



Advanced Access Control in GraphQL APIs

Group 4: Wilke Klausing, Huy Viet Nguyen





Structure

- Access Control
- Access Control Models
- GraphQL
 - GraphQL vs REST
- Our Task
- Validation Use-Case





Access Control

- Control and monitor access of resources
 - Allows or permits Read/Write/Delete operations
- Subjects, Objects, and Access Rights
- Implemented through different models



Technische Universität Berlin

Access Control Models

- Mandatory Access Control
 - Administration has authority
- Discretionary Access Control
 - Creator has authority
- Rule-Based Access Control
 - Grants access according to IF ELSE statements
- Purpose-Based Access Control
 - Defines access to purposes

HIGHLY SENSITIVE

SENSITIVE

INTERNAL

PUBLIC





GraphQL

- Query language and server-side runtime
- Queries similar to JSON
- Resolver for each requested field

```
{
  hero {
    name
  }
}

// mame is in the content of the c
```





GraphQL vs REST

- REST uses several endpoints
- Overfetching
- Underfetching





Our task

Create a re-usable component for at least one widely used framework/stack for developing GraphQL Web APIs allowing developers to easily add access control to their APIs

Minimum requirement: access-control on a field/parameter basis and at least one advanced access control scheme





Access Control Component

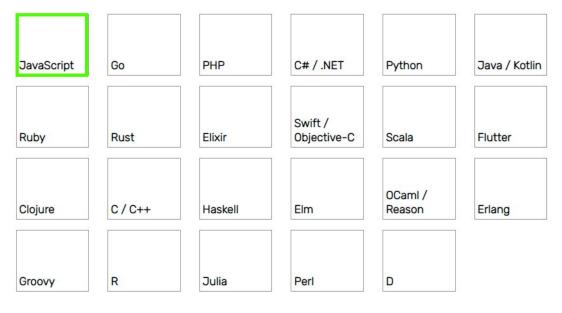
- Functionality:
 - Rule-Based Access Control (basic)
 - Purpose-Based Access Control (advanced)
 - more if we have enough time available





Our Thought Process

1. What languages does GraphQL support?







Our Thought Process

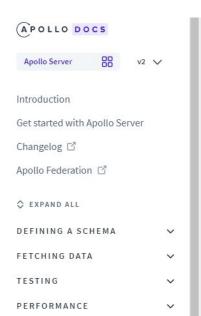
- 2. What are the most used JavaScript Frameworks?
 - GraphQL.js
 - Apollo Server
 - Express GraphQL





Our Thought Process

Game Changer:



Search Apollo Docs Launch Apollo Studio → Plugins Extend Apollo Server with custom functionality Plugins Plugins are available in Apollo Server 2.2.x and later. Creating a plugin Responding to events Plugins enable you to extend Apollo Server's core functionality by performing custom Request lifecycle event flow operations in response to certain events. Currently, these events correspond to individual phases of the GraphQL request lifecycle, and to the startup of Apollo Server End hooks itself. Inspecting request and response details For example, a basic logging plugin might log the GraphOL query string associated





Access Control Plugin

• System Integration:

```
r const server = new ApolloServer({
   typeDefs,
   resolvers,
   plugins: [
     require('./ourAccessControlPlugin'),
```





Access Control Plugin

- Configuration:
 - No database, only two configuration files:
 - 1. Purpose tree as configuration file
 - 2. Rules as configuration file
- Goal:
 - Flexible
 - Reuseable
 - Developer Friendly





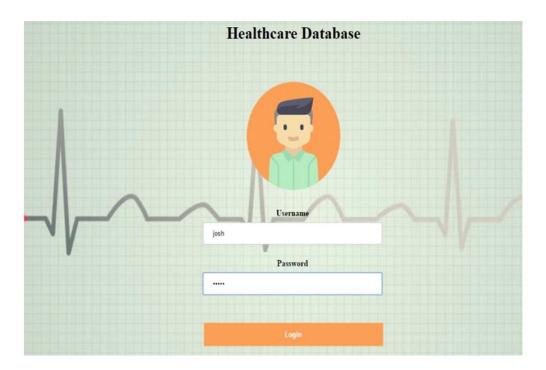
Validation Use-Case







Validation Use-Case







Validation Use-Case:

- Reasons to share health data:
 - Avoid medication errors
 - Avoid duplicate testing
 - Genetic studies
 - Chronic disease registries
 - Substance abuse
 - Population health management
 - Larger scale analytics
 - Epidemiology/disease tracking





Thank you for your attention.