**Summary**:

Code was located at: git clone <https://github.com/dinhanhvo/augen_test.git>

**Environment require:**

1. Java 8

2. Maven

3. Node

**Run Server:**

*cd augen\_test\augen\_beckend*

*mvn clean package install*

*mvnw spring-boot:run*

*server will run at port 8089*

**Run Client:**

*cd augen\_fontend*

*yarn || npm i*

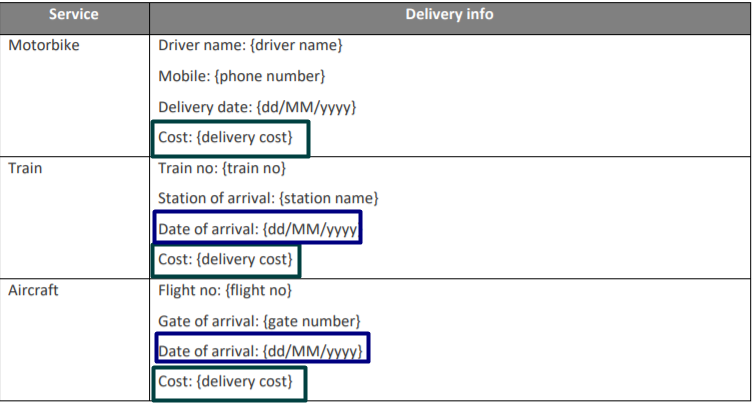
*ng serve --port 4300*

*(\*) run at port 4300 since ORIGIN configuration at Server*

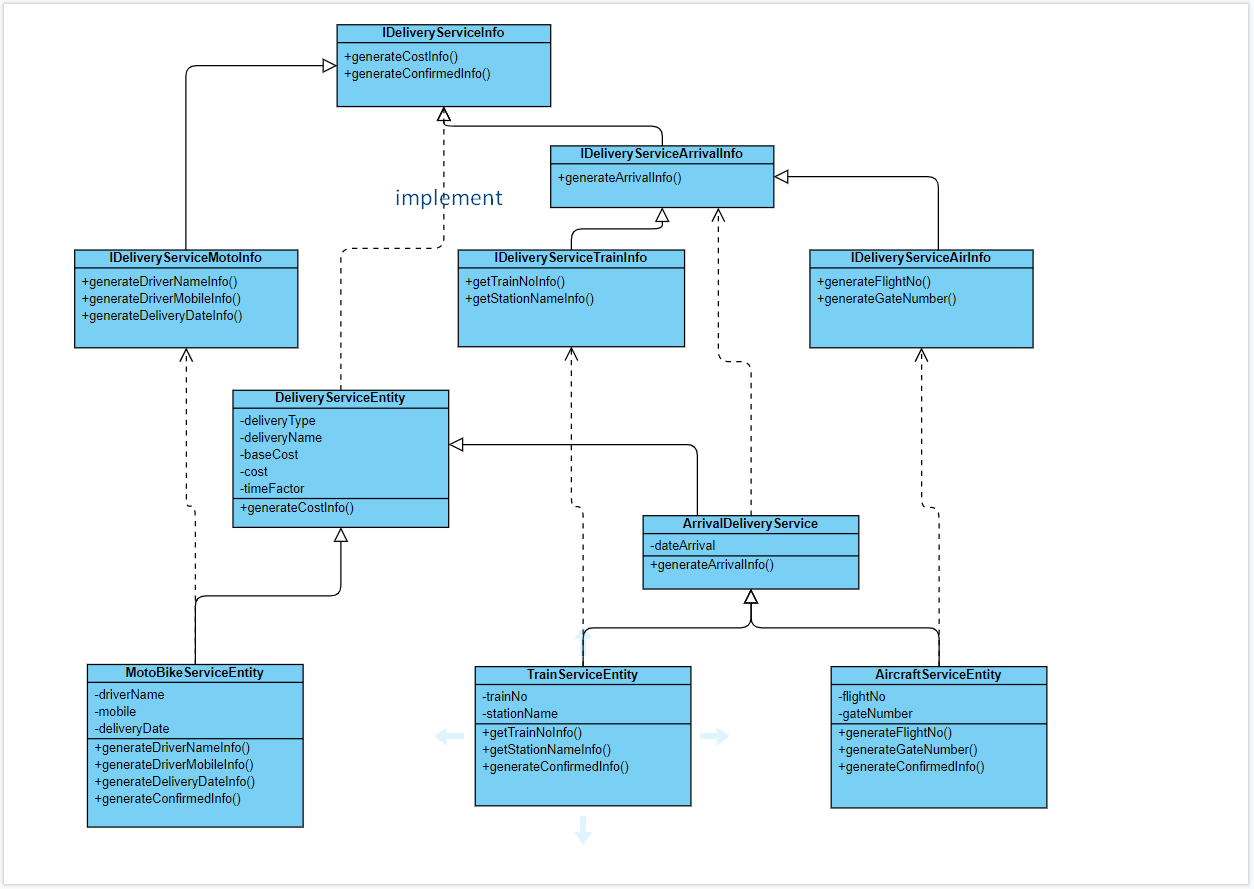
Client will run at port 4300, open “User Document.docs” to buy a book 😊

1. **Classes Design at Back-End:**

* From the table requirement



* Design for Classes: Class Diagram

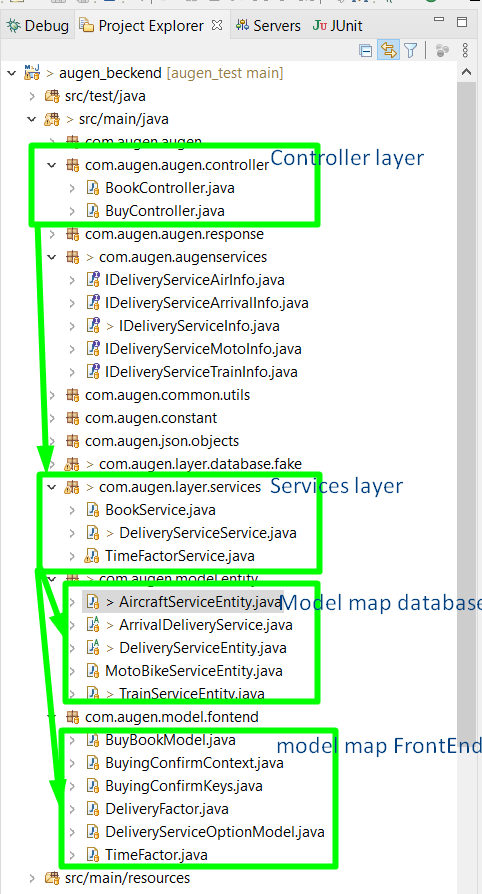


With common property:

* deliveryType: is type of delivery service, values are: [1, 2, 3]
* deliveryName: is name of delivery service, values are: [“Motobike”, “Train”, “Aircraft”]
* baseCost: is common attribute

Reasons:

* Each class [MotoBikeServiceEntity, TrainServiceEntity, AircraftServiceEntity]:
  + Just contains its attributes
  + Reuse code of its parent class
  + Responsibility and do not forget to implement methods from the interface.
* No redundant or duplicate code
* Separate to three class to loosely coupling. If in future there are more delivery service just add more class by extends and implement from existed classes.

1. **Explorer Code Tree:**

* Controller layer responsibility to handle api
* Service layer:

+ do business

+ get data to form

* Model to map database: contain models are stored from database (this test we fake database)
* Model to map to Front-End: contain models which response to Front-End and need to parse to Json

1. **Front-End Design:**

* Use the <app-book-card> template to reuse on home page and dialog with params.
* BookService and BuyService will call api to get data from Back-End
* Models to map with Front-End

🡺 **Reason to use Back-End and Front-End designs:**

* Clean code
* Easy to maintain
* Easy to scale