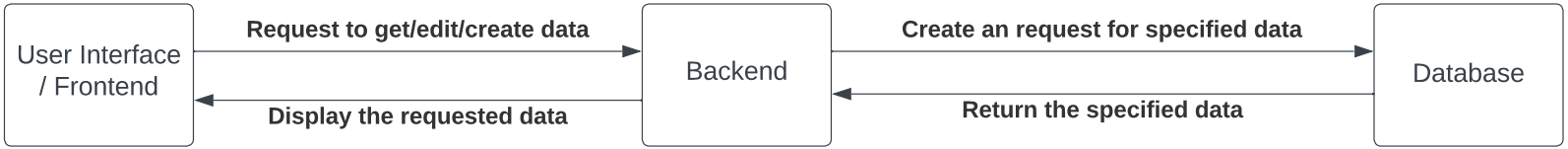

The user will be able to launch either a simple user interface created in java/python/c#/etc. or a more complex one using javascript/react/angular/etc. The second option would require starting a seperate npm server and creating API requests for the frontend and endpoints for the backend, which depending on the use case may be worth it when taking into account the extra design capabilities these languages and frameworks offer.

The backend will then have logic for the specified request of getting/creating/editing/etc. and make a request to the database. However, it is important to mention is that if the requests are only limited to getting, creating, editing, filtering. etc. having an intermediate backend step will not be necessary and it may be more efficient to directly make API requests from the front-end. However, if certain manipulation of the data may be necessary at some point that is not achievable using an SQL query, the backend will in many cases be a more efficient place to do that than on the frontend, for exampe: running a Machine Learning model on the data that will accurately predict when then next resupply moment should be, and then add that as a column value.

Lastly, the database can be queried either locally or an API request can be made to a cloud database. A cloud database would be more practical if the inventory system has to be used concurrently by multiple users, across multiple machines. However, in a simple scenario where a small company has its inventory altered at most once every day or less by a single user, storing this database locally would be easier and cheaper.