



**ИНСТРУКЦИЯ
ПО ЭКСПЛУАТАЦИИ СИСТЕМЫ
«СЕРВИС ГЕНЕРАЦИИ
ТЕКСТА НА ЯЗЫКАХ
ПРОГРАММИРОВАНИЯ»**

Версия 1.2 • 16/12/2024

ООО «МТС ИИ»
Г. МОСКВА

ОГЛАВЛЕНИЕ

1	О продукте	3
2	Архитектура	3
3	Настройка и установка плагина в среде Visual Studio Code	5
4	Работа с плагином	7
5	Справочник по API	9
5.1	Описание запроса POST/chat/completions	9
5.1.1	Использование запроса POST/v1/chat/completions	11
5.1.2	Результат выполнения запроса POST/v1/chat/completions	12
5.2	GET/v1/models	14
5.3	GET/health	15
5.4	POST/v1/completions	16
5.5	POST/v1/embeddings	17
5.6	POST/token	17
5.7	GET/users/me	18
5.8	GET/get_user_info	18
5.9	POST/create_user	19
5.10	POST/disable_by_email	20
5.11	POST/disable_by_token	20
5.12	POST/disable_all_expired	21
5.13	POST/delete_by_email	21
5.14	POST/delete_by_token	22
5.15	POST/delete_all_disabled	23

В каждую модель, предоставляемую нашим клиентам и партнерам, мы встраиваем уникальный водяной знак. Это необходимо для установления факта утечки модели от клиента или партнера. Просим вас ответственно относиться к обеспечению безопасности хранения модели.

1 — О ПРОДУКТЕ

«Сервис генерации текста на языках программирования» или «Kodify» (далее по тексту будет использовано второе наименование ПО) от MTS AI представляет собой AI-ассистента разработчика, инструмент, который использует искусственный интеллект для автоматизации рутинных процессов и помощи разработчикам в выполнении различных задач при написании программного кода. ПО предназначено для облегчения процесса разработки, предоставляя инструменты для генерации кода, улучшения его качества и автоматизации рутинных задач. Kodify поддерживает различные языки программирования и интеграции с популярными инструментами разработки.

Функционал продукта:

Kodify от MTS AI предоставляет функционал для автоматической генерации кода на основе данных из редактируемого пользователем файла. Продукт поддерживает различные языки программирования (например, Python, C#, Java, Go, JavaScript). ПО обеспечивает автопродление кода с помощью LLM, которая на основе уже написанного пользователем кода генерирует завершение строки. Интеграция с инструментами разработки включает встраивание специального плагина в популярные интегрированные среды разработки (IDE), таких как Visual Studio Code.

Преимущества продукта:

- **Повышение производительности:**

Kodify от MTS AI обеспечивает повышение производительности разработчиков за счет автоматической генерации кода и упрощения процесса разработки, что ускоряет процесс написания кода и снижает время на исправление ошибок и оптимизацию.

- **Улучшение качества программного обеспечения:**

Продукт способствует улучшению качества программного обеспечения посредством генерации кода высокого качества с меньшим количеством багов и уязвимостей.

2 — АРХИТЕКТУРА

Пользователь взаимодействует с системой через интерфейс Visual Studio Code (VSCode).

В данном контексте IDE используется как интерфейс для отправки запросов к системе Kodify через специальный плагин, устанавливаемый отдельно. Установка плагина в IDE даёт пользователю возможность обращаться к Kodify для автоматической генерации кода на основе данных из редактируемого пользователем файла, а также другим функциям Kodify.

Kodify — это основная система, которая выполняет обработку запросов. MTS Kodify предоставляет REST API, к которому обращается плагин, установленный в IDE.

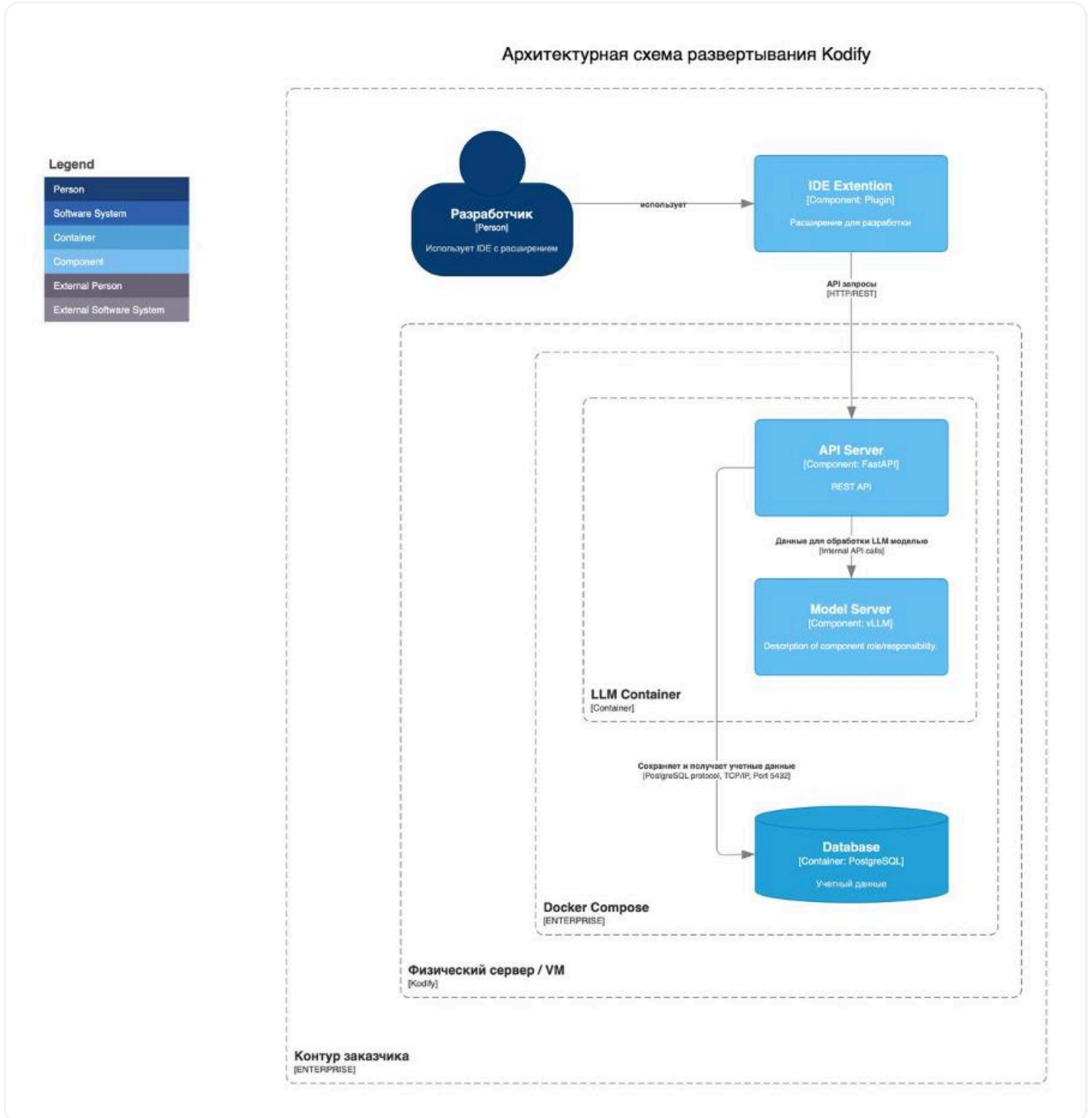


Схема 1. Взаимодействие компонентов

3 — НАСТРОЙКА И УСТАНОВКА ПЛАГИНА В СРЕДЕ VISUAL STUDIO CODE

На текущий момент поддерживается среды разработки Visual Studio Code.

1. Скачивание плагина и конфигурационного файла

Скачайте последнюю версию плагина Kodify для Visual Studio Code. Ссылку для скачивания предоставит сотрудник МТС ИИ.

2. Установка плагина в Visual Studio Code

2.1 Перейдите в раздел «extensions» на левой панели инструментов.

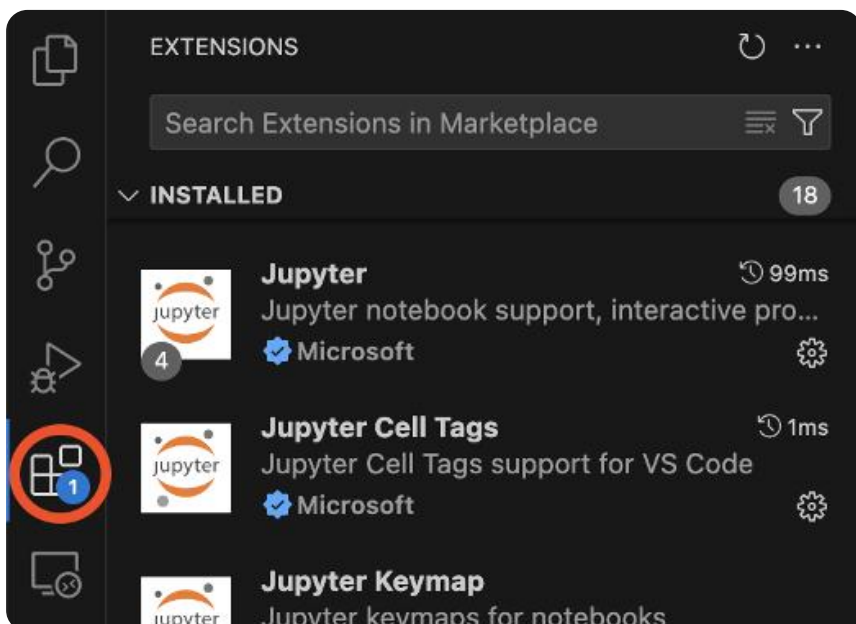


Рисунок 1. Раздел «Extensions» в панели инструментов

2.2 В диалоговом меню выберите пункт «Install from VSIX» и укажите скачанный файл плагина.

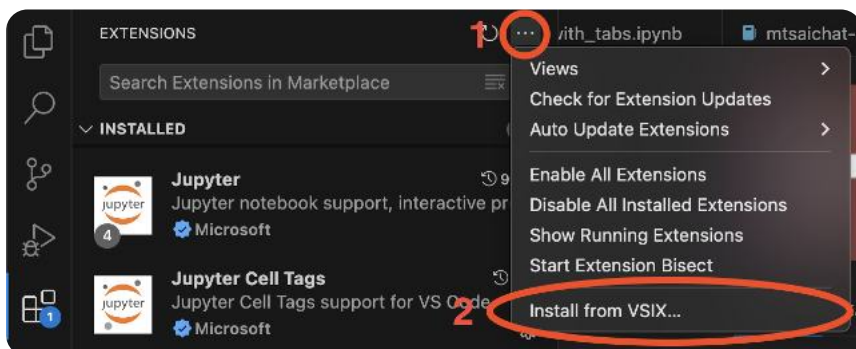


Рисунок 2. Выбор пункта «Install from VSIX» в диалоговом окне

3. Настройка плагина

3.1 После установки плагина откройте в IDE любой файл. используйте комбинацию клавиш CTRL + L, чтобы открыть боковую панель. на боковой панели нажмите на кнопку «Open Kodify Konfig».

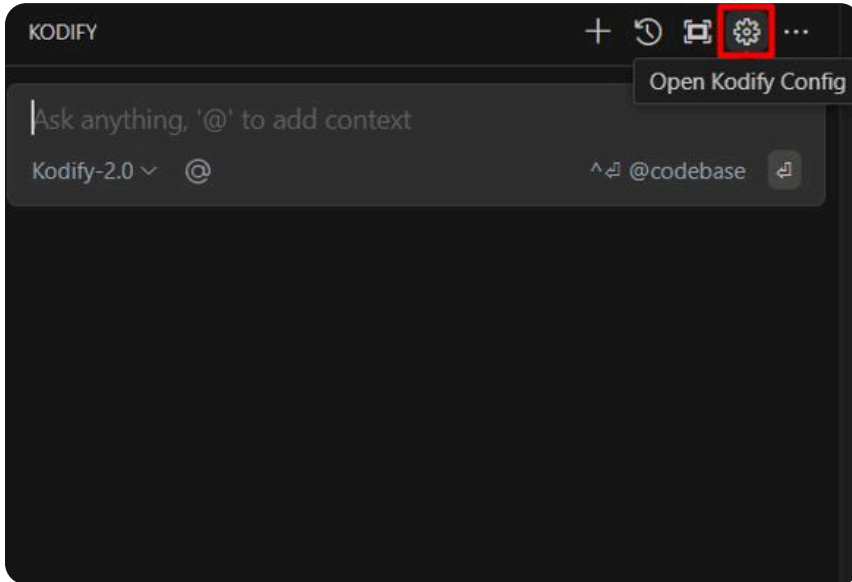


Рисунок 3. Кнопка «Open Kodify Konfig»

3.2 В заполнение конфигурационного файла.

Если вы планируете обращаться к Kodify в контуре МТС ИИ, то в конфигурационном файле для плагина необходимо указать токен, который будет использоваться плагином при отправке api-запросов к серверу. Для получения токена обратитесь к сотруднику МТС ИИ.

Токен требуется указать трижды:

- в разделе «tabAutocompleteModel» в поле «apiKey»
- в разделе «models» в поле «apiKey»
- в разделе «embeddingsProvider» в поле «apiKey»

Если же в контуре вашей компании развёрнут отдельный экземпляр Kodify, то помимо токена вам потребуется заполнить в конфигурационном файле следующие поля:

- в разделе «tabAutocompleteModel» следует указать «apiBase» (адрес сервера, к которому вы будете обращаться) и «model» (id модели)
- в разделе «models» следует указать «apiBase» и «model»
- в разделе «embeddingsProvider» следует указать «model»

4 — РАБОТА С ПЛАГИНОМ

Плагин настроен и готов к работе. Сервис подключится автоматически при необходимости перезагрузите IDE).

1. Автокомплита кода

Откройте любой файл с Python-кодом или иным языком программирования, поставьте курсор в любое свободное место и убедитесь, что плагин предлагает автодополнения.



```
def hello_world():  
    return print("Hello, World")
```

Рисунок 6. Пример работоспособности плагина

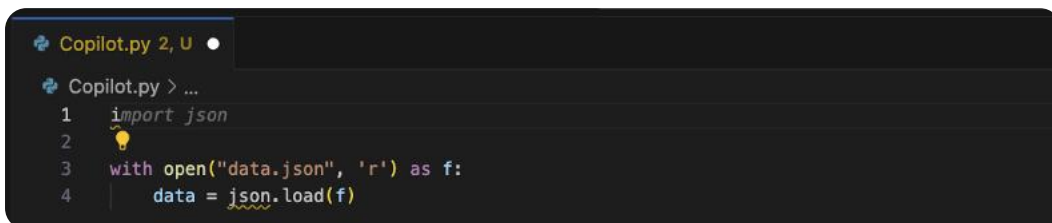
1.1 Например, введите строку:

```
with open("data.json", 'r') as f:
```

1.2 В строке выше введите:

```
im
```

1.3 Автодополнение должно предложить `import json`.



```
Copilot.py 2, U  
Copilot.py > ...  
1 import json  
2  
3 with open("data.json", 'r') as f:  
4     data = json.load(f)
```

Рисунок 7. Пример работы автокомплита

Если возникли проблемы при установке или настройке, пожалуйста, свяжитесь с нами удобным способом, и мы обязательно поможем.

2. Функционал помощника разработчика

2.1 Откройте файл с кодом и выделите часть кода. Плагин предложит нажать комбинацию клавиш «Ctrl+L» или «Ctrl+I»:

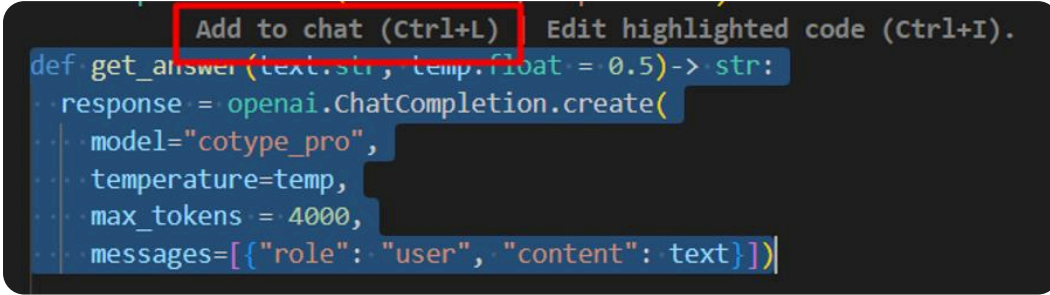


Рисунок 8. Выделение части кода

2.2 В открывшемся окне введите знак «/», появятся доступные варианты:

- /test - генерирует Unit-test к выделенной части кода;
- /docs - добавляет docstring к выделенной части кода;
- /explain (rus) - объясняет что делает код на русском языке;
- /fix - ищет и исправляет ошибки в выделенной части кода;
- Также вместо использования одного из предустановленных вариантов вы можете самостоятельно описать те действия, которые Kodify должен будет произвести с выделенной частью кода.

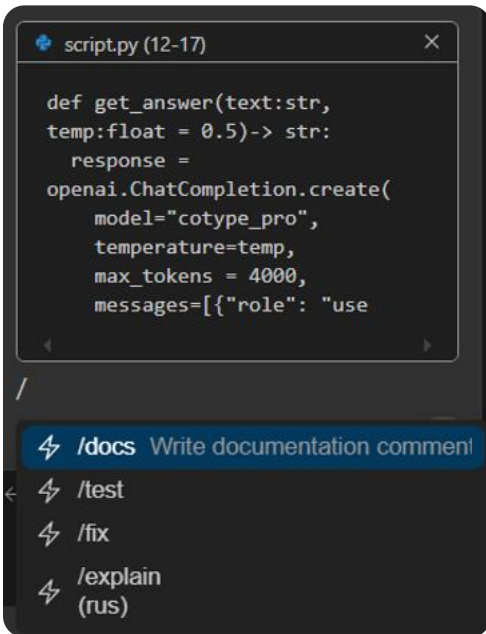


Рисунок 9. Функциональные возможности плагина

2.3 Выберите нужный вам вариант или опишите собственный и нажмите на кнопку отправки.

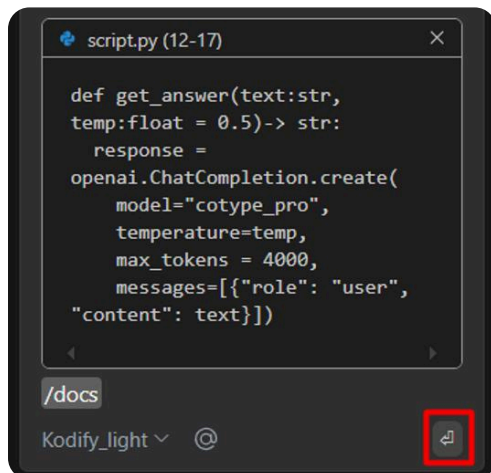


Рисунок 10. Кнопка отправки запроса в Kodify

2.4 Будет отправлен соответствующий запрос к Kodify. Ожидайте ответ от системы.

5 — СПРАВОЧНИК ПО API

Этот справочник по API предоставляет базовую информацию для работы с продуктом и сообщениями в системе.

5.1 Описание запроса POST/chat/completions

Основным методом для взаимодействия с моделью является метод POST/chat/completions. Он предназначен для отправки списка сообщений в формате чата. На основе полученных входных данных модель сгенерирует ответ на запрос пользователя. Метод может использоваться как для ведения диалогов, состоящих из нескольких последовательных сообщений, так и для выполнения задач, требующих однократного обращения без продолжительного разговора.

Тип запроса: POST

Запрос: http://IP_Address/v1/chat/completions

Где «IP_Address» необходимо заменить на IP-адрес вашей машины, если модель развёрнута внутри вашего контура. Если вы обращаетесь к демо-стенду внутри контура МТС ИИ, то адрес стенда будет передан вам отдельно.

Например: <http://1.1.1.1:8000/v1/chat/completions>

Таблица 1. Параметры запроса POST/chat/completions

Имя	Тип данных	Значение	Описание
Messages (обязательный)	string	Текстовый ответ	Одно сообщение или список из нескольких сообщений в формате чата, на основе которых модель должна сгенерировать ответ
Model (обязательный)	string	ID	ID модели, к которой вы обращаетесь
temperature	float	<p>Диапазон температуры — от 0 до 2</p> <p>Рекомендованное значение: 0.5</p>	Более низкие значения температуры приводят к более последовательным результатам (например, 0.2), в то время как более высокие значения генерируют более разнообразные и творческие результаты (например, 1.0)
top_p	int	Диапазон — от 0 до 1	<p>Чем ниже значение атрибута, тем более популярные и часто встречающиеся токены модель будет использовать для формирования ответа</p> <p>Рекомендуем изменять или этот атрибут, или temperature, но не оба сразу</p>
max_tokens	int	Натуральное число больше 0	Максимальное количество токенов, которые могут быть сгенерированы в ответ на запрос пользователя. Это позволяет регулировать длину ответа
n	int	<p>Натуральное число больше 0:</p> <p>По умолчанию: 1</p>	Количество ответов, которые модель сгенерирует
stream	bool	True/False	Если установить значение true, ответ модели будет возвращаться не целиком сразу, а итеративно, по мере его формирования моделью
frequency_penalty	int	Натуральное число от -2 до 2	Штраф за частоту — число между -2.0 и 2.0. Положительные значения штрафуют новые токены на основе их текущей частоты в тексте, снижая вероятность того, что модель повторит одну и ту же строку
presence_penalty	int	Натуральное число от -2 до 2	Положительные значения накладывают штраф на новые токены в зависимости от того, появляются ли они в тексте до сих пор, увеличивая вероятность того, что модель будет говорить о новых темах

5.1.1 Использование запроса POST/v1/chat/completions

Укажите **Bearer Token**. При активной однотокеновой авторизации токен одинаковый у всех пользователей, при пользовательской авторизации выдаются индивидуальные токены. Подробнее о типах авторизации и их настройке в разделе 5.2

Заполните тело запроса. Обязательные параметры: `model` и `messages`. В `model` необходимо указать ID вашей модели. Получить ID модели можно с помощью запроса `GET/models`, подробное описание в разделе 6.2.

В `messages` необходимо указать «`role`» и «`content`». У атрибута «`role`» может быть одно из двух значений:

- «`system`» - обозначение для системного промпта, который задает роль модели, например, что модель должна отвечать как учитель или как политик. Необязательный атрибут;
- «`user`» - обозначение пользовательского промпта, который содержит ваш вопрос или ваши инструкции для модели. Обязательный атрибут.

В параметр «`content`» запишите ваш системный или пользовательский промпт.

```
{
  "model": "//ID вашей модели",
  "messages": [
    {
      "role": "system",
      "content": "Отвечай как экскурсовод"
    },
    {
      "role": "user",
      "content": "Расскажи мне о Москве в 1 предложении."
    }
  ]
}
```

Пример заполнения представлен на рисунке ниже.

```
1  {
2    "model": "cotype_pro_16k_1.1",
3    "messages": [
4      {
5        "role": "system", "content": "Отвечай как экскурсовод"},
6      {
7        "role": "user", "content": "Расскажи мне о Москве в 1 предложении."}
8    ]}
9
```

Рисунок 11. Пример заполнения раздела `Body`

Пример cURL-запроса для выполнения запроса из командной строки:

```
curl https://{url}/v1/chat/completions \  
-H "Content-Type: application/json" \  
-H "Authorization: Bearer Token" \  
-d '{"model": "cotype_pro_16k_1.1", "messages": [  
  {"role": "system", "content": "Отвечай как экскурсовод"},  
  {"role": "user", "content": " Расскажи мне о Москве в 1 предложении."}]}'
```

5.1.2 Результат выполнения запроса POST/v1/chat/completions

Результат успешного запроса:

```
{  
  "id": "cmpl-e263c7d6179a43e98b2ca9580b57e4f6",  
  "object": "chat.completion",  
  "created": 1724069159,  
  "model": "cotype_pro_16k_1.1",  
  "choices": [  
    {  
      "index": 0,  
      "message": {  
        "role": "assistant",  
        "content": "Москва - это столица России, древний и современный  
город, объединяющий богатую историю и культуру, архитектурные шедевры,  
таких как Кремль и храм Василия Блаженного, с динамичной городской  
жизнью, современными технологиями и инфраструктурой, а также  
уникальной атмосферой, которая сочетает в себе традиции и инновации."  
      },  
      "logprobs": null,  
      "finish_reason": "stop",  
      "stop_reason": null  
    }  
  ],  
  "usage": {  
    "prompt_tokens": 89,  
    "total_tokens": 193,  
    "completion_tokens": 104  
  }  
}
```

Таблица 2. Параметры ответа на запрос POST/chat/completions

Имя	Тип данных	Пример значения	Описание
id	string	cmpl-e263c7d6179a43e98b2ca9580b57e4f6	Идентификатор запроса
created	string	1724069159	Временная метка UNIX, отмечающая дату и время, когда был получен запрос
Message	string	Москва - это столица России, древний и современный город, объединяющий богатую историю и культуру, архитектурные шедевры, таких как Кремль и храм Василия Блаженного, с динамичной городской жизнью, современными технологиями и инфраструктурой, а также уникальной атмосферой, которая сочетает в себе традиции и инновации	Сгенерированное моделью сообщение
Model	string	cotype_pro_16k_1.1	ID модели, к которой вы обращались и которая ответила на ваш запрос.
usage	int	«prompt_tokens»: 89, «total_tokens»: 193, «completion_tokens»: 104	Статистика по токенам в вашем запросе. prompt_tokens – количество токенов в запросе; completion_tokens – количество сгенерированных моделью токенов; total_tokens – сумма токенов в запросе и сгенерированных токенов.

Для продолжения общения с моделью в рамках текущего диалога необходимо дополнить ваш предыдущий запрос ответом модели, а также добавить новое сообщение от лица пользователя. Атрибут `role` показывает, кто является автором сообщения в диалоге. `Assistant` — ответ модели, `user` — сообщение пользователя.

Пример тела запроса:

```
{
  "model": "//имя вашей модели",
  "messages":
  [ {"role": "system", "content": "Отвечай как экскурсовод"},
    {"role": "user", "content": "Расскажи мне о Москве в 1 предложении."},
    {"role": "assistant", "content": "Москва - это столица России, древний и современный город, объединяющий богатую историю и культуру, архитектурные шедевры, таких как Кремль и храм Василия Блаженного, с динамичной городской жизнью, современными технологиями и инфраструктурой, а также уникальной атмосферой, которая сочетает в себе традиции и инновации." },
    {"role": "user", "content": "Расскажи мне более подробно о Кремле:"}
  ]
}
```

Модель сформирует ответ на ваш запрос с учётом контекста диалога. В данном примере расскажет именно о московском Кремле, а не о казанском или новгородском.

5.2 GET/v1/models

Метод для получения списка доступных вам моделей.

Тип запроса: GET

Запрос:

```
http://IP_Address/v1/models
```

- Где **IP_Address** необходимо заменить на IP-адрес вашей машины или адрес демо-стенда. Например:

```
http://1.1.1.1:8000/v1/models
```

Пример cURL-запроса для выполнения запроса из командной строки:

```
curl -X 'GET' \  
'https://{url}/v1/models' \  
-H 'accept: application/json' \  
-H 'Authorization: Bearer Token' \  

```

Пример ответа на запрос GET/models:

```
{
  "object": "list",
  "data": [
    {
      "id": "cotype_pro_16k_1.1",
      "object": "model",
      "created": 1724087433,
      "owned_by": "vllm",
      "root": "cotype_pro_16k_1.1",
      "parent": null,
      "permission": [
        {
          "id": "modelperm-1b7dd7dea5244bdb888971bf2b89f6ff",
          "object": "model_permission",
          "created": 1724087433,
          "allow_create_engine": false,
          "allow_sampling": true,
          "allow_logprobs": true,
          "allow_search_indices": false,
          "allow_view": true,
          "allow_fine_tuning": false,
          "organization": "*",
          "group": null,
          "is_blocking": false
        }
      ]
    }
  ]
}
```

В результате выполнения запроса вы получите список доступных вам моделей. Для обращения к модели скопируйте «id» — уникальный идентификатор модели, он выделен красным цветом в теле ответа на запрос. Затем используйте скопированный ID в запросе POST/chat/completions в параметре model.

5.3 GET/health

Назначение: Запрос проверяет работоспособность модели. Если модель работоспособна, будет получен ответ [200 OK]. В случае если сервис недоступен по какой-либо причине, ответ не будет получен.

Тип запроса: GET

Запрос:

```
http://IP_Addres/health
```

- Где **IP_Address** необходимо заменить на IP-адрес вашей машины или адрес демо-стенда. Например:

```
http://1.1.1.1:8000/health
```

Пример cURL-запроса для выполнения запроса из командной строки:

```
curl -X 'GET' \  
'https://{url}/health' \  
-H 'accept: application/json'
```

5.4 POST/v1/completions

Назначение: Устаревший функционал. Система дополняет и продолжает промпт пользователя. Не подходит для общения в чатовом режиме.

Тип запроса: POST

Запрос:

```
https://{IP-адрес}/v1/completions
```

- Где **IP_Address** необходимо заменить на IP-адрес вашей машины или адрес демо-стенда. Например:

```
http://1.1.1.1:8000/v1/completions
```

Пример cURL-запроса для выполнения запроса из командной строки:

```
curl -X 'POST' \  
'https://{IP-адрес}/v1/completions' \  
-H 'accept: application/json' \  
curl https://{IP-адрес}/v1/completions \  
-H "Content-Type: application/json" \  
-H "Authorization: Bearer Token" \  
-d '{"model": "cotypro_16k_1.1", "prompt": "Тестовый запрос"}'
```


5.5 POST/v1/embeddings

Назначение: Запрос трансформирует отправленный текст в эмбединги. Предназначен для представления текста в векторном виде.

Тип запроса: POST

Запрос:

```
https://IP_Address/v1/embeddings
```

- Где **IP_Address** необходимо заменить на IP-адрес вашей машины или адрес демо-стенда. Например:

```
http://1.1.1.1:8000/v1/embeddings
```

Пример cURL-запроса для выполнения запроса из командной строки:

```
curl -X 'POST' \  
  'https://{IP-адрес}/v1/embeddings' \  
  -H 'accept: application/json' \  
curl https://{IP-адрес}/v1/embeddings \  
  -H "Content-Type: application/json" \  
  -H "Authorization: Bearer Token" \  
  -d '{"text": ["Тестовый запрос"]}'
```

5.6 POST/token

Назначение: Получение токена доступа для работы с API.

Тип запроса: POST

Запрос:

```
https://IP_Address/token
```

- Где **IP_Address** необходимо заменить на IP-адрес вашей машины или адрес демо-стенда. Например:

```
http://1.1.1.1:8000/token
```

Пример cURL-запроса для выполнения запроса из командной строки:

```
curl -X 'POST' \  
  'https://{url}/token' \  
  -H 'accept: application/json' \  
  -H 'Content-Type: application/x-www-form-urlencoded' \  
  -d 'username=your.email@example.ru&password=your_password'
```

5.7 GET/users/me

Назначение: Получение информации о текущем авторизованном пользователе.

Тип запроса: GET

Запрос:

```
https://IP_Address/users/me
```

- Где **IP_Address** необходимо заменить на IP-адрес вашей машины или адрес демо-стенда. Например:

```
http://1.1.1.1:8000/users/me
```

Пример cURL-запроса для выполнения запроса из командной строки:

```
curl -X 'GET' \  
  'https://{url}/users/me' \  
  -H 'accept: application/json' \  
  -H 'Authorization: Bearer YOUR_TOKEN'
```

5.8 GET/get_user_info

Назначение: Получение информации о пользователе по email (только для администраторов).

Тип запроса: GET

Запрос:

```
https://IP_Address/get_user_info?email=user@example.ru
```

- Где **IP_Address** необходимо заменить на IP-адрес вашей машины или адрес демо-стенда. Например:

```
http://1.1.1.1:8000/get_user_info?email=user@example.ru
```

Пример cURL-запроса для выполнения запроса из командной строки:

```
curl -X 'GET' \  
'https://{url}/users/me' \  
-H 'accept: application/json' \  
-H 'Authorization: Bearer YOUR_TOKEN'
```

5.9 POST/create_user

Назначение: Создание нового пользователя (только для администраторов).

Тип запроса: POST

Запрос:

```
https://IP_Address/create_user
```

- Где **IP_Address** необходимо заменить на IP-адрес вашей машины или адрес демо-стенда. Например:

```
http://1.1.1.1:8000/create_user
```

Пример cURL-запроса для выполнения запроса из командной строки:

```
curl -X 'POST' \  
'https://{url}/create_user' \  
-H 'accept: application/json' \  
-H 'Authorization: Bearer ADMIN_TOKEN' \  
-H 'Content-Type: application/x-www-form-urlencoded' \  
-d 'email=new.user@example.ru&password=user_password&expire=28&privileged=false'
```

5.10 POST/disable_by_email

Назначение: Блокировка пользователя по email (только для администраторов).

Тип запроса: POST

Запрос:

```
https://IP_Address/disable_by_email
```

- Где **IP_Address** необходимо заменить на IP-адрес вашей машины или адрес демо-стенда. Например:

```
http://1.1.1.1:8000/disable_by_email
```

Пример cURL-запроса для выполнения запроса из командной строки:

```
curl -X 'POST' \  
  'https://{url}/disable_by_email' \  
  -H 'accept: application/json' \  
  -H 'Authorization: Bearer ADMIN_TOKEN' \  
  -H 'Content-Type: application/x-www-form-urlencoded' \  
  -d 'email=user@example.ru'
```

5.11 POST/disable_by_token

Назначение: Блокировка пользователя по токену (только для администраторов).

Тип запроса: POST

Запрос:

```
https://IP_Address/disable_by_token
```

- Где **IP_Address** необходимо заменить на IP-адрес вашей машины или адрес демо-стенда. Например:

```
http://1.1.1.1:8000/disable_by_token
```

Пример сURL-запроса для выполнения запроса из командной строки:

```
curl -X 'POST' \  
  'https://{url}/disable_by_token' \  
  -H 'accept: application/json' \  
  -H 'Authorization: Bearer ADMIN_TOKEN' \  
  -H 'Content-Type: application/x-www-form-urlencoded' \  
  -d 'token=user_token'
```

5.12 POST/disable_all_expired

Назначение: Блокировка всех пользователей с истекшим сроком действия (только для администраторов).

Тип запроса: POST

Запрос:

```
https://IP_Address/disable_all_expired
```

- Где **IP_Address** необходимо заменить на IP-адрес вашей машины или адрес демо-стенда. Например:

```
http://1.1.1.1:8000/disable_all_expired
```

Пример с URL-запроса для выполнения запроса из командной строки:

```
curl -X 'POST' \  
  'https://{url}/disable_all_expired' \  
  -H 'accept: application/json' \  
  -H 'Authorization: Bearer ADMIN_TOKEN'
```

5.13 POST/delete_by_email

Назначение: Удаление пользователя по email (только для администраторов).

Тип запроса: POST

Запрос:

```
https://IP_Address/delete_by_email
```

- Где **IP_Address** необходимо заменить на IP-адрес вашей машины или адрес демо-стенда. Например:

```
http://1.1.1.1:8000/delete_by_email
```

Пример cURL-запроса для выполнения запроса из командной строки:

```
curl -X 'POST' \  
'https://{url}/delete_by_email' \  
-H 'accept: application/json' \  
-H 'Authorization: Bearer ADMIN_TOKEN' \  
-H 'Content-Type: application/x-www-form-urlencoded' \  
-d 'email=user@example.ru'
```

5.14 POST/delete_by_token

Назначение: Удаление пользователя по токenu (только для администраторов).

Тип запроса: POST

Запрос:

```
https://IP_Address/delete_by_token
```

- Где **IP_Address** необходимо заменить на IP-адрес вашей машины или адрес демо-стенда. Например:

```
http://1.1.1.1:8000/delete_by_token
```

Пример cURL-запроса для выполнения запроса из командной строки:

```
curl -X 'POST' \  
'https://{url}/delete_by_token' \  
-H 'accept: application/json' \  
-H 'Authorization: Bearer ADMIN_TOKEN' \  
-H 'Content-Type: application/x-www-form-urlencoded' \  
-d 'token=user_token'
```

5.15 POST/delete_all_disabled

Назначение: Удаление всех заблокированных пользователей (только для администраторов).

Тип запроса: POST

Запрос:

```
https://IP_Address/delete_all_disabled
```

- Где **IP_Address** необходимо заменить на IP-адрес вашей машины или адрес демо-стенда. Например:

```
http://1.1.1.1:8000/delete_all_disabled
```

Пример cURL-запроса для выполнения запроса из командной строки:

```
curl -X 'POST' \  
  'https://{url}/delete_all_disabled' \  
  -H 'accept: application/json' \  
  -H 'Authorization: Bearer ADMIN_TOKEN'
```