



Использование REST API для взаимодействия Asterisk с внешними сервисами.



Обо мне

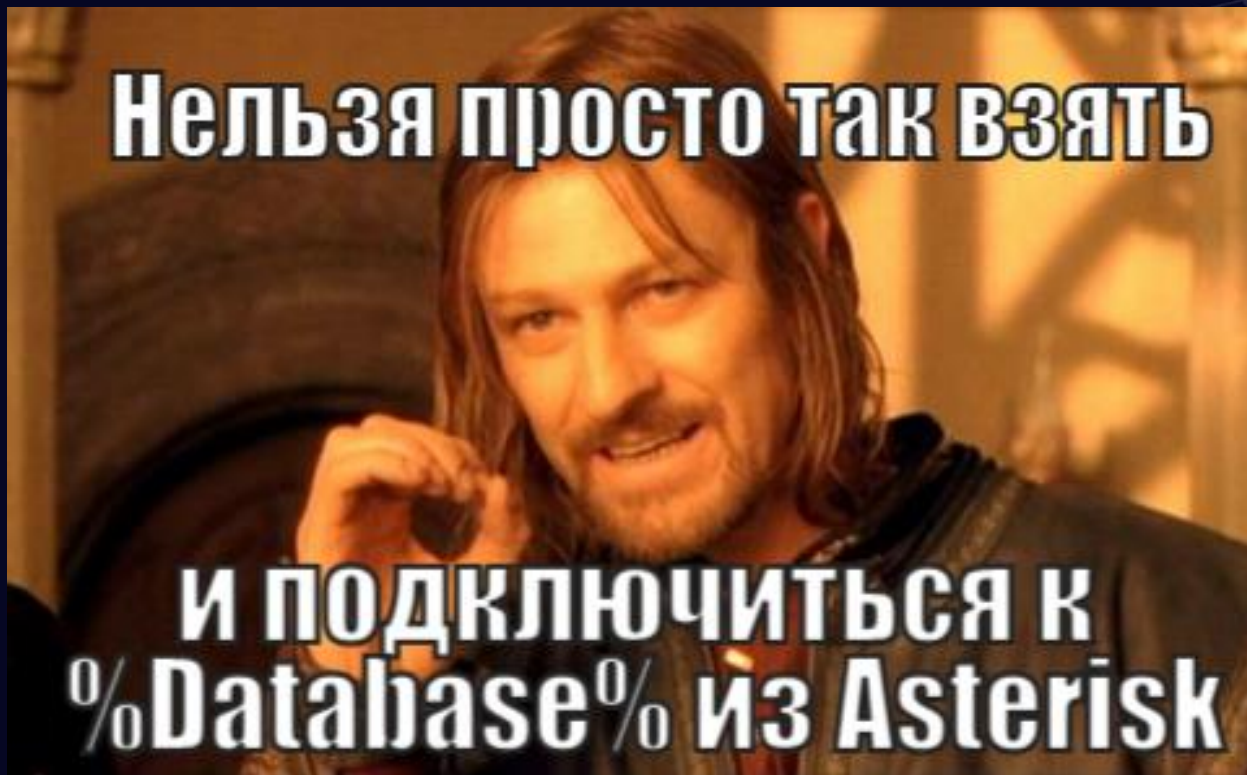


- Евгений Гостьков
- VoIP разработчик в Zadarma

 @gostkov

Промокод: Aster2021

О чем доклад?



Какие варианты есть



	ODBC	System()	Lua	AGI	ARI	REST API
Работа с mysql / postgresql	+	+	+	+	+	+
Работа с другими СУБД / брокерами сообщений	-	+/-	+/-	+	+	+
Удобство использования	-	-	+/-	+	+	+
Постоянное подключение к СУБД	-/+	-	-	+/-	+	+
Code reuse	-	-	+/-	+/-	+/-	+

* app_mysql и cdr_mysql – deprecated.

Что выберем?



Подытожим

Плюсы REST API:

- Можно написать на своём любимом языке программирования
- Можно написать более менее сложную логику и выполнять её 1 простым запросом
- Никак не зависим от используемого ПО(Kamailio, asterisk, freeswitch)
- Можем работать с любой СУБД, брокером сообщений, другим API

Минусы REST API:

- Нужно знать какой-то язык программирования
- Избыточно для простой инсталляции АТС

Разработка API сервиса



На чём написать REST API



**если я не PHP
программист**

Выбираем фреймворк



- Django REST
- Flask
- FastAPI
- Tornado
- aiohttp

Результаты сравнения



	Django	Flask	FastAPI	Tornado	aiohttp
Производительность	-	+/-	+	+/-	+
Функциональность	+	+	++	+	+/-
Удобство использования	+	+	+	+	-
Живой github	+	+	+	+/-	+
Документация	+	+	++	+	+

FastAPI features

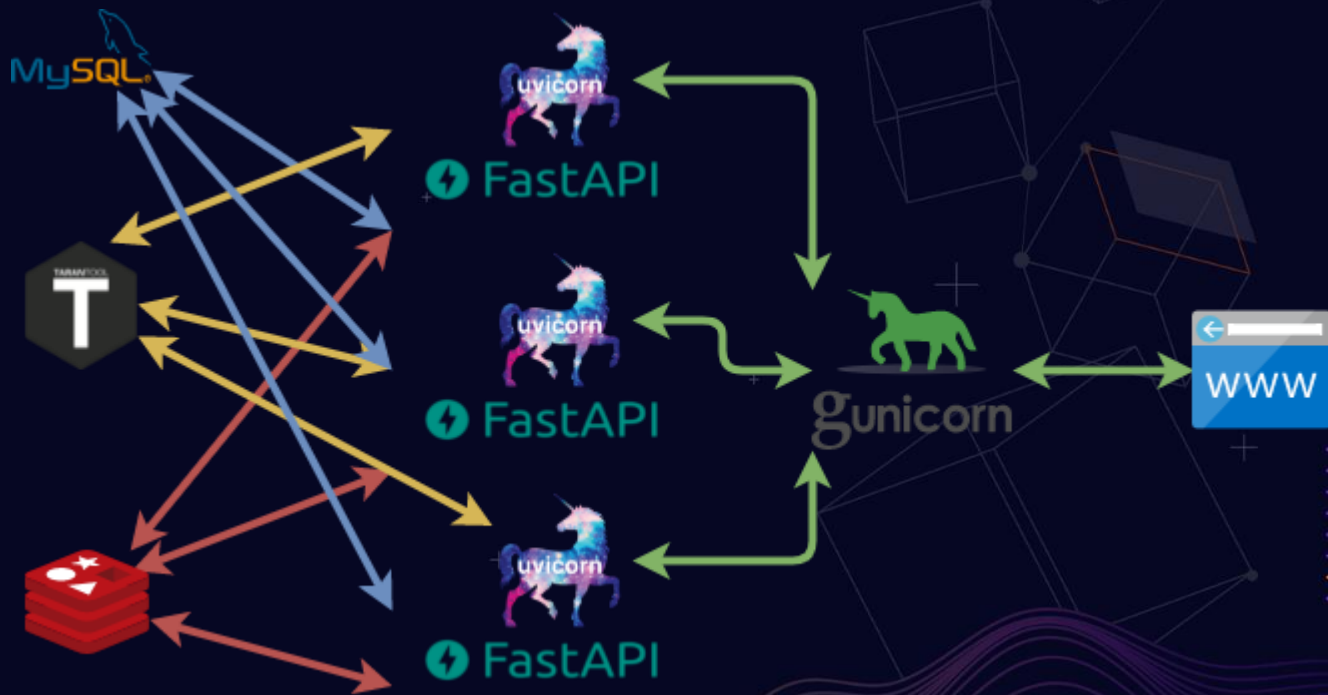


- Автоматическая генерация документации в openapi формате и встроенные интерфейсы Swagger / ReDoc
- Несколько механизмов аутентификации
- Валидация данных с помощью Pydantic
- Возможность запускать синхронные функции
- Background-tasks

Запуск и масштабирование



- Uvicorn – ASGI (Asynchronous Server Gateway Interface) server, который запускает код FastAPI
- Gunicorn - Python WSGI HTTP Server (web-сервер управляющий процессами uvicorn и передающий запросы от http клиентов



Пример простейшего API

```
api_test  api.py
Project
  api_test ~/Documen
    venv
    api.py
  External Libraries
  Scratches and Conso

1 import json
2 import aioredis
3 import uvicorn
4 from aioredis import Redis
5 from prometheus_fastapi_instrumentator import Instrumentator
6 from fastapi.responses import JSONResponse
7 from fastapi import FastAPI
8
9 app = FastAPI()
10 Instrumentator().instrument(app).expose(app)
11 redisconn: Redis
12
13
14 @app.on_event('startup')
15 async def on_startup() -> None:
16     global redisconn
17     print("Connecting to redis")
18     redisconn = aioredis.Redis(connection_pool=aioredis.ConnectionPool.from_url(
19         "redis://123456@127.0.0.1:6379/0", decode_responses=True, encoding="utf-8"))
20
21
22 @app.get('/geo_location/{phone}', tags=["geo"], response_class=JSONResponse)
23 async def geo_location(phone: str):
24     """Return information about the phone number..."""
32     return json.loads(await redisconn.get(phone))
33
34
35 @app.post('/geo_location/', tags=["geo"], response_class=JSONResponse)
36 async def geo_location_p():
37     """..."""
40     return "{}"
41
42
43 if __name__ == '__main__':
44     uvicorn.run(app, host='0.0.0.0', port=8000)
```

Пример простейшего API



```
Connecting to redis
INFO:      Started server process [9385]
INFO:      Waiting for application startup.
INFO:      Application startup complete.
INFO:      Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:      127.0.0.1:48208 - "GET /metrics HTTP/1.1" 200 OK
INFO:      127.0.0.1:48214 - "GET /geo_location/712121 HTTP/1.1" 200 OK
INFO:      127.0.0.1:48214 - "GET /geo_location/712121 HTTP/1.1" 200 OK
INFO:      127.0.0.1:48214 - "GET /geo_location/712121 HTTP/1.1" 200 OK
INFO:      127.0.0.1:48214 - "GET /geo_location/749911 HTTP/1.1" 200 OK
INFO:      127.0.0.1:48214 - "GET /geo_location/749911 HTTP/1.1" 200 OK
INFO:      127.0.0.1:48214 - "GET /metrics HTTP/1.1" 200 OK
```



default

GET /metrics Metrics

geo

GET /geo_location/{phone} Geo Location

Return information about the phone number

Args:

- **phone**:

Parameters

Try it out

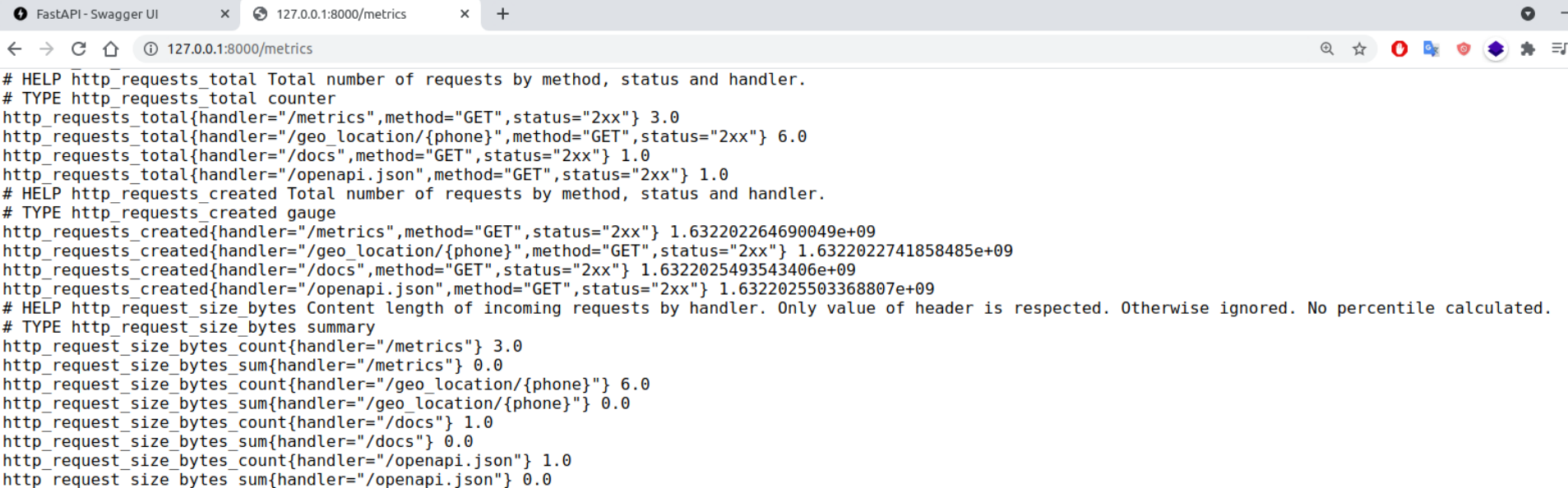
Name	Description
phone * required string (path)	phone

Responses

Code	Description	Links
200	Successful Response	No links

Media type

Метрики для Prometheus



The screenshot shows a browser window with the URL `127.0.0.1:8000/metrics`. The page displays Prometheus metrics for a FastAPI application. The metrics are organized into sections, each starting with a help text and a type declaration. The first section is for `http_requests_total`, which is a counter metric. It shows values for three different handlers: `/metrics` (3.0), `/geo_location/{phone}` (6.0), and `/docs` (1.0). The second section is for `http_requests_created`, which is a gauge metric. It shows values for the same three handlers: `/metrics` (1.632202264690049e+09), `/geo_location/{phone}` (1.6322022741858485e+09), and `/docs` (1.6322025493543406e+09). The third section is for `http_request_size_bytes`, which is a summary metric. It shows values for the same three handlers: `/metrics` (3.0), `/geo_location/{phone}` (6.0), and `/docs` (1.0). The browser's address bar shows the URL `127.0.0.1:8000/metrics` and the page title is `FastAPI - Swagger UI`. The browser's developer tools are not visible.

```
# HELP http_requests_total Total number of requests by method, status and handler.
# TYPE http_requests_total counter
http_requests_total{handler="/metrics",method="GET",status="2xx"} 3.0
http_requests_total{handler="/geo_location/{phone}",method="GET",status="2xx"} 6.0
http_requests_total{handler="/docs",method="GET",status="2xx"} 1.0
# HELP http_requests_created Total number of requests by method, status and handler.
# TYPE http_requests_created gauge
http_requests_created{handler="/metrics",method="GET",status="2xx"} 1.632202264690049e+09
http_requests_created{handler="/geo_location/{phone}",method="GET",status="2xx"} 1.6322022741858485e+09
http_requests_created{handler="/docs",method="GET",status="2xx"} 1.6322025493543406e+09
http_requests_created{handler="/openapi.json",method="GET",status="2xx"} 1.6322025503368807e+09
# HELP http_request_size_bytes Content length of incoming requests by handler. Only value of header is respected. Otherwise ignored. No percentile calculated.
# TYPE http_request_size_bytes summary
http_request_size_bytes_count{handler="/metrics"} 3.0
http_request_size_bytes_sum{handler="/metrics"} 0.0
http_request_size_bytes_count{handler="/geo_location/{phone}"} 6.0
http_request_size_bytes_sum{handler="/geo_location/{phone}"} 0.0
http_request_size_bytes_count{handler="/docs"} 1.0
http_request_size_bytes_sum{handler="/docs"} 0.0
http_request_size_bytes_count{handler="/openapi.json"} 1.0
http_request_size_bytes_sum{handler="/openapi.json"} 0.0
```

Как обратиться к API из Asterisk

```
[apitest]
exten => _XXX,1,Noop("API TEST")
same => n,Set(API_RESPONSE=${CURL(http://api.test.priv/geo_location/79699996701?token=12345)} )
same => n,Noop(API_RESPONSE)
same => n,Hangup()
```

```
[111@apitest:1] NoOp("PJSIP/501-00000012", ""API TEST") in new stack
[111@apitest:2] Set("PJSIP/501-00000012", "API_RESPONSE={"did":"79699996701","iso":"RU","ip":"78.29.41.17"} ")
[111@apitest:3] NoOp("PJSIP/501-00000012", "{"did":"79699996701","iso":"RU","ip":"78.29.41.17"} ") in new stack
[111@apitest:4] Hangup("PJSIP/501-00000012", "") in new stack
```

Спасибо за внимание!

У кого-нибудь есть вопросы?

tg @gostkov
zadarma.com

Промокод: Aster2021