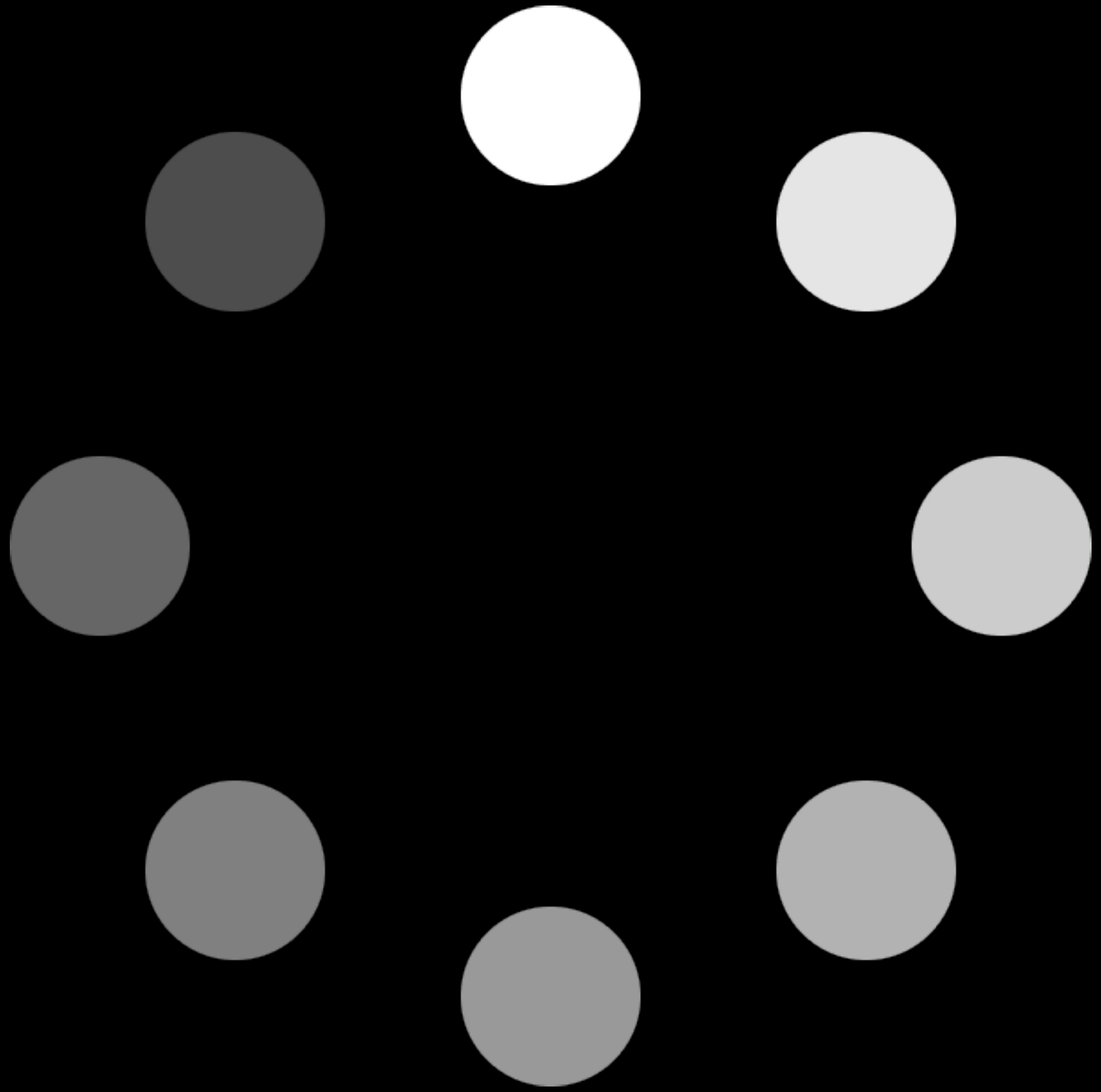


5 Antipatterns, that slowed down our React/GraphQL app

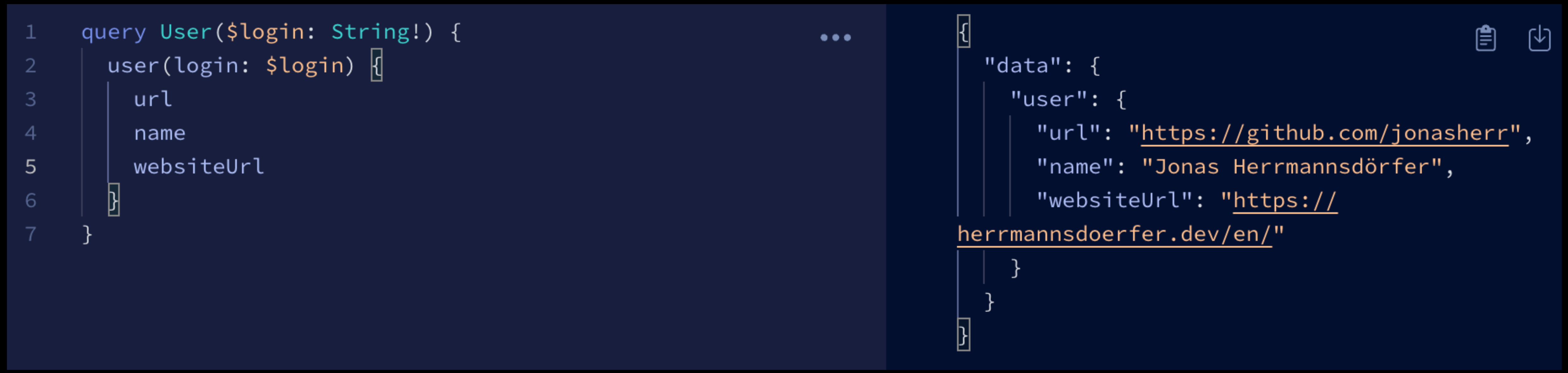
(And how we fixed them)

What is GraphQL?

- query language for your API
- and a server-side runtime for executing queries

Advantages of GraphQL

- Ask for what you need, get exactly that

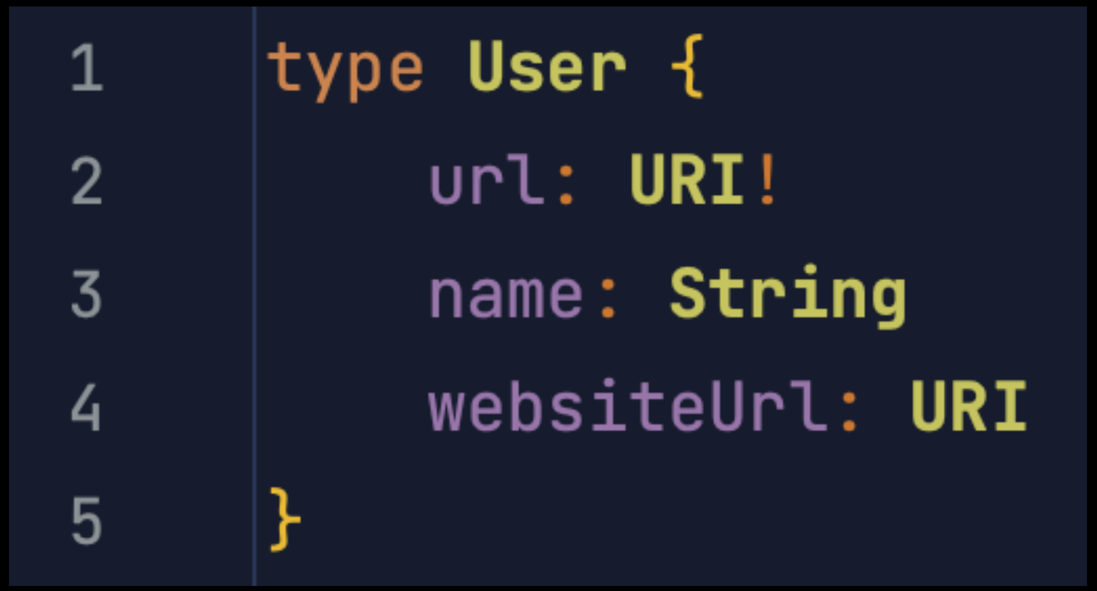


```graphql
query User($login: String!) {
  user(login: $login) {
    url
    name
    websiteUrl
  }
}
```

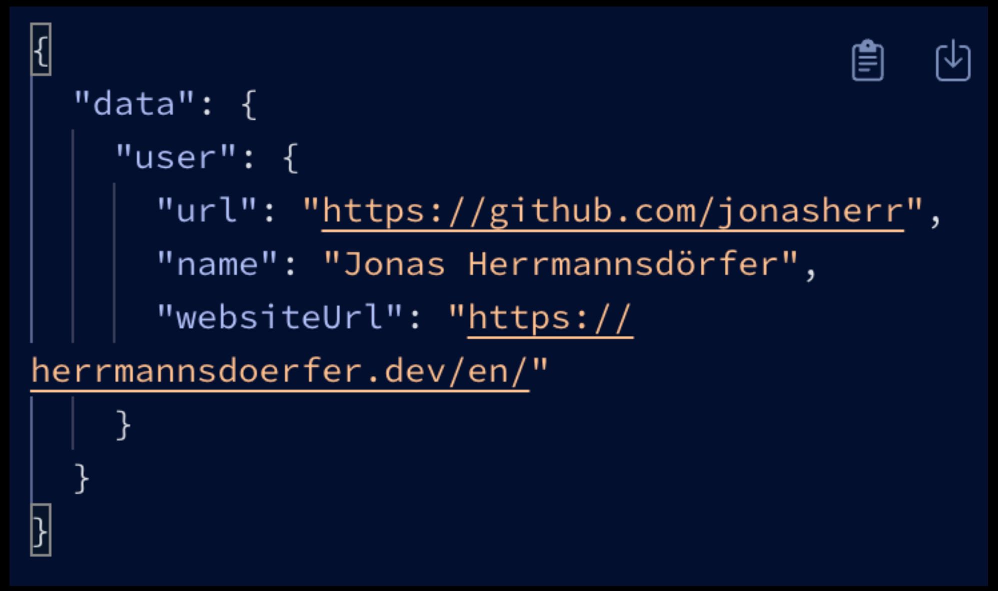
```json
{
  "data": {
    "user": {
      "url": "https://github.com/jonasherr",
      "name": "Jonas Herrmannsdörfer",
      "websiteUrl": "https://
herrmannsdoerfer.dev/en/"
    }
  }
}
```

Advantages of GraphQL

- Describe what's possible with a type system

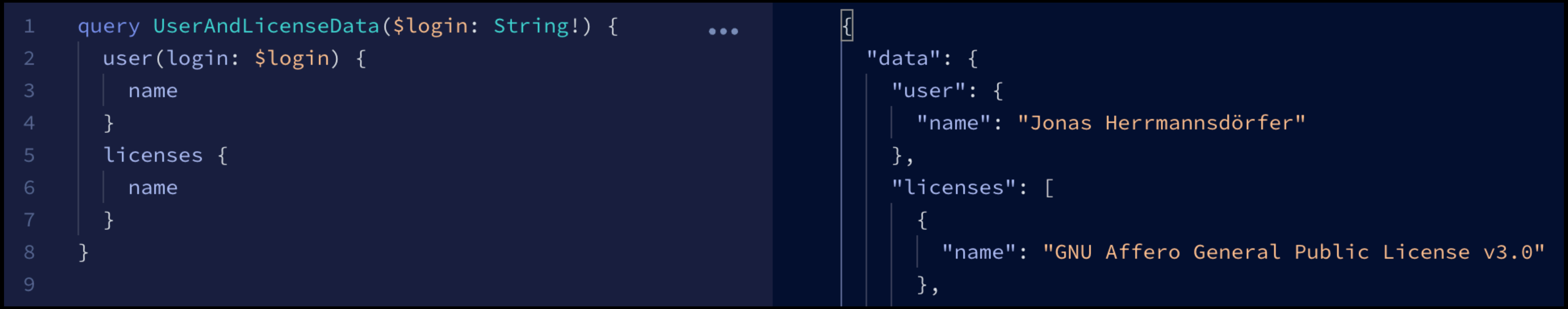


```graphql
type User {
    url: URI!
    name: String
    websiteUrl: URI
}
```



```json
{
  "data": {
    "user": {
      "url": "https://github.com/jonasherr",
      "name": "Jonas Herrmannsdörfer",
      "websiteUrl": "https://
herrmannsdoerfer.dev/en/"
    }
  }
}
```

Advantages of GraphQL



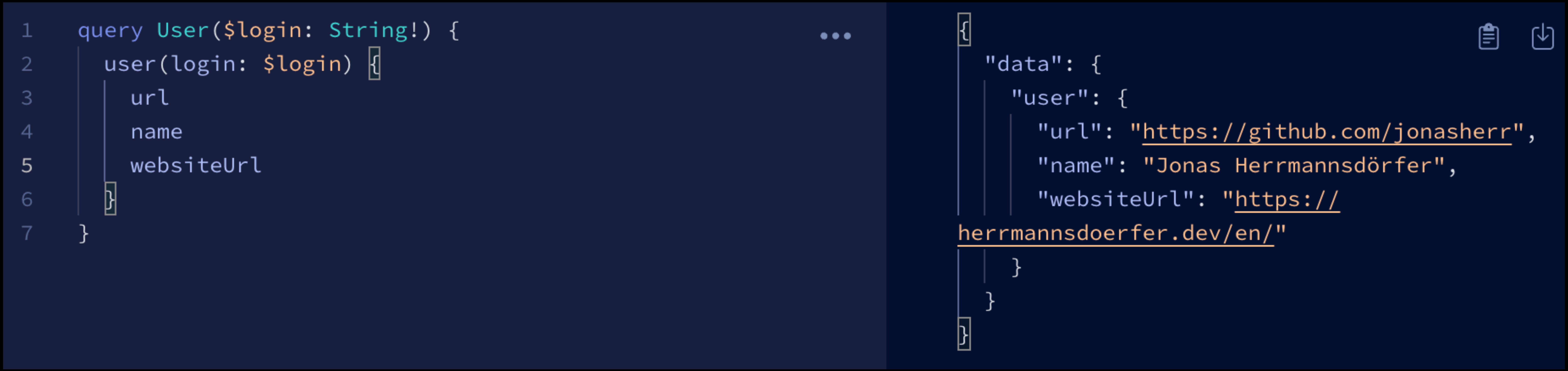
```
query UserAndLicenseData($login: String!) {
  user(login: $login) {
    name
  }
  licenses {
    name
  }
}
```

```
{
  "data": {
    "user": {
      "name": "Jonas Herrmannsdörfer"
    },
    "licenses": [
      {
        "name": "GNU Affero General Public License v3.0"
      },
```

- Get many resources in a single request

GraphQL Basics

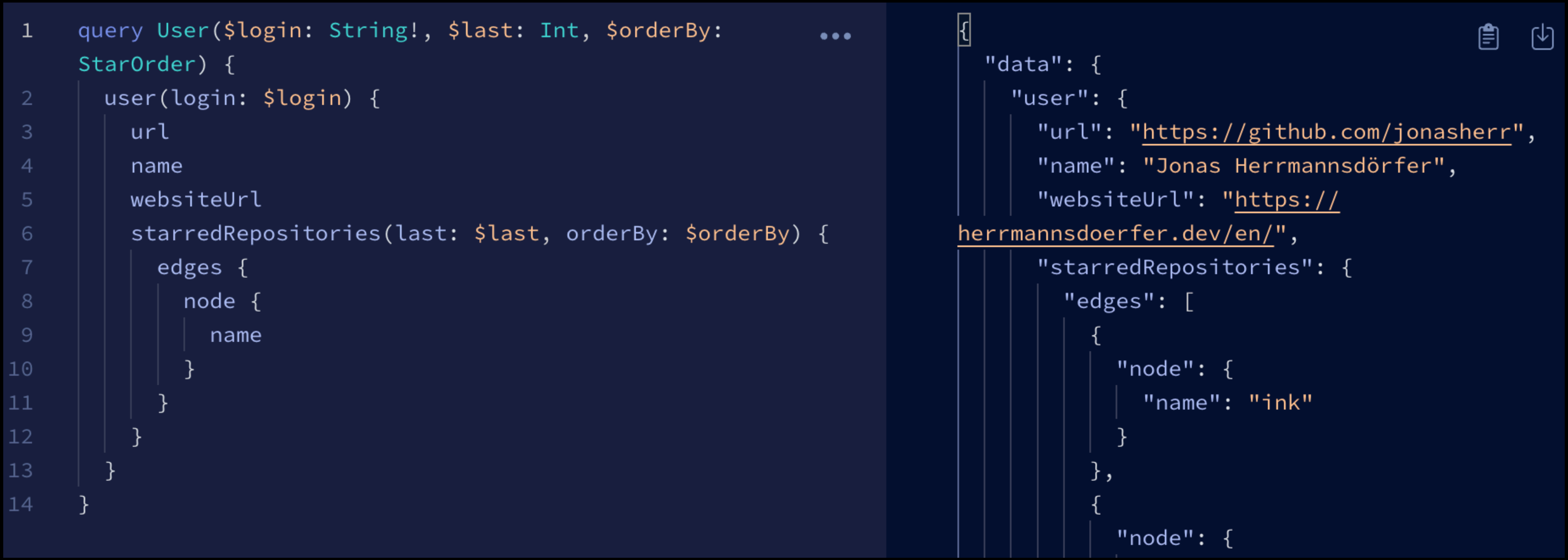
Queries - GraphQL Basics



```graphql
query User($login: String!) {
  user(login: $login) {
    url
    name
    websiteUrl
  }
}
```

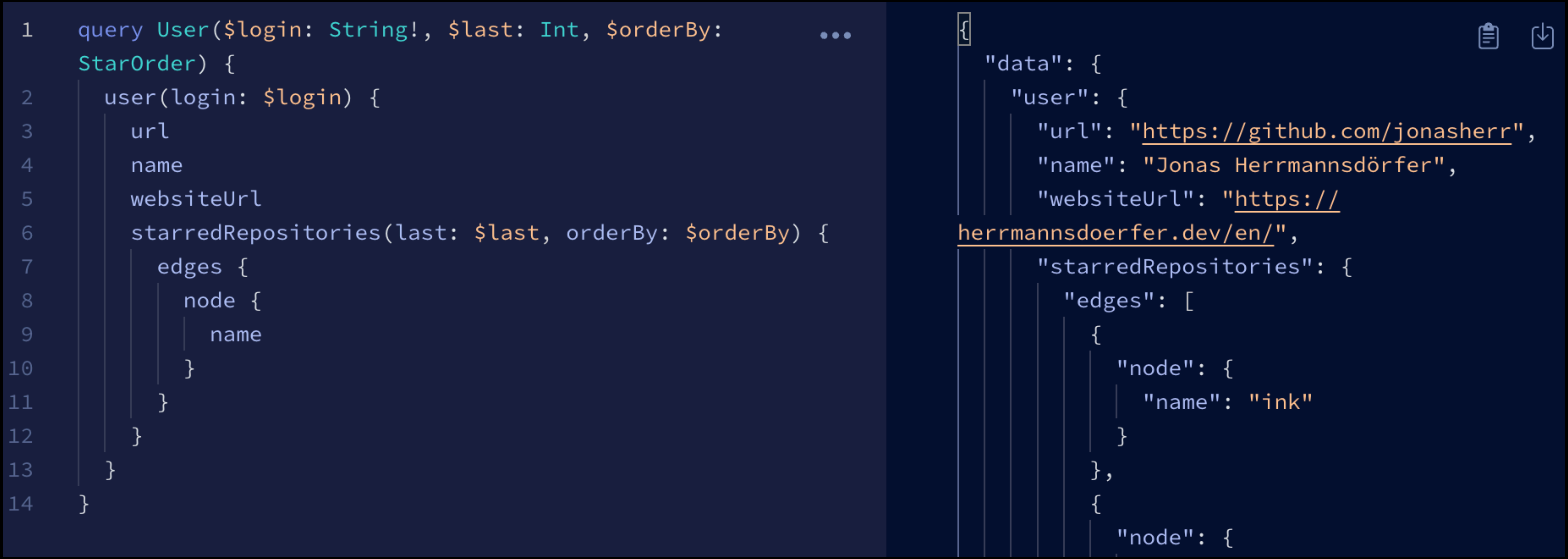
```json
{
  "data": {
    "user": {
      "url": "https://github.com/jonasherr",
      "name": "Jonas Herrmannsdörfer",
      "websiteUrl": "https://
herrmannsdoerfer.dev/en/"
    }
  }
}
```

Queries - GraphQL Basics



```graphql
query User($login: String!, $last: Int, $orderBy:
StarOrder) {
  user(login: $login) {
    url
    name
    websiteUrl
    starredRepositories(last: $last, orderBy: $orderBy) {
      edges {
        node {
          name
        }
      }
    }
  }
}
```

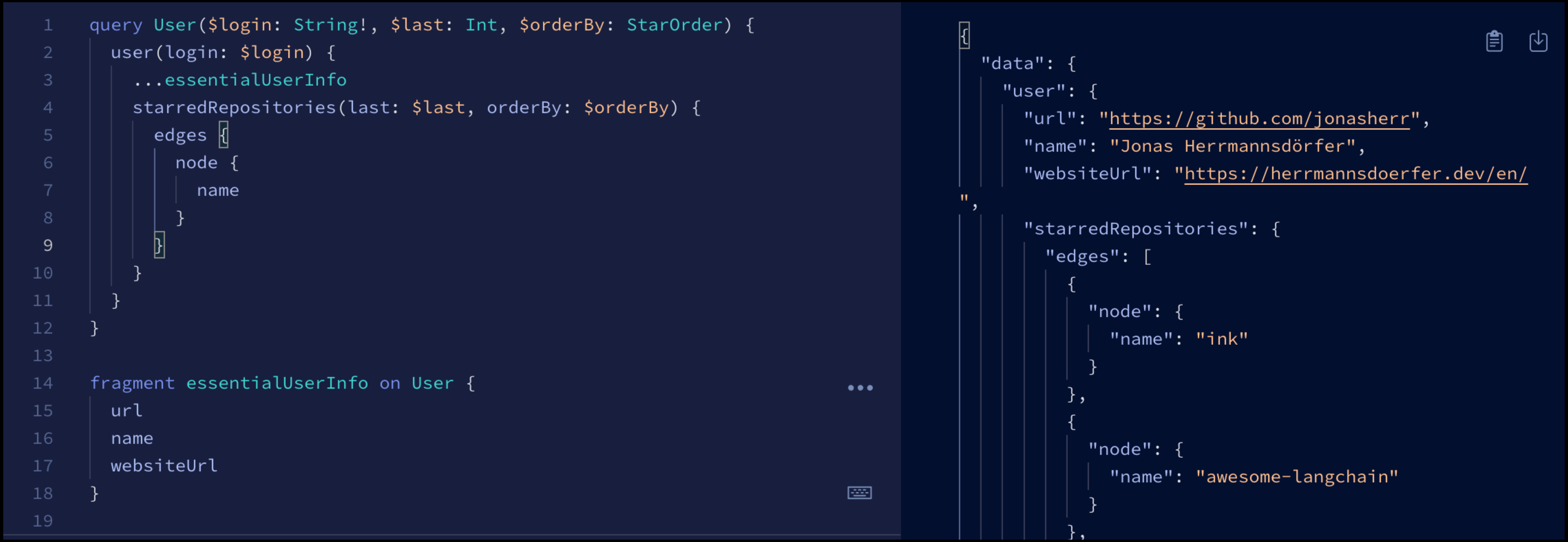
```json
{
  "data": {
    "user": {
      "url": "https://github.com/jonasherr",
      "name": "Jonas Herrmannsdörfer",
      "websiteUrl": "https://
herrmannsdoerfer.dev/en/",
      "starredRepositories": {
        "edges": [
          {
            "node": {
              "name": "ink"
            }
          },
          {
            "node": {
```



```graphql
query User($login: String!, $last: Int, $orderBy:
StarOrder) {
  user(login: $login) {
    url
    name
    websiteUrl
    starredRepositories(last: $last, orderBy: $orderBy) {
      edges {
        node {
          name
        }
      }
    }
  }
}
```

```json
{
  "data": {
    "user": {
      "url": "https://github.com/jonasherr",
      "name": "Jonas Herrmannsdörfer",
      "websiteUrl": "https://
herrmannsdoerfer.dev/en/",
      "starredRepositories": {
        "edges": [
          {
            "node": {
              "name": "ink"
            }
          },
          {
            "node": {
```

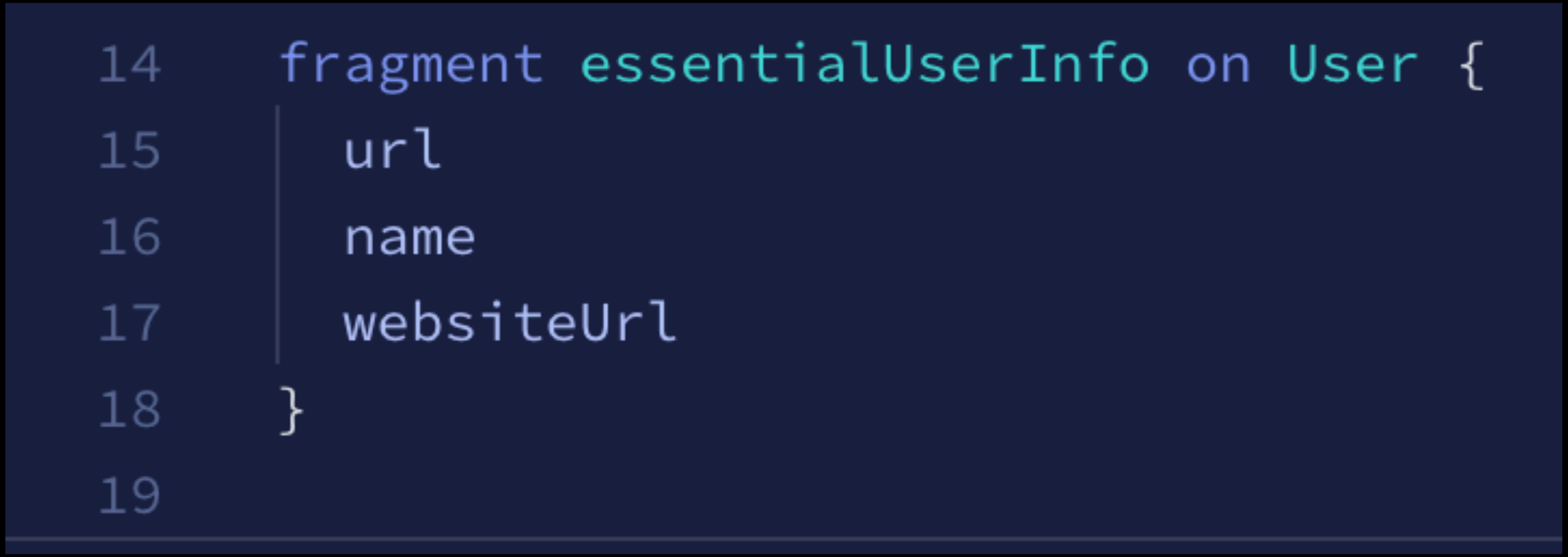
Fragments - GraphQL Basics



```graphql
query User($login: String!, $last: Int, $orderBy: StarOrder) {
  user(login: $login) {
    ...essentialUserInfo
    starredRepositories(last: $last, orderBy: $orderBy) {
      edges {
        node {
          name
        }
      }
    }
  }
}

fragment essentialUserInfo on User {
  url
  name
  websiteUrl
}
```

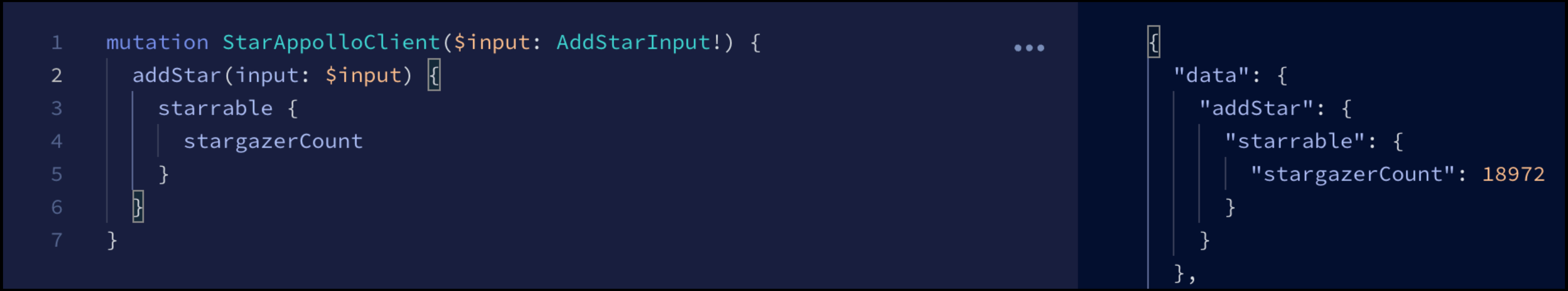
```json
{
  "data": {
    "user": {
      "url": "https://github.com/jonasherr",
      "name": "Jonas Herrmannsdörfer",
      "websiteUrl": "https://herrmannsdoerfer.dev/en/
",
      "starredRepositories": {
        "edges": [
          {
            "node": {
              "name": "ink"
            }
          },
          {
            "node": {
              "name": "awesome-langchain"
            }
          },
```



```graphql
fragment essentialUserInfo on User {
  url
  name
  websiteUrl
}

```

Mutations - GraphQL Basics



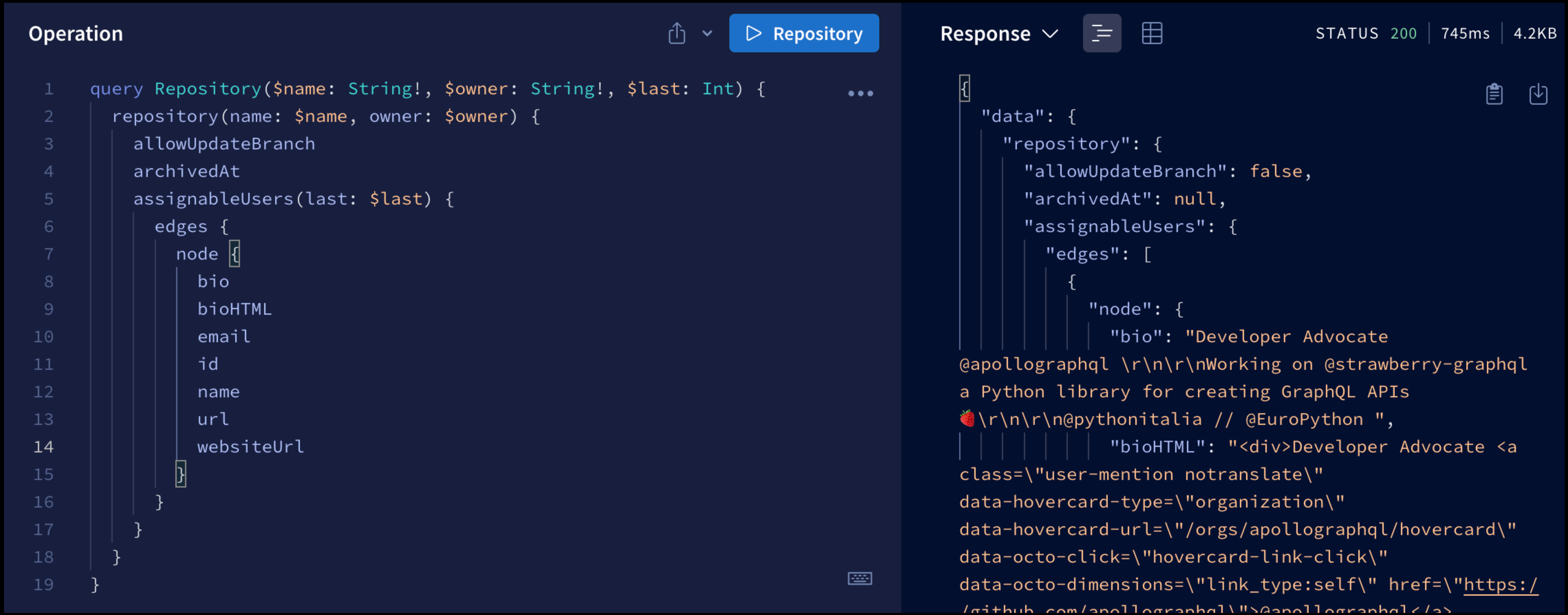
```graphql
mutation StarAppolloClient($input: AddStarInput!) {
  addStar(input: $input) {
    starrable {
      stargazerCount
    }
  }
}
```

```json
{
  "data": {
    "addStar": {
      "starrable": {
        "stargazerCount": 18972
      }
    }
  },
```

Antipatterns

Over-fetching

- Asking for too much data
- Makes your app slower than it has to be


Operation

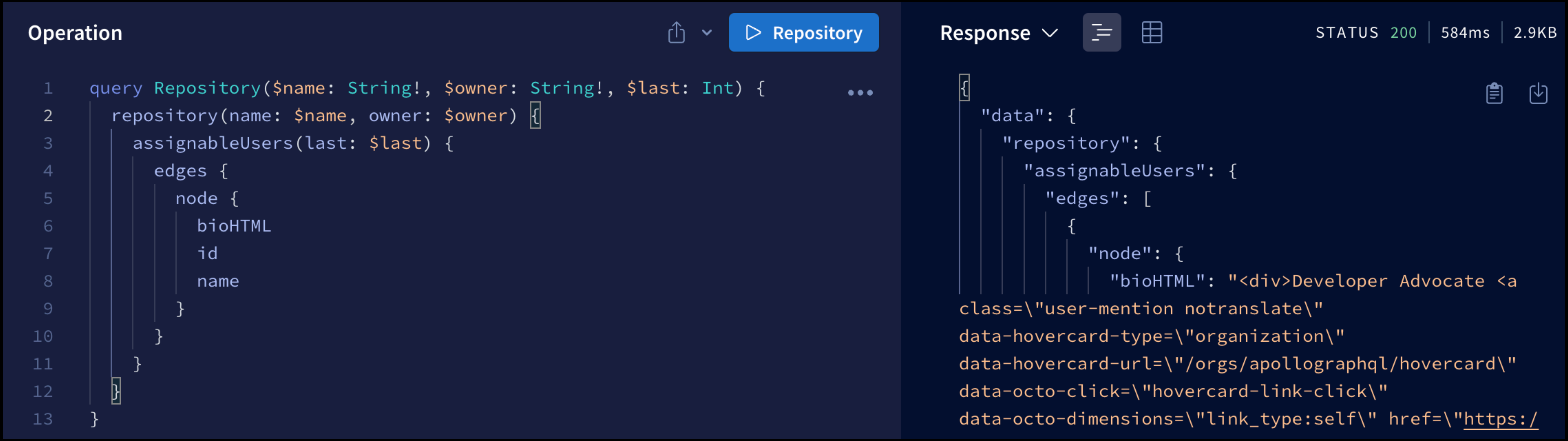
Repository

```graphql
query Repository($name: String!, $owner: String!, $last: Int) {
  repository(name: $name, owner: $owner) {
    allowUpdateBranch
    archivedAt
    assignableUsers(last: $last) {
      edges {
        node {
          bio
          bioHTML
          email
          id
          name
          url
          websiteUrl
        }
      }
    }
  }
}
```

Response

STATUS 200 | 745ms | 4.2KB

```json
{
  "data": {
    "repository": {
      "allowUpdateBranch": false,
      "archivedAt": null,
      "assignableUsers": {
        "edges": [
          {
            "node": {
              "bio": "Developer Advocate
@apollographql \r\n\r\nWorking on @strawberry-graphql
a Python library for creating GraphQL APIs
🍓\r\n\r\n@pythonitalia // @EuroPython ",
              "bioHTML": "<div>Developer Advocate <a
class=\"user-mention notranslate\"
data-hovercard-type=\"organization\"
data-hovercard-url=\"/orgs/apollographql/hovercard\"
data-octo-click=\"hovercard-link-click\"
data-octo-dimensions=\"link_type:self\" href=\"https:/
/github.com/apollographql\">@apollographql</a>
```



```
query Repository($name: String!, $owner: String!, $last: Int) {
  repository(name: $name, owner: $owner) {
    assignableUsers(last: $last) {
      edges {
        node {
          bioHTML
          id
          name
        }
      }
    }
  }
}
```

```
{
  "data": {
    "repository": {
      "assignableUsers": {
        "edges": [
          {
            "node": {
              "bioHTML": "<div>Developer Advocate <a
class=\"user-mention notranslate\"
data-hovercard-type=\"organization\"
data-hovercard-url=\"/orgs/apollographql/hovercard\"
data-octo-click=\"hovercard-link-click\"
data-octo-dimensions=\"link_type:self\" href=\"https:/
```

Size: -31 %

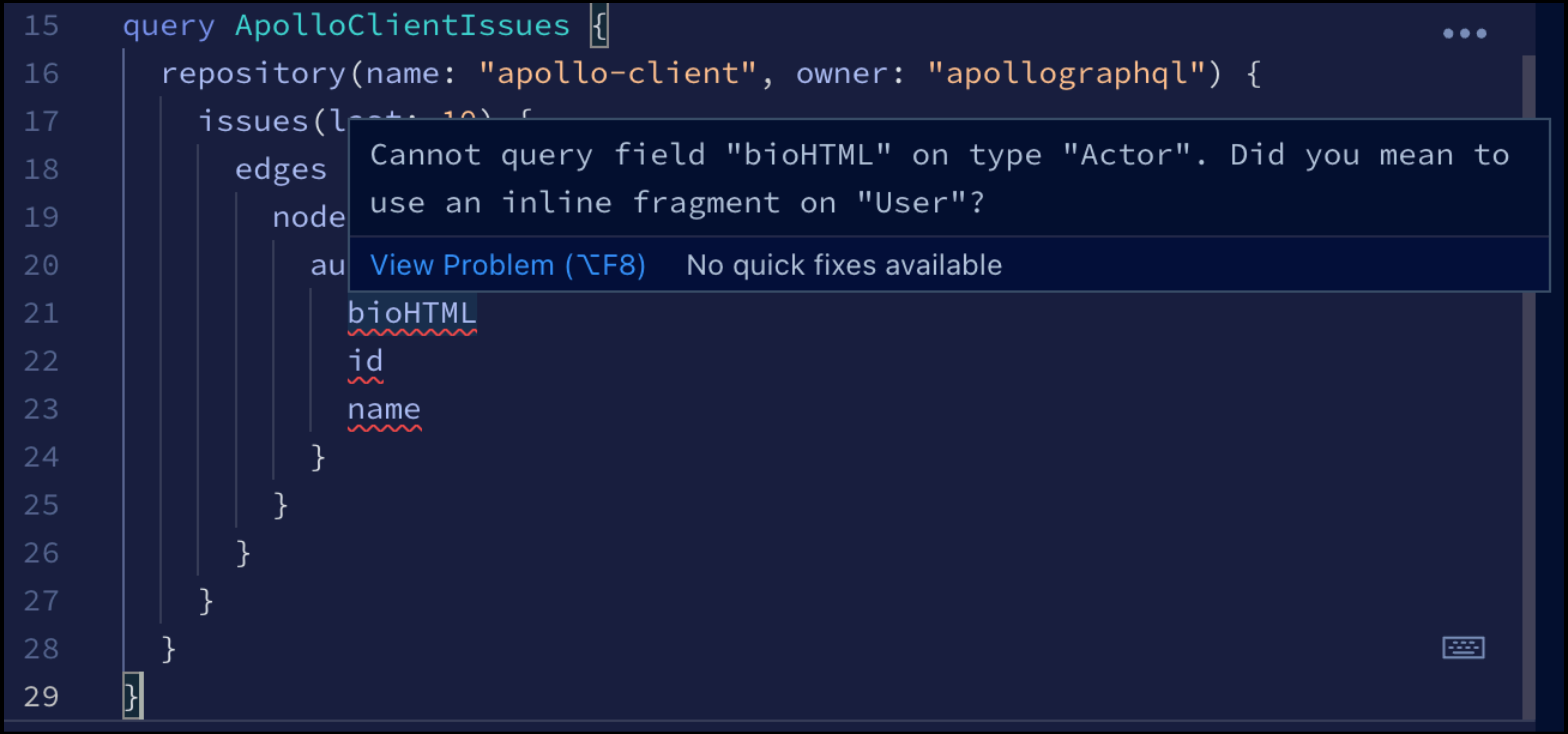
Time: -22 %

Solution - Over-fetching

- Ask for only the data you need
- Update queries if you do not use a field in your app anymore

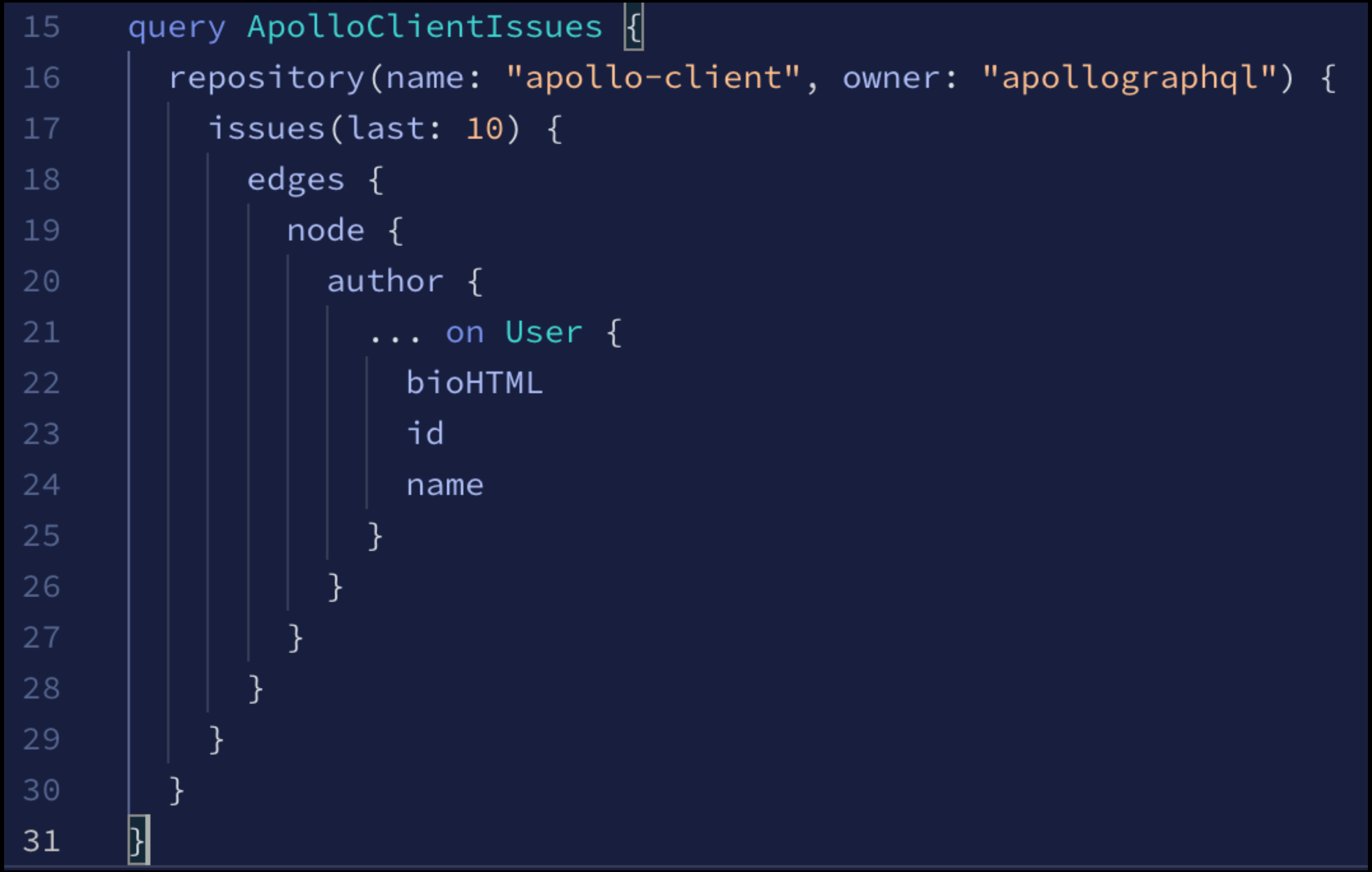
Why?

Misuse of fragments

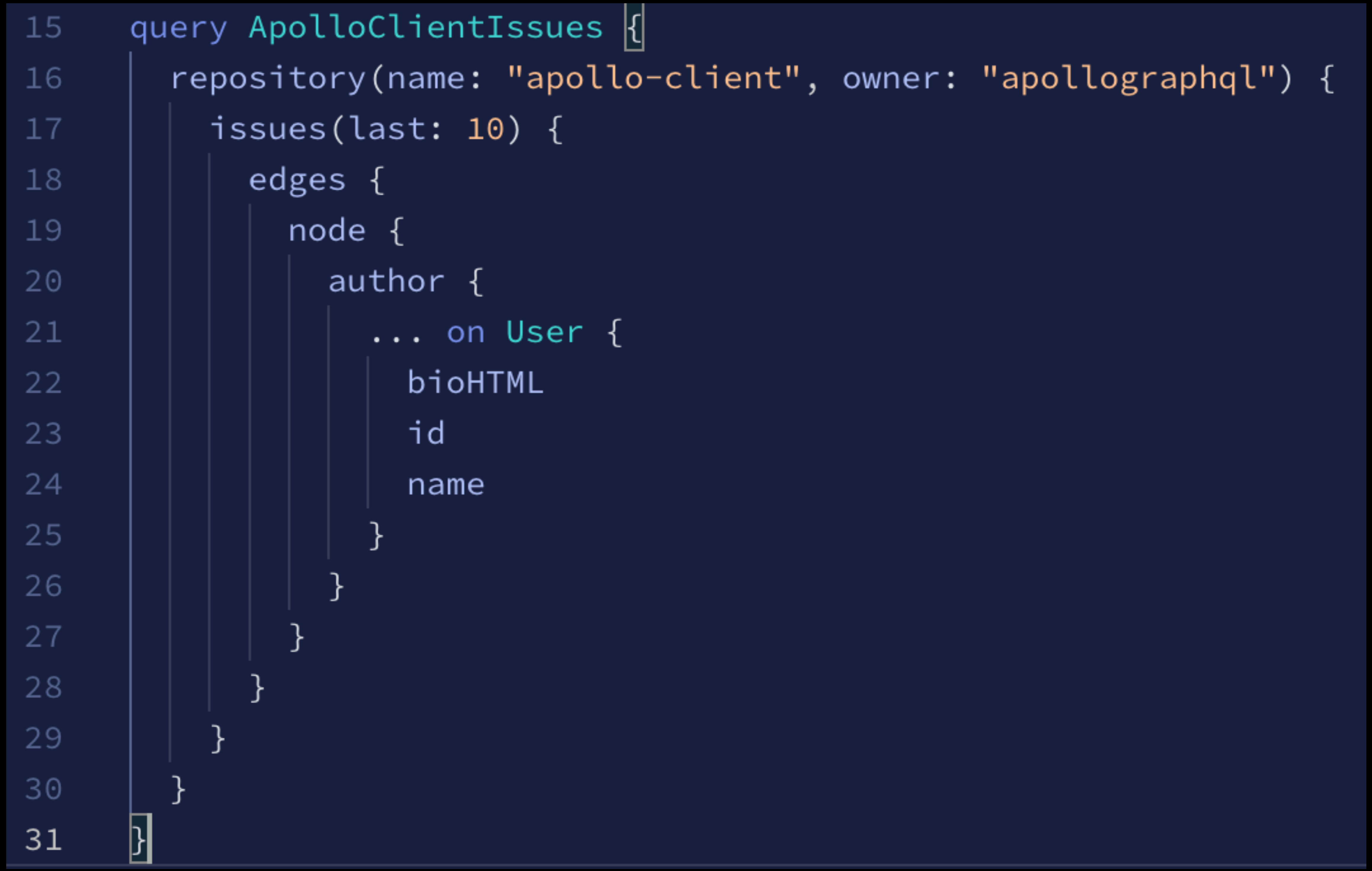


```
query ApolloClientIssues {
  repository(name: "apollo-client", owner: "apollographql") {
    issues(l
      edges
        node
          au
            bioHTML
            id
            name
          }
        }
      }
    }
  }
}
```

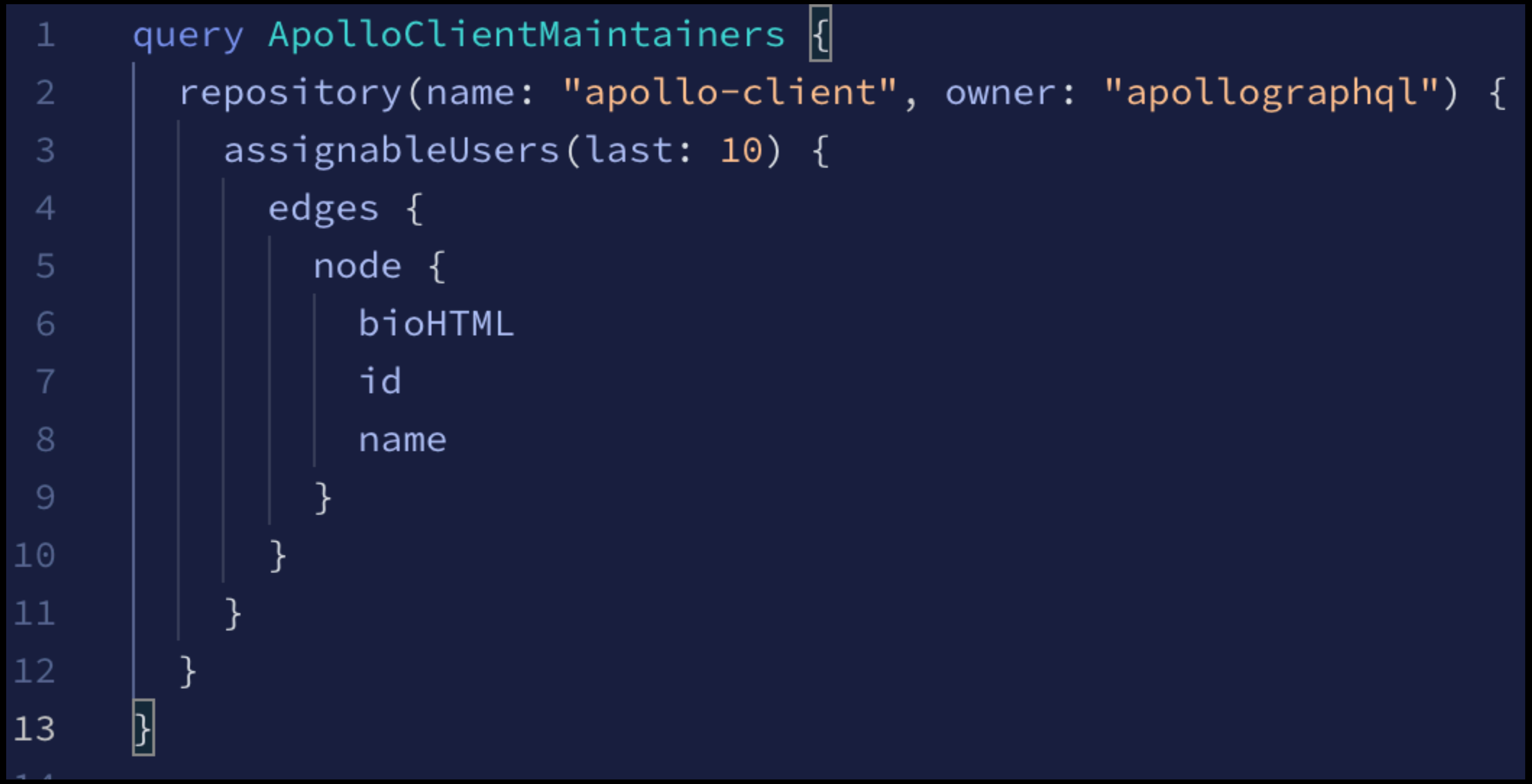
Cannot query field "bioHTML" on type "Actor". Did you mean to use an inline fragment on "User"?



```graphql
query ApolloClientIssues {
  repository(name: "apollo-client", owner: "apollographql") {
    issues(last: 10) {
      edges {
        node {
          author {
            ... on User {
              bioHTML
              id
              name
            }
          }
        }
      }
    }
  }
}
```



```graphql
query ApolloClientIssues {
  repository(name: "apollo-client", owner: "apollographql") {
    issues(last: 10) {
      edges {
        node {
          author {
            ... on User {
              bioHTML
              id
              name
            }
          }
        }
      }
    }
  }
}
```

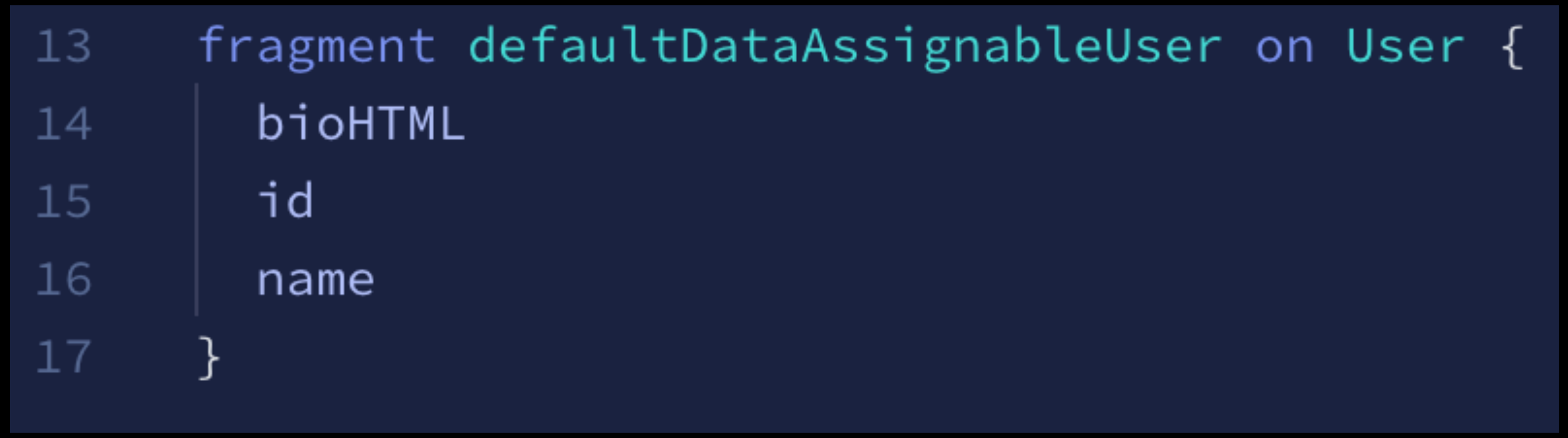


```graphql
query ApolloClientMaintainers {
  repository(name: "apollo-client", owner: "apollographql") {
    assignableUsers(last: 10) {
      edges {
        node {
          bioHTML
          id
          name
        }
      }
    }
  }
}
```

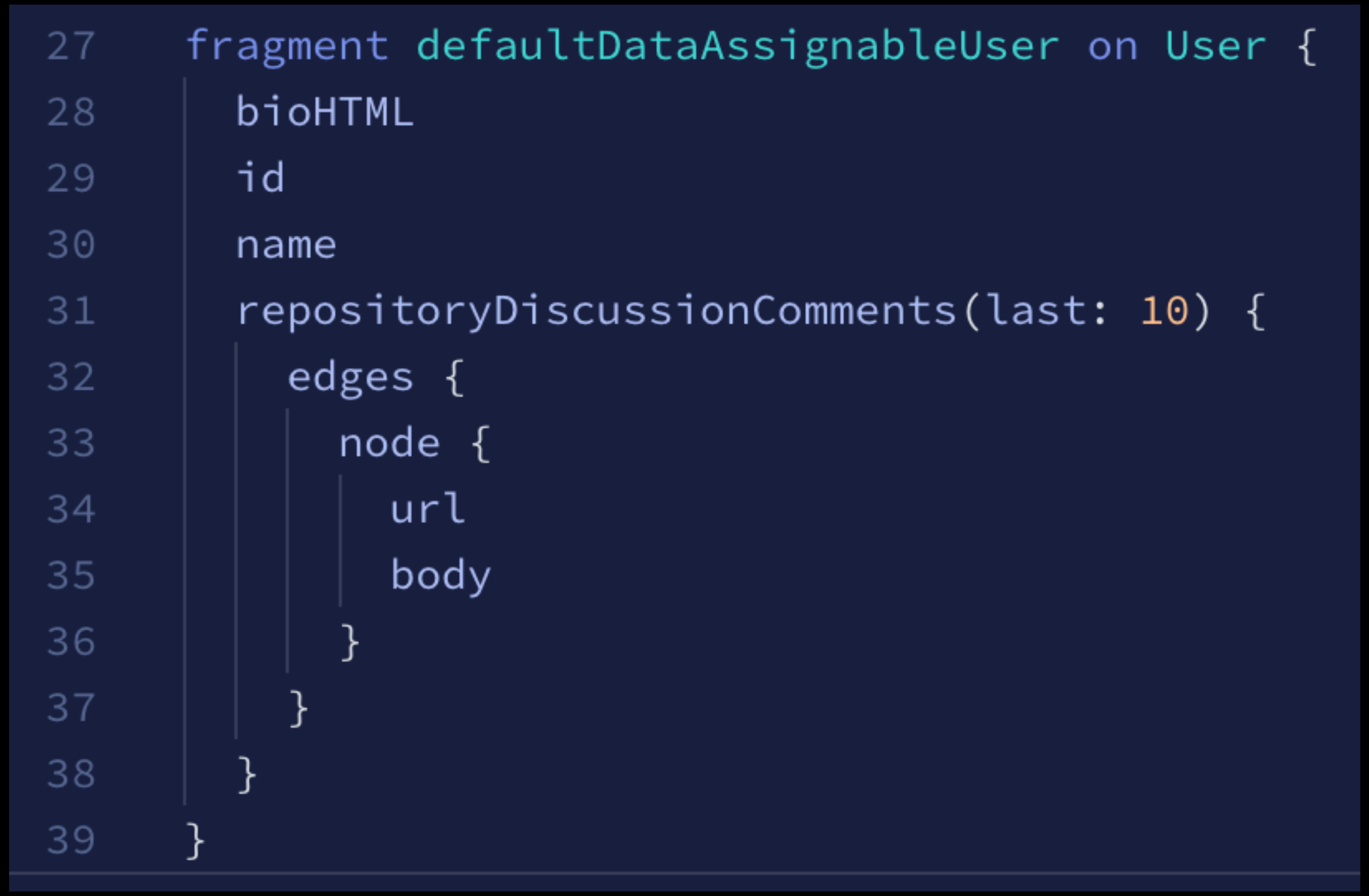
```graphql
query ApolloClientMaintainers {
  repository(name: "apollo-client", owner: "apollographql") {
    assignableUsers(last: 10) {
      edges {
        node {
          ...defaultDataAssignableUser
        }
      }
    }
  }
}
```

```graphql
query ApolloClientIssues {
  repository(name: "apollo-client", owner: "apollographql") {
    issues(last: 10 ) {
      edges {
        node {
          author  {
            ...defaultDataAssignableUser
          }
        }
      }
    }
  }
}
```

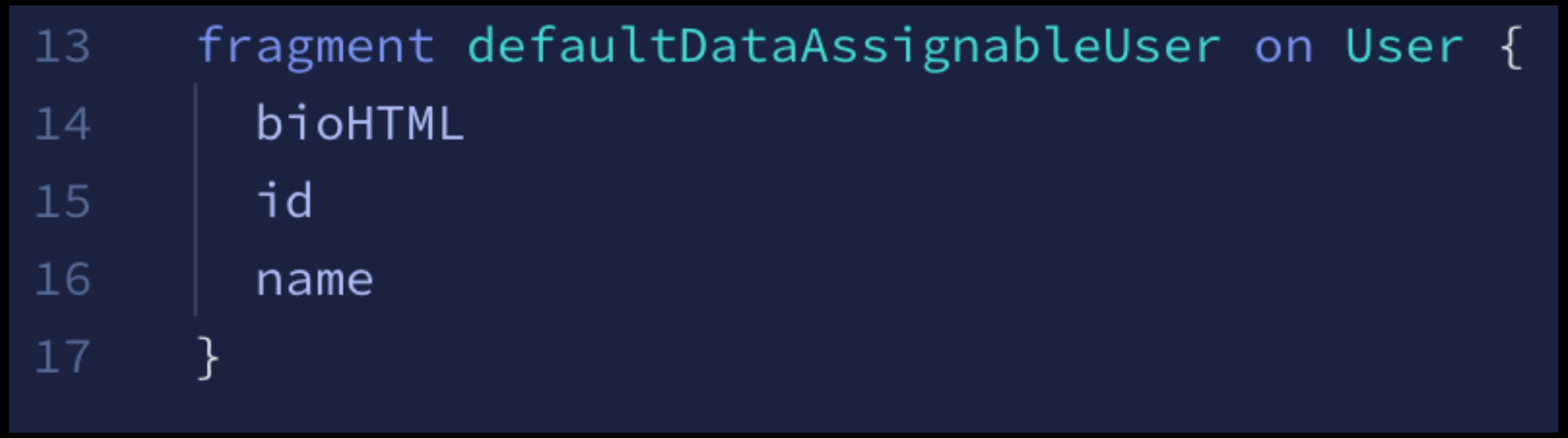
```graphql
fragment defaultDataAssignableUser on User {
  bioHTML
  id
  name
}
```



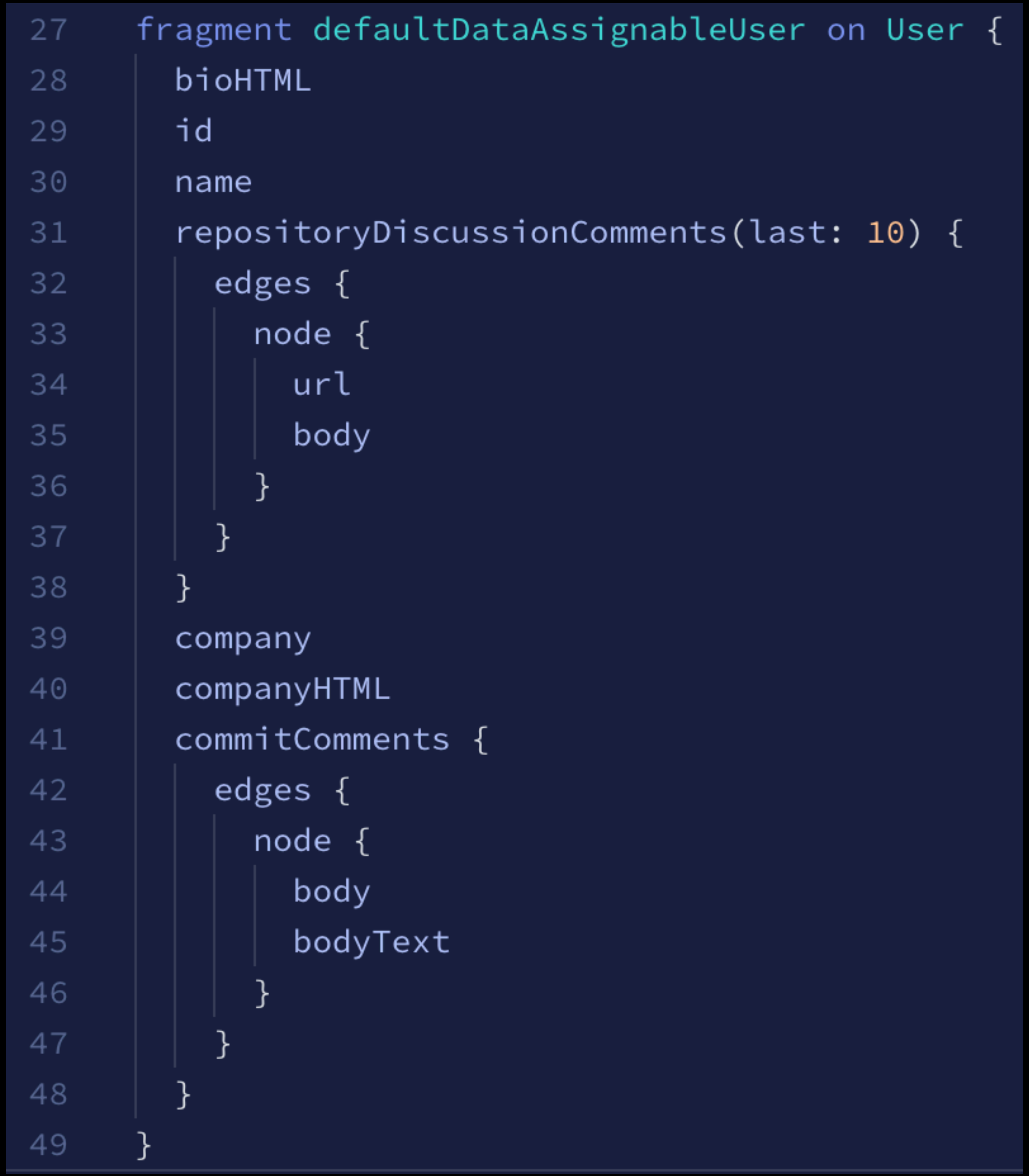
```graphql
fragment defaultDataAssignableUser on User {
  bioHTML
  id
  name
}
```



```graphql
fragment defaultDataAssignableUser on User {
  bioHTML
  id
  name
  repositoryDiscussionComments(last: 10) {
    edges {
      node {
        url
        body
      }
    }
  }
}
```



```graphql
fragment defaultDataAssignableUser on User {
  bioHTML
  id
  name
}
```



```graphql
fragment defaultDataAssignableUser on User {
  bioHTML
  id
  name
  repositoryDiscussionComments(last: 10) {
    edges {
      node {
        url
        body
      }
    }
  }
  company
  companyHTML
  commitComments {
    edges {
      node {
        body
        bodyText
      }
    }
  }
}
```

Solution - Misuse of fragments

- Be careful with fragments
- Do not create a bloated fragments.graphql file
- Store fragments in same file as the queries that are using it
- When updating a fragment: check if other queries need this information

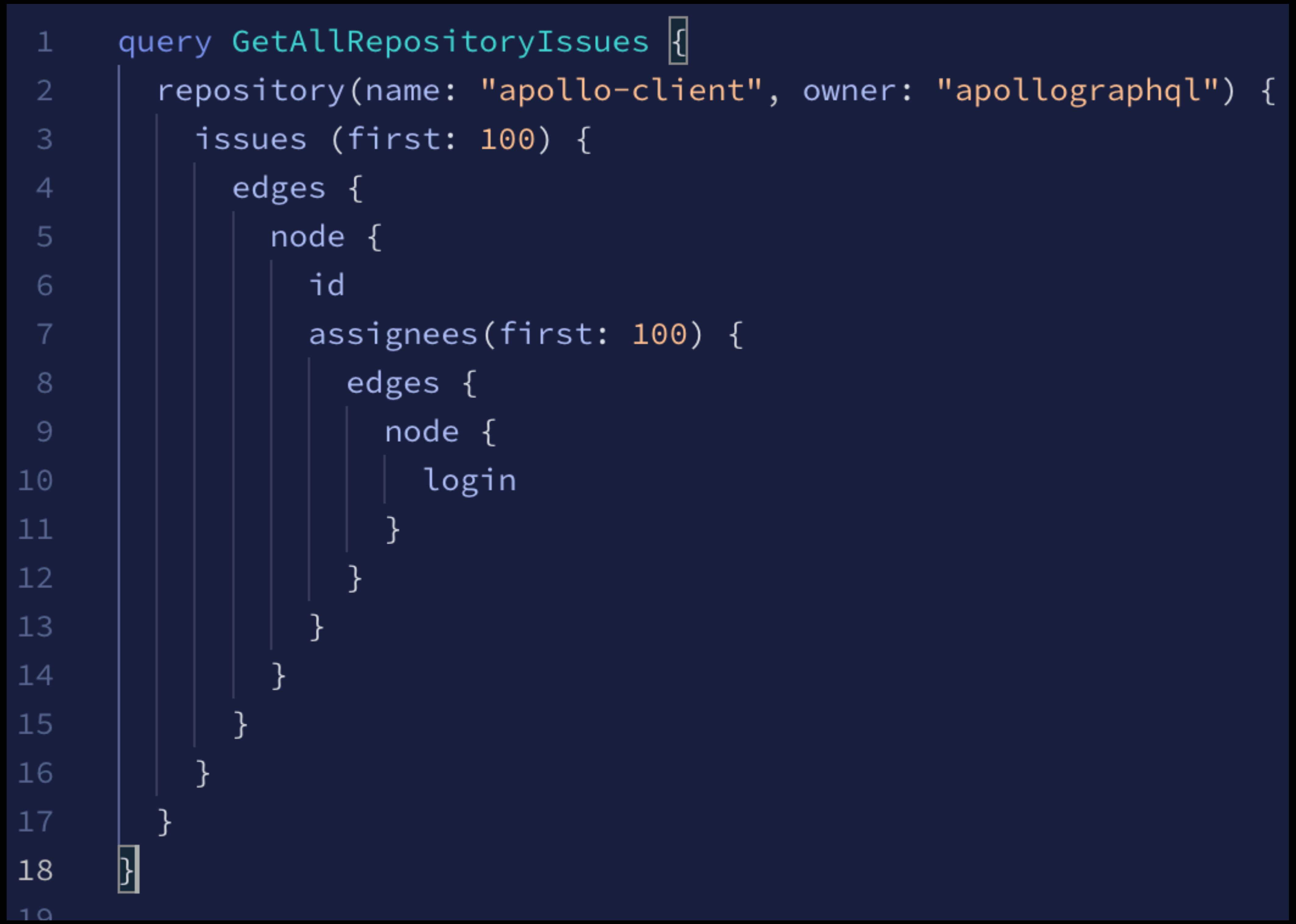
Under-fetching

- not asking for enough data in a query, forcing you to make a second query

Disadvantages multiple queries

- overhead for each request
- response compression will work better for the single request case.
- Several loading spinners
- Filtering, sorting and pagination cannot be easily handled server side

Goal: Get all issues from the Apollo client repository which are already assigned to someone.

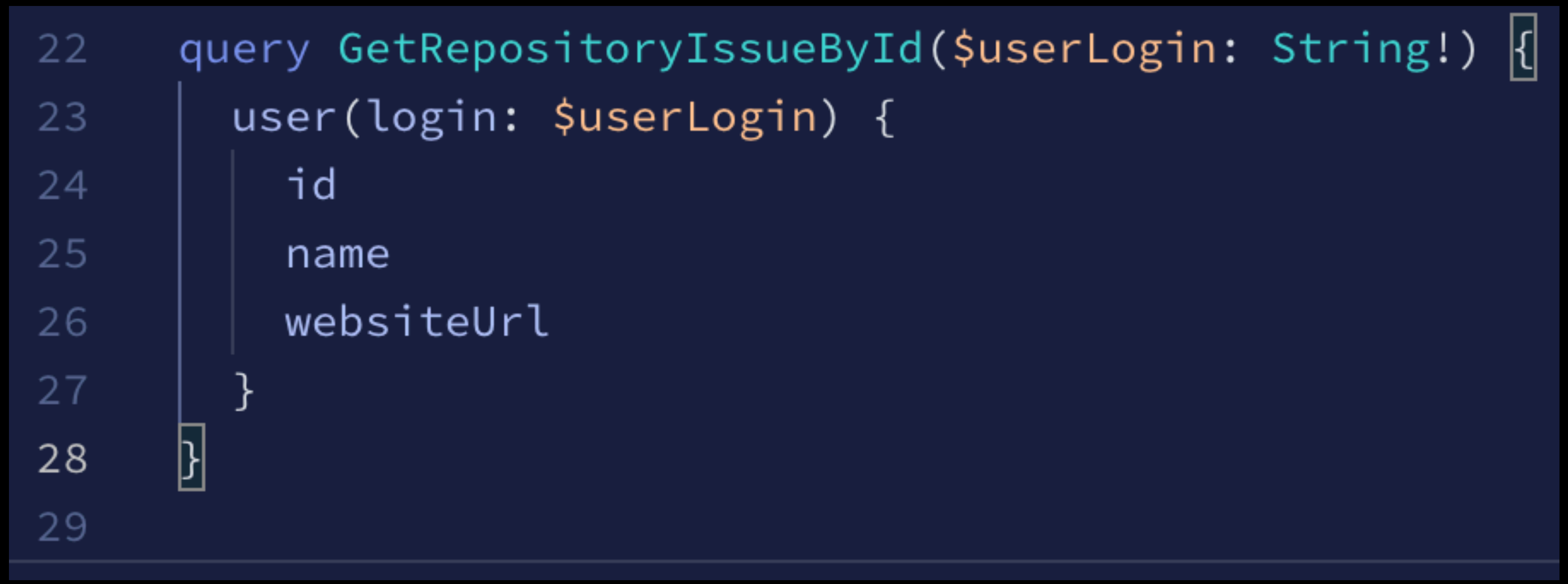


```graphql
query GetAllRepositoryIssues {
  repository(name: "apollo-client", owner: "apollographql") {
    issues (first: 100) {
      edges {
        node {
          id
          assignees(first: 100) {
            edges {
              node {
                login
              }
            }
          }
        }
      }
    }
  }
}
```

Filter on the client

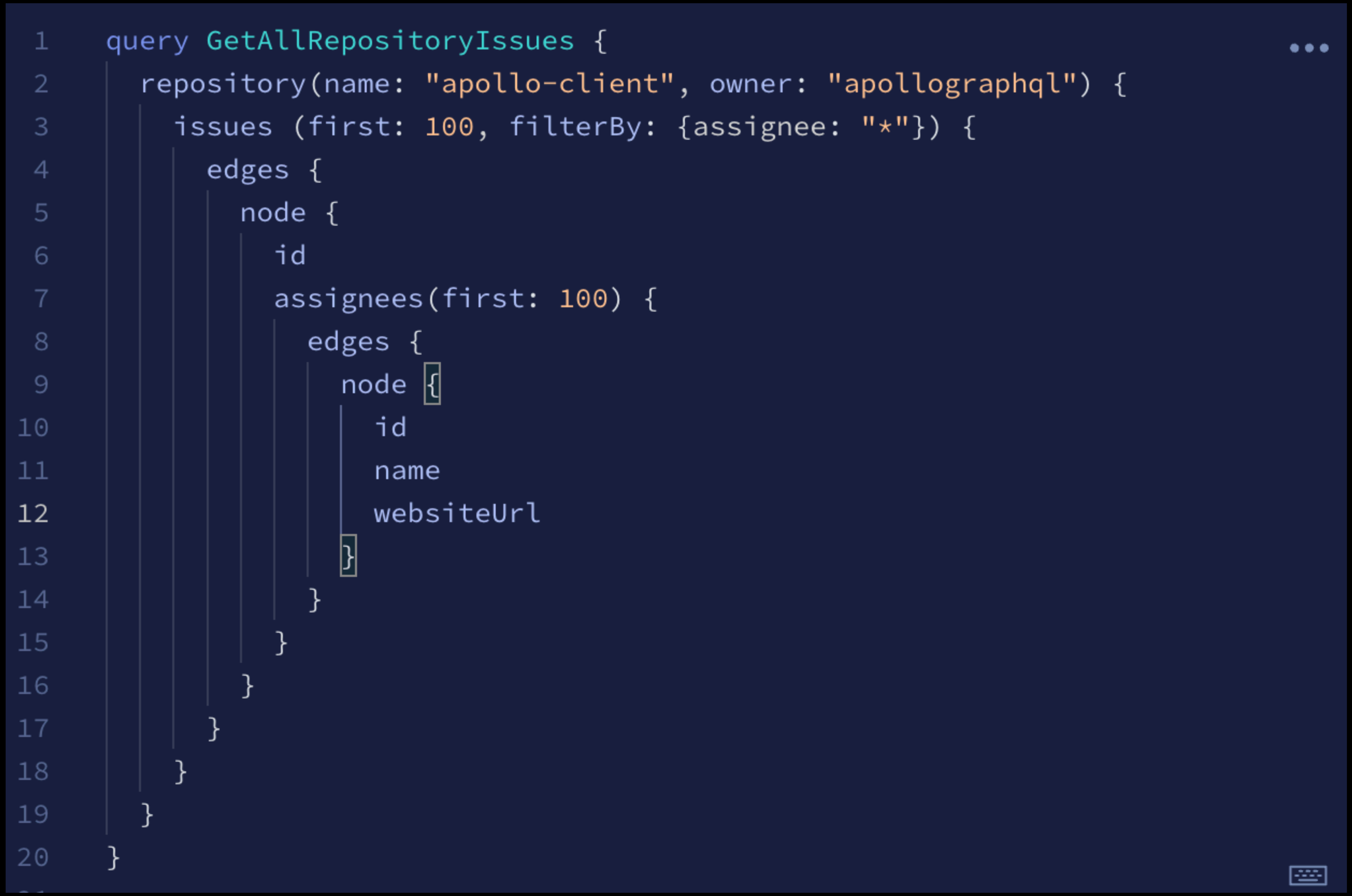
Map over logins

Use each login in a component to query the data



```graphql
query GetRepositoryIssueById($userLogin: String!) {
  user(login: $userLogin) {
    id
    name
    websiteUrl
  }
}
```

Render the user



```graphql
query GetAllRepositoryIssues {
  repository(name: "apollo-client", owner: "apollographql") {
    issues (first: 100, filterBy: {assignee: "*"}) {
      edges {
        node {
          id
          assignees(first: 100) {
            edges {
              node {
                id
                name
                websiteUrl
              }
            }
          }
        }
      }
    }
  }
}
```

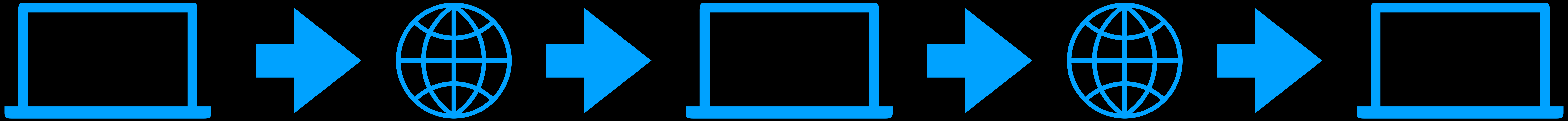
Map over assignees

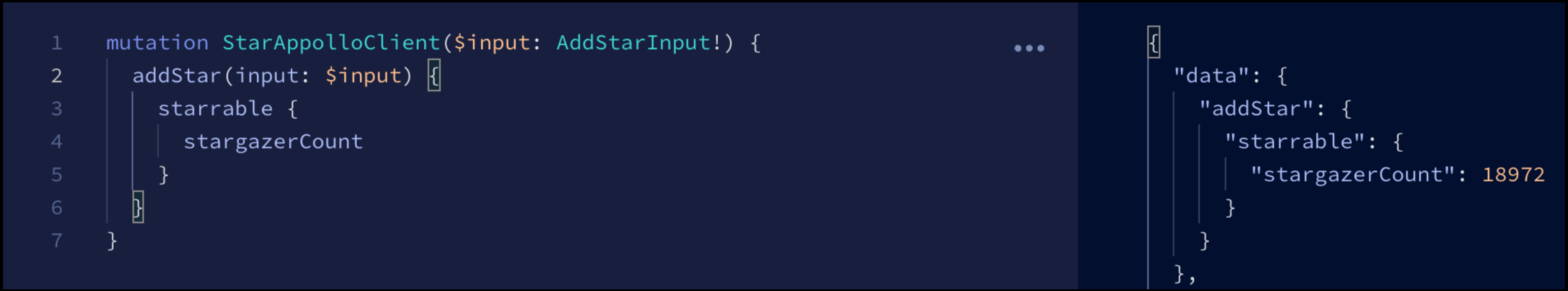
Render the user

Solution - Under-fetching

- User filters, sorting and pagination on the server

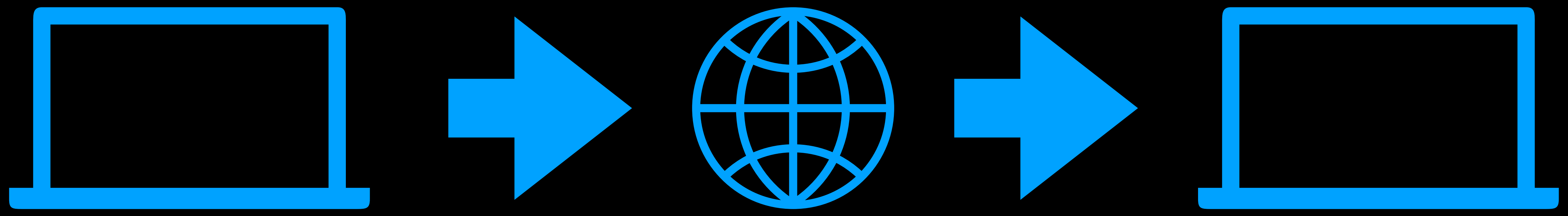
Not using return values of mutation

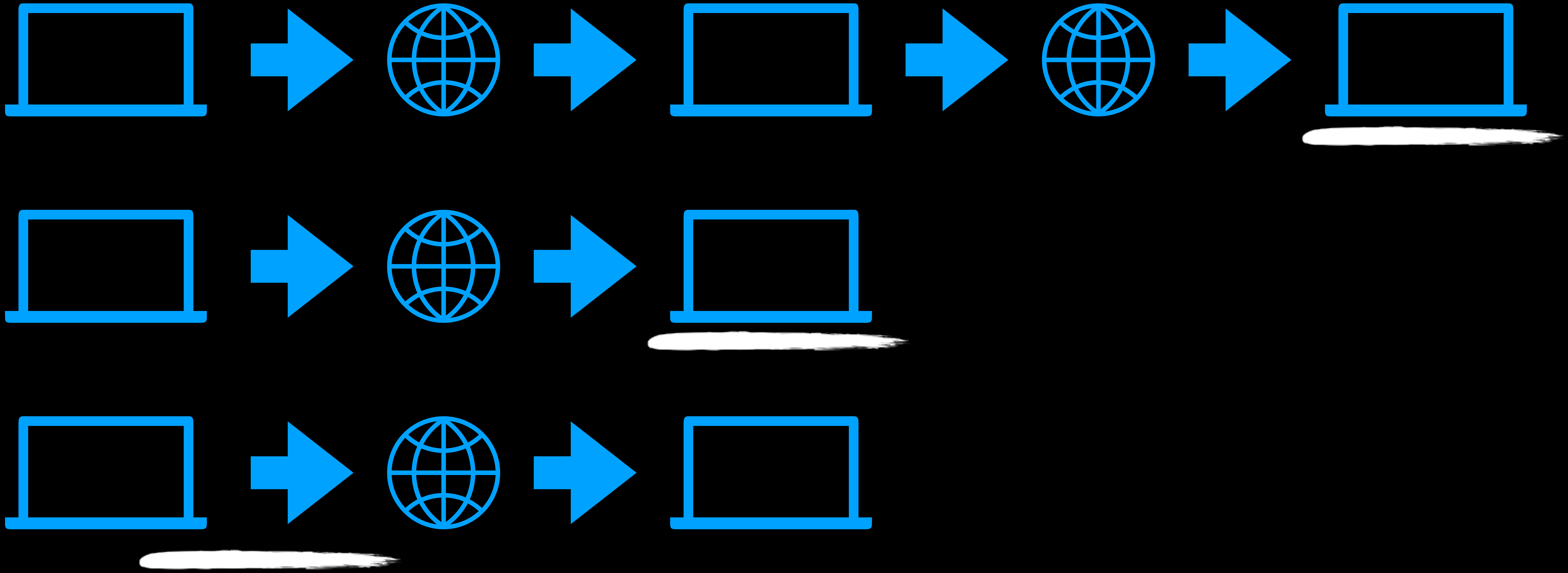




```graphql
mutation StarAppolloClient($input: AddStarInput!) {
  addStar(input: $input) {
    starrable {
      stargazerCount
    }
  }
}
```

```json
{
  "data": {
    "addStar": {
      "starrable": {
        "stargazerCount": 18972
      }
    }
  },
```





Solution - Not using return values of mutation

- Use return values of mutation
- Better: use optimistic updates pattern

Not suitable caching library

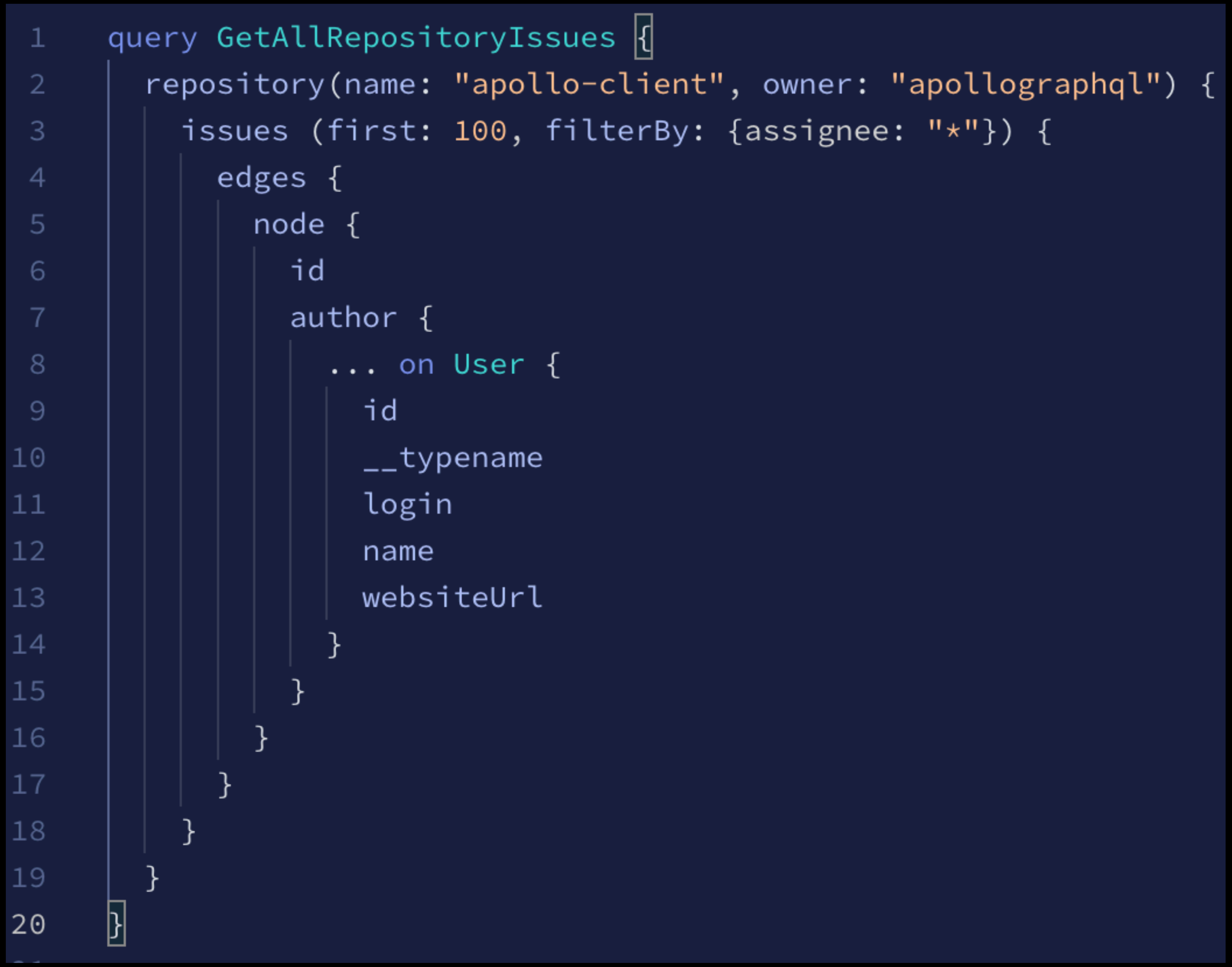
„Keep in mind that React Query **does not support normalized caching**.“

– React Query

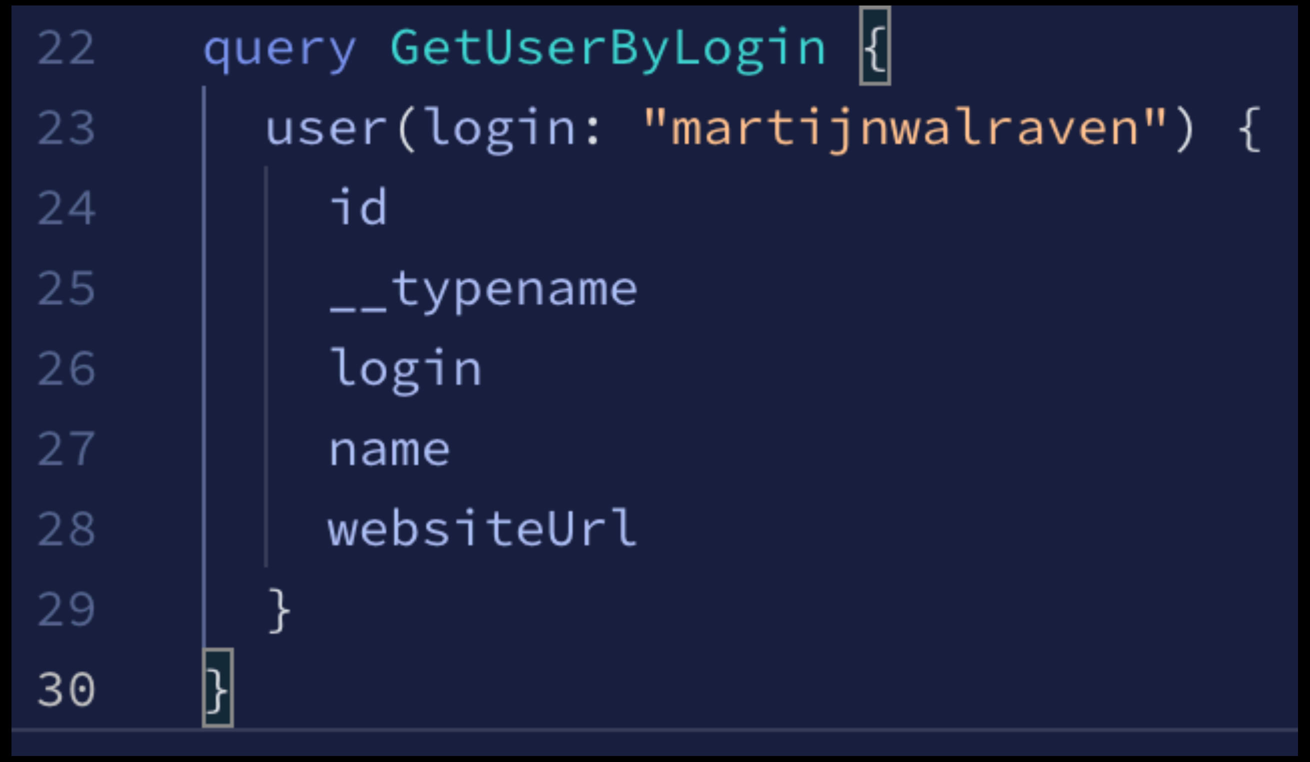
Normalized Cache

- Normalized Cache
- Data Storage
- Normalization Process
- Data Consistency
- Efficiency

Normalized Cache



```graphql
query GetAllRepositoryIssues {
  repository(name: "apollo-client", owner: "apollographql") {
    issues (first: 100, filterBy: {assignee: "*"}) {
      edges {
        node {
          id
          author {
            ... on User {
              id
              __typename
              login
              name
              websiteUrl
            }
          }
        }
      }
    }
  }
}
```



```graphql
query GetUserByLogin {
  user(login: "martijnwalraven") {
    id
    __typename
    login
    name
    websiteUrl
  }
}
```

Solution - Caching library

- Use a caching library that supports a normalized cache (if you benefit from it)
  - Apollo Client
  - URQL
  - ~~React Query~~
- Provide typename and id in query to make normalized cache work