Sales Use Case API Document

**1.** **Use Case:**

We have to store Product sales data into database.

We can use two table to store that information, where one table stores customer name, total sales amount, total quantity etc. and another stores each product information.

Here second table is the child table of first table. So, we should make relation with both tables.

**2.** **Table structure:**

**Table Name:** header

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Data Type** | **Default** | **PK** | **FK** | **NN** | AI |
| id | INT |  | Check mark, Wingdings font, character code 252 decimal. |  | Check mark, Wingdings font, character code 252 decimal. | Check mark, Wingdings font, character code 252 decimal. |
| cust\_name | VARCHAR | NULL |  |  |  |  |
| tot\_item\_sale | INT | 0 |  |  |  |  |
| tot\_amount\_sale | FLOAT | 0 |  |  |  |  |
| created\_on | DATETIME | NULL |  |  |  |  |
| updated\_on | DATETIME | NULL |  |  |  |  |
|  |  |  |  |  |  |  |

**Table Name:** items

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Data Type** | **Default** | **PK** | **FK** | **NN** | AI |
| id | INT |  | Check mark, Wingdings font, character code 252 decimal. |  | Check mark, Wingdings font, character code 252 decimal. | Check mark, Wingdings font, character code 252 decimal. |
| header\_id | INT | NULL |  | Check mark, Wingdings font, character code 252 decimal. |  |  |
| name | VARCHAR | NULL |  |  |  |  |
| qty | INT | 0 |  |  |  |  |
| unit\_price | double | 0 |  |  |  |  |
| total | double | 0 |  |  |  |  |
| created\_on | DATETIME | NULL |  |  |  |  |
| updated\_on | DATETIME | NULL |  |  |  |  |
|  |  |  |  |  |  |  |

**3.** **Table Script:**

**3.1.** **Table Name:** header

**Script:** CREATE TABLE `header` (

`id` INT NOT NULL AUTO\_INCREMENT,

`cust\_name` VARCHAR(45) NULL,

`tot\_item\_sale` INT NULL DEFAULT 0,

`tot\_amount\_sale` FLOAT DEFAULT 0,

`created\_on` DATETIME NULL,

`updated\_on` DATETIME NULL,

PRIMARY KEY (`id`));

3.2. Table Name: items

**Script:** CREATE TABLE `items` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`header\_id` int(11) DEFAULT NULL,

`name` varchar(45) DEFAULT NULL,

`qty` int(11) DEFAULT '0',

`unit\_price` double DEFAULT '0',

`total` double DEFAULT '0',

`created\_on` datetime DEFAULT NULL,

`updated\_on` datetime DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `fk\_items\_header` (`header\_id`),

CONSTRAINT `fk\_items\_header` FOREIGN KEY (`header\_id`) REFERENCES `header` (`id`)

)

**4.** **API:**

We must implement all APIs for CRUD operations.

1. Create: To insert or store data into table we must implement API by the HTTP POST method.

2. Read: To get or fetch data from table we must implement API by the HTTP GET method.

3. Update: To update table data, we must implement API by the HTTP PUT method.

4. Delete: To delete data from table we must implement API by the HTTP DELETE method.

**API Response Metadata:**

Those API returns two types of Objects as JSON metadata. Those descriptions are given below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Object** | **Property** | **Type** | **Description** | **Example** |
| sales | id | Integer | The unique identifier by which to identify the Sales. | {"id":9,"custName":"Malay Sanyal","totSalesItem":10,"totSalesAmount":1100,"items":[{"id":10,"name":"Pen","qty":50,"unitprice":20,"total":100}]} |
| custName | String | Customer Name |
| totSalesItem | Integer | Total sales items count. |
| totSalesAmount | Double | Total sales amount. |
| itemsList | Array | Customer purchase items or products list. |
|  |  |  |  |  |
| items | id | Integer | The unique identifier by which to identify the Sales. | {"id":1,"name":"Pen","qty":2,"unitprice":20,"total":40,"sales":{"id":5,"custName":"Amit Dey","totSalesItem":11,"totSalesAmount":351,"itemsList":null}} |
|  | name | String | Item name. |
|  | qty | Integer | Total sales item count. |
|  | unitprice | Double | Item price. |
|  | total | Double | Total Item price (qty \* unitprice). |
|  | sales | Object | Sales Object. |
|  |  |  |  |  |

**Technology Uses:**

Language: JAVA 1.8

Framework: SpringBoot 2.4.5

Project Builder: Maven

**API Details:**

**1.** **Title:** Create Sales & Items

**URL:** localhost:8080/sales/create

**HTTP Method:** POST

**Request Body (Raw):**

{

"custName": "Animesh Sanyal",

"itemsList": [

{

"name": "Shirt",

"qty": 5,

"unitprice": 200

},

{

"name": "jeans",

"qty": 5,

"unitprice": 1500

}

]

}

**Response Body:**

{

"status": 0,

"message": "",

"data": {

“id”: 22,

"custName": "Animesh Sanyal",

"totSalesItem": 10,

"totSalesAmount": 8500,

"itemsList": [

{

“id”: 45,

"name": "Shirt",

"qty": 5,

"unitprice": 200,

“total”: 1000

},

{

“id”: 46,

"name": "jeans",

"qty": 5,

"unitprice": 1500,

“total”: 7500

}

]

}

}

**2.** **Title:** Create Items

**URL:** localhost:8080/sales/{salesId}/items/create

**HTTP Method:** POST

**Request Body (Raw):**

{

"name": "Pen 140001333",

"qty": 5,

"unitprice": 20

}

Note: Need to pass id property with value if update required.

**Response Body:**

{

"status": 0,

"message": "",

"data": {

"id": 46,

"name": "Nike Shirt",

"qty": 5,

"unitprice": 20,

"total": 100

}

}

**3.** **Title:** Update Sales & Items

To update record, we have to add id otherwise it will create new record.

**URL:** localhost:8080/sales/update

**HTTP Method:** PUT

**Request Body (Raw):**

{

“id”: 22,

"custName": "Animesh Sarkar",

"totSalesItem": 10,

"totSalesAmount": 8500,

"itemsList": [

{

“id”: 45

"name": "Levi’s Jeans",

"qty": 5,

"unitprice": 200

},

{

"name": "Pepe jeans",

"qty": 5,

"unitprice": 1500

}

]

}

**Response Body:**

{

"status": 0,

"message": "",

"data": {

“id”: 22,

"custName": "Animesh Sarkar",

"totSalesItem": 10,

"totSalesAmount": 8500,

"itemsList": [

{

“id”: 45

"name": "Levi’s Jeans",

"qty": 5,

"unitprice": 200,

“total”: 1000

},

{

“id”: 46

"name": "Pepe jeans",

"qty": 5,

"unitprice": 1500,

“total”: 7500

}

]

}

}

**4.** **Title:** Update Items

To update record, we have to add id otherwise it will not work.

**URL:** localhost:8080/sales/items/update

**HTTP Method:** PUT

**Request Body (Raw):**

{

"id": 60,

"name": " Nike Shirt",

"qty": 5,

"unitprice": 2500

}

Note: Need to pass id property with value if update required.

**Response Body:**

{

"status": 0,

"message": "",

"data": {

"id": 60,

"name": "Nike Shirt",

"qty": 5,

"unitprice": 2500,

"total": 12500

}

}

**5.** **Title:** Get all Sales and Items records

**URL:** localhost:8080/sales/items/update

**HTTP Method:** GET

**Response Body:**

{

"status": 0,

"message": "",

"data": [

{

"id": 24,

"custName": "Malay Sanyal 5",

"totSalesItem": 10,

"totSalesAmount": 1100,

"itemsList": [

{

"id": 25,

"name": "Pen 555",

"qty": 5,

"unitprice": 20,

"total": 100,

"sales": null

}

]

}

]

}

**6.** **Title:** Get all Items records

**URL:** localhost:8080/sales/items/get

**HTTP Method:** GET

**Response Body:**

{

"status": 0,

"message": "",

"data": [

{

"id": 1,

"name": "Pen",

"qty": 2,

"unitprice": 20,

"total": 40,

"sales": {

"id": 5,

"custName": "Amit Dey",

"totSalesItem": 11,

"totSalesAmount": 351,

"itemsList": null

}

}

]

}

**7.** **Title:** Delete Sales & all child Items

**URL:** localhost:8080/sales/delete/{id}

**HTTP Method:** DELETE

**Response Body:**

{

"status": 0,

"message": "",

"data": “”

}

**8.** **Title:** Delete one Items

**URL:** localhost:8080/sales/items/delete/{id}

**HTTP Method:** DELETE

**Response Body:**

{

"status": 0,

"message": "",

"data": “”

}

**Github Url:** <https://github.com/pradyut-20-19-jun/cg-sales-project.git>

**Github repository Clone:**

1. Create folder like “git-project”.

cd mkdir “git-project”

2. Go to this folder.

cd git-project

2. Go to this folder.

Git init

3. Clone Repository:

git clone <https://github.com/pradyut-20-19-jun/cg-sales-project.git>

4. Go to respective dir:

cd cg-sales-project

6. Follow README.md file.