

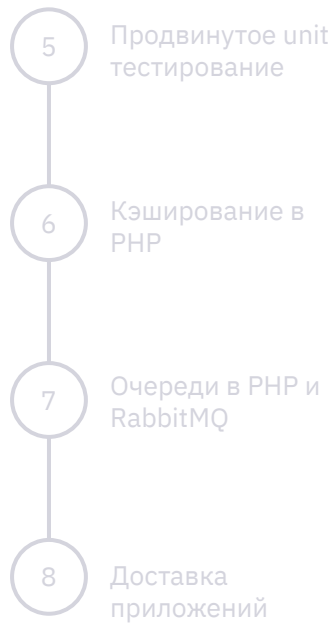
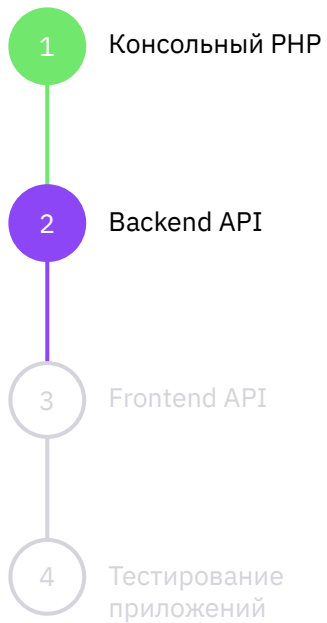
Backend API

Урок 2










План курса



WELCOME



Что будет на уроке сегодня

-  Викторина, которая построена на основании реальных вопросов, которые задают на собеседовании
-  Имитация работы выполнения заданий от тимлида
-  Опыт получения ТЗ от тимлида
-  Создание бота в Telegram
-  Реализация отправки и получения запросов к Telegram



Викторина

Построена на основании реальных вопросов,
которые задают на собеседовании



Преподаватель демонстрирует вопросы викторины, зачитывает их, а студенты пишут ответы в чате



1. Как вы можете объяснить, что такое API? Для чего он используется?

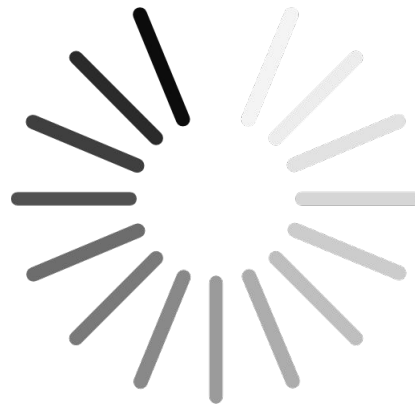
Вопрос без вариантов ответа



Подсказка: загляните в конспект



Совет: напишите ответ в чат или проговорите его





2. Какие методы HTTP существуют?

1. GET

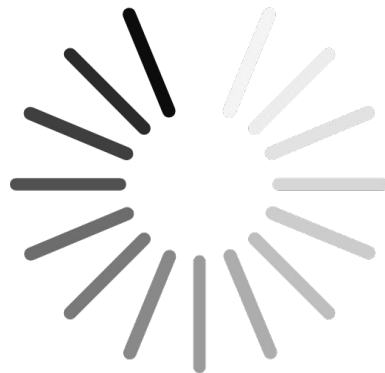
2. POST

3. PUT

4. DELETE

Дополнительные вопросы:

- Каким методом вы будете получать, к примеру, состав заказа в интернет магазине?
- А каким будете отправлять или редактировать заказ?



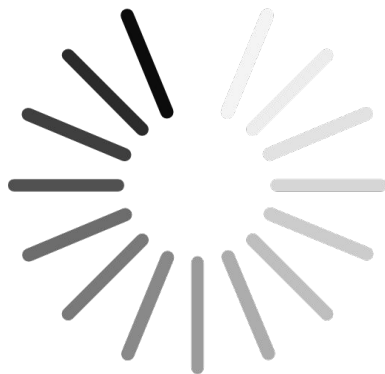


3. В чем ключевое отличие между GET и POST запросом?

1. GET используется для получения данных с сервера, а POST – для отправки данных на сервер
2. GET-запросы имеют ограничение на размер данных, которые можно передать. POST-запросы не имеют такого ограничения
3. GET-запросы могут быть легко кэшированы, а POST-запросы обычно не кэшируются
4. GET-запросы используются для безопасных операций, а POST-запросы – для более рискованных операций

Дополнительный вопрос:

В чем особенность передачи параметров в этих запросах?





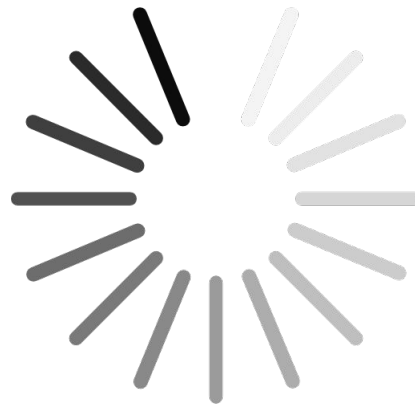
4. Какие виды (классификации) API существуют?

1. по составу участников

2. по состоянию

3. по доступности

4. по времени





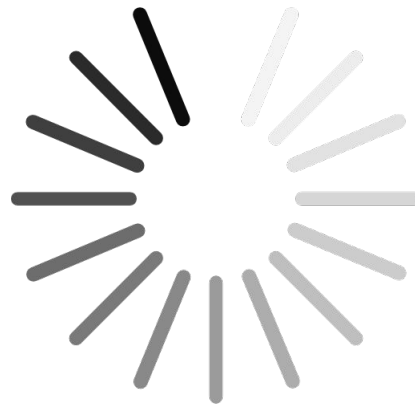
5. Какие форматы данных, используемых в веб-API, существуют?

1. JSON

2. HTML

3. CSS

4. XML





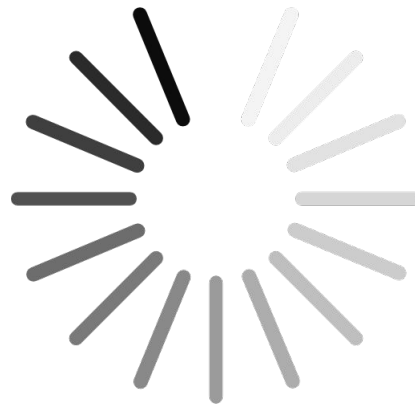
6. Какими методами в PHP мы можем преобразовать JSON в массив и обратно?

1. `Никак()`

2. `json_decode()`

3. `json_encode()`

4. `json_HTML()`





7. В чем особенность протокола ProtoBuf?

**В каких случаях стоит задуматься об
использовании этого протокола?**

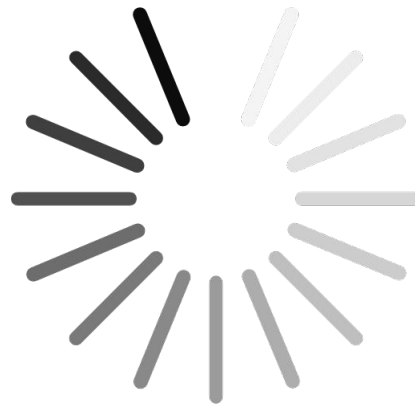
Вопрос без вариантов ответа



Подсказка: загляните в конспект



Совет: напишите ответ в
чат или проговорите его





8. Какие инструменты для тестирования API существуют?

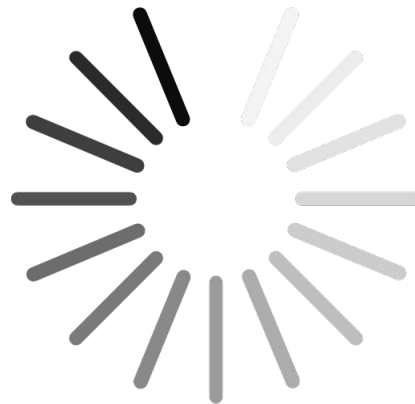
1. Postman

2. SoapUI

3. Katalon Studio

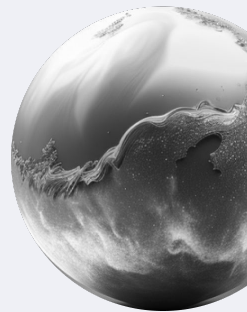
4. Swagger UI

5. Confluence





Будьте внимательны, также на собеседовании
вас спросят, с какими инструментами
тестирования API доводилось работать





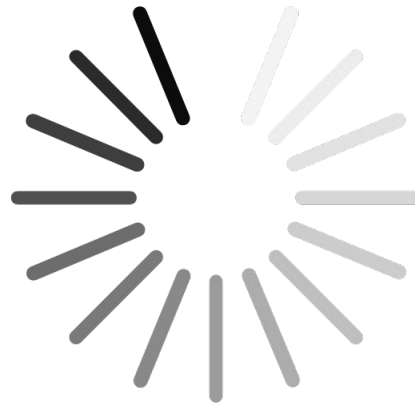
9. Какие инструменты документирования API существуют?

1. Redoc

2. Swagger/OpenAPI

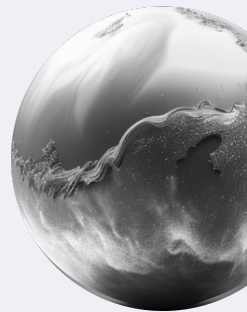
3. Postman

4. Confluence





Будьте внимательны, также на собеседовании
вас спросят, какие инструменты
документирования API видели в реальных
проектах?
Какие доводилось писать самостоятельно?



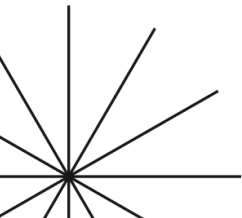


Перечень вопросов, которые также могут также задать:

1. Что такое технология RPC? В каких случаях используется, какие особенности и проблемы? С какими форматами данных (json, xml, protobuf) можно использовать RPC?
2. Что такое SOAP и какие у него особенности? Какой формат данных использует?
3. Что такое WebSocket? Опишите сценарий его работе, к примеру, при постоянном обновлении счетчика сообщений в браузере.
4. Что такое REST? Что такое RESTful API? Какие у него требования? Какие лучшие практики при проектировании RESTful API вы помните? Какие запросы в RESTful API идемпотентные и безопасные, а какие нет?
5. В чем разница между аутентификацией и авторизацией? Опишите основные инструменты для аутентификации и/или авторизации (api ключи, JWT, OAuth, Bearer Tokens). Какие из них поддерживают авторизацию, а какие – аутентификацию?



Рекомендую на них ответить самостоятельно
Ответы к ним вы найдете в лекции 🙌





Разбор домашнего задания





Вопрос

Опишите результат взаимодействия с `systemctl`.

Как вы считаете, в каких задачах использование демонов лучше, чем использование `crontab`?





Практика



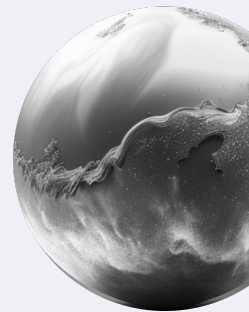


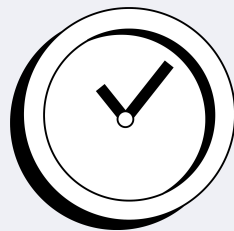
Вы написали логику обработку событий, и вашему тимлиду на глаза попался заголовок:

“Количество пользователей в телеграм перевалило за 900 миллионов”.

“Хмм” – подумал ваш тимлид и решил, что лучше делать бота с помощью Telegram.

Тем более большинство коллег им пользуются!





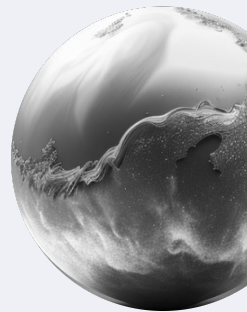
15 мин.



База нашего бота-напоминалки уже есть, осталось разобраться с API Telegram.

Для этого мы переходим в [гайд](#) и смотрим как общаться с Telegram API.

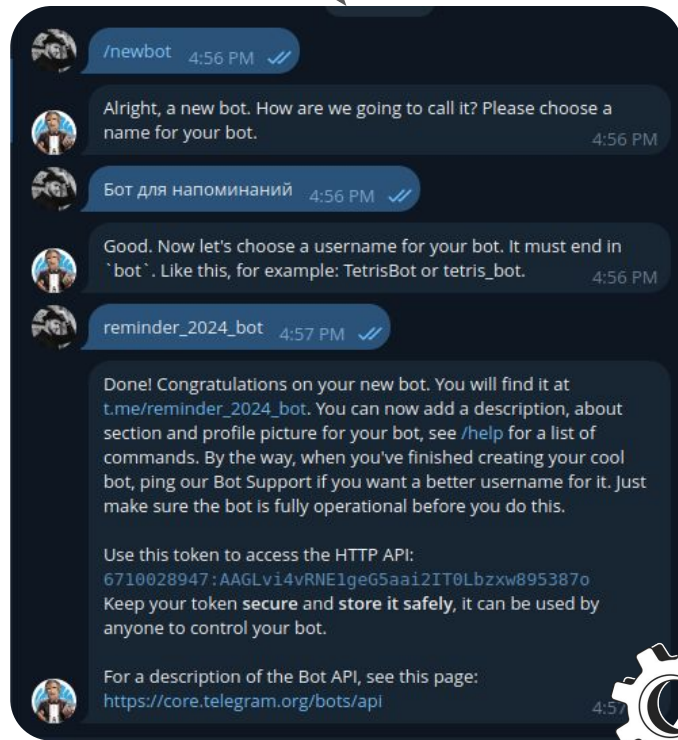
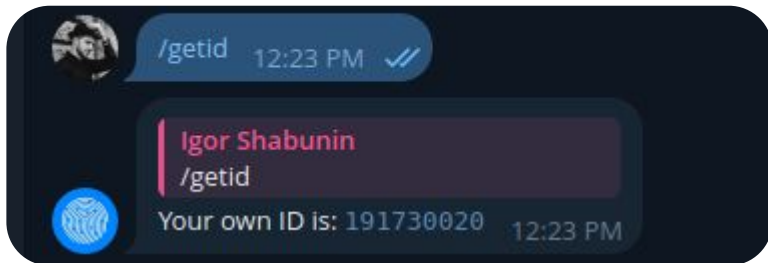
Тут нам нужно реализовать **2 метода**:
отправки и получения сообщений –
[getUpdates](#) и [sendMessage](#)





Задание 1. Подготовка ТЗ к интеграции с Telegram

1. Создать новый бот при помощи <https://t.me/BotFather>
2. Сохраняем полученный токен, переходим в наш бот и пишем (что угодно)
3. Узнать свой идентификатор в telegram. Для этого переходим в [Telegram: Contact @myidbot](#) и получаем идентификатор



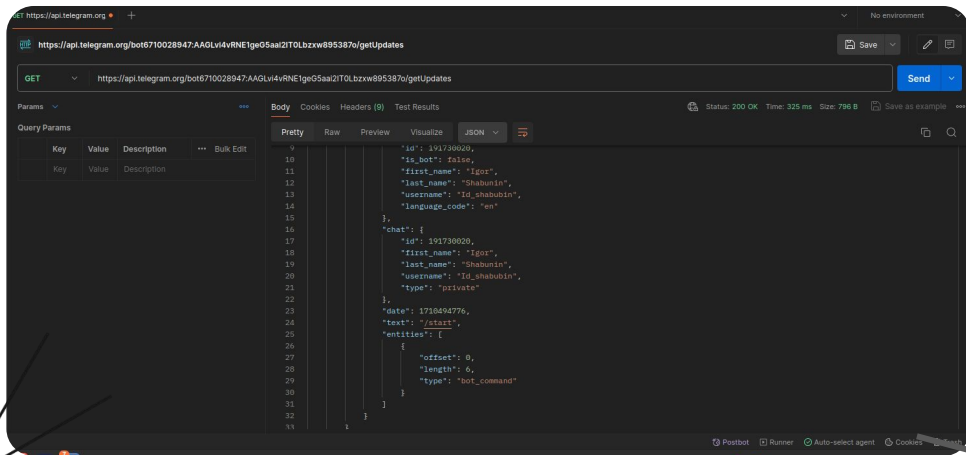


Задание 1. Подготовка ТЗ к интеграции с Telegram

4. Переходим в [postman](#)

5. Начнем с получения сообщений. Для этого нам понадобится метод [getUpdates](#). [Tyt](#) более интерактивная документация.

6. Мы можем получить сообщения просто обратившись get запросом по адресу <https://api.telegram.org/bot{token}/getUpdates>



Также мы можем использовать get параметр **offset**. В него мы должны указать значение **на 1 больше** предыдущего полученного идентификатора сообщения (**параметр update_id**).

Это будет полезно, когда мы будем реализовать получение сообщений в нашем скрипте





GET https://api.telegram.org

No environment

https://api.telegram.org/bot6710028947:AAGLvI4vRNE1geG5aal2IT0Lbzxw895387o/getUpdates

Save



GET https://api.telegram.org/bot6710028947:AAGLvI4vRNE1geG5aal2IT0Lbzxw895387o/getUpdates

Send

Params

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (9) Test Results

Status: 200 OK Time: 325 ms Size: 796 B

Save as example

Pretty

Raw

Preview

Visualize

JSON



```
9      "id": 191730020,
10      "is_bot": false,
11      "first_name": "Igor",
12      "last_name": "Shabunin",
13      "username": "Id_shabubin",
14      "language_code": "en"
15    },
16    "chat": {
17      "id": 191730020,
18      "first_name": "Igor",
19      "last_name": "Shabunin",
20      "username": "Id_shabubin",
21      "type": "private"
22    },
23    "date": 1710494776,
24    "text": "/start",
25    "entities": [
26      {
27        "offset": 0,
28        "length": 6,
29        "type": "bot_command"
30      }
31    ]
32  }
33 }
```

Postbot

Runner

Auto-select agent

Cookies

Trash



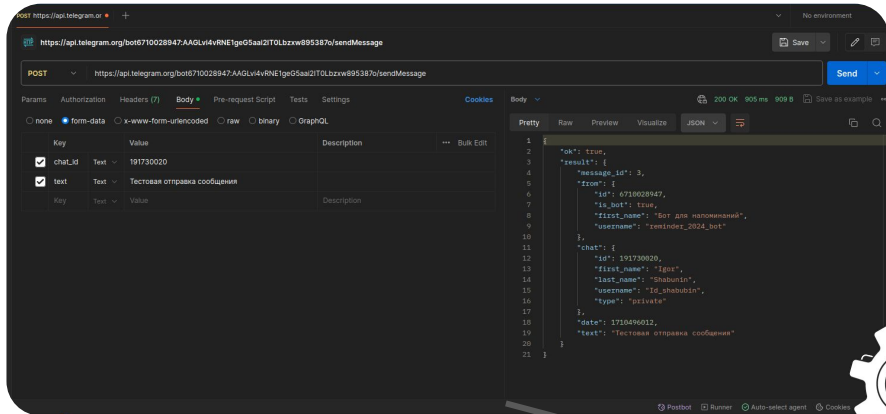
Задание 1. Подготовка ТЗ к интеграции с Telegram

7. Для отправки сообщения мы можем использовать метод [sendMessage](#). [Тут](#) более интерактивная документация.

Для этого нам нужно отправить POST запрос на **<https://api.telegram.org/bot{token}/sendMessage>**, в котором передать 2 параметра:

- ➔ chat_id(идентификатор пользователя, который получили в п. 3)
- ➔ text(наш текст сообщения).

Вот так это можно сделать в postman:





POST https://api.telegram.or +

No environment

https://api.telegram.org/bot6710028947:AAGLvI4vRNE1geG5aal2IT0Lbzxw895387o/sendMessage

Save



POST https://api.telegram.org/bot6710028947:AAGLvI4vRNE1geG5aal2IT0Lbzxw895387o/sendMessage

Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Cookies

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	Key		Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	chat_id	Text	191730020			
<input checked="" type="checkbox"/>	text	Text	Тестовая отправка сообщения			
	Key	Text	Value	Description		

Body

200 OK 905 ms 909 B Save as example

Pretty

Raw

Preview

Visualize

JSON

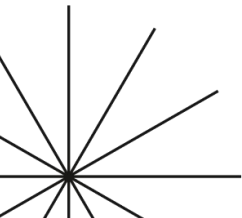
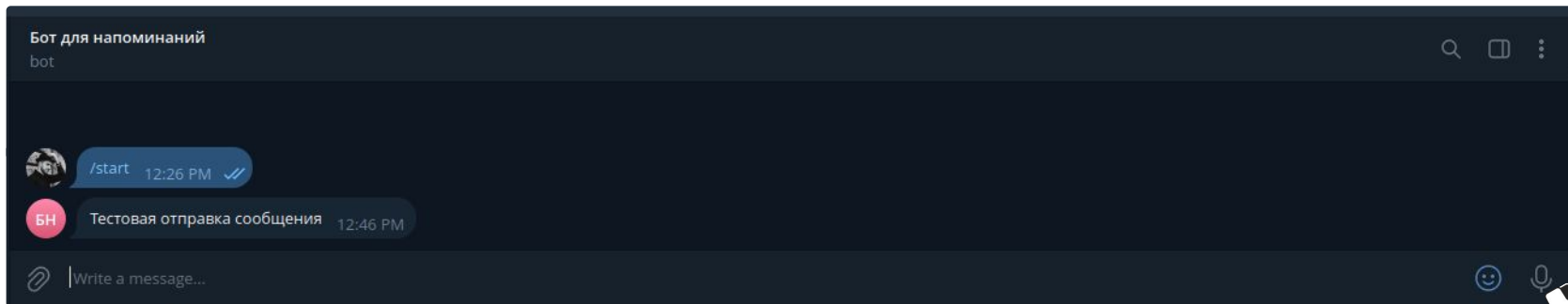



```
1 {
2   "ok": true,
3   "result": {
4     "message_id": 3,
5     "from": {
6       "id": 6710028947,
7       "is_bot": true,
8       "first_name": "Бот для напоминаний",
9       "username": "reminder_2024_bot"
10    },
11    "chat": {
12      "id": 191730020,
13      "first_name": "Igor",
14      "last_name": "Shabunin",
15      "username": "Id_shabubin",
16      "type": "private"
17    },
18    "date": 1710496012,
19    "text": "Тестовая отправка сообщения"
20  }
21 }
```



Задание 1. Подготовка ТЗ к интеграции с Telegram

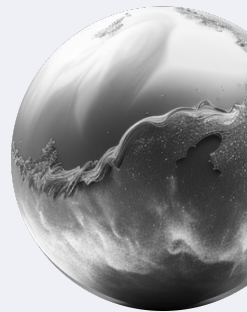
После чего мы видим наше сообщение:





ТЗ от тимлида вы взяли в работу!

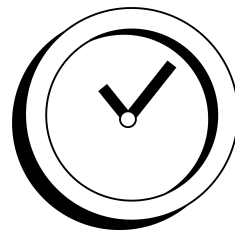
Теперь вы начинаете заниматься
реализацией этого ТЗ (задание № 2)





Задание 2. Реализация отправки сообщений в Telegram

1. Создать класс **TelegramApi**, который будет отвечать за отправку и получение сообщений. Создать 2 метода: **getMessages** и **sendMessage**
2. В класс **EventSender** (который мы создали на предыдущем уроке) добавить вызов метода `sendMessage`
3. Метод **sendMessage** обращается по пути <https://api.telegram.org/bot{token}/sendMessage>, передает **chat_id** и **text**.



15 мин.



Что нужно для
решения задачи?





Задание 2. Реализация отправки сообщений в Telegram

Алгоритм:

1. Добавьте в `.env.example` и `.env` токен Telegram

```
1 TELEGRAM_TOKEN=
```



Задание 2. Реализация отправки сообщений в Telegram

2. Реализуйте метод sendMessage() из интерфейса

```
1 <?php
2
3 namespace App\Telegram;
4
5 interface TelegramApi
6 {
7     public function __construct(string $token);
8
9     /**
10      * @return TelegramMessageDto[]
11      */
12     public function getMessages(int $offset): array;
13
14     public function sendMessage(string $chatId, string $text);
15 }
16
```

Для реализации нужно вспомнить предыдущее задание.

Каким образом мы отправили сообщение в postman?

Такой же алгоритм и нужно повторить.



Задание 2. Реализация отправки сообщений в Telegram

В качестве отправителя можно использовать обычный curl:

```
1 $ch = curl_init($url);
2 $jsonData = json_encode($data);
3 curl_setopt($ch, CURLOPT_POST, true); // Указывает, что отправляется
   POST запрос
4 curl_setopt($ch, CURLOPT_POSTFIELDS, $jsonData); // Прикрепляет
   данные
5 curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-
   Type:application/json')); // Устанавливает тип контента
   application/json
6 curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
7 curl_exec($ch);
8 curl_close($ch);
9
```



Задание 2. Реализация отправки сообщений в Telegram

3. В класс **EventSender** (который мы создали на предыдущем уроке) добавить вызов метода **sendMessage**
4. С помощью консольной команды **php runner -c save_event** из предыдущего урока добавьте в базу данных событие, которое будет отправлять в чат с вашим ID любое сообщение, например:

```
1 php runner -c save_event --name 'Тестовая отправка' --receiver {Ваш
  ID, полученный от myidbot} --text 'Текст тестовой отправки
  сообщения' --cron '* * * * *'
```

5. И проверьте, что сообщение было доставлено вам с помощью команды из прошлого урока:

```
1 php runner -c handle_events
```

Вам в telegram должно уйти сообщение с текстом *'Текст тестовой отправки сообщения'*.

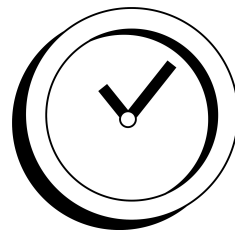


Задание 3. Реализация получения сообщений из Telegram

1. Метод `getMessages` обращается по пути:
`https://api.telegram.org/bot{token}/getUpdates?timeout=1&offset={ID последнего полученного сообщения + 1}`.

Сам экземпляр класса `TelegramApi` должен хранить `offset` либо метод `getMessages` должен принимать на вход параметр `offset`.

2. Для проверки результата `getMessages` можно создать **`App\Commands\TgMessagesCommand`**, в котором будем вызывать **`getMessages`** и выводить в консоль.



15 мин.



Что нужно для
решения задачи?





Задание 3. Реализация получения сообщений из Telegram. Алгоритм

1. Создайте новую команду `App\Commands\TgMessagesCommand`.

Пусть она выводит в консоль все сообщения из чата при вызове команды

`php runner -c tg_messages`

2. Реализуйте метод **`getMessages(int $offset)`** класса `TelegramApiImpl`
3. В **`TgMessagesCommand`** выведите в консоль результат вызова `getMessages()`.



Домашнее задание





Домашнее задание

Задание

1. Сделайте новую ветку из той, которую мы создали на прошлом уроке. Это нужно для того, чтобы мы могли работать с кодом из прошлого урока.
2. Загрузите весь код из сегодняшнего урока в Git в новую ветку, создайте новый pull request. Пришлите на проверку ссылку на pull request.
3. Сегодня мы работали с вами с токеном Telegram. **Какой тип аутентификации мы использовали?**
4. ✖ На занятии мы создали класс **TgMessagesCommand**. Сделайте так, чтобы он мог общаться с пользователем и запрашивать данные для сохранения расписания.

```
21.06.23 Bot: /start
21.06.23 Bot: Укажите название события
21.06.23 Пользователь: Напоминание о очередной еженедельной
встрече по поводу цвета кнопок на сайте
21.06.23 Bot: Укажите ID пользователя
21.06.23 Пользователь: 78479879843
21.06.23 Bot: Укажите текст напоминания
21.06.23 Пользователь: Очень важная встреча по поводу кнопок на
сайте пройдет в 14:00. Очень важно, чтобы ты был!
21.06.23 Bot: Укажите в какие дни Вам нужно отправлять сообщения
21.06.23 Пользователь: 0 0 * * 1
21.06.23 Bot: Я записал Ваше событие. Для нового события введите /
start
01.06.24 Bot: Очень важная встреча по поводу кнопок на сайте
пройдет в 14:00. Очень важно, чтобы ты был!
```

16:22 ✓

На проверку отправляйте:

- ссылку на pull request в вашем репозитории с домашним заданием

✖ – дополнительное задание 💪

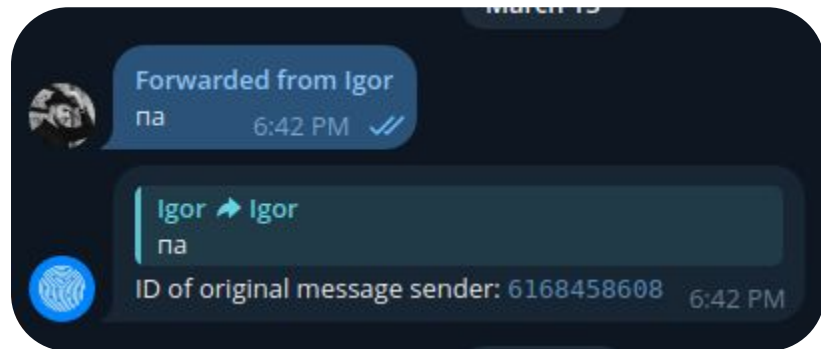
Домашнее задание



Для решения доп. задания № 4 нужно:

1. Создать из **TgMessagesCommand** демона
2. Хранить всю историю сообщений, пришедших за время работы скрипта.
3. Хранить **offset** получения сообщений в TgMessagesCommand

Если пришло новое сообщение, то оно сохраняется в историю сообщений, после чего определяется какое сообщение мы должны отправить пользователю. Например, если пользователь отправил “/start” мы должны отправить “Укажите название события”, если пользователь отправил следующее сообщение после “/start” мы должны отправить “Укажите ID пользователя” и так далее. Если мы получили 4 сообщения от пользователя (название события, ID пользователя, Текст напоминания и cron расписание), то мы сохраняем их в базу, а пользователю отправляем “Я записал Ваше событие. Для нового события введите /start”. Отправка сообщений пользователю происходит так же, как мы это делали в 1-м задании, когда отправляли сообщения пользователю по расписанию.



* Если вы хотите отправить сообщение не себе, то получатель должен начать диалог с вашим ботом. Узнать идентификатор нужного вам пользователя вы можете, переслав боту **@myidbot** любое сообщение этого пользователя








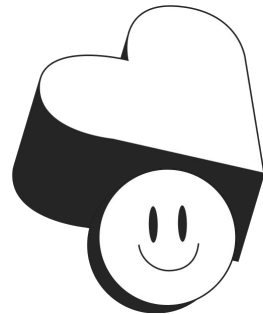
Подведем итоги





Подведение итогов

-  потренировались отвечать на вопросы про API на собеседованиях
-  научились работать в команде, понимать требования и выполнять задачи, поставленные тимлидом
-  научились разбивать задачу на более мелкие подзадачи, чтобы легче было управлять проектом и следить за прогрессом
-  сделали новый уверенный шаг в создании бота — создали интеграцию с Telegram
-  научились работать с Telegram API и Postman





Спасибо за внимание