

Как нужно реализовывать API, чтобы придерживаться этих концепций?

Низкая связанность

1. Модульный дизайн API: разделите ваш API на более мелкие, сфокусированные компоненты или сервисы, которые отвечают за конкретные функции. Это позволяет клиентам взаимодействовать только с необходимыми частями, снижая зависимость и влияние при внесении изменений.
2. Используйте стандартизированные протоколы и форматы: придерживайтесь широко используемых протоколов, таких как HTTP/HTTPS, и форматов данных, таких как JSON или XML. Это позволяет клиентам и серверам общаться без жесткой привязки к конкретной реализации, что облегчает адаптацию к изменениям.
3. Версионность API: внедрите версионность для обработки изменений API без ущерба для существующих клиентов. Поддерживайте старые версии в течение разумного времени, чтобы дать клиентам возможность переходить на новые версии в своем собственном темпе.
4. Гипермедийные ссылки (HATEOAS): включение гипермедийных ссылок в ответы API позволяет клиентам обнаруживать и перемещаться по функциональности API, уменьшая необходимость в жестко закодированных конечных точках и фиксированных структурах.
5. Шлюз API (паттерн API, рассмотрим в отдельном модуле): используйте API-шлюз для управления маршрутизацией, преобразованием и агрегацией. Таким образом, клиенты могут оставаться изолированными от деталей реализации API, что облегчает адаптацию изменений.

Высокая сплоченность

1. Единая ответственность: каждый компонент или сервис API должен быть сфокусирован на одной обязанности или функциональности. Это приводит к созданию хорошо организованного API, что облегчает его поддержку и понимание.
2. Группируйте связанные функциональные возможности: организуйте конечные точки и ресурсы API на основе связанных функциональных возможностей. Это создает более логичную структуру и позволяет клиентам легко находить и использовать необходимые сервисы.
3. Согласованные соглашения об именовании: установите и соблюдайте последовательные соглашения об именовании для конечных точек, ресурсов и параметров API. Это создает последовательный и предсказуемый API, улучшая работу разработчиков.
4. Последовательная обработка ошибок: реализуйте последовательную и предсказуемую обработку ошибок во всем API. Это облегчает клиентам понимание и обработку различных сценариев ошибок, уменьшая жесткую привязку к конкретным реализациям обработки

ошибок.

5. Только интерфейс: скрывайте внутренние детали реализации(логику работы) и раскрывайте клиентам только необходимую функциональность(в интерфейсе API). Это гарантирует, что клиенты смогут взаимодействовать с чистым, понятным интерфейсом, и уменьшает влияние внутренних изменений на клиентов.