Данный бот предназначен для вывода заранее заготовленных сообщений пользователю.

С ботом работают три категории людей: программист, администратор, пользователь.

Администратор через веб приложение, создание на базе Django, может добавляь, редактировать и удалять вопросы, а также работать с ответами пользователя.

Поля формы ввода вопросов в админке.

|  |  |  |  |
| --- | --- | --- | --- |
| Поле ввода текста сообщения (text\_message).  Текстовое поле. | Номер предыдущего сообщения (parent\_id). Текстовое поле. | Ответ на предыдущее сообщение (answer).  Текстовое поле. | Указываем нужно ли сохранить ответ (write\_ answer).  CheckBox |

Поля формы для работы с ответами пользователя

|  |  |
| --- | --- |
| Вопрос 1 | Ответ на вопрос 1 |
| Вопрос 2 | Ответ на вопрос 2 |

?????? ПОЛЯ ДЛЯ РАБОТЫ С ДАННЫМИ ПОЛЬЗОВАТЕЛЯ

Диалог выглядит как древовидная структура с условными переходами.

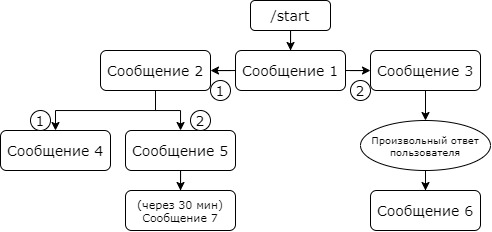


Таблица с сообщениями в БД (message\_table)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| pk\_message | text\_message | parent\_id | answer | write\_answer |
| 1 | Сообщение 1 | None | None | False |
| 2 | Сообщение 2 | 1 | 1 | True |
| 3 | Сообщение 3 | 1 | 2 | True |
| 4 | Сообщение 4 | 2 | 1 | False |
| 5 | Сообщение 5 | 2 | 2 | False |
| 6 | Сообщение 6 | 3 | None | False |
| 7 | Сообщение 7 | 5 | None | False |

Таблица с ответами БД (answer\_table)

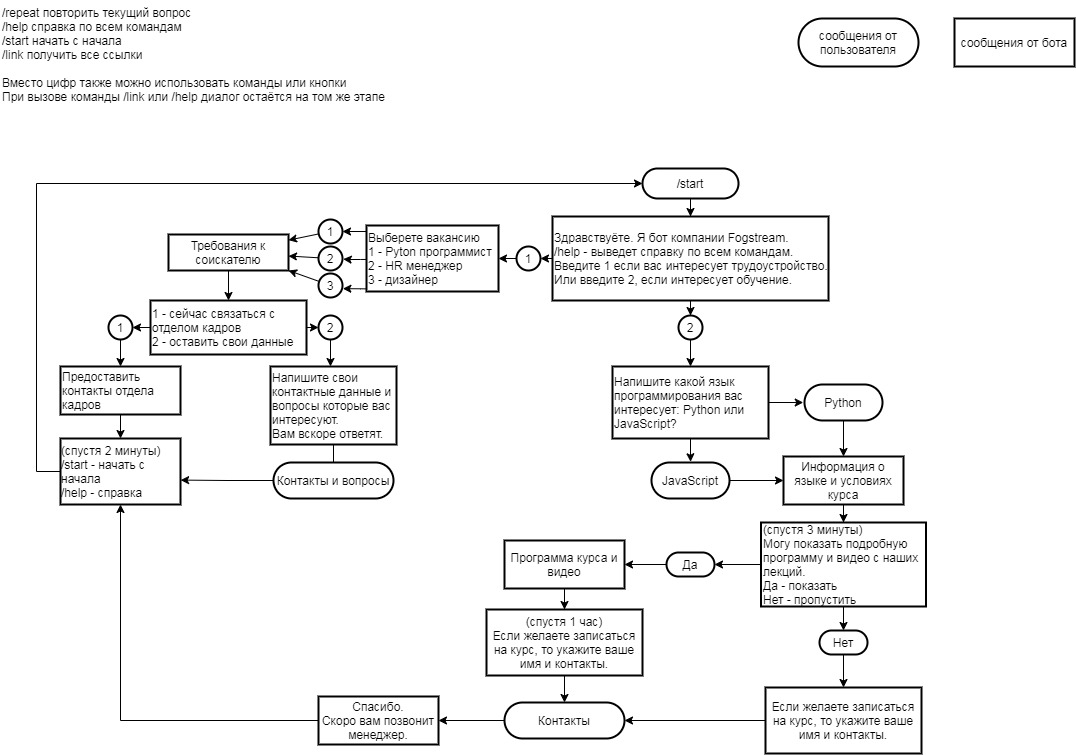
|  |  |  |
| --- | --- | --- |
| message\_table. pk\_message | text\_answer | user\_table .user\_id |
| 2 | Ответ А | 1 |
| 3 | Ответ Б | 1 |

Таблица с информацией о пользователе (user\_table)

|  |  |  |  |
| --- | --- | --- | --- |
| user\_id | user\_name | first\_name | last\_name |
| 1 | Жора | Евгений | Петров |
| 2 | Хацкер | Алексей | Иванов |

Пользователь заходит в бот, получает от него сообщения и отвечает на эти сообщения.

Сценарий диалога с ботом.



Программист разрабатывает и настраивает бота на базе фреймворка Django и Telegram Bot API. Для хранения сообщений, информации о пользователе, ответов пользователя используется база данных.

Описание работы бота.

Пользователь вводит /start

Бот обрабатывает эту команду обработчиком - @bot.message\_handler(commands=['start'])

Из сообщения от пользователя берёт

from\_user.id

from\_user.username

from\_user.last\_name

from\_user.first\_name

и делает запрос серверу. Сервер регистрирует пользователя в БД, если пользователя с таким from\_user.id ещё нет.

Если есть такой from\_user.id есть. Например, пользователь запустил /start и решил пройти по другой ветке диалога. Тогда новые ответы перезапишут старые или добавятся те, которых не было.

Сервер даёт в ответе первое сообщение боту

bot\_mes = {pk\_message: 1, text\_message: Сообщение 1, parent\_id: None, answer: 1, write\_answer: False}.

bot\_mes может быть атрибутом класса, например.

Бот выводит его пользователю bot.send\_message(message.chat.id, " Сообщение 1")

Если пользователь вводит ответ, ответ пользователя это текст.

Бот отслеживает текст обработчиком @bot.message\_handler(content\_types=['text'])

Если пользователь ввел 1

if message.text == 1

Бот берёт данные из ответа пользователя, переменной bot\_mes и формирует запрос на сервер.

{mes\_number: bot\_mes[pk\_message] , answer\_user: message.text, write\_answer: bot\_mes[write\_answer]},

содержимое { mes\_number: 1, answer\_user: 1, write\_answer: False}

Сервер приняв запрос, не сохраняет answer\_user, так как write\_answer == False.

На сервере выбирается сообщение из таблицы message\_table, у которого message\_table.parent\_id равен номеру последнего сообщения и message\_table.answer равен ответу пользователя

mes\_number == parent\_id and answer\_user == answer.

Это сообщение отправляется боту, тот его принимает

bot\_mes = {pk\_message: 2, text\_message: Сообщение 2, parent\_id: 1, answer: 1, write\_answer: True}

Бот выводит его пользователю

bot.send\_message(message.chat.id, " Сообщение 2")

Если пользователь ввел 2

if message.text == 2

Алгоритм обработки ответа пользователя будет тот же, для данного случая на ответ от пользователя, сервер пришлёт "Сообщение 3" и бот его выведет.

Аналогично будут обрабатываться все заранее запрограммированные ответы.

ТОЧНОЕ РЕШЕНИЕ ЕЩЁ НЕ ОПРЕДЕЛЕНО

Если ответ заранее точно не определён, ветка else. Например “Сообщение 3” запрашивает номер телефона.

Пользователь пишет номер: message.text = 89243456789. На сервер уходит запрос { mes\_number: 3, answer\_user: 89243456789, write\_answer: True}

Сервер приняв запрос, сохраняет answer\_user, так как write\_answer == True.

Но при поиске следующей строки в БД проверяем условие

mes\_number == parent\_id and answer\_user == answer

mes\_number == message\_table.parent\_id сработает

answer\_user == message\_table.answer не сработает

у нас

answer\_user = 89243456789, а message\_table.answer = None

Тут нужно смотреть на сочетание

write\_answer =True

и что

message\_table.parent\_id = 3

в единичном экземпляре, то есть у Сообщения 3 только один потомок

После вывода последних сообщений (4, 6, 7) можно запустить таймер и если никаких команд от пользователя не последует самостоятельно выполнить /start.

appadmin это админка

appserver хранит все что связано с Django REST Framеwork

telegrambot это часть бота

Выводимое сообщение определяется ответом на предыдущее сообщение (условие перехода) и предыдущим сообщением - предком.

Каждое сообщение имеет уникальный id.

номер сообщения

message\_id

номер пользователя

from\_user.id

ник

from\_user.username

имя

from\_user.first\_name

фамилия

from\_user.last\_name

/repeat повторить текущий вопрос

/help справка по всем командам

/start начать с начала

/link получить все ссылки