Lairi Piia
Montonen Joonas
Muittari Samuel
Rautiainen Aleksis
Rusi Romeo

Contents

Purpose of this document

This document describes what REST services our Omppu&Rane backend-application provides. It contains all needed information so that client can be implemented.

Lairi Piia
Montonen Joonas
Muittari Samuel
Rautiainen Aleksis
Rusi Romeo

REST resources (services)

1. Get all products -> http://localhost:8080/products
2. Get all manufacturers -> http://localhost:8080/manufacturers
3. Insert new product-> http://localhost:8080/products
4. Edit product -> http://localhost:8080/products/{id}
5. Delete product -> http://localhost:8080/products/{id}
6. Get one product-> http://localhost:8080/products/id/{id}
7. Get type from product-> http://localhost:8080/products/type/{type}
8. Get all reservations -> http://localhost:8080/reservations
9. Insert new reservation -> http://localhost:8080/reservations
10. Edit reservation -> http://localhost:8080/reservations/{id}
11. Delete reservation -> http://localhost:8080/reservations/{id}
12. Get one reservation -> http://localhost:8080/reservations/id/{id}

Lairi Piia
Montonen Joonas
Muittari Samuel
Rautiainen Aleksis
Rusi Romeo

REST requests and responses

1 Get all products

HTTP method: GET

API call: http://localhost:8080/products

Request body: -

Response example below:

```
[
    {
        "id": 1,
        "name": "Kiva",
        "type": "shirt",
        "size": "M",
        "color": "blue",
        "price": 15.15,
        "manufacturer": {
            "manufacturerid": 1,
            "name": "Adidas"
        }
    },
    {
        "id": 2,
        "name": "Ok",
        "type": "shirt",
        "size": "M",
        "color": "red",
        "price": 10.1,
        "manufacturer": {
            "manufacturerid": 2,
            "name": "Puma"
        }
    }
]
```

Lairi Piia
Montonen Joonas
Muittari Samuel
Rautiainen Aleksis
Rusi Romeo

2 Get all manufacturers

HTTP method: GET

API call: http://localhost:8080/manufacturers

Request body: -

Response example below:

```
[
    {
        "manufacturerid": 1,
        "name": "Adidas",
        "products": [
            {
                "id": 1,
                "name": "Kiva",
                "type": "shirt",
                "size": "M",
                "color": "blue",
                "price": 15.15
            }
        ]
    },
    {
        "manufacturerid": 2,
        "name": "Puma",
        "products": [
            {
                "id": 2,
                "name": "Ok",
                "type": "shirt",
                "size": "M",
                "color": "red",
                "price": 10.1
            }
        ]
    }
]
```

Lairi Piia
Montonen Joonas
Muittari Samuel
Rautiainen Aleksis
Rusi Romeo

3 Insert new product

HTTP method: POST

API call: http://localhost:8080/products

Request body:

```
{
    "name": "Tiikeri",
    "type": "jacket",
    "size": "M",
    "color": "orange",
    "price": 33.33,
    "manufacturer": {
        "manufacturerid": 2,
        "name": "Puma"
    }
}
```

Response example below (Status 200 OK):

```
{
    "id": 5,
    "name": "Tiikeri",
    "type": "jacket",
    "size": "M",
    "color": "orange",
    "price": 33.33,
    "manufacturer": {
        "manufacturerid": 2,
        "name": "Puma"
    }
}
```

Lairi Piia
Montonen Joonas
Muittari Samuel
Rautiainen Aleksis
Rusi Romeo

4 Edit product

HTTP method: PUT

API call: http://localhost:8080/products/5

Request body:

```
{
    "name": "Tiikeri Muokattuna",
    "type": "jacket",
    "size": "S",
    "color": "orange",
    "price": 44.44,
    "manufacturer": {
        "manufacturerid": 2,
        "name": "Puma"
    }
}
```

Response example below (Status 200 OK):

```
{
    "id": 5,
    "name": "Tiikeri Muokattuna",
    "type": "jacket",
    "size": "S",
    "color": "orange",
    "price": 44.44,
    "manufacturer": {
        "manufacturerid": 2,
        "name": "Puma"
    }
}
```

Lairi Piia
Montonen Joonas
Muittari Samuel
Rautiainen Aleksis
Rusi Romeo

5 Delete product

HTTP method: DELETE

API call: http://localhost:8080/products/5

Request body: -

Response: Status 200 OK

6 Get one product

HTTP method: GET

API call: http://localhost:8080/products/id/1

Request body: -

Response example below:

```json
{
    "id": 1,
    "name": "Kiva",
    "type": "shirt",
    "size": "M",
    "color": "blue",
    "price": 15.15,
    "manufacturer": {
        "manufacturerid": 1,
        "name": "Adidas"
    }
}
```

Lairi Piia
Montonen Joonas
Muittari Samuel
Rautiainen Aleksis
Rusi Romeo

7 Get type from product

HTTP method: GET

API call: http://localhost:8080/products/type/shirt

Request body: -

Response example below:

```
[
    {
        "id": 1,
        "name": "Kiva",
        "type": "shirt",
        "size": "M",
        "color": "blue",
        "price": 15.15,
        "manufacturer": {
            "manufacturerid": 1,
            "name": "Adidas"
        }
    },
    {
        "id": 2,
        "name": "Ok",
        "type": "shirt",
        "size": "M",
        "color": "red",
        "price": 10.1,
        "manufacturer": {
            "manufacturerid": 2,
            "name": "Puma"
        }
    }
]
```

Lairi Piia
Montonen Joonas
Muittari Samuel
Rautiainen Aleksis
Rusi Romeo

8 Get all reservations

HTTP method: GET

API call: http://localhost:8080/reservations

Request body: -

Response example below:

```
[
    {
        "reservationId": 1,
        "custName": "Sipe Santapukki",
        "email": "sipe@gmail.com",
        "phone": 10528558
    },
    {
        "reservationId": 2,
        "custName": "Toni Virtanen",
        "email": "toni@gmail.com",
        "phone": 10592977
    }
]
```

9 Insert new reservation

HTTP method: POST

API call: http://localhost:8080/reservations

Request body:

```
{
    "custName": "Romeo",
    "email": "romeo@rome.fi",
    "phone": 1234
}
```

Response example below(Status 200 OK):

```
{
    "reservationId": 3,
    "custName": "Romeo",
    "email": "romeo@rome.fi",
    "phone": 1234
}
```

Lairi Piia
Montonen Joonas
Muittari Samuel
Rautiainen Aleksis
Rusi Romeo

10 Edit reservation

HTTP method: PUT

API call: http://localhost:8080/reservations/3

Request body:

```
{
    "custName": "Romeo muokattuna",
    "email": "romeo@rome.com",
    "phone": 4321
}
```

Response example below(Status 200 OK):

```
{
    "reservationId": 3,
    "custName": "Romeo muokattuna",
    "email": "romeo@rome.com",
    "phone": 4321
}
```

11 Delete reservation

HTTP method: DELETE

API call: http://localhost:8080/reservations/3

Request body: -

Response: Status 200 OK

Lairi Piia
Montonen Joonas
Muittari Samuel
Rautiainen Aleksis
Rusi Romeo

12 Get one reservation

HTTP method: GET

API call: http://localhost:8080/reservations/id/1

Request body: -

Response example below:

```
{
    "reservationId": 1,
    "custName": "Sipe Santapukki",
    "email": "sipe@gmail.com",
    "phone": 10528558
}
```

Lairi Piia
Montonen Joonas
Muittari Samuel
Rautiainen Aleksis
Rusi Romeo

HTTP methods

Http Method	**Resource EndPoint**	**Input**	**Success Response**	**Error Response**	**Description**
GET	/products	Body: empty	Status: 200 List of products	Status 500	Retrieves all products
GET	/manufacturers	Body: empty	Status: 200 List of manufacturers	Status 500	Retrieves all manufacturers
POST	/products	Body: new product data	Status: 200 New product with id information	Status: 500	Creates new product
PUT	/products	N/A	N/A	Status: 400	Method not allowed
DELETE	/products	N/A	N/A	Status: 400	Method not allowed
GET	/products/id/{id}	Body: empty	Status: 200 product with id information	Status: 500	Get information of one product
GET	/products/type/{type}	Body: empty	Status: 200 List of products with type/{type}	Status: 500	Get information of a certain type from products
POST	/products/id/{id}	N/A	N/A	Status: 400	Method not allowed
PUT	/products/{id}	Body: product data with updates	Status: 200 Body: updated product with id information	Status:404 or 500	Updates an existing product
DELETE	/products/{id}	Body: empty	Status: 200 List of remaining products	Status:404 or 500	Deletes an existing product
GET	/reservations	Body: empty	Status: 200 List of reservations	Status 500	Retrieves all reservations

Lairi Piia
Montonen Joonas
Muittari Samuel
Rautiainen Aleksis
Rusi Romeo

POST	/reservations	Body: new reservation data	Status: 200 New reservation with id information	Status: 500	Creates new reservation
PUT	/reservations	N/A	N/A	Status: 400	Method not allowed
DELETE	/reservations	N/A	N/A	Status: 400	Method not allowed
GET	/reservations/id/{id}	Body: empty	Status: 200 reservation with id information	Status: 500	Get information of one reservation
PUT	/reservations/{id}	Body: reservation data with updates	Status: 200 Body: updated reservation with id information	Status:404 or 500	Updates an existing reservation
DELETE	/reservations/{id}	Body: empty	Status: 200 List of remaining reservations	Status:404 or 500	Deletes an existing reservation