Overview

The **lib microservice** is a simplified file manager providing graphQL API. It has three features:

- provide a listing of directory contents.
- transfer a file to user.
- Source files can either come from local file system or from a gitlab instance.

Gitlab setup

For this microserivce to be functional, a certain directory or gitlab project structure is expected. The microservice expects that the gitlab consisting of one group, DTaaS, and within that group, all of the projects be located, **user1**, **user2**, ... , as well as a **commons** project. Each project corresponds to files of one user. A sample file structure can be seen in gitlab dtaas group. You can visit the gitlab documentation on groups for help on the management of gitlab groups.

You can clone the git repositories from the `dtaas` group to get a sample file system structure for the lib microservice.

Configuration setup

The microservices uses `.env` environment files to receive configuration.

In order to create this environment, you need to create a `.env` file, wherein you create the following environment variables, and insert with the correct-information relevant for your setup:

```
PORT='4001'
MODE='local' or 'gitlab'
LOCAL_PATH ='/Users/<Username>/DTaaS/files'
GITLAB_GROUP ='dtaas'
GITLAB_URL='https://gitlab.com/api/graphql'
TOKEN='123-sample-token'
LOG_LEVEL='debug'
TEST_PATH='/Users/<Username>/DTaaS/servers/lib/test/data/test_assets'
APOLLO_PATH='/lib' or ''
GRAPHQL_PLAYGROUND='false' or 'true'
```

The `TOKEN` should be set to your GitLab Group access API token. For more information on how to create and use your access token, gitlab page.

Once you've generated a token, copy it and replace the value of `TOKEN` with your token for the gitlab group, can be found.

Developer Commands

```
yarn install    # Install dependencies for the microservice
yarn build      # build the application
yarn start      # start the application
```

You can press `Ctl+C` to halt the application. If you wish to run the microservice in the background, use

```
nohup yarn start & disown
```

Developer Commands

```
yarn install    # Install dependencies for the microservice
yarn syntax     # analyzes source code for potential errors, style
violations, and other issues,
yarn build      # compile ES6 files into ES5 javascript files and copy all
JS files into build/ directory
yarn test -a      # run all tests
yarn test -e      # run end-to-end tests
yarn test -i      # run integration tests
yarn test -u      # run unit tests
yarn start      # start the application
yarn clean      # deletes directories "build", "coverage", and "dist"
```

Service Endpoint

The URL endpoint for this microservice is located at: `localhost:PORT/lib`

GraphQL API Calls

The lib microservice takes two distinct GraphQL queries.

Directory Listing

This query receives directory path and provides list of files in that directory. A sample query and response are given here.

```
query {
  listDirectory(path: "user1") {
    repository {
      tree {
        blobs {
          edges {
            node {
              name
              type
            }
          }
        }
        trees {
          edges {
            node {
              name
              type
            }
          }
        }
      }
    }
  }
}
```

```
{
  "data": {
    "listDirectory": {
      "repository": {
        "tree": {
          "blobs": {
            "edges": []
          },
          "trees": {
            "edges": [
              {
                "node": {
                  "name": "common",
                  "type": "tree"
                }
              },
              {
                "node": {
                  "name": "data",
                  "type": "tree"
                }
              },
              {
                "node": {
                  "name": "digital twins",
                  "type": "tree"
                }
              },
              {
                "node": {
                  "name": "functions",
                  "type": "tree"
                }
              },
              {
                "node": {
                  "name": "models",
                  "type": "tree"
                }
              },
              {
                "node": {
                  "name": "tools",
                  "type": "tree"
                }
              }
            ]
          }
        }
      }
    }
  }
}
```

Fetch a file

This query receives directory path and send the file contents to user in response. A sample query and response are given here.

```
query {
  readFile(path: "user2/data/sample.txt") {
    repository {
      blobs {
        nodes {
          name
          rawBlob
          rawTextBlob
        }
      }
    }
  }
}
```

```
{
  "data": {
    "readFile": {
      "repository": {
        "blobs": {
          "nodes": [
            {
              "name": "sample.txt",
              "rawBlob": "hello world",
              "rawTextBlob": "hello world"
            }
          ]
```

```
                }
            }
        }
    }
}
```