GraphQL and Angular Primer

so we can sound smart

STORYTIME!

REST is **awesome**

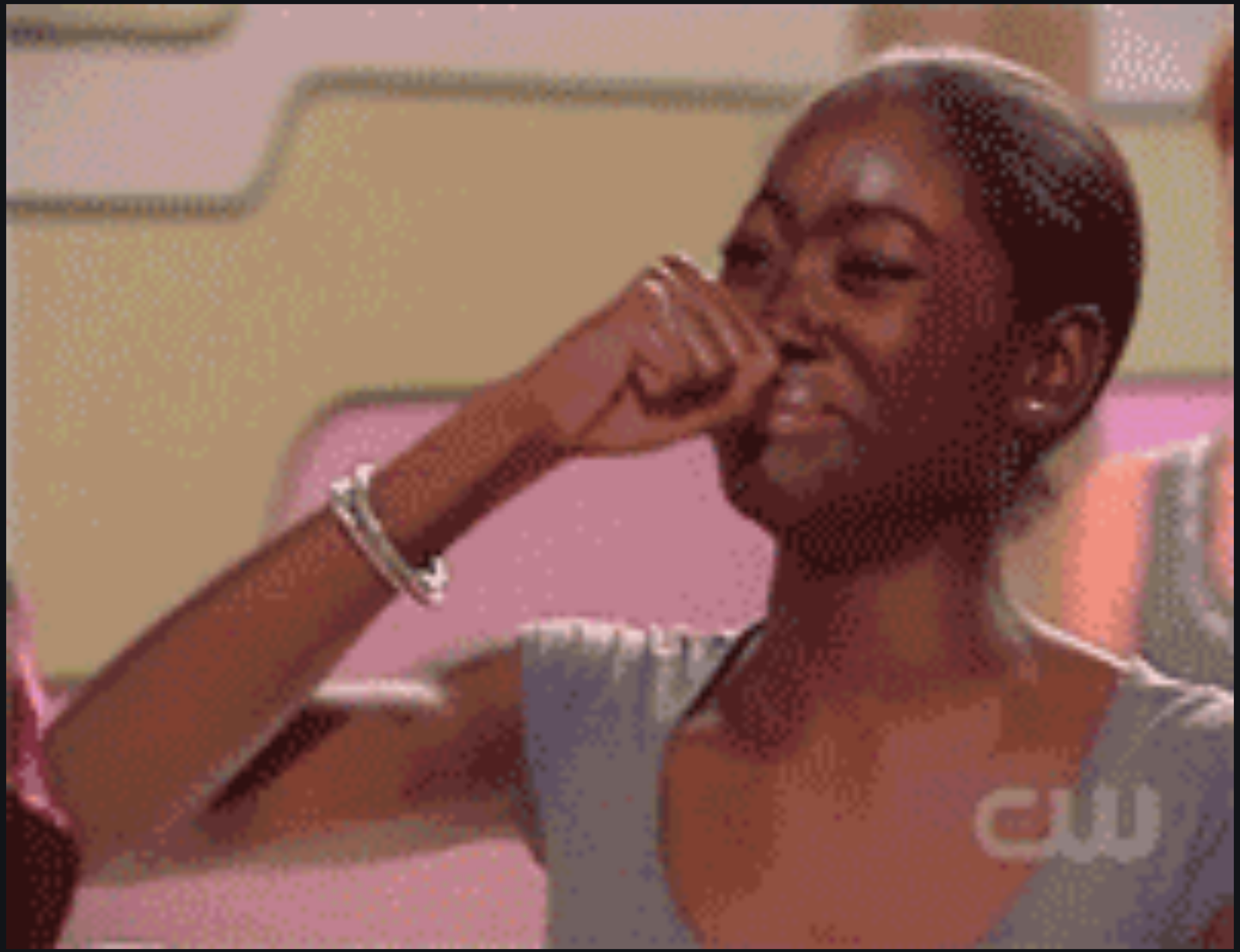
REST is **limited**

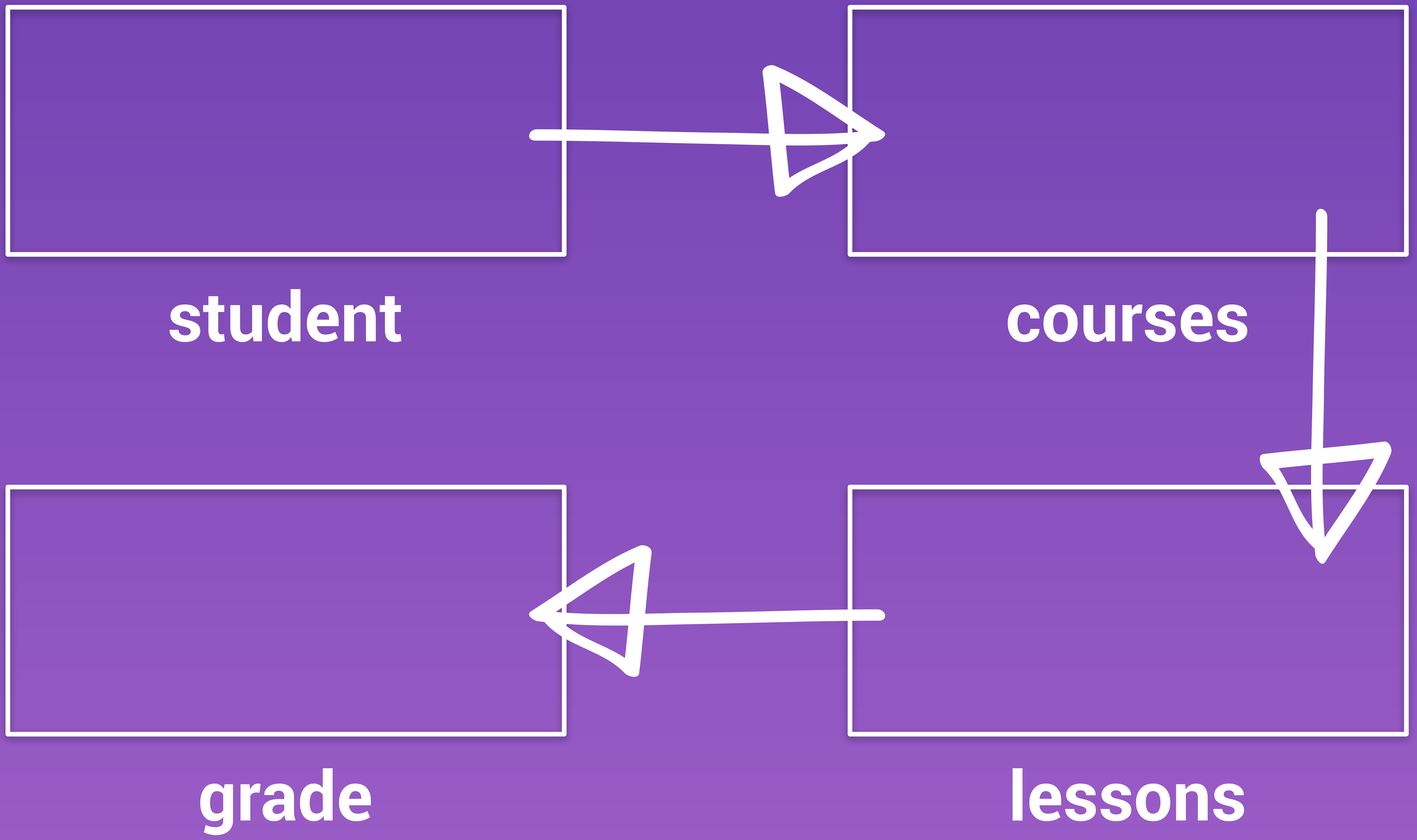
Endpoint for every.single.thing...

that **returns**
every.single.thing...

unless it **needs**
some.other.thing...

and that could require another **hundred*** calls.




student
courses
lessons
grade

$$\begin{tikzcd}
\underset{\textbf{student}}{\boxed{1}} \arrow[r] & \underset{\textbf{courses}}{\boxed{2}} \arrow[d] \\
\underset{\textbf{grade}}{\boxed{61}} & \underset{\textbf{lessons}}{\boxed{30}} \arrow[l]
\end{tikzcd}$$

```json
{
  "standardColumns": [
  {
    "id": 1,
    "displayName": "Account holder name",
    "columnName": "ACCT_CLIENT_SHORT_NAME",
    "columnType": "STRING",
    "entityName": "account",
    "entityPath": null,
    "decimalPlaces": 0,
    "tooltip": null,
    "viewOnScreen": false,
    "columnIndex": 1,
    "sortOrder": "NOT_SPECIFIED",
    "excelOnly": false,
    "expressionName": null
  },
  {
    "id": 2,
    "displayName": "Account number",
    "columnName": "ACCT_NUM",
    "columnType": "STRING",
    "entityName": "account",
    "entityPath": null,
    "decimalPlaces": 0,
    "tooltip": null,
    "viewOnScreen": false,
    "columnIndex": 2,
    "sortOrder": "NOT_SPECIFIED",
    "excelOnly": false,
    "expressionName": null
  }
],
  "relationshipEvaluationList": [
  {
    "relationshipId": "ACCT44438948",
    "relationshipName": "test_relationship_name",
    "value": 2,
    "primaryInsight": "IRA Contributions Not Made",
    "evaluations": [
      {
        "id": 199,
        "batch": {
          "id": 1,
          "startDate": 1444060272377,
          "endDate": 1444060283358,
          "status": "COMPLETE"
        },
        "ruleSet": {
          "personId": 0,
          "createdTs": null,
          "updatedTs": null,
          "id": 1,
          "ruleSetName": null,
          "rules": [],
          "conjunction": null,
          "reason": null,
          "evaluationInterval": "SIXTYMINUTE",
          "startDate": null,
          "endDate": null,
          "status": "ACTIVE"
        },
        "executionDate": 1444060282824,
        "contextData": "{\"ACCOUNT\":{\"ACCT_NUM\":\"44438948\",\"ACCT_CLIENT_RLTNP_TIER_NAME\":\"*Unknown\",\"ACCT_CLIENT_SHORT_NAME\":\"Stempek William\",\"ACCT_CLIENT_RLTNP_NAME\":\"Stempek William (44438948)\"}}",
        "relationshipId": "ACCT44438948",
        "faNumber": "4143",
        "insightName": "IRA Contributions Not Made",
        "insightId": 1,
        "rawRanking": 88877896,
        "ranking": 5,
        "customColumns": [
          {
            "id": 15,
            "displayName": "Account holder name",
            "columnName": "ACCT_CLIENT_SHORT_NAME",
            "columnType": "STRING",
            "entityName": "account",
            "entityPath": null,
            "decimalPlaces": 0,
            "tooltip": null,
            "viewOnScreen": false,
            "columnIndex": 1,
            "sortOrder": "NOT_SPECIFIED",
            "excelOnly": false,
            "expressionName": null
          },
          {
            "id": 16,
            "displayName": "Account number",
            "columnName": "ACCT_NUM",
            "columnType": "STRING",
            "entityName": "account",
            "entityPath": null,
            "decimalPlaces": 0,
            "tooltip": null,
            "viewOnScreen": false,
            "columnIndex": 2,
            "sortOrder": "NOT_SPECIFIED",
            "excelOnly": false,
            "expressionName": null
          }
        ]
      }
    ]
  }
]
}
```

What you get

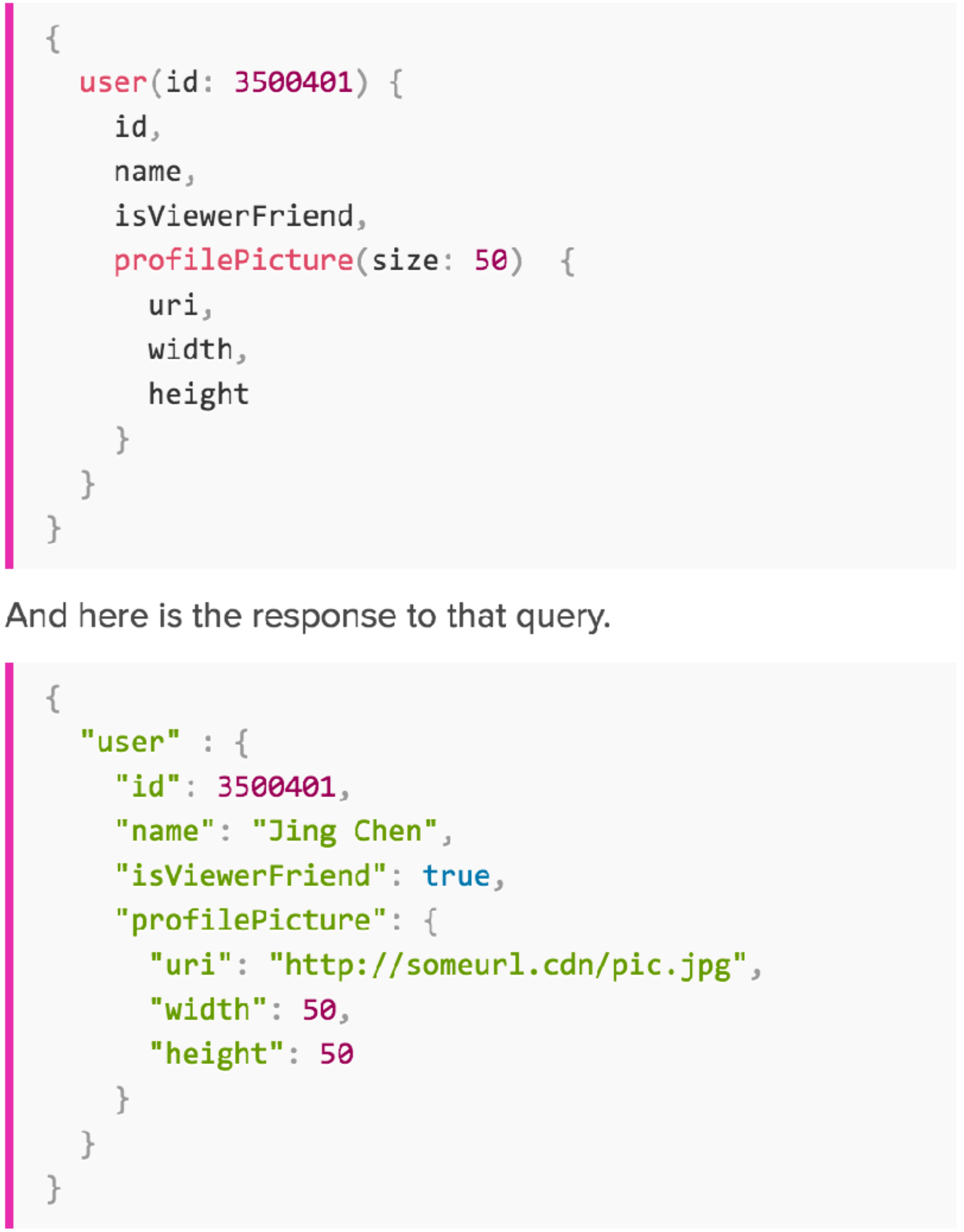
```
{
  "standardColumns": [
  {
    "id": 1,
    "displayName": "Account holder name",
    "columnName": "ACCT_CLIENT_SHORT_NAME",
    "columnType": "STRING",
    "entityName": "account",
    "entityPath": null,
    "decimalPlaces": 0,
    "tooltip": null,
    "viewOnScreen": false,
    "columnIndex": 1,
    "sortOrder": "NOT_SPECIFIED",
    "excelOnly": false,
    "expressionName": null
  },
  {
    "id": 2,
    "displayName": "Account number",
    "columnName": "ACCT_NUM",
    "columnType": "STRING",
    "entityName": "account",
    "entityPath": null,
    "decimalPlaces": 0,
    "tooltip": null,
    "viewOnScreen": false,
    "columnIndex": 2,
    "sortOrder": "NOT_SPECIFIED",
    "excelOnly": false,
    "expressionName": null
  }
],
  "relationshipEvaluationList": [
  {
    "relationshipId": "ACCT44438948",
    "relationshipName": "test_relationship_name",
    "value": 2,
    "primaryInsight": "IRA Contributions Not Made",
    "evaluations": [
      {
        "id": 199,
        "batch": {
          "id": 1,
          "startDate": 1444060272377,
          "endDate": 1444060283358,
          "status": "COMPLETE"
        },
        "ruleSet": {
          "personId": 0,
          "createdTs": null,
          "updatedTs": null,
          "id": 1,
          "ruleSetName": null,
          "rules": [],
          "conjunction": null,
          "reason": null,
          "evaluationInterval": "SIXTYMINUTE",
          "startDate": null,
          "endDate": null,
          "status": "ACTIVE"
        },
        "executionDate": 1444060282824,
        "contextData": "{\"ACCOUNT\":{\"ACCT_NUM\":\"44438948\",\"ACCT_CLIENT_RLTNP_TIER_NAME\":\"*Unknown\",\"ACCT_CLIENT_SHORT_NAME\":\"Stempek William\",\"ACCT_CLIENT_RLTNP_NAME\":\"Stempek William (44438948)\"}}",
        "relationshipId": "ACCT44438948",
        "faNumber": "4143",
        "insightName": "IRA Contributions Not Made",
        "insightId": 1,
        "rawRanking": 88877896,
        "ranking": 5,
        "customColumns": [
          {
            "id": 15,
            "displayName": "Account holder name",
            "columnName": "ACCT_CLIENT_SHORT_NAME",
            "columnType": "STRING",
            "entityName": "account",
            "entityPath": null,
            "decimalPlaces": 0,
            "tooltip": null,
            "viewOnScreen": false,
            "columnIndex": 1,
            "sortOrder": "NOT_SPECIFIED",
            "excelOnly": false,
            "expressionName": null
          },
          {
            "id": 16,
            "displayName": "Account number",
            "columnName": "ACCT_NUM",
            "columnType": "STRING",
            "entityName": "account",
            "entityPath": null,
            "decimalPlaces": 0,
            "tooltip": null,
            "viewOnScreen": false,
            "columnIndex": 2,
            "sortOrder": "NOT_SPECIFIED",
            "excelOnly": false,
            "expressionName": null
          }
        ]
      }
    ]
  }
]
}
```

What you need

API design is **hard!!!**

IN A PERFECT WORLD

- declare your data needs like you think about it
- Know in advance what you could fetch
- Decoupling from the server
- Each Components will declare its own data needs and it will merge into one round trip
- Single endpoint



```
{
  user(id: 3500401) {
    id,
    name,
    isViewerFriend,
    profilePicture(size: 50)  {
      uri,
      width,
      height
    }
  }
}
```

And here is the response to that query.

```
{
  "user" : {
    "id": 3500401,
    "name": "Jing Chen",
    "isViewerFriend": true,
    "profilePicture": {
      "uri": "http://someurl.cdn/pic.jpg",
      "width": 50,
      "height": 50
    }
  }
}
```

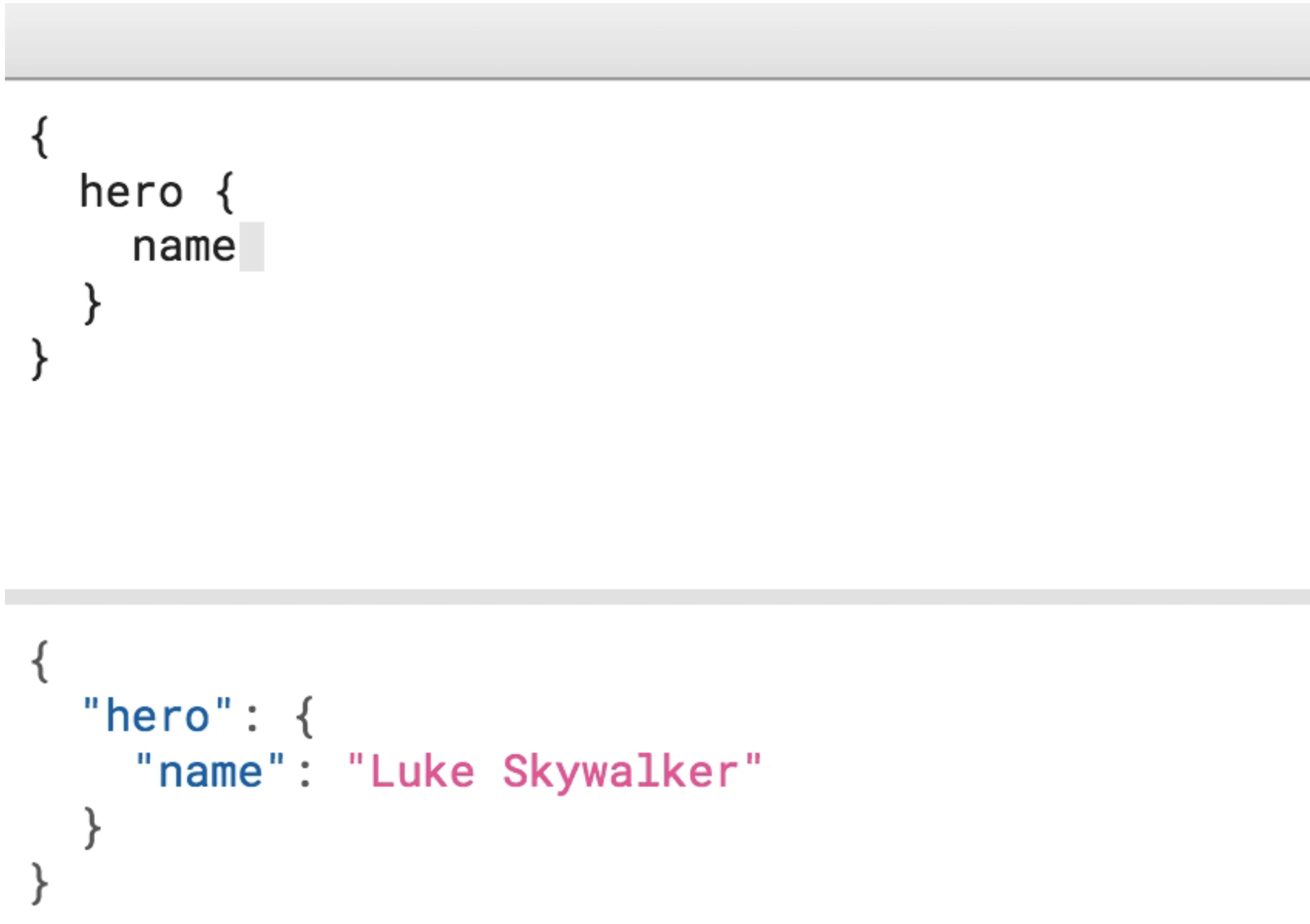
facebook
700 M
1.4 B
600 M
300 M

GraphQL: an API query language

GraphQL is **flexible**

GRAPHQL

- Client **asks** for and **gets exactly** what it needs



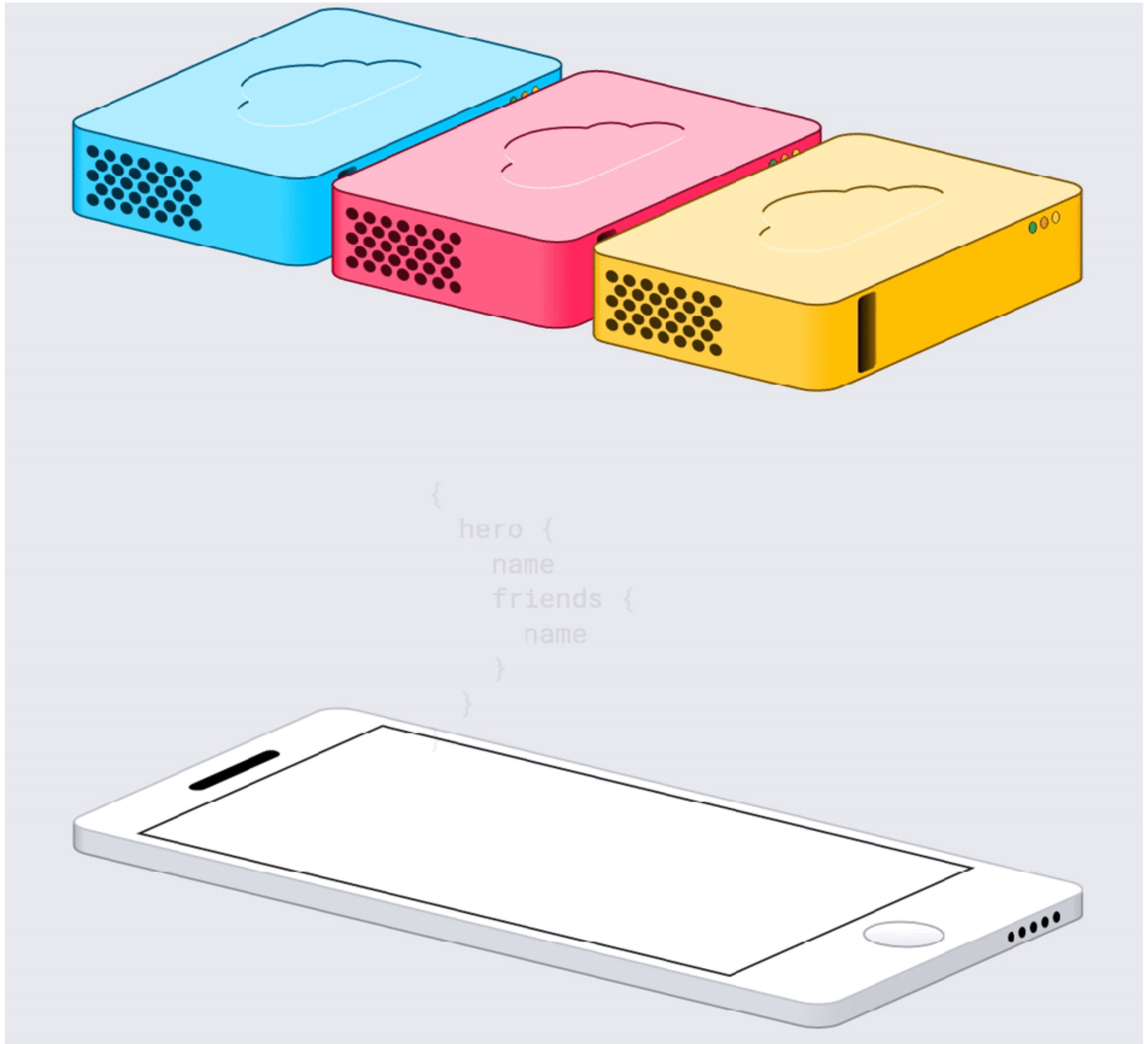
```
{
  hero {
    name
  }
}
```

```
{
  "hero": {
    "name": "Luke Skywalker"
  }
}
```

GraphQL is **performant**

G R A P H Q L

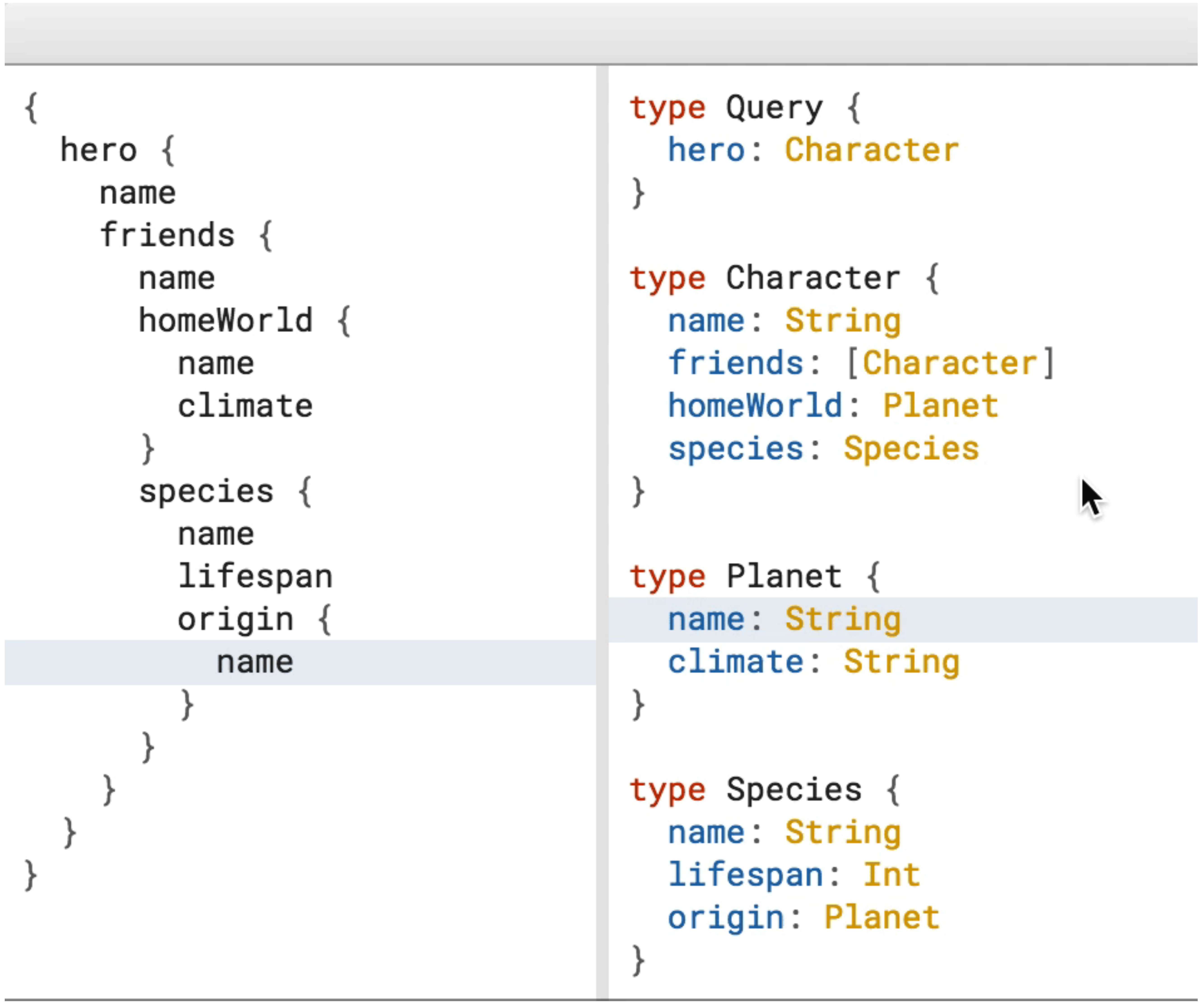
- **Multiple resources** in **single** request



```
{
  hero {
    name
    friends {
      name
    }
  }
}
```

GRAPHQL

- **Typed** API



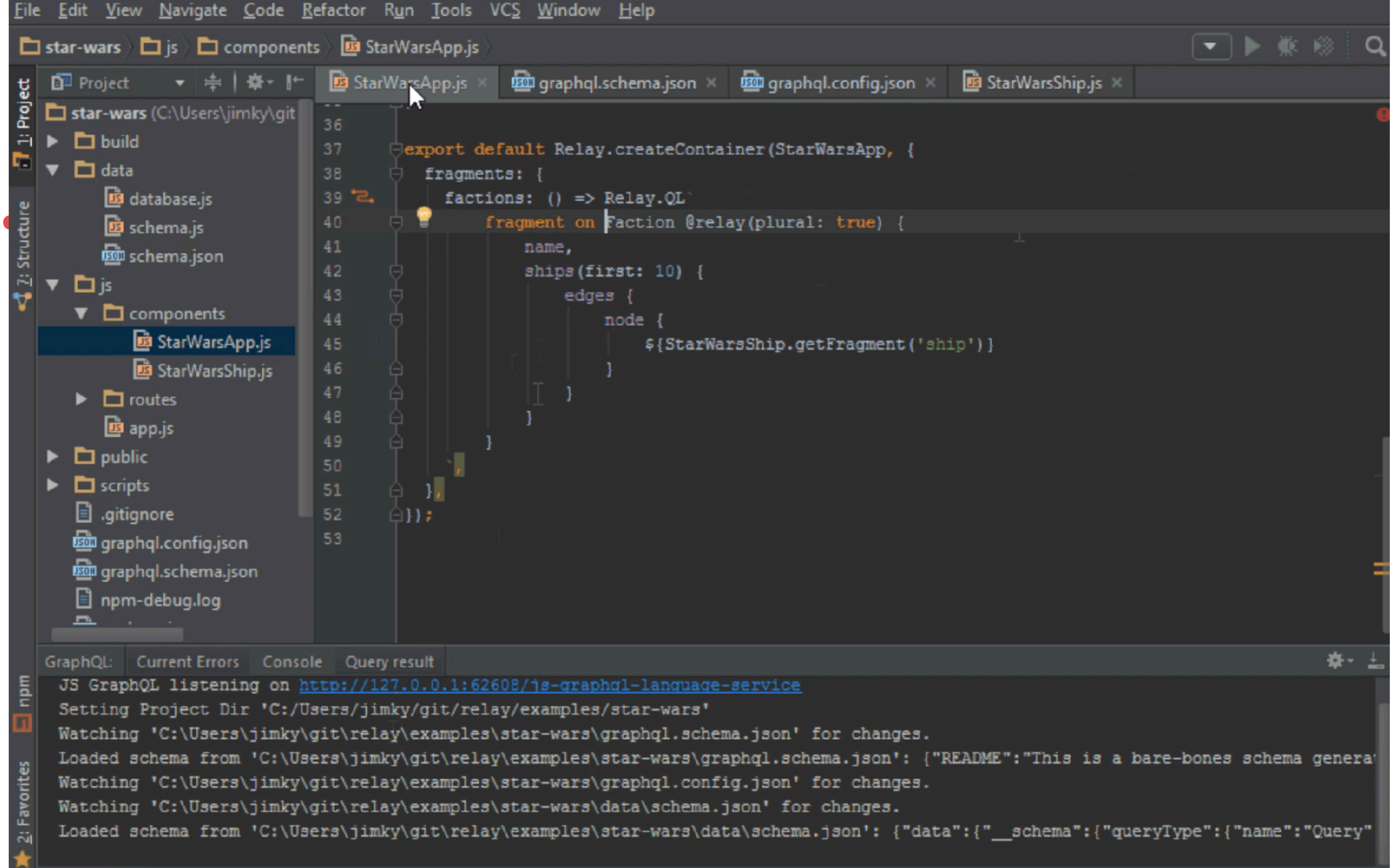
```
{
  hero {
    name
    friends {
      name
      homeWorld {
        name
        climate
      }
      species {
        name
        lifespan
        origin {
          name
        }
      }
    }
  }
}
```

```
type Query {
  hero: Character
}

type Character {
  name: String
  friends: [Character]
  homeWorld: Planet
  species: Species
}

type Planet {
  name: String
  climate: String
}

type Species {
  name: String
  lifespan: Int
  origin: Planet
}
```



```

export default Relay.createContainer(StarWarsApp, {
  fragments: {
    factions: () => Relay.QL`
      fragment on Faction @relay(plural: true) {
        name,
        ships(first: 10) {
          edges {
            node {
              ${StarWarsShip.getFragment('ship')}
            }
          }
        }
      }
    `,
  },
});

```

```
JS GraphQL listening on http://127.0.0.1:62608/js-graphql-language-service
Setting Project Dir 'C:/Users/jimky/git/relay/examples/star-wars'
Watching 'C:\Users\jimky\git\relay\examples\star-wars\graphql.schema.json' for changes.
Loaded schema from 'C:\Users\jimky\git\relay\examples\star-wars\graphql.schema.json': {"README":"This is a bare-bones schema genera
Watching 'C:\Users\jimky\git\relay\examples\star-wars\graphql.config.json' for changes.
Watching 'C:\Users\jimky\git\relay\examples\star-wars\data\schema.json' for changes.
Loaded schema from 'C:\Users\jimky\git\relay\examples\star-wars\data\schema.json': {"data":{"__schema":{"queryType":{"name":"Query"
```

GraphQL is **not** a Storage Engine

GraphQL leverages your

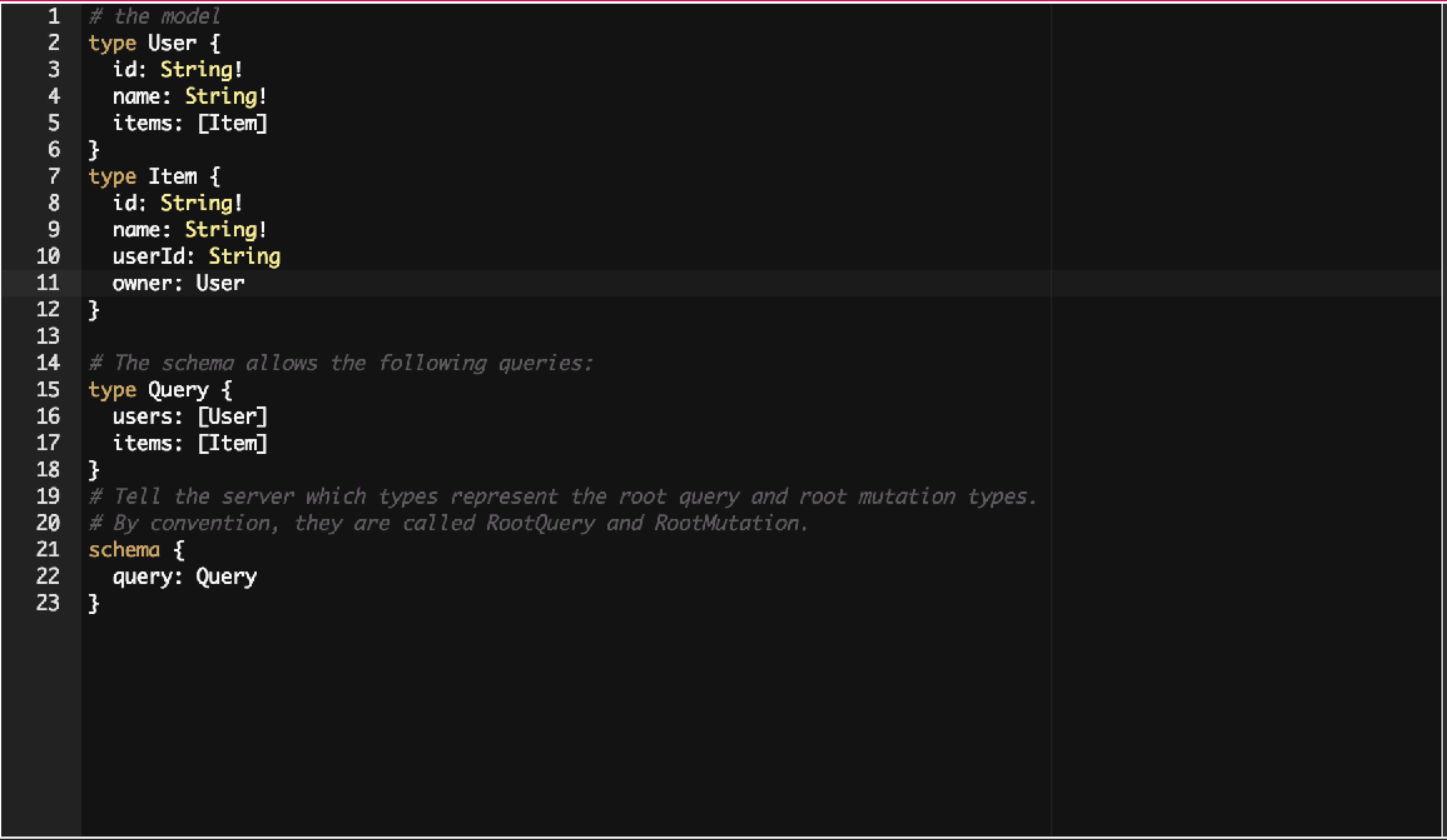
existing code

APOLLO

Apollo Client

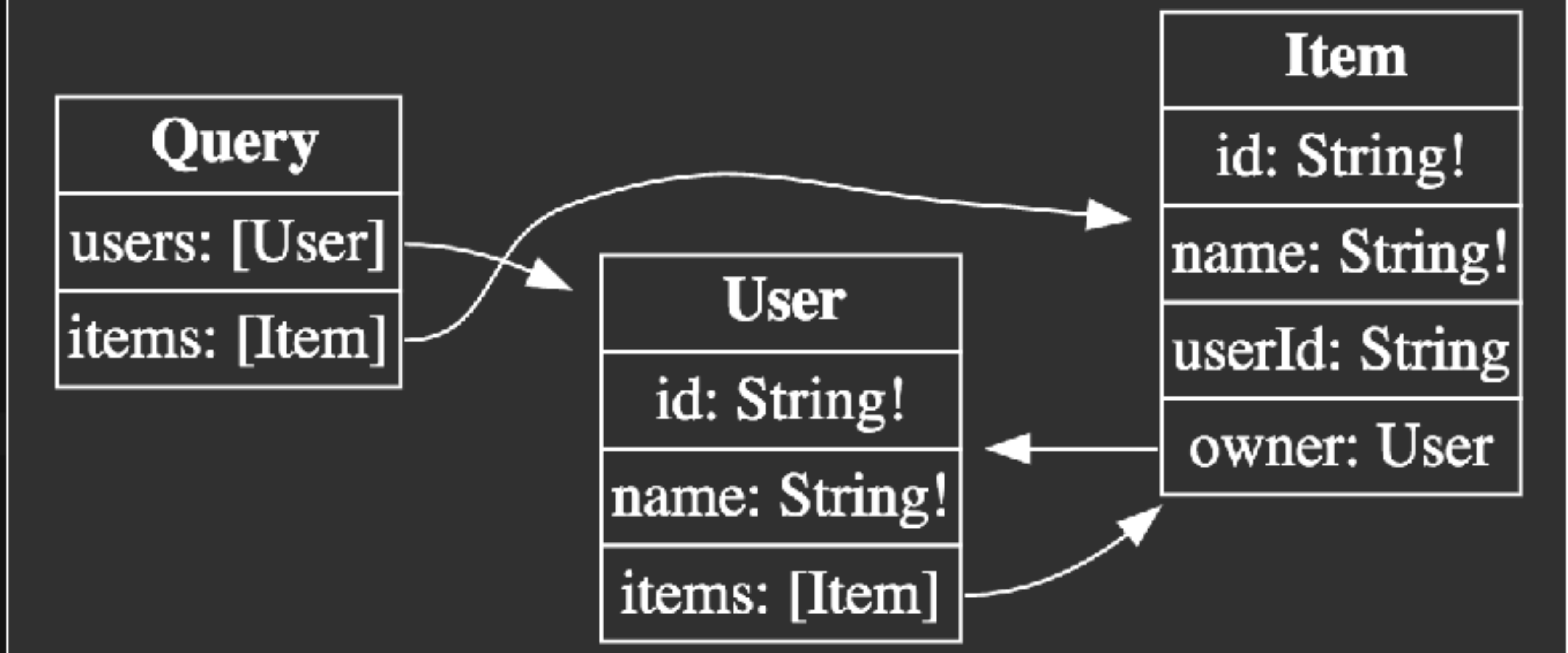
- Help network interactions feel instant
- Small, Open Source, Client Only
- Integrated Gradually
- Pluggable Immutable Cache (Redux + RxJS)
- Modern Data Standard - GraphQL

https://github.com/onehungrymind/graphql-simple-app



```graphql
# the model
type User {
  id: String!
  name: String!
  items: [Item]
}
type Item {
  id: String!
  name: String!
  userId: String
  owner: User
}

# The schema allows the following queries:
type Query {
  users: [User]
  items: [Item]
}
# Tell the server which types represent the root query and root mutation types.
# By convention, they are called RootQuery and RootMutation.
schema {
  query: Query
}
```



```
import { ApolloModule } from 'apollo-angular';
import { getClient } from './client';

imports: [
  BrowserModule,
  FormsModule,
  HttpModule,
  MaterialModule.forRoot(),
  StoreModule.provideStore({users, items}),
  ApolloModule.forRoot(getClient)
]
```

ApolloModule

```
import { ApolloClient } from 'apollo-client';
import { networkInterface } from './network-interface';

const client = new ApolloClient({
    networkInterface,
    dataIdFromObject: (object: any) => object.__typename + object.id,
});
export function getClient(): ApolloClient {
    return client;
}
```

ApolloClient

```
import { createNetworkInterface } from 'apollo-client';

const networkInterface = createNetworkInterface(
  {
    uri: 'http://localhost:3000/graphql'
  });

export {
  networkInterface
}
```

networkInterface

```
import gql from 'graphql-tag';

const usersQuery = gql`
  query users {
    users {
      id
      name
    }
  }
`;
```

graphql-tag

```typescript
users$: Observable<User[]> = this.apollo.watchQuery({
  query: usersQuery
}).map((result: any) => result.data.users);

constructor(
    private usersService: UsersService,
    private apollo: Apollo) {}
```

watchQuery

```
const itemsQuery = gql`
  query items {
    items {
      id
      name
      owner {
        id
      }
    }
  }
`;
```

Nested Query

```
const usersItemsQuery = gql`
  query usersItems {
    users {
      id
      name
      items {
        id
        name
      }
    }
  }
`;
```

Nested Query

https://learngraphql.com/
http://dev.apollodata.com/angular2/
https://scaphold.io/community/learn/
https://egghead.io/courses/build-a-graphql-server

@simpulton

I REQUEST THE HIGHEST OF FIVES

Thanks!