Подробное руководство по тестированию API

Вот как можно протестировать ваш API различными способами:

**1. Тестирование через браузер**

Самый простой способ - вводить URL-адреса прямо в адресную строку браузера.

**Примеры:**

* Получить первые 10 товаров:

text

http://localhost:8000/api/products/

* Товары от 5000 до 20000 руб.:

text

http://localhost:8000/api/products/?min\_price=5000&max\_price=20000

* Товары с рейтингом ≥ 4.5:

text

http://localhost:8000/api/products/?min\_rating=4.5

**2. Тестирование через Swagger UI (интерактивная документация)**

FastAPI автоматически генерирует документацию:

1. Откройте в браузере:

text

http://localhost:8000/docs

1. Найдите эндпоинт /api/products/
2. Нажмите "Try it out"
3. Заполните параметры фильтрации
4. Нажмите "Execute"

**3. Тестирование через командную строку (cURL)**

Для любителей терминала:

bash

*# Базовый запрос*

curl -X 'GET' 'http://localhost:8000/api/products/' -H 'accept: application/json'

*# С фильтрами*

curl -X 'GET' 'http://localhost:8000/api/products/?min\_price=5000&max\_price=20000&min\_rating=4' -H 'accept: application/json'

**4. Тестирование через Python (requests)**

Создайте отдельный скрипт для тестирования:

python

import requests

*# Базовый запрос*

response = requests.get("http://localhost:8000/api/products/")

print(response.json())

*# Сложный фильтр*

params = {

"min\_price": 5000,

"max\_price": 20000,

"min\_rating": 4,

"min\_feedbacks": 10,

"limit": 5

}

response = requests.get("http://localhost:8000/api/products/", params=params)

print(response.json())

**5. Тестирование через Postman**

1. Создайте новый GET-запрос
2. Укажите URL: http://localhost:8000/api/products/
3. Во вкладке "Params" добавьте параметры:
   * Key: min\_price, Value: 5000
   * Key: max\_price, Value: 20000
4. Отправьте запрос

**6. Примеры ответов**

**Успешный ответ (200 OK):**

json

{

"success": true,

"data": [

{

"id": 1,

"name": "Ноутбук ASUS",

"price": 45000,

"rating": 4.8,

"...": "..."

}

],

"meta": {

"total": 15,

"limit": 10,

"offset": 0

}

}

**Ошибка (500):**

json

{

"detail": "Database connection error"

}

**7. Советы по тестированию**

1. **Проверьте граничные значения**:
   * min\_price=0
   * max\_rating=5
2. **Попробуйте неверные параметры**:
   * min\_price=abc (должна быть ошибка валидации)
3. **Проверьте пагинацию**:
   * limit=1&offset=0 → первый товар
   * limit=1&offset=1 → второй товар
4. **Комбинируйте фильтры**:
   * Цена + рейтинг + отзывы

**8. Автоматическое тестирование (pytest)**

Создайте файл test\_api.py:

python

import pytest

import requests

BASE\_URL = "http://localhost:8000/api/products/"

def test\_basic\_request():

response = requests.get(BASE\_URL)

assert response.status\_code == 200

assert "data" in response.json()

def test\_price\_filter():

response = requests.get(BASE\_URL, params={"min\_price": 5000})

data = response.json()

assert all(item["price"] >= 5000 for item in data["data"])

Запустите тесты:

bash

pytest test\_api.py -v

Выберите наиболее удобный для вас способ тестирования! Для начала рекомендую Swagger UI (п.2) - это самый наглядный вариант.

Протестировал Тестирование через Swagger UI (интерактивная документация) FastAPI автоматически генерирует документацию: Откройте в браузере: text http://localhost:8000/docs Найдите эндпоинт /api/products/ Нажмите "Try it out" Заполните параметры фильтрации Нажмите "Execute" Что дальше, где и как достать данные.

После выполнения запроса в Swagger UI вы можете получить и использовать данные несколькими способами:

1. Непосредственно в Swagger UI

1. После нажатия "Execute" вы увидите:
   * **Response body** - JSON с данными товаров
   * **Response headers** - техническая информация
   * **Curl** - готовую команду для командной строки
2. Пример ответа:

json

{

"success": true,

"data": [

{

"id": 123,

"name": "Ноутбук ASUS VivoBook",

"price": 54990,

"rating": 4.7,

"...": "..."

}

],

"meta": {...}

}

2. Копирование данных для использования

**Вариант A: Вручную**

1. Выделите весь Response body
2. Копируйте (Ctrl+C)
3. Вставляйте в:
   * Файл .json для хранения
   * Онлайн-валидатор JSON (например, <https://jsonformatter.org/>)
   * Свой код Python через json.loads()

**Вариант B: Через Curl**

1. Скопируйте команду из раздела "Curl" в Swagger
2. Выполните в терминале:

bash

curl -X 'GET' 'http://localhost:8000/api/products/?min\_price=50000' -H 'accept: application/json' > output.json

1. Данные сохранятся в файл output.json

3. Программное использование (Python)

Создайте скрипт для работы с API:

python

import requests

import pandas as pd

*# 1. Отправка запроса*

response = requests.get(

url="http://localhost:8000/api/products/",

params={

"min\_price": 50000,

"min\_rating": 4.5,

"limit": 100

}

)

*# 2. Проверка ответа*

if response.status\_code == 200:

data = response.json()

*# 3. Конвертация в DataFrame*

df = pd.DataFrame(data['data'])

*# 4. Сохранение в Excel*

df.to\_excel("filtered\_products.xlsx", index=False)

print(f"Сохранено {len(df)} товаров")

*# 5. Анализ данных*

print(f"\nСредняя цена: {df['price'].mean():.2f} руб.")

print(f"Лучший товар: {df.loc[df['rating'].idxmax()]['name']}")

else:

print(f"Ошибка: {response.status\_code}", response.text)

4. Дальнейшие действия с данными

1. **Анализ**:

python

*# Топ-5 самых дорогих товаров*

print(df.sort\_values('price', ascending=False).head(5))

1. **Визуализация**:

python

import matplotlib.pyplot as plt

df.plot(kind='scatter', x='price', y='rating', alpha=0.5)

plt.title('Зависимость рейтинга от цены')

plt.show()

1. **Интеграция с другими системами**:

python

*# Отправка данных в Telegram*

import telegram

bot = telegram.Bot(token="YOUR\_TOKEN")

bot.send\_message(chat\_id=12345, text=f"Найдено {len(df)} товаров")

5. Для промышленного использования

1. **Автоматизация**:
   * Настройте регулярные запросы через cron или Airflow
   * Добавьте обработку ошибок и повторные попытки
2. **Оптимизация**:

python

*# Пакетная обработка (по 100 товаров за раз)*

for offset in range(0, 1000, 100):

params = {"limit": 100, "offset": offset}

response = requests.get(url, params=params)

*# Обработка данных*

1. **Мониторинг**:
   * Логируйте все запросы
   * Отслеживайте скорость ответа API

Если данные не возвращаются

1. Проверьте:
   * Запущен ли сервер (http://localhost:8000)
   * Нет ли ошибок в консоли сервера
   * Правильность параметров фильтрации
2. Пример диагностики:

python

response = requests.get("http://localhost:8000/api/products/")

print(response.status\_code) *# Должно быть 200*

print(response.text) *# Посмотреть сырой ответ*

Теперь вы можете:

* Сохранять данные в файлы (JSON/Excel)
* Анализировать их в Python
* Строить отчеты и визуализации
* Интегрировать с другими системами

Нужны подробные пошаговые объяснения

Подробное пошаговое руководство: Как получить данные из API

Шаг 1: Открытие документации API

1. Запустите ваш FastAPI сервер (если ещё не запущен)
2. Откройте браузер и введите:

text

http://localhost:8000/docs

1. Вы увидите автоматически сгенерированную документацию Swagger UI

Шаг 2: Нахождение нужного эндпоинта

1. В списке эндпоинтов найдите:

text

GET /api/products/

1. Нажмите на него, чтобы раскрыть детали

Шаг 3: Тестирование эндпоинта

1. Нажмите кнопку "Try it out"
2. Параметры станут редактируемыми
3. Заполните нужные параметры (примеры ниже)

Шаг 4: Примеры параметров для теста

1. **Без фильтров** (первые 10 товаров):
   * Оставьте все поля пустыми
   * Нажмите "Execute"
2. **Фильтр по цене**:

text

min\_price: 50000

max\_price: 100000

1. **Фильтр по рейтингу**:

text

min\_rating: 4.5

1. **Комбинированный фильтр**:

text

min\_price: 30000

max\_price: 60000

min\_rating: 4.2

min\_feedbacks: 10

limit: 20

Шаг 5: Получение результатов

После нажатия "Execute":

1. Сервер обработает запрос (1-3 секунды)
2. Вы увидите:
   * **Код ответа** (200 = успех)
   * **Response body** - данные в JSON формате
   * **Curl** - команда для повторения запроса из терминала

Шаг 6: Копирование данных

1. В разделе "Response body":
2. Нажмите кнопку "Copy" справа от JSON
3. Или выделите текст вручную и скопируйте (Ctrl+C)

Шаг 7: Сохранение данных в файл

1. **Способ 1** (через браузер):
   * Вставьте (Ctrl+V) скопированный JSON в текстовый редактор
   * Сохраните как products.json
2. **Способ 2** (через curl):
   * Скопируйте команду из раздела "Curl"
   * Вставьте в терминал, добавив > filename.json в конец:

bash

curl -X 'GET' 'http://localhost:8000/api/products/?min\_price=50000' -H 'accept: application/json' > products.json

Шаг 8: Использование данных в Python

Создайте файл process\_api.py:

python

import requests

import json

import pandas as pd

*# 1. Делаем запрос к API*

response = requests.get(

"http://localhost:8000/api/products/",

params={

"min\_price": 30000,

"min\_rating": 4.0,

"limit": 50

}

)

*# 2. Проверяем успешность запроса*

if response.status\_code == 200:

*# 3. Парсим JSON ответ*

data = response.json()

*# 4. Преобразуем в pandas DataFrame*

df = pd.DataFrame(data['data'])

*# 5. Сохраняем в Excel*

df.to\_excel("filtered\_products.xlsx", index=False)

*# 6. Выводим статистику*

print(f"Получено товаров: {len(df)}")

print(f"Средняя цена: {df['price'].mean():.2f} руб.")

print(f"Самый популярный бренд: {df['brand'].mode()[0]}")

*# 7. Дополнительный анализ*

print("\nТоп-5 товаров по рейтингу:")

print(df.sort\_values('rating', ascending=False).head(5)[['name', 'price', 'rating']])

else:

print(f"Ошибка {response.status\_code}: {response.text}")

Шаг 9: Визуализация данных

Добавьте в скрипт:

python

import matplotlib.pyplot as plt

*# График распределения цен*

plt.figure(figsize=(10, 6))

plt.hist(df['price'], bins=20, edgecolor='black')

plt.title('Распределение цен')

plt.xlabel('Цена (руб)')

plt.ylabel('Количество товаров')

plt.savefig('prices\_distribution.png')

plt.show()

*# График зависимости рейтинга от цены*

plt.figure(figsize=(10, 6))

plt.scatter(df['price'], df['rating'], alpha=0.5)

plt.title('Зависимость рейтинга от цены')

plt.xlabel('Цена (руб)')

plt.ylabel('Рейтинг')

plt.savefig('price\_rating.png')

plt.show()

Шаг 10: Автоматизация (дополнительно)

1. Создайте файл automate.py:

python

import schedule

import time

from process\_api import main as process\_data

def job():

print("Автоматический сбор данных...")

process\_data()

print("Готово!")

*# Запуск каждый день в 10:00*

schedule.every().day.at("10:00").do(job)

while True:

schedule.run\_pending()

time.sleep(60)

Возможные проблемы и решения:

1. **Нет данных в ответе**:
   * Проверьте параметры фильтрации
   * Убедитесь, что БД содержит товары
   * Попробуйте запрос без фильтров
2. **Ошибка подключения**:
   * Убедитесь, что сервер запущен
   * Проверьте http://localhost:8000 в браузере
3. **Ошибки в Python скрипте**:
   * Установите недостающие библиотеки:

bash

pip install requests pandas matplotlib

Теперь вы можете:

1. Тестировать API через Swagger UI
2. Сохранять результаты в JSON/Excel
3. Анализировать данные в Python
4. Строить графики
5. Настроить автоматический сбор данных

Подробное объяснение: Шаг 7: Сохранение данных в файл Способ 1 (через браузер): Вставьте (Ctrl+V) скопированный JSON в текстовый редактор Сохраните как products.json Способ 2 (через curl): Скопируйте команду из раздела "Curl" Вставьте в терминал, добавив > filename.json в конец: bash curl -X 'GET' 'http://localhost:8000/api/products/?min\_price=50000' -H 'accept: application/json' > products.json

Подробное объяснение: Шаг 7 - Сохранение данных в файл

Способ 1: Через браузер (ручное сохранение)

1. **Копирование данных из Swagger UI:**
   * После выполнения запроса в Swagger UI найдите раздел "Response body"
   * Нажмите кнопку "Copy" справа от JSON-данных (или выделите весь текст вручную и скопируйте через Ctrl+C)
2. **Создание файла:**
   * Откройте любой текстовый редактор (Блокнот, VS Code, Sublime Text и др.)
   * Вставьте скопированный JSON (Ctrl+V)
   * Пример содержимого:

json

{

"success": true,

"data": [

{

"id": 1,

"name": "Ноутбук ASUS",

"price": 54990

}

]

}

1. **Сохранение:**
   * Нажмите "Файл" → "Сохранить как"
   * Выберите папку для сохранения
   * В поле "Имя файла" введите: products.json
   * В "Тип файла" выберите "All files" или "JSON"
   * Нажмите "Сохранить"
2. **Проверка:**
   * Откройте файл двойным кликом (должен открыться в браузере или редакторе)
   * Убедитесь, что данные отображаются корректно

Способ 2: Через curl (автоматическое сохранение)

1. **Копирование команды из Swagger:**
   * В Swagger UI после выполнения запроса найдите раздел "Curl"
   * Скопируйте всю команду (пример):

bash

curl -X 'GET' 'http://localhost:8000/api/products/?min\_price=50000' -H 'accept: application/json'

1. **Модификация команды:**
   * Добавьте > filename.json в конец команды
   * Итоговая команда:

bash

curl -X 'GET' 'http://localhost:8000/api/products/?min\_price=50000' -H 'accept: application/json' > products.json

1. **Выполнение в терминале:**
   * **Windows:**
     1. Откройте командную строку (Win+R → cmd)
     2. Вставьте команду (правой кнопкой → Вставить)
     3. Нажмите Enter
   * **MacOS/Linux:**
     1. Откройте Terminal
     2. Вставьте команду (Cmd/Ctrl+Shift+V)
     3. Нажмите Enter
2. **Проверка результата:**
   * В той же папке, где запускалась команда, появится файл products.json
   * Проверить содержимое можно командой:

bash

cat products.json *# Linux/Mac*

type products.json *# Windows*

Дополнительные варианты:

**Для Windows (если curl не работает):**

1. Установите curl:
   * Скачайте с <https://curl.se/windows/>
   * Или используйте встроенный в Windows 10+ (проверьте curl --version)

**Форматирование JSON для читаемости:**  
Добавьте | python -m json.tool перед сохранением:

bash

curl -X 'GET' 'http://localhost:8000/api/products/' | python -m json.tool > pretty\_products.json

**Пример структуры полученного файла:**

json

{

"success": true,

"data": [

{

"id": 123,

"name": "Ноутбук ASUS VivoBook",

"price": 54990,

"rating": 4.7,

"brand": "ASUS"

}

],

"meta": {

"total": 15,

"limit": 10

}

}

Проблемы и решения:

1. **Файл не создается:**
   * Проверьте права на запись в папке
   * Убедитесь, что в конце команды стоит > filename.json
2. **Пустой файл:**
   * Проверьте, возвращает ли API данные (тестируйте без фильтров)
   * Добавьте флаг -v к curl для диагностики:

bash

curl -v 'http://localhost:8000/api/products/' > debug.log

1. **Ошибка кодировки:**
   * Для Windows явно укажите кодировку UTF-8:

bash

curl ... | Out-File -Encoding UTF8 products.json

Теперь у вас есть:

* Ручной способ сохранения через браузер
* Автоматический способ через curl
* Готовый файл JSON для дальнейшего анализа

Нужно подробное объяснение: Шаг 8: Использование данных в Python Создайте файл process\_api.py: python import requests import json import pandas as pd # 1. Делаем запрос к API response = requests.get( "http://localhost:8000/api/products/", params={ "min\_price": 30000, "min\_rating": 4.0, "limit": 50 } ) # 2. Проверяем успешность запроса if response.status\_code == 200: # 3. Парсим JSON ответ data = response.json() # 4. Преобразуем в pandas DataFrame df = pd.DataFrame(data['data']) # 5. Сохраняем в Excel df.to\_excel("filtered\_products.xlsx", index=False) # 6. Выводим статистику print(f"Получено товаров: {len(df)}") print(f"Средняя цена: {df['price'].mean():.2f} руб.") print(f"Самый популярный бренд: {df['brand'].mode()[0]}") # 7. Дополнительный анализ print("\nТоп-5 товаров по рейтингу:") print(df.sort\_values('rating', ascending=False).head(5)[['name', 'price', 'rating']]) else: print(f"Ошибка {response.status\_code}: {response.text}")

Server busy, please try again later.

Нужно подробное объяснение: Шаг 8: Использование данных в Python Создайте файл process\_api.py: python import requests import json import pandas as pd # 1. Делаем запрос к API response = requests.get( "http://localhost:8000/api/products/", params={ "min\_price": 30000, "min\_rating": 4.0, "limit": 50 } ) # 2. Проверяем успешность запроса if response.status\_code == 200: # 3. Парсим JSON ответ data = response.json() # 4. Преобразуем в pandas DataFrame df = pd.DataFrame(data['data']) # 5. Сохраняем в Excel df.to\_excel("filtered\_products.xlsx", index=False) # 6. Выводим статистику print(f"Получено товаров: {len(df)}") print(f"Средняя цена: {df['price'].mean():.2f} руб.") print(f"Самый популярный бренд: {df['brand'].mode()[0]}") # 7. Дополнительный анализ print("\nТоп-5 товаров по рейтингу:") print(df.sort\_values('rating', ascending=False).head(5)[['name', 'price', 'rating']]) else: print(f"Ошибка {response.status\_code}: {response.text}")

Подробное пошаговое объяснение: Использование данных API в Python (файл process\_api.py)

1. Подготовка скрипта

Создайте новый файл process\_api.py и откройте его в редакторе кода (VS Code, PyCharm и т.д.)

2. Импорт необходимых библиотек

python

import requests *# Для отправки HTTP-запросов к API*

import json *# Для работы с JSON-данными (в данном случае необязателен, так как requests имеет встроенный .json())*

import pandas as pd *# Для работы с табличными данными*

3. Отправка запроса к API (шаг 1)

python

response = requests.get(

"http://localhost:8000/api/products/", *# URL вашего API*

params={ *# Параметры запроса*

"min\_price": 30000, *# Минимальная цена товара*

"min\_rating": 4.0, *# Минимальный рейтинг*

"limit": 50 *# Ограничение количества товаров*

}

)

4. Проверка ответа (шаг 2)

python

if response.status\_code == 200: *# 200 = успешный HTTP-ответ*

*# Действия при успешном ответе*

else:

print(f"Ошибка {response.status\_code}: {response.text}")

5. Парсинг JSON (шаг 3)

python

data = response.json() *# Преобразуем ответ в Python-словарь*

Пример структуры data:

python

{

"success": True,

"data": [

{"id": 1, "name": "Ноутбук", "price": 35000, ...},

...

],

"meta": {...}

}

6. Создание DataFrame (шаг 4)

python

df = pd.DataFrame(data['data']) *# Преобразуем список товаров в таблицу*

Теперь df содержит:

text

id name price rating ...

0 1 Ноутбук ASUS 54990 4.7

1 2 Ноутбук HP 42990 4.5

...

7. Сохранение в Excel (шаг 5)

python

df.to\_excel("filtered\_products.xlsx", index=False)

* Создаст файл filtered\_products.xlsx в текущей папке
* index=False - не сохранять номера строк

8. Анализ данных (шаги 6-7)

python

*# Основная статистика*

print(f"Получено товаров: {len(df)}")

print(f"Средняя цена: {df['price'].mean():.2f} руб.")

print(f"Самый популярный бренд: {df['brand'].mode()[0]}")

*# Детальный анализ*

print("\nТоп-5 товаров по рейтингу:")

print(df.sort\_values('rating', ascending=False) *# Сортировка по рейтингу*

.head(5) *# Первые 5 записей*

[['name', 'price', 'rating']]) *# Выбор конкретных столбцов*

9. Запуск скрипта

1. Откройте терминал в папке с файлом
2. Выполните:

bash

python process\_api.py

1. В результате вы увидите:

text

Получено товаров: 42

Средняя цена: 45678.50 руб.

Самый популярный бренд: ASUS

Топ-5 товаров по рейтингу:

name price rating

12 Ноутбук ASUS 54990 4.9

5 Ноутбук Dell 48990 4.8

...

1. И найдете файл filtered\_products.xlsx в папке

Дополнительные возможности

**Фильтрация данных:**

python

*# Товары с ценой до 40000 руб*

cheap\_products = df[df['price'] < 40000]

**Группировка:**

python

*# Средняя цена по брендам*

brand\_stats = df.groupby('brand')['price'].mean()

**Визуализация (добавьте в конец скрипта):**

python

import matplotlib.pyplot as plt

df['price'].plot(kind='hist', bins=20)

plt.title('Распределение цен')

plt.savefig('prices\_histogram.png')

plt.show()

Возможные ошибки и решения

1. **ModuleNotFoundError**:

bash

pip install requests pandas openpyxl matplotlib

1. **ConnectionError**:
   * Убедитесь, что API сервер запущен
   * Проверьте URL (http://localhost:8000)
2. **KeyError**:
   * Проверьте структуру ответа API (print(data.keys()))
   * Убедитесь, что нужные поля существуют

Этот скрипт можно расширять - добавлять обработку ошибок, логирование, интеграцию с другими системами.