Criteria	Rest	GraphQL	gRPC
Client / Server	Yes	Yes	Yes
Stateless	Yes	Yes	Yes
Cacheable	Yes	Possible	No
Layered	Yes	Possible	No
Code on Demand	Yes	No	No
Uniform Interface			
- Resource Identifiers	Yes	No	No
- Resource Representation	Yes	No	No
- Self Describing	Yes	Yes	No
- Hyptertext as an Engine of State	Yes	Possible	No

Figura 1

Fonte:https://cdn.ymaws.com/ukoug.org/resource/collection/904331A4-A794-4730-AC6B-6AB8B1DD1E36/gRPC,_REST,_GraphQL_-_TechFest19_-_upload_version.pdf

Capacidade	REST	GraphQL	gRPC
Compressão de carga	Gzip	Gzip + campos requisitados	Binário
Requisições assíncronas	HTTP	AMQP requisitados	HTTP/2
Requisições reduzidas	Requer uma api composta	Recupera apenas o que está especificado na requisição	Escreva sua função
Padrões de mercado	Muitas ferramentas e padrões	Uso experimental	Uso experimental
Reuso	Baseado em Domínios de Negócios	Função específica	Função específica
Compatibilidade com arquitetura de evento ou reativa	Solicitações de sincronização	Sintaxe de mutação e chamadas assíncronas	Chamadas assíncronas e flexibilidade de sintaxe
Parser de quadros otimizado	JSON	JSON	Binario

Figura 2

Comparando as 3 tecnologias, temos diversas vantagens e desvantagens umas sobre as outras. Temos que olhar para o cenário atual do projeto e verificar qual a melhor tecnologia para nos atender.

O GRPC, por exemplo, é uma boa escolha para componentes internos críticos de desempenho para a sua arquitetura, pois ele é por volta de 10 vezes mais rápido que algumas versões do GraphQ(sem o Keep Alive) e REST. Porém, o GRPC é mais difícil de desenvolver, depurar e detectar alterações de esquema.

O GraphQL mais fácil de testar e depurar e possui algumas otimizações de desempenho como carregador de dados e é melhor para o consumo de vários clientes. Porém,

possui diversas desvantagens como: Em relação ao monitoramento da aplicação, pois a API sempre irá retornar um Status Code 200 para erro ou execução com sucesso; O GraphQL também limita a quantidade de consultas do usuário, tornando a aplicação limitada. Trabalhar com cache em API's GraphQL é muito mais complexo do que em comparações com os outros tipos de API, pois como a consulta é customizada pelo usuário, o armazenamento de cache é comprometido.

O REST, como mostrado na Figura 1, é um tipo de API muito mais completo em comparação aos outros, pois possui uma comunidade mais ampla, trazendo uma maturidade muito maior. Na Figura 2, podemos perceber que o REST tem um padrão de mercado rico e bem definido, ao contrário dos outros que estão em uso experimental.