trendyol tech

Follow Us!
Trendyol Tech Medium
Trendyol Open Source GitHub
Trendyol Tech Youtube
Trendyol Tech MeetUp

Fleet Management Backend Application

Description:

In this application, we want you to design a small-scale fleet management system where vehicles make deliveries to predetermined locations along a certain route.

The system includes two different types of shipments that can be transported in vehicles and unloaded at delivery points. Delivery points, barcode numbers and volumetric weight are specified on the shipments.

Shipment Types:
Package: Refers to shipments with a single item.
Bag: Refers to shipments with multiple items.

Vehicles must have license plates to be registered in the system.

The system includes three different delivery points.
Branch: Only package-type shipments can be unloaded. Bags and packages in bags may not be unloaded.
Distribution Center: Bags, packages in bags and packages not assigned to any bags may be unloaded.
Transfer Center: Only bags and packages in bags may be unloaded.

Shipments with delivery points that do not meet the aforementioned criteria may not be unloaded. In such cases, these particular shipments must be skipped and the remaining shipments should be checked if they meet the criteria for unloading.

The status of a shipment is “created” when it is first created, “loaded into bag” when loaded into the bag, and “unloaded” when unloaded at the delivery point.

The packages loaded into a bag must have the same delivery point as the bag.

When a shipment is loaded into the bag, the shipment’s status is updated to “loaded into bag,” while the bag’s status remains “created.”

If the bag itself is unloaded with the packages still inside, the status of the bag and all the packages inside is updated to “unloaded”.

Vehicles must go to their assigned delivery points and ensure that the relevant shipments are unloaded at the relevant delivery points.

Status Table:

Consider this as an enum or a table.

Package Status	**Value**
Created	1
Loaded into Bag	2
Loaded	3
Unloaded	4

Bag Status	**Value**
Created	1
Loaded	3
Unloaded	4

To design the system above:
We expect you to write the rest endpoints with the following parameters and store them in a database of your choice. Except for the ones in clause 6, you are free to create the parameters and names of all the rest endpoints. You may develop the database (it could also be an in-memory database) and domain models according to the following clauses.

1. You must create the following vehicle with the endpoint used to create vehicles.

License Plate
34 TL 34

2. You must create the following delivery points with the endpoint used to create delivery points.

Delivery Point	Value
Branch	1
Distribution Center	2
Transfer Center	3

3. You must create the following bags with the endpoint used to create bags.

Barcode	Delivery Point for Unloading
C725799	2
C725800	3

4. You must create the following shipment list with the endpoint used to create packages.

Barcode	Delivery Point for Unloading	Volumetric Weight
P7988000121	1	5
P7988000122	1	5
P7988000123	1	9
P8988000120	2	33
P8988000121	2	17
P8988000122	2	26
P8988000123	2	35
P8988000124	2	1
P8988000125	2	200
P8988000126	2	50
P9988000126	3	15

P9988000127	3	16
P9988000128	3	55
P9988000129	3	28
P9988000130	3	17

5. You must assign the following packages to the relevant bags with the endpoint used to assign packages to bags.

Barcode	**Bag Barcode**
P8988000122	C725799
P8988000126	C725799
P9988000128	C725800
P9988000129	C725800

6. Using the json content below, you must call the endpoint you have created to have the vehicles take the packages to the relevant delivery points, and make sure that the packages are distributed.
In the first step, you must update the status to “loaded” for all packages and bags that are specified in the list and loaded into the assigned bags. You must then unload the shipments at the delivery points specified in the list, and update the status of these shipments to “unloaded.”

```
{
 "plate": "34 TL 34",
 "route": [
  {
    "deliveryPoint": 1,
    "deliveries": [
      {"barcode": "P7988000121"},
      {"barcode": "P7988000122"},
      {"barcode": "P7988000123"},
      {"barcode": "P8988000121"},
      {"barcode": "C725799"}
    ]
```

```
    },
    {
      "deliveryPoint": 2,
      "deliveries": [
        {"barcode": "P8988000123"},
        {"barcode": "P8988000124"},
        {"barcode": "P8988000125"},
        {"barcode": "C725799"}
      ]
    },
    {
      "deliveryPoint": 3,
      "deliveries": [
        {"barcode": "P9988000126"},
        {"barcode": "P9988000127"},
        {"barcode": "P9988000128"},
        {"barcode": "P9988000129"},
        {"barcode": "P9988000130"}
      ]
    }
  ]
}
```

Test Results:

- While the packages are unloaded at the delivery points, we expect you to create a table to log the incorrectly sent Delivery Point - Barcode pairs in the json above.
- We expect you to show the status of loaded and unloaded shipments on the database.
- We expect the status of the shipment with the barcode number P8988000120 to remain “created”.
- We expect to see logs for barcode numbers P8988000121 and C725799 (due to attempt to deliver to the wrong point)
- We expect the status of the shipment with the barcode number P8988000121 to remain “loaded”.
- We expect the status of the bag with the barcode number C725800 to be “unloaded”.
- We expect the status of the shipment with the barcode number P8988000122 to be “unloaded” status.
- We expect the status of the shipment with the barcode number P8988000126 to be “unloaded”.

Expected Result:


```
{
  "plate": "34 TL 34",
  "route": [
    {
      "deliveryPoint": 1,
      "deliveries": [
        {"barcode": "P7988000121", "state":4},
        {"barcode": "P7988000122", "state":4},
        {"barcode": "P7988000123", "state":4},
        {"barcode": "P8988000121", "state":3},
        {"barcode": "C725799", "state":3}
      ]
    },
    {
      "deliveryPoint": 2,
      "deliveries": [
        {"barcode": "P8988000123", "state":4},
        {"barcode": "P8988000124", "state":4},
        {"barcode": "P8988000125", "state":4},
        {"barcode": "C725799", "state":4}
      ]
    },
    {
      "deliveryPoint": "3",
      "deliveries": [
        {"barcode": "P9988000126", "state":3},
        {"barcode": "P9988000127", "state":3},
        {"barcode": "P9988000128", "state":4},
        {"barcode": "P9988000129", "state":4},
        {"barcode": "P9988000130", "state":3}
      ]
    }
  ]
}
```

trendyol tech

Follow Us!

Trendyol Tech Medium
Trendyol Open Source GitHub
Trendyol Tech Youtube
Trendyol Tech MeetUp

You should be aware of the following conventions while you are working on this exercise:

- Publish a running project
- Apply SOLID and OOP principles
- Document your project for a developer
- Handle exceptions for resilient web service
- Test Driven Development is a good choice, but it's up to you how you are testing your code.
- Nice to have: Dockerize your project

You should commit to a local git repository and include the git repository (.git/) in the upload.

NOTE: Please DO NOT publish the project on Github, Gitlab, etc. We will provide a private gitlab repository for you and we expect you to push your changes to that private repository or send your workspace folder as compressed.