

**Асинхронные взаимодействия.** Вот некоторые из наиболее распространенных:

HTTP - асинхронные технологии:

- HTTP Callbacks (Webhooks): Webhook представляет собой HTTP POST-запрос, который сервер отправляет на predetermined URL, когда наступает определенное событие. Это позволяет асинхронно получать обратную связь без необходимости постоянного опроса сервера.
  - Сложность использования: Низкая. Webhooks являются относительно простым механизмом, требующим лишь настройки URL-адреса для получения обратных вызовов и обработки входящих запросов.
  - Стоимость использования: Низкая. Использование webhooks обычно связано с низкими затратами, так как данные передаются только при наступлении определенного события.
  - Когда стоит задуматься: Webhooks идеально подходят для ситуаций, когда вам нужно получать уведомления о событиях в реальном времени без необходимости постоянного опроса сервера. Они широко используются в интеграциях API, таких как GitHub, Slack и другие.
- HTTP Polling: Polling - это техника, при которой клиент регулярно отправляет запросы к серверу для получения информации.
  - Сложность использования: Низкая. Требуется лишь настроить интервалы опроса и обрабатывать ответы.
  - Стоимость использования: Средняя-Высокая. Затраты могут увеличиваться, особенно при высокой частоте опроса из-за большого количества запросов.
  - Когда стоит задуматься: Polling может быть полезным, когда информация обновляется редко или когда реальное время не является критическим.
- HTTP Long Polling: Это улучшенная версия polling, при которой сервер удерживает HTTP-запрос и отправляет ответ только после того, как произойдет событие или будет достигнут тайм-аут.
  - Сложность использования: Средняя. Требуется управление долгими запросами и таймаутами на стороне сервера.
  - Стоимость использования: Средняя. Несмотря на то, что затраты на трафик меньше, чем при обычном опросе, удержание соединений может потребовать больше ресурсов.

- Когда стоит задуматься: Long polling стоит рассмотреть, когда важно минимизировать задержку, и обновления могут происходить в любое время, но не требуется двунаправленное соединение в реальном времени, как в случае с WebSockets.

#### Другие технологии:

- WebSockets: Этот протокол позволяет установить двустороннее соединение между клиентом и сервером, которое остается открытым, пока одна из сторон его явно не закроет. Таким образом, клиент и сервер могут обмениваться сообщениями в любое время, что делает его идеальным для реального времени и других асинхронных взаимодействий.
  - Сложность использования: Средняя. Для работы с WebSockets требуется понимание двусторонних связей и управления состояниями соединений. Однако многие языки программирования и фреймворки предоставляют библиотеки, упрощающие работу с этим протоколом.
  - Стоимость использования: Низкая-Средняя. Стоимость зависит от количества активных соединений и объема передаваемых данных. Важно помнить, что удержание открытых соединений может потребовать больше ресурсов.
  - Когда стоит задуматься: WebSockets стоит рассмотреть, если вам нужна возможность обмениваться сообщениями в реальном времени между сервером и клиентом. Идеально подходит для чат-приложений, игр и других интерактивных приложений.
- Server-Sent Events (SSE): SSE - это технология, которая позволяет серверу отправлять обновления клиенту в любое время, без необходимости постоянного опроса со стороны клиента. SSE использует обычные HTTP-соединения, что облегчает его использование и интеграцию.
  - Сложность использования: Низкая. SSE использует обычные HTTP-соединения, что делает его простым в использовании. Единственная сложность может возникнуть с обработкой переподключений.
  - Стоимость использования: Низкая. SSE использует стандартные HTTP-соединения, и поэтому не требует дополнительных инвестиций.

- Когда стоит задуматься: SSE подходит для случаев, когда серверу нужно посылать обновления клиенту без активного запроса со стороны клиента. Это может быть полезно для приложений с потоковыми обновлениями, такими как ленты новостей или живые тикеры
- gRPC (с использованием потоков): gRPC также может поддерживать асинхронные взаимодействия с использованием потоковых вызовов. Это позволяет клиенту и серверу обмениваться несколькими сообщениями в рамках одного вызова.
  - Сложность использования: Средняя-Высокая. gRPC требует понимания модели потоковых вызовов, и, несмотря на наличие библиотек, может потребоваться время для освоения.
  - Стоимость использования: Средняя. gRPC построен на базе HTTP/2 и требует поддержки этого протокола со стороны сервера и клиента.
  - Когда стоит задуматься: gRPC особенно полезен при работе с тяжелыми работами и большими объемами данных, которые требуют эффективного обмена данными и высокой производительности.
- Брокер RabbitMQ с Advanced Message Queuing Protocol (AMQP): AMQP - это стандартный протокол обмена сообщениями, который поддерживает гибкие модели обмена сообщениями, включая точка-точка, публикация-подписка и маршрутизация.
  - Сложность использования: Высокая. AMQP - это более сложный протокол, который предлагает больше гибкости, но требует более глубокого понимания.
  - Стоимость использования: Средняя-Высокая. Стоимость зависит от выбранной системы брокера сообщений и объема обрабатываемых сообщений.
  - Когда стоит задуматься: AMQP подходит для сложных распределенных систем, которым требуется надежный и гибкий протокол обмена сообщениями.
- Apache Kafka (Использует собственный двоичный протокол передачи данных на основе [TCP](#)): Apache Kafka - это распределенная публикация-подписка система, которая может обрабатывать большие объемы данных в реальном времени. Он предоставляет высокопроизводительное и надежное решение для обмена сообщениями в распределенных системах.

- Сложность использования: Высокая. Apache Kafka - это мощный инструмент, который требует понимания концепций, таких как топики, потребители, производители и брокеры.
- Стоимость использования: Высокая. Kafka требует специализированных ресурсов и знаний для установки и управления, что может повысить стоимость использования.
- Когда стоит задуматься: Kafka стоит рассмотреть для обработки больших объемов данных в реальном времени в сложных распределенных системах.