Тестирование API

Способ 1: Тестирование через документацию Swagger UI (Самый простой)

1. Откройте документацию API

После запуска сервера откройте в браузере:

http://127.0.0.1:8000/docs

или если используете порт 8001:

http://127.0.0.1:8001/docs

2. Тестирование GET /api/v1/models

В документации Swagger:

1. Найдите раздел "simulations"
2. Найдите метод "GET /api/v1/models"
3. Нажмите кнопку "Try it out"
4. Нажмите "Execute"
5. Посмотрите результат в разделе "Responses"

3. Тестирование POST /api/v1/simulations/run

В документации Swagger:

1. Найдите метод "POST /api/v1/simulations/run"
2. Нажмите "Try it out"
3. Измените параметры в JSON (или оставьте значения по умолчанию):

```
{
  "server_capacity": 10,
  "model_name": "Service System Demo",
  "experiment_name": "Baseline"
}
```

4. Нажмите "Execute"
5. Посмотрите результат

Результат:

AnyLogic Cloud API Integration 1.0.0 OAS 3.1

/openapi.json

FastAPI приложение для работы с AnyLogic Cloud (демо-ключ)

simulations

POST `/api/v1/simulations/run` Run Simulation

Запуск демо-модели Service System Demo через официальную Python-библиотеку.

Parameters Cancel

No parameters

Request body required — application/json

```
{
  "server_capacity": 8,
  "model_name": "Service System Demo",
  "experiment_name": "Baseline"
}
```

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8001/api/v1/simulations/run' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "server_capacity": 8,
  "model_name": "Service System Demo",
  "experiment_name": "Baseline"
}'
```

Request URL

```
http://127.0.0.1:8001/api/v1/simulations/run
```

Server response

Code	Details
200	**Response body**

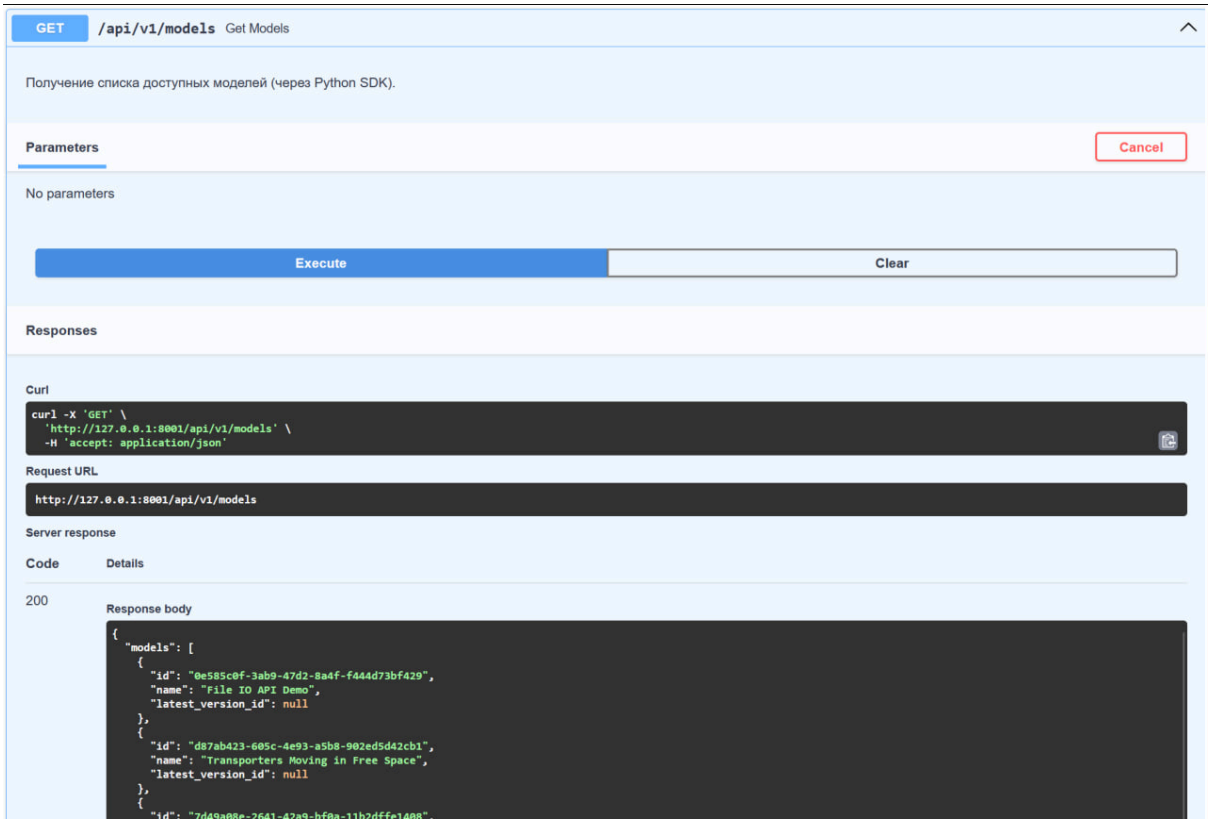

```
{
  "simulation_id": "n/a",
  "server_capacity": 8,
  "mean_queue_size": 0.9988466025848514,
  "server_utilization": 0.3127586081685163,
  "raw_outputs": {
    "Queue size stats": "{\"type\":\"CONTINUOUS\",\"count\":1255584,\"min\":0.0,\"max\":7.0,\"mean\":0.9988466028875719,\"variance\":0.0027334062484944965,\"totalTime\":999999.2699032243}",
    "Total time in system|Total time in system": "{\"statistics\":{\"type\":\"DISCRETE\",\"count\":1000987,\"min\":1.6001570096705109,\"max\":18.533258249757637,\"mean\":2.501238408878067,\"variance\":1.2835661096259896},\"hits\":[800912,159870,29594,5804,3399,1073,257,56,12,9,1,0,0,0,0,0,0,0,0,0],\"hitsOutLow\":0,\"hitsOutHigh\":0,\"lowerBound\":1.6,\"intervalWidth\":1.6}",
    "Utilization|Server utilization": "0.3127586081685163",
    "Mean queue size|Mean queue size": "0.9988466025848514"
  },
  "status": "completed"
}
```

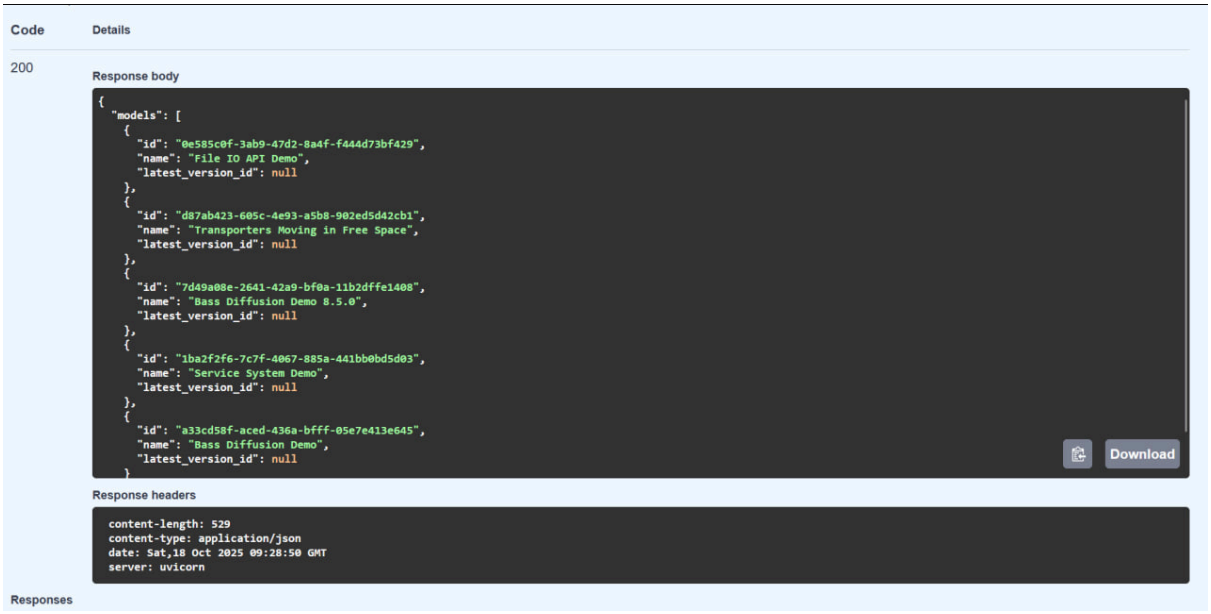

Download

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: *
content-length: 825
content-type: application/json
date: Sat,18 Oct 2025 09:26:32 GMT
server: uvicorn
```

Responses





Способ 2: Тестирование с помощью Python скриптов

Создайте тестовый скрипт `test_api.py`:

import requests
import json

```
# Базовый URL вашего API
BASE_URL = "http://127.0.0.1:8000/api/v1"

def test_get_models():
    """
    Тестирование GET запроса для получения списка моделей
    """
    print("=== Тестирование GET /api/v1/models ===")

    # Формируем полный URL для запроса
    url = f"{BASE_URL}/models"

    try:
        # Выполняем GET запрос
        response = requests.get(url)

        # Проверяем статус ответа
        print(f"Статус код: {response.status_code}")

        # Если запрос успешен (статус 200)
        if response.status_code == 200:
            # Преобразуем JSON ответ в словарь Python
            data = response.json()
            print("✅ Успешный ответ!")
            print(f"Получено моделей: {len(data.get('models', []))}")

            # Выводим информацию о каждой модели
            for model in data.get('models', []):
                print(f"  - Модель: {model.get('name')} (ID: {model.get('id')})")

        else:
            print("❌ Ошибка при запросе")
            print(f"Ответ: {response.text}")

    except Exception as e:
        print(f"❌ Произошла ошибка: {e}")

def test_post_simulation(server_capacity=8):
    """
    Тестирование POST запроса для запуска симуляции

    Args:
        server_capacity (int): Количество серверов для симуляции
    """
    print(f"\n=== Тестирование POST /api/v1/simulations/run ===")
    print(f"Параметр server_capacity: {server_capacity}")

    # Формируем полный URL для запроса
```

```
    url = f"{BASE_URL}/simulations/run"

    # Подготавливаем данные для отправки (тело запроса)
    payload = {
        "server_capacity": server_capacity,
        "model_name": "Service System Demo",
        "experiment_name": "Baseline"
    }

    # Указываем заголовки (Content-Type для JSON)
    headers = {
        "Content-Type": "application/json"
    }

    try:
        # Выполняем POST запрос
        # json=payload автоматически преобразует словарь в JSON и устанавливает
заголовки
        response = requests.post(url, json=payload, headers=headers)

        print(f"Статус код: {response.status_code}")

        if response.status_code == 200:
            data = response.json()
            print("✅ Симуляция успешно выполнена!")
            print(f"ID симуляции: {data.get('simulation_id')}")
            print(f"Размер очереди: {data.get('mean_queue_size')}")
            print(f"Загрузка серверов: {data.get('server_utilization')}")

            # Дополнительная информация
            print("\n📊 Детали результатов:")
            raw_outputs = data.get('raw_outputs', {})
            for key, value in list(raw_outputs.items())[:5]:  # Покажем первые 5 результатов
                print(f"  {key}: {value}")

        else:
            print("❌ Ошибка при выполнении симуляции")
            print(f"Ответ сервера: {response.text}")

    except Exception as e:
        print(f"❌ Произошла ошибка: {e}")

def test_multiple_simulations():
    """
    Тестирование нескольких симуляций с разными параметрами
    """
    print("\n=== Тестирование нескольких симуляций ===")
```

```
    # Тестируем с разным количеством серверов
    for capacity in [5, 8, 12, 15]:
        test_post_simulation(server_capacity=capacity)
        print("-" * 50)

if __name__ == "__main__":
    """
    Главная функция - точка входа в программу
    """
    print("🚀 Начало тестирования AnyLogic FastAPI")
    print("=" * 60)

    # Тест 1: Получение списка моделей
    test_get_models()

    # Тест 2: Одиночная симуляция
    test_post_simulation(server_capacity=10)

    # Тест 3: Несколько симуляций (раскомментируйте для теста)
    # test_multiple_simulations()

    print("\n" + "=" * 60)
    print("✅ Тестирование завершено!")
```

Как запустить тестовый скрипт:

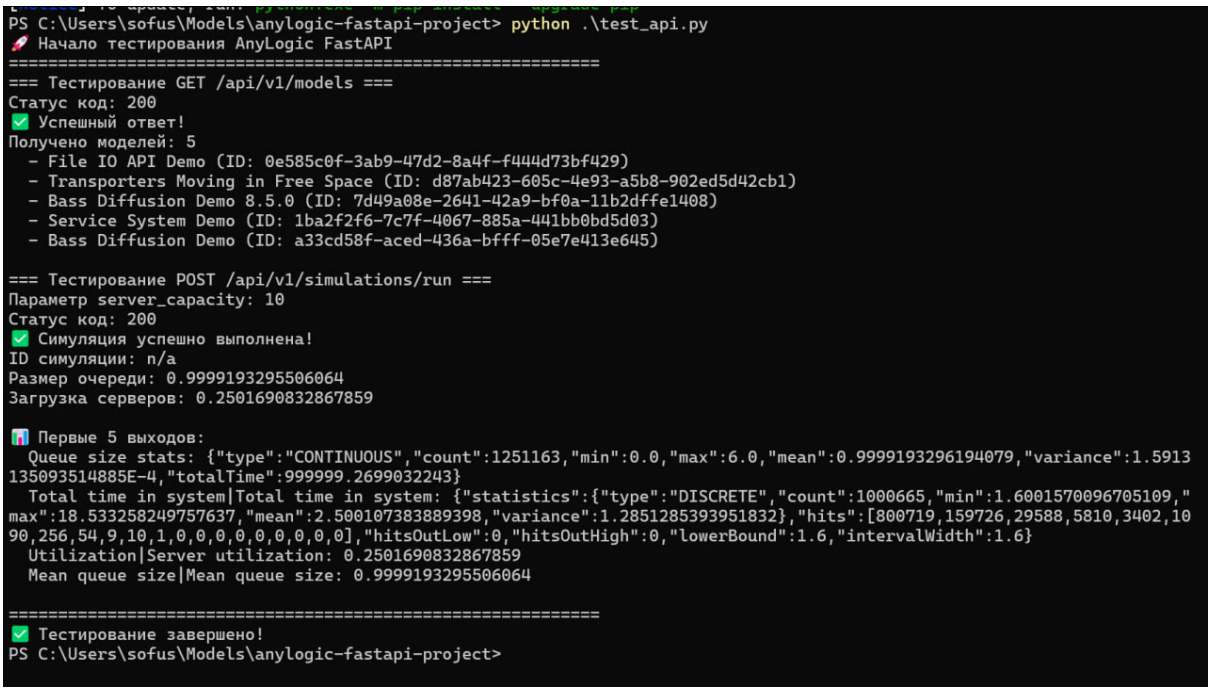
1. **Установите библиотеку requests** (если еще не установлена):

```
pip install requests
```

2. **Запустите скрипт**:

```
python test_api.py
```

Результат:



Способ 3: Тестирование с помощью curl (командная строка)

GET запрос для получения моделей:

```
curl -X 'GET' \
  'http://127.0.0.1:8000/api/v1/models' \
  -H 'accept: application/json'
```

POST запрос для запуска симуляции:

```
curl -X 'POST' \
  'http://127.0.0.1:8000/api/v1/simulations/run' \
  -H 'Content-Type: application/json' \
  -d '{
  "server_capacity": 10,
  "model_name": "Service System Demo",
  "experiment_name": "Baseline"
}'
```

Результат:

```
PS C:\Users\sofus\Models\anylogic-fastapi-project> Invoke-RestMethod -Method Post `
>>   -Uri "http://127.0.0.1:8001/api/v1/simulations/run" `
>>   -ContentType "application/json" `
>>   -Body (@{
>>       server_capacity = 10
>>       model_name      = "Service System Demo"
>>       experiment_name = "Baseline"
>>   } | ConvertTo-Json)


simulation_id      : n/a
server_capacity    : 10
mean_queue_size    : 0,9999193295506064
server_utilization : 0,2501690832867859
raw_outputs        : @{Queue size stats={"type":"CONTINUOUS","count":1251163,"min":0.0,"max":6.0,"mean":0.9999193296194
                     079,"variance":1.5913135093514885E-4,"totalTime":999999.2699032243}; Total time in system|Total ti
                     me in system={"statistics":{"type":"DISCRETE","count":1000665,"min":1.6001570096705109,"max":18.53
                     3258249757637,"mean":2.500107383889398,"variance":1.285128539351832},"hits":[800719,159726,29588,
                     5810,3402,1090,256,54,9,10,1,0,0,0,0,0,0,0,0,0],"hitsOutLow":0,"hitsOutHigh":0,"lowerBound":1.6,"i
                     ntervalWidth":1.6}; Utilization|Server utilization=0.2501690832867859; Mean queue size|Mean queue
                     size=0.9999193295506064}
status             : completed
```


```
PS C:\Users\sofus\Models\anylogic-fastapi-project> curl.exe -X GET "http://127.0.0.1:8001/api/v1/models" -H "accept: application/json"
{"models":[{"id":"0e585c0f-3ab9-47d2-8a4f-f444d73bf429","name":"File IO API Demo","latest_version_id":null},{"id":"d87ab423-605c-4e93-a5b8-902ed5d42cb1","name":"Transporters Movin
g in Free Space","latest_version_id":null},{"id":"7d49a08e-2641-42a9-bf0a-11b2dffe1408","name":"Bass Diffusion Demo 8.5.0","latest_version_id":null},{"id":"1ba2f2f6-7c7f-4067-885a
-441bb0bd5d03","name":"Service System Demo","latest_version_id":null},{"id":"a33cd58f-aced-436a-bfff-05e7e413e645","name":"Bass Diffusion Demo","latest_version_id":null}]}
```


Полный пример с обработкой ошибок

```
import requests
import time

def advanced_api_test():
    """
    Продвинутое тестирование с обработкой ошибок и повторами
    """
    BASE_URL = "http://127.0.0.1:8000/api/v1"

    # Ждем пока сервер запустится
    print("⏳ Ожидание запуска сервера...")
    time.sleep(2)

    # Тестируем доступность сервера
    try:
        health_response = requests.get("http://127.0.0.1:8000/health", timeout=5)
        if health_response.status_code == 200:
            print("✅ Сервер доступен")
        else:
            print("⚠️ Сервер отвечает, но с ошибкой")
    except:
        print("❌ Сервер не доступен. Убедитесь, что он запущен на порту 8000")
        return

    # Тест получения моделей
    print("\n1. Тестируем получение списка моделей...")
    try:
        response = requests.get(f"{BASE_URL}/models", timeout=10)
```

```
    if response.status_code == 200:
        models = response.json().get('models', [])
        if models:
            print(f"✅ Найдено {len(models)} моделей:")
            for model in models:
                print(f"   📁 {model['name']}")
        else:
            print("⚠️ Модели не найдены")
    else:
        print(f"❌ Ошибка HTTP {response.status_code}: {response.text}")

except requests.exceptions.Timeout:
    print("❌ Таймаут запроса")
except requests.exceptions.ConnectionError:
    print("❌ Ошибка подключения")
except Exception as e:
    print(f"❌ Неожиданная ошибка: {e}")

# Тест запуска симуляции
print("\n2. Тестируем запуск симуляции...")
test_data = [
    {"capacity": 5, "expected_queue": "high"},
    {"capacity": 8, "expected_queue": "medium"},
    {"capacity": 12, "expected_queue": "low"}
]

for test in test_data:
    print(f"\n   🧪 Тест с {test['capacity']} серверами:")

    payload = {
        "server_capacity": test["capacity"],
        "model_name": "Service System Demo",
        "experiment_name": "Baseline"
    }

    try:
        response = requests.post(
            f"{BASE_URL}/simulations/run",
            json=payload,
            timeout=30  # Даем больше времени для симуляции
        )

        if response.status_code == 200:
            result = response.json()
            queue_size = result.get('mean_queue_size', 0)
            utilization = result.get('server_utilization', 0)
```

```
            print(f"      ✅ Успех! Очередь: {queue_size:.2f}, Загрузка: {utilization:.1%}")
        else:
            print(f"      ❌ Ошибка {response.status_code}: {response.text}")

    except requests.exceptions.Timeout:
        print("      ❌ Таймаут - симуляция заняла слишком много времени")
    except Exception as e:
        print(f"      ❌ Ошибка: {e}")

if __name__ == "__main__":
    advanced_api_test()
```