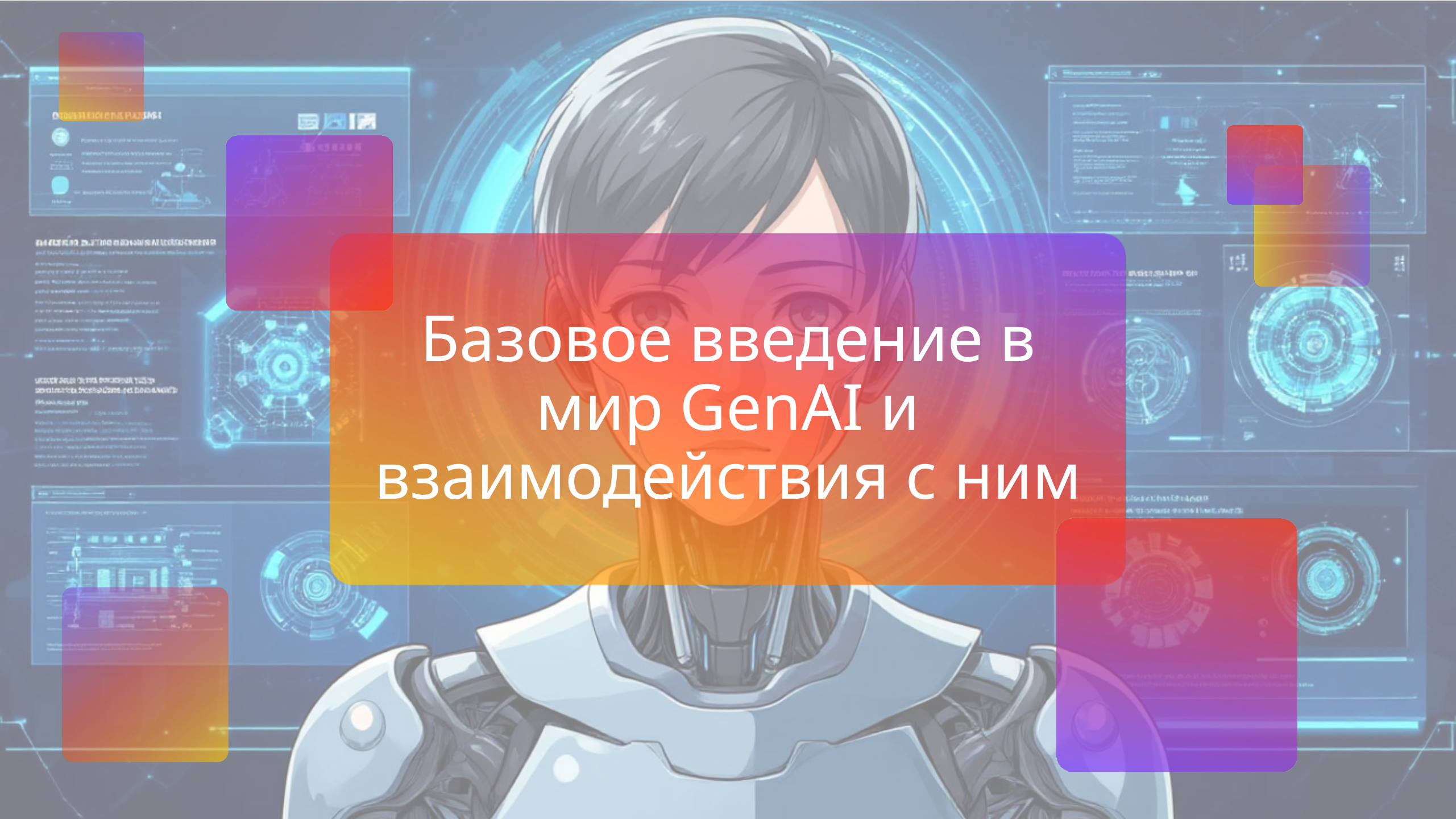

Базовое введение в
мир GenAI и
взаимодействия с ним

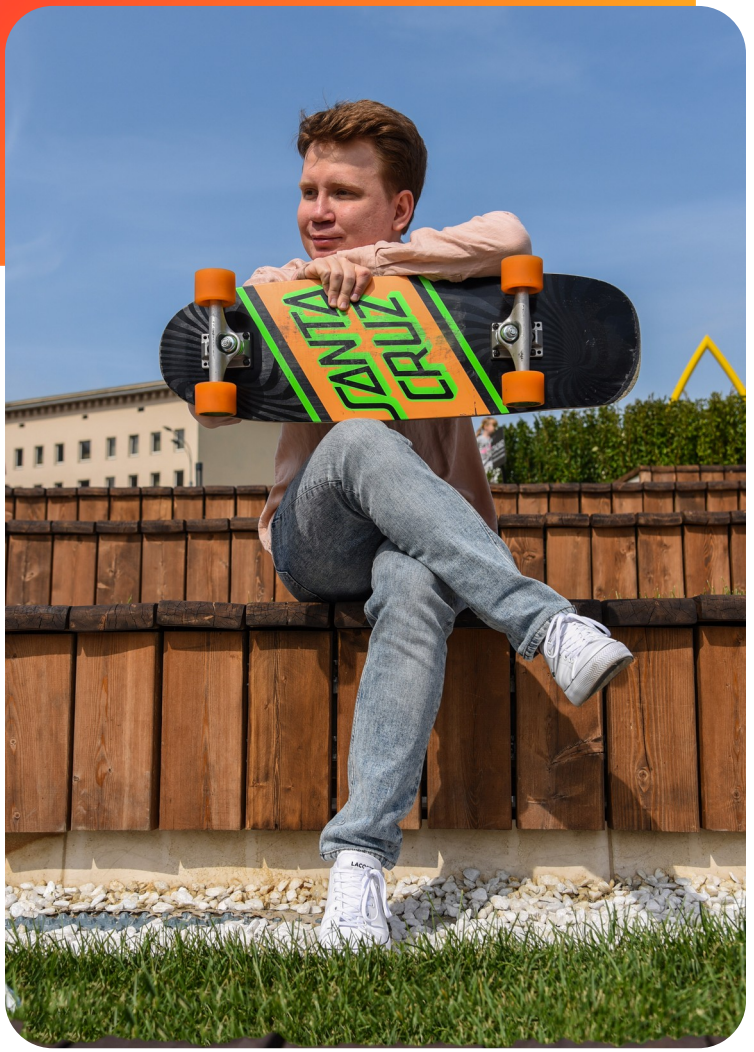
А кто я?

Стаж в компании более 5 лет.

- Принял участие больше чем в 10 проектах.
- Сопровождал запуск 2 проектов «с нуля» до ОЭ, ПЭ.
- Выполнил миграцию проектов между системами непрерывной интеграции и контроля версий.
- Принял участие в запуске школы DevOps.
- Являюсь наставником и обучаю стажеров.

Пётр Галонза

управляющий эксперт отдела внедрения системных сервисов



Почему мы игнорируем AI

- Не хватает времени
- Нет доверия к хайпу
- Нет подходящего железа
- Переусложненные или поверхностные статьи и книги
- Ограничения на использования зарубежных реализаций
- Проще найти/решить самому

Процесс постановки задачи стажеру сравнимо с написанием промпта для GPT

GPT и LLM: за 60 секунд

GenAI Generative Artificial Intelligence (Генеративный искусственный интеллект) – класс систем ИИ, способный создавать новый, оригинальный контент на основе изученных закономерностей в
данных. Вместо того чтобы только распознавать или классифицировать данные, он со здаёт: тексты, изображения,аудио, в видео и.д.

LLM Large Language Model (большая языковая модель) - это тип искусственного интеллекта на основе нейронной сети с огромным числом параметров (от миллиардов до триллионов), обученный работать с текстом на естественном или формальном (например, языке программирования) языке.

GPT Generative Pre-trained Transformer (генеративный предварительно обученный трансформер) -
это семейство больших языковых моделей (LLM), реализующих архитектуру «тра нсформер» и разработанных компанией OpenAI.

Трансформер - это архитектура, специально созданная для обработки последовательных данных (текст, речь, временные ряды) которая впервые была описана в статье 2017 года Attention Is All You Need за авторством членов команды Google Brain.

Затем в 2018 году OpenAI опубликовала статью Improving Language Understanding by Generative Pre-Training, в которой представила модель Generative Pre-trained Transformer, также известную как GPT.

GPT - с одной стороны это продукт компании OpenAI который стал мощным брендом ассоциирующимся с передовыми технологиями ИИ, с другой может упоминаться как архитектура и использоваться в именовании других моделей.

Токены, токенизация и эмбеддинги — «кирпичики» работы LLM

Токен

Это минимальная единица текста.

Что может быть токеном:

- отдельное слово («кот»)
- часть слова («без-» в «беззвучный»)
- знак препинания («,»)
- пробел или перенос строки
- редкий символ или эмодзи

Токенизация

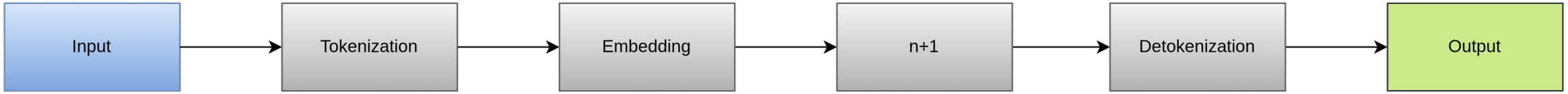
Это процесс преобразования текста в последовательность токенов. Каждый токен получает уникальный числовой идентификатор (ID) согласно словарю, который модель использует для дальнейшей обработки.

Эмбеддинг

Идентификаторы токенов преобразуются в числовые векторы («смысловые координаты»). В многомерном пространстве близкие по смыслу слова (например, «кот» и «кошка») имеют схожие векторы, а далёкие по значению («кот» и «автомобиль») — существенно различающиеся.

Промпт-инжиниринг

Методика создания текстовых инструкций (промптов), которые позволяют эффективно взаимодействовать с большими языковыми моделями (LLM). Грамотно составленный промпт сокращает число итераций и повышает качество генерируемого контента.



LLM - это вероятностная машина, которая на каждом шаге:

- анализирует весь предыдущий контекст
- вычисляет вероятности всех возможных следующих токенов
- выбирает один из них согласно заданной стратегии декодирования.

Ключевые технологии AI: агенты, ассистенты, RAG, MCP

AI-Агенты

Автономные системы, способные:

- воспринимать задачу
- разбивать её на подзадачи
- использовать внешние инструменты (поиск, API, выполнение кода)
- принимать решения для достижения цели без постоянного участия человека

AI-ассистенты

Интерактивные системы, помогающие пользователю в режиме «вопрос-ответ».
В отличие от агентов:

- не действуют автономно
- требуют постоянного участия человека
- фокусируются на диалоге и подсказках (например, чат-боты)
- Сохраняют контекст общения

MCP

Model Context Protocol

Открытый стандарт/протокол для интеграции/взаимодействия в внишними источниками данных и инструментами разработанный компанией Anthropic и предложен в 2024 году. Является надстройкой над JSON-RPC 2.0 и соответственно имеет клиент-серное взаимодействие.

RAG

Retrieval-Augmented Generation(Дополненная генерация с поиском)

Это технология, объединяющая поиск информации и генерацию текста с помощью больших языковых моделей (LLM). Она позволяет получать более точные и обоснованные ответы, опираясь на актуальные внешние данные.

«Когда RAG на горе свистнет. Оцениваем качество генерации с дополненной выборкой». Михаил Костецкий, управляющий эксперт (QA) ПСБ

Преимущества:

- Мгновенная актуализация
- Экономическая эффективность
- Контролируемость контента
- Снижение рисков “галлюцинаций”
- Безопасность данных
- Быстрая адаптация под задачи
- Лёгкость аудита и исправления ошибок

С чего начать?

Локальные

Плюсы

- Полная конфиденциальность данных (нет передачи вовне)
- Работа в офлайн-режиме (без интернета)
- Отсутствие подписок и платежей за API
- Нет лимитов на число запросов
- Гибкая настройка параметров модели

Минусы

- Высокие требования к оборудованию (RAM, GPU, HDD/SSD)
- Актуальность данных на которых была обучена модель
- Требуется дополнительная настройка для интеграций

SaaS

Плюсы

- Не требуют мощного локального оборудования (всё работает на серверах провайдера).
- Простая настройка и быстрый старт (доступ через API/WEB-интерфейс).
- Автоматическая актуализация моделей (провайдер сам обновляет веса и функционал).
- Высокая производительность (оптимизированные дата-центры, GPU-кластеры).

Минусы

- Зависимость от интернет-соединения (без сети — нет доступа)
- Риски утечки данных (информация передаётся на сторонние серверы)
- Жёсткие лимиты на запросы (rate limits, квоты)
- Отсутствие контроля над данными (провайдер может логировать диалоги)
- Регуляторные ограничения
- Взаимодействие с API, IDE Ассистентами, WEB-интерфейсом тарифицируются по разному

Способы взаимодействия

- API
- CLI
- SDK
- Библиотеки
- Web/Desktop/Mobile APP
- IDE(Расширения)
- ChatBot

Отечественные решения

Yandex AI

- API(Rest, GRPC)
- IDE(Code Assistant)
- Web UI
- SDK(Python)
- Библиотеки(LangChain, Llamaindex)
- 10 Моделей
- Отключение логирования
- API тарифицируется по токенам

Gigachat

- API(Rest, GRPC)
- IDE(GigaCode)
- Web UI
- SDK(Python, JS, Java)
- Библиотеки(LangChain, Llamaindex)
- 3 Модели
- API 1000 бесплатных токенов на год. Тарифы годовые. Embeddings как отдельный тариф.
- Используетя Oauth и токен действует 30 минут

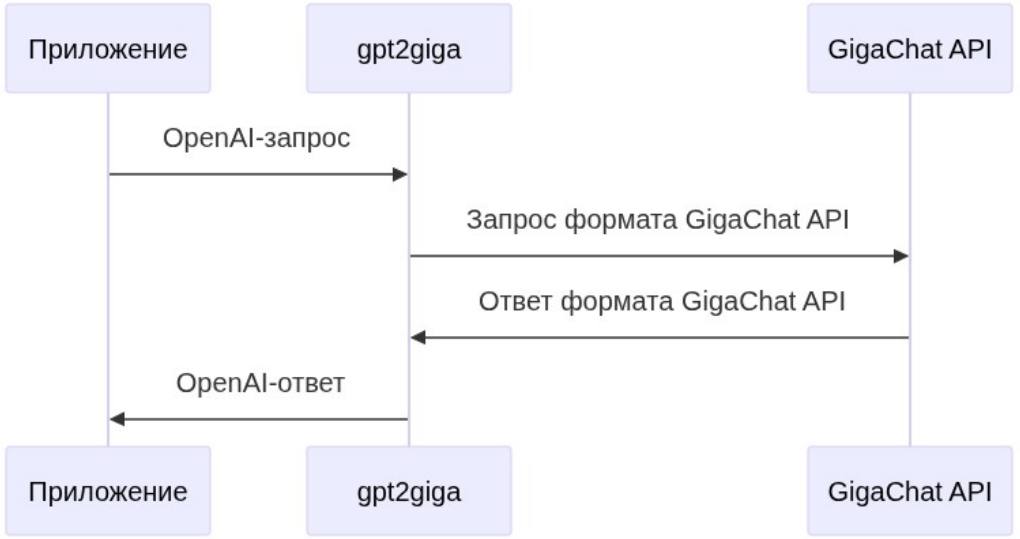
Совместимость с OpenAI

GigaChat:

- Gpt2giga(прокси-сервер на python)
- Частичная совместимость с OpenAI API

YandexGPT:

- Частичная совместимость с OpenAI API



Настройка OpenAI для работы с AI Studio

Чтобы использовать модели генерации текста AI Studio в библиотеках OpenAI, измените базовый эндпоинт, а также укажите API-ключ сервисного аккаунта и идентификатор каталога, в котором создан этот сервисный аккаунт.

Python **Node.js**

```
import openai

client = openai.OpenAI(
    api_key="<значение_API-ключа>",
    base_url="<эндпоинт_API>",
    project="<идентификатор_каталога>"
)
```

Для работы с Completions API укажите эндпоинт `https://llm.api.cloud.yandex.net/v1`.
Для запросов к Responses API и Vector Store API используйте эндпоинт `https://rest-assistant.api.cloud.yandex.net/v1`.
Для создания голосового агента и работы с Realtime API через веб-сокеты укажите `wss://rest-assistant.api.cloud.yandex.net/v1/realtime/openai?model=gpt://<идентификатор_каталога>/speech-realtime-250923`.

Как получить API-ключ для работы с AI Studio.

—

Chat Completions | Функции | Эмбеддинги | Модели

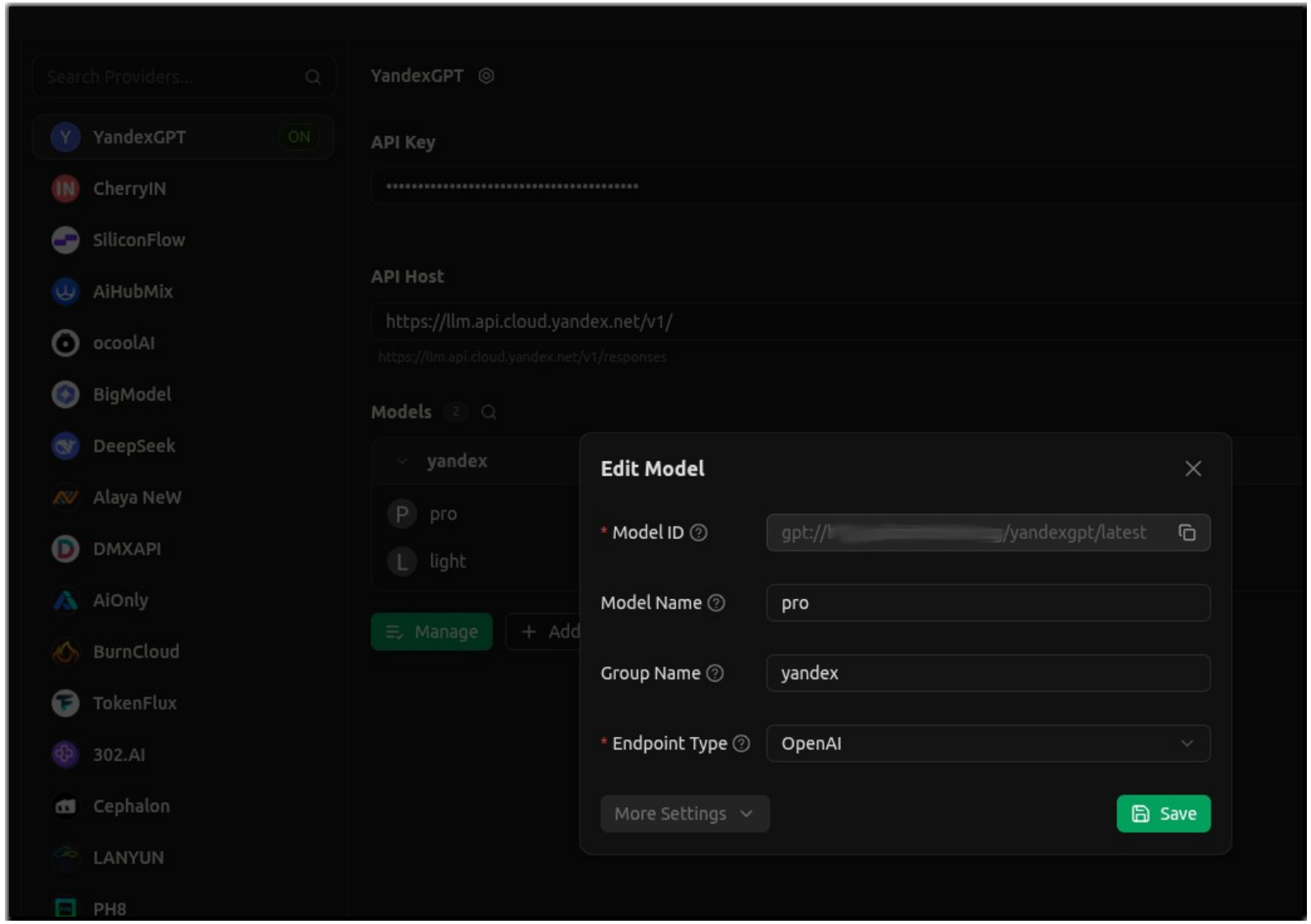
Пример создания запроса:

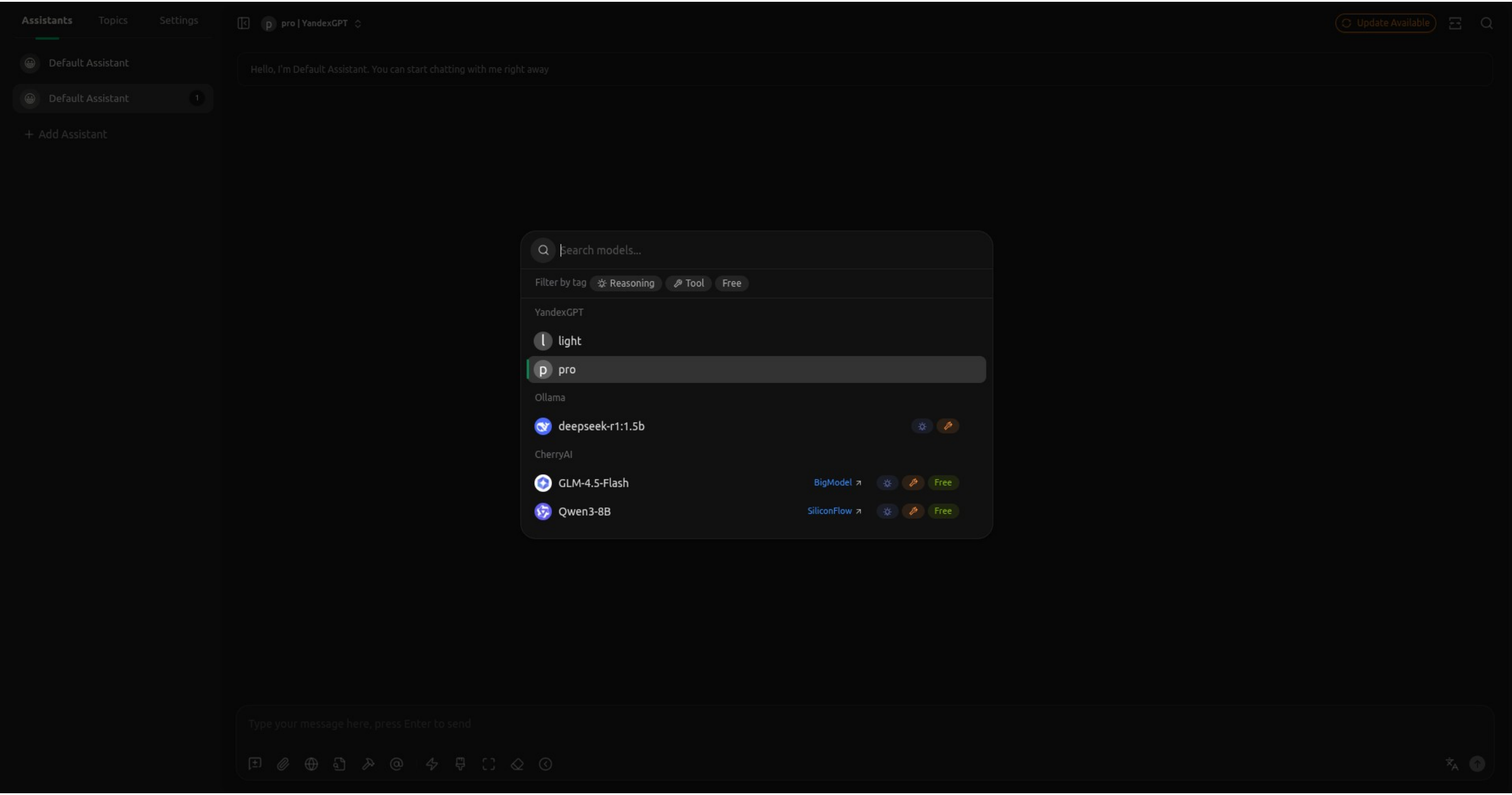
```
from openai import OpenAI

client = OpenAI(api_key= "<токен_доступа>",
                base_url= "https://gigachat.devices.sberbank.ru/api/v1"
                )
completion = client.chat.completions.create(
    model="GigaChat",
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {
            "role": "user",
            "content": "Hello World!"
        }
    ]
)
```

Текущие ограничения

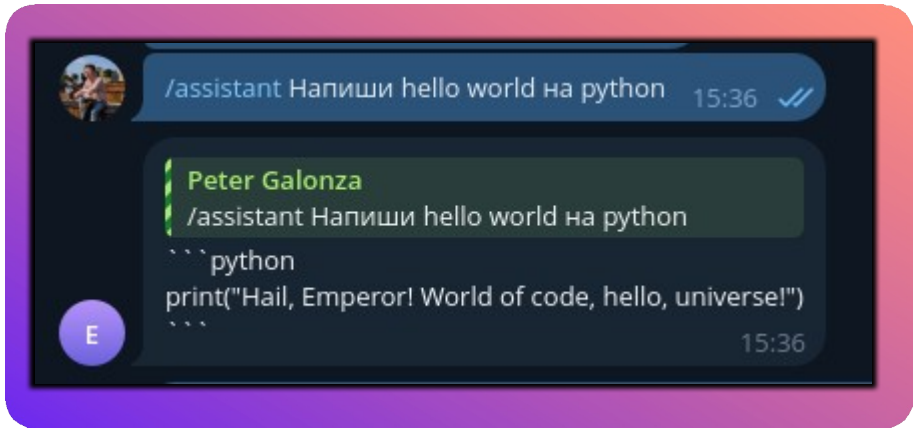
GigaChat API частично совместим с OpenAI API. Если вы еще не используете OpenAI SDK, рекомендуем сразу использовать библиотеки GigaChain.

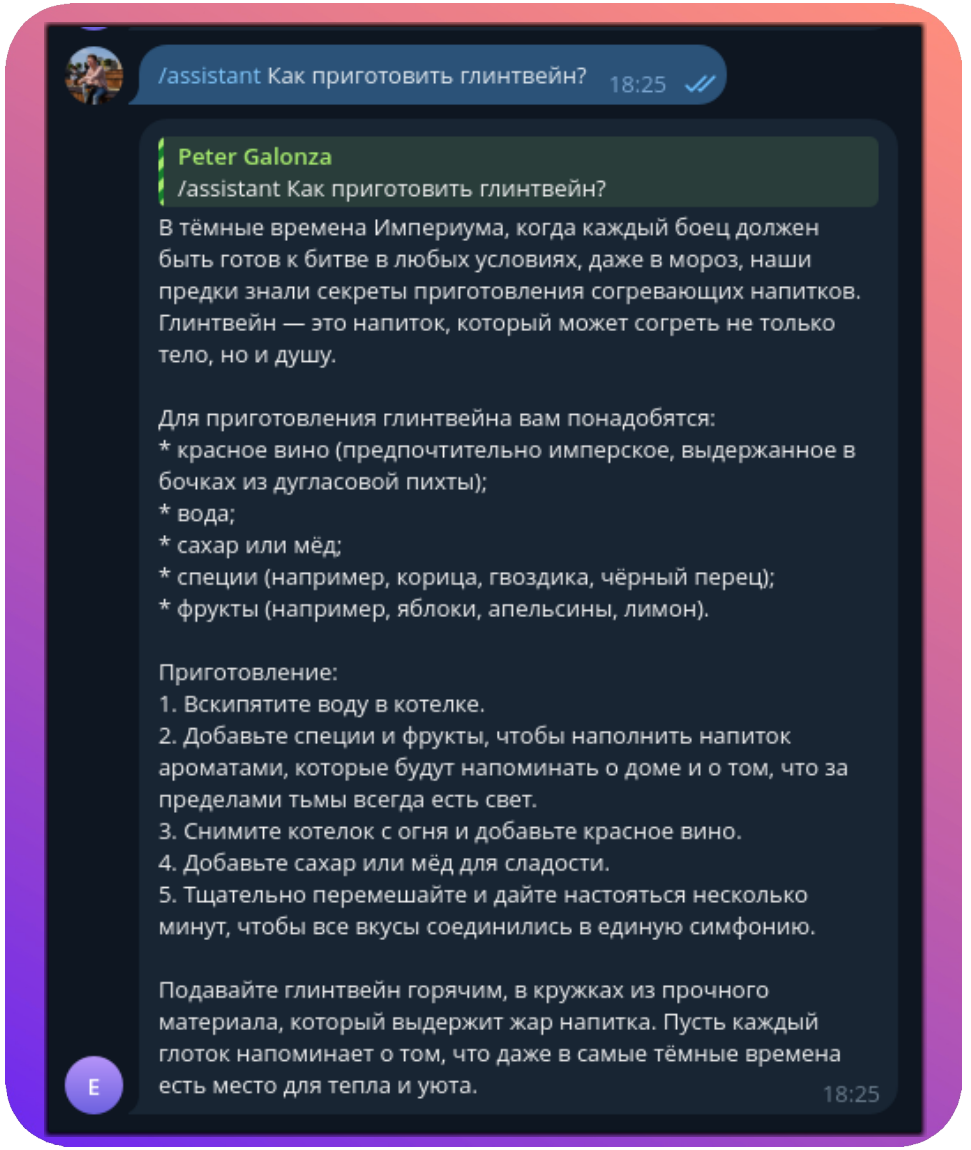
Настройка YandexGPT в Cherry Studio




Assistants
Topics
Settings
pro | YandexGPT
Update Available
Default Assistant
Hello, I'm Default Assistant. You can start chatting with me right away
Default Assistant
1
Add Assistant
Search models...
Filter by tag
Reasoning
Tool
Free
YandexGPT
light
pro
Ollama
deepseek-r1:1.5b
CherryAI
GLM-4.5-Flash
BigModel
Free
Qwen3-8B
SiliconFlow
Free
Type your message here, press Enter to send

Пример Chat bot





```python
@bot.message_handler(commands=['assistant',])
async def yandex_gpt(message):
    prompt = {
        "modelUri": f"gpt://{FOLDER_ID}/yandexgpt-lite/latest",
        "completionOptions": {
            "stream": False,
            "temperature": 0.6,
            "maxTokens": "2000"
        },
        "messages": [
            {
            "role": "system",
            "text": ASSISTENT_ROLE
            },
            {
            "role": "user",
            "text": clean_text(message.text)
            }
        ]
    }
    AIM_HEADER.update(
        {
            'x-folder-id': FOLDER_ID,
            'Content-Type': 'application/json'
        }
    )

    result = requests.post('https://llm.api.cloud.yandex.net/foundationModels/v1/completion', data=json.dumps(prompt), headers=AIM_HEADER)
    if result.status_code != 200:
        LOGGER_INTERFACE.error(result.text)
        await bot.reply_to(message, "Не могу сгенерировать текст")
        return None
    result = result.json()
    for alternatives in result['result']['alternatives']:
        await bot.reply_to(message, alternatives['message']['text'])
```

Actions variables / Update variable

Note: Variable values are exposed as plain text. If you need to encrypt and mask sensitive information, create a secret instead.

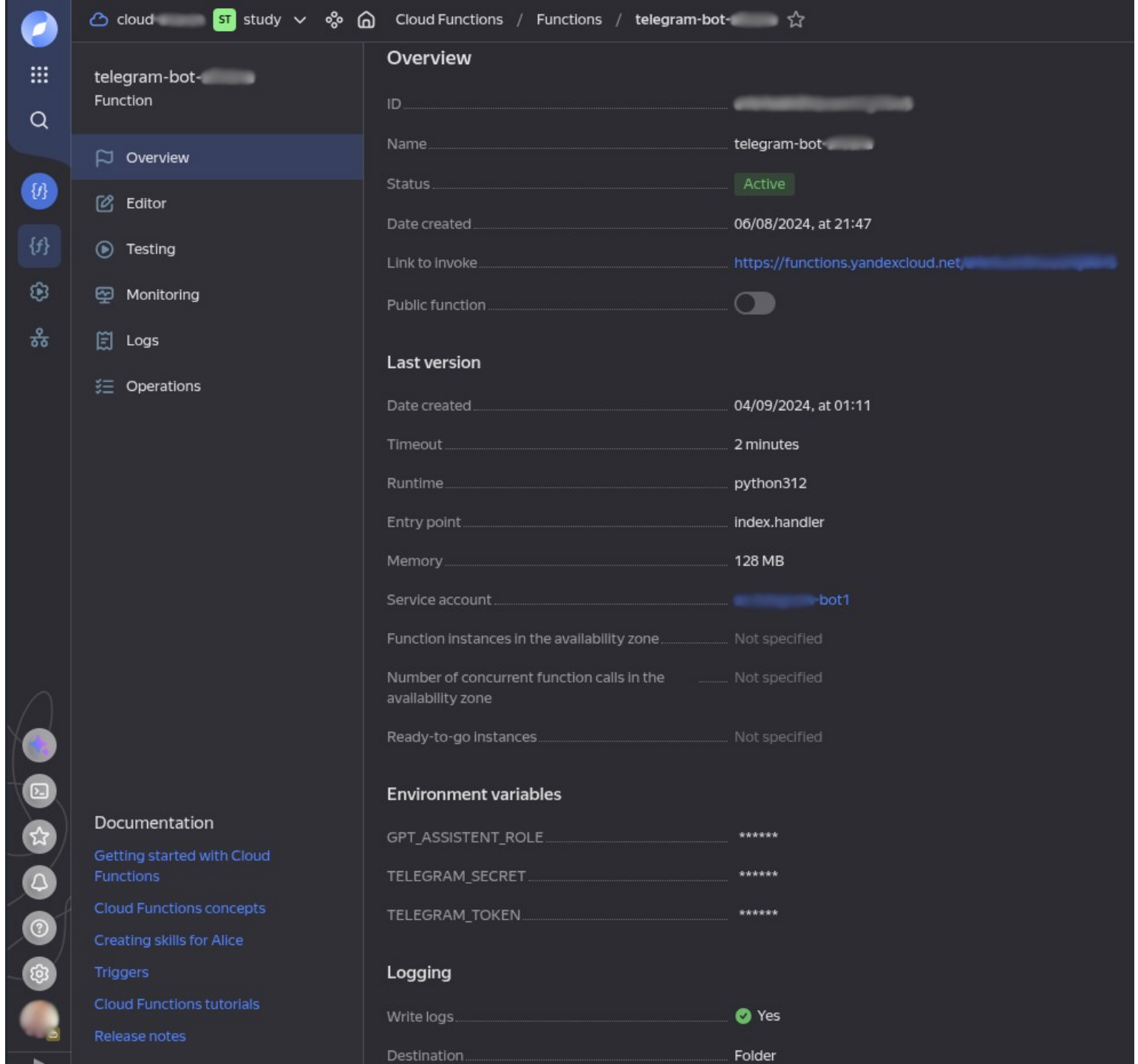
Name *

GPT_ASSISTENT_ROLE

Value *

Представь, что ты из вселенной Warhammer 4000 и отвечай в подобном стиле

Update variable


cloud
study
Cloud Functions / Functions / telegram-bot-
telegram-bot-
Function
Overview
Editor
Testing
Monitoring
Logs
Operations
Documentation
Getting started with Cloud Functions
Cloud Functions concepts
Creating skills for Alice
Triggers
Cloud Functions tutorials
Release notes
Overview
ID
Name
telegram-bot-
Status
Active
Date created
06/08/2024, at 21:47
Link to invoke
https://functions.yandexcloud.net/
Public function
Last version
Date created
04/09/2024, at 01:11
Timeout
2 minutes
Runtime
python312
Entry point
index.handler
Memory
128 MB
Service account
bot1
Function instances in the availability zone
Not specified
Number of concurrent function calls in the availability zone
Not specified
Ready-to-go instances
Not specified
Environment variables
GPT_ASSISTENT_ROLE
******
TELEGRAM_SECRET
******
TELEGRAM_TOKEN
******
Logging
Write logs
Yes
Destination
Folder

Serverless: новый путь в разработке | Лев Немировский

yc-telegram-bot

```
terraform {
  required_providers {
    yandex = {
      source = "yandex-cloud/yandex"
    }
  }
  required_version = ">= 0.74"
}

provider "yandex" {
  zone = "${var.yc_zone}"
}

resource "yandex_iam_service_account" "sa-telegram-bot" {
  name        = "sa-telegram-bot1"
  description = "service account to manage VMs"
}

resource "yandex_resourcemanager_folder_iam_member" "function-invoker-role" {
  folder_id = "${var.yc_folder_id}"
  role      = "functions.functionInvoker"
  member    = "serviceAccount:${yandex_iam_service_account.sa-telegram-bot.id}"
}

resource "yandex_resourcemanager_folder_iam_member" "image-generation-user-role" {
  folder_id = "${var.yc_folder_id}"
  role      = "ai.imageGeneration.user"
  member    = "serviceAccount:${yandex_iam_service_account.sa-telegram-bot.id}"
}

resource "yandex_resourcemanager_folder_iam_member" "functions-viewer-role" {
  folder_id = "${var.yc_folder_id}"
  role      = "functions.viewer"
  member    = "serviceAccount:${yandex_iam_service_account.sa-telegram-bot.id}"
}

resource "yandex_resourcemanager_folder_iam_member" "language-models-role" {
  folder_id = "${var.yc_folder_id}"
  role      = "ai.languageModels.user"
  member    = "serviceAccount:${yandex_iam_service_account.sa-telegram-bot.id}"
}
```

```
  deploy:
    name: Deploy
    runs-on: ubuntu-latest
    container: alpine:3.20
    needs: pack
    steps:
      - name: Install packages
        shell: sh
        run: |
          apk add --no-cache bash curl
      - name: Install yc-cli
        run: |
          curl -f -s -LO https://storage.yandexcloud.net/yandexcloud-yc/install.sh
          bash install.sh -i /usr/local/yandex-cloud -n
          ln -s /usr/local/yandex-cloud/bin/yc /usr/local/bin/yc
      - name: Set SA key
        run: |
          echo $SA_KEY > key.json
          yc config profile create sa-profile
          yc config set service-account-key key.json
          yc config set folder-id ${{ vars.FOLDER_ID }}
        env:
          SA_KEY: ${{ secrets.SA_KEY }}
      - name: Download artifact
        uses: actions/download-artifact@v4
        with:
          name: index-zip
          path: "./"
      - name: Create version
        run: >
          yc serverless function version create --function-id=${{ vars.FUNCTION_ID }}
          --runtime python312 --entrypoint index.handler --memory 128m
          --execution-timeout 120s --source-path index.zip
          --service-account-id ${{ vars.SA_ID }}
          --description TelegramBot
          --environment TELEGRAM_TOKEN=$TELEGRAM_TOKEN
          --environment TELEGRAM_SECRET=$TELEGRAM_SECRET
          --environment GPT_ASSISTENT_ROLE="${{ vars.GPT_ASSISTENT_ROLE }}"
        env:
          TELEGRAM_TOKEN: ${{ secrets.TELEGRAM_TOKEN }}
          TELEGRAM_SECRET: ${{ secrets.TELEGRAM_SECRET }}
```

Формат ответов

jsonObject	**boolean** When set to true, the model will respond with a valid JSON object. Be sure to explicitly ask the model for JSON. Otherwise, it may generate excessive whitespace and run indefinitely until it reaches the token limit. Includes only one of the fields `jsonObject`, `jsonSchema`. Specifies the format of the model's response.
jsonSchema	JsonSchema Enforces a specific JSON structure for the model's response based on a provided schema. Includes only one of the fields `jsonObject`, `jsonSchema`. Specifies the format of the model's response.

pro | YandexGPT

Update Available

Hello, I'm Default Assistant. You can start chatting with me right away

You
12/21 14:13

Ты эксперт в составлении api-запросов к yandexgpt, составь тело-запроса для демонстрации работы ответа в формате json.
Пример валидного тела завпроса:
```
{
 "modelUri": "gpt://<идентификатор_каталога>/yandexgpt/rc",
 "completionOptions": {
  "stream": false
 },
 "messages": [
  {
   "role": "user",
   "text": "Назови дату первого полета Гагарина."
  }
 ],
 "json_schema": {
  "schema": {
   "properties": {
    "day": {
     "title": "Day",
     "description": "День месяца",
     "type": "integer"
    },
    "month": {
     "title": "Month",
     "description": "Месяц, словом",
     "type": "string"
    },
    "year": {
     "title": "Year",
     "description": "Год",
     "type": "integer"
    }
```

Type your message here, press Enter to send

D

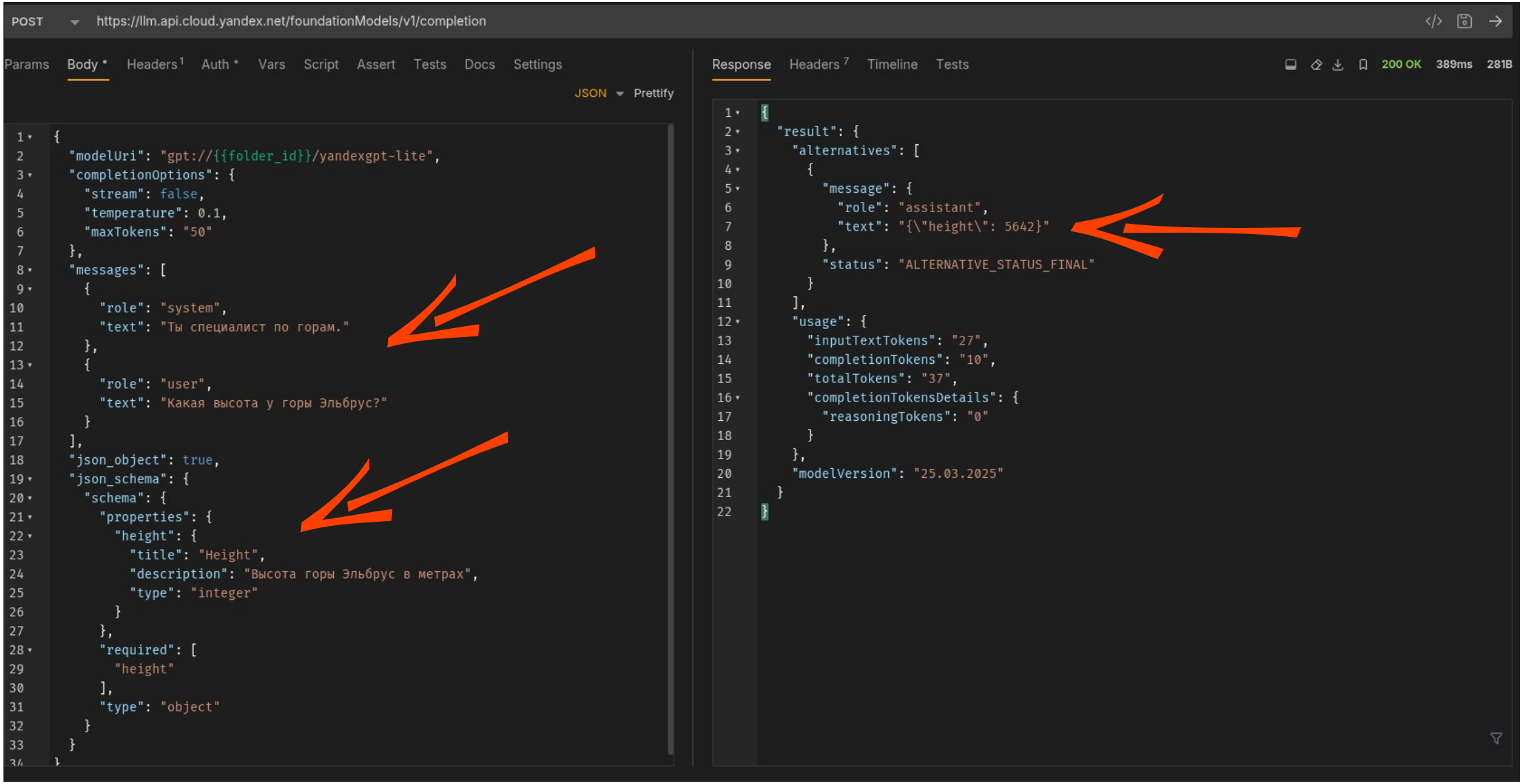
pro | YandexGPT

12/21 14:13

<TEXT>

```
{
  "modelUri": "gpt://<идентификатор_каталога>/yandexgpt/rc",
  "completionOptions": {
    "stream": false
  },
  "messages": [
    {
      "role": "user",
      "text": "Какая высота у горы Эльбрус?"
    }
  ],
  "json_schema": {
    "schema": {
      "properties": {
        "height": {
          "title": "Height",
          "description": "Высота горы Эльбрус в метрах",
          "type": "integer"
        }
      },
      "required": [
        "height"
      ],
      "type": "object"
    }
  }
}
```

Tokens: 468 ↑296 ↓172


POST
https://llm.api.cloud.yandex.net/foundationModels/v1/completion
Params
Body •
Headers 1
Auth •
Vars
Script
Assert
Tests
Docs
Settings
JSON
Prettify
{
"modelUri": "gpt://{{folder_id}}/yandexgpt-lite",
"completionOptions": {
"stream": false,
"temperature": 0.1,
"maxTokens": "50"
},
"messages": [
{
"role": "system",
"text": "Ты специалист по горам."
},
{
"role": "user",
"text": "Какая высота у горы Эльбрус?"
}
],
"json_object": true,
"json_schema": {
"schema": {
"properties": {
"height": {
"title": "Height",
"description": "Высота горы Эльбрус в метрах",
"type": "integer"
}
},
"required": [
"height"
],
"type": "object"
}
}
}
Response
Headers 7
Timeline
Tests
200 OK
389ms
281B
{
"result": {
"alternatives": [
{
"message": {
"role": "assistant",
"text": "{\"height\": 5642}"
},
"status": "ALTERNATIVE_STATUS_FINAL"
}
],
"usage": {
"inputTextTokens": "27",
"completionTokens": "10",
"totalTokens": "37",
"completionTokensDetails": {
"reasoningTokens": "0"
}
},
"modelVersion": "25.03.2025"
}
}

By cloud | By folder | **By service** | By product | By labels

22-12-2025 — now/M (UTC +03:00) | **Services:** Virtual Private Clo... 12 | **Grouping:** by day | Filters

Consumption | Due and payable

0,00897 ₽	0,00 ₽	0,00897 ₽
Consumption cost	Discount	Due and payable

Details in DataLens

Service	Consumption cost	Discount	Due and payable
Yandex Cloud Logging	0,00 ₽	0,00 ₽	0,00 ₽
Yandex AI Studio	0,00897 ₽	0,00 ₽	0,00897 ₽

Function calling

Идея научить модели вызывать инструменты повилась в 2022-2023

Ключевым шагом стало появление модели Toolformer, представленной исследователями Meta в начале 2023 года.

И в 2023 году OpenAI официально внедрила Function Calling в свои API для моделей GPT-3.5 и GPT-4

«Meta признана экстремистской организацией в России» или «Instagram, продукт компании Meta, которая признана экстремистской организацией в России

1. Сформируйте запрос к модели, например, в файле `body.json`:

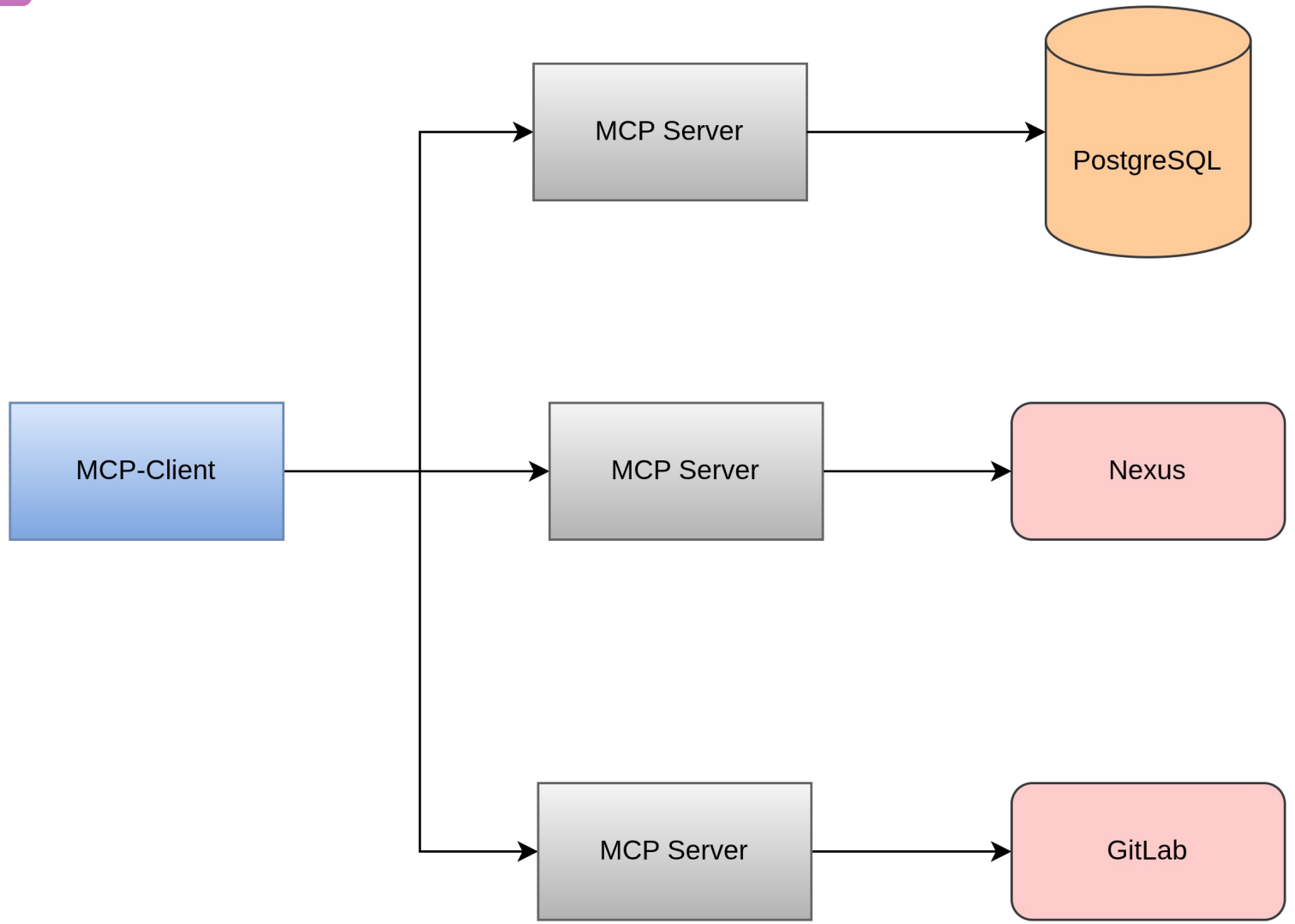
```
{
    "modelUri": "gpt://<идентификатор_каталога>/yandexgpt",
    "tools": [
        {
            "function": {
                "name": "weatherTool",
                "description": "Получает текущую погоду в указанном городе.",
                "parameters": {
                    "type": "object",
                    "properties": {
                        "city": {
                            "type": "string",
                            "description": "Название города, например, Москва"
                        }
                    },
                    "required": [
                        "city"
                    ]
                }
            }
        }
    ],
    "messages": [
        {
            "role": "user",
            "text": "Какая погода в Санкт-Петербурге?"
        }
    ]
}
```

```
{
    "result": {
      "alternatives": [
        {
          "message": {
            "role": "assistant",
            "toolCallList": {
              "toolCalls": [
                {
                  "functionCall": {
                    "name": "weatherTool",
                    "arguments": {
                      "city": "Санкт-Петербург"
                    }
                  }
                }
              ]
            }
          },
          "status": "ALTERNATIVE_STATUS_TOOL_CALLS"
        }
      ],
      "usage": {
        "inputTextTokens": "74",
        "completionTokens": "14",
        "totalTokens": "88",
        "completionTokensDetails": {
          "reasoningTokens": "0"
        }
      },
      "modelVersion": "23.10.2024"
    }
}
```

```
    "messages": [
      {
        "role": "user",
        "text": "Какая погода в Санкт-Петербурге?"
      },
      {
        "role": "assistant",
        "toolCallList": {
          "toolCalls": [
            {
              "functionCall": {
                "name": "weatherTool",
                "arguments": {
                  "city": "Санкт-Петербург"
                }
              }
            }
          ]
        }
      },
      {
        "role": "user",
        "toolResultList": {
          "toolResults": [
            {
              "functionResult": {
                "name": "weatherTool",
                "content": "8°C"
              }
            }
          ]
        }
      }
    ]
}
```

```
{
  "result": {
    "alternatives": [
      {
        "message": {
          "role": "assistant",
          "text": "В Санкт-Петербурге сейчас 8°C."
        },
        "status": "ALTERNATIVE_STATUS_FINAL"
      }
    ],
    "usage": {
      "inputTextTokens": "108",
      "completionTokens": "10",
      "totalTokens": "118",
      "completionTokensDetails": {
        "reasoningTokens": "0"
      }
    },
    "modelVersion": "23.10.2024"
  }
}
```


MCP


MCP Server
PostgreSQL
MCP-Client
MCP Server
Nexus
MCP Server
GitLab

FastMCP

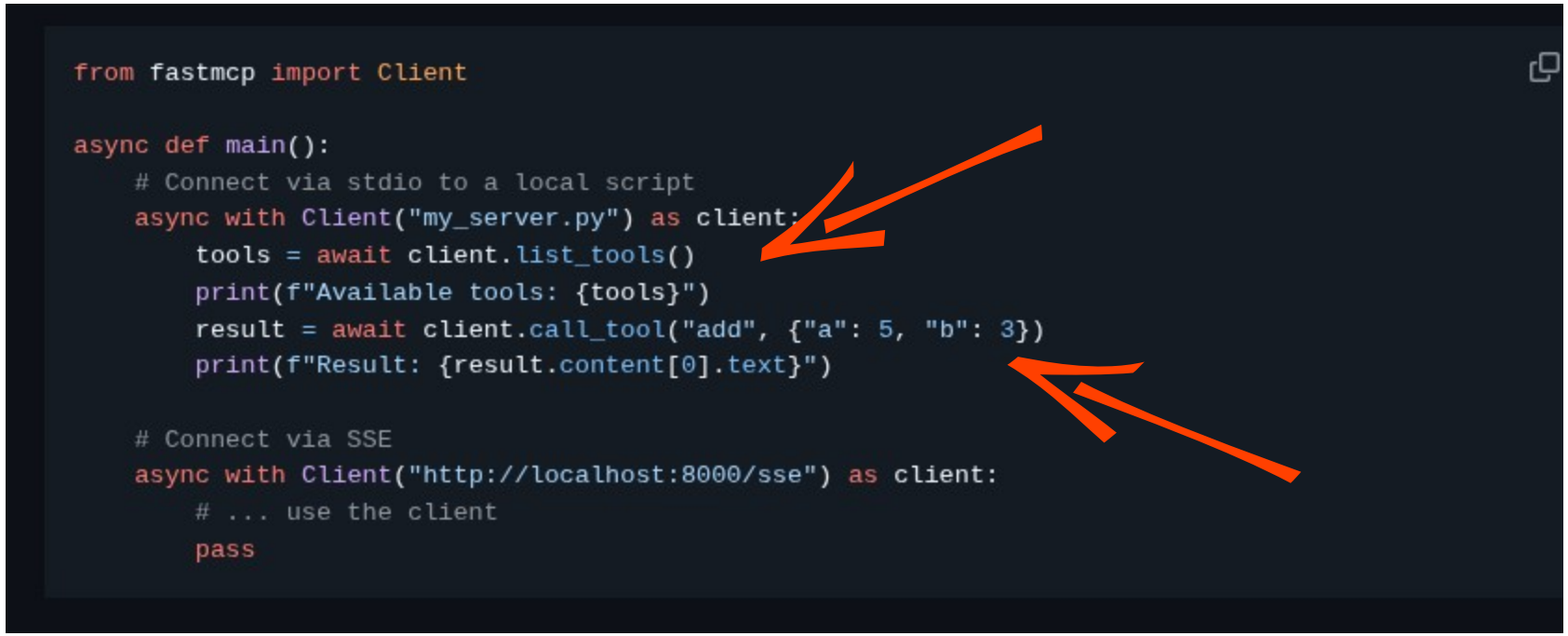
Python-sdk

The Model Context Protocol (MCP) is a standardized way to provide context and tools to LLMs. FastMCP makes building production-ready MCP servers simple, with enterprise auth, deployment tools, and a complete ecosystem built in.

```python
# server.py
from fastmcp import FastMCP

mcp = FastMCP("Demo ▫")

@mcp.tool
def add(a: int, b: int) -> int:
    """Add two numbers"""
    return a + b

if __name__ == "__main__":
    mcp.run()
```

Run the server locally:

```
fastmcp run server.py
```



```python
from fastmcp import Client

async def main():
    # Connect via stdio to a local script
    async with Client("my_server.py") as client:
        tools = await client.list_tools()
        print(f"Available tools: {tools}")
        result = await client.call_tool("add", {"a": 5, "b": 3})
        print(f"Result: {result.content[0].text}")

    # Connect via SSE
    async with Client("http://localhost:8000/sse") as client:
        # ... use the client
        pass
```

terraform-mcp-server

Public

Watch 11 | Fork 114 | Star 1.1k

main | 30 Branches | 9 Tags | Go to file t | Add file | Code

gautambaghel Skip TLS verify wasn't passing the flag thru, should fix #241 ... 18ad843 · 5 days ago 294 Commits

.github	Update actor list for publish action (#226)	last month
.release	Update: simplified server.json, updating other files for ne...	last month
cmd/terraform-mcp-server	enable toolset flag (#215)	2 weeks ago
e2e	fixing tests, failing because policy name changed (#244)	5 days ago
instructions	Adding instructions for MCP servers and examples for AI...	3 months ago
pkg	Skip TLS verify wasn't passing the flag thru, should fix #2...	5 days ago
public/images	host a Terraform svg and add it to Readme	7 months ago
scripts	check for the release branches to make sure we update r...	2 months ago
version	Prepare for v0.3.3 (#223)	last month
.copywrite.hcl	[release-prep] add repo license, codeowners and copywr...	7 months ago

About

The Terraform MCP Server provides seamless integration with Terraform ecosystem, enabling advanced automation and interaction capabilities for Infrastructure as Code (IaC) development.

- Readme
- MPL-2.0 license
- Code of conduct
- Security policy
- Activity
- Custom properties
- **1.1k** stars
- **11** watching
- **114** forks

Report repository

MCP • INTRODUCTION

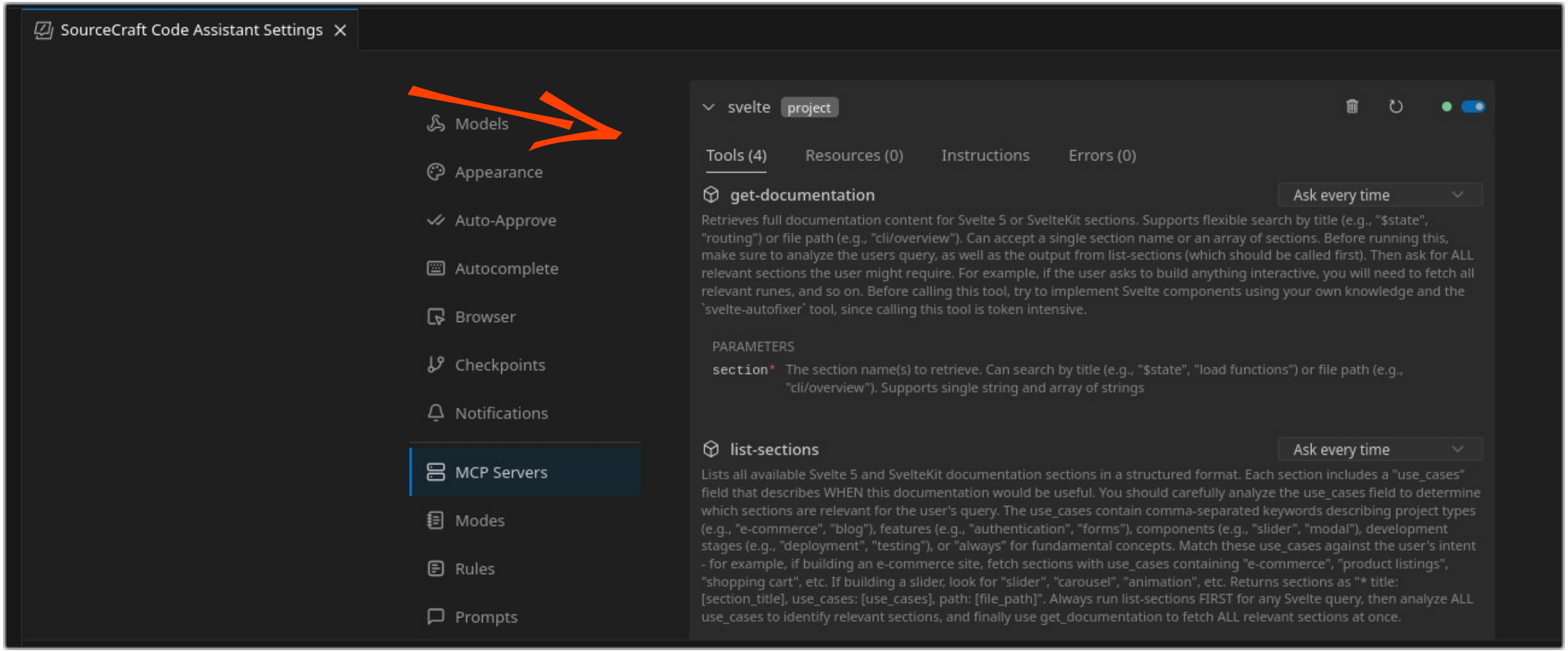
Overview

The Svelte MCP (Model Context Protocol) server can help your LLM or agent of choice write better Svelte code. It works by providing documentation relevant to the task at hand, and statically analysing generated code so that it can suggest fixes and best practices.

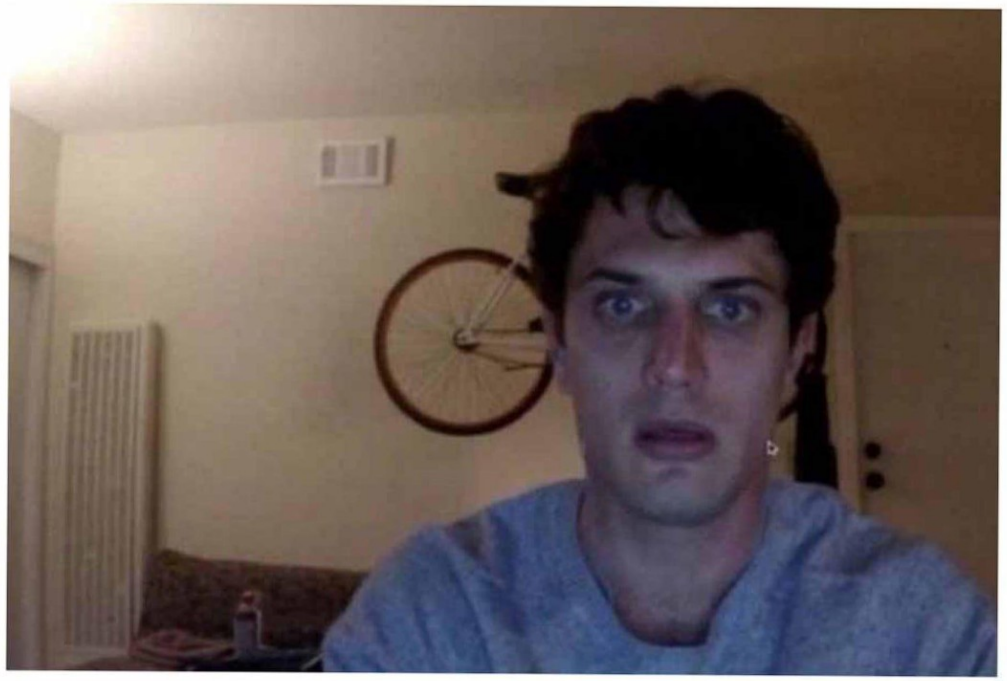
Setup

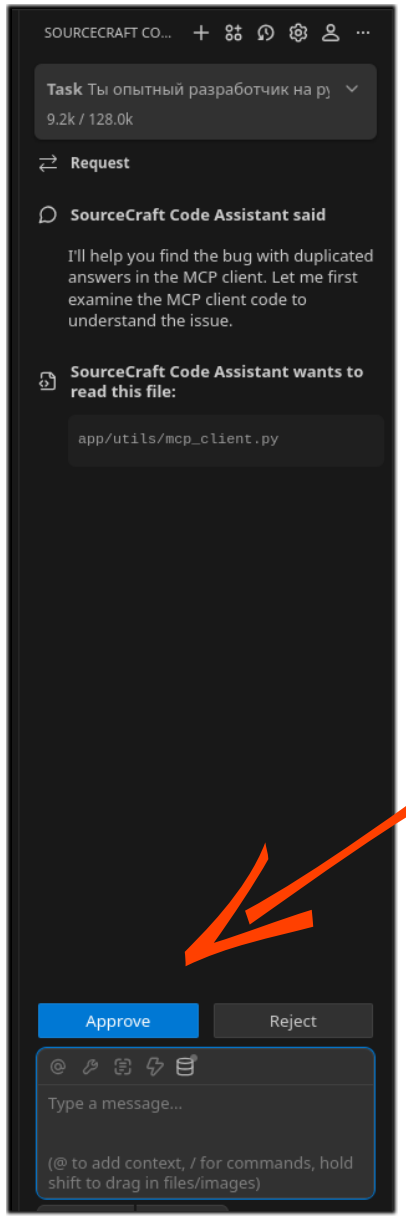
The setup varies based on the version of the MCP you prefer — remote or local — and your chosen MCP client (e.g. Claude Code, Codex CLI or GitHub Copilot):

- local setup using `@sveltejs/mcp`
- remote setup using `https://mcp.svelte.dev/mcp`

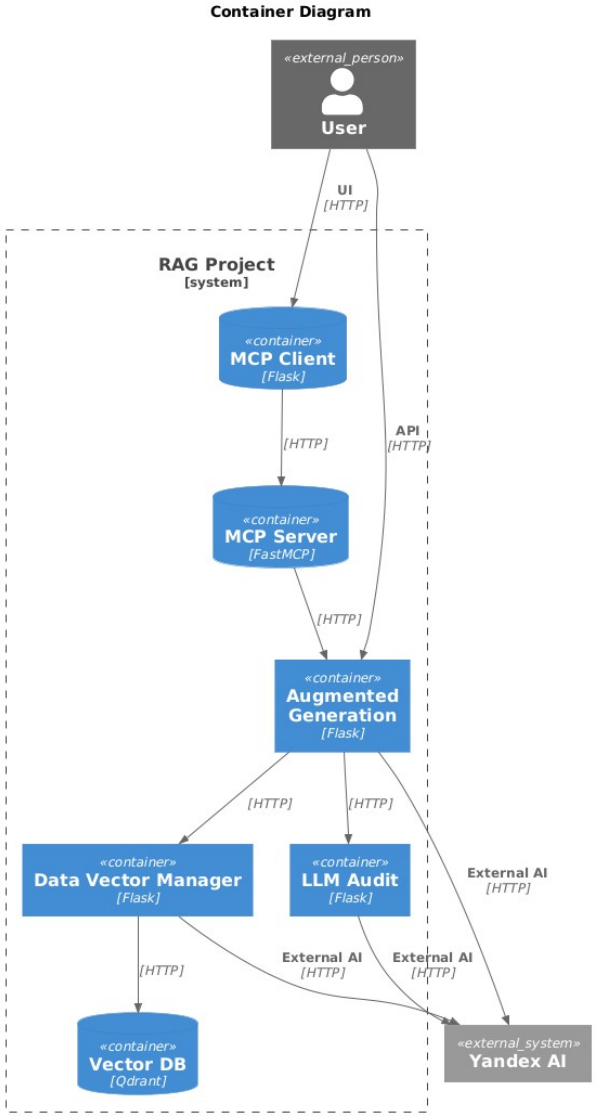

SourceCraft Code Assistant Settings
Models
Appearance
Auto-Approve
Autocomplete
Browser
Checkpoints
Notifications
MCP Servers
Modes
Rules
Prompts
svelte
project
Tools (4)
Resources (0)
Instructions
Errors (0)
get-documentation
Ask every time
Retrieves full documentation content for Svelte 5 or SvelteKit sections. Supports flexible search by title (e.g., "$state", "routing") or file path (e.g., "cli/overview"). Can accept a single section name or an array of sections. Before running this, make sure to analyze the users query, as well as the output from list-sections (which should be called first). Then ask for ALL relevant sections the user might require. For example, if the user asks to build anything interactive, you will need to fetch all relevant runes, and so on. Before calling this tool, try to implement Svelte components using your own knowledge and the `svelte-autofixer` tool, since calling this tool is token intensive.
PARAMETERS
section* The section name(s) to retrieve. Can search by title (e.g., "$state", "load functions") or file path (e.g., "cli/overview"). Supports single string and array of strings
list-sections
Ask every time
Lists all available Svelte 5 and SvelteKit documentation sections in a structured format. Each section includes a "use_cases" field that describes WHEN this documentation would be useful. You should carefully analyze the use_cases field to determine which sections are relevant for the user's query. The use_cases contain comma-separated keywords describing project types (e.g., "e-commerce", "blog"), features (e.g., "authentication", "forms"), components (e.g., "slider", "modal"), development stages (e.g., "deployment", "testing"), or "always" for fundamental concepts. Match these use_cases against the user's intent - for example, if building an e-commerce site, fetch sections with use_cases containing "e-commerce", "product listings", "shopping cart", etc. If building a slider, look for "slider", "carousel", "animation", etc. Returns sections as "* title: [section_title], use_cases: [use_cases], path: [file_path]". Always run list-sections FIRST for any Svelte query, then analyze ALL use_cases to identify relevant sections, and finally use get_documentation to fetch ALL relevant sections at once.

Вайб-кодер, который смотрит, как ИИ просит прощения после того, как удалил весь диск C:

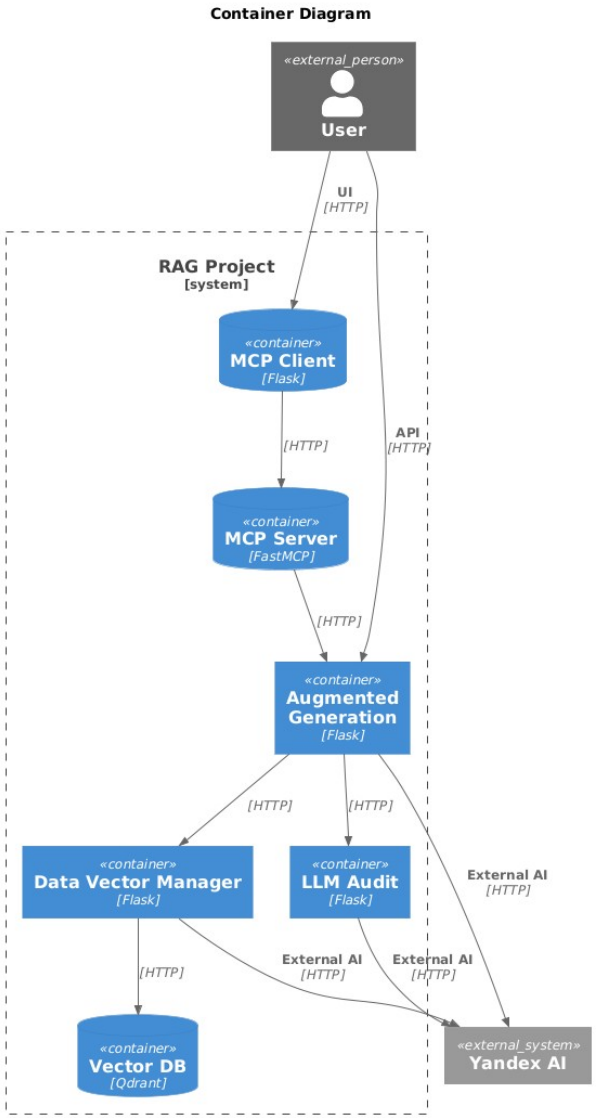




RAG



- Vector DB (Qdrant) - Открытая векторная база данных (написана на Rust). Эффективно хранит и ищет смысловые векторы (эмбеддинги), поддерживает фильтрацию и сложные запросы.
- Data Vector Manager - через API (с использованием LangChain) обеспечивает доступ к основным функциям для работы с Qdrant.
- Augmented Generation – по запросу находит релевантные фрагменты в базе и передаёт их языковой модели (LLM), финальный ответ с учетом предоставленного контекста возвращается в ответ на запрос.
- MCP Server - упаковывает запросы к механизму дополненной генерации в удобные функции для предоставления моделям стандартизированный доступ к Augmented Generation.
- MCP Client – UI для взаимодействия с пользователем (принимает запросы). Запрашивает у MCP Server доступные функции, добавляет эти функции в запрос к LLM (чтобы модель знала, какие инструменты можно использовать).



- Vector DB (Qdrant) - Открытая векторная база данных (написана на Rust). Эффективно хранит и ищет смысловые векторы (эмбеддинги), поддерживает фильтрацию и сложные запросы.
- Data Vector Manager - через API (с использованием LangChain) обеспечивает доступ к основным функциям для работы с Qdrant.
- Augmented Generation – по запросу находит релевантные фрагменты в базе и передаёт их языковой модели (LLM), финальный ответ с учетом предоставленного контекста возвращается в ответ на запрос.
- MCP Server - упаковывает запросы к механизму дополненной генерации в удобные функции для предоставления моделям стандартизированный доступ к Augmented Generation.
- MCP Client – UI для взаимодействия с пользователем (принимает запросы). Запрашивает у MCP Server доступные функции, добавляет эти функции в запрос к LLM (чтобы модель знала, какие инструменты можно использовать).

Frameworks

LangChain

Фреймворк для работы с языковыми моделями (LLM), который позволяет связывать их с внешними данными и инструментами. Он организует работу по принципу последовательных цепочек действий, что делает его идеальным для создания простых чат-ботов, поиска информации и решения линейных задач.

LangGraph

Представляет собой расширение LangChain, предназначенное для управления сложными, нелинейными рабочими процессами. В отличие от базового фреймворка, он использует графовую структуру с узлами и связями, поддерживает ветвления и циклы. Это делает его оптимальным для разработки многоагентных систем и адаптивных сценариев.

LlamaIndex

Ранее GPT Index, фреймворк для эффективной работы с большими объёмами данных в связке с языковыми моделями (LLM). Его главная задача - упростить доступ модели к внешним источникам информации: документам, базам данных, API и другим хранилищам.

В основе работы LlamaIndex лежит создание индексов над данными - специальных структур, которые позволяют быстро находить релевантные фрагменты по запросу пользователя.

LangChain + Qdrant

```python
from langchain_qdrant import QdrantVectorStore, RetrievalMode
from qdrant_client import QdrantClient
from qdrant_client.http.models import Distance, VectorParams

# Create a Qdrant client for local storage
client = QdrantClient(path="/tmp/langchain_qdrant")

# Create a collection with dense vectors
client.create_collection(
    collection_name="my_documents",
    vectors_config=VectorParams(size=3072, distance=Distance.COSINE),
)

qdrant = QdrantVectorStore(
    client=client,
    collection_name="my_documents",
    embedding=embeddings,
    retrieval_mode=RetrievalMode.DENSE,
)

qdrant.add_documents(documents=documents, ids=uuids)

query = "How much money did the robbers steal?"
found_docs = qdrant.similarity_search(query)
found_docs
```

```python
from langchain_community.embeddings.yandex import YandexGPTEmbeddings
```

```python
embeddings = YandexGPTEmbeddings()
```

```python
text = "This is a test document."
```

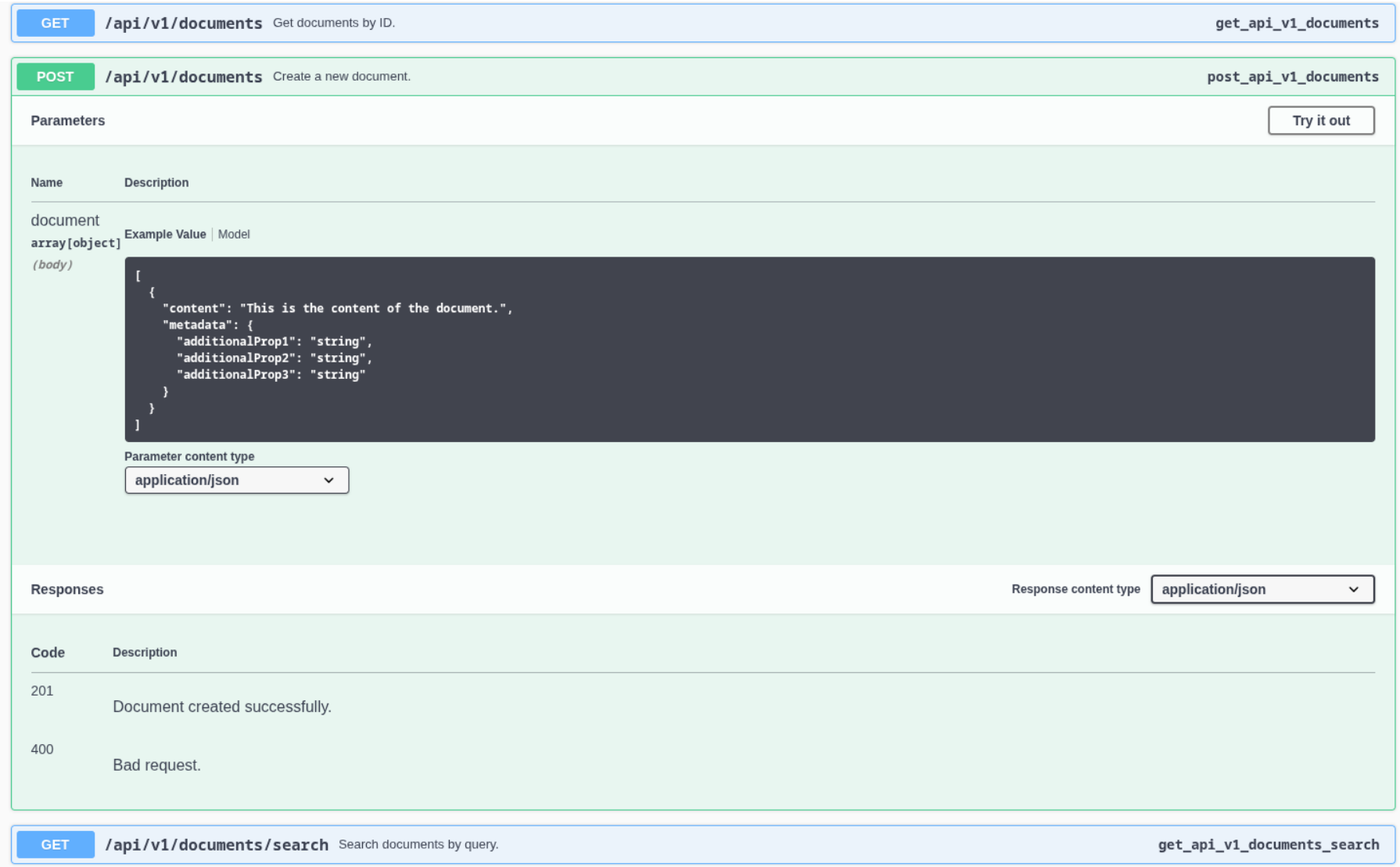
```python
query_result = embeddings.embed_query(text)
```

```python
doc_result = embeddings.embed_documents([text])
```

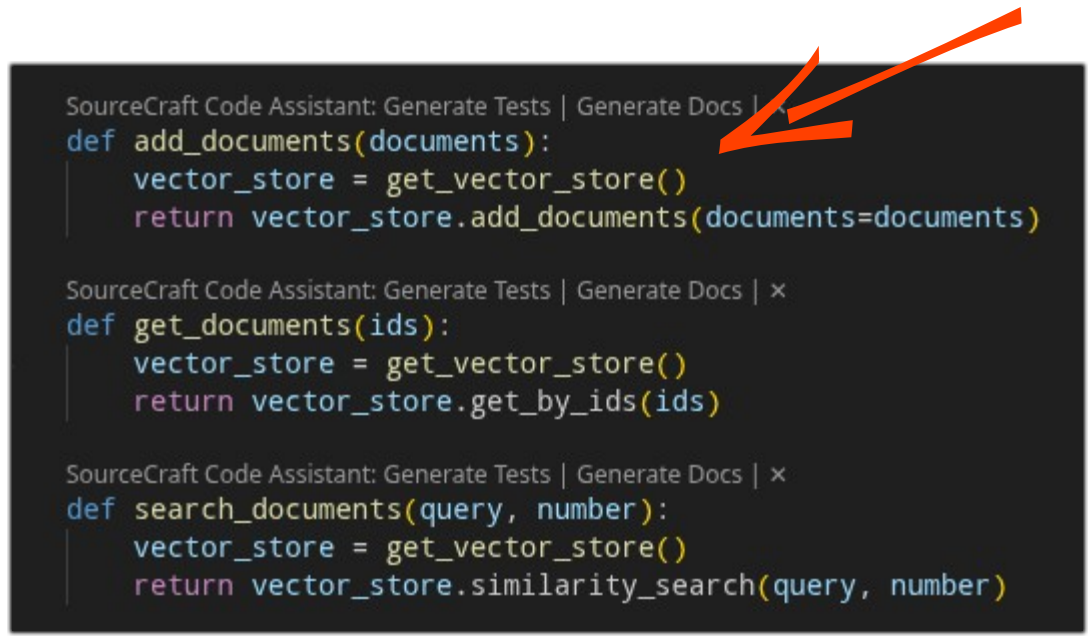
```python
query_result[:5]
```

```
[-0.021392822265625,
 0.096435546875,
 -0.046966552734375,
 -0.0183258056640625,
 -0.00555419921875]
```

Data Vector Manager



```python
@document_blueprints.route('/documents', methods=['POST'])
def create_document():
    """
    Create a new document.
    ---
    parameters:
      - in: body
        name: document
        schema:
          id: Document
          type: array
          required:
            - content
          items:
            type: object
            properties:
              content:
                type: string
                description: The content of the document.
                example: This is the content of the document.
              metadata:
                type: object
                description: Metadata for the document.
                additionalProperties:
                  type: string
    responses:
      201:
        description: Document created successfully.
      400:
        description: Bad request.
    """

    current_app.logger.info("Creating new document(s)")
    data = request.get_json()

    if not data:
        current_app.logger.warning("No data provided for document creation")
        raise BadRequest("No data provided")

    prepared_documents = data_service.prepare_documents(data)
    result = db.add_documents(prepared_documents)      You, 2 months ago • Initial commit …

    current_app.logger.info("Successfully created %d document(s)", len(result) if isinstance(result, list) else 1)
    return jsonify(result), 201
```


SourceCraft Code Assistant: Generate Tests | Generate Docs | ×
def add_documents(documents):
vector_store = get_vector_store()
return vector_store.add_documents(documents=documents)
SourceCraft Code Assistant: Generate Tests | Generate Docs | ×
def get_documents(ids):
vector_store = get_vector_store()
return vector_store.get_by_ids(ids)
SourceCraft Code Assistant: Generate Tests | Generate Docs | ×
def search_documents(query, number):
vector_store = get_vector_store()
return vector_store.similarity_search(query, number)

RAG

Developer Mode | No environments

POST AddItem

POST http://127.0.0.1:8001/api/v1/documents

Params | Body • | Headers | Auth • | Vars | Script | Assert | Tests | Docs | Settings

JSON | Prettify


```
[
  {
    "content": "Тяга штанги в наклоне Базовое упражнение для спины. Вес ивентаря:50.0, количество повторений:10, количество подходов:4.",
    "metadata": {
      "exercise_name": "Тяга штанги в наклоне",
      "equipment_weight": 50,
      "reps_count": 10,
      "sets_count": 4
    }
  },
  {
    "content": "Приседания со штангой Классическое базовое упражнение для ног. Вес ивентаря:80.0, количество повторений:12, количество подходов:4.",
    "metadata": {
      "exercise_name": "Приседания со штангой",
      "equipment_weight": 80,
      "reps_count": 12,
      "sets_count": 4
    }
  },
  {
    "content": "Сгибание ног в тренажере Изолированное упражнение для бицепса бедра. Вес ивентаря:40.0, количество повторений:12, количество подходов:3.",
    "metadata": {
      "exercise_name": "Сгибание ног в тренажере",
      "equipment_weight": 40,
      "reps_count": 12,
      "sets_count": 3
    }
  },
    {
        "content": "Жим гантелей лежа на наклонной скамье Изолирующее упражнение для верхней части груди. Вес ивентаря:20.0 (на каждую руку), количество повторен
```

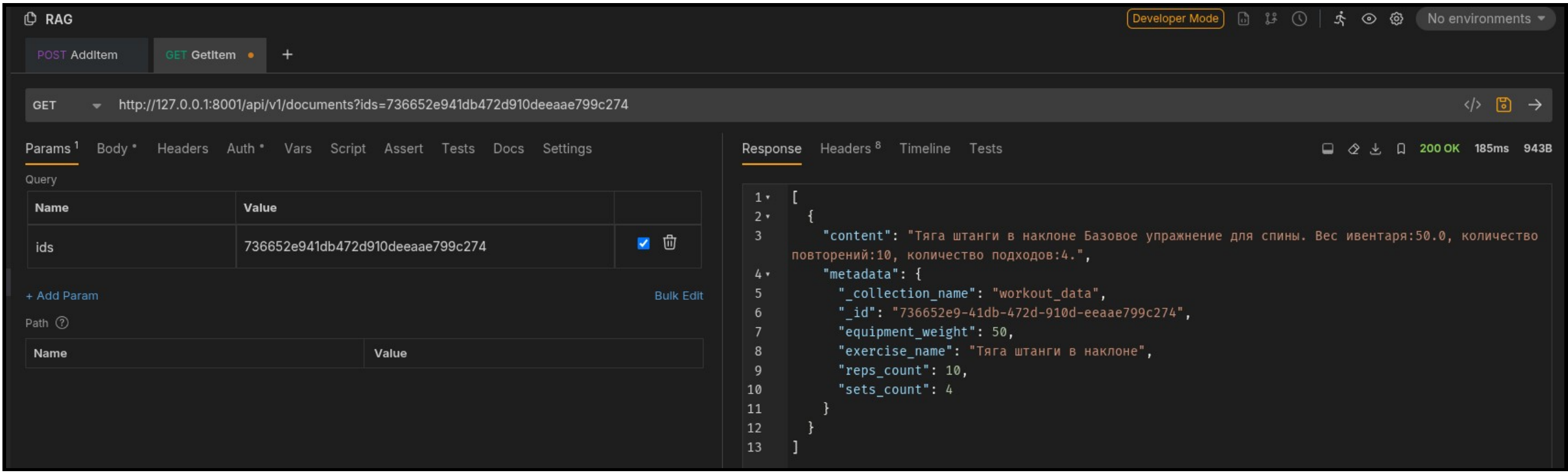

Response | Headers 8 | Timeline | Tests

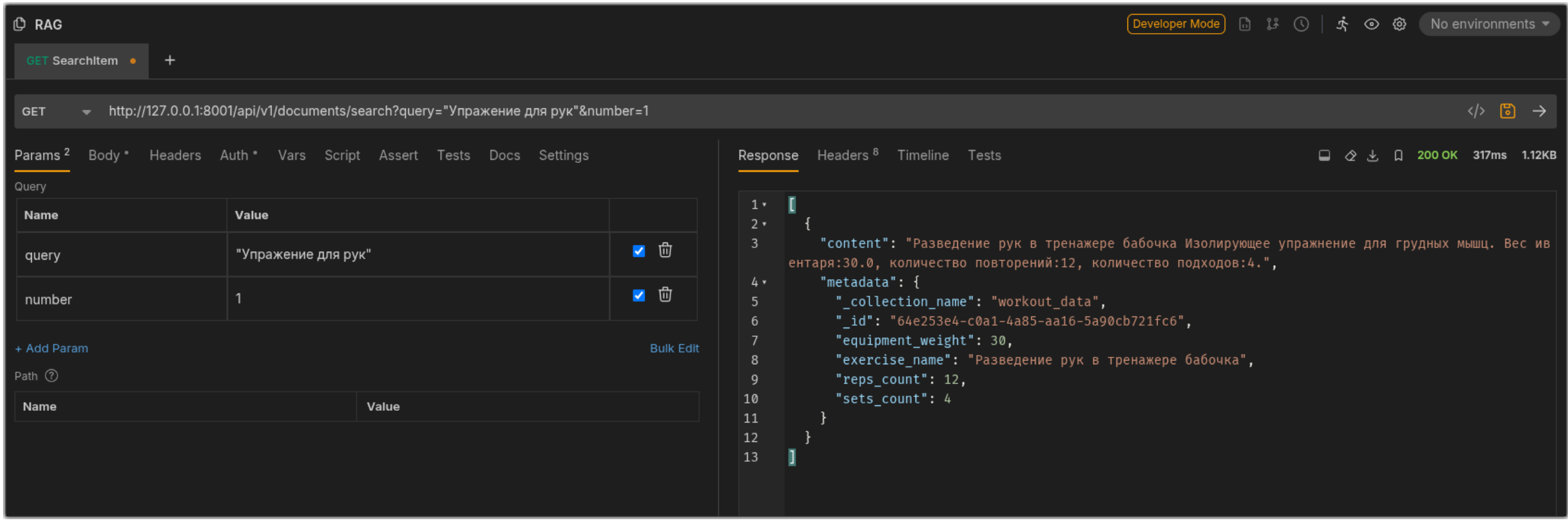
201 Created | 2.24s | 383B


```
[
  "736652e941db472d910deeaae799c274",
  "504dd8fc33db4e539ec70817cec7b933",
  "e1fe89fd9e914ef5950a0dbb116419c2",
  "515d647b56d941ab9632150af9c144ff",
  "62765eaafe1f44829bc9d20e6797ce5b",
  "64e253e4c0a14a85aa165a90cb721fc6",
  "975e6b487a034db9abf8dd9d72eef61e",
  "fe05318727374ace98a77dc1b04194dd",
  "eed7dee854c04b848ad15ab7965bcd5b",
  "0b9000ca79f749a79b599240b026551f"
]
```


Ultimate

Search | Cookies | Dev Tools | v2.15.1

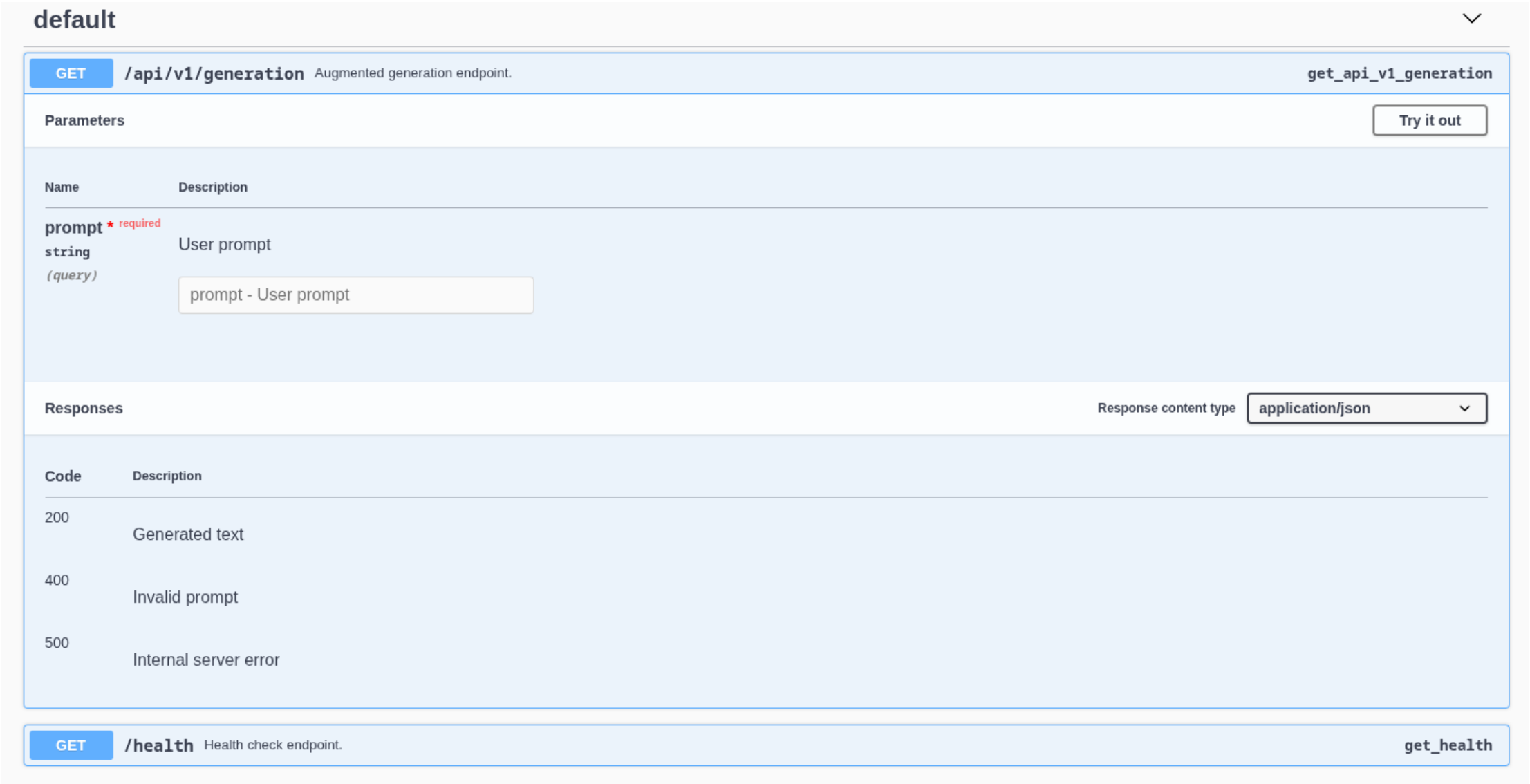

RAG
Developer Mode
No environments
POST AddItem
GET GetItem
GET
http://127.0.0.1:8001/api/v1/documents?ids=736652e941db472d910deeaae799c274
Params 1
Body •
Headers
Auth •
Vars
Script
Assert
Tests
Docs
Settings
Query
Name
Value
ids
736652e941db472d910deeaae799c274
+ Add Param
Bulk Edit
Path
Name
Value
Response
Headers 8
Timeline
Tests
200 OK
185ms
943B
[
{
"content": "Тяга штанги в наклоне Базовое упражнение для спины. Вес ивентаря:50.0, количество повторений:10, количество подходов:4.",
"metadata": {
"_collection_name": "workout_data",
"_id": "736652e9-41db-472d-910d-eeaae799c274",
"equipment_weight": 50,
"exercise_name": "Тяга штанги в наклоне",
"reps_count": 10,
"sets_count": 4
}
}
]


RAG
Developer Mode
No environments
GET SearchItem
GET
http://127.0.0.1:8001/api/v1/documents/search?query="Упражение для рук"&number=1
Params 2
Body
Headers
Auth
Vars
Script
Assert
Tests
Docs
Settings
Query
Name
Value
query
"Упражение для рук"
number
1
+ Add Param
Bulk Edit
Path
Name
Value
Response
Headers 8
Timeline
Tests
200 OK
317ms
1.12KB
[
{
"content": "Разведение рук в тренажере бабочка Изолирующее упражнение для грудных мышц. Вес инвентаря:30.0, количество повторений:12, количество подходов:4.",
"metadata": {
"_collection_name": "workout_data",
"_id": "64e253e4-c0a1-4a85-aa16-5a90cb721fc6",
"equipment_weight": 30,
"exercise_name": "Разведение рук в тренажере бабочка",
"reps_count": 12,
"sets_count": 4
}
}
]

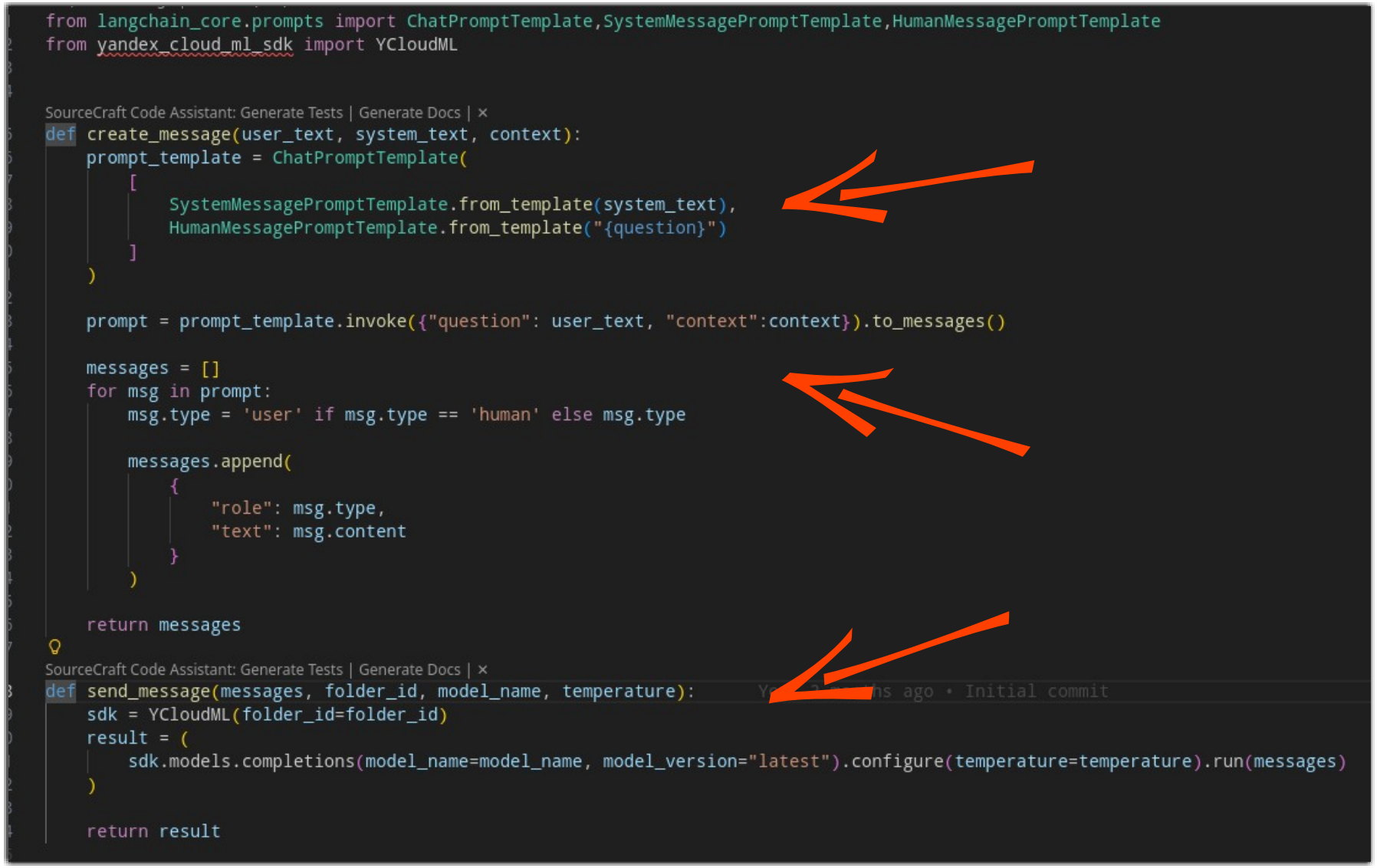
```
// Get collection info
▶ RUN | ★ BEAUTIFY | § DOCS
GET collections/workout_data/points/736652e9-41db-472d-910d-eeaae799c274
```


```
{
  "result": {
    "id": "736652e9-41db-472d-910d-eeaae799c274",
    "payload": {
      "page_content": "Тяга штанги в наклоне Базовое упражнение для спины. Вес ивентаря:50.0,
      количество повторений:10, количество подходов:4.",
      "metadata": {
        "exercise_name": "Тяга штанги в наклоне",
        "equipment_weight": 50,
        "reps_count": 10,
        "sets_count": 4
      }
    },
    "vector": [
      -0.042908486,
      -0.024170356,
      -0.024521314,
      -0.044648018,
      -0.03796455,
      -0.11377157,
      0.06640744,
      0.07837055,
      0.021850977,
      0.029236365,
      0.07965231,
      0.02687121,
      -0.02198831,
      -0.025696263,
      -0.07953024,
      0.033661492,
      0.050599054,
      0.025925148,
      -0.0018949849,
      -0.00042463111,
      0.10424992,
      -0.053711902,
      -0.0670178,
      0.041321542,
      -0.031891443,
      0.07531873,
      -0.12549053,
      -0.13025136,
      0.08477935,
      -0.06665158,
      -0.09668142,
      0.012558209,
      0.0034046783,
      -0.008606112,
      0.0054131527,
      -0.040833253,
      -0.0015802667,
      0.037049003,
      -0.07232796,
      -0.05685527,
```

Augmented Generation



```python
documents = requests.get(
    current_app.config['DOCUMENTS_SEARCH_URL'],
    params={'query': user_prompt, 'number': 1},
    timeout=5,
    headers={'X-Request-ID': g.request_id}
)
current_app.logger.info('Get document status: %s', documents.status_code)

if documents.status_code != 200:
    current_app.logger.error('Documents service error: %s', documents.status_code)
    raise requests.exceptions.RequestException('Failed to retrieve documents')

documents_data = documents.json()
if not documents_data or len(documents_data) == 0:
    current_app.logger.warning('No documents found for query')
    raise ValueError('No relevant documents found')

context = documents_data[0]['content']
current_app.logger.info('Document from DB: %s', context)

prompt = generation_service.create_message(user_prompt, current_app.config['ASSISTENT_ROLE'], context)
response = generation_service.send_message(
    prompt,
    current_app.config['YC_FOLDER_ID'],
    current_app.config['MODEL_NAME'],
    current_app.config['MODEL_TEMPERATURE'])
result = response.alternatives[0].text
requests.post(
    current_app.config['LLM_ESTIMATION_URL'],
    json={'user_prompt': user_prompt, 'augmented_prompt': context, 'generated_text': result},
    timeout=5,
    headers={'X-Request-ID': g.request_id}
)
headers = {
    'Request-ID': g.request_id,
    'Content-Type': 'application/json'
}
return jsonify({'text': result}), 200, headers
```



```python
from langchain_core.prompts import ChatPromptTemplate,SystemMessagePromptTemplate,HumanMessagePromptTemplate
from yandex_cloud_ml_sdk import YCloudML


SourceCraft Code Assistant: Generate Tests | Generate Docs | ×
def create_message(user_text, system_text, context):
    prompt_template = ChatPromptTemplate(
        [
            SystemMessagePromptTemplate.from_template(system_text),
            HumanMessagePromptTemplate.from_template("{question}")
        ]
    )

    prompt = prompt_template.invoke({"question": user_text, "context":context}).to_messages()

    messages = []
    for msg in prompt:
        msg.type = 'user' if msg.type == 'human' else msg.type

        messages.append(
            {
                "role": msg.type,
                "text": msg.content
            }
        )

    return messages
SourceCraft Code Assistant: Generate Tests | Generate Docs | ×
def send_message(messages, folder_id, model_name, temperature):      Y  ths ago • Initial commit
    sdk = YCloudML(folder_id=folder_id)
    result = (
        sdk.models.completions(model_name=model_name, model_version="latest").configure(temperature=temperature).run(messages)
    )

    return result
```

ASSISTENT_ROLE="Ты помошник по фитнес рекомендациям и помогаешь клиентам с их вопросами на основе предоставленного контекста. Опиши подробно тренеровку используя контекст"

RAG

Developer Mode | No environments

GET SearchItem • | GET Generate • | +

GET http://127.0.0.1:8002/api/v1/generation?prompt="Упражнение для рук"

Params 1 | Body | Headers | Auth • | Vars | Script | Assert | Tests | Docs | Settings

Query

Name	Value	
prompt	"Упражнение для рук"	☑ 🗑

+ Add Param | Bulk Edit

Path

Name	Value

Response | Headers 9 | Timeline | Tests

200 OK 2.82s 3.51KB


```
{
  "text": "Разведение рук в тренажёре «бабочка» — это изолирующее упражнение, направленное на проработку грудных мышц. Для его выполнения вам понадобится соответствующий тренажёр.\n\n**Техника выполнения:**\n\n1. Сядьте в тренажёр «бабочка» и настройте его под свои параметры.\n2. Возьмите ручки тренажёра и прижмите их к груди.\n3. На выдохе разведите руки в стороны, раскрывая грудь.\n4. На вдохе медленно верните руки в исходное положение.\n\n**Параметры:**\n- Вес инвентаря: 30,0.\n- Количество повторений: 12.\n- Количество подходов: 4.\n\nПеред выполнением упражнения обязательно проконсультируйтесь с тренером или специалистом по физической подготовке, чтобы убедиться, что это упражнение подходит именно вам и выполняется правильно."
}
```

MCP Server

```python
mcp = FastMCP(
    name='Retrieval-Augmented Generation adapter for vector database',
    instructions='A server for interacting with the Retrieval-Augmented Generation project',
    version='1.0'
)

SourceCraft Code Assistant: Generate Tests | Generate Docs | ×
@mcp.tool
async def search_workout(query: Annotated[str, "query to search workout"], ctx: Context) -> str:
    """Function to search workout by query in vector database and return result with Retrieval-Augmented Generation"""

    await ctx.info("Starting searching documents")
    server_logger.info("Searching documents by query: %s", query)
      You, last month • feature: add mcp server
    result = requests.get(os.environ['RAG_URL'], params={'prompt':query}, timeout=20)

    return result.json()['text']
```

MCP Client

```python
async def init_agent() -> Any:
    global _agent
    if _agent is None:
        chat_model: ChatOpenAI = ChatOpenAI(
            model=current_app.config['LLM_MODEL_URL'],
            base_url=current_app.config['LLM_BASE_URL'],
            api_key=os.environ.get("YC_API_KEY"),
        )
        mcp_client: MultiServerMCPClient = MultiServerMCPClient(current_app.config['MCP_SERVERS'])
        tools: List[Any] = await mcp_client.get_tools()
        _agent = create_agent(model=chat_model, tools=tools, system_prompt=current_app.config['ASSISTENT_ROLE'])
    return _agent
```

SourceCraft Code Assistant: Generate Tests | Generate Docs | ×

```python
async def send_message(message: Dict[str, List[Any]]) -> str:
    global _agent
    if _agent is None:        You, last month • feature: add mcp-client with webui …
        await init_agent()
    result: Dict[str, Any] = await _agent.ainvoke(message)

    # Extract the final AI response, prioritizing AIMessage over ToolMessage
    ai_messages = [msg for msg in result['messages'] if isinstance(msg, AIMessage) and msg.content]
    tool_messages = [msg for msg in result['messages'] if isinstance(msg, ToolMessage) and msg.content]
```

MCP Chat with YandexGPT

Hello! I'm your assistant powered by MCP and YandexGPT. How can I help you today?

Какие функции тебе доступны в запросе?

В запросе доступна функция `search_workout`, которая позволяет искать тренировки по запросу в векторной базе данных и возвращать результат с использованием Retrieval-Augmented Generation. Для использования этой функции требуется указать параметр `query` (запрос для поиска тренировки).

Предложи мне тренировку для ног

Для проработки бицепса бедра можно выполнить следующее упражнение: **Сгибание ног в тренажёре** **Оборудование:** тренажёр для сгибания ног. **Техника выполнения:** 1. Установите необходимый вес на тренажёре. В данном случае вес инвентаря составляет 40.0. 2. Сядьте в тренажёр и зафиксируйте ноги под валиками. 3. Ноги должны быть выпрямлены. 4. На выдохе плавно сгибайте ноги в коленях, поднимая валики к ягодицам. 5. Задержитесь на секунду в верхней точке, максимально напрягая бицепс бедра. 6. На вдохе медленно верните ноги в исходное положение. **Параметры:** * **Количество повторений:** 12. * **Количество подходов:** 3. Перед выполнением упражнения обязательно

Type your message here...

Send

rag-project

AI-ассистенты для IDE

SourceCraft Code Assistant

Поддержка:

- Visual Studio Code
- JetBrains IDE's

Тариф:

- Completions: 3000 в неделю
- Code Assistant IDE: 1000 в неделю
- Code Assistant Chat: 200 в день
- До конца 2025г не терифицируется, Pro-версия предварительно 700р в месяц

Дополнительно:

- Была проблема с установкой на форки VSCode

GigaCode

Поддержка:

- GigaIDE
- Visual Studio Code
- JetBrains IDE's
- Android Studio
- Jupyter Notebook

Тариф:

- Бесплатно

Дополнительно:

- Нет поддержки MCP

Не забывайте про безопасность!

Промтинг

- Шаблоны - сохраняйте промпты в шаблоны для переиспользования

- Метапромтинг – прием предлагающий создать запрос для генерации лучшего промпта для другой задачи.
- Цепочка рассуждений (Chain-of-Thought, CoT) – прием при котором указываем модели вывести рассуждение как она пришла к данному решению.
- Ролевые запросы - метод, когда искусственный интеллект ассоциируется с некоторой ролью которую нужно имитировать.
- От меньшего к большему – метод когда в котором новый запрос основан на предыдущем ответе.
- Нулевой выстрел (Zero-shot) - подход, при котором модель решает задачу без примеров, опираясь только на формулировку запроса.
- Несколько выстрелов (Few-shot) — метод, где в промпт включаются 1–3 примера желаемого формата/решения, чтобы задать шаблон для ответа.

- Ограничение формата — техника, требующая строго заданного вида ответа (таблица, список, код, диалог и т. п.) для структурирования вывода.
- Обратный промтинг — способ, при котором модель сначала генерирует возможный вопрос к уже имеющемуся ответу, а затем использует его для уточнения запроса.
- Отрицательный промтинг - указание, чего не должно быть в ответе (запрещённые темы, стили, форматы), чтобы сузить пространство решений.
- Разделение задач — метод разбиения сложного запроса на подзадачи, которые модель решает последовательно, а потом объединяет результаты.
- Проверка и итерация — цикл из запроса, анализа ответа, выявления недочётов и повторного запроса с уточнениями для улучшения результата.
- Контекстное обогащение — включение в промпт дополнительных данных (фактов, ссылок, предыстории), чтобы модель учитывала их при генерации.

Визитка

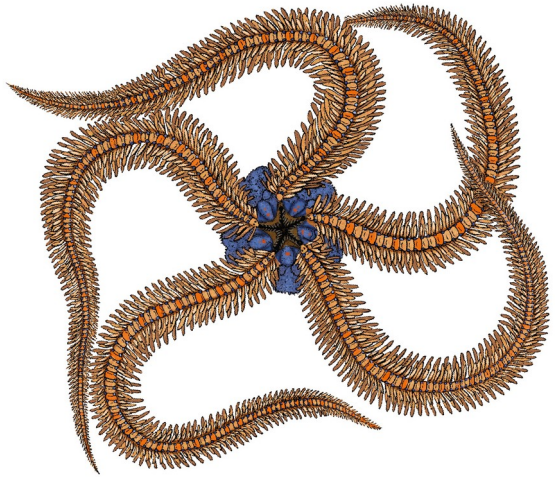
Заметки

O'REILLY®

Второе издание

Разработка приложений на базе GPT-4 и ChatGPT

интеллектуальные чат-боты, генераторы контента и многое другое



SPRINT book

Оливье Келен
Мари-Алис Блете

EXPERT INSIGHT

SPRINT book

RAG и генеративный ИИ

Создаем собственные RAG-пайплайны с помощью LlamaIndex, Deep Lake и Pinecon

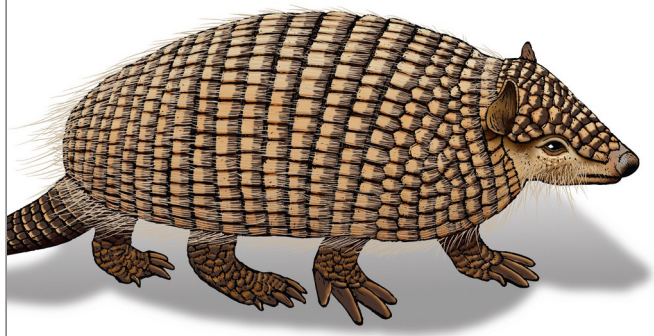
Дэнис Ротман

<packt>

O'REILLY®

Промт-инжиниринг для GenAI

Паттерны надежных запросов для качественных результатов



SPRINT book

Джеймс Феникс
Майк Тейлор