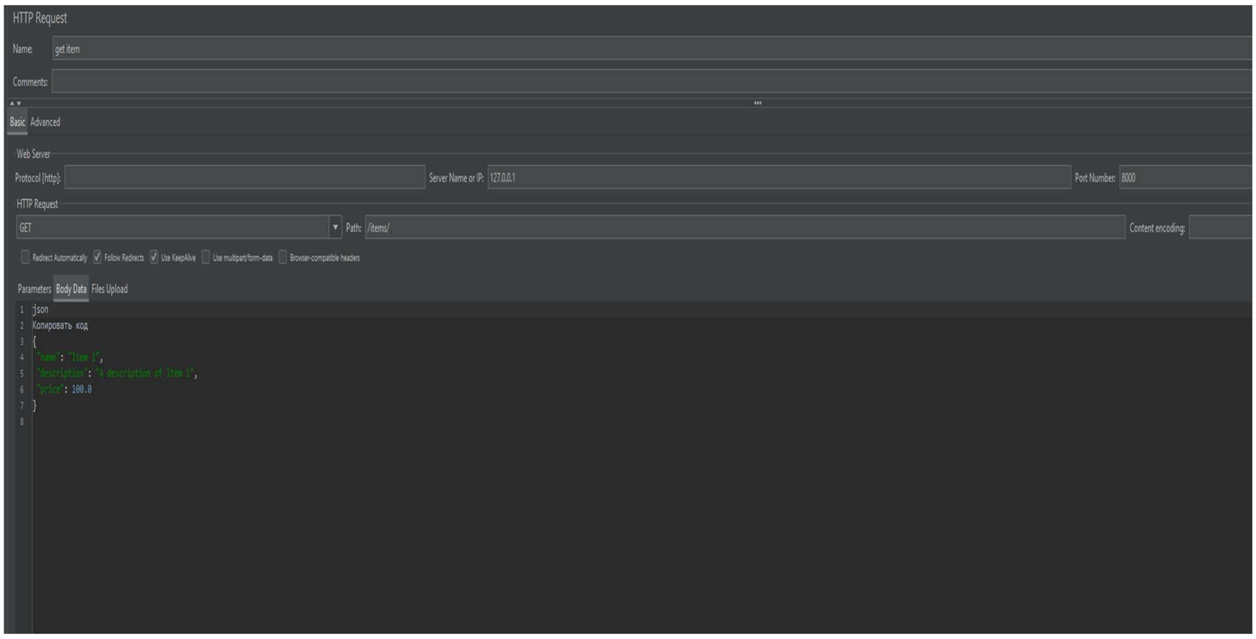
Лаба 5

Листинг кода

```
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
from typing import List, Optional
import sqlite3
from contextlib import contextmanager

app = FastAPI()


# Database connection helper
@contextmanager
def get_db_connection():
    conn = sqlite3.connect('auction.db')
    conn.row_factory = sqlite3.Row
    try:
        yield conn
    finally:
        conn.close()


# Database setup
def init_db():
    with get_db_connection() as conn:
        cursor = conn.cursor()
        cursor.execute('''
        CREATE TABLE IF NOT EXISTS items (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL,
            description TEXT,
            price REAL NOT NULL,
            created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
        )
        ''')
        cursor.execute('''
        CREATE TABLE IF NOT EXISTS bids (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            item_id INTEGER NOT NULL,
            bidder_name TEXT NOT NULL,
            amount REAL NOT NULL,
            bid_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
            FOREIGN KEY (item_id) REFERENCES items (id) ON DELETE CASCADE
        )
        ''')
        conn.commit()


init_db()


class Item(BaseModel):
    name: str
    description: Optional[str] = None
    price: float


class Bid(BaseModel):
    bidder_name: str
    amount: float


@app.post("/items/")
def create_item(item: Item):
    with get_db_connection() as conn:
        cursor = conn.cursor()
        cursor.execute('''
        INSERT INTO items (name, description, price)
        VALUES (?, ?, ?)
        ''', (item.name, item.description, item.price))
        conn.commit()
        item_id = cursor.lastrowid

        cursor.execute('SELECT * FROM items WHERE id = ?', (item_id,))
        new_item = cursor.fetchone()

    return dict(new_item)


@app.get("/items/")
def get_items():
    with get_db_connection() as conn:
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM items ORDER BY created_at DESC')
        rows = cursor.fetchall()

    return [dict(row) for row in rows]


@app.get("/items/{item_id}")
def get_item(item_id: int):
    with get_db_connection() as conn:
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM items WHERE id = ?', (item_id,))
        row = cursor.fetchone()

    if row is None:
        raise HTTPException(status_code=404, detail="Item not found")

    return dict(row)


@app.post("/items/{item_id}/bid")
def place_bid(item_id: int, bid: Bid):
    with get_db_connection() as conn:
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM items WHERE id = ?', (item_id,))
        item = cursor.fetchone()

        if item is None:
            raise HTTPException(status_code=404, detail="Item not found")

        current_price = item['price']

        if bid.amount <= current_price:
            raise HTTPException(
                status_code=400,
                detail=f"Bid must be higher than current price:
${current_price}"
            )

        cursor.execute('UPDATE items SET price = ? WHERE id = ?',
(bid.amount, item_id))
        cursor.execute('INSERT INTO bids (item_id, bidder_name, amount)
VALUES (?, ?, ?)',
                       (item_id, bid.bidder_name, bid.amount))
        conn.commit()

        cursor.execute('SELECT * FROM bids WHERE id = ?',
(cursor.lastrowid,))
        new_bid = cursor.fetchone()

    return dict(new_bid)


@app.get("/")
def reload_root():
    return {"message": "Auction API is running!"}


if __name__ == "__main__":
    import uvicorn

    uvicorn.run(app, host="127.0.0.1", port=8000, workers=True)
```

Создали сервер с локальным адресом

Создаем правила для тестирования сервера



HTTP Request

Name: get item

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: 127.0.0.1 Port Number: 8000

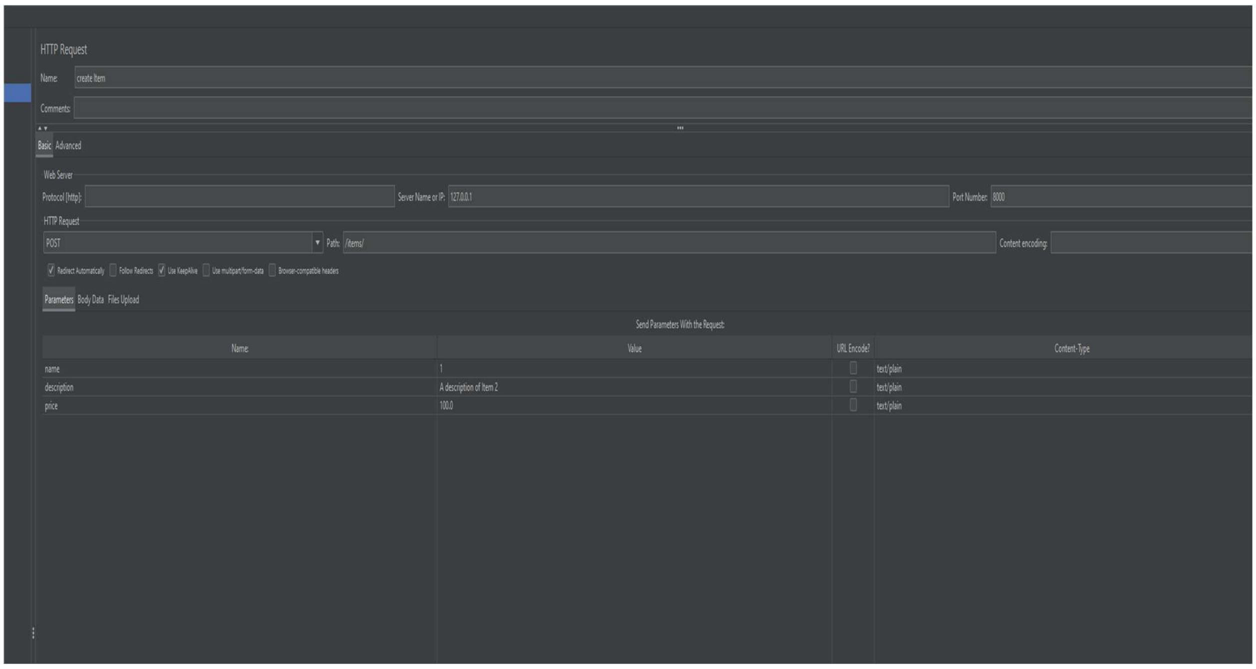
HTTP Request

GET Path: /items/ Content encoding:

Redirect Automatically [✓ Follow Redirects] [✓ Use KeepAlive] Use multipart/form-data Browser-compatible headers

Parameters **Body Data** Files Upload

```
json
Копировать код
{
 "name": "Item 1",
 "description": "A description of Item 1",
 "price": 100.0
}
```



HTTP Request

Name: create Item

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: 127.0.0.1 Port Number: 8000

HTTP Request

POST Path: /items/ Content encoding:

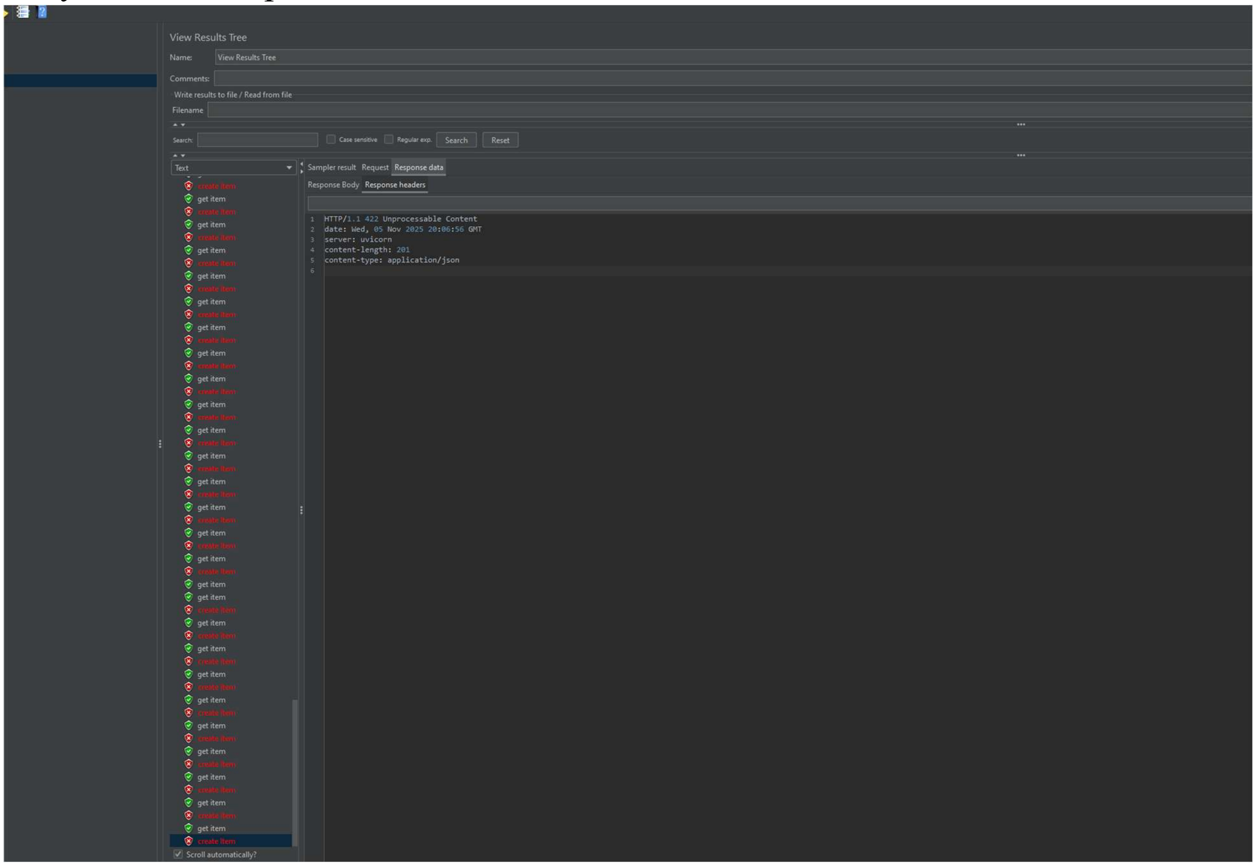
[✓ Redirect Automatically] Follow Redirects [✓ Use KeepAlive] Use multipart/form-data Browser-compatible headers

Parameters Body Data Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type
name	1	☐	text/plain
description	A description of Item 2	☐	text/plain
price	100.0	☐	text/plain

Запускаем тестирование



```
HTTP/1.1 422 Unprocessable Content
date: Wed, 05 Nov 2025 20:06:56 GMT
server: uvicorn
content-length: 201
content-type: application/json
```

По непонятным причинам получаем странные ошибки которые не в состоянии исправить.