

Министерство образования и науки РФ

**АРЗАМАССКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО
УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ «НИЖЕГОРОДСКИЙ
ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ.Р.Е. АЛЕКСЕЕВА»
(АПИ НГТУ)**

Лабораторная работа № 5
по дисциплине
«о Тестирование ПО»

Выполнили:
Зинина А.А.
Прозоровская Я.А.
гр. АЗИС 22-2

Проверил:
Комаров А.О.

Арзамас,
2025 г.

Установили Apache JMeter

Запустили сервер на python, код представлен ниже:

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

app = FastAPI()

# Database connection manager
@contextmanager
def get_db_connection():
    conn = sqlite3.connect('auction.db')
    conn.row_factory = sqlite3.Row # This allows accessing columns by name
    try:
        yield conn
    finally:
        conn.close()

# Database setup with better error handling
def init_db():
    try:
        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 CHECK(price >= 0),
                    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
                )
            ''')
            conn.commit()
    except sqlite3.Error as e:
        print(f"Database initialization error: {e}")

init_db()

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

    model_config = ConfigDict(
        json_schema_extra={
            "example": {
                "name": "Vintage Watch",
                "description": "A beautiful vintage timepiece",
                "price": 150.50
            }
        }
    )
```

```

class Item(ItemCreate):
    id: int
    created_at: Optional[str] = None

    model_config = ConfigDict(from_attributes=True)

# Dependency for database connection
def get_db():
    with get_db_connection() as conn:
        yield conn

@app.post("/items/", response_model=Item, status_code=201)
def create_item(item: ItemCreate, conn: sqlite3.Connection = Depends(get_db)):
    try:
        if item.price < 0:
            raise HTTPException(status_code=400, detail="Price cannot be negative")

        cursor = conn.cursor()
        cursor.execute('''
            INSERT INTO items (name, description, price)
            VALUES (?, ?, ?)
            ''', (item.name, item.description, item.price))
        conn.commit()

        # Get the created item with all fields
        cursor.execute('SELECT * FROM items WHERE id = ?',
            (cursor.lastrowid,))
        row = cursor.fetchone()

        return dict(row)

    except sqlite3.Error as e:
        raise HTTPException(status_code=500, detail=f"Database error: {str(e)}")

@app.get("/items/", response_model=List[Item])
def get_items(conn: sqlite3.Connection = Depends(get_db)):
    try:
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM items ORDER BY created at DESC')
        rows = cursor.fetchall()
        return [dict(row) for row in rows]
    except sqlite3.Error as e:
        raise HTTPException(status_code=500, detail=f"Database error: {str(e)}")

@app.get("/items/{item_id}", response_model=Item)
def get_item(item_id: int, conn: sqlite3.Connection = Depends(get_db)):
    try:
        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)
    except sqlite3.Error as e:
        raise HTTPException(status_code=500, detail=f"Database error: {str(e)}")

@app.put("/items/{item_id}", response_model=Item)
def update_item(item_id: int, item: ItemCreate, conn: sqlite3.Connection = Depends(get_db)):
    try:
        cursor = conn.cursor()
        cursor.execute(''
UPDATE items SET name = ?, description = ?, price = ?
WHERE id = ?
'', (item.name, item.description, item.price, item_id))
        conn.commit()

        if cursor.rowcount == 0:
            raise HTTPException(status_code=404, detail="Item not found")

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

    except sqlite3.Error as e:
        raise HTTPException(status_code=500, detail=f"Database error: {str(e)}")

@app.delete("/items/{item_id}")
def delete_item(item_id: int, conn: sqlite3.Connection = Depends(get_db)):
    try:
        cursor = conn.cursor()
        cursor.execute('DELETE FROM items WHERE id = ?', (item_id,))
        conn.commit()

        if cursor.rowcount == 0:
            raise HTTPException(status_code=404, detail="Item not found")

        return {"message": "Item deleted successfully"}
    except sqlite3.Error as e:
        raise HTTPException(status_code=500, detail=f"Database error: {str(e)}")

# Новые endpoints для дополнительного функционала

@app.get("/items/search/{name}", response_model=List[Item])
def search_items(name: str, conn: sqlite3.Connection = Depends(get_db)):
    """Поиск товаров по имени"""
    try:
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM items WHERE name LIKE ? ORDER BY
created_at DESC',
                        (f'%{name}%',))
        rows = cursor.fetchall()
        return [dict(row) for row in rows]
    except sqlite3.Error as e:
        raise HTTPException(status_code=500, detail=f"Database error: {str(e)}")

```

```

@app.get("/stats/")
def get_stats(conn: sqlite3.Connection = Depends(get_db)):
    """Получение статистики по товарам"""
    try:
        cursor = conn.cursor()

        # Общее количество товаров
        cursor.execute('SELECT COUNT(*) as total_items FROM items')
        total_items = cursor.fetchone()['total_items']

        # Средняя цена
        cursor.execute('SELECT AVG(price) as avg_price FROM items')
        avg_price = cursor.fetchone()['avg_price'] or 0

        # Самый дорогой товар
        cursor.execute('SELECT MAX(price) as max_price FROM items')
        max_price = cursor.fetchone()['max_price'] or 0

        return {
            "total_items": total_items,
            "average_price": round(avg_price, 2),
            "max_price": max_price
        }
    except sqlite3.Error as e:
        raise HTTPException(status_code=500, detail=f"Database error: {str(e)}")

if __name__ == "__main__":
    import uvicorn

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

```

Адрес сервера 127.0.0.1:8000

Создание тестов в JMeter

Теперь создадим тесты для этого бэкенда в JMeter.

1. Создание тестового плана
 - Откройте JMeter.
 - Создайте новый тестовый план (Test Plan).
2. Добавление Thread Group
 - Правый клик на Test Plan → Add → Threads (Users) → Thread Group.
 - Установите "Number of Threads (users)" на 5, "Ramp-Up Period (seconds)" на 10 и "Loop Count" на 1.
3. Добавление HTTP Request Sampler для POST запроса
 - Правый клик на Thread Group → Add → Sampler → HTTP Request.
 - В настройках HTTP Request: Name: Create Item
 - Server Name or IP: 127.0.0.1
 - Port Number: 8000
 - Method: POST
 - Path: /items/
 - Body Data: Вставьте пример данных:

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

Добавление HTTP Request Sampler для GET запроса

-Правый клик на Thread Group → Add → Sampler → HTTP Request.

-В настройках HTTP Request:

Name: Get Items

- Server Name or IP: 127.0.0.1
- Port Number: 8000
- Method: GET
- Path: /items/

Добавление Listener о Правый клик на Thread Group → Add → Listener → View Results Tree.

Запуск тестов - Нажмите кнопку "Start" на панели инструментов JMeter. После завершения теста, просмотрите результаты в "View Results Tree

Результат:

