**Remotech mappings (1.0v snapshot API)**

1. **Creating/Updating product and product details instantly:**

**(POST localhost:8080/api/products)**

**(PUT localhost:8080/api/products/{productid})**

{

    "shortName" : "Example Product I",

    "price": 21.37,

"category" : {

        "id": 1

    },

    "productDetails": {

     "fullName": "Example Full Name Product I",

     "description": "Detailed description of the product"

    }

}

Note: To create product, you need to specify existing category as JSON object and have productDetails object in the same transaction (1:1 relationship)!!!

1. **Creating/Updating category**

**(POST localhost:8080/api/categories)**

**(PUT localhost:8080/api/categories/{categoryid})**

1. {
2. "name" : "example category",
3. "description" : "lorem ipsum dolor 123"
4. }
5. **Creating/Updating user and cart instantly:**

**(POST localhost:8080/api/users)**

**(PUT localhost:8080/api/users/{userId})**

{

    "firstName": "John",

    "lastName": "Doe",

    "password": "password",

    "email": "example@gmail.com",

    "phone": "512345123",

    "enabled": 1

}

1. **Adding Products to Cart:**

****

Where 5 = /{userId}/ - cart PK is user ID, so cart\_items PK is also user\_id + product\_id from product column

1. **Deleting Products from Cart:**

****

Where 7 = /{cartId}/, and last 7 = {productid}

**All other GET methods and DELETE:**

1. **GET categories: GET *localhost:8080/api/categories***
2. **GET specific category: GET *localhost:8080/api/categories/{categoryId}***
3. **DELETE category: DELETE *localhost:8080/api/categories/{categoryId}***
4. **GET product: GET *localhost:8080/api/products***
5. **GET specific product: GET *localhost:8080/api/products/{productId}***
6. **DELETE product: DELETE *localhost:8080/api/product/{productId}***
7. **GET user: GET *localhost:8080/api/users***
8. **GET specific user: GET *localhost:8080/api/users/{userId}***
9. **DELETE user: DELETE *localhost:8080/api/users/{userId}***

**#1 There are same endpoints for cart… but it shouldn’t be used separately – it’s nested with user JSON response, so any cart operation should be done with User service, not cart itself.**

**#2 PATCH mapping can be added as needed in the future**