Структура таблиц базы данных:

1. Таблица "Пользователи":
- Идентификатор пользователя
- ФИО
- Номер телефона
- Электронная почта
- Пароль

Комментарий: Идентификатор пользователя (id) является первичным ключом в таблице пользователей. Для каждого пользователя не будем выделять роль покупатель/продавец в этой таблице, т.к. один и тот же пользователь может быть в разные моменты времени как продавцом так и покупателем.

2. Таблица "Товары":
- Идентификатор товара
- Название(описание)
- Цена

Комментарий: Выделим товары в отдельную таблицу, чтоб создать связь один ко многим со сделками, т.к. одно и то же наименование товара может продаваться несколько раз.

3. Таблица "Транзакции":
- Идентификатор транзакции
- Идентификатор покупателя
- Идентификатор продавца
- Идентификатор товара
- Сумма транзакции
- Статус транзакции (ожидает подтверждения, завершена (ок), отменена)

Комментарий: Вынесем поля ролей в таблицу с транзакциями, свяжем роли и id пользователя через внешний ключ.

Статусы:

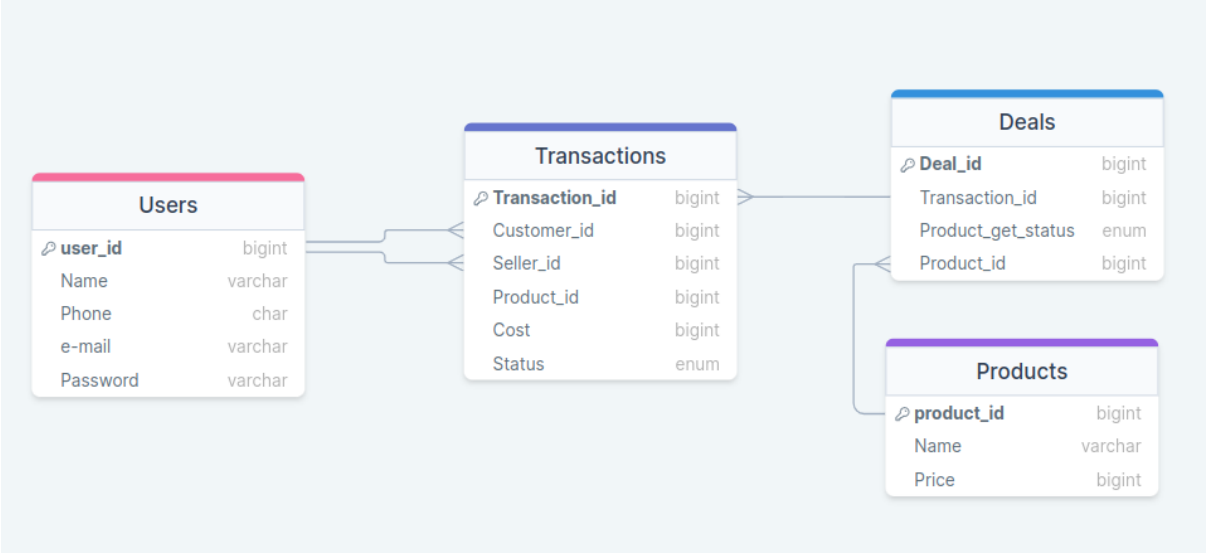
Ожидает подтверждения - покупатель оплатил товар, но получение еще не подтверждено, транзакция в “подвешенном” состоянии

Завершена (ок) - получение подтверждено, транзакция завершена.

4. Таблица "Сделки":
- Идентификатор покупки
- Идентификатор транзакции
- Статус получения товара (получен, не получен)

Комментарий: Создадим дополнительную таблицу со сделками, свяжем ее один ко многим с транзакциями внешним ключом - так мы сможем реализовать N-N взаимодействие, где у каждой сделки может быть несколько разных транзакций, при этом из-за наличия разных ролей в поле таблицы “транзакции” такое

взаимодействие будет двусторонним, т.е. может быть как несколько транзакций с одним покупателем и разными продавцами, так и наоборот. Случаи 1-N и N-1 взаимодействия будут частными для данной схемы.



Спецификация:

URL:

1) …/api/CustomerTransactions

Метод: GET

Комментарий: Предполагается что на стороне бэкенда происходит фильтрация транзакция по их статусу, метод возвращает массив транзакций, у которых статус Wait

2) …/CustomerTransactions/{CustomerTransaction_id}

Методы: GET,PUT

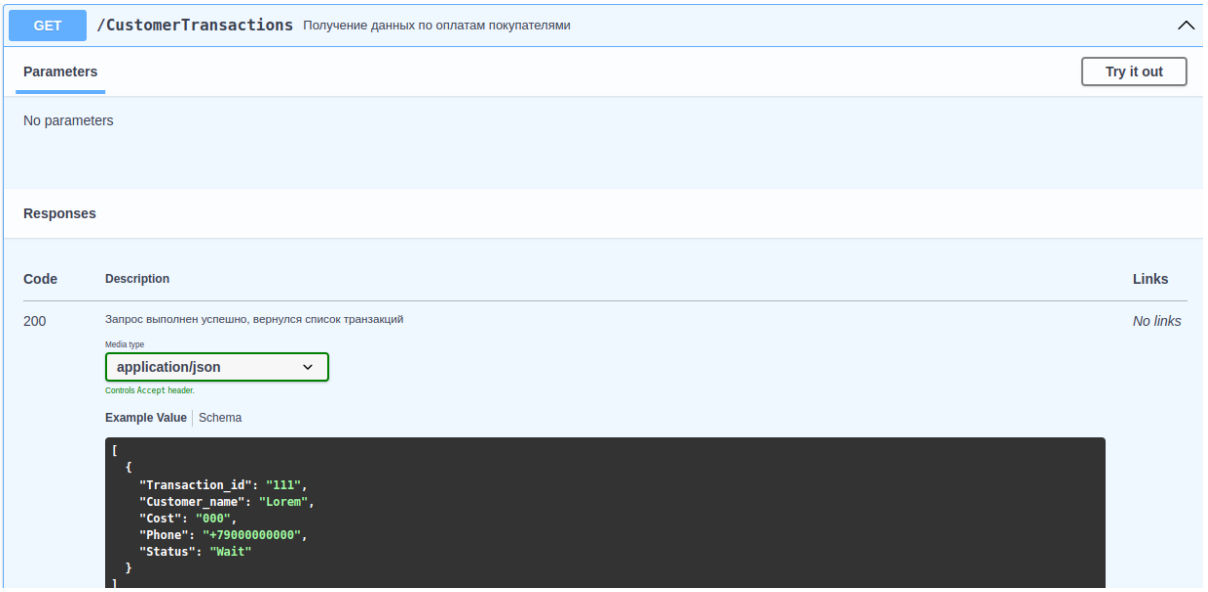
Комментарий: Аналогично при использовании метода GET списку транзакций возвращаться будет транзакция с указанным в параметрах запроса идентификатором. PUT предполагается использовать для отправки “сигнала” о завершении транзакции - запрос обновляет данные в покупательских транзакциях (меняет статус на ok), теперь при фильтрации эта транзакция будет показываться как завершенная (SellerTransaction)

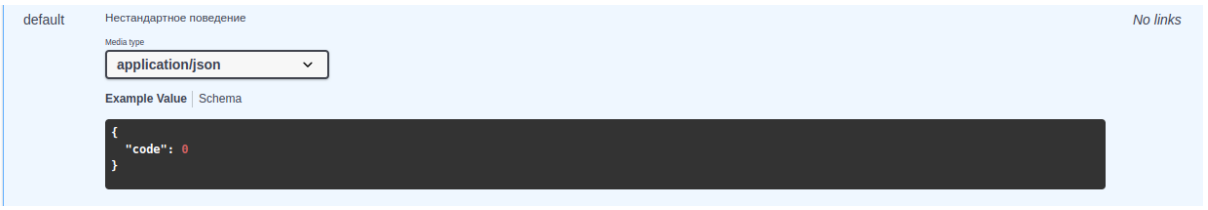
3) …/SellerTransactions/{SellerTransaction_id}

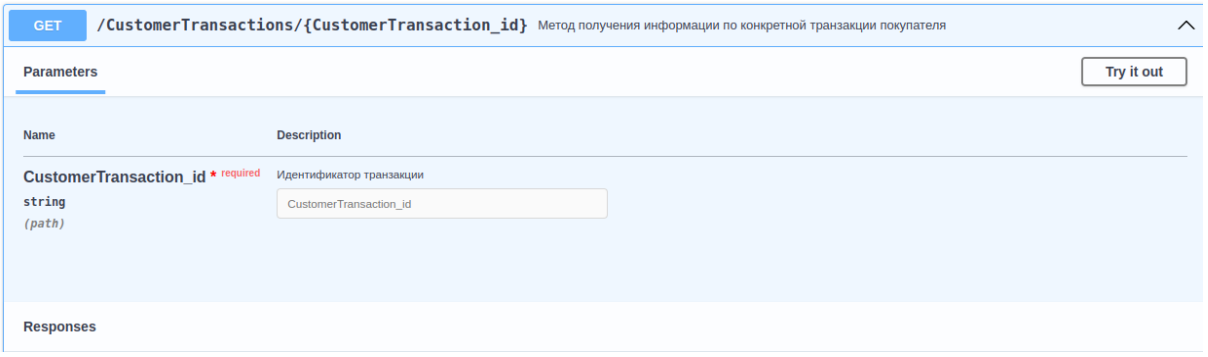
Аналогично пункту 2 за исключением отсутствия реализации метода PUT
4) …/SellerTransactions/

Аналогично пункту 1

Более подробно можно увидеть реализацию на скриншотах и на языке YAML:


GET
/CustomerTransactions
Получение данных по оплатам покупателями
Parameters
Try it out
No parameters
Responses
Code
Description
Links
200
Запрос выполнен успешно, вернулся список транзакций
No links
Media type
application/json
Controls Accept header.
Example Value
Schema
[
{
"Transaction_id": "111",
"Customer_name": "Lorem",
"Cost": "000",
"Phone": "+79000000000",
"Status": "Wait"
}
]


default
Нестандартное поведение
No links
Media type
application/json
Example Value
Schema
{
"code": 0
}


GET
/CustomerTransactions/{CustomerTransaction_id}
Метод получения информации по конкретной транзакции покупателя
Parameters
Try it out
Name
Description
CustomerTransaction_id * required
string
(path)
Идентификатор транзакции
CustomerTransaction_id
Responses

Responses

Code	Description	Links
200	Запрос выполнен успешно, возвращена транзакция по id Media type: application/json Controls Accept header. **Example Value** | Schema `{ "Transaction_id": "111", "Customer_name": "Lorem", "Cost": "000", "Phone": "+79000000000", "Status": "Wait" }`	*No links*
default	Нестандартное поведение Media type: application/json **Example Value** | Schema `{ "code": 0 }`	*No links*

PUT `/CustomerTransactions/{CustomerTransaction_id}` Метод обновления состояния транзакции

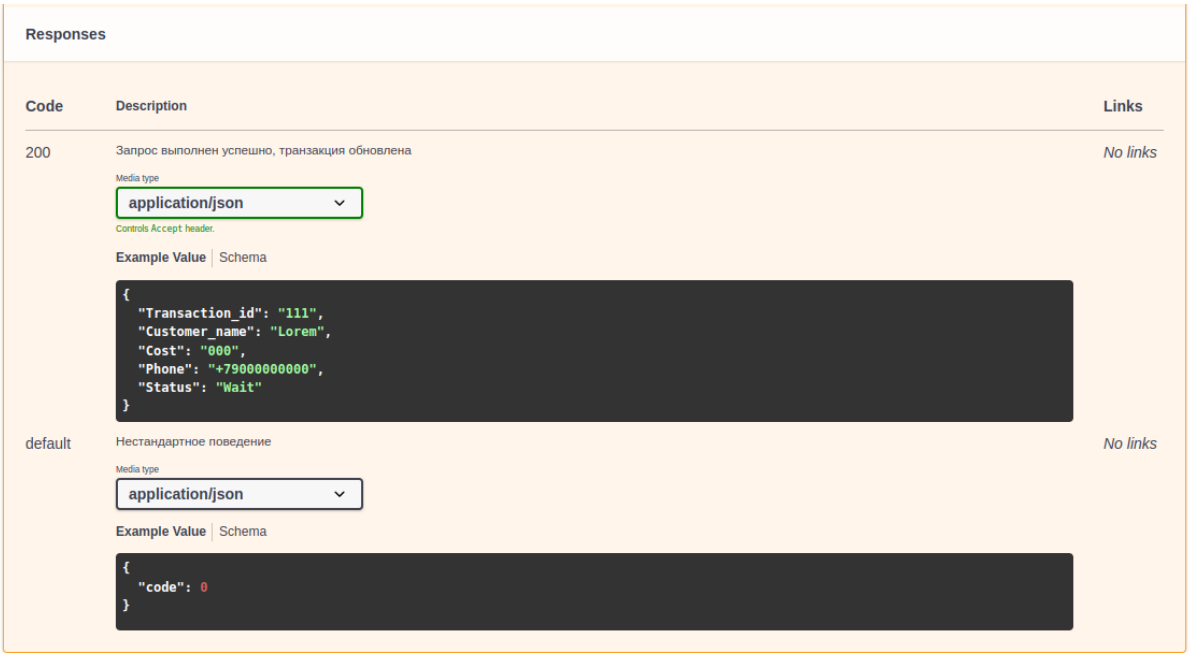
Parameters | Try it out

No parameters

Request body required — application/json

Example Value | Schema

```
{
  "Transaction_id": "111",
  "Customer_name": "Lorem",
  "Cost": "000",
  "Phone": "+79000000000",
  "Status": "Wait"
}
```




```
openapi: 3.0.0
info:
  title: Сервис безопасных сделок
  version: 0.0.1
servers:
  - url: http://localhost:8080/api/
    description: Dev
paths:
  /CustomerTransactions:
    get:
      summary: Получение данных по оплатам покупателями
      responses:
        '200':
          description: Запрос выполнен успешно, вернулся список транзакций
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/CustomerTransactions"

        'default':
          description: Нестандартное поведение
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Error"


 /CustomerTransactions/{CustomerTransaction_id}:
    get:
      summary: Метод получения информации по конкретной транзакции покупателя
   parameters:
```

```
      - name: CustomerTransaction_id
        in: path
        required: true
        description: Идентификатор транзакции
        schema:
          type: string
    responses:
      '200':
        description: Запрос выполнен успешно, возвращена транзакция по id
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/CustomerTransaction"
      'default':
        description: Нестандартное поведение
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/Error"

  put:
    summary: Метод обновления состояния транзакции
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: "#/components/schemas/CustomerTransaction"
    responses:
      '200':
        description: Запрос выполнен успешно, транзакция обновлена
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/CustomerTransaction"
      'default':
        description: Нестандартное поведение
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/Error"

/SellerTransactions/{SellerTransaction_id}:
  get:
    summary: Метод получения информации по конкретной транзакции продавца
    parameters:
      - name: SellerTransaction_id
        in: path
     required: true
        description: Идентификатор транзакции
        schema:
          type: string
    responses:
      '200':
        description: Запрос выполнен успешно, возвращена транзакция по id
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/SellerTransaction"
```

```
        'default':
          description: Нестандартное поведение
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Error"




  /SellerTransactions:
    get:
      summary: Получение данных по выплатам продавцам
      responses:
        '200':
          description: Запрос выполнен успешно, вернулся список транзакций
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/SellerTransactions"

        'default':
          description: Нестандартное поведение
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Error"



components:
  schemas:
    CustomerTransaction:
      type: object
      properties:
        Transaction_id:
          type: string
          example: "111"
        Customer_name:
          type: string
          example: Lorem
        Cost:
          type: string
          example: "000"
     Phone:
          type: string
          example: "+79000000000"
        Status:
          enum:
              - Wait
              - Ok
              - Abort

    SellerTransaction:
      type: object
      properties:
        Transaction_id:
```

```
      type: string
      example: "111"
    Seller_name:
      type: string
      example: Lorem
    Cost:
      type: string
      example: "000"
    Phone:
      type: string
      example: "+79000000000"
    Status:
      enum:
          - Wait
          - Ok
          - Abort

CustomerTransactions:
  type: array
  items:
    $ref: "#/components/schemas/CustomerTransaction"

SellerTransactions:
  type: array
  items:
    $ref: "#/components/schemas/SellerTransaction"

Error:
  type: object
  properties:
    code:
      type: integer
```