API QUERY FORMAT SHEET

Contents

[CAFÉ 2](#_Toc15176443)

[GET: Get café details by café name and password 2](#_Toc15176444)

[POST: Add a café 2](#_Toc15176445)

[PUT: Update a café 3](#_Toc15176446)

[CUP 3](#_Toc15176447)

[POST: Add a cup 3](#_Toc15176448)

[PUT: Update a cup 4](#_Toc15176449)

[GET: Total count of cups 4](#_Toc15176450)

[SALE 5](#_Toc15176451)

[POST: Add a sale record 5](#_Toc15176452)

[PUT: Update a sale record 5](#_Toc15176453)

[POST: Add bulk records from sale cache 6](#_Toc15176454)

[GET: Total count of sales 6](#_Toc15176455)

[GET: Sales count per café per day 7](#_Toc15176456)

[DISHWASHER 8](#_Toc15176457)

[GET: Get dishwasher details by name and password 8](#_Toc15176458)

[POST: Add a dishwasher 8](#_Toc15176459)

[PUT: Update a dishwasher 9](#_Toc15176460)

[BIN 9](#_Toc15176461)

[POST: Add a bin 9](#_Toc15176462)

[PUT: Update a bin 10](#_Toc15176463)

[RETURN 10](#_Toc15176464)

[POST: Add a return record 10](#_Toc15176465)

[PUT: Update a return record 11](#_Toc15176466)

[GET: Total count of returns 11](#_Toc15176467)

[POST: Add bulk records from return cache 12](#_Toc15176468)

[RETURN RATE 12](#_Toc15176469)

[GET: Get weekly return rate 12](#_Toc15176470)

# CAFÉ

## GET: Get café details by café name and password

URL: <http://host:port/api/cafe/name/password>

Example: <http://host:port/api/cafe/nesso/pass>

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| GET | - | - | name and password are not case-sensitive. |

Response:

[

{

"id": 101,

"name": "NESSO",

"password": "PASS",

"latitude": -90,

"longitude": 180,

"created\_at": "2019-03-09T10:52:40.000Z",

"updated\_at": null

}

]

## POST: Add a café

URL: <http://host:port/api/cafe>

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| POST | 'Content-Type' = ‘application/json' | JSON | Id and date will be taken care by the database and API respectively. |

{

    "name" : "CCD",

    "password" : "pass",

    "latitude" : 50,

    "longitude" : -50

}

## PUT: Update a café

URL: <http://host:port/api/cafe/3>

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| PUT | 'Content-Type' = ‘application/json' | JSON | * Update date will be taken care by the API. * You may include all 4 properties or any 3 or 2 or just 1 in any order. |

{

    "name" : "CCD",

    "longitude" : -50

}

{

    "password" : "pass",

    "latitude" : 50,

    "longitude" : -50

}

{

    "name" : "CCD",

    "password" : "pass"

}

{

    "name" : "CCD"

}

# CUP

## POST: Add a cup

URL: <http://host:port/api/cup>

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| POST | 'Content-Type' = ‘application/json' | JSON | * Only date will be taken care by the API. * Rest you have to supply in the body. |

{

    "id": 102,

    "size": "S",

    "status": "A",

    "batch\_id": 1

}

## PUT: Update a cup

URL: <http://host:port/api/cup/3>

{

    "batch\_id": 1

}

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| PUT | 'Content-Type' = ‘application/json' | JSON | * Update date will be taken care by the API. * You may include all 3 properties or any 2 or just 1 in any order. |

{

    "size": "S",

    "status": "A"

}

{

    "size": "S",

    "status": "A",

    "batch\_id": 1

}

## GET: Total count of cups

URL: <http://host:port/api/cup/count?startDate=2019/06/20&endDate=2019/06/21>

Get total count of cups added between 2 dates (former inclusive and latter exclusive)

Note:

* Dates should be in format YYYY/MM/DD without any single/double quotes
* Both dates are optional. You can choose to provide start date or end date or both or none
* **startDate** and **endDate** should be used exactly in the same Camel-Case style i.e. **D** will be uppercase in date words.
* When both dates are provided, data between those dates is considered (startDate inclusive but endDate exclusive) for counting
* When only startDate is provided, all data from that date (inclusive) until the latest record is considered for counting
* When only endDate is provided, all data until the end date (exclusive) is considered for counting
* When no dates are provided, all data count is returned

# SALE

## POST: Add a sale record

URL: <http://host:port/api/sale>

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| POST | 'Content-Type' = ‘application/json' | JSON | * Id and scanned\_at will be taken care by the database and API respectively. * Make sure that the cup and café ids already exists in respective tables. |

{

    "cup\_id" : 123,

    "cafe\_id" : 101

}

## PUT: Update a sale record

URL: <http://host:port/api/sale/5>

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| PUT | 'Content-Type' = ‘application/json' | JSON | * You may include all three properties or any two or just one in any order. |

{

    "cup\_id" : 123,

    "cafe\_id" : 101,

    "scanned\_at" : "2019-01-31 20:59:59"

}

{

    "cup\_id" : 123,

    "cafe\_id" : 101

}

{

    "scanned\_at" : "2019-01-31 20:59:59"

}

## POST: Add bulk records from sale cache

URL: <http://host:port/api/sale/cache>

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| POST | 'Content-Type' = ‘application/json' | JSON | * Id will be taken care by the database and API respectively. * Make sure that the cup and café ids already exists in respective tables. |

[{

  "cup\_id" : 101,

  "cafe\_id" : 101,

  "scanned\_at" : "2019-03-05 22:22:00"

},

{

  "cup\_id" : 102,

  "cafe\_id" : 102,

  "scanned\_at" : "2019-03-05 22:33:00"

},

{

  "cup\_id" : 103,

  "cafe\_id" : 103,

  "scanned\_at" : "2019-03-05 22:44:00"

}]

## GET: Total count of sales

URL: <http://host:port/api/sale/count?startDate=2019/06/20&endDate=2019/06/21>

Get total count of sales done between 2 dates (former inclusive and latter exclusive)

Note:

* Dates should be in format YYYY/MM/DD without any single/double quotes
* Both dates are optional. You can choose to provide start date or end date or both or none
* **startDate** and **endDate** should be used exactly in the same Camel-Case style i.e. **D** will be uppercase in date words
* When both dates are provided, data between those dates is considered (startDate inclusive but endDate exclusive) for counting
* When only startDate is provided, all data from that date (inclusive) until the latest record is considered for counting
* When only endDate is provided, all data until the end date (exclusive) is considered for counting
* When no dates are provided, all data count is returned

## GET: Sales count per café per day

URL: <http://host:port/api/sale/salepercafeperday>

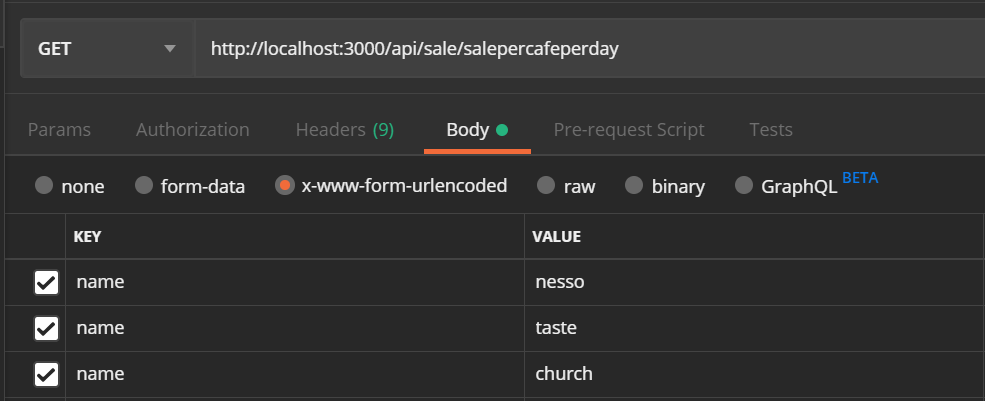
Get sales count per café per day in the following ways:

1. Get the data for every café for each day using the above link with no parameters and headers.
2. Get the data for one café by specifying ‘Content-Type’ to be ‘application/x-www-form-urlencoded’ and supplying **“name”** parameter as key and café name as value in the body of the request.
3. Get the data for multiple cafés of your choice by specifying ‘Content-Type’ to be ‘application/x-www-form-urlencoded’ and supplying **“name”** parameter as keys and café names as values in the body of the request i.e. to repeat ‘**name**’ key for each café you want.

**For ways 2 and 3 above:**

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| GET | 'Content-Type' = ‘application/x-www-form-urlencoded’ | x-www-form-urlencoded | * Pass key-value pairs where key will be ‘name’ and value will be café name. * Check café names by doing a GET all café first. |

In Postman, it looks like this:



Output format:

[

{

"CAFE\_ID": 101,

"NAME": "nesso",

"DATE": "21-03-2019",

"COUNT": 100

},

{

"CAFE\_ID": 101,

"NAME": "nesso",

"DATE": "22-03-2019",

"COUNT": 24

}

]

# DISHWASHER

## GET: Get dishwasher details by name and password

URL: <http://host:port/api/dishwasher/name/password>

Example: <http://host:port/api/dishwasher/campus%20centre/wash>

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| GET | - | - | name and password are not case-sensitive. |

Response:

[

{

"id": 101,

"name": "Campus Centre",

"password": "wash",

"latitude": -37.911786,

"longitude": 145.132916,

"created\_at": "2019-03-04T19:11:00.000Z",

"updated\_at": null

}

]

## POST: Add a dishwasher

URL: <http://host:port/api/>dishwasher

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| POST | 'Content-Type' = ‘application/json' | JSON | Id and date will be taken care by the database and API respectively. |

{

    "name" : "Campus Centre",

    "password" : "password",

    "latitude" : 50,

    "longitude" : -50

}

## PUT: Update a dishwasher

URL: <http://host:port/api/dishwasher/3>

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| PUT | 'Content-Type' = ‘application/json' | JSON | * Update date will be taken care by the API. * You may include all 4 properties or any 3 or 2 or just 1 in any order. |

{

    "name" : "CCD",

    "longitude" : -50

}

{

    "password" : "pass",

    "latitude" : 50,

    "longitude" : -50

}

{

    "name" : "CCD",

    "password" : "pass"

}

{

    "name" : "CCD"

}

# BIN

## POST: Add a bin

URL: <http://host:port/api/bin>

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| POST | 'Content-Type' = ‘application/json' | JSON | Id and date will be taken care by the database and API respectively. |

{

    "latitude" : 11.11,

    "longitude" : 22.22,

    "level" : 6

}

## PUT: Update a bin

URL: <http://host:port/api/bin/1>

{

    "level" : 6

}

{

    "latitude" : 11.11,

    "level" : 6

}

{

    "latitude" : 11.11,

    "longitude" : 22.22,

    "level" : 6

}

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| PUT | 'Content-Type' = ‘application/json' | JSON | * Update date will be taken care by the API. * You may include all three properties or any two or just one in any order. |

# RETURN

## POST: Add a return record

URL: <http://host:port/api/return>

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| POST | 'Content-Type' = ‘application/json' | JSON | * Id and scanned\_at will be taken care by the database and API respectively. * Make sure that the cup, dishwasher and bin ids already exists in respective tables. |

{

    "cup\_id": 101,

    "bin\_id": 102,

    "dishwasher\_id": 105

}

## PUT: Update a return record

URL: <http://host:port/api/return/5>

{

    "cup\_id": 102

}

{

    "bin\_id": 103,

    "dishwasher\_id": 104,

    "scanned\_at": "2019-01-31 20:59:59"

}

{

    "cup\_id": 102,

    "bin\_id": 103,

    "dishwasher\_id": 104,

    "scanned\_at": "2019-01-31 20:59:59"

}

{

    "cup\_id": 102,

    "dishwasher\_id": 104

}

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| PUT | 'Content-Type' = ‘application/json' | JSON | * You may include all four properties or any three or two or just one in any order. |

## GET: Total count of returns

URL: <http://host:port/api/return/count?startDate=2019/06/20&endDate=2019/06/21>

Get total count of returns done between 2 dates (former inclusive and latter exclusive)

Note:

* Dates should be in format YYYY/MM/DD without any single/double quotes
* Both dates are optional. You can choose to provide start date or end date or both or none
* **startDate** and **endDate** should be used exactly in the same Camel-Case style i.e. **D** will be uppercase in date words
* When both dates are provided, data between those dates is considered (startDate inclusive but endDate exclusive) for counting
* When only startDate is provided, all data from that date (inclusive) until the latest record is considered for counting
* When only endDate is provided, all data until the end date (exclusive) is considered for counting
* When no dates are provided, all data count is returned

## POST: Add bulk records from return cache

URL: <http://host:port/api/return/cache>

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD | HEADERS | BODY | COMMENTS |
| POST | 'Content-Type' = ‘application/json' | JSON | * Id will be taken care by the database and API respectively. * Make sure that the cup, bin and dishwasher ids already exists in respective tables. |

[

{

"cup\_id": 101,

"bin\_id": 101,

"dishwasher\_id": 101,

"scanned\_at": "2019-03-09 14:27:53"

},

{

"cup\_id": 102,

"bin\_id": 101,

"dishwasher\_id": 101,

"scanned\_at": "2019-03-10 01:42:00"

},

{

"cup\_id": 103,

"bin\_id": 101,

"dishwasher\_id": 101,

"scanned\_at": "2019-03-10 01:43:00"

}

]

# RETURN RATE

## GET: Get weekly return rate

URL: <http://host:port/api/returnrate>

Get weekly return rate for the whole system in the following format:

[

{

"YEAR": 2019,

"WEEK": 12,

"SALE\_TOTAL": 3,

"RETURN\_TOTAL": 1,

"RATE": 33.3333

},

{

"YEAR": 2019,

"WEEK": 15,

"SALE\_TOTAL": 1,

"RETURN\_TOTAL": null,

"RATE": null

}

]