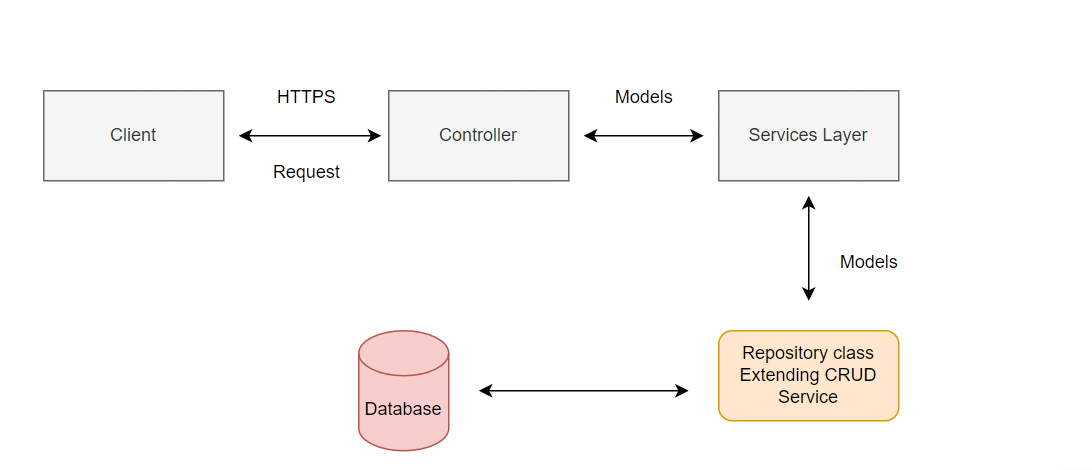
High Level Design Document



Different Components Of Trade Aggregation

Models: Trade, CancelTrade, Party, Institution

Services: Business logic for aggregating the Trade and placing trades in CancelTrade table, Institution, Party and Trade will execute the functionalities of CRUD operations

Controller: API Request and Response Handler

Repository/Dao: Validation of a trade, used for CRUD operation, searching for a trade using partyId and partRefNum

Swagger: Connected to controller for implementing API requests

API 1 (Create a New Trade):

* Post request from swagger will be handled through controller and based on logic in service class, trade data will be saved in the database.
* Validation is done before saving the data in database.
* Party and Counterparty must be valid.
* **Validation will be of following type:**

1. Based on valid Party and Institution names in the database.
2. Based on valid trade dates, notional amount.
3. Party and Counterparty can’t be same.
4. Status will be unconfirmed by default for an unmatched trade.
5. Institution of Party and Counterparty must be different.

* Matched trade set to CANCELED and in place of that a single, aggregated trade is created with updated fields.

**For Aggregation:**

Aggregation will be done on the basis of (Party,Counterparty,tradedate,effectivedate,instrumentId,MaturityDate,Currency,seller,buyer,status).

If all the fields match aggregate the Notional Amount and create a single aggregated trade in the Trade table.

**API 2(Search For a trade):**

* Based on party name and status: Search all the trades of a party with provided status. (Input string => cancel, unconfirmed)
* Based on party name and trade reference number:
* If user input invalid string or null values: throw custom error.

**API 3(Find all cancelled trades from given aggregated trade):**

To find out all the cancelled trades which have satisfied the specified criteria of matching field values to form an aggregated trade.