SORT

Using GraphQL with RESTful APIs

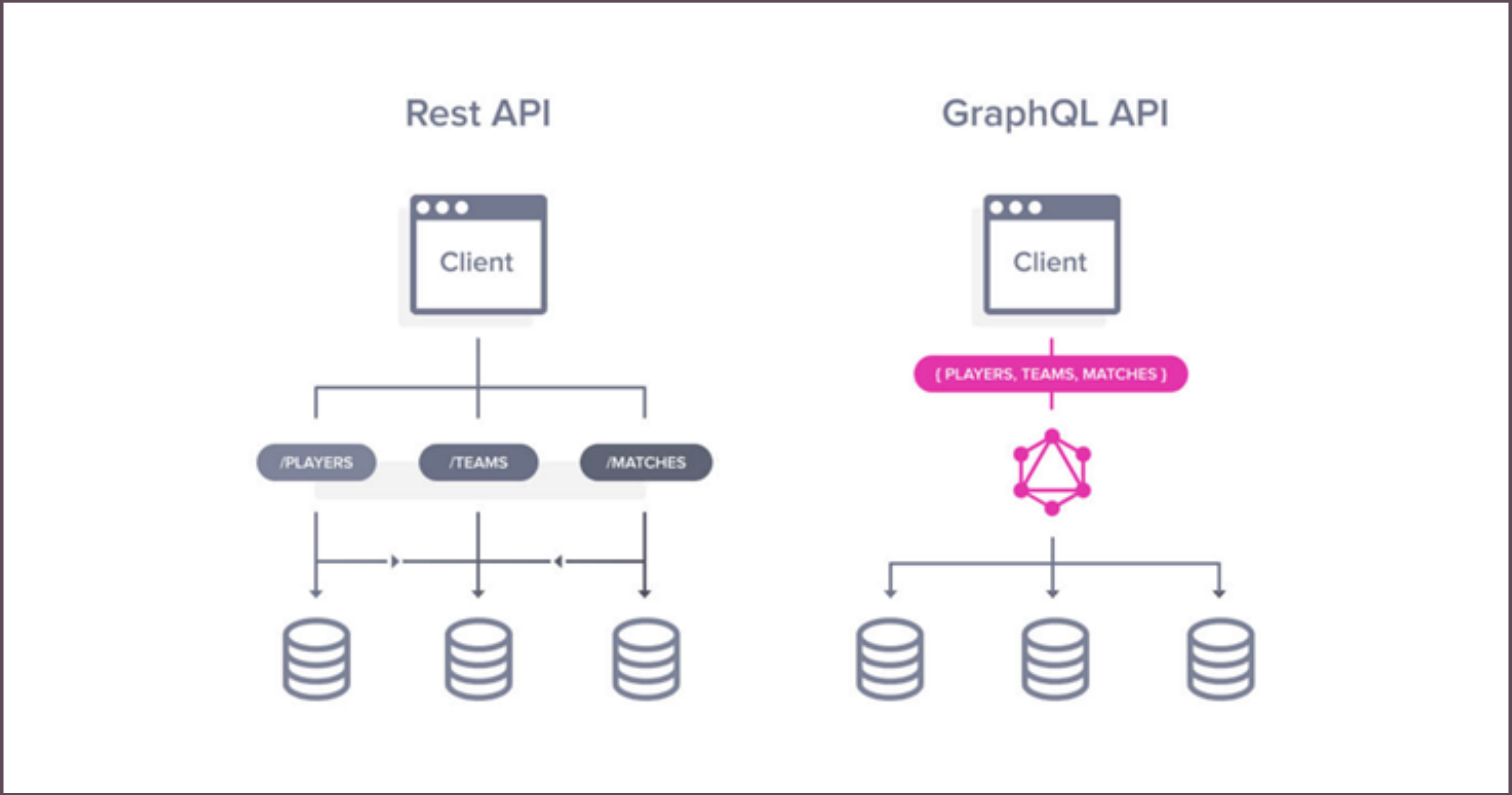
David Gurney, Logan Allred

October 8, 2019

What is GraphQL

- Query language spec for your APIs and data that is strongly typed
- Clients declare precisely the data shape they need, and servers return just the data that was requested.
- Uses a single POST endpoint with various resolvers to gather the data, resolves to the property level
- Improve DX: provides helpful tooling and built-in documentation
- Not a silver bullet

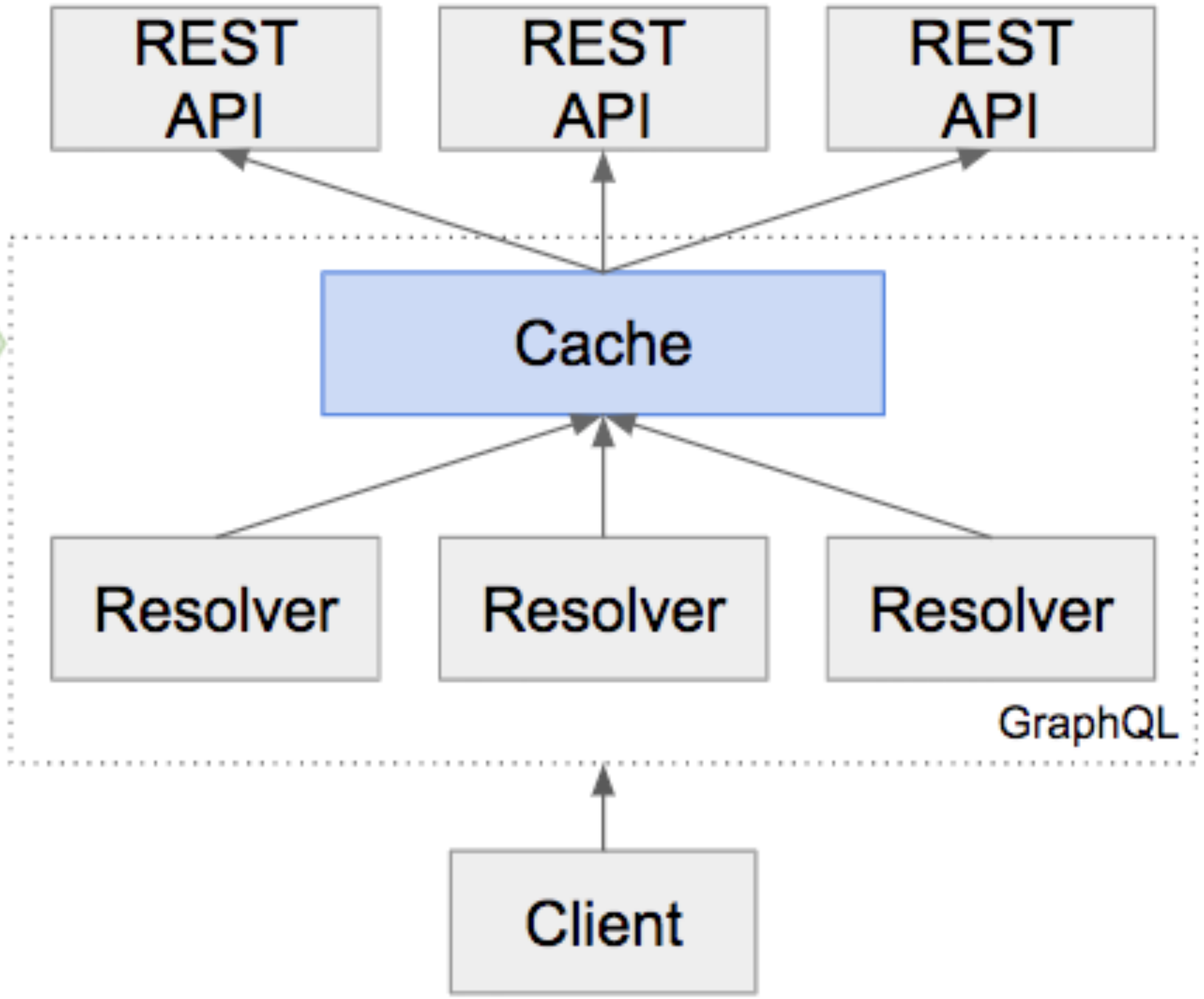
SORT



https://everis.passle.net/post/102fjj7/graphql-is-it-the-new-rest-part-2-of-4

GraphQL to REST Flow

1. Client sends query to server
2. Server parses and analyzes query
3. Server calls resolvers for each type/property defined
4. Resolvers request data from REST APIs through the Cache
5. Cache resolves any data already saved in the cache, passes through remaining requests
6. Data is returned from REST APIs and matched to the resolvers
7. Once all resolvers are resolved 🤦‍♀️, response is returned to client



Why wrap REST APIs with GraphQL?

MOBILE PERFORMANCE

ABSTRACTION/ STABILITY

ORCHESTRATION/ AGGREGATION

CLIENT SIMPLIFICATION

SORT

Mobile Performance

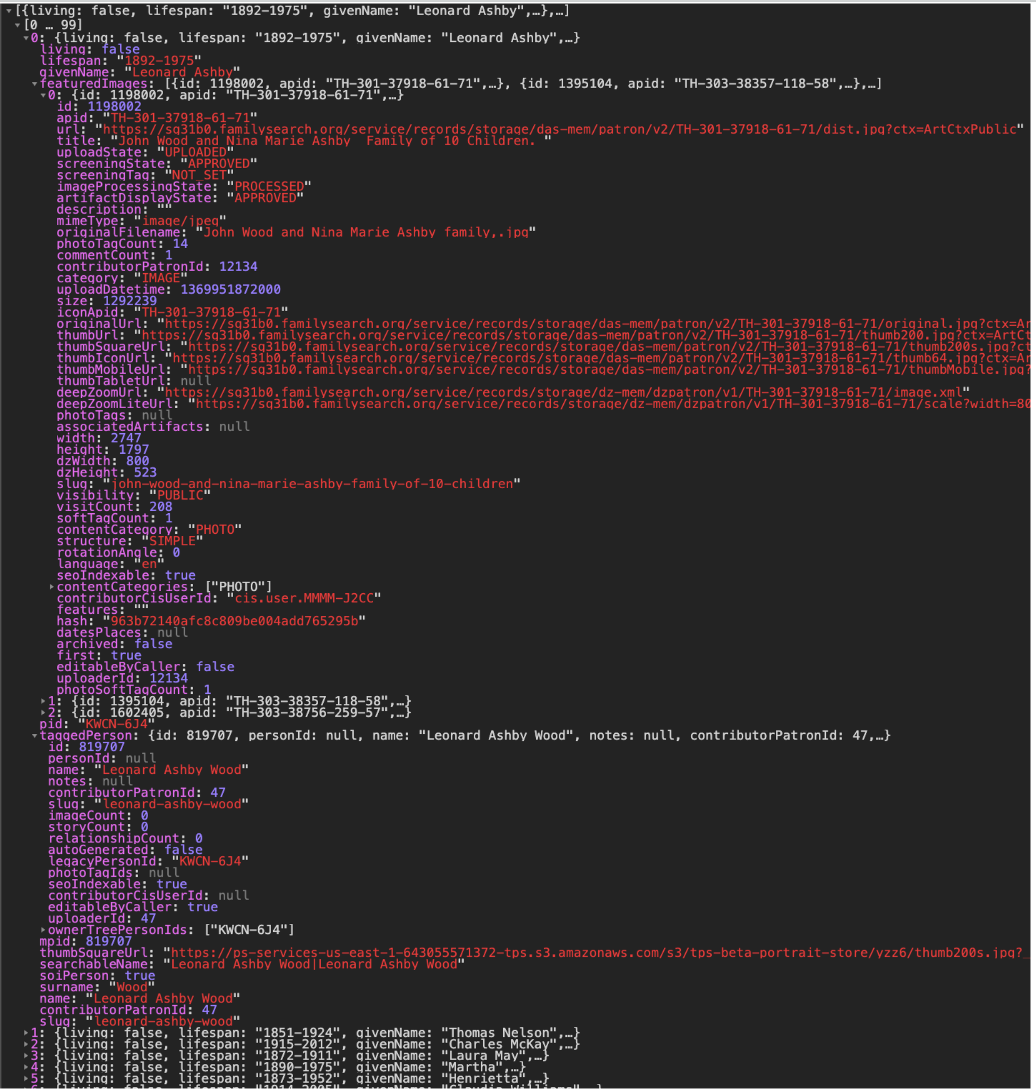
```
GET /users
GET /users/1/posts
GET /users/2/posts
GET /users/3/posts
GET /users/4/posts
GET /posts/1/attachments
GET /posts/2/attachments
GET /posts/3/attachments
```

```
POST /graphql
```

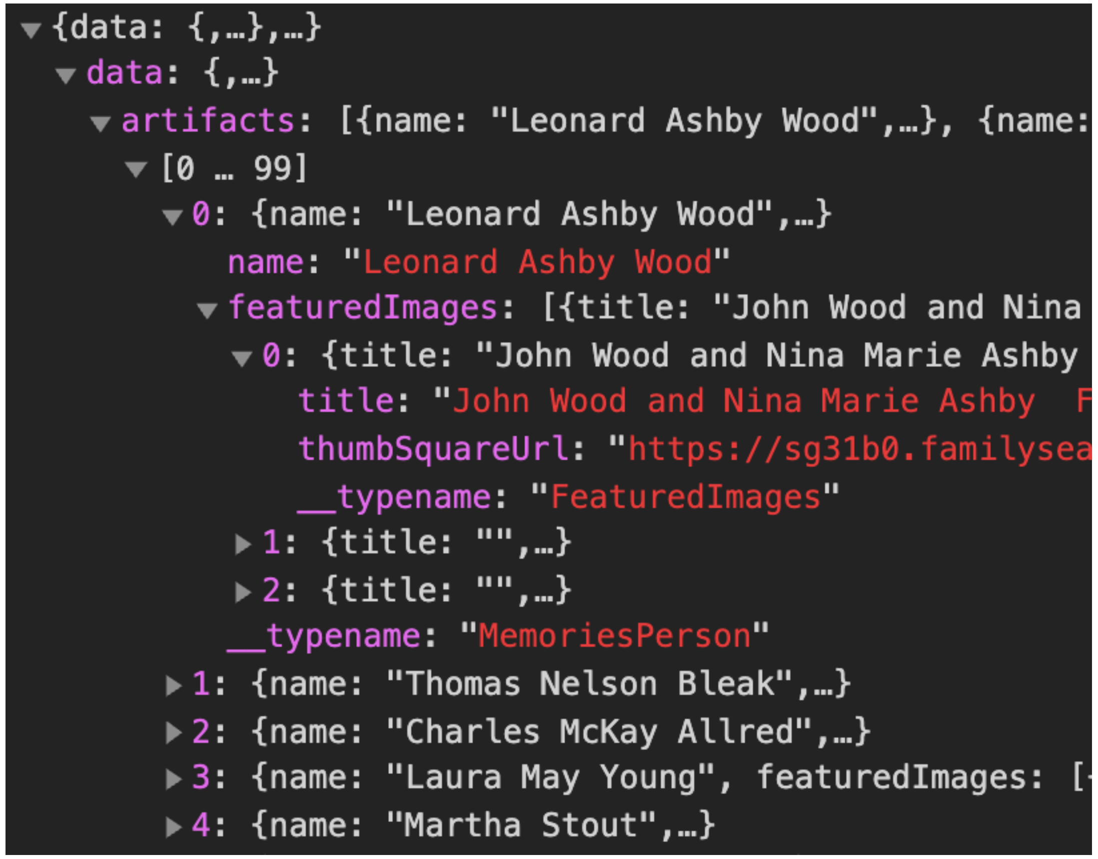
```
{
 users{
  name
  posts {
    id
    title
    content
    url
    attachments {
      src
    }
  }
}
```

SORT

Mobile Performance



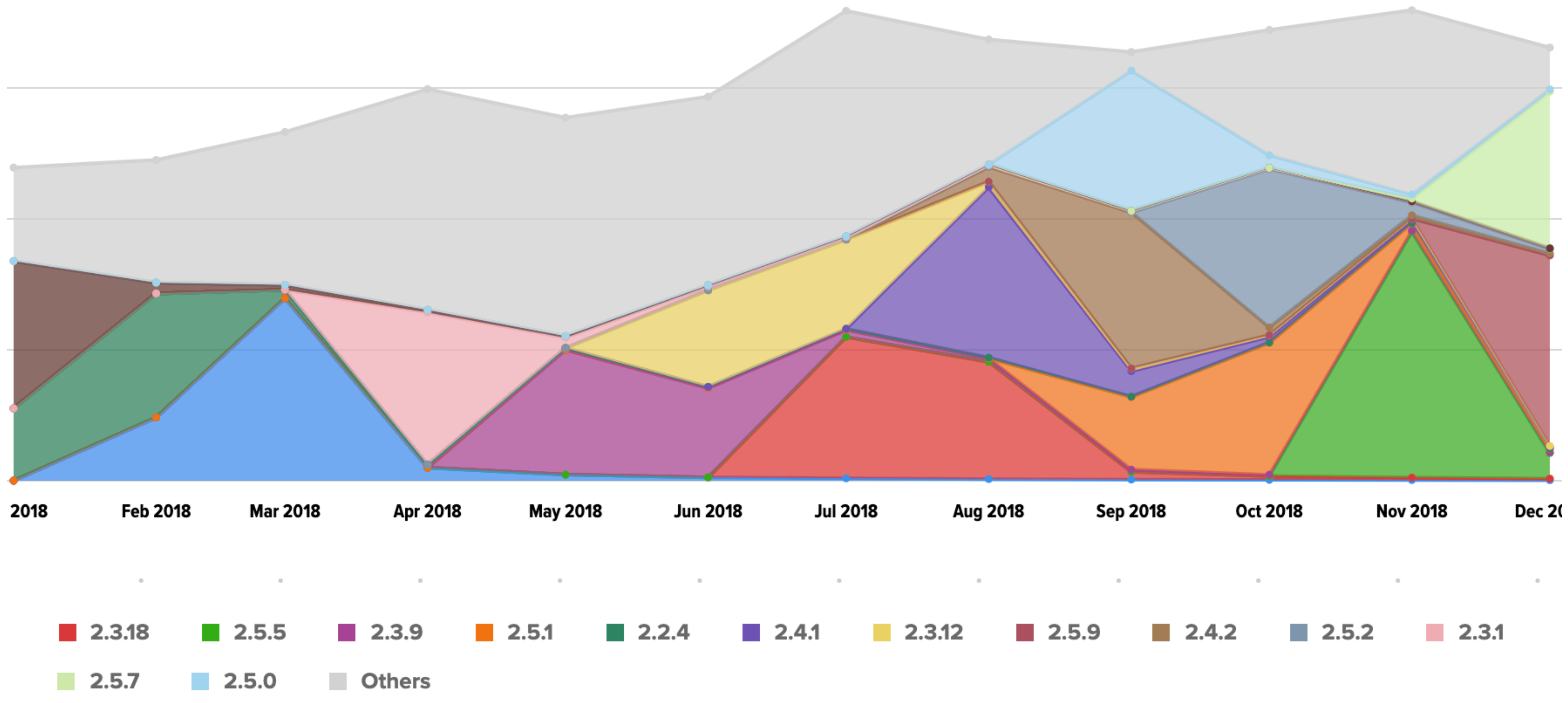
```
▾[{living: false, lifespan: "1892–1975", givenName: "Leonard Ashby",…},…]
 ▾[0 … 99]
  ▾0: {living: false, lifespan: "1892–1975", givenName: "Leonard Ashby",…}
     living: false
     lifespan: "1892–1975"
     givenName: "Leonard Ashby"
    ▾featuredImages: [{id: 1198002, apid: "TH-301-37918-61-71",…}, {id: 1395104, apid: "TH-303-38357-118-58",…},…]
     ▾0: {id: 1198002, apid: "TH-301-37918-61-71",…}
        id: 1198002
        apid: "TH-301-37918-61-71"
        url: "https://sg31b0.familysearch.org/service/records/storage/das-mem/patron/v2/TH-301-37918-61-71/dist.jpg?ctx=ArtCtxPublic"
        title: "John Wood and Nina Marie Ashby  Family of 10 Children. "
        uploadState: "UPLOADED"
        screeningState: "APPROVED"
        screeningTag: "NOT_SET"
        imageProcessingState: "PROCESSED"
        artifactDisplayState: "APPROVED"
        description: ""
        mimeType: "image/jpeg"
        originalFilename: "John Wood and Nina Marie Ashby family,.jpg"
        photoTagCount: 14
        commentCount: 1
        contributorPatronId: 12134
        category: "IMAGE"
        uploadDatetime: 1369951872000
        size: 1292239
        iconApid: "TH-301-37918-61-71"
        originalUrl: "https://sg31b0.familysearch.org/service/records/storage/das-mem/patron/v2/TH-301-37918-61-71/original.jpg?ctx=Ar
        thumbUrl: "https://sg31b0.familysearch.org/service/records/storage/das-mem/patron/v2/TH-301-37918-61-71/thumb200.jpg?ctx=ArtCt
        thumbSquareUrl: "https://sg31b0.familysearch.org/service/records/storage/das-mem/patron/v2/TH-301-37918-61-71/thumb200s.jpg?ct
        thumbIconUrl: "https://sg31b0.familysearch.org/service/records/storage/das-mem/patron/v2/TH-301-37918-61-71/thumb64.jpg?ctx=Ar
        thumbMobileUrl: "https://sg31b0.familysearch.org/service/records/storage/das-mem/patron/v2/TH-301-37918-61-71/thumbMobile.jpg?
        thumbTabletUrl: null
        deepZoomUrl: "https://sg31b0.familysearch.org/service/records/storage/dz-mem/dzpatron/v1/TH-301-37918-61-71/image.xml"
        deepZoomLiteUrl: "https://sg31b0.familysearch.org/service/records/storage/dz-mem/dzpatron/v1/TH-301-37918-61-71/scale?width=80
        photoTags: null
        associatedArtifacts: null
        width: 2747
        height: 1797
        dzWidth: 800
        dzHeight: 523
        slug: "john-wood-and-nina-marie-ashby-family-of-10-children"
        visibility: "PUBLIC"
        visitCount: 208
        softTagCount: 1
        contentCategory: "PHOTO"
        structure: "SIMPLE"
        rotationAngle: 0
        language: "en"
        seoIndexable: true
       ▸contentCategories: ["PHOTO"]
        contributorCisUserId: "cis.user.MMMM-J2CC"
        features: ""
        hash: "963b72140afc8c809be004add765295b"
        datesPlaces: null
        archived: false
        first: true
        editableByCaller: false
        uploaderId: 12134
        photoSoftTagCount: 1
     ▸1: {id: 1395104, apid: "TH-303-38357-118-58",…}
     ▸2: {id: 1602405, apid: "TH-303-38756-259-57",…}
     pid: "KWCN-6J4"
    ▾taggedPerson: {id: 819707, personId: null, name: "Leonard Ashby Wood", notes: null, contributorPatronId: 47,…}
      id: 819707
      personId: null
      name: "Leonard Ashby Wood"
      notes: null
      contributorPatronId: 47
      slug: "leonard-ashby-wood"
      imageCount: 0
      storyCount: 0
      relationshipCount: 0
      autoGenerated: false
      legacyPersonId: "KWCN-6J4"
      photoTagIds: null
      seoIndexable: true
      contributorCisUserId: null
      editableByCaller: true
      uploaderId: 47
     ▸ownerTreePersonIds: ["KWCN-6J4"]
     mpid: 819707
     thumbSquareUrl: "https://ps-services-us-east-1-643055571372-tps.s3.amazonaws.com/s3/tps-beta-portrait-store/yzz6/thumb200s.jpg?_
     searchableName: "Leonard Ashby Wood|Leonard Ashby Wood"
     soiPerson: true
     surname: "Wood"
     name: "Leonard Ashby Wood"
     contributorPatronId: 47
     slug: "leonard-ashby-wood"
  ▸1: {living: false, lifespan: "1851–1924", givenName: "Thomas Nelson",…}
  ▸2: {living: false, lifespan: "1915–2012", givenName: "Charles McKay",…}
  ▸3: {living: false, lifespan: "1872–1911", givenName: "Laura May",…}
  ▸4: {living: false, lifespan: "1890–1975", givenName: "Martha",…}
  ▸5: {living: false, lifespan: "1873–1952", givenName: "Henrietta",…}
  ▸6: {living: false, lifespan: "1914–2005", givenName: "Claudia Williams",…}
```



```
▼{data: {,…},…}
  ▼data: {,…}
    ▼artifacts: [{name: "Leonard Ashby Wood",…}, {name:
      ▼[0 … 99]
        ▼0: {name: "Leonard Ashby Wood",…}
           name: "Leonard Ashby Wood"
          ▼featuredImages: [{title: "John Wood and Nina
            ▼0: {title: "John Wood and Nina Marie Ashby
               title: "John Wood and Nina Marie Ashby  F
               thumbSquareUrl: "https://sg31b0.familysea
               __typename: "FeaturedImages"
            ▶1: {title: "",…}
            ▶2: {title: "",…}
           __typename: "MemoriesPerson"
        ▶1: {name: "Thomas Nelson Bleak",…}
        ▶2: {name: "Charles McKay Allred",…}
        ▶3: {name: "Laura May Young", featuredImages: [
        ▶4: {name: "Martha Stout",…}
```

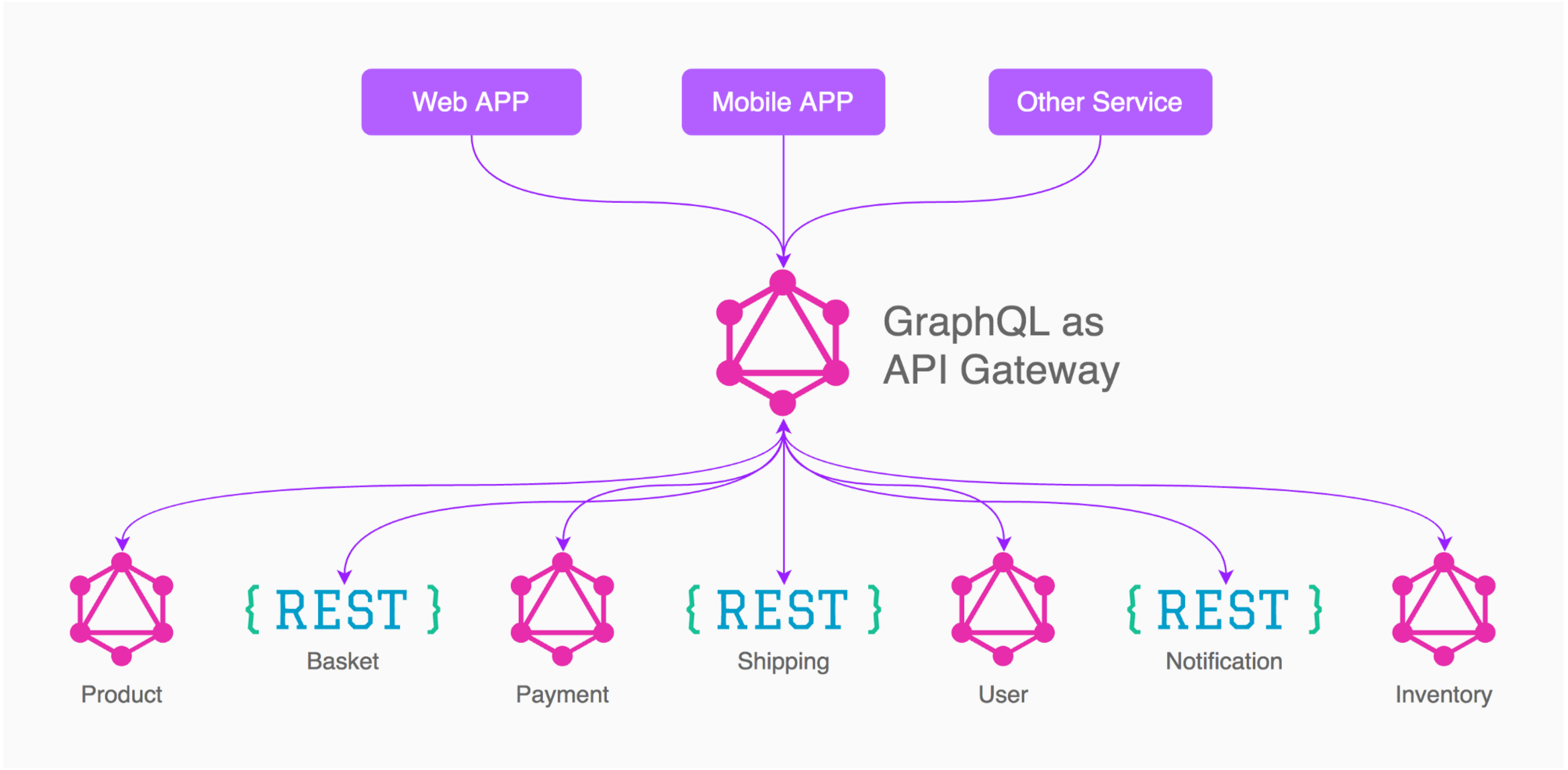
SORT

Abstraction/Stability



SORT

Orchestration/Aggregation



https://labs.getninjas.com.br/sharing-data-in-a-microservices-architecture-using-graphql-97db59357602

SORT

Client Simplification

```
try {
  dispatch(’loading’)
  const user = await fetch('/users/me')
  const friendIDs = await fetch(`/friends/${user.id}`)
  const promises = friendIDs.map((id) => {
    return fetch(`/users/${id}`);
  });
  const friends = await Promise.all(promises);
  const names = friends.map(friend=>friend.name)
  dispatch(‘success’, names)
} catch (error) {
  dispatch(‘error’, error);
}
```

```
const QUERY = gql`
 {
   me {
     friends {
       name
     }
   }
 }
`
const {loading,error,data} = useQuery(QUERY)
```

SORT

Let’s build a GraphQL Server

What do we need?

- Server/middleware (a few options, will use Apollo Server today)
- Schema (defines queries and types of data)
- Resolvers (functions that resolve properties for queries)
- DataSources (code to fetch data from REST APIs for resolvers)

SORT

Live Coding

What could possibly go wrong? ¯_(ツ)_/¯

SORT

Next Steps

- Authentication and Authorization
- Error Handling / Partial Success
- Caching
- Logging / Monitoring / Metrics
- Scaling / Performance / Optimization
- Expand the Schema / Federation

SORT

Thoughts on Schemas

- Schemas are really important, so take time to do them well but avoid analysis paralysis
- It probably shouldn’t look just like your REST JSON or DB tables
- Have a client-centric view, focus on client use cases
- Learn with several smaller experimental schemas before you build large or critical ones
- You will probably get them wrong the first few times, plan for iteration in the beginning
- Start simple, don’t expose every property just because it’s available
- Adding new properties is safe and easy so prefer that over modification or deletion
- This is a new and specialized skill, so grow, plan and focus accordingly
- Deprecation and schema evolution is pretty well handled in GraphQL so embrace it

SORT

Clients

- Apollo Client
- Relay Modern
- Urql
- Draqula, graphql-request, micro-graphql-react, graphql.js, FetchQL, grafoo, etc.
- None: just use fetch, axios, etc.

SORT

Demo Time

We don’t need no stinkin’ wifi

SORT

Different paradigms

REST	GraphQL
Resource/Request-centric	Schema/Data-centric
Server focused	Client focused
Many endpoints	Single endpoint
HTTP Caching	Custom Caching
HTTP Status Codes	200 OK + Errors array

SORT

Learnings/Observations

- Payload from 1.5MB to 20k initial load and 250k total load
- Another went from 140k to 25k (gzipped)
- Generally positive comments, said easy to use and client code feels easier and quicker to write
- Dev workflow is definitely different, sometimes better, sometimes worse
- Concerns about optimizing complex queries and having ”wild west” schema development and confusion
- Can compete / duplicate with existing endpoints / efforts / resources

SORT

GraphQL Considerations

Pros

- Latency/Bandwidth
- Discoverability
- Flexibility
- Consistency
- Client Simplicity

Cons

- New Server/Ops Patterns
- Learning curve
- Maturity
- Handling complex queries

SORT

Q&A

You’ve got questions, we’ve got blank stares

SORT

THE CHURCH OF
JESUS CHRIST
OF LATTER-DAY SAINTS

Information and
Communication Services