Technical Assessment | Bosta

Database Design

The database designed was a relational Postgres database. It had two tables: products and suppliers. The suppliers have a one-to-many relationship to the products. For simplicity the concept of variants is encoded in the name of the product. We can then just search for a product or any variant using the `LIKE` clause in the SQL query.

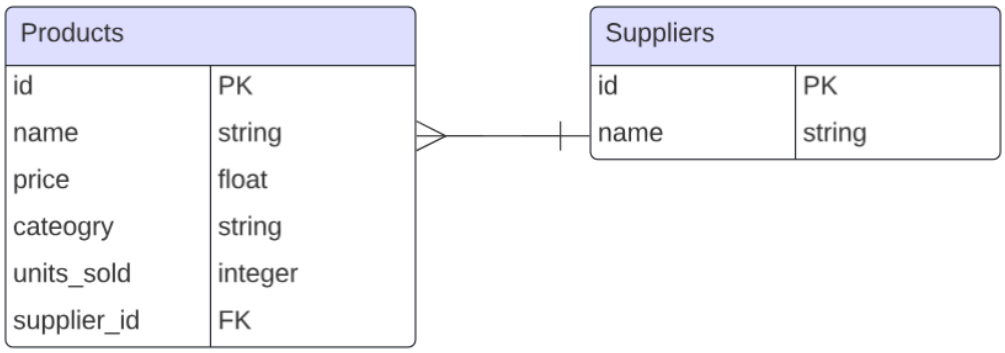

Figure 1 ERD Diagram for the Database

Infrastructure

The backend was designed following the MVC architecture with a product model, and a product controller that called the database and a view for the APIs. As requested, the only implemented endpoint was the search API which was labelled as a `GET` request and the endpoint expects a `JSON` object with the attributes to be filtered.

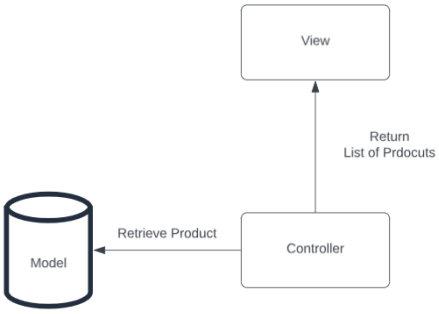

Figure 2 API Design follows MVC architecture

Questions

- **What would you change in the solution if the ranking will be based on most viewed items or the most/best reviewed?**
  We would add a field called viewed to the products table and increment it any time a product is in any query. The search query will be then ordered by the viewed field descending.
- **How would you think about linking relevant products together?**
  - **Supplier A has a Red T-shirt V-Neck**
  - **Supplier B has a Red T-shirt V-Neck & Green T-Shirt**
  - **How would the system know they are relevant to be recommended?**

  A possible solution would be to introduce a tag table. The tag table will have a many-to-many relationship with the products table. Tags would include “red”, “green”, “V-Neck”, etc. We can then try to try finding other products that have similar tags to the selected product which would be usually a relevant recommendation.

- **How would you think about securing such service?**
  The main threats of the search endpoint would be SQL injection attack, cross-side scripting, and denial of service attacks. To handle the first two attacks, the input should be sanitized first before being executed on the database. For the DoS attacks, a quota should be set on the number of requests allowed by a single user. Authorization might also be useful but will make the endpoint very restrictive as it forces anyone who wants to search the catalog to be registered.
- **What would be the type of servers needed for such like service? would it be RAM or CPU optimized types of machines and why?**
  The following search endpoint would require more CPU as there is not much memory needed as the database is read and rarely updated which means that there is no need to store local variables, but a faster CPU will help improve query speed and performance.