

Frontend Developer Trial Task

This trial task is simple from features perspective and aims mostly on technical solution and approaches used to solve some common problems. Besides the technology requirements listed below, it's up to a candidate what open source libraries and solutions to choose.

Introduction

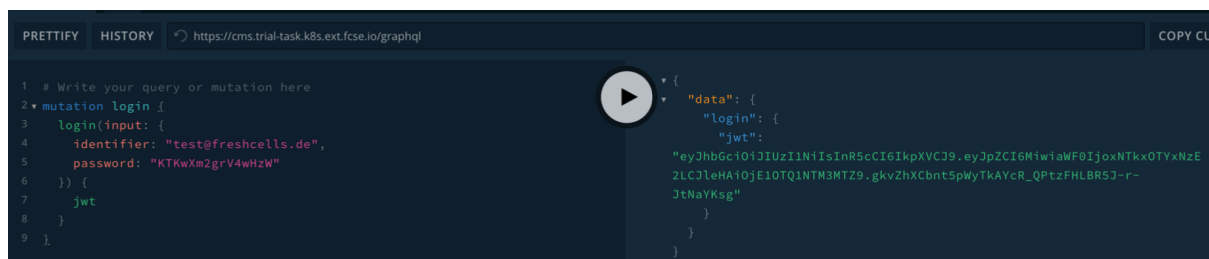
Backend details for integration with the trial task:

Backend GraphQL endpoint - <https://cms.trial-task.k8s.ext.fcse.io/graphql> Trial user email and password - should be provided along with the trial task

There are two RPCs to use at backend GraphQL endpoint:

- login GraphQL mutation

Example:



The screenshot shows the GraphQL Playground interface. The URL bar displays `https://cms.trial-task.k8s.ext.fcse.io/graphql`. The query editor on the left contains the following mutation:

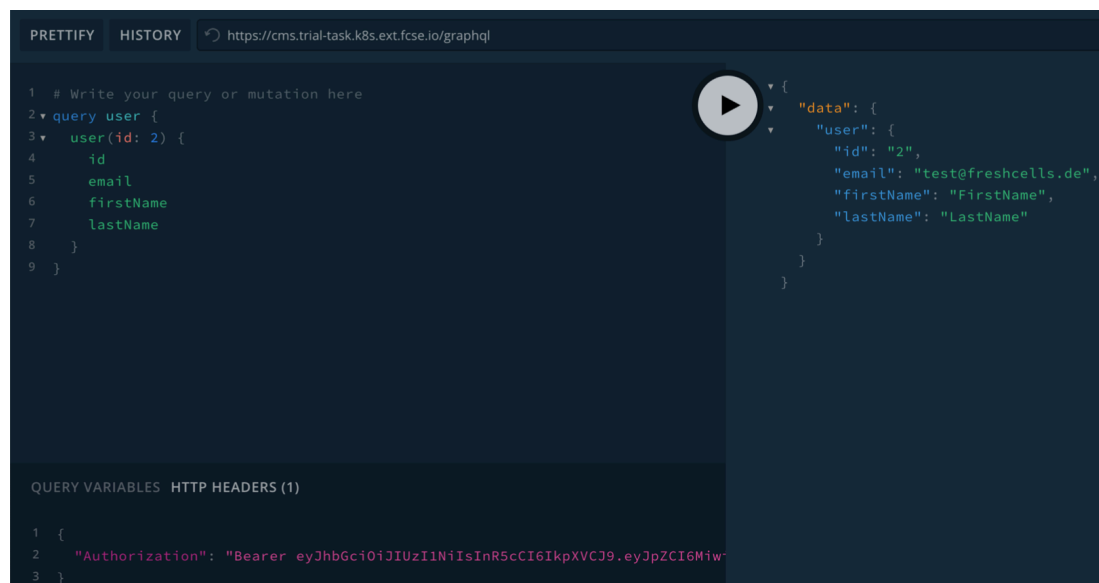
```
1 # Write your query or mutation here
2 mutation login {
3   login(input: {
4     identifier: "test@freshcells.de",
5     password: "KTKwXm2grV4wHzW"
6   }) {
7     jwt
8   }
9 }
```

The results pane on the right shows the JSON response:

```
{
  "data": {
    "login": {
      "jwt": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1wZW91b3R5b250YXNzE2LCJleHAiOiE1OTQ1NTM3MTZ9.gkvZhxXbnt5pWytKAYcR_QPtzFHLBR5J-r-JtNaYKsg"
    }
  }
}
```

- user GraphQL query

Example:



The screenshot shows the GraphQL Playground interface. The URL bar displays `https://cms.trial-task.k8s.ext.fcse.io/graphql`. The query editor on the left contains the following query:

```
1 # Write your query or mutation here
2 query user {
3   user(id: 2) {
4     id
5     email
6     firstName
7     lastName
8   }
9 }
```

The results pane on the right shows the JSON response:

```
{
  "data": {
    "user": {
      "id": "2",
      "email": "test@freshcells.de",
      "firstName": "FirstName",
      "lastName": "LastName"
    }
  }
}
```

Below the results pane, the 'QUERY VARIABLES' section is expanded, showing the following variables:

```
1 {
2   "Authorization": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1wZW91b3R5b250YXNzE2LCJleHAiOiE1OTQ1NTM3MTZ9.gkvZhxXbnt5pWytKAYcR_QPtzFHLBR5J-r-JtNaYKsg"
3 }
```

GraphQL schema with query, mutation and data type details can be reviewed at GraphQL playground by opening GraphQL endpoint in a web- browser.

The implemented web application can be shared as github or bitucket repositories or sent in archive with readme instructions on how to launch it.

Technologies

The required technologies to use are:

- JavaScript
- ReactJS
- GraphQL

Requirements

Application should contain 2 screens to login and display account details.

UI

Please use the attached mockup as a reference (as it is not a figma mockup, don't spend a lot of time on making it 100% pixel perfect).

Helvetica font is used, base font is 16px , 32px title, 14px button and a note.

Base font color is #333, secondary #777, accent #FF6F61

Background is #F9F9F9, borders are #E5E5E5

It is recommended not to use any frameworks/libraries for UI (like MaterialUI, Bootstrap), stick to css in any preferable form (preprocessed or not, with/without CSS modules etc).

Herzlich Willkommen

"Meine Probeaufgabe"

E-mail

Bitte verwenden Sie die GraphQL-Mutation mit den angegebenen Anmeldeinformationen und vergessen Sie nicht, Backend-Fehler zu behandeln, falls Sie ein falsches E-Mail-Login / Passwort angegeben haben.

Passwort

LOGIN

Login Screen

Login screen should contain email and password text inputs and login button. On login button click the inputs should be validated according to the rules below:

- email - required, matches email pattern
- password - required

Please use the login GraphQL mutation with provided credentials and please don't forget about handling backend errors in case of wrong email / password provided. Successful login should lead to the Account screen.

Account Screen

Account screen should contain 2 non-editable text fields displaying logged-in user first name and last name and logout button. Please use user GraphQL query to get user's firstName and lastName data.

Clicking the logout button should logout the user and switch the application to the login screen.

Preferences

Would be a plus:

- Flow or TypeScript
- SPA navigation
- Localization
- Uncaught error handling
- UX feedback (loading states, error notifications)
- Clean UI
- Unit tests