ABOUT THE PROJECT

The goal of this application is to implement RESTful JSON APIs that will enable the management of purchasing groups.

DESCRIPTION

The user through the use of our backend will be able to make api calls to our service so that they can add, edit, delete and update products, customers and orders.

HOW TO USE THE APPLICATION

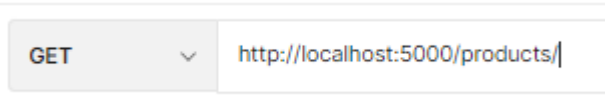
The application does not have any front-end, in order to be able to view, add, update and delete data it is advisable to use the service offered by Postman.

The back-end provide for the use of three paths for data manipulation:

1. Products;
2. Users;
3. Orders.

PRODUCTS

To perform manipulation of product data you will need to make requests at the following link:



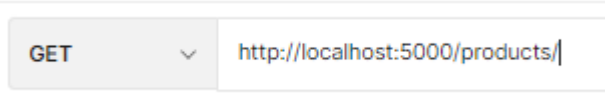
POST

To add a new product you will need to use the following schema:

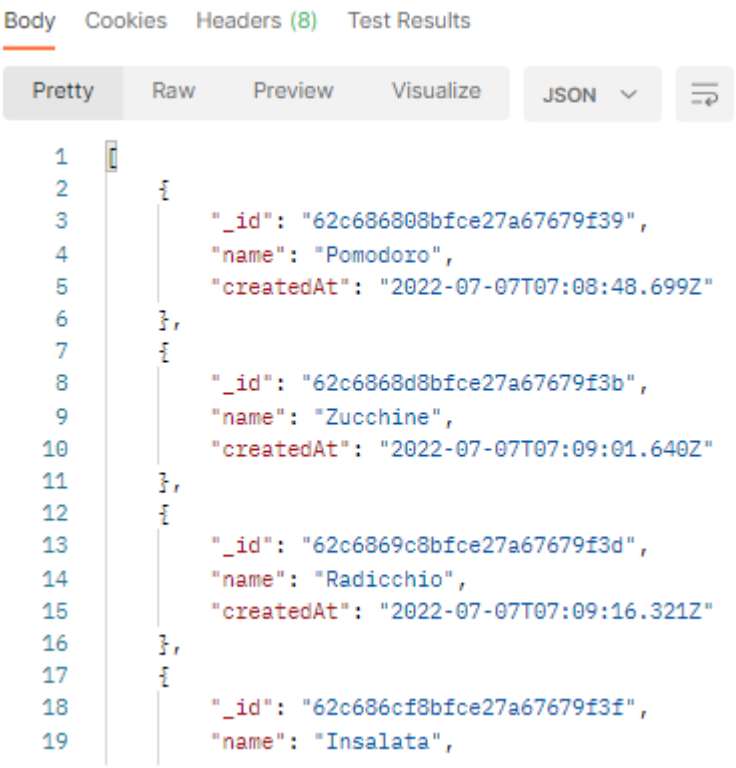
```
{
    "name": "Carote"
}
```

GET

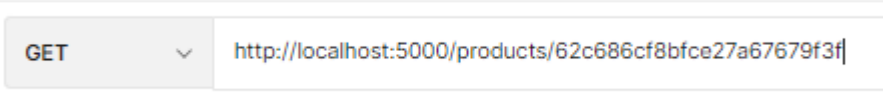
Requests for product display will be able to be made generically:



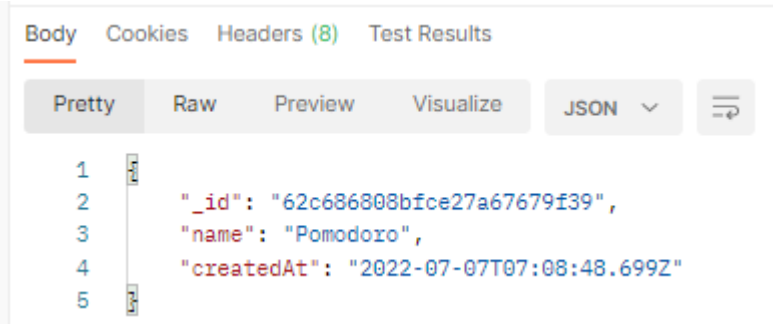
Result:



or specific using (if you know) the id of the product:



Result:

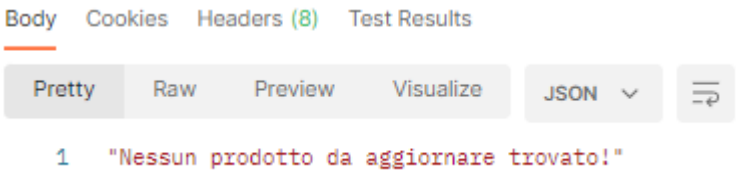


PATCH

In order to make the request to edit a product, you will first need to know its ID: The pattern will be the same as that of a POST call.

If successful, the following message will be displayed:

In case of an error this message will be displayed:

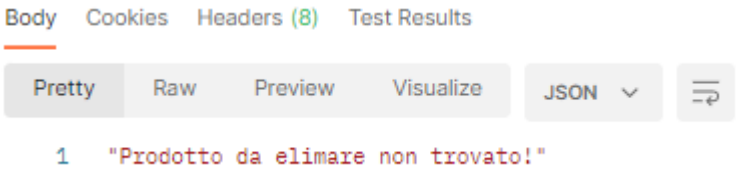


DELETE

To be able to carry out the deletion of a product it will be enough to know only its ID:

Display if successful:

Display in case of error:



USERS

To carry out manipulation of product data, requests will need to be made at the following link:

```
http://localhost:5000/users/
```

POST

To add a new user you will need to use the following scheme:

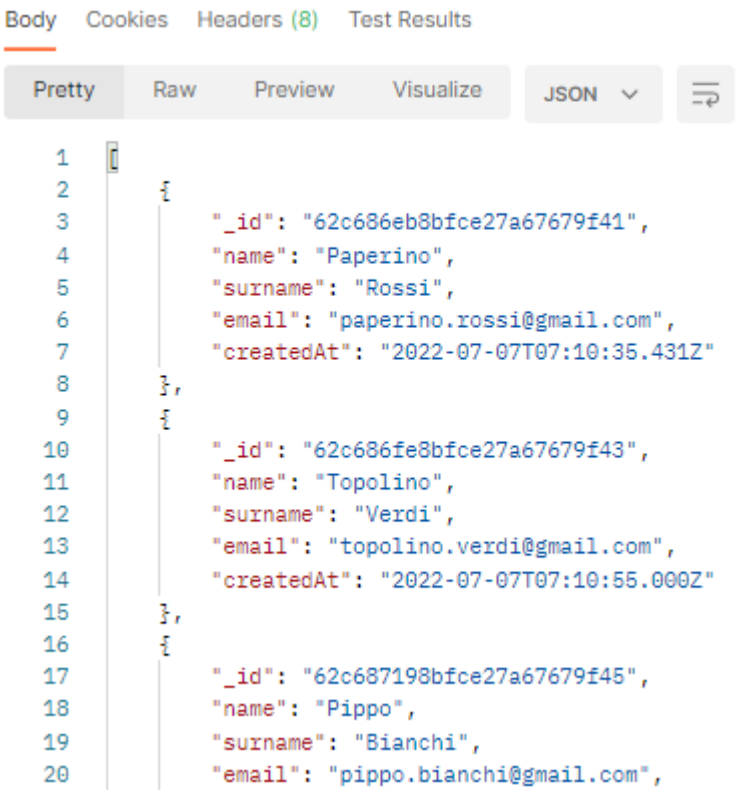
```
{
    "name": "Marco",
    "surname": "Neri",
    "email": "marco.neri@gmail.com"
}
```

GET

Requests for viewing users will be able to be made generically:

```
http://localhost:5000/users/
```

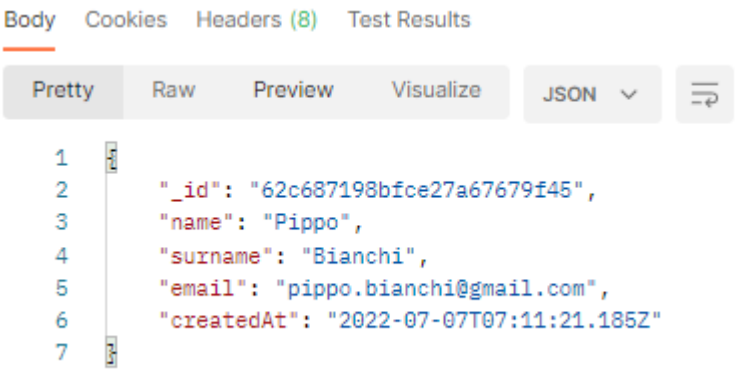
Result:



or specific using (if you know) the user id:

```
http://localhost:5000/users/62c687198bfce27a67679f45
```

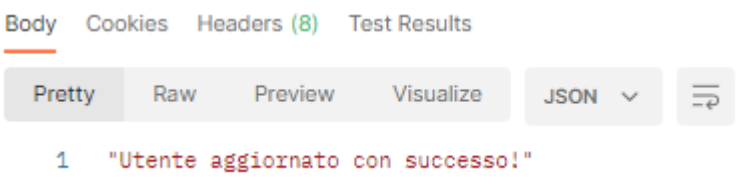
Result:



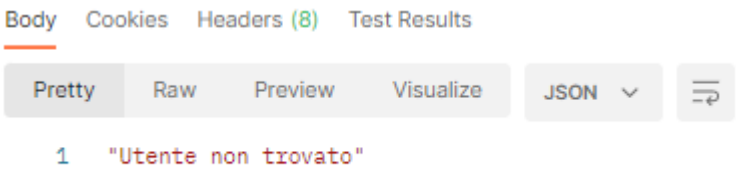
PATCH

In order to make the request to edit a user, you will first need to know the user's id: The pattern will be the same as that of a POST call.

If successful, the following message will be displayed:

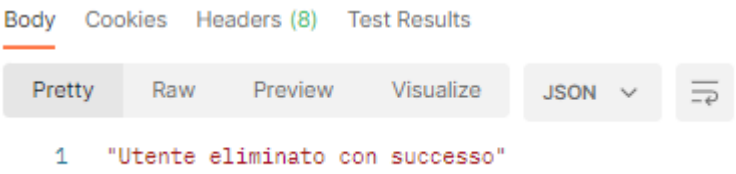


In case of an error this message will be displayed:

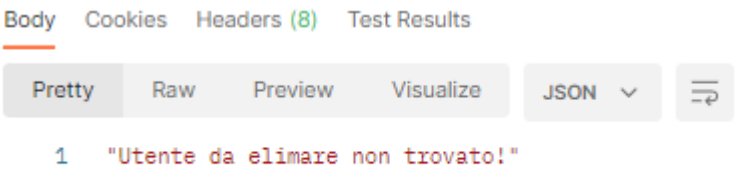


DELETE

In order to be able to carry out the deletion of a user it will be enough to know only his ID: Display in case of success:



Display in case of error:



ORDERS

To carry out manipulation of order data, requests will need to be made at the following link:

```
http://localhost:5000/orders/
```

POST

To add a new user you will need to use the following scheme:

```
{
    "id_user": "629a13d70af28565dea5973d",
    "products": [
        {
            "id_product": "62c6869c8bfce27a67679f3d"
        },
        {
            "id_product": "62c93067d1943cfc11293cfa"
        }
    ]
}
```

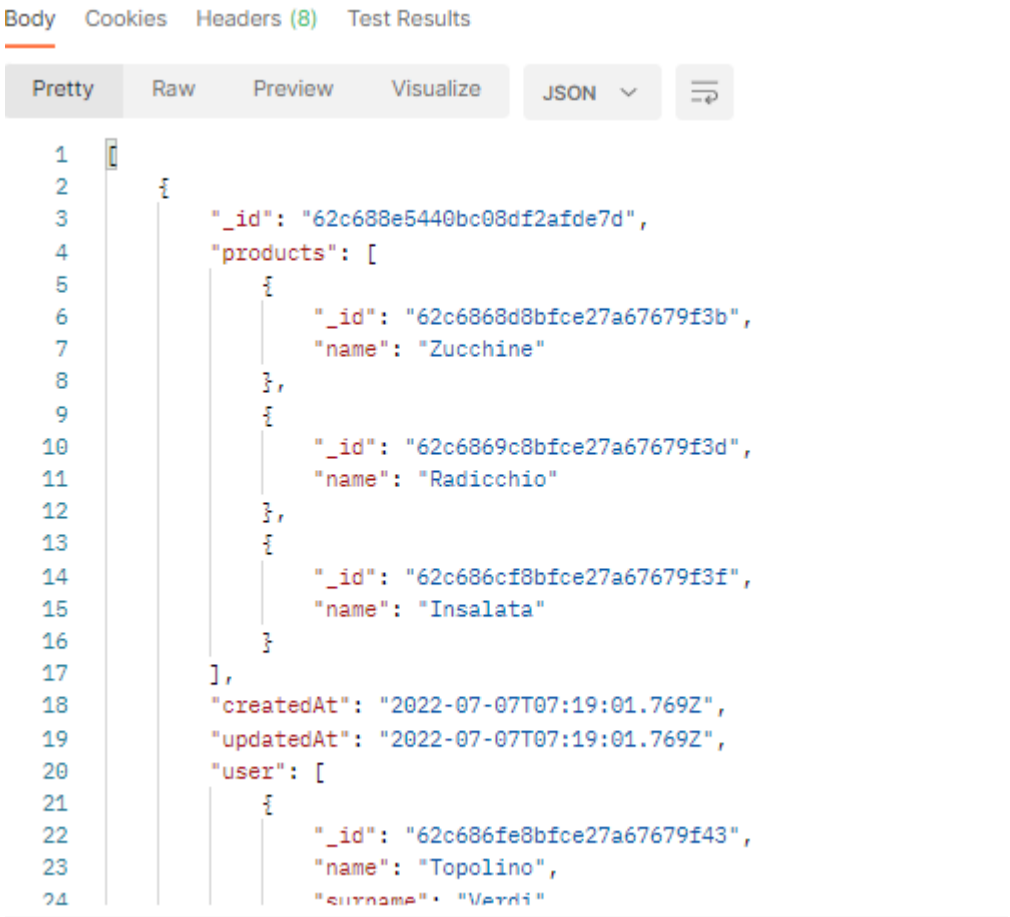
GET

Requests for displaying products may be made generically or using the date and name filter:

1. GENERIC

```
http://localhost:5000/orders/
```

Result:



1. SPECIFY

Or specify using (if you know) the product id.

```
http://localhost:5000/orders/62ca89f2bc765ef55aed7015
```

Result:

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON


```
[
    {
        "_id": "62ca89f2bc765ef55aed7015",
        "products": [
            {
                "_id": "62c686cf8bfce27a67679f3f",
                "name": "Insalata"
            }
        ],
        "createdAt": "2022-07-10T08:12:34.506Z",
        "updatedAt": "2022-07-10T08:12:34.506Z",
        "user": [
            {
                "_id": "62c686fe8bfce27a67679f43",
                "name": "Topolino",
                "surname": "Verdi",
                "email": "topolino.verdi@gmail.com"
            }
        ]
    }
]
```


3. USING THE FILTER

```
http://localhost:5000/orders/date=2022-07-10&name=insalata
```

Result:

Body Cookies Headers (8) Test Results

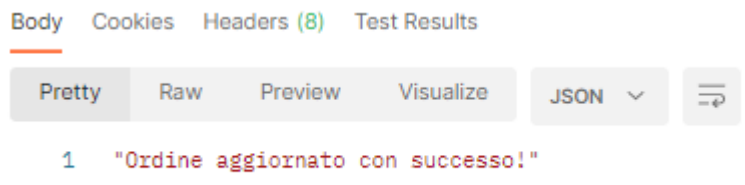
Pretty Raw Preview Visualize JSON


```
[
    {
        "_id": "62ca89f2bc765ef55aed7015",
        "products": [
            {
                "_id": "62c686cf8bfce27a67679f3f",
                "name": "Insalata"
            }
        ],
        "createdAt": "2022-07-10T08:12:34.506Z",
        "updatedAt": "2022-07-10T08:12:34.506Z",
        "user": [
            {
                "_id": "62c686fe8bfce27a67679f43",
                "name": "Topolino",
                "surname": "Verdi",
                "email": "topolino.verdi@gmail.com"
            }
        ]
    }
]
```

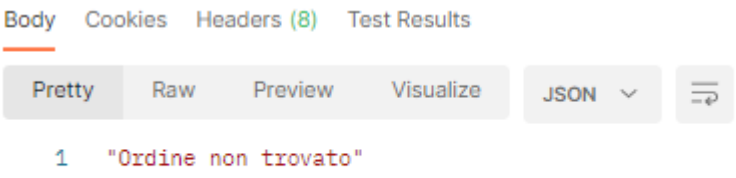
PATCH

In order to make the request to the modification of a product, we will first need to know its ID. The pattern will be the same as that of a POST call.

If successful, the following message will be displayed:

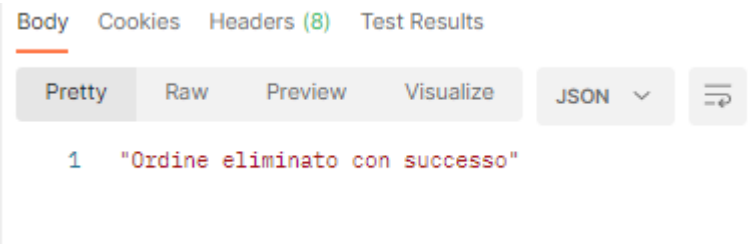


In case of an error this message will be displayed:

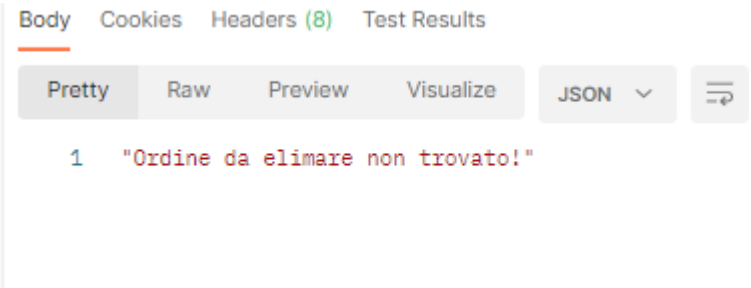


DELETE

To be able to carry out the deletion of a product it will be enough to know only its ID: Display if successful:



Display in case of error:



APPLICATION DEPENDENCIES

CORS

CORS is a node.js package for providing a Connect/Express middleware that can be used to enable CORS with various options.

DOTENV

Dotenv is a zero-dependency module that loads environment variables from a .env file into process.env. Storing configuration in the environment separate from code is based on The Twelve-Factor App methodology.

EXPRESS

Fast, unopinionated, minimalist web framework for node.

MONGOOSE

Mongoose is a MongoDB object modeling tool designed to work in an asynchronous environment. Mongoose supports both promises and callbacks.

CONTACTS

Email: gennuso.biagio@gmail.com

Github: PoF