

|  |  |
| --- | --- |
| Student name: | Altynbekov Ali, Akhmetova Aruzhan, Ashimkhan Temirlan |
| Group name: | Group 2 |
| Teacher name: | Kamshat Asmaganbetova |
| Project name: | Weather |
| Project type: | Group / Individual |
| Number of words | Must be at least 1500 words, but no more than 2000 words |

**INTRODUCTION**

Regardless of what kind of website we conceived – a one-page or 15-20 pages - it was necessary to understand that it requires investment of time and effort. After all, this is a kind of real estate – our private property, the construction and arrangement of which we personally manage. And it is our responsibility to make sure that the site will turn out to be as high-quality and convenient for use as possible

**MAIN PART**

We started by identifying the target audience and analyzing it. It is important to understand your target audience. Then he will be able to adapt his website to their interests and get his own benefit from it. We have determined the purpose of our site. And this is the weather.

An extremely important step is the creation of the site itself or its prototype. For clarity, we created a prototype with a detailed arrangement of elements and navigation. Provided the most user-friendly interface that we could do

After the completion of the designer's work, we began to make up the site. In general, we adapted the layout of the site. All the site attributes made by the designer and assembled by the layout designer were written in the programming language. The stage of layout and programming was performed simultaneously, as the scale of the project allows.

The content requires stable updating and filling, as a result of which we have used kazhydromet information.

We can't make the site without testing. The site has been repeatedly tested to fix bugs in the end.

**CONCLUSION**

It was difficult and long. But by following the rules and the plan, we were able to make our project.

import axios from "axios";

import authHeader from './auth-header';

const API\_URL = "http://localhost:8000/api/v1/";

// axios.defaults.baseURL = API\_URL;

// axios.defaults.headers.post['Content-Type'] = 'application/json;charset=utf-8';

// axios.defaults.headers.post['Access-Control-Allow-Origin'] = '\*';

class AuthService {

login(username, password) {

console.log(username, password);

return axios

.post(API\_URL + "token/", {

username,

password

})

.then(response => {

console.log(response.data);

if (response.data.access) {

localStorage.setItem("user", JSON.stringify(response.data));

}

console.log(response.data.accessToken);

return response.data;

});

}

logout() {

localStorage.removeItem("user");

}

register(username, email, password) {

return axios.post(API\_URL + "signup", {

username,

email,

password

});

}

refresh() {

return axios

.post(

API\_URL + "token/refresh/", {

"access": localStorage.getItem("access"),

"refresh": localStorage.getItem("refresh")

}

)

.then(response => {

console.log(response.data);

})

}

getCurrentUser() {

console.log(API\_URL + `auth/profile/`);

return axios

.get(API\_URL + 'auth/profile/', { headers: authHeader() });

}

}

export default new AuthService();

export default function authHeader() {

const user = JSON.parse(localStorage.getItem('user'));

if (user && user.access) {

return {

'Content-Type': 'application/json',

'Accept': 'application/json',

'Authorization': 'Bearer ' + user.access

}; // for Spring Boot back-end

// return { 'x-access-token': user.accessToken }; // for Node.js Express back-end

} else {

return {};

}

}

import axios from 'axios';

import authHeader from './auth-header';

const API\_URL = 'http://localhost:8000/api/v1/';

class UserService {

getPublicContent() {

return axios.get(API\_URL + 'all');

}

getUserBoard() {

return axios.get(API\_URL + 'user', { headers: authHeader() });

}

getModeratorBoard() {

return axios.get(API\_URL + 'mod', { headers: authHeader() });

}

getAdminBoard() {

return axios.get(API\_URL + 'admin', { headers: authHeader() });

}

}

That is services code as the example, it requests to API to get response. We made site on javascript language. And as you can see we learned the new programming language to make this site.

|  |
| --- |
| import React, { Component } from "react"; |
|  | import AuthService from "../services/auth.service"; |
|  |  |
|  |  |
|  | export default class Profile extends Component { |
|  | constructor(props) { |
|  | super(props); |
|  |  |
|  |  |
|  | this.state = { |
|  | content: { |
|  | "noting": 'fasdfsdf' |
|  | } |
|  | }; |
|  | } |
|  |  |
|  |  |
|  | componentDidMount() { |
|  | AuthService.getCurrentUser().then( |
|  | response => { |
|  | console.log(response.data); |
|  | this.setState({ |
|  | content: response.data |
|  | }); |
|  | }, |
|  | error => { |
|  | this.setState({ |
|  | content: |
|  | (error.response && error.response.data) || |
|  | error.message || |
|  | error.toString() |
|  | }); |
|  | } |
|  | ); |
|  | } |
|  |  |
|  |  |
|  | render() { |
|  | const { content } = this.state; |
|  | console.log(content); |
|  | console.log("current user"); |
|  | // const [x, setX] = useState(false); |
|  |  |
|  |  |
|  | return ( |
|  | <div className="container"> |
|  | <header className="jumbotron"> |
|  | <h3> |
|  | {/\* <strong>{currentUser.username}</strong> Profile \*/} |
|  | </h3> |
|  | </header> |
|  | <p> |
|  | <strong>Логин:</strong>{" "} |
|  | {content.username} |
|  | {/\* {content.email} \*/} |
|  | </p> |
|  | <p> |
|  | <strong>Личное имя:</strong>{" "} |
|  | {content.first\_name} |
|  | </p> |
|  | <p> |
|  | <strong>Фамилия:</strong>{" "} |
|  | {content.last\_name} |
|  | </p> |
|  | <p> |
|  | <strong>Почта:</strong>{" "} |
|  | {content.email} |
|  | </p> |
|  | <p> |
|  | <strong>Сотрудник канцелярий:</strong>{" "} |
|  | <input type="checkbox" checked={content.is\_staff} /> |
|  | </p> |
|  | {/\* <strong>Authorities:</strong> \*/} |
|  | {/\* <ul> |
|  | {currentUser.roles && |
|  | currentUser.roles.map((role, index) => <li key={index}>{role}</li>)} |

This is component to display data from rest API. There we are declaring the initial state of the data. In the componentditmount we return to service to request API to get response or data. And then displaying them in HTML tags.

|  |  |
| --- | --- |
| import React, { Component } from "react"; | |
|  | | import Form from "react-validation/build/form"; | |
|  | | import Input from "react-validation/build/input"; | |
|  | | import CheckButton from "react-validation/build/button"; | |
|  | |  | |
|  | | import AuthService from "../services/auth.service"; | |
|  | |  | |
|  | | const required = value => { | |
|  | | if (!value) { | |
|  | | return ( | |
|  | | <div className="alert alert-danger" role="alert"> | |
|  | | This field is required! | |
|  | | </div> | |
|  | | ); | |
|  | | } | |
|  | | }; | |
|  | |  | |
|  | | export default class Login extends Component { | |
|  | | constructor(props) { | |
|  | | super(props); | |
|  | | this.handleLogin = this.handleLogin.bind(this); | |
|  | | this.onChangeUsername = this.onChangeUsername.bind(this); | |
|  | | this.onChangePassword = this.onChangePassword.bind(this); | |
|  | |  | |
|  | | this.state = { | |
|  | | username: "", | |
|  | | password: "", | |
|  | | loading: false, | |
|  | | message: "" | |
|  | | }; | |
|  | | } | |
|  | |  | |
|  | | onChangeUsername(e) { | |
|  | | this.setState({ | |
|  | | username: e.target.value | |
|  | | }); | |
|  | | } | |
|  | |  | |
|  | | onChangePassword(e) { | |
|  | | this.setState({ | |
|  | | password: e.target.value | |
|  | | }); | |
|  | | } | |
|  | |  | |
|  | | handleLogin(e) { | |
|  | | e.preventDefault(); | |
|  | |  | |
|  | | this.setState({ | |
|  | | message: "", | |
|  | | loading: true | |
|  | | }); | |
|  | |  | |
|  | | this.form.validateAll(); | |
|  | |  | |
|  | | if (this.checkBtn.context.\_errors.length === 0) { | |
|  | | AuthService.login(this.state.username, this.state.password).then( | |
|  | | () => { | |
|  | | this.props.history.push("/profile"); | |
|  | | window.location.reload(); | |
|  | | }, | |
|  | | error => { | |
|  | | const resMessage = | |
|  | | (error.response && | |
|  | | error.response.data && | |
|  | | error.response.data.message) || | |
|  | | error.message || | |
|  | | error.toString(); | |
|  | |  | |
|  | | this.setState({ | |
|  | | loading: false, | |
|  | | message: resMessage | |
|  | | }); | |
|  | | } | |
|  | | ); | |
|  | | } else { | |
|  | | this.setState({ | |
|  | | loading: false | |
|  | | }); | |
|  | | } | |
|  | | } | |
|  | |  | |
|  | | render() { | |
|  | | return ( | |
|  | | <div className="col-md-12"> | |
|  | | <div className="card card-container"> | |
|  | | <img | |
|  | | src="//ssl.gstatic.com/accounts/ui/avatar\_2x.png" | |
|  | | alt="profile-img" | |
|  | | className="profile-img-card" | |
|  | | /> | |
|  | |  | |
|  | | <Form | |
|  | | onSubmit={this.handleLogin} | |
|  | | ref={c => { | |
|  | | this.form = c; | |
|  | | }} | |
|  | | > | |
|  | | <div className="form-group"> | |
|  | | <label htmlFor="username">Username</label> | |
|  | | <Input | |
|  | | type="text" | |
|  | | className="form-control" | |
|  | | name="username" | |
|  | | value={this.state.username} | |
|  | | onChange={this.onChangeUsername} | |
|  | | validations={[required]} | |
|  | | /> | |
|  | | </div> | |
|  | |  | |
|  | | <div className="form-group"> | |
|  | | <label htmlFor="password">Password</label> | |
|  | | <Input | |
|  | | type="password" | |
|  | | className="form-control" | |
|  | | name="password" | |
|  | | value={this.state.password} | |
|  | | onChange={this.onChangePassword} | |
|  | | validations={[required]} | |
|  | | /> | |
|  | | </div> | |
|  | |  | |
|  | | <div className="form-group"> | |
|  | | <button | |
|  | | className="btn btn-primary btn-block" | |
|  | | disabled={this.state.loading} | |
|  | | > | |
|  | | {this.state.loading && ( | |
|  | | <span className="spinner-border spinner-border-sm"></span> | |
|  | | )} | |
|  | | <span>Login</span> | |
|  | | </button> | |
|  | | </div> | |
|  | |  | |
|  | | {this.state.message && ( | |
|  | | <div className="form-group"> | |
|  | | <div className="alert alert-danger" role="alert"> | |
|  | | {this.state.message} | |
|  | | </div> | |
|  | | </div> | |
|  | | )} | |
|  | | <CheckButton | |
|  | | style={{ display: "none" }} | |
|  | | ref={c => { | |
|  | | this.checkBtn = c; | |
|  | | }} | |
|  | | /> | |
|  | | </Form> | |
|  | | </div> | |
|  | | </div> | |
|  | | ); | |
|  | | } | |
|  | | } | |
| import React from 'react'; |
|  | import ReactDOM from 'react-dom'; | |
|  | import './index.css'; | |
|  | import App from './App'; | |
|  | import reportWebVitals from './reportWebVitals'; | |
|  | import { BrowserRouter } from "react-router-dom"; | |
|  |  | |
|  | ReactDOM.render( | |
|  | <BrowserRouter> | |
|  | <App /> | |
|  | </BrowserRouter>, | |
|  | document.getElementById('root') | |
|  | ); | |
|  |  | |
|  | // If you want to start measuring performance in your app, pass a function | |
|  | // to log results (for example: reportWebVitals(console.log)) | |
|  | // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals | |
|  | reportWebVitals(); | |

This is the main code to run all this project.

As you can see, creating a website is quite a difficult job. The quality of the entire resource, the ratio of the desired to the real, the impression depends on each stage and its unmistakable implementation. Planning, design, layout and programming of the site. These are the 4 key stages that make up the development of absolutely any website.