Design Document

Overview

The application is a desktop application which helps a person to create customer / product and sale order on behalf of users. The UI shown here merely to illustrate the underlying functionality. The actual look and feel will be developed over time with the input of graphics designers and iterative user feedback.

The specs does not cover entire functionalty this will only provide high level view and how entities are related to each other.

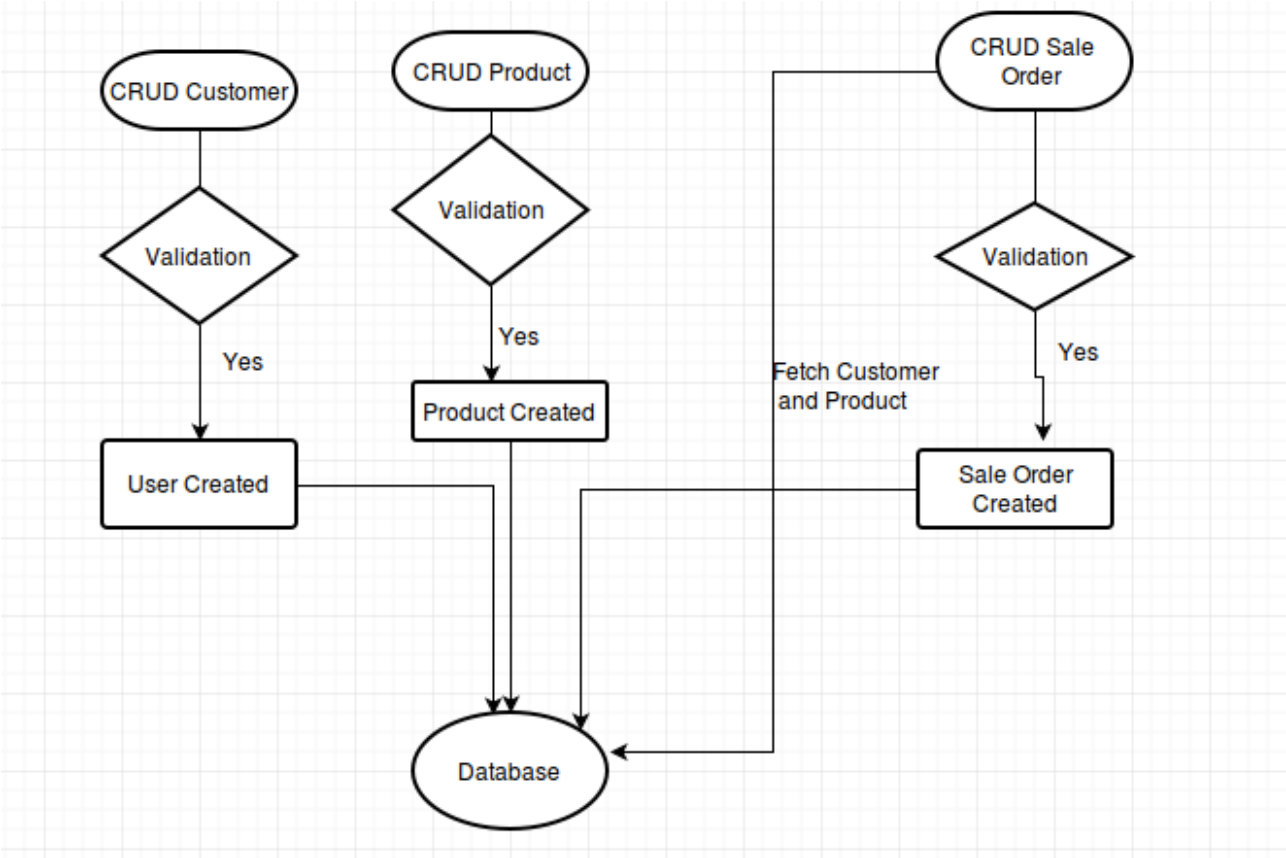
Scenarios

1. User can perform CRUD operations on Customer.
2. User can perform CRUD operations on Product.
3. User can perform CRUD operations on Sale Order.

Non Goals

1. Because of time limitation cache is not implmented for faster response.
2. Most of the validations is not added because of time limitation.
3. Issue in front end are not handled as per the information provided.

Flow Chart

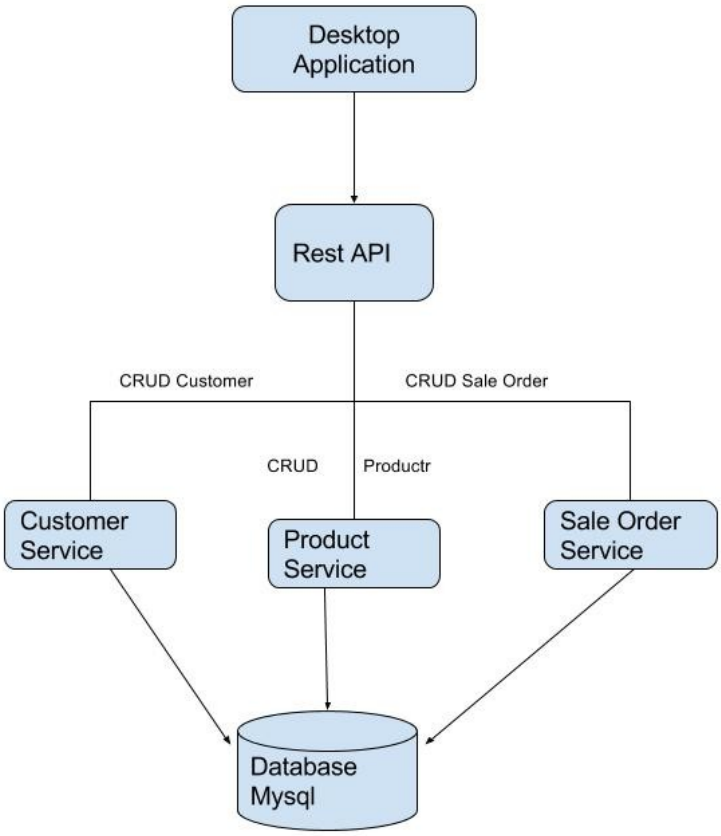


High Level Diagram

Back system was needed for desktop application where an admin can create Customer, Product and Sale Order on behalf of the User. In the process we need to maintain the concurrency as well as consistency of Credit Limit and Quantity of the Product.

For scalability and faster devlopment Spring boot is the best so in the development of the system we have used Spring Boot / Hibernare / MySQL.

Front end will be communicating to backend using RestTemplate over the HTTP.



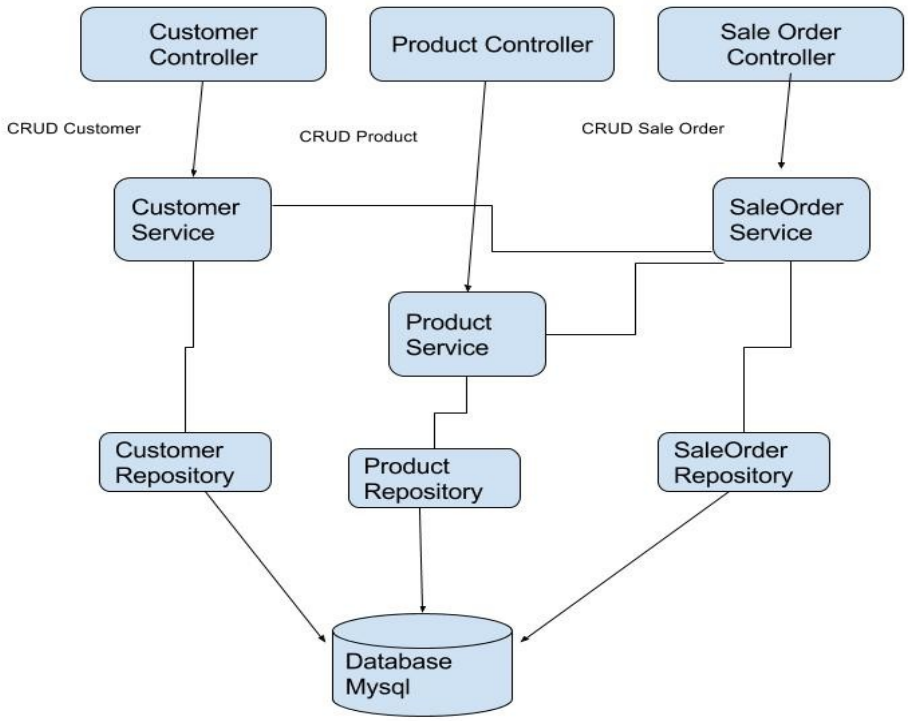
Components

- **Controllers :** For each entity controllers are created which will provide basic CRUD operations.
    - CustomerController
    - ProductController
    - SaleOrderController

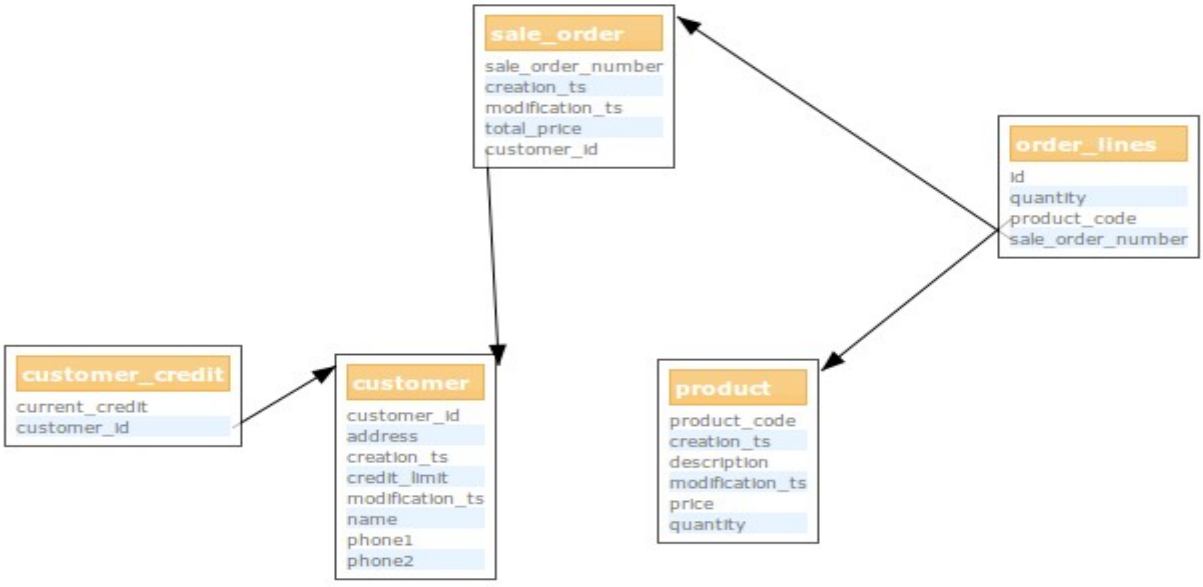
- **Services [Business Logic] :** Every controller will interact with services which contains business logic. Singleton Design + Factory Design pattern are used.
    - CustomerService

        - ProductService
        - SaleOrderService

- **Repositories [Persistence] :** This will save and fetch data from the database.
        - `CustomerRepository`
        - `CustomerCreditRepository`
        - `OrderLinesRepository`
        - `ProductRepository`
        - `SaleOrderRepository`
- **Junit Test Cases :** For each service separate test cases are added.
        - **ProductTest**
            - Creation of Product
            - Deletion Of Product
            - Fetch Product
        - **CustomerTest**
            - Creation Of Customer
            - Deletion Of Customer
            - Fetch Customer
        - **SaleOrderTest**
            - Creation Of Customer
            - Creation Of Product
            - Creation of SaleOrder
            - Fetch SaleOrder
            - Delete Sale Order



Database Schema



Benefits, assumptions, risks/issues

This will perform basic CRUD operations but there might be threading related issues when creating sale order and modifyig CreditLimit and Quantity of the Product.

Conclusion

Functionalities were not easy to implment but there is lot of scope to improve them. The difficult part is working through a logical design before you get to coding. Once you have a vision of how the objects and entities are arranged, writing the details is easy