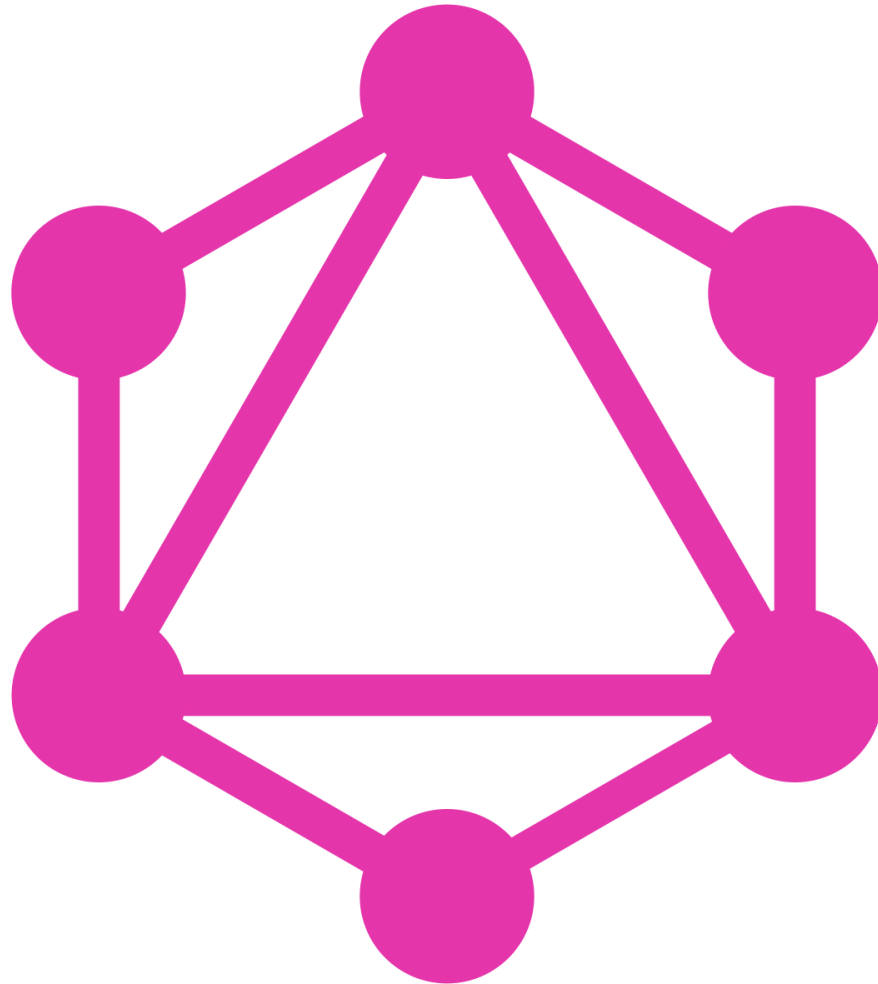









GraphQL





- Autorem GraphQL je Facebook
- V roce 2012 se začal trh orientovat stále více na mobilní aplikace
- Byl vytvořen tým vývojařů, kteří začli pracovat na nativní iOS aplikaci
- Ihned narazili na potíže se stávající API, konkrétně s newsfeed API

 Hledat 


Hlavní stránka | Hledat přátele    

 Vybrané příspěvky ...

 Messenger

 Marketplace

Zástupci

 Cistepc.cz 8

1

10+

1


4


12


20+


Zobrazit další...


Prozkoumat


 15 Události 1


 Skupiny


 Stránky


 Seznamy přátel


 V tento den 1

 Vysílání Live


 Správce reklam


 Vytvořit příspěvek

 Album fotek nebo videí

 Živé vysílání


Co se vám honí hlavou,


 Fotka nebo video


 Pocit nebo aktivita


...

Vaše stránky ...

 Příspěvek

 Fotka

 Živě


 Pozvat

To se mi líbí

Zobrazení

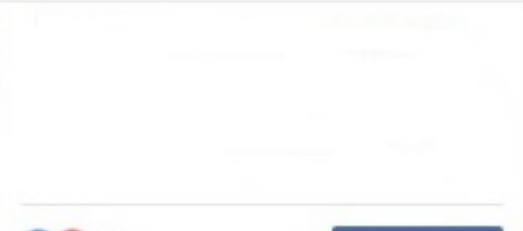
Příspěvky

Vyplňte dotazník a získáte doporučení

 Vyplňte minutový dotazník a získáte personalizované doporučení, které vám pomůže s příští reklamou dosáhnout vytčených obchodních cílů.

Vyplnit dotazník

Recent Posts



Create Promotion

News Feed

- Stávající API byla navržena tak, že většina logiky probíhala na serveru
- Pro nativní iOS appku bylo potřeba, aby většina logiky probíhala v appce a na server se chodilo jen pro data
- Skrze Facebook se začalo uvažovat o možnostech úpravy stávající API



Lee Byron

Dan Schafer

Nick Schrock

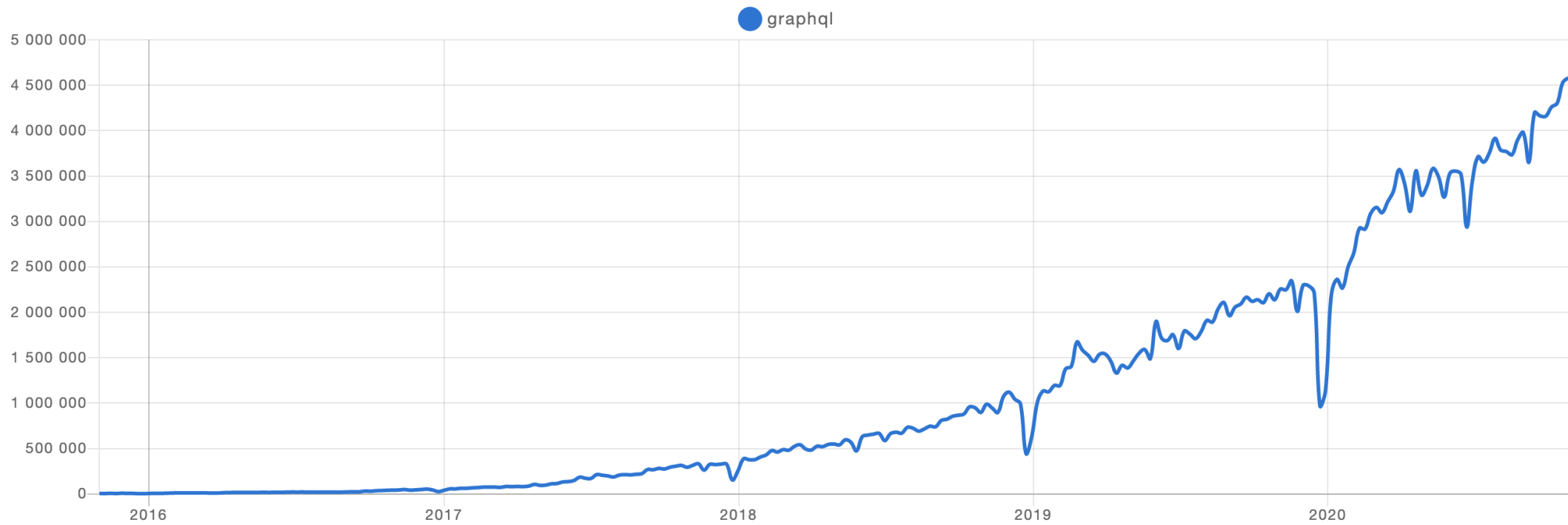
- Nick Schrock napsal první prototyp pojmenovaný **SuperGraph**
- To se velmi líbilo Leeovi (super IT genius), který začal přidávat vlastní nápady
- Brzy se přidal i Dan



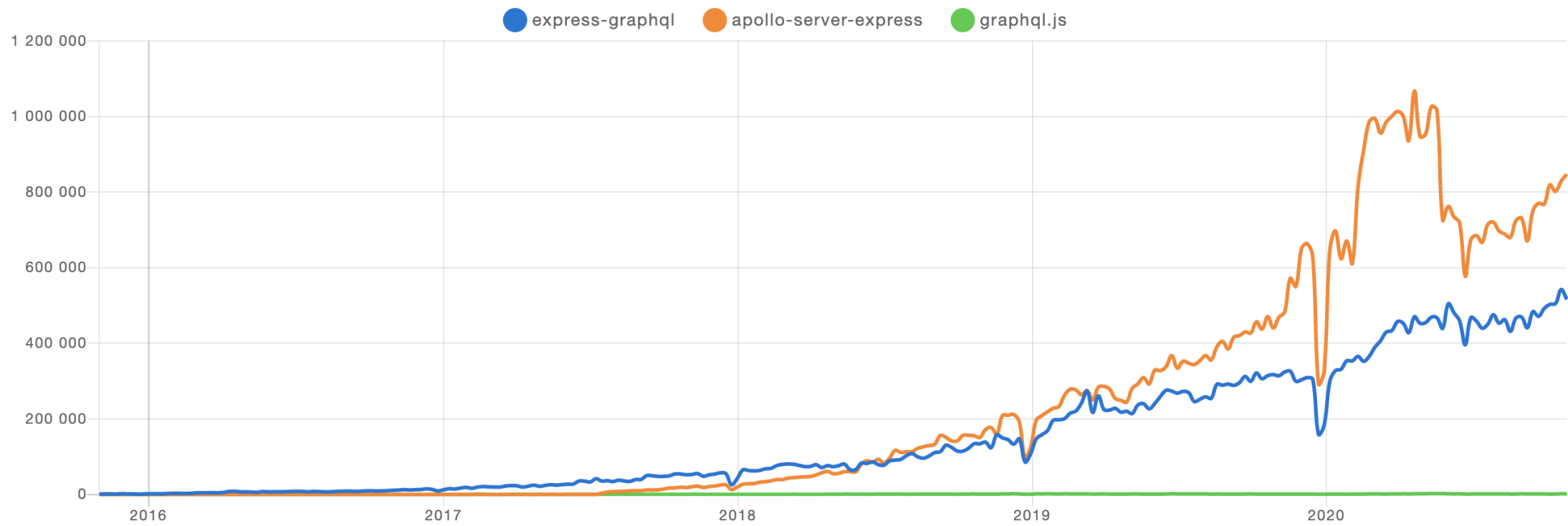
- Během 14ti dní měli hotovou první funkční verzi a během pár měsíců vytvořili první produkční verzi, zatím pouze pro Facebook (2012)
- V roce 2015 přišel Nick s nápadem GraphQL open sourceovat, nesetkal se s vřelým přijetím, ale eventuálně se odsouhlasilo, že se GraphQL stane open source projektem. Došlo k tomu v druhé půlce roku 2015. Nejprve však specifikace prošla kompletním refactoringem.
- Nebyla zveřejněna konkrétní interní implementace Facebooku, nýbrž pouze dokument, co to GraphQL je. Konkrétní implementace začala tvořit komunita, nadšená z tohoto nápadu
- Brzy se zjistilo, že spousta firem řeší podobné problémy, jako řešil Facebook
- Došlo k rychlé adopci GraphQL a během pár měsíců byli hotové implementace pro mnoho jazyků

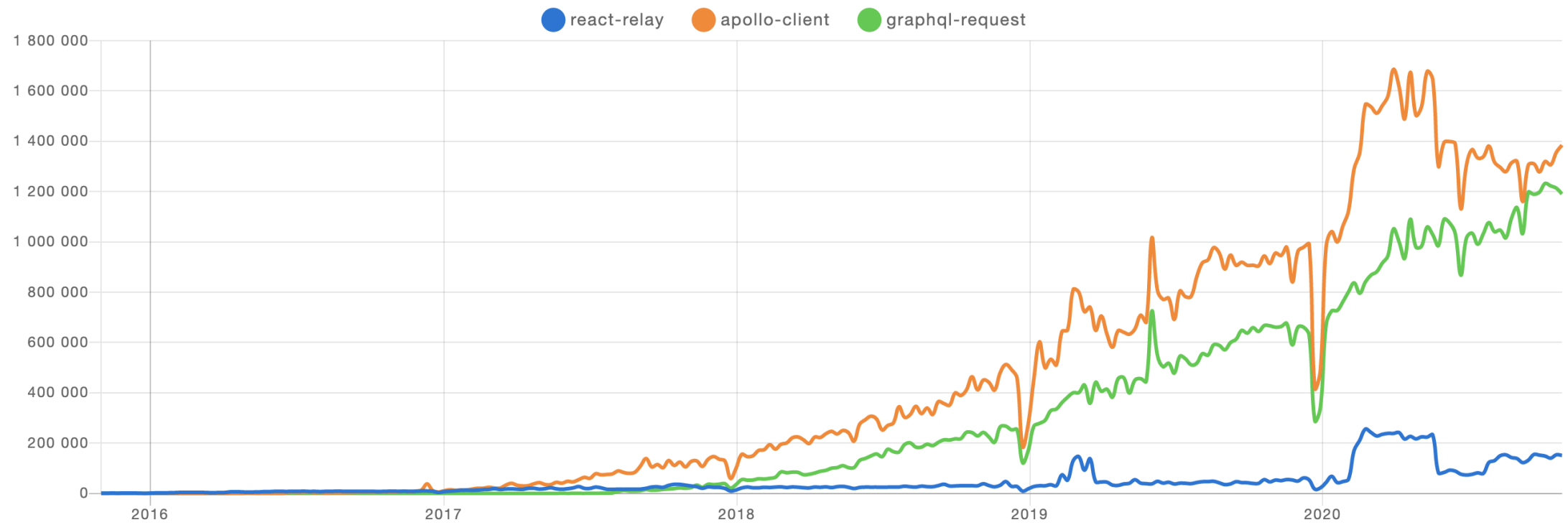
Kdo GraphQL používá?

- Facebook
- Twitter
- AirBnb
- Prisma
- GitHub
- Shopify
- Pinterest
- Audi
- DailyMotion
- PayPal
- Yelp
- a mnoho dalších.....



- GraphQL samo o sobě není implementace, nýbrž specifikace postavená nad HTTP protokolem
- Implementace jsou tvořeny týmy třetích stran
- Příklady serverových knihoven pro Node - `graphql.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`)





- REST vrací JSON odpovědi, 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": "Johnny 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 funkcionality

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 info moc info nemám :-D
- Resolvery mohou být asynchroní