https://github.com/phillip-kruger/membership

```
$ git clone https://github.com/phillip-kruger/membership.git
$ cd membership
$ mvn clean install
$ firefox http://localhost:8080/membership/
```

History

- Developed and open sourced by Facebook
- Specification (http://facebook.github.io/graphql/)
- Alternative to REST
- Not data storage (GraphDB)
- Declarative data fetching
  - Increased mobile usage
  - Variety of different frontend frameworks
  - Rapid feature development
- Since 2012. Publically 2015

Community adoption

Credit: http://graphql.org/users/

REST vs GraphQL

HTTP GET

```
{
    "user": {
        "id": "er3tg439frjw"
        "name": "Mary",
        "address": { … },
        "birthday": "July 26, 1982"
    }
}
```

/users/<id>

/users/<id>/posts

/users/<id>/followers

HTTP GET

```
{
    "posts": [{
        "id": "ncwon3ce89hs"
        "title": "Learn GraphQL today",
        "content": "Lorem ipsum … ",
        "comments": [ … ],
    }]
```
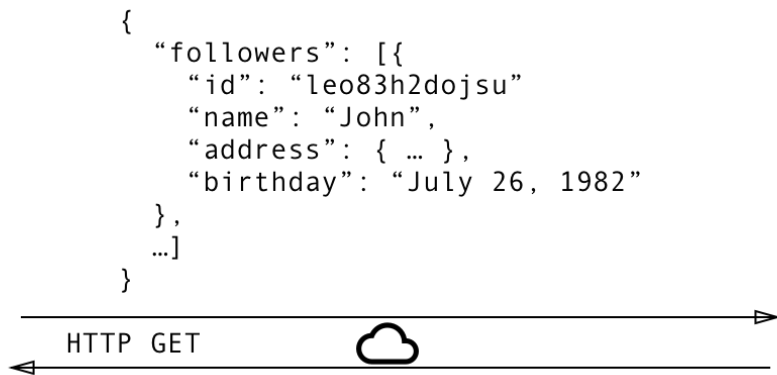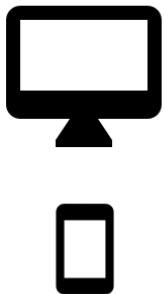
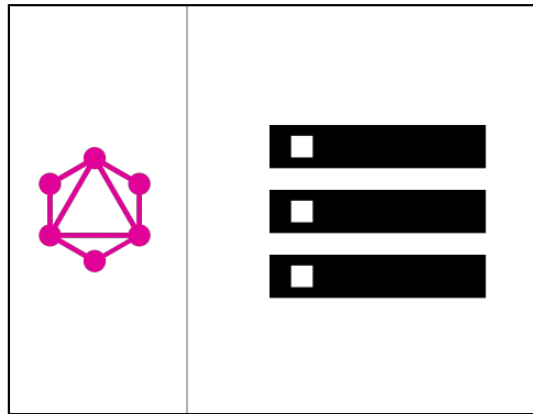/users/<id>

/users/<id>/posts

/users/<id>/followers
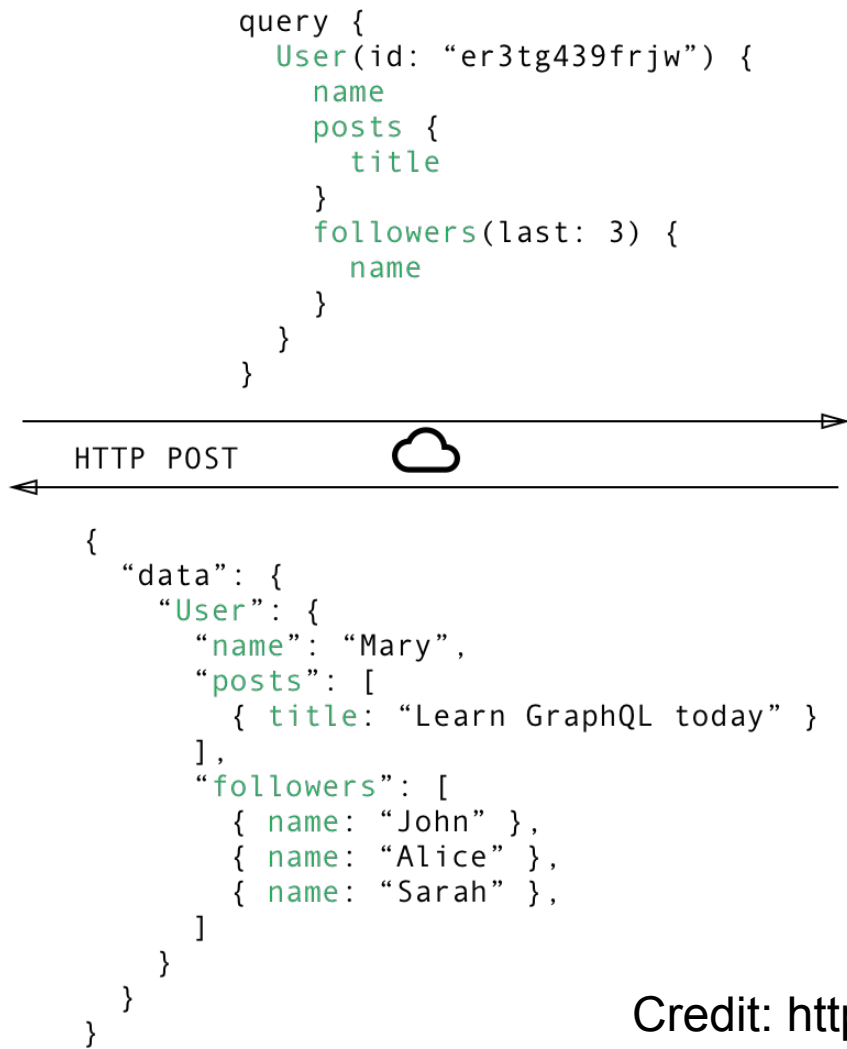
Credit: https://www.howtographql.com/

REST vs GraphQL

```graphql
query {
    User(id: "er3tg439frjw") {
        name
        posts {
            title
        }
        followers(last: 3) {
            name
        }
    }
}
```

HTTP POST

```json
{
    "data": {
        "User": {
            "name": "Mary",
            "posts": [
                { title: "Learn GraphQL today" }
            ],
            "followers": [
                { name: "John" },
                { name: "Alice" },
                { name: "Sarah" },
            ]
        }
    }
}
```

Credit: https://www.howtographql.com/

GraphQL

Benefits

- No more Over- and Under Fetching
- Rapid Product Iterations on the Frontend
- Insightful Analytics on the Backend
- Benefits of a Schema & Type System

```
type Person {
    name: String!
    age: Int!
}
```

GraphiQL

GraphiQL ▶

```
1 ▼ query TodoAppQuery($n: Int) {
2 ▼   globalTodoList {
3 ▼     items(first:$n) {
4 ▼       edges {
5           node {
6             text
7             complete
8           }
9         }
10      }
11    }
12  }
```

QUERY VARIABLES

```
1  {
2    "n": 2
3  }
```
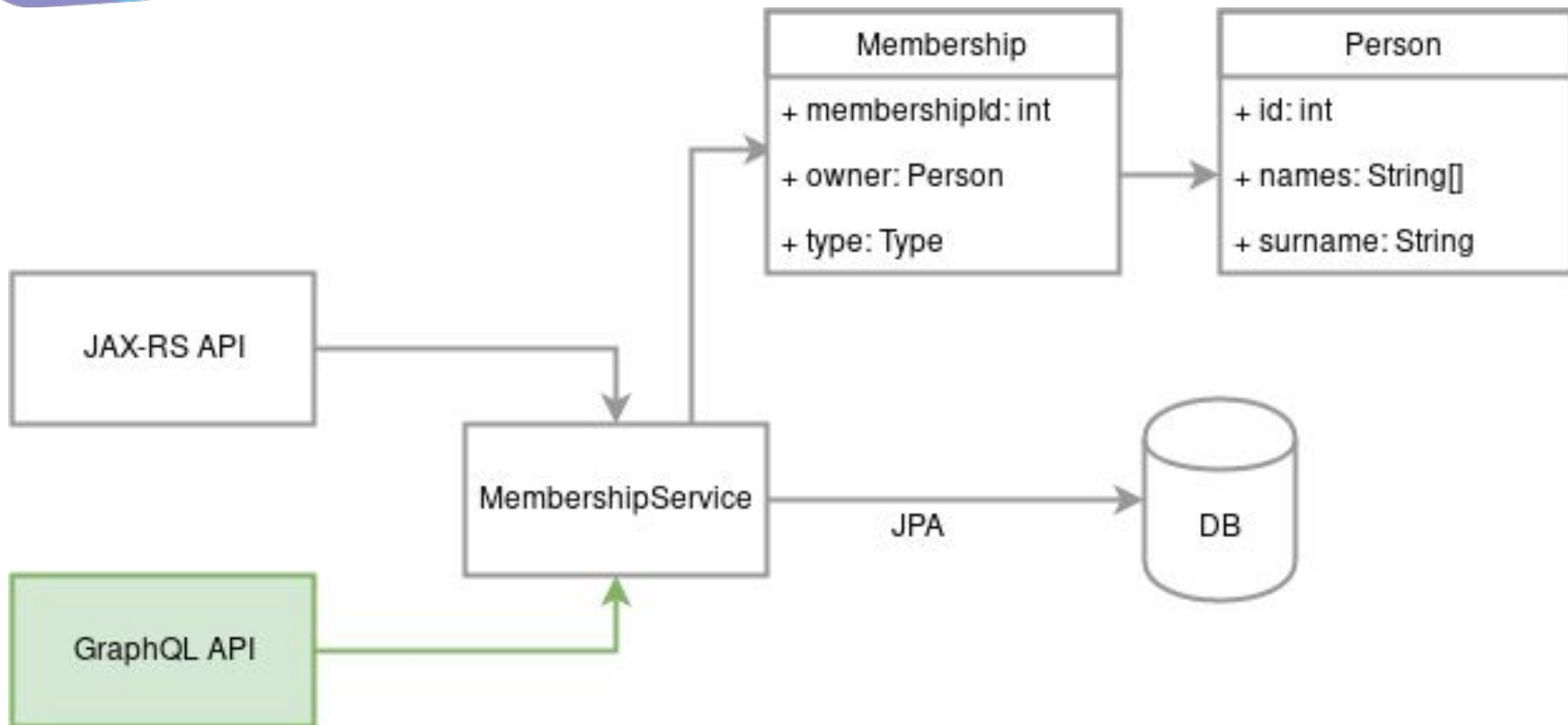
```
{
  "data": {
    "globalTodoList": {
      "items": {
        "edges": [
          {
            "node": {
              "text": "Release GraphiQL",
              "complete": true
            }
          },
          {
            "node": {
              "text": "Attend @Scale 2015",
              "complete": false
            }
          }
        ]
      }
    }
  }
}
```

Example application

- https://github.com/phillip-kruger/membership
- Build on existing JAX-RS Application
- Use annotations to generate schema

Membership
+ membershipId: int
+ owner: Person
+ type: Type

Person
+ id: int
+ names: String[]
+ surname: String

JAX-RS API

MembershipService

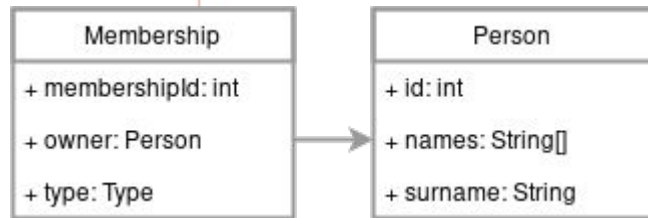GraphQL API

JPA

DB

{ REST }

Example application

```java
@Data @AllArgsConstructor @NoArgsConstructor
@Entity
@XmlRootElement @XmlAccessorType(XmlAccessType.FIELD)
@NamedQueries({
    @NamedQuery(name = Membership.QUERY_FIND_ALL, query = "SELECT m FROM Membership m"),
    @NamedQuery(name = Membership.QUERY_FIND_ALL_TYPE, query = "SELECT m FROM Membership m WHERE m.type=:type")
})
public class Membership implements Serializable {
    private static final long serialVersionUID = -8531040143398373846L;

    public static final String QUERY_FIND_ALL = "Membership.findAll";
    public static final String QUERY_FIND_ALL_TYPE = "Membership.findAllType";

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int membershipId;

    @OneToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JoinColumn(name = "owner_id")
    @NotNull(message = "Owner can not be empty")
    private Person owner;

    @NotNull(message = "Type can not be empty")
    private Type type;

}
```

Membership
+ membershipId: int
+ owner: Person
+ type: Type

Person
+ id: int
+ names: String[]
+ surname: String

Example application

http://localhost:8080/membership/rest

{ REST }

```java
@Log
@Path("/")
public class MembershipRestApi {

    @Inject
    private MembershipService membershipService;

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public List<Membership> getAllMemberships(){
        return membershipService.getAllMemberships();
    }

    @GET @Path("{id}")
    @Produces(MediaType.APPLICATION_JSON)
    public Membership getMembership(@NotNull @PathParam(value = "id") int id){
        return membershipService.getMembership(id);
    }

    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    public void joinMember(@NotNull Membership membership) {
        membershipService.createMembership(membership);
    }
}
```

JAX-RS API

MembershipService

Example application
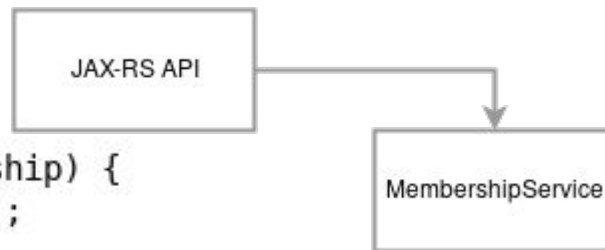
```java
@RequestScoped
public class MembershipGraphQLApi {

    @Inject
    private MembershipService membershipService;

    @GraphQLQuery(name = "memberships")
    public List<Membership> getAllMemberships() {
        return getAllMemberships(Optional.empty());
    }

    @GraphQLQuery(name = "memberships")
    public List<Membership> getAllMemberships(Optional<MembershipFilter> filter){
        return getAllMemberships(filter,Optional.empty(),Optional.empty());
    }

    @GraphQLQuery(name = "memberships")
    public List<Membership> getAllMemberships(@GraphQLArgument(name = "skip") Optional<Integer> skip,
                                              @GraphQLArgument(name = "first") Optional<Integer> first){
        return getAllMemberships(Optional.empty(),skip,first);
    }

    @GraphQLQuery(name = "memberships")
    public List<Membership> getAllMemberships(Optional<MembershipFilter> filter,
                             @GraphQLArgument(name = "skip") Optional<Integer> skip,
                             @GraphQLArgument(name = "first") Optional<Integer> first) {
        return membershipService.getAllMemberships(filter, skip, first);
    }

    @GraphQLQuery(name = "membership")
    public Membership getMembership(@GraphQLArgument(name = "membershipId") int
```

MembershipService

GraphQL API

GraphQL Query

```
{
    memberships {
        owner {
            names
        }
        type
    }
}
```

```
{
    membership(membershipId:2){
        owner{
            names
            surname
        }
    }
}
```

Fragments & Named

```
query Memberships {
    memberships{
        ...fullMembership
    }
}
```

```
fragment fullMembership on Membership
{
        membershipId
    owner{
        ...owner
    }
    type
}
```

```
fragment owner on Person {
    id
    names
    surname
}
```

Filtering

```
query FilteredMemberships {
    memberships(filter:{
        type:FREE
    }){
        ...fullMembership
    }
}
```

```
query FilteredMemberships {
    memberships(filter:{
    surnameContains: "Kru"

    }){
    ...fullMembership
    }
}
```

Variables

```
query Membership($id:Int!) {

membership(membershipId:$id){
        ...fullMembership
    }
}
```

```
{
    "id":1
}
```

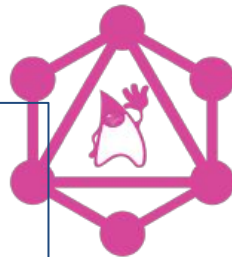**Mutation (add)**

```
mutation CreateMember($membership: MembershipInput!) {
    createMembership(membership:$membership) {
        ...fullMembership
    }
}
```

```
{
    "membership": {
        "owner": {
            "names": "Christina",
            "surname": "Storm"
        },
        "type": "FULL"
    }
}
```
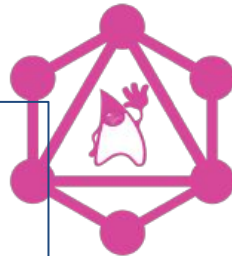
Mutation (edit)

```
mutation EditMember($membership: MembershipInput!) {
    createMembership(membership:$membership) {
        ...fullMembership
    }
}
```

```
{
    "membership": {
        "membershipId": 2,
        "owner": {
            "names": [
                "Charmaine",
                "Juliet"
            ],
            "surname": "Krüger"
        },
        "type": "FULL"
    }
}
```

Mutation (delete)

```
mutation DeleteMembership($id:Int!){
    deleteMembership(membershipId:$id){
        ...fullMembership
    }
}
```

```
{
    "id":2
}
```

```
{
    "membership": {
        "membershipId": 2,
        "owner": {
            "names": [
                "Charmaine",
                "Juliet"
            ],
            "surname": "Krüger"
        },
        "type": "FULL"
    }
}
```

Build-in client validation
Custom Handler (eg. ConstraintViolationException)

Errors

```graphql
mutation CreateMember($membership: MembershipInput!) {
    membership(membership:$membership) {
        ...fullMembership
    }
}
```

```json
{
    "membership": {
        "owner": {
            "names": "Christina",
            "surname": "S"
        },
        "type": "FULL"
    }
}
```
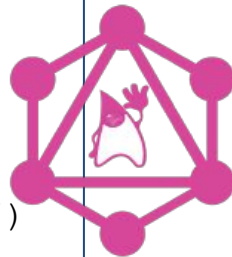
```java
@Size(min=2, message = "Surname '${validatedValue}' is too short, minimum {min} characters")
private String surname;
```

Introspection

```
{
    __schema {
        queryType {
            name
            fields {
                name
            }
        }
        mutationType{
            name
            fields{
                name
            }
        }
        subscriptionType {
            name
            fields{
                name
            }
        }
    }
}
```

```
{
    __type(name: "Membership")
{
        name
        kind
        fields {
            name
            args {
                name
            }
        }
    }
}
```

Other

- Directives (include/skip fields based on condition)
  - @include(if: Boolean)
  - @skip(if: Boolean)
- Pagination
  - memberships (first:2 after:4)
  - memberships (first:2 offset:6)
- Subscriptions
  - Realtime updates (event stream)
- Conditions in filter (OR & AND)
- Java Client

GraphQL

Links

http://graphql.org/
http://facebook.github.io/graphql/
https://www.howtographql.com/
https://github.com/graphql-java/graphql-java
http://graphql-java.readthedocs.io/en/v7/
https://github.com/graphql-java/graphql-java-servlet
https://github.com/leangen/GraphQL-SPQR
https://github.com/graphql/graphiql
https://github.com/phillip-kruger/membership