Лабораторная работа 22

ПвИ

**Задание 01**

1. Создайте Telegram-bot и приложение **22-01**, выполняющее long poll запросы с помощью Telegram API и обеспечивающее echo-поведение.
2. Добавьте функцию подписки на ежедневную рассылку случайного факта. Пользователь может подписаться на рассылку по команде */subscribe*, а отписаться - по команде */unsubscribe*. При подписке пользователь должен быть добавлен в базу данных подписчиков, а каждый день он должен получать новый случайный факт от бота. Используйте пакет для планировки задач [***node-cron***](https://www.npmjs.com/package/node-cron).
3. Добавьте возможность отправки пользователю стикеров в ответ на определенные фразы. Например, при вводе фразы "*привет*", бот должен отправить стикер с приветствием.
4. Создайте команду */weather*, которая будет предоставлять пользователю информацию о погоде (температура, влажность, давление, ветер, время восхода и заката, продолжительность дня) в определенном городе. Для этого можно использовать сторонние API, например, [OpenWeatherMap API](https://openweathermap.org/current).
5. Создайте команду */joke*, которая будет отправлять пользователю случайную шутку.
6. Реализуйте возможность отправки изображений пользователю в ответ на определенные запросы. Например, при запросе */cat* бот должен отправить случайное изображение кота.

**Задание 02**. Ответьте на вопросы

1. Поясните принцип работы LongPoll-сервера?
2. Поясните принцип работы webhook’ов?

### 7. Принцип работы LongPoll-сервера

LongPolling (длинные опросы) — это метод взаимодействия клиента (обычно веб-браузера) с сервером, который позволяет серверу немедленно отправлять обновления клиенту по мере их появления. Рассмотрим принцип работы LongPoll-сервера пошагово:

1. \*\*Инициализация запроса клиента\*\*: Клиент инициирует HTTP-запрос к серверу, запрашивая обновления. Этот запрос может быть синхронным или асинхронным.

2. \*\*Обработка сервером\*\*: Сервер получает запрос от клиента и начинает обработку.

3. \*\*Проверка наличия новых данных\*\*: Вместо того чтобы мгновенно ответить на запрос, сервер проверяет наличие новых данных или обновлений, которые могут быть полезны для клиента.

4. \*\*Ожидание изменений\*\*: Если обновлений нет, сервер не отвечает немедленно. Вместо этого он удерживает соединение открытым (ожидает), пока не произойдут изменения или не истечет определенный таймаут.

5. \*\*Отправка ответа\*\*: Как только происходит изменение (например, появляется новое сообщение или обновление), сервер сразу же отправляет ответ клиенту с этими данными.

6. \*\*Закрытие соединения\*\*: После отправки ответа сервер закрывает текущее соединение с клиентом.

7. \*\*Инициация нового запроса\*\*: Клиент сразу же после получения ответа начинает новый запрос к серверу для получения следующих обновлений.

Преимущества LongPolling включают простоту реализации и совместимость с большинством веб-браузеров, однако у этого метода есть свои недостатки, такие как задержка между запросами, высокое потребление ресурсов сервера и возможные проблемы с масштабируемостью при большом количестве одновременных соединений.

### 8. Принцип работы webhook’ов

Webhook (веб-хук) — это механизм, который позволяет веб-приложению автоматически отправлять определенные данные или события другому веб-приложению по мере их возникновения. Принцип работы webhook'ов можно описать следующим образом:

1. \*\*Настройка веб-хука\*\*: Владелец веб-приложения, которое должно получать уведомления, настраивает URL-адрес webhook'а в своем приложении. Этот URL-адрес обычно предоставляется как эндпоинт для приема данных.

2. \*\*Событие, инициирующее webhook\*\*: В другом веб-приложении (например, сервисе, который отправляет уведомления), происходит событие, которое должно вызвать отправку данных на webhook.

3. \*\*Формирование и отправка запроса\*\*: После возникновения события, приложение формирует HTTP-запрос (обычно POST-запрос) на заранее указанный URL-адрес webhook'а.

4. \*\*Получение и обработка данных\*\*: Веб-приложение, принимающее webhook, получает этот HTTP-запрос с данными о событии. Затем оно обрабатывает эти данные согласно своей логике (например, обновляет базу данных, отправляет уведомления и т. д.).

5. \*\*Ответ\*\*: При необходимости веб-приложение, принимающее webhook, может ответить на запрос, подтверждая получение данных или информируя отправителя о результатах обработки.

Преимущества использования webhook'ов включают возможность мгновенной реакции на события, экономию ресурсов (по сравнению с периодическим опросом), а также более эффективную асинхронную обработку данных. Однако важно учитывать безопасность (например, проверку подлинности запросов) и обрабатывать возможные ошибки или тайм-ауты при использовании webhook'ов.