API QUERY FORMAT SHEET

Contents

[CAFÉ 3](#_Toc70588267)

[GET: Get café details by café name and password 3](#_Toc70588268)

[POST: Add a café 3](#_Toc70588269)

[PUT: Update a café 4](#_Toc70588270)

[GET: Get all cafés 4](#_Toc70588271)

[GET: Get a café by its id 4](#_Toc70588272)

[DELETE: Delete a café by its id 4](#_Toc70588273)

[DELETE: Delete a café by the full café name 4](#_Toc70588274)

[CUP 5](#_Toc70588275)

[POST: Add a cup 5](#_Toc70588276)

[PUT: Update a cup 5](#_Toc70588277)

[GET: Total count of cups 5](#_Toc70588278)

[GET: Get all cups 6](#_Toc70588279)

[GET: Get a cup by its id 6](#_Toc70588280)

[DELETE: Delete a cup by its id 6](#_Toc70588281)

[SALE 7](#_Toc70588282)

[POST: Add a sale record 7](#_Toc70588283)

[PUT: Update a sale record 7](#_Toc70588284)

[POST: Add bulk records from sale cache 8](#_Toc70588285)

[GET: Total count of sales 8](#_Toc70588286)

[GET: Sales count per café per day 9](#_Toc70588287)

[GET: Last sale records with café names and ids 10](#_Toc70588288)

[GET: Get all sales 10](#_Toc70588289)

[GET: Get a sale by its id 10](#_Toc70588290)

[DELETE: Delete a sale by its id 10](#_Toc70588291)

[DISHWASHER 11](#_Toc70588292)

[GET: Get dishwasher details by name and password 11](#_Toc70588293)

[POST: Add a dishwasher 11](#_Toc70588294)

[PUT: Update a dishwasher 12](#_Toc70588295)

[GET: Get all dishwashers 12](#_Toc70588296)

[GET: Get a dishwasher by its id 12](#_Toc70588297)

[DELETE: Delete a dishwasher by its id 12](#_Toc70588298)

[BIN 13](#_Toc70588299)

[POST: Add a bin 13](#_Toc70588300)

[PUT: Update a bin 13](#_Toc70588301)

[GET: Get all bins 13](#_Toc70588302)

[GET: Get a bin by its id 14](#_Toc70588303)

[DELETE: Delete a bin by its id 14](#_Toc70588304)

[RETURN 15](#_Toc70588305)

[POST: Add a return record 15](#_Toc70588306)

[PUT: Update a return record 15](#_Toc70588307)

[GET: Total count of returns 16](#_Toc70588308)

[POST: Add bulk records from return cache 16](#_Toc70588309)

[GET: Last return records with café names and ids 17](#_Toc70588310)

[GET: Get all returns 17](#_Toc70588311)

[GET: Get a return by its id 17](#_Toc70588312)

[DELETE: Delete a return by its id 17](#_Toc70588313)

[RETURN RATE 18](#_Toc70588314)

[GET: Get weekly return rate 18](#_Toc70588315)

[ANALYSIS 18](#_Toc70588316)

[GET: Get number of cups by times used 18](#_Toc70588317)

# 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,

"café\_name": "Café Nesso",

"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. |

{

    "café\_name" : "CCD",

"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"

}

## GET: Get all cafés

URL: [http://host:port/api/cafe](http://host:port/api/cafe/3)

Get all cafes present in the database. No body or header required.

## GET: Get a café by its id

URL: [http://host:port/api/cafe](http://host:port/api/cafe/3)/101

Get a café present in the database by its id. No body or header required.

## DELETE: Delete a café by its id

URL: [http://host:port/api/cafe](http://host:port/api/cafe/3)/101

Delete a cafe present in the database by its id. No body or header required.

## DELETE: Delete a café by the full café name

URL: <http://host:port/api/cafe?cafeName=chuRch> oF sEcUlar CoFFEE

Delete a cafe present in the database by its full name. The cafeName value is case insensitive. Mind that the ‘cafeName’ should be passed exactly as shown above i.e. in camel-case style. CaféName value must not be enclosed within inverted commas (single or double). However, if the name contains an apostrophe, it must be provided.

# 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 must supply in the body. |

{

    "id": 102,

    "size": "S",

    "status": "R",

    "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": "R"

}

{

    "size": "S",

    "status": "R",

    "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

## GET: Get all cups

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

Get all cups present in the database. No body or header required.

## GET: Get a cup by its id

URL: <http://host:port/api/cup>/101

Get a cup present in the database by its id. No body or header required.

## DELETE: Delete a cup by its id

URL: <http://host:port/api/cup>/101

Delete a cup present in the database by its id. No body or header required.

# 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. |

{

    "scanned\_at" : "2019-01-31T20:59:59.000+1100"

}

{

    "cup\_id" : 123,

    "cafe\_id" : 101

}

{

    "cup\_id" : 123,

    "cafe\_id" : 101,

    "scanned\_at" : "2019-01-31T20:59:59.000+1100"

}

## 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" : "2019-03-05T22:22:00.000+1100"

},

{

  "cup\_id" : 102,

  "cafe\_id" : 102,

  "scanned\_at" : "2019-03-05T22:33:00.000+1100"

},

{

  "cup\_id" : 103,

  "cafe\_id" : 103,

  "scanned\_at" : "2019-03-05T22:44:00.000+1100"

}]

## 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>

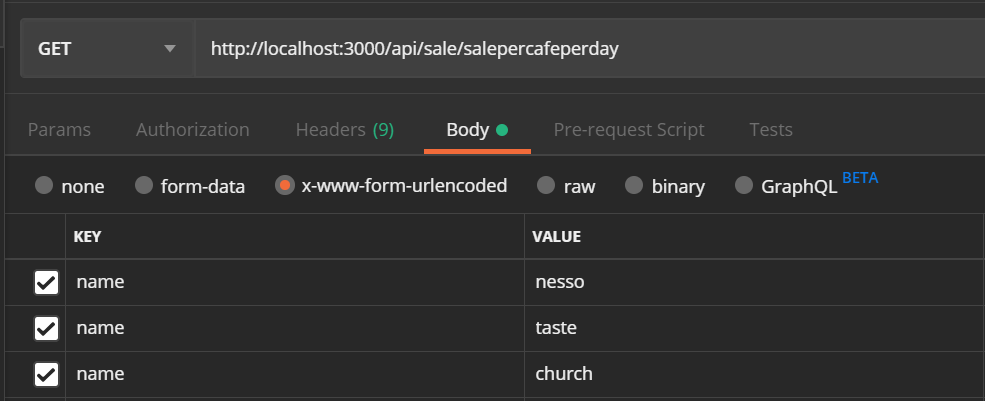
Get sales count per café per day in the following ways:

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

}

]

## GET: Last sale records with café names and ids

URL: <http://host:port/api/sale/last?count=100>

Get last sale records with café names and ids.

Note:

***count*** is a query parameter and should always be specified. It should be greater than zero, otherwise you would receive a 400 Bad Request error (Yes, error handled here! :P). ***count*** cannot begin with capital C.

## GET: Get all sales

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

Get all sale records present in the database. No body or header required.

## GET: Get a sale by its id

URL: <http://host:port/api/sale>/101

Get a sale record present in the database by its id. No body or header required.

## DELETE: Delete a sale by its id

URL: <http://host:port/api/sale>/101

Delete a sale record present in the database by its id. No body or header required.

# 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"

}

## GET: Get all dishwashers

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

Get all dishwashers present in the database. No body or header required.

## GET: Get a dishwasher by its id

URL: [http://host:port/api/dishwasher](http://host:port/api/cafe/3)/101

Get a dishwasher present in the database by its id. No body or header required.

## DELETE: Delete a dishwasher by its id

URL: <http://host:port/api/dishwasher>/101

Delete a dishwasher present in the database by its id. No body or header required.

# 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. |

## GET: Get all bins

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

Get all bins present in the database. No body or header required.

## GET: Get a bin by its id

URL: <http://host:port/api/bin>/101

Get a bins present in the database by its id. No body or header required.

## DELETE: Delete a bin by its id

URL: <http://host:port/api/bin>/101

Delete a bins present in the database by its id. No body or header required.

# 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>

|  |  |  |  |
| --- | --- | --- | --- |
| 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. |

{

    "bin\_id": 103,

    "dishwasher\_id": 104,

    "scanned\_at": "2019-01-31T20:59:59.000+1100"

}

{

    "cup\_id": 102,

    "bin\_id": 103,

    "dishwasher\_id": 104,

    "scanned\_at": "2019-01-31T20:59:59.000+1100"

}

{

    "cup\_id": 102

}

{

    "cup\_id": 102,

    "dishwasher\_id": 104

}

## 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": "2019-03-09T14:27:53.000+1100"

},

{

"cup\_id": 102,

"bin\_id": 101,

"dishwasher\_id": 101,

"scanned\_at": "2019-03-10T01:42:00.000+1100"

},

{

"cup\_id": 103,

"bin\_id": 101,

"dishwasher\_id": 101,

"scanned\_at": "2019-03-10T01:43:00.000+1100"

}

]

## GET: Last return records with café names and ids

URL: <http://host:port/api/return/last?count=100>

Get last return records with café names and ids.

Note:

***count*** is a query parameter and should always be specified. It should be greater than zero, otherwise you would receive a 400 Bad Request error (Yes, error handled here! :P). ***count*** cannot begin with capital C.

## GET: Get all returns

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

Get all return records present in the database. No body or header required.

## GET: Get a return by its id

URL: <http://host:port/api/return>/101

Get a return record present in the database by its id. No body or header required.

## DELETE: Delete a return by its id

URL: <http://host:port/api/return>/101

Delete a return record present in the database by its id. No body or header required.

# 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

}

]

# ANALYSIS

## GET: Get number of cups by times used

URL: <http://host:port/api/analysis/numberofcupsbytimesused>

Get data on cups usage as how many cups used how many times:

Sample output:

[

    {

       "times\_used":3,

       "number\_of\_cups":233

    },

    {

       "times\_used":2,

       "number\_of\_cups":211

    },

    {

       "times\_used":1,

       "number\_of\_cups":193

    }

 ]