

Использование бота StGptBot для генерации тестовых кейсов

Как найти бота

Чтобы найти бота в поиске в телеграмм необходимо ввести его username - @CTT_gpt_bot

Функции бота

/start - старт работы с ботом

/help - вывод списка функций бота




/clearSession - удаление сессии (токены в Jira и Wiki самостоятельно, например, когда ввели ошибочный токен и система его приняла)

/getcasejira - генерация тестового кейса в Jira по переданному идентификатору сущности (**берется описание задачи и по нему генерируется тестовые кейсы**)

После вызова данной функции необходимо сделать следующее:

1) Выбрать инструмент для генерации тестового кейса (модель gpt):

- YandexGPT (работает лучше всего)
- ChatGPT 3.5 Turbo (работает через проксирование, дольше чем YaGPT, но неплохо генерирует кейсы)
- SberGigachat (странно генерирует кейсы, но надо тестировать)

 Write a message...  

YaGPT

ChatGPT 3.5 Turbo

Sber Gigachat

2) Необходимо передать Jira API Token, чтобы вычитать описание задачи, в формате JiraToken {MyToken}. Пример: JiraToken abcd123. Как получить токен можно почитать тут - [Получение api token в Jira/Wiki и работа со swagger для генерации кейсов](#)

Токен живет в системе до следующего календарного дня, поэтому при повторных вызовах метода передавать токен будет не нужно.

3) Передать идентификатор задачи в Jira в формате ERS-XXXX

Как кейсы будут сгенерированы, придет сообщение с готовыми кейсами:

Кейсы готовы: 18:32

Конечно, вот несколько функциональных тестовых кейсов для требований "Необходимо проработать макеты для редизайна страницы 'Бизнесу'":

1. ****Проверка внешнего вида и оформления страницы:****

- Загрузить страницу "Бизнесу".
- Убедиться, что внешний вид соответствует утвержденным макетам для редизайна.
- Проверить цветовую гамму, шрифты, стилизацию элементов и их расположение.

2. ****Проверка навигации:****

- Проверить работу основного меню.
- Убедиться, что все разделы страницы доступны из главного меню.
- Проверить корректность переходов между разделами и подразделами.

3. ****Проверка контента:****

- Проверить наличие основной информации о продуктах или услугах для бизнеса.
- Убедиться, что контент соответствует требованиям и предоставленным макетам.
- Проверить наличие и корректность ссылок на дополнительные ресурсы или материалы.

Дальше будет предложены следующие варианты:

- Сгенерировать еще дополнительные кейсы (доступно только при выборе yaGPT), позволяет получить еще больше кейсов по переданной ранее информации (/getMoreCases). Необходимо выполнять до отправки результата (/sendToJira), после генерации дополнительных кейсов будет возможность отправить общий набор(его можно вызывать только сразу после получения сгенерированных кейсов, в другой момент вызов данного метода не доступен)
- Опубликовать текущие сгенерированные кейсы в задачу Jira в комментарии через метод /sendToJira (его можно вызывать только сразу после получения сгенерированных кейсов/дополнительно сгенерированных кейсов, в другой момент вызов данного метода не доступен)
- Отправить сгенерированные кейсы в комментарий к другой задаче (например, если необходима отправка кейсов в отдельную задачу на тестирование). Для этого необходимо передать номер тикета, куда отправить кейсы в формате send ERS-XXXX, после чего выполнить /sendToJira

*** Если хочешь сгенерировать дополнительные кейсы - выполни /getMoreCases** Необходимо выполнять до отправки результата!

*** Если хочешь опубликовать текущие кейсы комментарием в тикет в Jira - выполни /sendToJira**

*** Если хочешь отправить текущие результаты генерации в другой тикет - передай номер тикета, куда отправить кейсы в формате send ERS-XXXX, после чего выполни /sendToJira**

21:53

После успешной публикации кейсов в комментарии придет отбивка, что кейсы опубликованы, а также в задачу проставится метка `ctt_gpt`.

/getcasewiki - генерация тестового кейса по сценарию на вики по переданному id страницы. (для генерации тестовых кейсов подходят страницы со сценариями от аналитики, например, 2. Получение кнопок-ссылок для блока новостроек в карточке застройщика/ГК, страницы без сценариев либо не по формату - не обрабатываются)

Описание поддерживаемых форматов страниц - Требования к шаблону оформления аналитики для генерации тестовых кейсов по ним через CTGptBot

После вызова данной функции необходимо сделать следующее:

- 1) Выбрать инструмент для генерации тестового кейса (модель gpt):

- YandexGPT (работает лучше всего)
- ChatGPT 3.5 Turbo (в функциональности генерации через сценарии - зависит, лучше не использовать)
- SberGigachat (странно генерирует кейсы, но надо тестировать)

 Write a message...



YaGPT

ChatGPT 3.5 Turbo

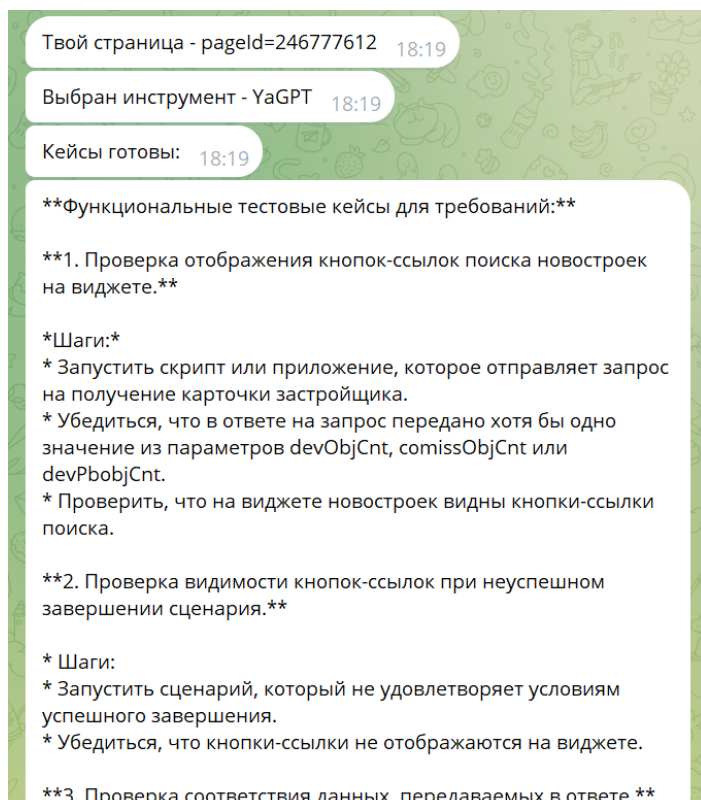
Sber Gigachat

2) Необходимо передать Wiki API Token, чтобы вычитать сценарий, в формате WikiToken {MyToken}. Пример: WikiToken abcd123. Как получить токен можно почитать тут - [Получение api token в Jira/Wiki и работа со swagger для генерации кейсов](#)

Токен живет в системе до следующего календарного дня, поэтому при повторных вызовах метода передавать токен будет не нужно.

3) Передать id страницы в Wiki в формате pageId={id}, пример - pageId=246777612 (вытаскиваем из query params URL страницы на вики)

Как кейсы будут сгенерированы, придет сообщение с готовыми кейсами:



Твоя страница - pageId=246777612 18:19

Выбран инструмент - YaGPT 18:19

Кейсы готовы: 18:19

****Функциональные тестовые кейсы для требований:****

****1. Проверка отображения кнопок-ссылок поиска новостроек на виджете.****

Шаги:

- * Запустить скрипт или приложение, которое отправляет запрос на получение карточки застройщика.
- * Убедиться, что в ответе на запрос передано хотя бы одно значение из параметров devObjCnt, comissObjCnt или devPbobjCnt.
- * Проверить, что на виджете новостроек видны кнопки-ссылки поиска.

****2. Проверка видимости кнопок-ссылок при неуспешном завершении сценария.****

*** Шаги:**

- * Запустить сценарий, который не удовлетворяет условиям успешного завершения.
- * Убедиться, что кнопки-ссылки не отображаются на виджете.

****3. Проверка соответствия данных, передаваемых в ответе.****

Дальше будут предложены следующие варианты:

- Сгенерировать еще дополнительные кейсы (доступно только при выборе yaGPT), позволяет получить еще больше кейсов по переданной ранее информации (/getMoreCases). Необходимо выполнять до отправки результата (/sendToWiki), после генерации дополнительных кейсов будет возможность отправить общий набор (его можно вызывать только сразу после получения сгенерированных кейсов, в другой момент вызов данного метода не доступен)
- Опубликовать текущие сгенерированные кейсы в статью на Wiki в комментарии через метод /sendToWiki (его можно вызывать только сразу после получения сгенерированных кейсов/дополнительно сгенерированных кейсов, в другой момент вызов данного метода не доступен)

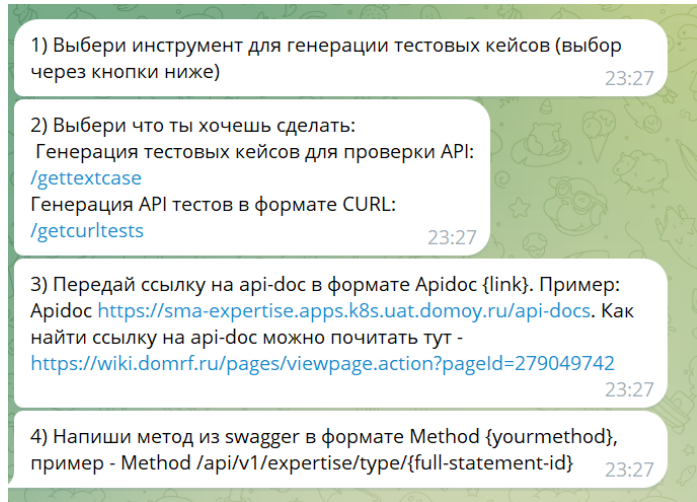
*** Если хочешь сгенерировать дополнительные кейсы - выполни `/getMoreCases` Необходимо выполнять до отправки результата!**

*** Если хочешь опубликовать текущие кейсы комментарием в статью на Wiki выполни `/sendToWiki`**

21:59

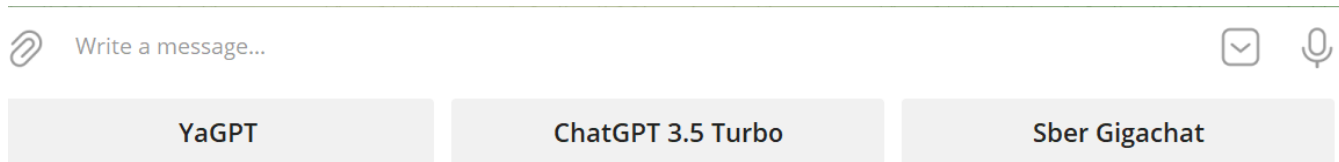
После успешной публикации кейсов в комментарии придет отбивка, что кейсы опубликованы, а также в задачу проставится метка **ctt_gpt**.

/getcaseapi - генерация тестового кейса для API по swagger и нужному методу



После вызова данной функции необходимо сделать следующее:

- 1) Выбрать инструмент для генерации тестового кейса (модель gpt):
 - YandexGPT (работает лучше всего)
 - ChatGPT 3.5 Turbo (**Возможно зависит**)
 - SberGigachat (странно генерирует кейсы, но надо тестировать)



- 2) Выбрать одну из двух функций:

/gettextcase - сгенерирует текстовое описание кейсов для проверки API метода

/getcurltests - сгенерирует API тесты в формате CURL (которые можно в 2 клика импортировать в Postman), описание как это сделать [Как импортировать CURL запрос в Postman \(из бота CTGptBot\)](#)

3) Необходимо передать ссылку на api-doc в формате Apidoc {link}. Пример: Apidoc <https://sma-expertise.apps.k8s.uat.domoy.ru/api-docs>. Как найти ссылку на api-doc можно почитать тут - <https://wiki.domrf.ru/pages/viewpage.action?pageId=279049742>

Сетевая связность у бота только до адресов UAT контура, поэтому работать ссылки на Apidoc будут только в домене uat.domoy.ru / [fap](https://fap.domoy.ru)

4) Передать метод из swagger, для которого необходимо сгенерировать тестовые кейсы в формате Method {yourmethod}, пример - Method /api/v1/expertise/type/{full-statement-id} (**копируем из swagger в точности как там**)

Как кейсы будут сгенерированы, придет сообщение с готовыми кейсами:

Ссылка на Method принята 21:40

Выбран инструмент - YaGPT 21:40

Кейсы готовы: 21:40

****Функциональные тестовые кейсы для API по представленной спецификации:****

****Критерий 1. Проверка успешного обновления типа запроса на получение заключения экспертизы.****

Шаги теста:

1. Проверить, что API доступен.
2. Отправить POST-запрос на обновление типа запроса с правильными параметрами.
3. Проверить ответ API: код 200, в ответе содержится информация об успешном обновлении типа запроса.

Также в случае, если использовалась модель yaGPT, можно вызвать метод `/getMoreCases`, чтобы получить больше кейсов по переданной информации

В случае, если была выбрана функция `/getcurltests` - придет сообщение с возможностью выбора дополнительной функции:

`/getjavatests` - генерация API тестов на Java + RestAssured (инструкция, как пользоваться сгенерированным кодом будет предоставлена позже)

`/getpythontests` - генерация API тестов на Python + Requests (инструкция, как пользоваться сгенерированным кодом будет предоставлена позже)

Можно трансформировать curl тесты в тесты на Python и Java,
для этого выбери:

`/getjavatests`
`/getpythontests`

23:28

В результате придет код API тестов на Python:

Вот как можно преобразовать указанные запросы в API-тесты на Python с использованием библиотеки Requests:

1. **Запрос с корректным идентификатором и данными:**

```
python
import requests

url = "https://sma-expertise.apps.k8s.uat.domoy.ru/api/v1/expertise/files-category/"

payload = {
    "files": [
        {
            "file_name": "example.pdf",
            "category": "Документация"
        }
    ],
    "signature_details": {
        "signed_by": "Иванов Иван Иванович",
        "date": "2023-01-01"
    }
}

headers = {"Content-Type": "application/json"}

response = requests.post(url, json=payload, headers=headers)
assert response.status_code == 200
```

/getPageObject - генерация описания объектов(xpath - локаторов) html-фрагмента страницы для АТ по переданным данным (через wiki)

Что нужно:

- 1) Создать страницу на wiki, чтобы в название статьи входило сочетание - "Generate", либо взять существующую тестовую статью - [Generate](#).
[Тестовый пример](#)
- 2) Добавить html фрагмента, который мы хотим покрыть локаторами, например:

The screenshot displays a web browser window with the URL `portal.apps.k8s.uat.domoy.ru`. The page is a real estate portal titled "Инвестиционный калькулятор". It features a search bar with filters for "Населенный пункт", "Статус дома", "Класс жилья", and "Срок ввода до". Below the search bar, there is a "Найти" button. The page also includes a section for "Полезные сервисы" with links to "Найти новостройку", "Купить квартиру", "Узнать о застройщике", and "Сравнить новостройки". The DevTools console is open, showing the HTML structure of the page. A specific block of HTML is highlighted in the DOM tree, showing a search bar with filters and a "Найти" button. The console also shows the "Overlay display settings" and "Grid overlays" sections.

Далее копируем html данного блока:



Далее вставляем ее в нужную статью в тело:

[ЦТ. Тестирование](#) / [Страницы](#) / ... / [Прочее \(AT\)](#) / [Generate. Тестовый пример](#)

Generate. Тестовый пример

[illegible]

3) Вызываем `/createPageObject`

1) Выбираем инструмент "yagpt"

2) Передаем Wiki API Token, чтобы вычитать сценарий, в формате WikiToken {MyToken}. Пример: WikiToken abcd123. Как получить токен можно почитать тут - [Получение api token в Jira/Wiki и работа со swagger для генерации кейсов](#)

Токен живет в системе до следующего календарного дня, поэтому при повторных вызовах метода передавать токен будет не нужно.

3) Передаем prompt в формате Prompt {prompt} (на период отладки данного метода и подбора лучшего варианта prompt для данной задачи). На текущий момент рекомендуется использовать prompt вида: **"Prompt Составь локаторы для полей: фео, телефон, емэйл, ИНН, должность по xpath по html"** с перечислением названий полей, по которым необходимо составить локаторы.

3) Передать id страницы в Wiki в формате pageid={id}, пример - pageid=246777612 (вытаскиваем из query params URL страницы на вики)

4) Описание xpath готово

5) Дальше будет предложено опубликовать сгенерированные xpath в статью Wiki в комментарии через метод /sendToWiki (**его можно вызывать только сразу после получения результатов, в другой момент вызов данного метода не доступен**):