САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа 2

Выполнил:

Пономарев Константин

К33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

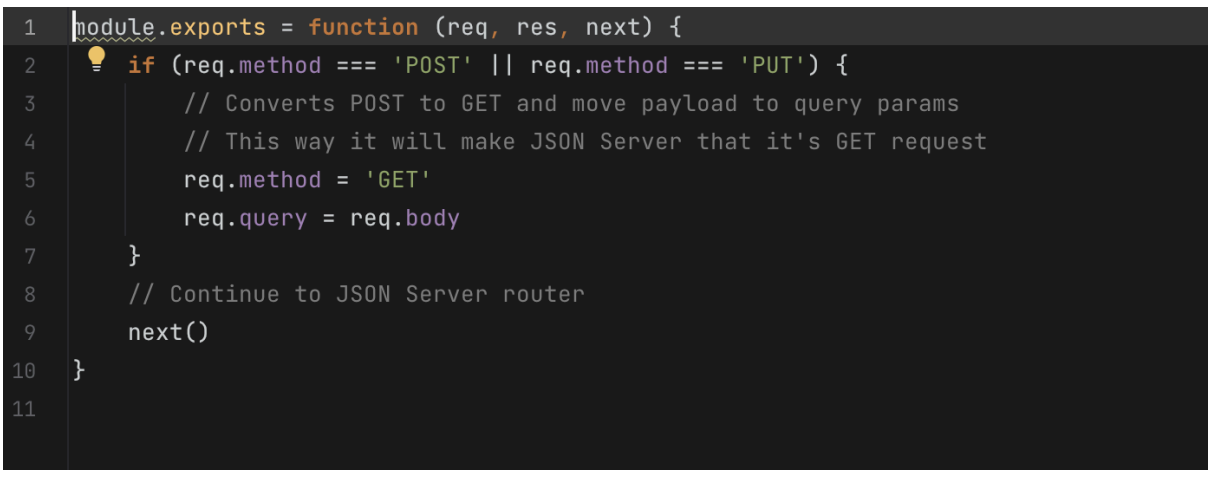
Варианты остаются прежними. Теперь Вам нужно привязать то, что Вы делали в ЛР1 к внешнему API средствами fetch/axios/xhr.

Ход работы

Для начала я подключил нужные либы, в моем случае это был axios и json-server для моков. После установки приступил к настройке. Прежде всего надо настроить моки

```
"registration": {
  "success": true,
  "response": {
    "token": "someReallyCoolToken",
    "userId": "first-cool-user"
  }
},
"bodyParams": {
  "success": true,
  "response": {
    "calories": 2500,
    "proteins": 100,
    "fats": 100,
    "carbohydrates": 100
  }
},
"calculatedRation": {
  "success": true,
  "response": {
    "calories": 2500,
    "proteins": 100,
    "fats": 100,
    "carbohydrates": 100
  }
},
```

Однако на этапе отправки POST и PUT запросов, я понял, что эта штука работает не как Proxyman и Charles, а просто перезаписывало в db.json request body и пришлось заюзать такой костыль с middleware



```
module.exports = function (req, res, next) {
    if (req.method === 'POST' || req.method === 'PUT') {
        // Converts POST to GET and move payload to query params
        // This way it will make JSON Server that it's GET request
        req.method = 'GET'
        req.query = req.body
    }
    // Continue to JSON Server router
    next()
}
```

Для отправки запросов я использовал дефолтные методы из axios (post, put, get)

```
authUser = async (userData) => {
    return await this.API.post(
        '/auth',
        userData,
        {
            headers: {
                'Content-Type': 'application/json'
            }
        }
    )
}

2 usages  ko.ponomarev
sendBodyParameters = async (userId, bodyParameters) => {
    return await this.API.put(
```

Создание инстанса

```
const apiURL = 'http://localhost:3000'

const instance = axios.create({
    baseURL: apiURL
})
// Add a request interceptor
ko.ponomarev
instance.interceptors.request.use(function (config) {
    // Do something before request is sent
    console.log("request: " + JSON.stringify(config))
    return config;
}, function (error) {
    // Do something with request error
    console.log("request err: " + JSON.stringify(error))
    return Promise.reject(error);
});

// Add a response interceptor
ko.ponomarev
instance.interceptors.response.use(function (response) {
    // Any status code that lie within the range of 2xx cause this function to trigger
    // Do something with response data
    console.log("response:" + JSON.stringify(response))
    return response;
}, function (error) {
    // Any status codes that falls outside the range of 2xx cause this function to trigger
    // Do something with response error
    console.log("response err:" + JSON.stringify(error))
    return Promise.reject(error);
});
```

Дополнительно добавил interceptors для логирования запросов и ответов, очень помогало отлавливать неправильный мок json.

Связывал с ui слоем через store

```
actions: {
    async loadData() {
        try {
            const userId = useUserInfoStore().userInfo.userId
            const calculatedRationWithDailyAteFood = await caloriesRepository.getCalculatedCalories(userId)
            this.setCalculatedRation(calculatedRationWithDailyAteFood)
        } catch (e) {
            console.log(e)
        }
    },
```

Или через стандартное использование Vue

```
data() {
  return {
    form: {
      sex: Sex.MALE,
      height: 0,
      weight: 0,
      age: 0,
      bodyActivity: BodyActivity.LOW,
      weightGoal: WeightGoal.LOSS,
      formula: Formula.HARRIS
    },
    uiState: {
      showLoading: false,
      wasValidated: false,
    }
  }
},
methods: {
  async prefilledParams() {
    try {
      const userId = useUserInfoStore().userInfo.userId
      this.form = await caloriesRepository.getBodyParameters(userId)
```

Вывод

По итогу удалось поработать с axios, настроить запросы для получения разной информации необходимой для ui сайта. Также было интересно поработать с json-server, который позволил очень просто сделать моки.

Помимо этого я узнал, как вообще создаются инстансы апи и обращения к ним