Day 8 Graphql, Pagination, Filter, rate limit
Sunday, October 13, 2024

API

Simple API:
- Simple Object Access Protocol
- rest based
- has more security, also more overhead

Rest API:
- light weight, stateless
- very common
- works with http

HTTP:
- Header, status code, Body
- Methods

Post → Create
Get → Read
Put → Update
Delete → Delete

CRUD

amazon.com/products
ebay.com/cart
/users

Versioning:
amazon.com /v1/products
/v2/products
- Stay have backward compatibility

This is captured in Documentation
Documentation → Swagger (OpenAPI)

Error Handling → error code/Purpose
- is also mentioned in Documentation

HTTP level caching

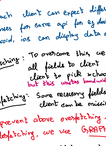
Client → GET /products
Server → 200 OK, Etag: "abc123", Body: [product_1, product_2]
Client → GET /products, If-None-Match: "abc123"
Server → 304 Not Modified, Etag: "abc123"
Client → 200 OK, Etag: "def456", Body: [product_1, product_2, product_3]

Cache
- Client Cache: Stores responses locally.
- Proxy Cache: Intermediary server caches responses.
- Server Cache: Caches data on the server.

Cache invalidation:
- Time-based: Cache expires after a set time.
- Event-based: Cache invalidated when data changes.
- Manual: Developer explicitly clears cache.

Cache mechanisms:
→ Etag: If-None-Match: etag
→ Cache-Control: max-age=3600
→ Last-Modified Header: If-Modified-Since: Date

GRAPHQL



Overfetching:
- Client receives more data than needed.
- Example: Fetching user profile when only needing name.

Underfetching:
- Client needs multiple requests to gather all required data.
- Example: Fetching user profile and their posts separately.

To prevent above overfetching and underfetching, we use GRAPHQL

GraphQL: Can fetch exact field every time. Client can request what are all the fields to expect, so GraphQL can fetch it from server.

We tell fields in GraphQL schema

Rule: GraphQL

Create, update, deletion is called Mutation

GraphQL requests are always POST as it has payload

GraphQL is a query language for APIs

Which tells client needs exactly what fields. It helps to fetch only what it needs

Reference:
https://www.apollographql.com/docs/apollo-server/guides/error-codes/

GraphQL helps to overcome overfetching and underfetching.

Pagination & Filter

Pagination → get products on front page, 20 products show, other products are in pagination

Interview: Should talk about GET API, talk about Pagination & Filtering, talk about Rate Limiting, Caching, Versioning, Error handling

Query Parameters:
- Filters = category=shoes&price=50
- sort=price:desc
- Caching
- Compression

In below we are retrieving subset of data

Pagination

/api/products? page=2 &limit=20
page=20

RATE LIMITING

Client → Server

to avoid more spam request we can use Rate Limiting

→ Prevent Abuse
→ Avoid downtime
→ CPU, Memory, Bandwidth
→ Fair usage
→ Security (Prevent DDOS)

we can work about
1) per Api
2) per User

Algo
- Token Bucket
- Leaky Bucket
- Fixed window
- Sliding window

Redis

Client → Server → Redis → Server → Client

Client, Server, Redis

Client → Server: GET /api/data
Server → Redis: INCR client:ip:1234
Redis → Server: 1
Server → Client: 200 OK, Data

Client → Server: GET /api/data
Server → Redis: INCR client:ip:1234
Redis → Server: 101
Server → Client: 429 Too Many Requests

Redis Limiting Pagination:
→ Fixed window Algo
→ Rolling window Algo

INTERVIEW

5 Interview: Interview we talk about GET in Interview

Pagination
Filtering
API versioning
Rate limiting
GraphQL

API Design

Resources: /products, /orders, /users

Crud Functions

→ Name
→ Method / body
→ Parameters
→ Response
→ Error handling

Youtube

F.R.: upload video (user id, title, video_id, video_description, tags [], date)

Functions

Tweet (user_id, tweet_text, location, user_id)

Search (user_id, search_term, page, limit)

Both of following are Reverse Proxies

Api gateway	Load Balancer
Api management Rate limiting Analytics Only at entry point	directly load balancing multiple servers Increase fault tolerance can be at many level Eg: nginx

Sometimes both can be combined but remember each functionality