Grundlegende mögliche Vor- und Nachteile von GraphQL vs REST:

<https://medium.com/gdplabs/graphql-case-study-kaskus-groups-7602b33722ea>:

* Performance Verbesserung in GraphQL durch Vermeidung von:
  + Under-fetching: Mehrere Anfragen nötig, um alle benötigten Daten zu holen
  + Over-fetching: Endpoint liefert mehr Daten als benötigt werden
* Caching:
  + Kein natives Caching in GraphQL
  + Automatisches Browser Caching von GET-HTTPs bei REST, Mögliches Caching von POST-HTTPS
* Mögliche Probleme bei der Skalierbarkeit

<https://medium.com/airbnb-engineering/reconciling-graphql-and-thrift-at-airbnb-a97e8d290712>:

* Ein stark typisiertes API-Schema
* Flexibilität bei der Feldauswahl, sowohl auf Makroebene (fordern Sie keinen teuren Vorgang an, wenn Sie ihn nicht benötigen) als auch auf Mikroebene (fordern Sie kein bestimmtes Feld an, das Sie nicht mehr verwenden)
* Plattformübergreifende clientseitige Entwicklung (Web, iOS, Android), ohne dass Backend-Ingenieure für die Iteration erforderlich sind
* Die beträchtliche Welt der clientseitigen Vorteile, die das Apollo-Ökosystem bietet, einschließlich Caching, Synthese des lokalen Zustands und des Netzwerkzustands, Analysen auf Feldebene und mehr

<https://blog.logrocket.com/5-reasons-you-shouldnt-be-using-graphql-61c7846e7ed3/>:

* Komplizierteres Error-Handling?
* Caching

<https://www.ionos.de/digitalguide/websites/web-entwicklung/graphql/>:

* Zitat: „GraphQL im Backend so umzusetzen, dass es performant und sicher läuft, ist also wesentlich schwieriger.“

<https://www.howtographql.com/basics/1-graphql-is-the-better-rest/>:

Unternehmen mit GraphQL:

* Paypal: <https://medium.com/paypal-engineering/tagged/graphql>
* Airbnb: <https://medium.com/airbnb-engineering/reconciling-graphql-and-thrift-at-airbnb-a97e8d290712>
* Shopify: <https://engineering.shopify.com/blogs/engineering/solving-the-n-1-problem-for-graphql-through-batching>

<https://blog.graphqleditor.com/>

<https://www.dev-insider.de/graphql-als-alternative-zu-rest-a-752074/>

<https://medium.com/@back4apps/graphql-vs-rest-62a3d6c2021d>

<https://leapgraph.com/graphql-file-uploads>

<https://leapgraph.com/why-graphql-is-bad>

https://www.upwork.com/hiring/data/graphql-projects

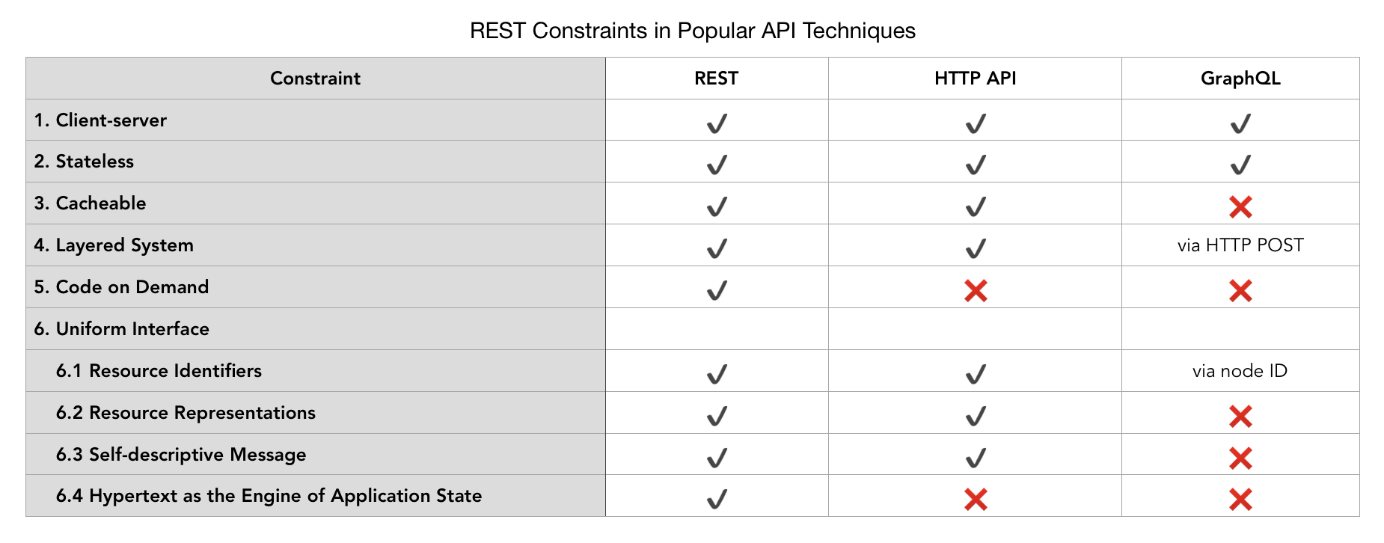
https://www.indiehackers.com/forum/in-what-projects-does-it-make-sense-to-go-for-graphql-c81a510853

https://www.moesif.com/blog/technical/graphql/REST-vs-GraphQL-APIs-the-good-the-bad-the-ugly/

https://www.manning.com/books/the-design-of-web-apis

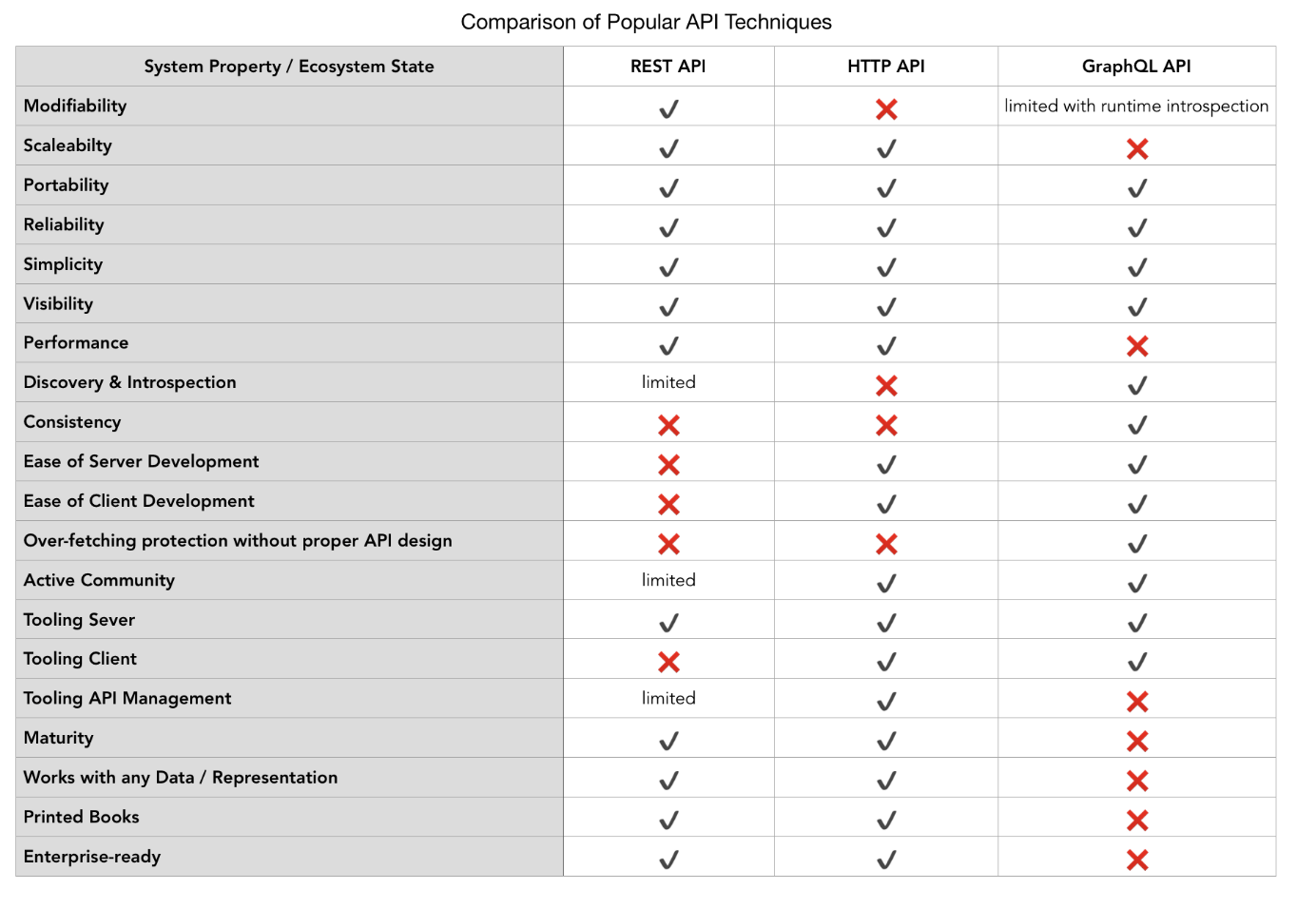
https://ebookcentral.proquest.com/lib/htwg-konstanz/reader.action?docID=5764425&query=GraphQL# => P. 214-220!

<https://goodapi.co/blog/rest-vs-graphql> **!Anschauen: Viele Pros und Cons, noch genauer anschauen**



Zitat: „If you have decided to go with the microservices architecture, REST is probably the best style to enable for independent evolution of services and context separation“

* Liegt womöglich auch daran, dass der eine Vorteil von GraphQL, nur ein Endpoint, damit eigentlich wegfällt



<https://www.graphql.com/articles/4-years-of-graphql-lee-byron> **Anschauen!**

<https://www.youtube.com/watch?v=pywcFELoU8E> **Anschauen!**

Caching:

<https://blog.graphqleditor.com/grapqhl-vs-rest-caching/>

Beide können spezifische Attribute einer Route ausgeben (REST-> index.ts, z.310),

aber REST kannn nur genau ein Attribut auswählen und auch nur für genau eine Route.

This is in fact one of the core benefits of GraphQL in general: It enforces that the API

actually behaves in the way that is promised by the schema definition! This way, everyone

who has access to the GraphQL schema can always be 100% sure about the API operations and

data structures that are returned by the API.