# Brief

Design a new app to manage stock for a shop. The app will serve three main functions:

1. Display to the user what products are in the shop
2. Display the total amount of stock value the shop has (i.e. if there are only 3 carrots at a price of $2 each and one $3 aubergine, then the total stock value is $9)
3. Allow users to add and remove items of stock from the shop

# Setup

The project will live in your public GitHub repo under the folder name `Shop` and is expected to have the following folder structure:

`Shop`:
- `front` : This folder can be empty for now
- `back` : This is where the express app lives
-- `app.js` : This is where the Express app lives

The backend will be an `express` server written in `Node.js` on port 5000.

**Resources**
I would start with the data models. What is the "stock" exactly? What properties would an item have? It could look like:

```
{
    id: 0,
    name: 'Candyfloss',
    description: 'Sweet and tasty sugar',
    price: 12,
    currency: 'EUR',
    quantity: 350
}
```

Create an ERD (Entity-Relationship Diagram) using `draw.io` to detail the database tables, their rows and relationships to other tables. Once this is done, create the table structure in your Postgres instance.

Use `knex` to interact with the database in the BE APIs. Take a look at http://knexjs.org/ for more information on installation, connection and methods. Create a `database` folder in your BE and inside that create a `connection.js` folder that you will use to maintain a `knex` connection to the database for all interactions. The you should be able to import this `connection.js` to other files where you can run `select` and `update` statements on your resources.

# APIs

Once you have the data structures thought out, start to think about the APIs. Take a look here to get an idea of the basics and to see what HTTP status codes you should return with each response. Then take a look at the express docs. Start to think about how are we going to `GET` all the items? How will someone `POST` an item to the store and how will they `DELETE` one? How can we `GET` the total value of all the stock?

All responses should have the form:

```
{
    data: 'Mixed type holding the content of the response'
    message: 'Message describing what happened'
}
```

In your documentation, we can assume all response definitions will only detail the `data field`.

Example documentation could be:

## Get all stock

**Definition**

- `GET /api/stock`
  **Response**
- `200 OK` on success

```
[
    {
        id: 0,
        name: 'Candyfloss',
        description: 'Sweet and tasty sugar',
        price: 12,
        currency: 'EUR',
        quantity: 350
    },
    {
        id: 2,
        name: 'Stroopwaffel',
        description: 'Caramelly deliciousness',
        price: 1,
        currency: 'EUR',
        quantity: 2000
    }
]
```

For `POST` and `DELETE` requests, think about error handling and input validation. We don't want users adding/deleting the wrong things now do we.

**Resources**:

- Use express.js for the API server.
- Use `knex` to interact with the database in the BE apis. Take a look at http://knexjs.org/ for more information on installation, connection and methods.