# CSCI318 - Task 2

## Contents

# CSCI318 - Task 2

## Dependencies

- Spring Boot 2.5.3
- Java 8
- Spring Web
- Spring Data JPA
- H2

## Building

```
cd csci318-project/customer-service && mvn compile && mvn package
cd ../ordering-service && mvn compile && mvn package
cd ../product-service && mvn compile && mvn package
```

## Running

```
java -jar customer-service-0.0.1-SNAPSHOT.jar
java -jar ordering-service-0.0.1-SNAPSHOT.jar
java -jar product-service-0.0.1-SNAPSHOT.jar
```

**Note:**

- Customer Service runs on Port 8082
- Ordering Service runs on Port 8081
- Product Service runs on Port 8080

# Description of the project

## Structure

```
$ tree -A ./csci318-project
csci318-project
├── customer-service
│   ├── mvnw
│   ├── mvnw.cmd
│   ├── pom.xml
│   └── src
│       ├── main
│       │   ├── java
│       │   │   └── onlineordering
│       │   │       └── customerservice
│       │   │           ├── api
│       │   │           │   ├── ContactController.java
│       │   │           │   ├── ContactExceptionController.java
│       │   │           │   ├── CustomerController.java
│       │   │           │   └── CustomerExceptionController.java
│       │   │           ├── CustomerServiceApplication.java
```

```
│       │   │           ├── model
│       │   │           │   ├── Contact.java
│       │   │           │   ├── ContactNotFoundException.java
│       │   │           │   ├── Customer.java
│       │   │           │   └── CustomerNotFoundException.java
│       │   │           ├── repository
│       │   │           │   ├── ContactRepository.java
│       │   │           │   └── CustomerRepository.java
│       │   │           └── service
│       │   │               ├── ContactService.java
│       │   │               └── CustomerService.java
│       │   └── resources
│       │       └── application.properties
│       └── customer-service-0.0.1-SNAPSHOT.jar
├── Documentation.pdf
├── LICENSE
├── ordering-service
│   ├── mvnw
│   ├── mvnw.cmd
│   ├── pom.xml
│   ├── src
│   │   └── main
│   │       ├── java
│   │       │   └── onlineordering
│   │       │       └── orderingservice
│   │       │           ├── api
│   │       │           │   └── OrderController.java
│   │       │           ├── model
│   │       │           │   ├── OrderEvent.java
│   │       │           │   └── Order.java
│   │       │           ├── OrderingServiceApplication.java
│   │       │           ├── repository
│   │       │           │   └── OrderEventRepository.java
│   │       │           └── service
│   │       │               ├── OrderEventHandler.java
│   │       │               └── OrderService.java
│   │       └── resources
│   │           └── application.properties
│   └── target
│       └── ordering-service-0.0.1-SNAPSHOT.jar
├── product-service
│   ├── src
│   │   └── main
│   │       ├── java
│   │       │   └── onlineordering
│   │       │       └── productservice
│   │       │           ├── api
```

```
│   │       │           │   ├── ProductController.java
│   │       │           │   ├── ProductDetailController.java
│   │       │           │   ├── ProductDetailExceptionController.java
│   │       │           │   └── ProductExceptionController.java
│   │       │           ├── model
│   │       │           │   ├── ProductDetail.java
│   │       │           │   ├── ProductDetailNotFoundException.java
│   │       │           │   ├── Product.java
│   │       │           │   └── ProductNotFoundException.java
│   │       │           ├── ProductServiceApplication.java
│   │       │           ├── repository
│   │       │           │   ├── ProductDetailRepository.java
│   │       │           │   └── ProductRepository.java
│   │       │           └── service
│   │       │               ├── ProductDetailService.java
│   │       │               └── ProductService.java
│   │       └── resources
│   │           └── application.properties
│   └── target
│       └── product-service-0.0.1-SNAPSHOT.jar
└── README.md
```

The onlineordering service is made up from three separate application with its own bounded context.

These applications follow the N-Tier Architecture which is split into API Layer, Service Layer, Data Access Layer and Database.

The API Layer handles REST Requests and calls Service Functions in the Service Layer. The Service Layer also consists of Domain Models specific to the application. The Service Layer handle function calls and access the Data Access Layer which has repositories specific to the domains models. A repository will then access entries or save data into the database as required.

# Order Service

## API Layer

### OrderController

The order controller only has one functionality which is `addOrder`.A JSON Object with the Order is required to be passed into the REST Request.

The Order Object should have the keys: - supplier - productName - quantity

## Service Layer

The Order Domain has two models, Order and OrderEvent.

The OrderEvent stores the following variables: - productName - orderQuantity - stockQuantity - productPrice

#### OrderService

The `addOrder()` function requires a customerId and a JSON Object which makes up the Order Object.

This service uses RestTemplate to make REST Requests. The first REST Request checks if a customer with the customerId is present. The second checks if the product with the productName stored in the Order Object is in stock. This service publishes a OrderEvent if the customerId is valid and the product is in stock or has sufficient stock.

#### OrderEventHandler

This service handles events published which consists of a OrderEvent. The `handle()` function accepts an OrderEvent Object. This service also uses RestTemplate which makes a PUT request to update the stock on the Product Application.

### Data Access Layer

The data access layer for this application saves the previous order events.

## Product Service

### API Layer

#### ProductController

This Class has the following functionality:

- addProduct()
  - Returns product and accepts a JSON body which takes in the following keys:
    - productCategory
    - productName
    - price
    - stockQuantity
- findAllProduct()
  - Returns all products stored in the database.
- findProductById()
  - Returns the product with the “{id}” specified in the request.
- deleteProductById()
  - Deletes the product with the “{id}” specified in the request.
- updateStock()
  - Updates the stock of the product by name and accepts the JSON body with the key:
    - stockQuantity
- updateProductById()
  - Updates the product with the new product specified in the JSON body.
- addProductDetail()
  - Links the product with the product detail using two ids.

#### ProductDetailController

The class has the following functions:

- addProductDetail()
  - Returns the product detail and accepts a JSON body with the following keys:
    - description
    - comment

- findAllProductDetail()
  - Returns all product detail items stored in the database.
- findProductDetailById()
  - Returns the specified product detail with “{id}” specified in the request.
- updateProductDetailById()
  - Returns the new product detail. Updates the product detail with the parameters in the JSON body.
- deleteProductDetailById()
  - Deletes the product detail with the “{id}” from the database.

### Service Layer

In the Product Domain there are two models, Product and Product Detail. The Services for these models respectively implements the functionality specified in the section above.

### Data Access Layer

The repository interfaces extends the JpaRepository. The project uses the default functions. ProductRepository has a function `findByProductName()` which enables search for product by name.

## Customer Service

This service handles requests that are in the Customer Domain.

### API Layer

There are two classes in the API Layer, ContactController and CustomerController. ContactController accepts REST Requests with the path set to ‘/contact’. This class handles REST Requests and has the following functionality:

#### ContactController

- getAllContact()
  - Returns all the contacts stored in the database.
- getContactById()
  - Returns the contact stored in the database which matches the “{id}” specified in the request.
- addContact()
  - Returns the new contact after accepting a JSON body which takes these following keys:
    - name
    - phone
    - email
    - position
- deleteContactById()
  - Deletes the contact specified by “{id}” in the database.
- updateContactById()
  - Accepts a JSON body which is similar to addContact() and updates the contact in the database which matches the specified “{id}”. The function returns the updated contact.

#### CustomerController

The CustomerController class accepts REST requests with the path ‘/customer’ and has the following functionality:

- getAllCustomer()
  - Returns all the customers stored in the database.
- addCustomer()
  - Returns the new customer after accepting a JSON body which takes in the following keys:
    - companyName
    - address
    - country
- getCustomerById()
  - Returns the customer which matches the “{id}” specified.
- getCustomerByCompanyName()
  - Requires the path ‘/customer/company’ and the search param ‘name’.
  - Returns the customer which matches the search parameter.
- updateCustomer()
  - Accepts a JSON body similar to addCustomer() and updates the customer in the database which matches the “{id}” specified in the path.
- addCustomerContact()
  - Links an existing customer with a contact which exists in the database.
  - Requires specifying with ‘/customer/{id}/contact/{contactId}’
  - Returns the updated customer.

### Service Layer

In the Customer Domain, two models created for this task, Customer and Contact.

A Customer model has the following variables:

- companyName
- address
- country
- contact

A Contact model has the following variables:

- name
- phone
- email
- position
- customer

### Data Access Layer

The repositories for this task extends on the JpaRepository from the Spring framework. Thus, the default function such as save(), findById() and delete() are used.

The CustomerRepository has an extra definition which enables the application to search the database when provided the search term. This is implemented as the function findByCompanyName() and returns a List of the matches.

### Database

The project currently uses the H2 in memory database.

## Examples of Input and Output

### Customer Service

**Request 1: Creating a new customer entry.**

- Input:

```
curl -X POST -H "Content-Type:application/json" -d \
    '{
        "companyName":"Microsoft",
        "address" : "Redmond",
        "country" : "USA"
    }' http://localhost:8082/customer
```

- Output:

```
"id":1,"companyName":"Microsoft","address":"Redmond","country":"USA","contact":null}
```

**Request 2: Creating a new contact entry.**

- Input:

```
curl -X POST -H "Content-Type:application/json" -d \
    '{
        "name" : "Bill Gates",
        "phone" : "04555555555",
        "email" : "thisisgates@gmail.com",
        "position" : "CEO"
    }' http://localhost:8082/contact
```

- Output:

```
{"id":2,"name":"Bill Gates","phone":"04555555555","email":"thisisgates@gmail.com",
"position":"CEO"}
```

**Request 3: Linking a Customer and a Contact.**

- Input:

```
curl -i -X PUT http://localhost:8082/customer/1/contact/2
```

- Output:

```
{"id":1,"companyName":"Microsoft","address":"Redmond","country":"USA",
"contact":{"id":2,"name":"Bill Gates","phone":"04555555555","email":"thisisgates@gmail.com",
"position":"CEO"}}
```

**Request 4: Updating a specific contact by ID.**

- Input:

```
curl -i -X PUT -H "Content-Type:application/json" -d \
    '{
        "name" : "Tim Cook",
        "phone" : "04555555557",
        "email" : "thisistim@hotmail.com",
        "position" : "CEO",
        "customer" : "Apple"
    }' http://localhost:8082/contact/2
```

- Output:

```
{"id":2,"name":"Tim Cook","phone":"04555555557","email":"thisistim@hotmail.com",
"position":"CEO"}
```

**Request 5: Get all customers in the database.**

- Input:

```
curl -X POST -H "Content-Type:application/json" -d \
    '{
        "companyName":"Apple",
        "address" : "California",
        "country" : "USA"
    }' http://localhost:8082/customer

curl -X GET http://localhost:8082/customer
```

- Output:

```
{"id":3,"companyName":"Apple","address":"California","country":"USA","contact":null}
[{"id":1,"companyName":"Microsoft","address":"Redmond","country":"USA",
"contact":{"id":2,"name":"Tim Cook","phone":"04555555557","email":"thisistim@hotmail.com",
"position":"CEO"}},{"id":3,"companyName":"Apple","address":"California",
"country":"USA","contact":null}]
```

**Request 6: Get a specific customer in the database.**

- Input:

```
curl -X GET http://localhost:8082/customer/3
```

- Output:

```
{"id":3,"companyName":"Apple","address":"California","country":"USA","contact":null}
```

**Request 7: Search for a customer by company name.**

- Input:

```
curl -X GET http://localhost:8082/customer/company?name=Apple
```

- Output:

```
[{"id":3,"companyName":"Apple","address":"California","country":"USA","contact":null}]
```

**Request 8: Get all contacts in the database.**

- Input:

```
curl -X POST -H "Content-Type:application/json" -d \
    '{
        "name" : "Bill Gates",
        "phone" : "04555555555",
        "email" : "thisisgates@gmail.com",
        "position" : "CEO"
    }' http://localhost:8082/contact
```

```
curl -X GET http://localhost:8082/contact
```

- Output:

```
{"id":4,"name":"Bill Gates","phone":"04555555555","email":"thisisgates@gmail.com",
"position":"CEO"}[{"id":2,"name":"Tim Cook","phone":"04555555557",
"email":"thisistim@hotmail.com","position":"CEO"},{"id":4,"name":"Bill Gates",
"phone":"04555555555","email":"thisisgates@gmail.com","position":"CEO"}]
```

**Request 9: Get a specific contact in the database.**

- Input:

```
curl -X GET http://localhost:8082/contact/4
```

- Output:

```
{"id":4,"name":"Bill Gates","phone":"04555555555","email":"thisisgates@gmail.com",
"position":"CEO"}
```

## Product Service

**Request 1: Post Product Detail.**

- Input:

```
curl -X POST -H "Content-Type:application/json" -d \
    '{
        "description" : "A pretty cool chocolate cookie.",
        "comment" : "Might have peanuts."
    }' http://localhost:8080/product/detail
```

- Output:

```
{"id":1,"description":"A pretty cool chocolate cookie.","comment":"Might have peanuts."}
```

**Request 2: Post Product.**

- Input:

```
curl -X POST -H "Content-Type:application/json" -d \
    '{
        "productCategory" : "Confectionery",
        "productName" : "Triple Chocolate Cookie",
        "price" : "100",
        "stockQuantity" : "5"
    }' http://localhost:8080/product
```

- Output:

```
{"id":2,"productCategory":"Confectionery","price":100,"stockQuantity":5,"productDetail":null,
"name":"Triple Chocolate Cookie"}
```

**Request 3: Get Product Detail.**

- Input:

```
curl -X GET http://localhost:8080/product/detail/
```

- Output:

```
[{"id":1,"description":"A pretty cool chocolate cookie.","comment":"Might have peanuts."}]
```

#### Request 4: Get Product

- Input:

```
curl -X GET http://localhost:8080/product/
```

- Output:

```
{"id":2,"productCategory":"Confectionery","price":100,"stockQuantity":5,"productDetail":null,
"name":"Triple Chocolate Cookie"}
```

#### Request 5: Link Product - Product Detail

- Input:

```
curl -i -X PUT http://localhost:8080/product/2/detail/1
```

- Output:

```
{"id":2,"productCategory":"Confectionery","price":100,"stockQuantity":5,"productDetail":{"id":1,
"description":"A pretty cool chocolate cookie.","comment":"Might have peanuts."},
"name":"Triple Chocolate Cookie"}
```

#### Request 6: Update Stock

- Input:

```
curl -i -X PUT -H "Content-Type:application/json" -d \
    '{
        "stockQuantity" : "5"
    }' "http://localhost:8080/product/update?name=Triple%20Chocolate%20Cookie"
```

- Output:

```
{"id":2,"productCategory":"Confectionery","price":100,"stockQuantity":5,"productDetail":{"id":1,
"description":"A pretty cool chocolate cookie.","comment":"Might have peanuts."},
"name":"Triple Chocolate Cookie"}
```

### Order Service

#### Request 1: Post Order

- Input:

```
curl -X POST -H "Content-Type:application/json" -d \
    '{
        "supplier" : "Coles",
        "productName" : "Triple Chocolate Cookie",
        "quantity" : "2"
    }' http://localhost:8081/order/customer?id=1
```

- Output:

```
Customer Address : "Redmond"
Customer Contact : "04555555555"
Unit Price : 100
Unit(s) Ordered : 2
```

**Request 2: Post Order without Valid Customer**

- Input:

```
curl -X POST -H "Content-Type:application/json" -d \
    '{
        "supplier" : "Coles",
        "productName" : "Triple Chocolate Cookie",
        "quantity" : "2"
    }' http://localhost:8081/order/customer?id=5
```

- Output:

```
Customer with id : 5 is Not Found
```

**Request 3: Post Order without Sufficient Stock**

- Input:

```
curl -X POST -H "Content-Type:application/json" -d \
    '{
        "supplier" : "Coles",
        "productName" : "Triple Chocolate Cookie",
        "quantity" : "20"
    }' http://localhost:8081/order/customer?id=1
```

- Output:

```
Customer Address : "Redmond"
Customer Contact : "04555555555"
Not Enough Stock
```

**Request 4: Post Order without Valid Product**

- Input:

```
curl -X POST -H "Content-Type:application/json" -d \
    '{
        "supplier" : "Coles",
        "productName" : "Better Multiple Chocolate Cookie",
        "quantity" : "2"
    }' http://localhost:8081/order/customer?id=1
```

- Output:

```
Customer Address : "Redmond"
Customer Contact : "04555555555"
Product Not Found
```