Report on

Refactoring Duplicated Components in the SIGMA Project Developed by ReactJS

First Step: API Calls Module

Related to the tasks SIG-126 and MOJ-202

Mohammad Mollaahmadi
MOJ Secure Company
Tehran, Iran
Jul 2024

Introduction

The SIGMA project, developed using the ReactJS library, currently contains multiple instances of duplicated components. This redundancy increases maintenance efforts and can lead to inconsistencies across the codebase. We recently reviewed the project's current state and devised a three-step plan for refactoring: the API calls module, the routes handling hook, and the screen components.

This document discusses our approach to addressing the first section: the API calls module, and presents the results of our changes. We focused on ensuring seamless API integration by creating a unified API caller to achieve our goal. Integrating all API calls into a single module allows the refactored components to consistently use this unified caller, enhancing maintainability and reducing redundancy.

Implementation

To streamline the API calls module in the SIGMA project, we proposed developing a centralized module to manage different endpoints based on their respective base URLs. The following steps outline our implementation:

Step 1: Develop a General Module for API Calls

- **Objective**: Create a centralized module that handles API calls and integrates with the Redux Toolkit for efficient state management.
- **Implementation**:
    - We organized the API-related code within the `src/api` directory, which includes `urls`, `headers`, and `general-api-caller`.
    - Developed a general API caller module to call all the services behind the gateway and integrated it with the Redux Toolkit.

Note: In the future, we can create the same module to call services behind other addresses.

Step 2: Integrate the API Caller into the Components

- **Objective**: Utilize the newly developed API caller across all relevant components.
- **Implementation**:
    - Integrated the API caller into components to replace direct API calls with the centralized module.

Here is an example of this integration:

```
src > pages > import > viewFile > billoflading > businessDeclarationTable > JS index.jsx > ...
      You, 4 minutes ago | 5 authors (andishe abbasian and others)
1     import { RouteMap } from '@/RouteMap';
2     import { useLazyGetDataQuery } from '@/api/services/gateway';
3     import { gatewayURIs } from '@/api/urls';
4     import { EyeIcon } from '@/assets/icon';
5     import Table from '@/components/table';
6     import MergedTr from '@/components/table/mergedTr/MergedTr';
7     import { useSelector } from 'react-redux';
8     import { useLocation, useNavigate } from 'react-router-dom';
9
10    const BusinessDescriptionTable = ({ requestNumber }) => {
11        const [getData, resultData] = useLazyGetDataQuery();
12        const navigate = useNavigate();
13        const { state } = useLocation();
14        const location = useLocation();
15
16  >     let infoTable = [ …
108       ];
109       const { selectedDeclarationVersion } = useSelector((state) => state.declarationVersion);
110       return (
111           <div className="mb-10">
112               <Table
113                   infoTable={infoTable}
114                   url={`${gatewayURIs["import"].query}/Internal/${requestNumber}/${selectedDeclarationVersion}/bill-of-ladings-list`}
115                   noFilter
116                   getData={getData}
117                   resultData={resultData}
118                   showPagination={false}
119                   data={resultData?.data?.data?.list}
120                   pageSize={resultData?.data?.data?.pageSize}
121                   totalPage={resultData?.data?.data?.totalPages}
122                   totalCount={resultData?.data?.data?.totalCount}
123                   currentPage={resultData?.data?.data?.currentPage}
124               />
125           </div>
126       );
127   };
128
129   export default BusinessDescriptionTable;
130
```

Note: After using a single component instead of all duplicated components, we have to use a variable instead of “import” to make it dynamic.

Conclusion

The following report outlines the work completed on the API calls module refactoring for the SIGMA project. The work was conducted on the git branch `MOJ-202-api-caller-module` over approximately 29 hours. This refactoring involved significant changes across the codebase and resulted in improved maintainability and reduced redundancy.

Overview of Changes

- **Total Files Modified**: 329 files
- **Insertions**: 1,542 lines
- **Deletions**: 2,500 lines
- **Duplicate Removal**: Successfully removed a set of duplicate components identified by the script mentioned in the previous report.

src/api/services/Gateway/index.js
src/api/services/ImportCommand/index.js
src/api/services/ImportQuery/index.js
src/api/services/PortalDocumentsQuery/index.js
src/api/services/ThroughTransportCommand/index.js
src/api/services/ThroughTransportQuery/index.js
src/api/services/bascule/index.js
src/api/services/csc/index.js
src/api/services/csq/index.js
src/api/services/exq/index.js
src/api/services/file/index.js
src/api/services/finc/index.js
src/api/services/finq/index.js
src/api/services/gatec/index.js
src/api/services/gateq/index.js
src/api/services/iamapi/index.js
src/api/services/mdm/index.js
src/api/services/mock/index.js
src/api/services/rtnq/index.js
src/api/services/tec/index.js
src/api/services/teq/index.js
src/api/services/trc/index.js
src/api/services/trq/index.js
src/api/services/value-management/index.js