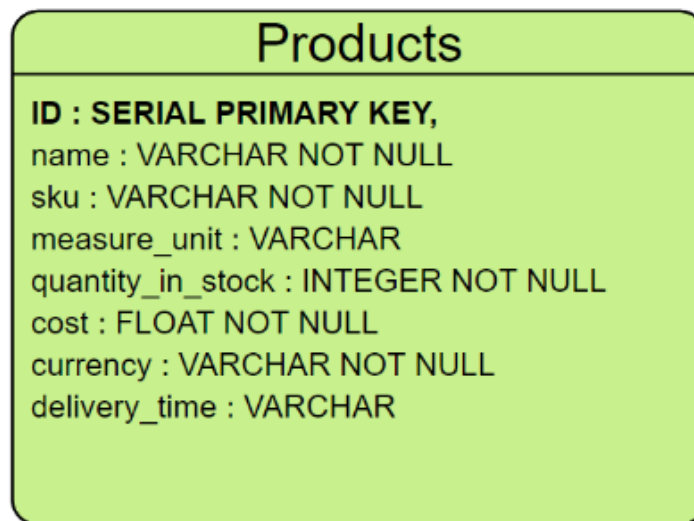# Data Modeling:

Produce an Entity relation diagram (ERD), and a more detailed UML Class diagram for your solution using PlantUML.

Initial products table contains all necessary info for commercial offer creating:
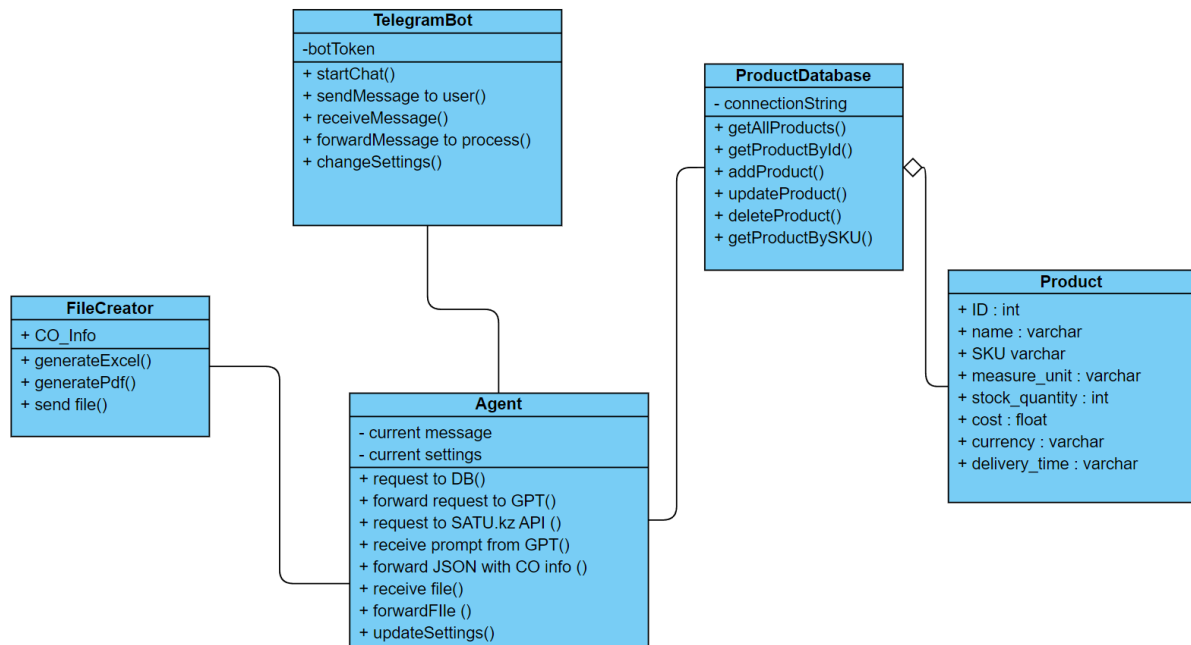- product name
- SKU
- measure units
- quantity in stock
- cost
- currency

Create detailed UML Class diagram for your solution

Main entities of our class structure:
- Telegram Bot - communication with user
- Product Database - contains information about products and provides their parameters change
- Agent - provides communication between other components with
- File Creator - creates file with Commercial offer

TelegramBot

-botToken

+ startChat()
+ sendMessage to user()
+ receiveMessage()
+ forwardMessage to process()
+ changeSettings()

ProductDatabase

- connectionString

+ getAllProducts()
+ getProductById()
+ addProduct()
+ updateProduct()
+ deleteProduct()
+ getProductBySKU()

Product

+ ID : int
+ name : varchar
+ SKU varchar
+ measure_unit : varchar
+ stock_quantity : int
+ cost : float
+ currency : varchar
+ delivery_time : varchar

FileCreator

+ CO_Info

+ generateExcel()
+ generatePdf()
+ send file()

Agent

- current message
- current settings

+ request to DB()
+ forward request to GPT()
+ request to SATU.kz API ()
+ receive prompt from GPT()
+ forward JSON with CO info ()
+ receive file()
+ forwardFIle ()
+ updateSettings()

API Design:
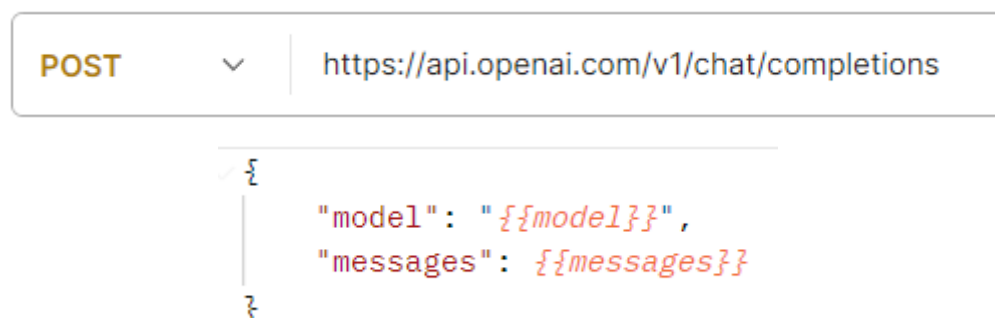## Specify your API according to best practices and mock it with Postman.

Our API will need:

1. Requests to ChatGPT API for communication

This will be used for message processing and creating automated requests for various parts of application

- **Simple chat completion functionality:**

POST request with message and specified GPT model

```
POST        v        https://api.openai.com/v1/chat/completions
```

```
{
    "model": "{{model}}",
    "messages": {{messages}}
}
```

Response contains completed text message

```
    "object": "chat.completion",
    "created": 1718605433,
    "model": "gpt-3.5-turbo-0125",
    "choices": [
        {
            "index": 0,
            "message": {
                "role": "assistant",
                "content": "The 2020 World Series was played
            },
```

● **Chat with function calling**

This function will send requests for specified function based on context
Example - user writes "add info about product #xxxx in my commercial offer" and application
will receive formed request with all arguments

```
    {
        "role": "user",
        "content": "What is the weather in San Francisco?"
    }


    "type": "function",
    "function": {
        "name": "get_current_weather",
        "description": "Get the current weather",
```

In response, function with specified message will be called

```
    "tool_calls": [
        {
            "id": "call_H5zLfaW6i3LWQRqUAxuSJ8qS",
            "type": "function",
            "function": {
                "name": "get_current_weather",
                "arguments": "{\"location\":\"San Francisco\"}"
            }
        }
    ]
```

2. Requests to satu.kz API for later addition of getting extensive product information (images for example)

On the first stage of the project we can send request to server which imitates Satu.kz:

GET ∨ | https://namazlive.herokuapp.com/trade/products/list

We will get list of products with description:

```
"id": 115900610,
"external_id": null,
"name": "Припой Castolin RB 5286 д.2,0мм, упак.15 прутков",
"sku": "350011",
"main_image": "https://images.satu.kz/219144554_w200_h200_pripoj-castolin-rb.jpg",
        "quantity_in_stock": null,
        "measure_unit": "",
```