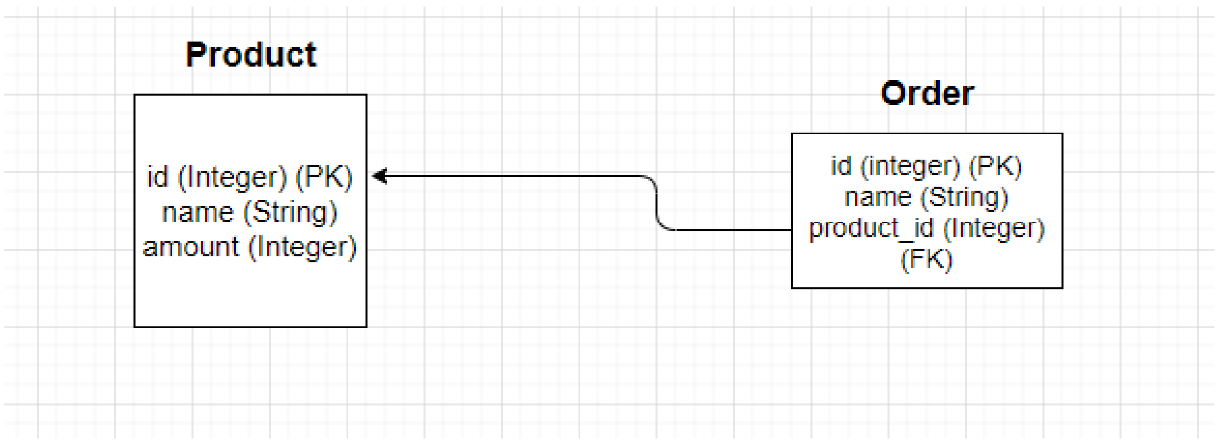
Technologies:

- Flask
- Postgres
- Postman

Database:



Functionality:

- Connecting with postgresql

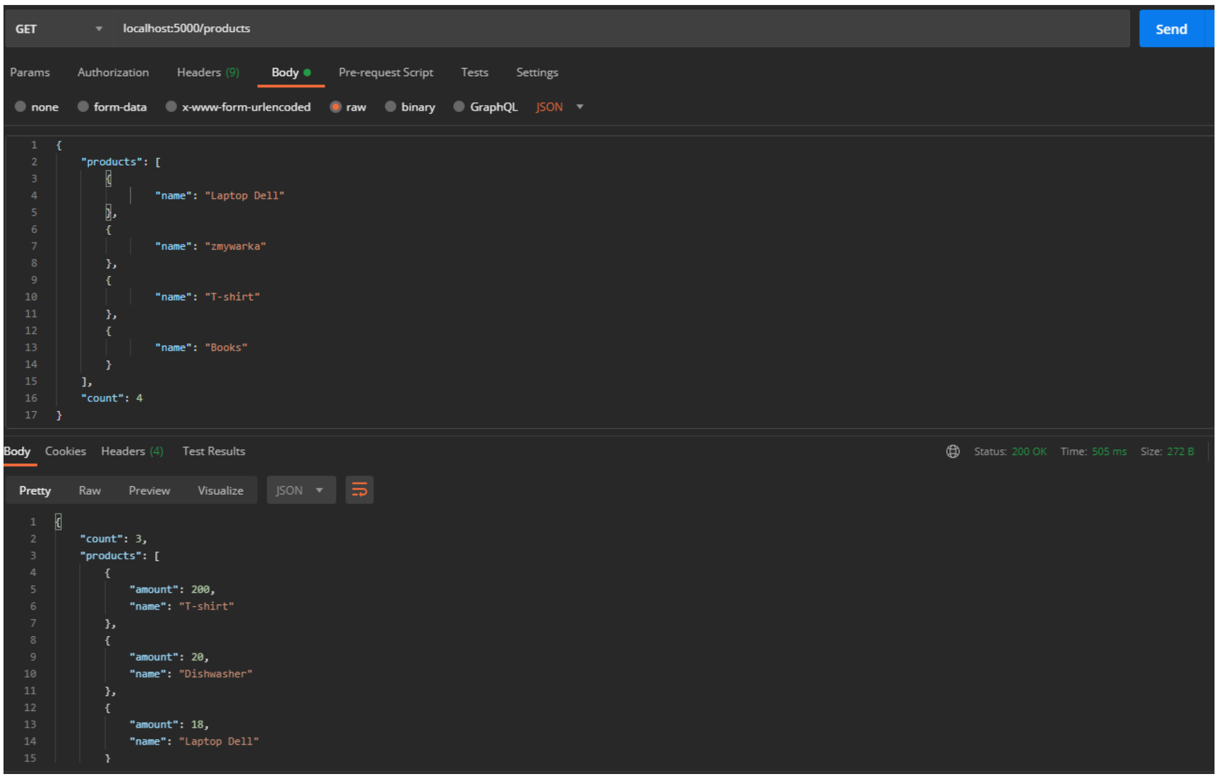
```
app.config['SQLALCHEMY_DATABASE_URI'] =
"postgresql://postgres:postgres@localhost:5432/amazon_api"
db = SQLAlchemy(app)
migrate = Migrate(app, db)
```

```
flask db init
flask db migrate
flask db upgrade
```

- Listing products

```
@app.route('/products', methods=['GET'])
def handle_product():
    if request.method == 'GET':
        products = ProductModel.query.all()
        results = [
            {
                "name": product.name,
                "amount": product.amount
            } for product in products]

        return {"count": len(results), "products": results}
```



```
localhost:5000/products
```

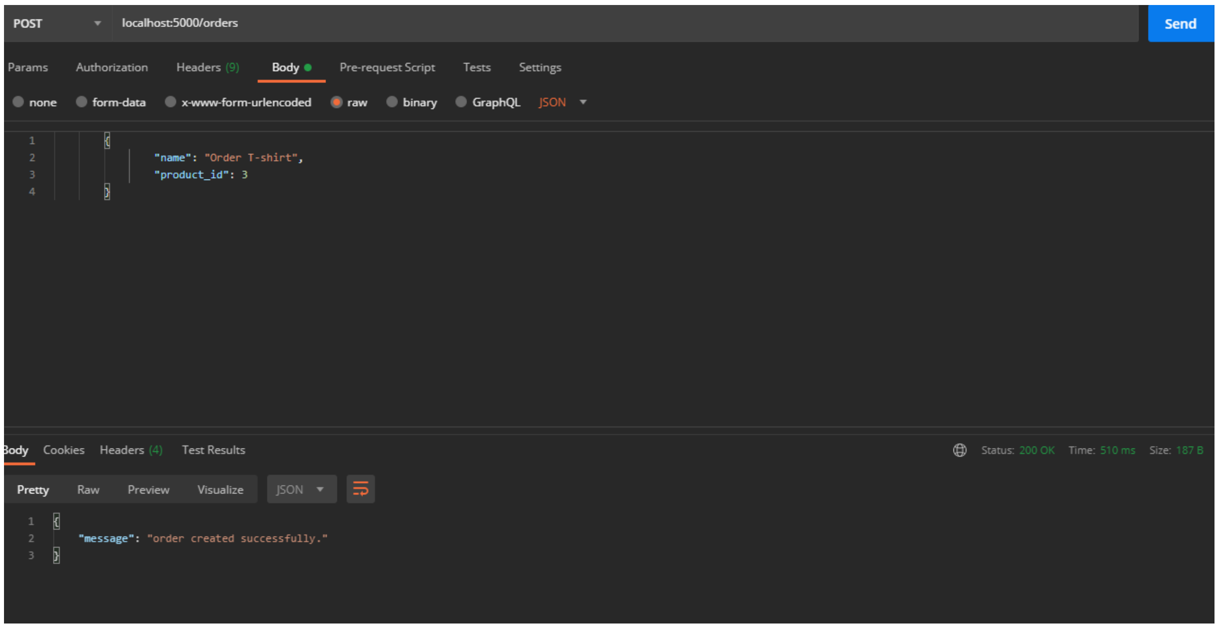
```
{
    "products": [
        {
            "name": "Laptop Dell"
        },
        {
            "name": "zmywarka"
        },
        {
            "name": "T-shirt"
        },
        {
            "name": "Books"
        }
    ],
    "count": 4
}
```

Status: 200 OK Time: 505 ms Size: 272 B

```
{
    "count": 3,
    "products": [
        {
            "amount": 200,
            "name": "T-shirt"
        },
        {
            "amount": 20,
            "name": "Dishwasher"
        },
        {
            "amount": 18,
            "name": "Laptop Dell"
        }
```

- Creating an order

```
@app.route('/orders', methods=['POST', 'GET'])
def handle_orders():
    if request.method == 'POST':
        if request.is_json:
            data = request.get_json()
            new_order = OrderModel(name=data['name'],
product_id=data['product_id'])
            db.session.add(new_order)
            db.session.commit()
            return {"message": f"order created successfully."}
        else:
            return {"error": "not in JSON format"}
```



POST localhost:5000/orders

Body: raw JSON

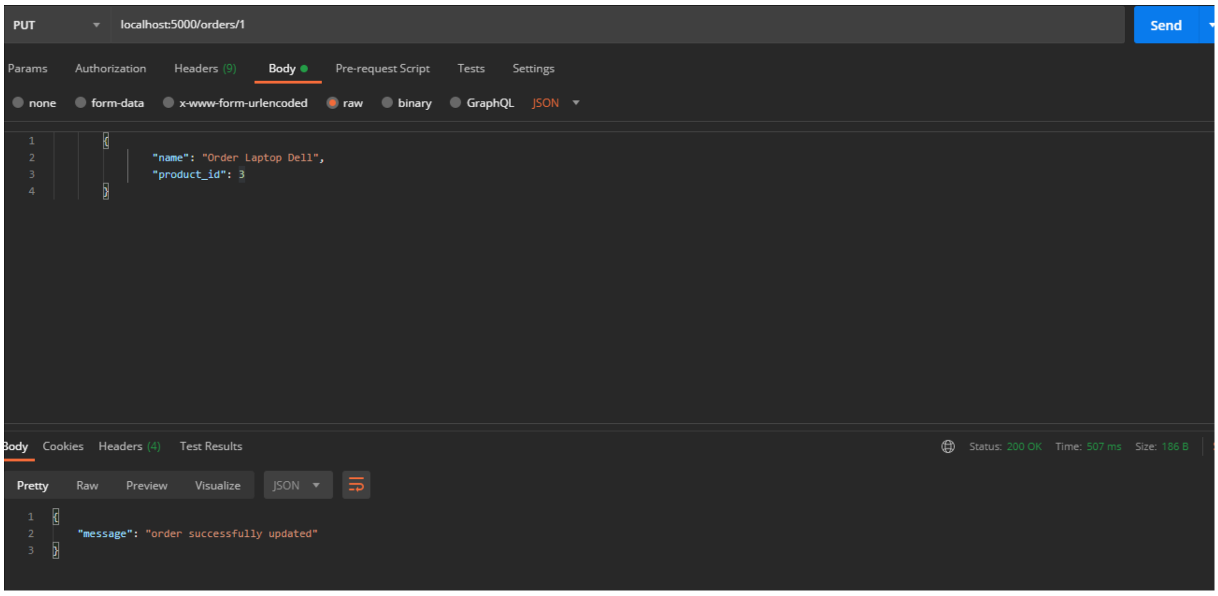
```
{
    "name": "Order T-shirt",
    "product_id": 3
}
```

Status: 200 OK Time: 510 ms Size: 187 B

```
{
    "message": "order created successfully."
}
```

➢ Updating an order

```
elif request.method == 'PUT':
    data = request.get_json()
    order.name = data['name']
    order.product_id = data['product_id']
    db.session.add(order)
    db.session.commit()
    return {"message": f"order successfully updated"}
```



PUT localhost:5000/orders/1

Body: raw JSON

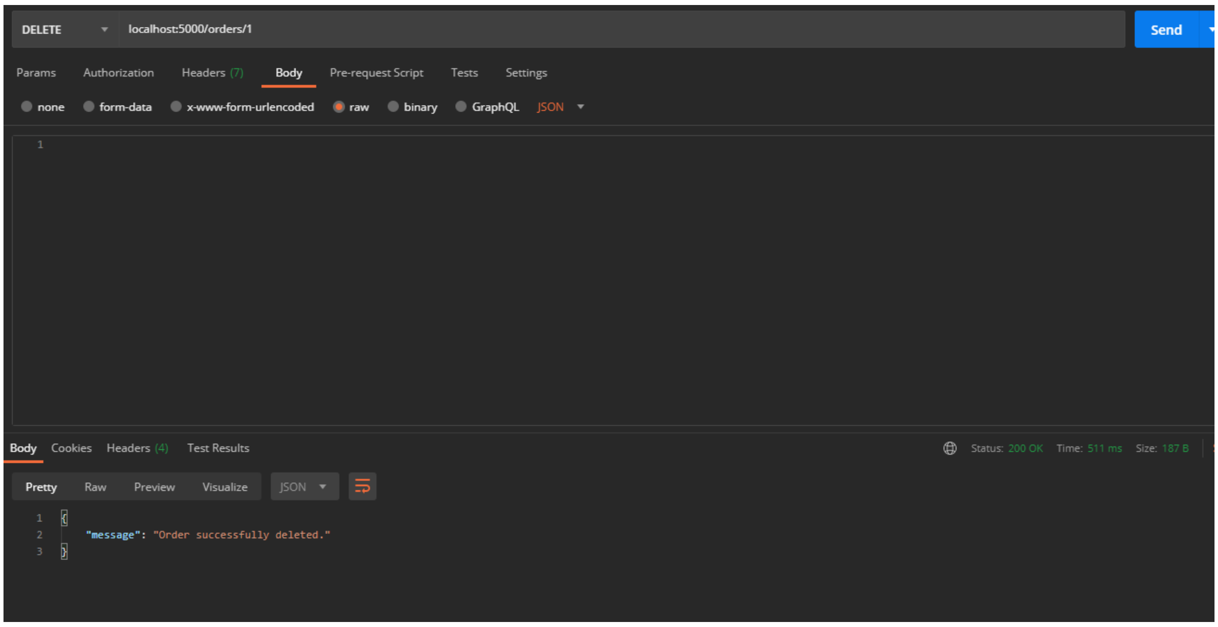
```
{
    "name": "Order Laptop Dell",
    "product_id": 3
}
```

Status: 200 OK Time: 507 ms Size: 186 B

```
{
    "message": "order successfully updated"
}
```

➢ Deleting an order

```
elif request.method == 'DELETE':
    db.session.delete(order)
    db.session.commit()
    return {"message": f"Order successfully deleted."}
```



localhost:5000/orders/1

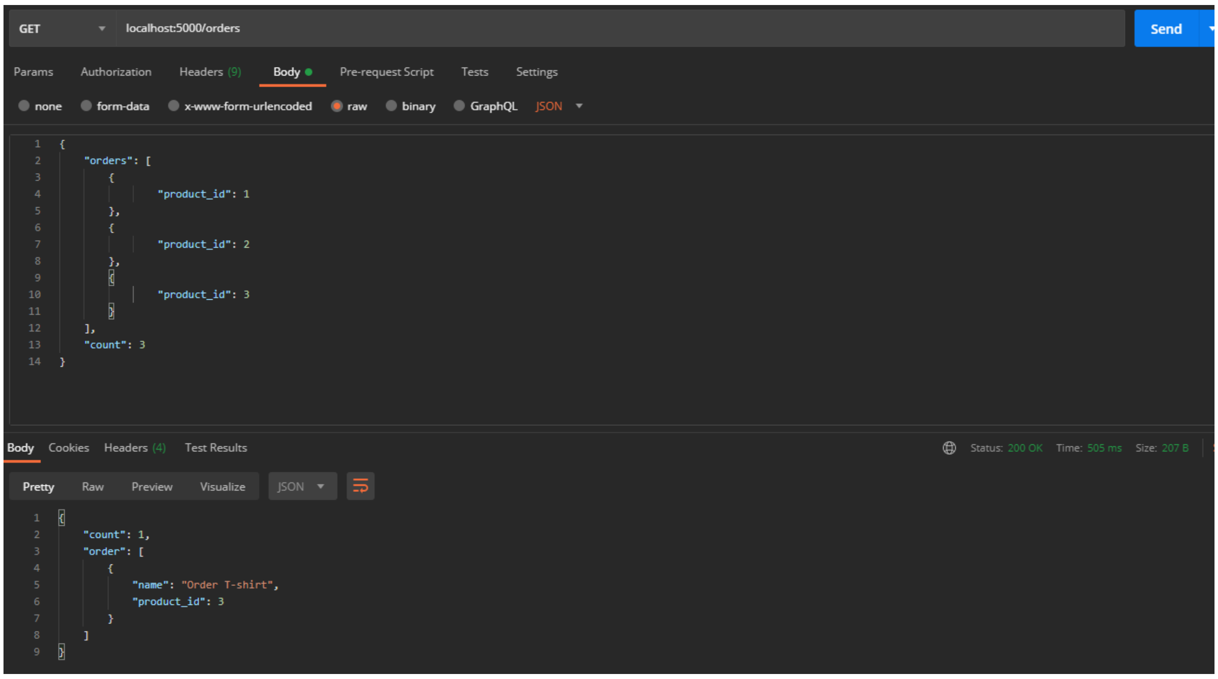
Status: 200 OK Time: 511 ms Size: 187 B

```
{
    "message": "Order successfully deleted."
}
```

➢ Listing existing orders

```
elif request.method == 'GET':
    orders = OrderModel.query.all()
    results = [
        {
            "name": order.name,
            "product_id": order.product_id,
        } for order in orders]

    return {"count": len(results), "order": results}
```



localhost:5000/orders

```
{
    "orders": [
        {
            "product_id": 1
        },
        {
            "product_id": 2
        },
        {
            "product_id": 3
        }
    ],
    "count": 3
}
```

Status: 200 OK Time: 505 ms Size: 207 B

```
{
    "count": 1,
    "order": [
        {
            "name": "Order T-shirt",
            "product_id": 3
        }
    ]
}
```

Testing

```python
import pytest

from app import ProductModel, OrderModel


@pytest.fixture(scope='module')
def product():
    product = ProductModel("desk", 12)
    return product

@pytest.fixture(scope='module')
def order():
    order = OrderModel("Order 1: T-shirt", 1)
    return order

def properly_product(product):
    assert product.name == ("desk",)
    assert product.amount == 12

def properly_order(order):
    assert order.name == ("Order1",)
    assert order.product_id == 1

def not_proper_amount_of_product(product):
    assert product.name == ("desk",)
    assert not product.amount == 10


def not_proper_name_of_order(order):
    assert not order.name == ("xyz",)
    assert order.product_id == 1
```