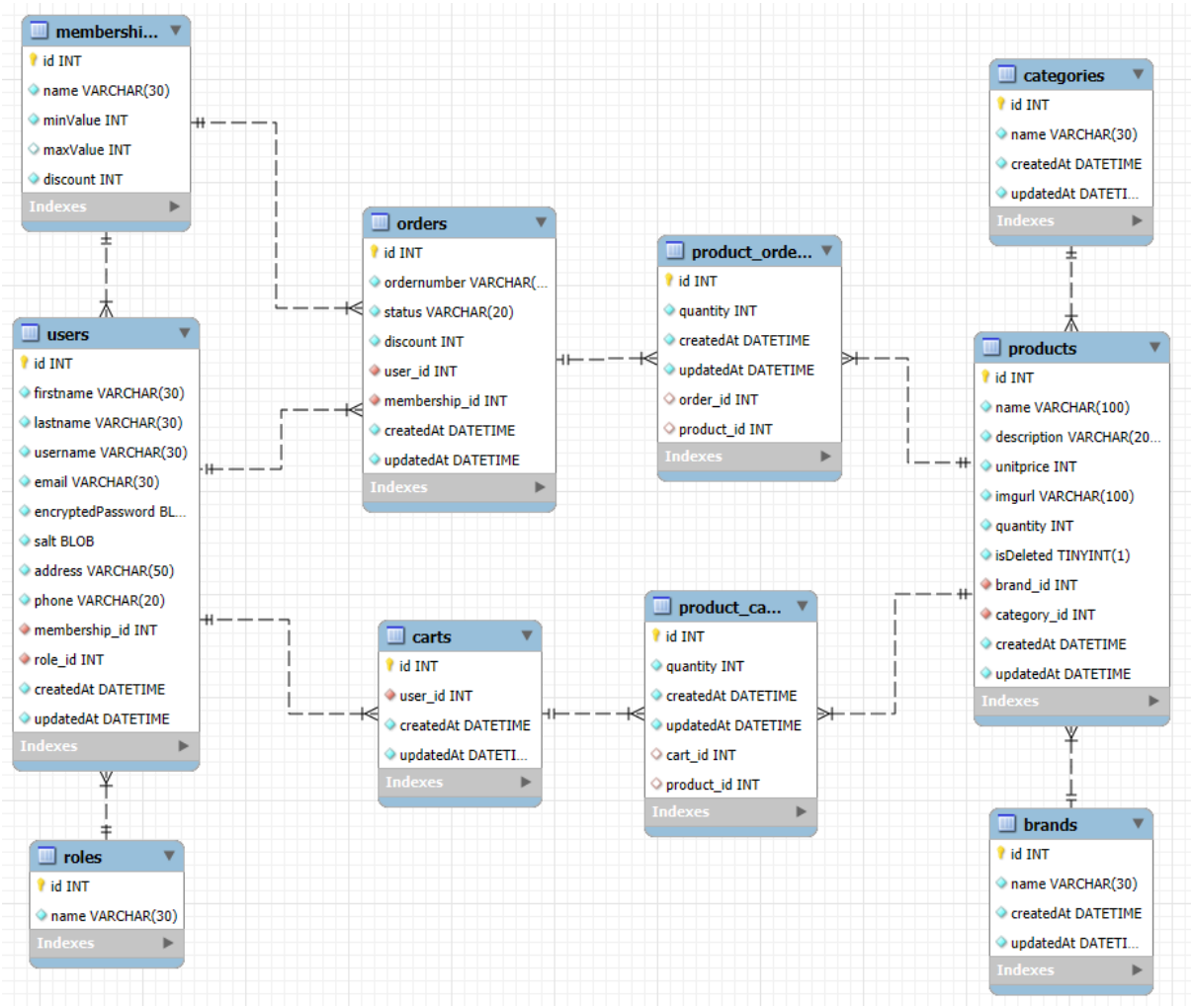
Reflection report

Database

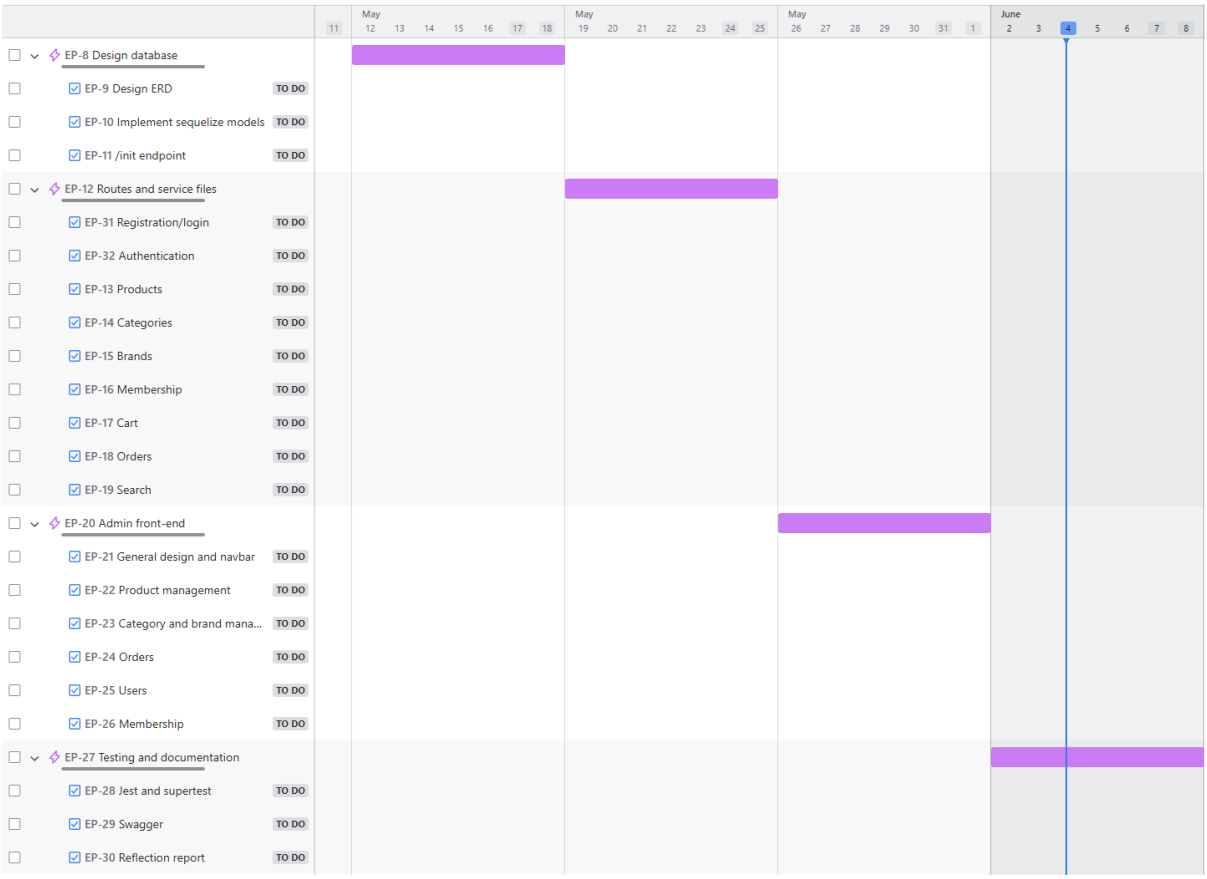


This diagram shows all the tables, properties and relationships in the database in third normal form. The relationships are as follows:

- A category and a brand can each have many products, while a product belongs to one category and one brand. Two one-to-many relationships.
- A role and a membership can have many users, and a user have one role and one membership. Two one-to-many relationships.
- A cart has one user, and a user can have one cart. A one-to-one relationship.
- An order has one user, and a user can have many orders. A one-to-many relationship.
- An order has one membership, and a membership can have many orders. A one-to-many relationship.

- A cart can have many products, and product can have many carts. This forms a many-to-many relationship, but through a linking table called product_cart.
- An order can have many products, and a product can have many orders. This forms a many-to-many relationship, but through a linking table called product_order.

Jira



This is the Jira timeline that shows all the tasks, epics and sprints. I planned four 1-week sprints for each epic:

- Week 1: Plan the project in Jira, design the database, and implement it with sequelize.
- Week 2: Implement the routes, endpoints and service files to complete the API.
- Week 3: Implement the admin front-end.
- Week 4: Finalize tests, write Swagger documentation and write the reflection report.

Progression

I used the first days of week 1 to plan the project. I started by dividing everything that needed to be done as tasks in Jira and group them into epics. I found this very helpful as it gave a nice oversight of the entire project and made it easier to plan from there. To design the database, i used an online ERD tool to draw it up and used that as a reference point when implementing the sequelize models.

I completed the first epic before time and got a head start on implementing the routes. This was a big part of the project, so it did take a little more than a week to get everything up and running. For each route i began with a service file with CRUD operations and created endpoints to use them. After completing a route and service file for a table, i set up a suite of tests using Jest and Supertest to make sure it worked as expected. I ended up with a lot of tests when finished, but the ones specified in the assignment are in a separate bulk named 'Assignment tests'.

When implementing the front-end, i decided to start with the products page as it had the most functionality, and then the other pages would be easier to implement. I started with filling the table with data and then implementing the edit/add/delete functions. Then i added the search functionality, and finally general design with Bootstrap. For the visual design i tried to make as similar as the example given in the instructions. When the products page was done i created a navbar for the other pages and mainly reused a lot of the same code in products to complete them.

In the final week the project was mainly done. I went over the all the files for minor adjustments and to clean up the code. Then i added Swagger documentation for all the endpoints. After a couple of days i had the remainder of the week for report-writing and any last changes that needed to be done.

Challenges

In the first week, i found some challenge in designing the database in 3NF. It was some back and forth with the relationships and linking tables, but once i had a full visual representation, it was easy implement.

When implementing the routes, the real challenge was when i got to /carts. You really had to think about the logic for what happens when a user adds products and checks out. It also had a lot of moving parts including users, products, memberships and orders. This was the part of the API i spent the most time on. The /search endpoint was also a little challenging as it had to use a query with three optional parameters which was tricky to figure out.

For the front-end, i spent some time figuring out the general structure and logic in communicating with the backend. The hardest part was adding/editing entries. This led to some backtracking in the API to optimize it for use in the frontend. But once the products view was complete, the other views was fairly easy. There was an issue when the logged in admin changed its own role to user and was blocked out, so this took some time to work around. I also had to relearn a lot of Bootstrap functionality for design and the modal forms.

The Swagger documentation was a little challenging as it's a little unclear how detailed it is expected to be. But once i had a structure i was happy with, it was easy to reuse in the other endpoints.

When working on a big project with many components like this, it can be very time-consuming to find the problem when errors happen. So i found it very helpful to have a well-defined error handler and a set of tests for each component. It made it easier to see where the problem was happening and which parts where affected when changes were made in the code.

.env file example

```
HOST=localhost
ADMIN_USERNAME=root
ADMIN_PASSWORD=password
DATABASE_NAME=ecommercedb
DIALECT=mysql
PORT=3000
ACCESS_TOKEN_SECRET=qwepoimnbzxc
```