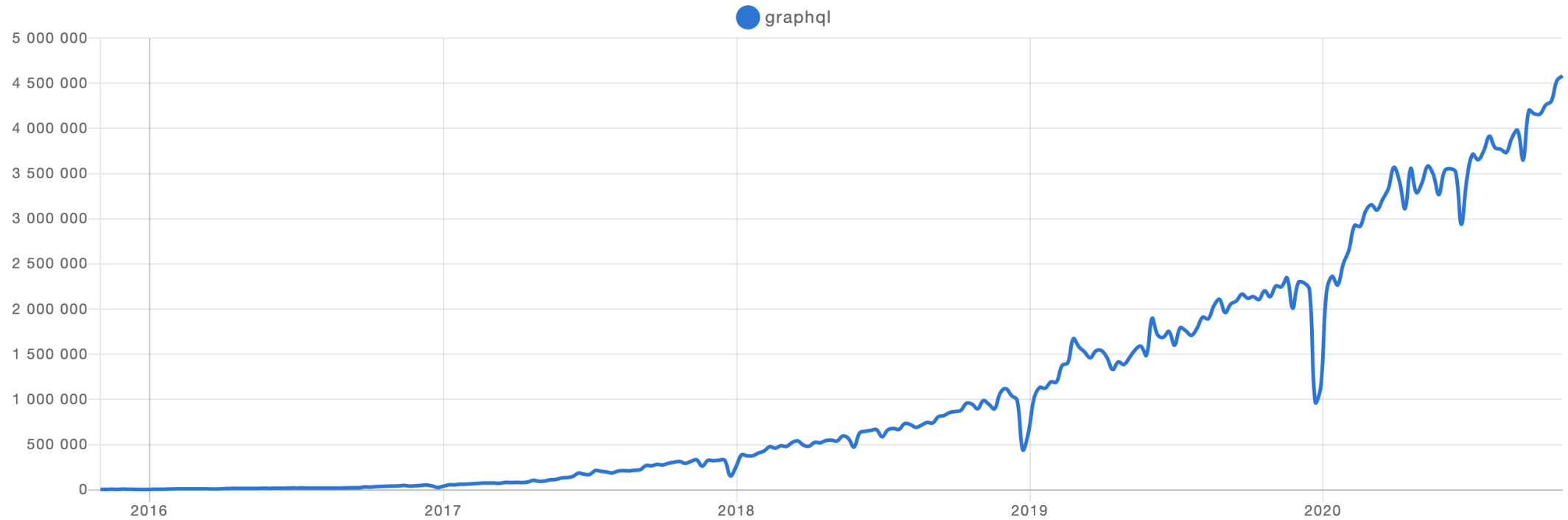
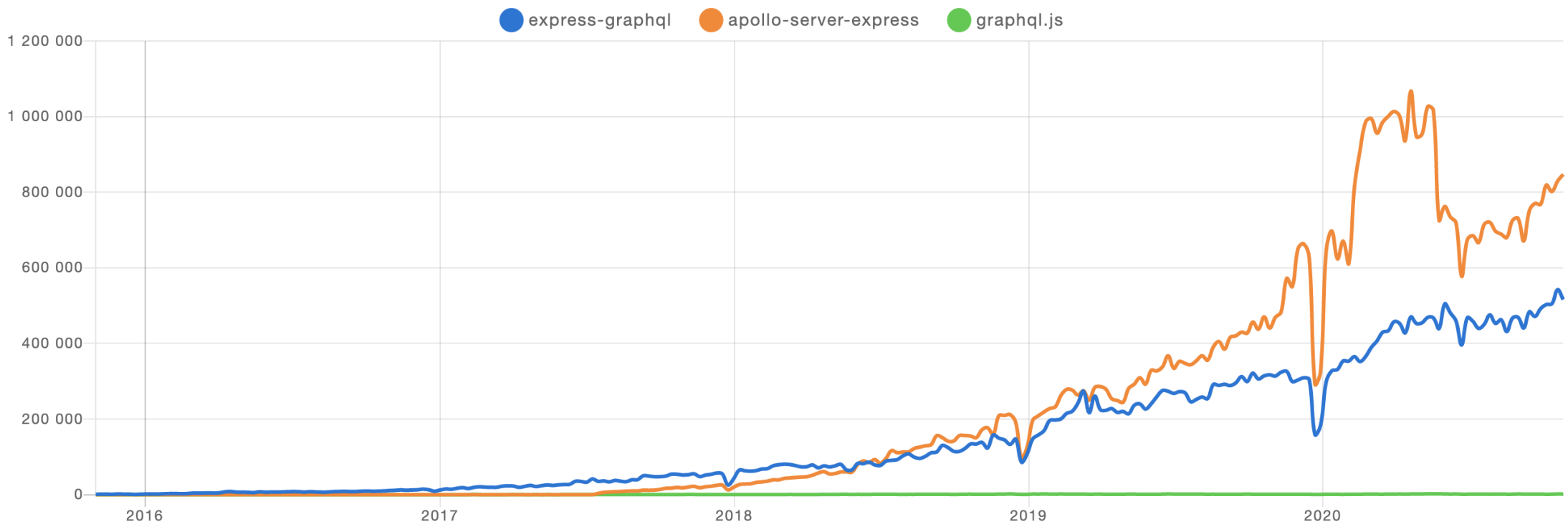
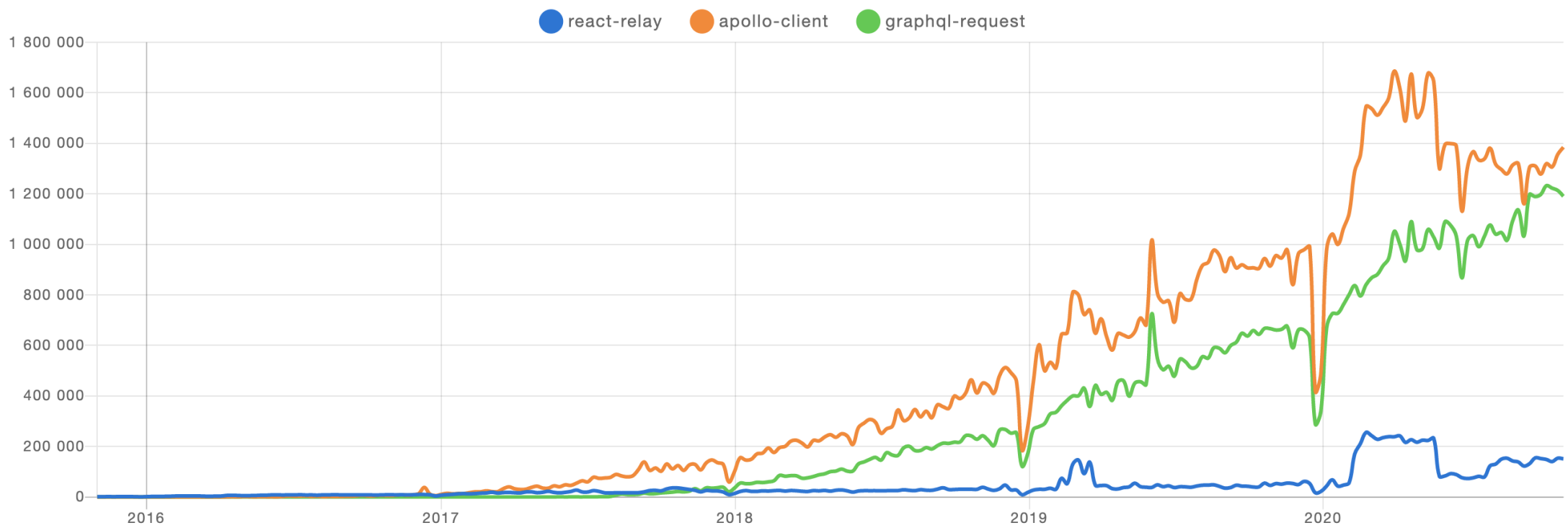
GraphQl

- Vzniklo v roce 2012 pro potřeby Facebooku
- Open source release byl v roce 2015, od té doby nabíra prudce na popularitě


graphql
5 000 000
4 500 000
4 000 000
3 500 000
3 000 000
2 500 000
2 000 000
1 500 000
1 000 000
500 000
0
2016
2017
2018
2019
2020

- GraphQL samo o sobě není implementace, nýbrž specifikace postavená nad HTTP protokolem
- Implementace jsou tvořeny týmy třetích stran
- Příklady serverových knihoven pro Node - `grapql.js, express-graphql, apollo-server atd.`
- Příklady klientských knihoven - `Relay, Apollo Client, graphql-request, lokka, nanogql atd.`
- Oproti tradičnímu RESTu GraphQL vystavuje pouze jeden endpoint a následné interakce se zpracovávají tzv. resolvery
- Dotaz na GraphQL server se nazývá `query`
- Každý GraphQL server má out-of-the-box API dokumentaci
- apollo-server automaticky cacheuje data pro `typeDefs` objekt, lze však cacheovat pole i manuálně pomocí `@cacheControl`)


express-graphql
apollo-server-express
graphql.js
1 200 000
1 000 000
800 000
600 000
400 000
200 000
0
2016
2017
2018
2019
2020


react-relay
apollo-client
graphql-request
1 800 000
1 600 000
1 400 000
1 200 000
1 000 000
800 000
600 000
400 000
200 000
0
2016
2017
2018
2019
2020

- REST vrací JSON odpověďi, které jsou častokrát velmi obsáhlé a zároveň né vždy úplné, tzn. že pro získání konečných dat je potřeba udělat několik requestů a následně data pospojovat (toto se dá samozřejmě řešit na BE, nicméně není to pravidlem)
- GraphQL oproti tomu umožňuje navzájem propojená data vracet v jednom query, tím se docílí optimalizace síťového přenosu a přenesených dat
- Klient má pod kontrolou, jaká data chce

REST Příklad

GET /users

```
{"users": [
  {"name": "Johny Mačeta", "id": 7}
  .....
]}
```

GET /users/:id/books

```
{"books": [
  {"authorId": 7, "id": 2, "title": "My Little Pony" }
  .....
]}
```

GraphQL Příklad

```
query {
  users {
    id
    name
    books {
      id
      title
    }
  }
}
```

Response

```
{"users": [
  {"name": "Johny Mačeta", "id": 7, "books": [{"id": 2, "title": "My Little Pony" }]}
  .....
]}
```

- GraphQL je silně otypované, díky tomu je celkem obtížné "šlápnout vedle"
- Věškeré interakce a datové modely jsou součástí schématu, ve kterém jsou zaznamenány i vztahy mezi jednotlivými modely

3 základní pilíře GraphQL schématu

- `Queries` - dotazy
- `Mutations` - změny dat
- `Subscriptions` - real time funkcionalita

Každý pilíř je složen z tzv. resolverů, které se starají o zpracování query.

- Resolver je funkce, která zpracuje query a vrátí požadovanou odpověď
- Každý resolver dostává 4 argumenty:
  - `parent` - ve specifických případech odkazuje na rodičovský objekt
  - `args` - argumenty předané do volání query
  - `context` - např. obsahuje informace o requestu, headers atd. (musí se zaregistrovat)
  - `info` - bohužel, o infu moc infa nemám :-D
- Resolvery můžou být asynchroní