Table of Contents

[1](#_heading=h.gjdgxs) Introduction 2

[2](#_heading=h.30j0zll) getQuote 2

[2.1](#_heading=h.1fob9te) Rate data 2

[2.2](#_heading=h.3znysh7) Request 3

[2.3](#_heading=h.2et92p0) Response 4

[2.4](#_heading=h.tyjcwt) Process 4

[3](#_heading=h.3dy6vkm) createShipment 4

[3.1](#_heading=h.1t3h5sf) Request 4

[3.2](#_heading=h.4d34og8) Response 5

[3.3](#_heading=h.2s8eyo1) Process 5

[4](#_heading=h.17dp8vu) getShipment 5

[4.1](#_heading=h.3rdcrjn) Request 5

[4.2](#_heading=h.26in1rg) Response 6

[4.3](#_heading=h.lnxbz9) Process 7

[5](#_heading=h.35nkun2) deleteShipment 7

[5.1](#_heading=h.1ksv4uv) Request 7

[5.2](#_heading=h.44sinio) Response 7

[5.3](#_heading=h.2jxsxqh) Process 7

[6](#_heading=h.z337ya) trackShipment 8

[6.1](#_heading=h.3j2qqm3) Request 8

[6.2](#_heading=h.1y810tw) Response 8

# Introduction

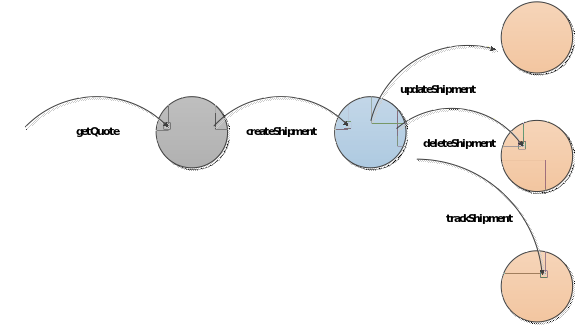
We need to use NodeJS to complete this exercise, by cloning the master branch from this repo: <https://gitlab.com/sondn.vinsofts/vinsofts-js-day-8>.

Test coverage must be at least 90%

Merge request to master branch must be created when exercise is finished

The idea is to simulate a simple shipping workflow with below **REST APIs**,

* getQuote
* createShipment
* getShipment
* deleteShipment



# getQuote

## Rate data

|  |  |
| --- | --- |
| **Weight** | **Price** |
| <= 250 g | 12,43 USD |
| 500 g | 12,43 USD |
| 750 g | 15,42 USD |
| 1 000 g | 15,42 USD |
| 2 000 g | 20,77 USD |
| 3 000 g | 26,07 USD |
| 4 000 g | 31,43 USD |
| 5 000 g | 36,77 USD |
| 6 000 g | 42,13 USD |
| 7 000 g | 47,49 USD |
| 8 000 g | 52,83 USD |
| 9 000 g | 58,83 USD |
| 10 000 g | 63,54 USD |
| 11 000 g | 88,19 USD |
| 12 000 g | 88,19 USD |
| 13 000 g | 88,19 USD |
| 14 000 g | 88,19 USD |
| 15 000 g | 88,19 USD |
| > 15 000 g | 100 USD |

Load rate data into “rate” collection in MongoDB, this data will be used to query shipment price.

## Request

POST /client/getquote HTTP/1.1

{

"data": {

"origin": {

"contact": {

"name": "La Redoute Contact",

"email": "laredoute@example.com",

"phone": "07 1234 5678",

},

"address": {

"country\_code": "FR",

"locality": "Anzin",

"postal\_code": "59410",

"address\_line1": "Rue Jean Jaures",

}

},

"destination": {

"contact": {

"name": "Marquise de Pompadour",

"email": "marquise-de-pompadour@example.com",

"phone": "07 9876 5432",

},

"address": {

"country\_code": "FR",

"locality": "Marseille",

"postal\_code": "13006",

"address\_line1": "175 Rue de Rome"

}

},

"package": {

"dimensions": {

"height": 10,

"width": 10,

"length": 10,

"unit": "cm"

},

"grossWeight": {

"amount": 100,

"unit":kg,

}

}

}

}

}

## Response

{

"data": [

{

"id": "\_\_QUOTE\_ID\_\_",

"amount": 100, // USD

}]

}

## Process

* Verify that the request conforms to schema, country\_code must be a valid ISO 3166-1 alpha-2 country code
* Query rate based on “rate” table.
* Origin country, destination country, weight fields are required
* Unit tests are required

# createShipment

## Request

POST /client/creatshipment HTTP/1.1

{

"data": {

"quote": {

"id": "\_\_QUOTE\_ID\_\_"

},

"origin": {

"contact": {

"name": "La Redoute Contact",

"email": "laredoute@example.com",

"phone": "07 1234 5678",

},

"address": {

"country\_code": "FR",

"locality": "Anzin",

"postal\_code": "59410",

"address\_line1": "Rue Jean Jaures",

"organisation": true

}

},

"destination": {

"contact": {

"name": "Marquise de Pompadour",

"email": "marquise-de-pompadour@example.com",

"phone": "07 9876 5432",

},

"address": {

"country\_code": "FR",

"locality": "Marseille",

"postal\_code": "13006",

"address\_line1": "175 Rue de Rome",

"organisation": false

}

},

"package": {

"dimensions": {

"height": 10,

"width": 10,

"length": 10,

"unit": "cm",

},

"grossWeight": {

"amount": 100,

"unit":kg,

}

}

}

}

}

## Response

{

"data": [

{

"ref": "\_\_REFERENCE\_NUMBER\_\_", // Random number in 10 character

"created\_at": "2015-05-13T07:00:08+0000",

"cost": 10.00 // USD

}]

}

## Process

* Request must be validated against schema
* Shipment needs to be quoted again in case the rate has been changed
* Request data needs to be stored in “shipment” collection, it will then be used to lookup for other requests.
* All the request fields are required
* For the response
  + “ref” field will be generated randomly and store with the shipment
  + “cost” is the price of the shipment, this is queried by origin country, destination country and weight of the shipment.
* Unit tests are required

# getShipment

## Request

GET /client/getshipment HTTP/1.1

{

"data": {

"ref": "\_\_REFERENCE\_NUMBER\_\_", // Use to lookup the shipment

}

}

## Response

**Shipment found:**

{

"data": {

"ref" : ""\_\_REFERENCE\_NUMBER\_\_",

"origin": {

"contact": {

"name": "La Redoute Contact",

"email": "laredoute@example.com",

"phone": "07 1234 5678",

},

"address": {

"country\_code": "FR",

"locality": "Anzin",

"postal\_code": "59410",

"address\_line1": "Rue Jean Jaures",

"organisation": true

}

},

"destination": {

"contact": {

"name": "Marquise de Pompadour",

"email": "marquise-de-pompadour@example.com",

"phone": "07 9876 5432",

},

"address": {

"country\_code": "FR",

"locality": "Marseille",

"postal\_code": "13006",

"address\_line1": "175 Rue de Rome",

"organisation": false

}

},

"package": {

"dimensions": {

"height": 10,

"width": 10,

"length": 10,

"unit": "cm",

},

"grossWeight": {

"amount": 100,

"unit":kg,

}

}

}

}

}

**Shipment not found:**

{

"data": {

"ref" : ""

}

}

## Process

* Using “ref” number to query the shipment, if the shipment exists, return format as **4.2**
* “ref” field is required
* Return an error if shipment is not found
* Unit tests are required

# deleteShipment

## Request

POST /client/deleteshipment HTTP/1.1

{

"data": {

"ref": "\_\_REFERENCE\_NUMBER", // Use to lookup the shipment

}

}

## Response

{

"data": [

{

"status": "OK”,

"message": "shipment has been deleted",

}]

}

## Process

* Using “ref” number to query the shipment, if the shipment exists, return format as 4.2
* “ref” field is required
* Status:
  + “OK” if the shipment is deleted successfully
  + “NOK” if the shipment is not found or if there is any error
* Message:
  + Delete shipment successfully: “Shipment has been deleted”
  + Cannot found the shipment: “Shipment not found”
* Unit tests are required

# trackShipment

## Request

**OUT OF SCOPE**

## Response

**OUT OF SCOPE**