1. **Поясните понятие GraphQL.**

[GraphQL](https://graphql.org/) — язык запросов с открытым исходным кодом, разработанный Facebook. Он создавался как более эффективная альтернатива REST для разработки и использования программных интерфейсов приложений.

GraphQL обладает множеством достоинств, например:

1. Вы получаете информацию именно в том объёме, в котором запрашиваете. В отличие от REST, ответ на запрос не будет содержать ненужных данных.
2. Вам будет необходима всего одна конечная точка, никаких дополнительных версий для единственного API.
3. GraphQL — сильно типизированный язык, что позволяет предварительно оценить корректность запроса в рамках системы типов этого синтаксиса, до исполнения. Это позволяет разрабатывать более мощные API.

GraphQL — это язык запросов, используемый клиентскими приложениями для работы с данными.

1. **Поясните понятие схема GraphQL.**

Для создания схем в GraphQL используется собственный язык Schema Definition Language (SDL). SDL обладает интуитивно понятным синтаксисом и универсален для любой используемой технологии.

Позволяет организовывать создание, чтение, обновление и удаление данных в вашем приложении (то есть — перед нами четыре базовые функции, используемые при работе с хранилищами данных, которые обычно обозначают акронимом CRUD — create, read, update, delete).

1. **Расшифруйте аббревиатуру SDL GraphQL**.

Schema Definition Language

Для создания схем в GraphQL используется собственный язык Schema Definition Language (SDL). SDL обладает интуитивно понятным синтаксисом и универсален для любой используемой технологии.

1. **Поясните понятие resolver GraphQL.**

Resolver — это набор функций, которые генерируют ответ на запрос GraphQL. Проще говоря, распознаватель действует как обработчик запросов GraphQL.

**Resolver или распознаватель** — функция, которая возвращает данные для определённого поля. Resolver'ы возвращают данные того типа, который определён в схеме. Распознаватели могут быть асинхронными. С их помощью можно получать данные из REST API, базы данных или другого источника.

1. **Поясните понятие query GraphQL.**

Получение данных их БД (select-запрос).

С помощью запросов GraphQL получает необходимые данные с сервера. Тип запроса Query в GraphQL — аналог GET в REST. ... Query описывает данные, которые необходимо получить с сервера.

1. **Поясните понятие mutation GraphQL.**

Любое изменение в БД. В GraphQL изменения — способ модифицировать данные на сервере и получить обработанную информацию. Этот процесс можно рассматривать как аналогичный концепции CUD (Create, Update, Delete) в стандарте REST.

В GraphQL изменения — способ модифицировать данные на сервере и получить обработанную информацию. Этот процесс можно рассматривать как аналогичный концепции CUD (Create, Update, Delete) в стандарте REST.

1. **Поясните понятие subscription GraphQL.**

При обновлении данных сервер выполняет определённый в подписке запрос GraphQL и рассылает обновлённые данные клиентам. С помощью подписок поддерживается постоянная связь между клиентами и сервером.

**Подписки GraphQL** - это способ передачи данных с сервера клиентам, которые выбирают прослушивание сообщений с сервера в режиме реального времени.

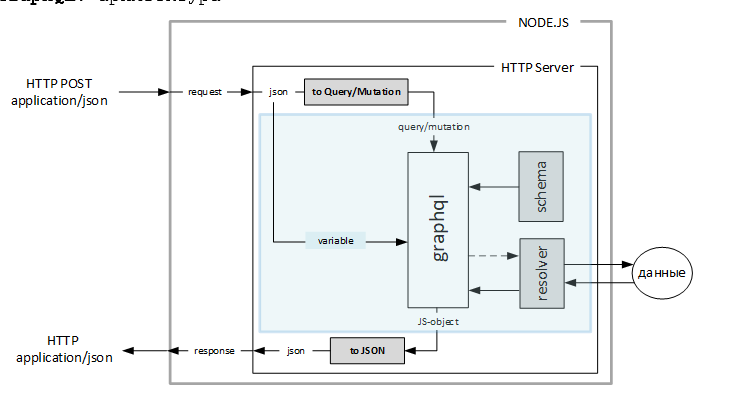
1. **Поясните понятие context GraphQL.**

Общий объект для распознавания resolver’ом.

Контекст представляет из себя объект, хранящий информацию о подключении пользователя к базе данных.

1. **Поясните схему работы модуля graphql**

//Рассказать, как вызываются функции



1. **Поясните следующие компоненты subscription, interface, enum, fragment, union схемы GraphQL.**

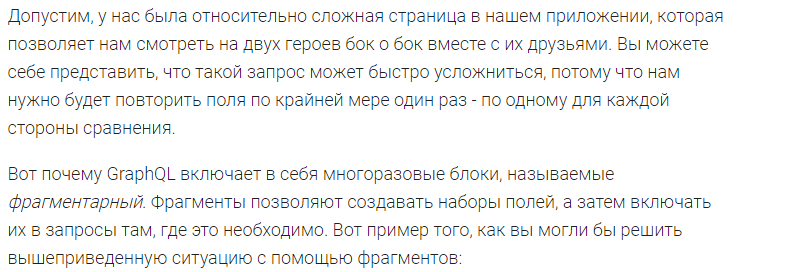
Как и многие системы типов, GraphQL поддерживает интерфейсы. Интерфейс-это абстрактный тип, который включает в себя определенный набор полей, которые тип должен включать в себя для реализации интерфейса.

**Interface -** способом построения и использования схем GraphQL посредством использования абстрактных типов

**Enum -** специальным типом, который мы можем использовать для перечисления всех возможных значений в поле. Используя Enums, мы добавляем другой вид проверки в существующую схему GraphQL. Указанные значения типа Enum являются единственно возможными вариантами, которые принимаются.

**Fragment**

**Fragment** позволяют создавать наборы полей, а затем включать их в запросы, где это необходимо.

****

**Union**

Типы объединения очень похожи на интерфейсы, но они не задают никаких общих полей между типами. **Union** указывает на то, что поле может возвращать более одного типа объекта