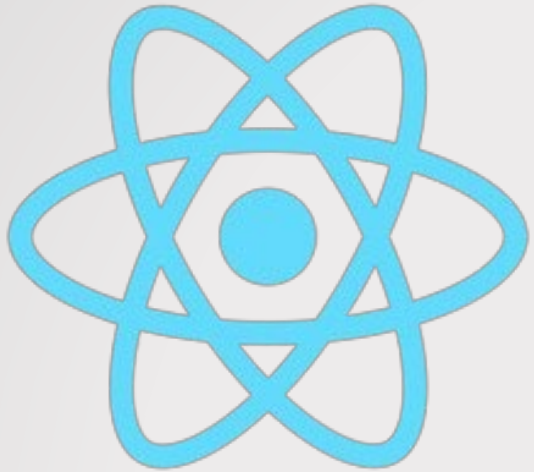


Authentication//Authorization with Keycloak and React App deployment

Presenter:

Software Engineer Tataridis Paris-Rafail

Targets

- ✓ Learn to code in react. (Front-end)
- ✓ Create a simple react app.
- ✓ Learn about Keycloak and its features. (Security)
- ✓ Implement keycloak auth and role-based access control(rbac) in the simple react app.
- ✓ Implement keycloak into the Flash-Energy template.

Everything we use in this internship must follow these rules:

- Open-source communities.
- Reusable modules and procedures.
- Scalable for the production environment.

What is Keycloak?

- Keycloak is an open-source identity and access management solution that provides secure user authentication, authorization, and single sign-on capabilities for applications and services.

What is React?

- React is an open-source JavaScript library used for building user interfaces in web applications, focusing on the efficient rendering of components and managing dynamic UI updates.

What I need to run Keycloak?

- Keycloak runs as an image inside a **Docker** container.
- We use this command in terminal to start the image:

```
docker run -p 8080:8080 -e KEYCLOAK_ADMIN=admin -e KEYCLOAK_ADMIN_PASSWORD=admin
quay.io/keycloak/keycloak:22.0.0 start-dev
```

Full documentation here.

What I need to run React?

- React needs **Node.js** and **mpn** in order to run the app and install all the required dependencies.

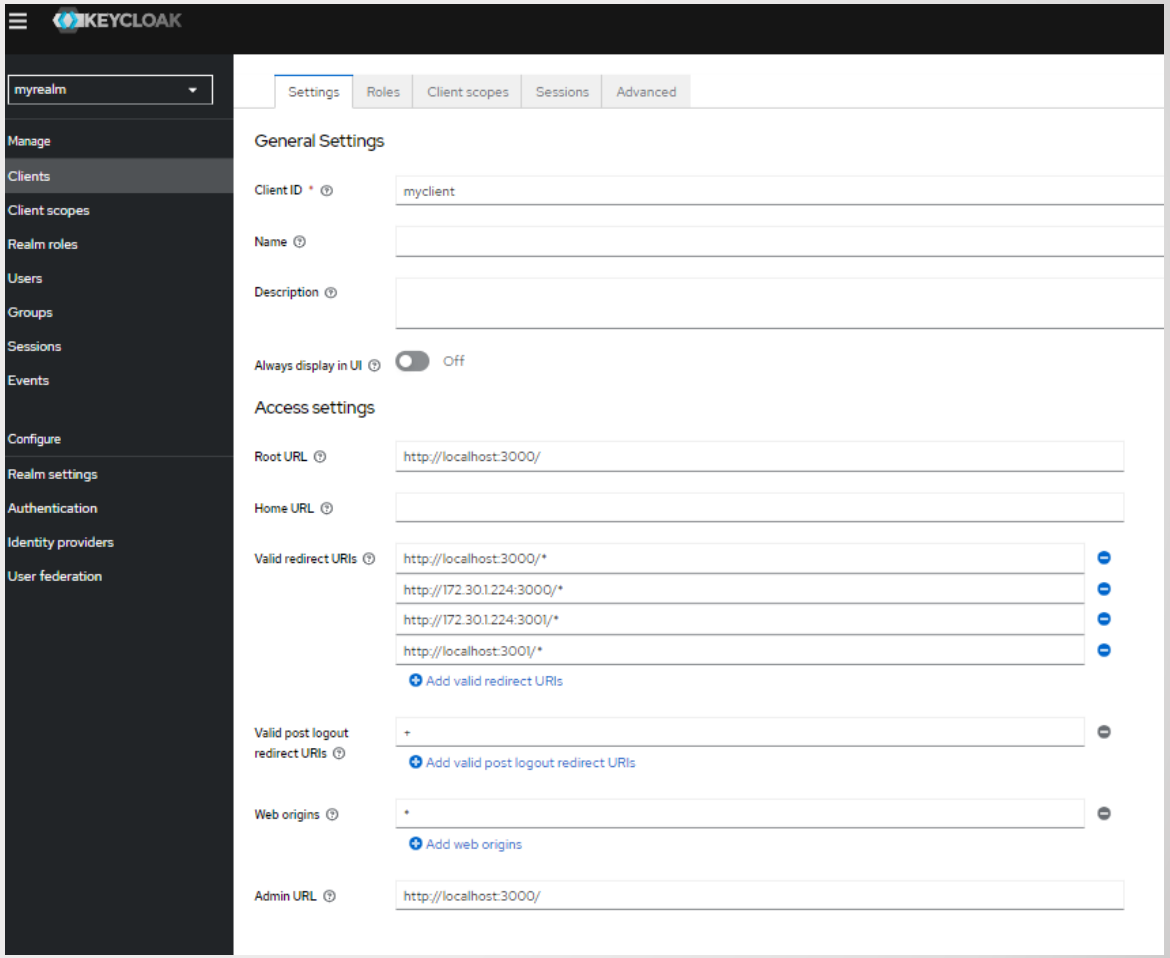
Full code here.

Key keycloak features

Features		Implemented
Single Sign-On (SSO) . Access to multiple applications by a single log in.	✓	**Completed**
Identity Federation. Enabling users to authenticate through their existing accounts.(SAML,Oauth,OpenID…)	✓	**Completed**
User Management and Authentication. Offers user management capabilities like user registration, password reset, and multi-factor authentication.	✓	**Completed**
Realm Authorization and Role-Based Access Control. Admins assign roles to users and control access to specific resources within applications based on user roles and permissions.	−	**Tested**
Social Login Integration. Users to sign in using their social media accounts (like Google, Facebook, and Twitter)	−	**Partial Completion**

How to set app keycloak in order to connect it to react app.

- Keycloak is running at http://localhost:8080/ with the command we used earlier
- By default react runs locally at http://localhost:3000/
- We make sure to add the required urls to the keycloak client/app.
- We are set to call keycloak from react.



Social Media Login

Key React features

Features	Implemented	
Component-Based Architecture. Interfaces are broken down into reusable components.	✓	**Completed**
Virtual DOM. Changes are first made to a lightweight virtual DOM, and then React efficiently updates the real DOM, minimizing direct manipulations and enhancing performance.	✓	**Completed**
Unidirectional Data Flow. It makes it easier to understand how data changes and flows through the application. This helps to prevent unexpected side effects and makes debugging more predictable.	✓	**Completed**
JSX (JavaScript XML). JSX is a syntax extension of JavaScript. It provides a readable and declarative way to define UI elements and their relationships	−	**Rare use**
Rich Ecosystem and Community. Vast open-source community.	✓	**Heavily used!**

How to implement keycloak in react app

- In keycloak we create realms. A realm can contain many apps (or as keycloak calls them clients). Each client/app has a clientId.
- In order to use the correct keycloak realm we have to give react some information like shown in the image.
- After exporting keycloak. We can import it in any file of the application (react) we want.

src > JS Keycloak.js > default

```
1  import Keycloak from "keycloak-js";
2
3  const hostname = window.location.hostname;
4  console.log(window.location);
5
6  const keycloak = new Keycloak(
7      {
8          url: `http://${hostname}:8080/` ,
9          realm: 'myrealm' ,
10         clientId: 'myclient' ,
11     }
12 );
13
14 export default keycloak;
```

1. Import Keycloak

```
import React from "react";
// nodejs library that concatenates classes
import classNames from "classnames";

//keycloak
import { useKeycloak } from "@react-keycloak/web";
```

2. Create login/logout consts

```
const {keycloak} = useKeycloak();

const handleLogin = () => {
  keycloak.login();
};

const handleLogout = () => {
  keycloak.logout();
};
```

3. Call the consts where needed

```
<NavLink tag="li">
  {!keycloak.authenticated && (
  <DropdownItem className="nav-item" onClick={handleLogin}>Log in</DropdownItem>)}
  {console.log(keycloak.tokenParsed)}
  {keycloak.authenticated && (
  <DropdownItem className="nav-item" onClick={handleLogout}>Log out</DropdownItem>)}
</NavLink>
```

Role-Based Access control with react routing.

We create these helpers in order to protect certain routes in our application.
Most importantly they are reusable in ANY app we want to integrate them.

Login Protected Route

```
/*
This helper is used to protect any routes from logged out users.
When a logged out user tries to view the page an alert is pressented that asks him to login.
*/

import React, { useEffect } from "react";
import { useKeycloak } from "@react-keycloak/web";
import { useNavigate } from "react-router-dom";

const ProtectedRoute = ({ children }) => {
 const { keycloak } = useKeycloak();
 const navigate = useNavigate();
 const isLoggedIn = keycloak.authenticated;

 useEffect(() =>{
 if (!isLoggedIn) {
    // Show a browser alert if the user is not logged in
    window.alert("Please log in to view this page!");
    navigate("/admin/dashboard"); // Redirect without full reload
    }
    }, [isLoggedIn, navigate]);

 return <>{children}</>;
};

export default ProtectedRoute;
```

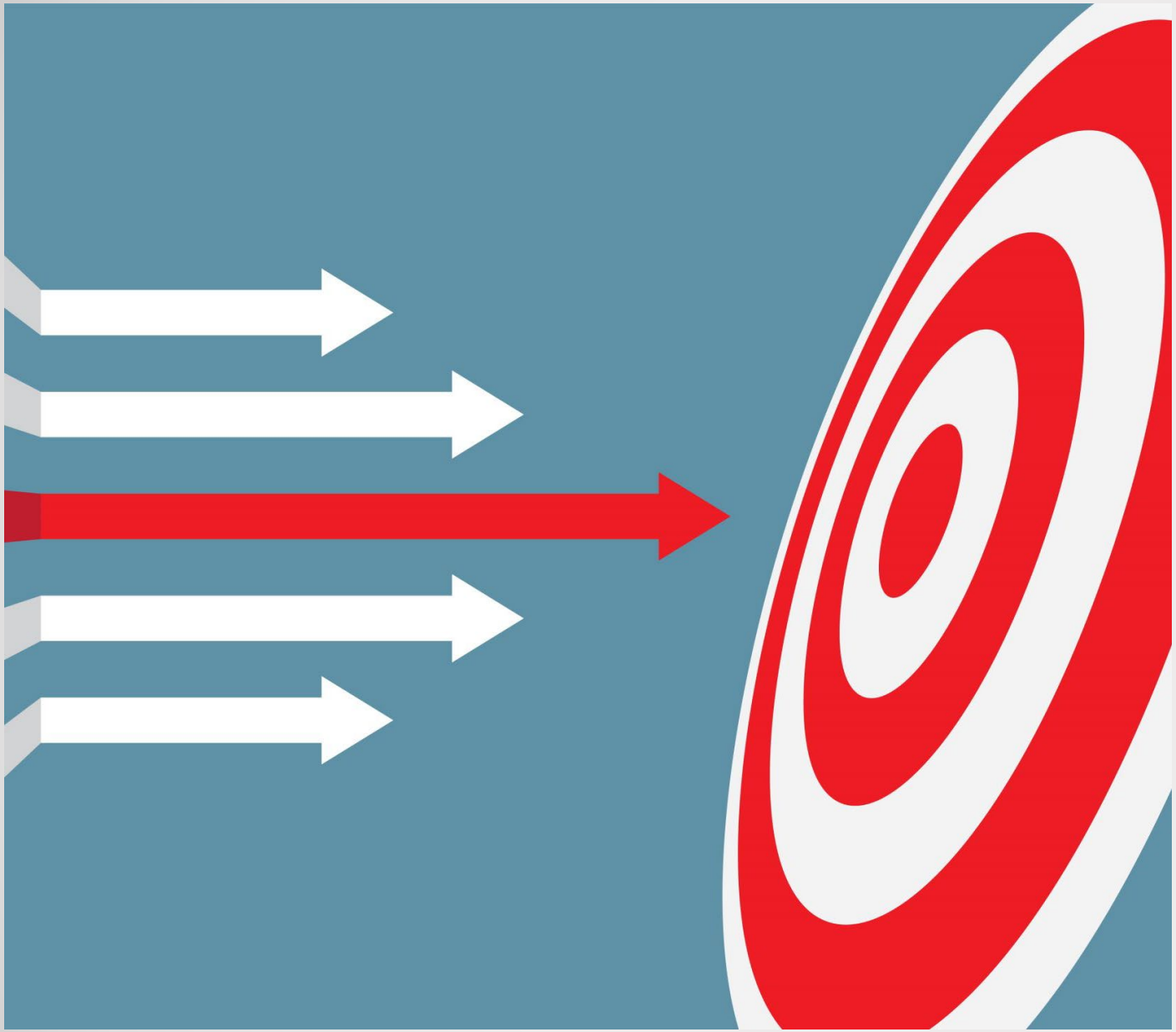
Role Protected Route

```
/*
This helper is used to protect any routes from unauthorized users (based on their role).
When an unauthorized user tries to view the page an alert is pressented that asks the user to have the required role.
*/

import React, { useEffect } from "react";
import { useKeycloak } from "@react-keycloak/web";
import { useNavigate } from "react-router-dom";

const RoleProtectedRoute = ({ children }) => {
 const { keycloak } = useKeycloak();
 const navigate = useNavigate();
 const isLoggedIn =
    keycloak.authenticated &&
    keycloak.tokenParsed?.realm_access?.roles.includes("RealmAdmin");

 useEffect(() =>{
 if (!isLoggedIn) {
    // Show a browser alert if the user is not logged in
    window.alert("Only Admins are allowed to view this page. Ask to upgrade your role!");
    navigate("/admin/dashboard"); // Redirect without full reload
    }
    }, [isLoggedIn, navigate]);

 return <>{children}</>;
};

export default RoleProtectedRoute;
```

```
//protected routes
import ProtectedRoute from "./helpers/ProtectedRoute";
import RoleProtectedRoute from "helpers/RoleProtectedRoute";


var routes = [
  {
    path: "/dashboard",
    name: "Dashboard",
    icon: "tim-icons icon-chart-pie-36",
    component: <Dashboard />,
    layout: "/admin",
  },
  {
    path: "/icons",
    name: "Icons",
    icon: "tim-icons icon-atom",
    component: <Icons />,
    layout: "/admin",
  },
  {
    path: "/map",
    name: "Map",
    icon: "tim-icons icon-pin",
    component: (<ProtectedRoute><Map /></ProtectedRoute>),
    layout: "/admin",
  },
  {
    path: "/notifications",
    name: "Notifications",
    icon: "tim-icons icon-bell-55",
    component: (<ProtectedRoute><Notifications /></ProtectedRoute>),
    layout: "/admin",
  },
  {
    path: "/user-profile",
    name: "User Profile",
    icon: "tim-icons icon-single-02",
    component: (<RoleProtectedRoute><UserProfile /></RoleProtectedRoute>),
    layout: "/admin",
  },
  {
    path: "/tables",
    name: "Table List",
    icon: "tim-icons icon-puzzle-10",
    component: (<RoleProtectedRoute><TableList /></RoleProtectedRoute>),
    layout: "/admin",
```

Securing our routes.

- We import the helpers.
- Protect the Routes with the required security.
- Either login based or role based.



Future Goals

Short Term

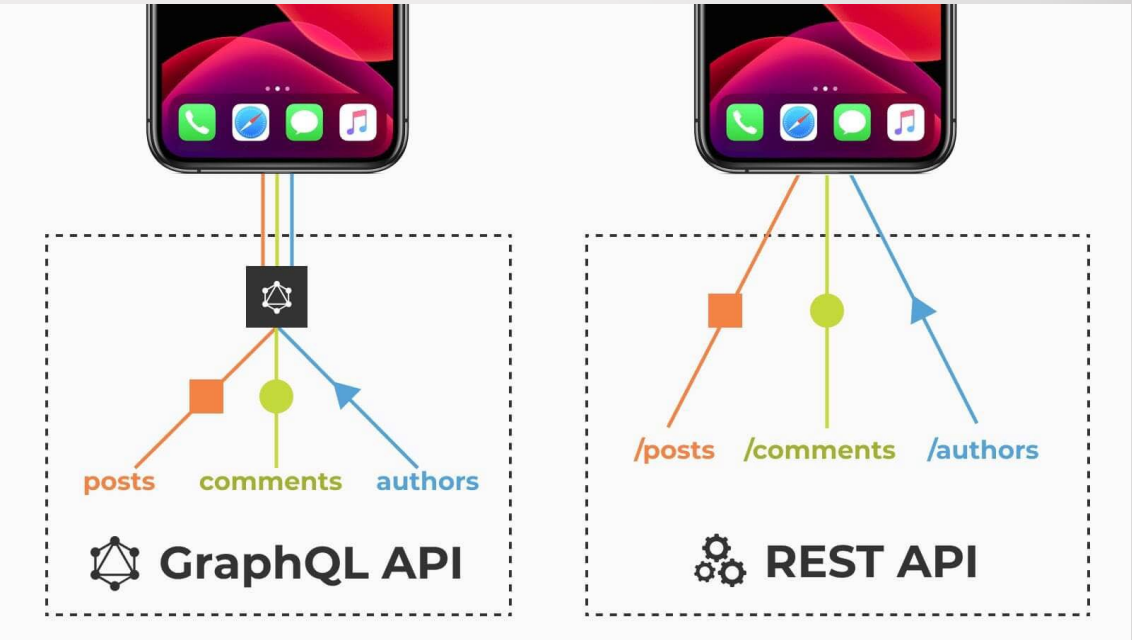
- Add user personalized widgets with api requests using graphql.
- Fully integrate the Flash-Energy project under Keycloak authentication.

Long Term

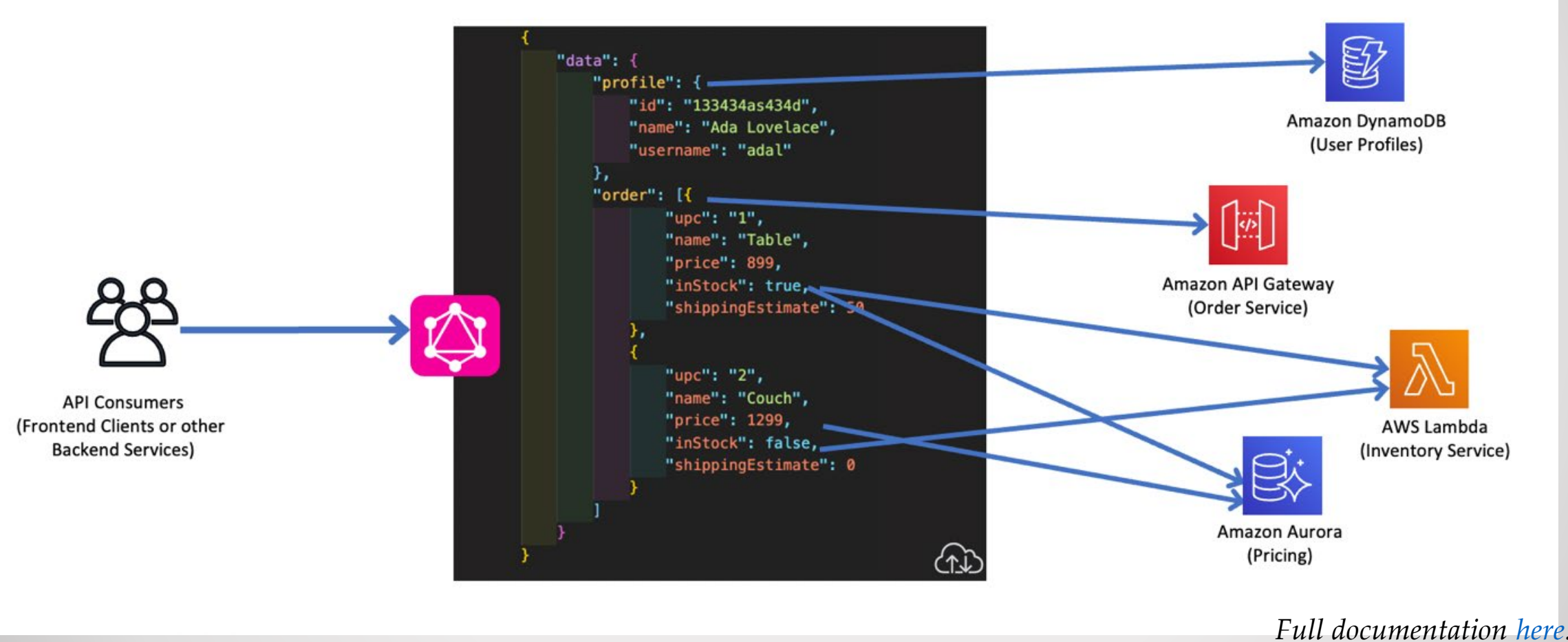
- Learn how to make UI components into a library for usability.
- Expand my security knowledge.
- Create and Secure a mobile app.

What is GraphQL?

- GraphQL is a query language for APIs that enables more efficient and flexible communication between clients and servers by allowing clients to request specific data structures and reducing over-fetching or under-fetching of information.



Top Benefits of GraphQL



Full documentation here.

Thank you for your time!

I would like to thank UniSystems for providing everything we needed to develop this application to this point.

- Made by **Tataridis Paris-Rafail**, intern (July-August)
- Project Coordinator **Ilias Aliferis**
- **Keycloak auth in React Project 2023 @Unisystems, for a better future.**