🧪 Использование тестового API-ключа

Для доступа к демо-модели вы можете использовать предоставленный AnyLogic тестовый ключ.

- **API-ключ**: `e05a6efa-ea5f-4adf-b090-ae0ca7d16c20`
- **Демо-модель**: Этот ключ предоставляет доступ к демонстрационной модели **"Service System Demo"**.

Имейте в виду, что полнофункциональный API-ключ для работы с вашими собственными моделями доступен только пользователям коммерческих версий AnyLogic Cloud.

🔧 Пример FastAPI приложения

Напишите код простого FastAPI приложения, которое использует клиентскую библиотеку AnyLogic Cloud для запуска демо-модели и получения результатов. main.py:

```
from fastapi import FastAPI, HTTPException
from anylogiccloudclient.client.cloud_client import CloudClient

# Инициализация FastAPI приложения и клиента AnyLogic Cloud
app = FastAPI()
client = CloudClient("e05a6efa-ea5f-4adf-b090-ae0ca7d16c20")

@app.get("/run-simulation/")
async def run_simulation(server_capacity: int = 8):
    """
    Запускает демо-модель 'Service System Demo' с заданным параметром 'Server capacity'.
    Возвращает ключевые результаты моделирования.
    """
    try:
        # Получение последней версии модели по имени
        version = client.get_latest_model_version("Service System Demo")

        # Создание входных параметров на основе эксперимента "Baseline"
        inputs = client.create_inputs_from_experiment(version, "Baseline")

        # Установка своего значения для параметра "Server capacity"
        inputs.set_input("Server capacity", server_capacity)

        # Создание и запуск симуляции
        simulation = client.create_simulation(inputs)

        # Ожидание завершения и получение результатов
        outputs = simulation.get_outputs_and_run_if_absent()
```

```
        # Извлечение конкретных выходных данных
        mean_queue_size = outputs.value("Mean queue size|Mean queue size")
        server_utilization = outputs.value("Utilization|Server utilization")

        return {
            "server_capacity": server_capacity,
            "mean_queue_size": mean_queue_size,
            "server_utilization": server_utilization,
            "raw_outputs": outputs.get_raw_outputs()  # Все сырые выходные данные
        }

    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Ошибка моделирования: {str(e)}")
```

📦 Установка зависимостей

Перед запуском приложения установите необходимые библиотеки. Клиент AnyLogic Cloud нужно скачать по прямой ссылке.

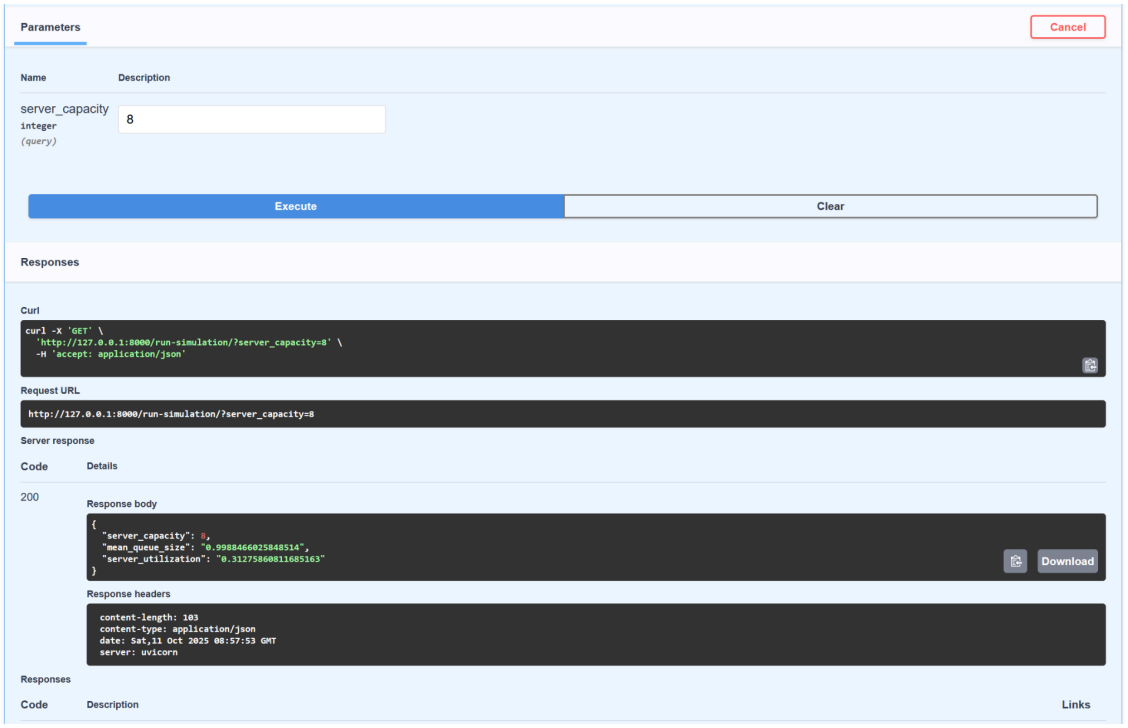
```
# Установка FastAPI и стандартного набора для сервера (например, Uvicorn)
pip install "fastapi[standard]"

# Установка клиентской библиотеки AnyLogic Cloud для Python
pip install
https://cloud.anylogic.com/files/api-8.5.0/clients/anylogiccloudclient-8.5.0-py3-none-any.whl
```

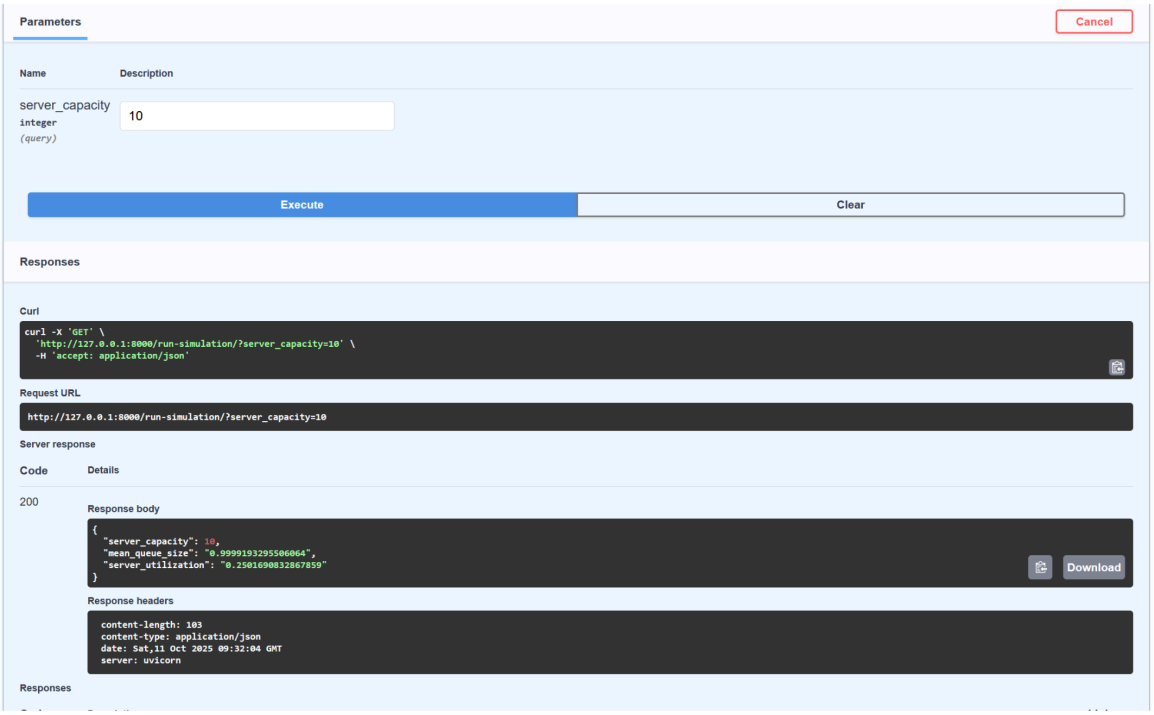
🚀 Запуск и проверка

Откройте браузере документацию API по адресу `http://127.0.0.1:8000/docs`. Вы сможете протестировать эндпоинт `/run-simulation/` прямо из интерфейса Swagger UI, меняя параметр `server_capacity`.Сохраните код в файл, например, `main.py`.
Запустите сервер с помощью команды:

```
 fastapi dev main.py
```

На скринах результат работы, можем менять параметр server_capacity и видеть такие KPI:
Mean queue size|Mean queue size → **0.9988466025848514**
Utilization|Server utilization → **0.31275860811685163**





🔄 Прямые REST API запросы

0) Задаем базовые переменные

```
PS C:\Users\sofus> $API  = "e05a6efa-ea5f-4adf-b090-ae0ca7d16c20"
PS C:\Users\sofus> $BASE = "https://cloud.anylogic.com/api/open/8.5.0"
```

1) GET /models → найти демо и её последнюю версию

```
PS C:\Users\sofus> $models = Invoke-RestMethod -Method Get -Uri "$BASE/models"
-Headers @{ Authorization = $API }
PS C:\Users\sofus>
PS C:\Users\sofus> # найдём «Service System Demo»
PS C:\Users\sofus> $demo    = $models | Where-Object { $_.name -eq "Service System
Demo" }
PS C:\Users\sofus> $modelId = $demo.id
PS C:\Users\sofus>
PS C:\Users\sofus> # demo.modelVersions — массив ID версий. Возьмём последнюю по
номеру версии:
PS C:\Users\sofus> $versions = foreach ($vid in $demo.modelVersions) {
>>   Invoke-RestMethod -Method Get -Uri "$BASE/models/$modelId/versions/$vid" -Headers
@{ Authorization = $API }
>> }
PS C:\Users\sofus> $latest    = $versions | Sort-Object version -Descending | Select-Object
-First 1
PS C:\Users\sofus> $versionId = $latest.id
PS C:\Users\sofus> $versionId
```

Результат:

```
d5c076f5-8967-468d-9757-1f83330241a6
```

2) POST /versions/{version-id}/runs → запустить прогон

```
PS C:\Users\sofus> # соберём полный список inputs из шаблона версии и
переопределим только "Server capacity"
PS C:\Users\sofus> $ver = Invoke-RestMethod -Method Get -Uri
"$BASE/models/$modelId/versions/$versionId" -Headers @{ Authorization = $API }
PS C:\Users\sofus>
PS C:\Users\sofus> $inputs = @()
PS C:\Users\sofus> foreach ($inp in $ver.experimentTemplate.inputs) {
>>   $val = "$($inp.value)"
>>   if ($inp.name -eq "Server capacity") { $val = "8" }  # ← тут меняй параметр
>>   $inputs += @{ name = $inp.name; type = $inp.type; units = $inp.units; value = $val }
>> }
PS C:\Users\sofus>
PS C:\Users\sofus> $runReq = @{ experimentType = "SIMULATION"; inputs = $inputs } |
ConvertTo-Json -Depth 10
PS C:\Users\sofus>
```

```
PS C:\Users\sofus> $run = Invoke-RestMethod -Method Post `
>>   -Uri "$BASE/versions/$versionId/runs" `
>>   -Headers @{ Authorization = $API } -ContentType "application/json" -Body $runReq
PS C:\Users\sofus>
PS C:\Users\sofus> # подождём завершения
PS C:\Users\sofus> do {
>>   Start-Sleep 1
>>   $run = Invoke-RestMethod -Method Post -Uri "$BASE/versions/$versionId/run"
-Headers @{ Authorization = $API } -ContentType "application/json" -Body $runReq
>>   $status = $run.status
>>   Write-Host "status: $status"
>> } while ($status -ne "COMPLETED" -and $status -ne "FAILED" -and $status -ne
"CANCELED")
```

Результат:

```
status: COMPLETED
```

3) POST /versions/{version-id}/results → получить результаты

```
PS C:\Users\sofus> # шаблон выходов (две метрики)
PS C:\Users\sofus> $outputsArray = @(
>>   @{ aggregationType="IDENTITY"; inputs=@(); outputs=@(@{ name="Mean queue
size|Mean queue size" }) },
>>   @{ aggregationType="IDENTITY"; inputs=@(); outputs=@(@{ name="Utilization|Server
utilization" }) }
>> )
PS C:\Users\sofus> # этот эндпоинт ждёт outputs СТРОКОЙ (JSON string)
PS C:\Users\sofus> $outputsString = ($outputsArray | ConvertTo-Json -Depth 6 -Compress)

PS C:\Users\sofus>
PS C:\Users\sofus> $expResultsReq = @{
>>   experimentType = "SIMULATION"
>>   inputs  = $inputs          # полный список инпутов из шага запуска
>>   outputs = $outputsString   # строка!
>> } | ConvertTo-Json -Depth 10
PS C:\Users\sofus>
PS C:\Users\sofus> $results = Invoke-RestMethod -Method Post `
>>   -Uri "$BASE/versions/$versionId/results" `
>>   -Headers @{ Authorization = $API } -ContentType "application/json" -Body
$expResultsReq
PS C:\Users\sofus>
PS C:\Users\sofus> # красиво вывести имя + число
PS C:\Users\sofus> $readable = $results | ForEach-Object { [PSCustomObject]@{ name =
$_.outputs[0].name; value = [double]$_.value } }
PS C:\Users\sofus> $readable
```

Результат:

(Видим те же значения, полученные при server_capacity=8)

name	value
Mean queue size|Mean queue size	0,998846602584851
Utilization|Server utilization	0,312758608116852