Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
Кафедра Програмної Інженерії

ЗВІТ

з дисципліни «Архітектура програмного забезпечення»

з лабораторної роботи №4

Виконав
ст. гр. ПЗПІ-20-7
Крупчак Євгеній

Перевірив:
Старш. викл. кафедри ПІ
Сокорчук І. П.

Харків 2022

ЛАБОРАТОРНА РОБОТА №3. РОЗРОБКА КЛІЄНТСЬКОЇ ЧАСТИНИ

1.1 Мета роботи

Розробити клієнтську частину програмної системи для проєкту за темою «Програмна система для автоматизації видачі боксів із їжею».

1.2 Хід роботи

Клієнтська частина додатку написана за допомогою React на мові программування JavaScript. У якості веб-серверу було використано Node.js. Використані бібліотеки:

1) MUI – Material Design для React;
2) Redux – для контролю стану;
3) Formik – для контролю стану форм;
4) I18Next – для локалізації;
5) Yup – для валідації форм;
6) DayJS – для локалізації дат та часу.

На рисунку 1 зображено діаграму прецедентів для сайту, на якій зображено відношення між акторами та прецедентами в системі.

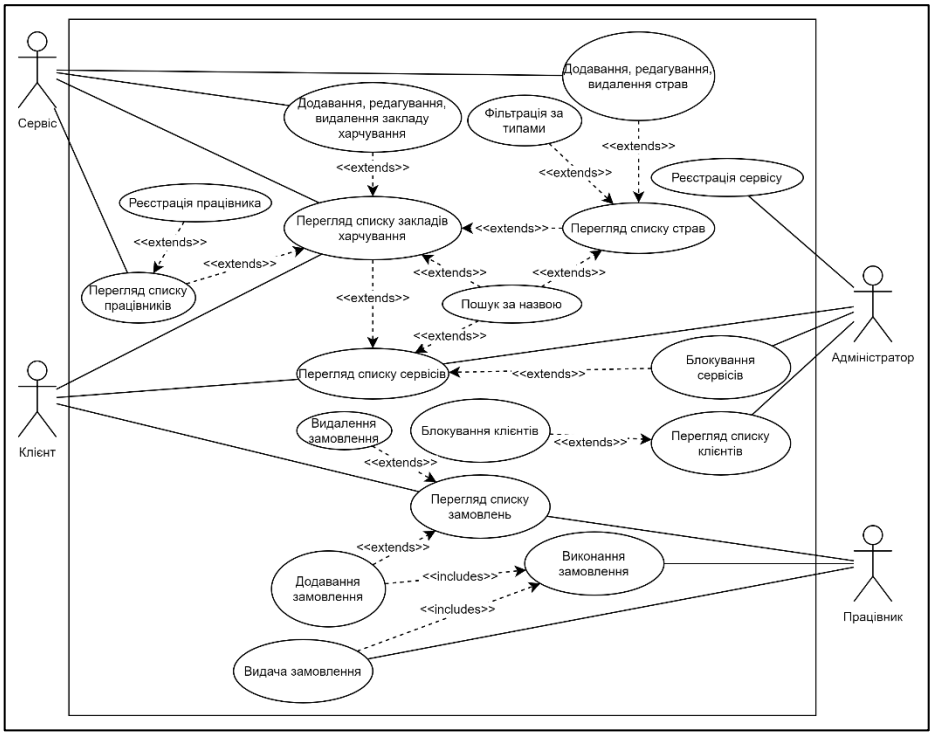

Рисунок 1 – Діаграма прецендентів клієнтської частини

На рисунку 2 зображено діаграму компонентів клієнтської частини.

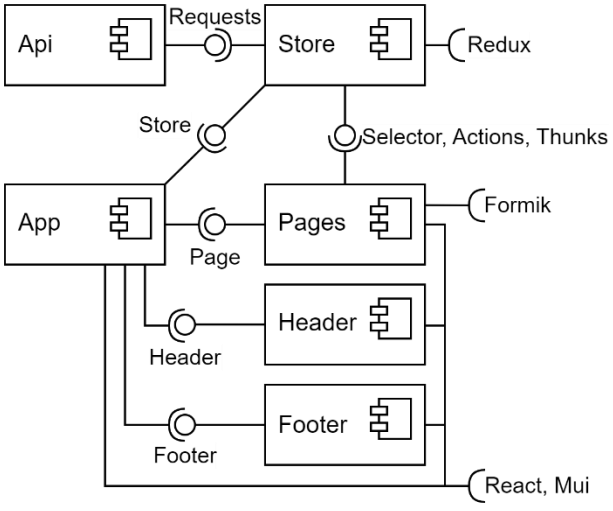

Рисунок 2 – Діаграма компонентів клієнтської частини

На рисунку 3 зображено діаграму діяльності процесу створення замовлення.

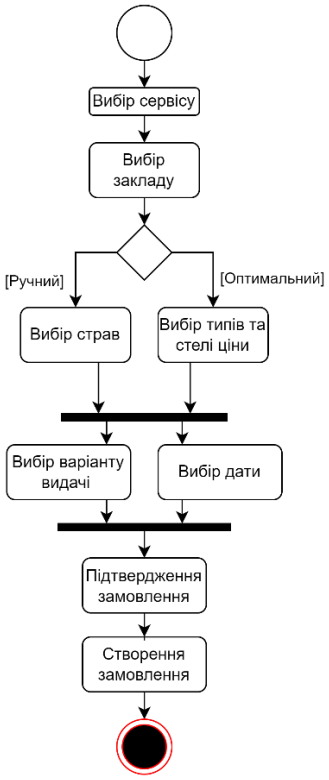

Рисунок 3 – Діаграма діяльності процесу створення замовлення

На рисунку 4 зображено діаграму пакетів клієнтської частини.

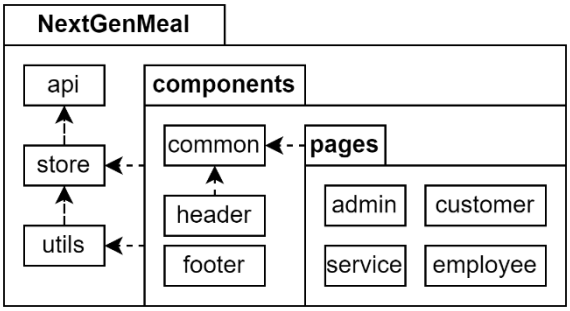

Рисунок 4 – Діаграма пакетів клієнтської частини

Пакет «» має у собі 4 основні складові:

1) Файли «index.js» та «App.js» – головні файли для відображення. Код файлу «» наведено у додатку А;

2) Файл «api.js», що має у собі обгортки для відправлення запитів на сервер. Його код наведено у додатку Б;
3) Папку «store», що має у собі файли, що представляють частини стану сайту. Приклад такої частини наведено у додатку В.
4) Папку «components», що має у собі компоненти UI. Приклад такого компоненту наведено у додатку Г.

1.3 Висновки

У данній лабораторній роботі було розроблено клієнтську частину програмної системи для проєкту за темою «Програмна система для автоматизації видачі боксів із їжею».

Посилання на папку "DEMO" із відео та кодом на Google Drive: https://drive.google.com/drive/folders/1sBPljA5AFl66BZejiDRfry6l0LPELsV7?usp=share_link

ДОДАТОК А

Програмний код файлу App.js

```
import './App.css';
import React, {useEffect} from "react";
import {Header} from "./components/header/Header";
import {useDispatch, useSelector} from "react-redux";
import {Route, Routes} from 'react-router-dom'
import {Box, CssBaseline, Stack} from "@mui/material";
import {LoginPage} from "./components/pages/LoginPage";
import {initialize, selector, setInitialized} from "./store/app";
import {RegisterPage} from "./components/pages/RegisterPage";
import {ConfirmPage} from "./components/pages/ConfirmPage";
import {Preloader} from "./components/common/Preloader";
import {CustomersPage} from "./components/pages/admin/CustomersPage";
import {CateringsPage} from "./components/pages/service/CateringsPage";
import {MenuPage} from "./components/pages/service/MenuPage"
import {NewOrderPage} from
"./components/pages/customer/orderPage/NewOrderPage";
import {MyOrdersPage} from "./components/pages/customer/MyOrdersPage";
import {EmployeesPage} from "./components/pages/service/EmployeesPage";
import {useUpdate} from "./utils/hook/hooks";
import {createTheme, ThemeProvider} from '@mui/material/styles';
import {OrderPage} from "./components/pages/employee/OrderPage";
import {OrdersPage} from "./components/pages/employee/OrdersPage";
import {Footer} from "./components/Footer";
import {getCustomers, getServices} from "./store/admin";
import mainImage from "./img/NextGenMealMain.png"
import {ServiceRegisterPage} from
"./components/pages/admin/ServiceRegisterPage";

const theme = createTheme({
    components: {
        MuiPaper: {
            defaultProps: {
                elevation: 3
            },
        },
    },
});
const TempMain = () => <Stack justifyContent="center" alignItems="center"
sx={{height: "100%", width: "100%"}}>
    <img src={mainImage} alt="NextGenMeal"/>
    <h1>Your ad could be here!</h1>
</Stack>;
const Temp404 = () => <div>404 NOT FOUND</div>
export const App = () => {
    const initialized = useSelector(selector("initialized"))
    const updated = useUpdate()
```

```
    const dispatch = useDispatch()
    useEffect(
        () => {
            dispatch(setInitialized(false));
            dispatch(initialize())
        },
        [updated]
    )
    return !initialized
        ? <Preloader/>
        : <ThemeProvider theme={theme}>
            <Stack spacing={5} sx={{minHeight: "100vh"}}>
                <CssBaseline/>
                <Header/>
                <Box component="main" sx={{height: "100%"}} flexGrow={1}>
                    <Routes>
                        <Route path='/' element={<TempMain/>} exact/>
                        <Route path='/confirm' element={<ConfirmPage/>}/>
                        <Route path='/login' element={<LoginPage/>}/>
                        <Route path='/register'
element={<RegisterPage/>}/>
                        <Route path="/register/confirm"
element={<div>Successfully registered. The confirmation link has been
sent on your email.</div>}/>
                        <Route path='/my_orders/new'
element={<NewOrderPage/>}/>
                        <Route path='/my_orders'
element={<MyOrdersPage/>}/>
                        <Route path='/orders/new'
element={<OrderPage/>}/>
                        <Route path='/orders' element={<OrdersPage/>}/>
                        <Route path='/service/caterings/:cateringId/menu'
element={<MenuPage/>}/>
                        <Route
path='/service/caterings/:cateringId/personnel'
element={<EmployeesPage/>}/>
                        <Route path='/service/caterings'
element={<CateringsPage/>}/>
                        <Route path='/admin/customers'
element={<CustomersPage getter={getCustomers}/>}/>
                        <Route path='/admin/services'
element={<CustomersPage getter={getServices}/>}/>
                        <Route path='/admin/services/register'
element={<ServiceRegisterPage/>}/>
                        <Route path='*' element={<Temp404/>}/>
                    </Routes>
                </Box>
                <Footer/>
            </Stack>
        </ThemeProvider>
}
```

ДОДАТОК Б

Програмний код файлу api.js

```
export const baseUrl = 'https://localhost:7168/api/'

export const sendRequest = async (url, method, {data = null, params =
null}) => {

    let input = `${baseUrl}${url}`;
    if (params)
        input += `?${new URLSearchParams(params)}`
    let init = {
        method,
        credentials: 'include',
        headers: {
            "Content-Type": "application/json",
        },
        mode: "cors"
    };
    if (data)
        init.body = JSON.stringify(data)
    try {
        return await fetch(input, init);
    } catch {
        return {
            ok: false,
            json: async () => ({
                errors: {
                    "Common": [
                        "Can't establish connection with server. Try
again later"
                    ]
                }
            })
        }
    }
}

export const get = async (url, params = null) => await sendRequest(url,
"GET", {params})
export const post = async (url, data, params = null) => await
sendRequest(url, "POST", {data, params})
export const put = async (url, data, params = null) => await
sendRequest(url, "PUT", {data, params})
export const patch = async (url, data, params = null) => await
sendRequest(url, "PATCH", {data, params})
export const del = async (url, params = null) => await sendRequest(url,
"DELETE", {params})
```

ДОДАТОК В

Програмний код файлу стану auth.js

```
import {createSlice} from "@reduxjs/toolkit";
import {commonGet, getSelector, handleResponse, setErrors, setUpdated}
from "./common";
import {del, get, patch, post} from "../api";

const {actions, name, reducer} = createSlice({
    name: 'auth',
    initialState: {
        confirmed: null,
        info: null,
        role: null,
        countries: []
    },
    reducers: {
        confirmEmailSuccess: (state, {payload}) => {
            state.confirmed = payload;
        },
        infoSuccess: (state, {payload}) => {
            state.info = payload;
        },
        logoutSuccess: (state) => {
            state.role = null;
            state.info = null;
        },
        roleSuccess: (state, {payload}) => {
            state.role = payload;
        },
        setCountries: (state, {payload}) => {
            state.countries = payload;
        },
    },
});
export default reducer
export const selector = getSelector(name)
export const {infoSuccess, roleSuccess, logoutSuccess,
confirmEmailSuccess, setCountries} = actions

export const getInfo = () => async dispatch => {
    const response = await get(`Account/Info`)
    if (response.ok) {
        const data = await response.json();
        dispatch(infoSuccess(data))
    }
}

export const getRole = () => async dispatch => {
```

```
47      const response = await get(`Account/Role`)
48      if (response.ok) {
49          const data = await response.text();
50          dispatch(roleSuccess(data))
51      }
52  }
53
54  export const signIn = ({email, password}) => async dispatch => {
55      const response = await post(`Account/Login`, {email, password})
56      await handleResponse(response, dispatch, setUpdated, setErrors);
57  }
58
59  export const signOut = () => async dispatch => {
60      const response = await del(`Account/Logout`)
61      if (response.ok) {
62          dispatch(logoutSuccess())
63          dispatch(setUpdated(false))
64      }
65  }
66
67  export const confirmEmail = ({id, code}) => async dispatch => {
68      const response = await get(`Account/ConfirmEmail`, {id, code})
69      dispatch(confirmEmailSuccess(response.ok))
70  }
71  export const register = ({name, email, password, confirmPassword}) =>
72      async dispatch => {
73          const data = {name, email, password, confirmPassword};
74          const response = await post(`Account/Register`, data,
75  {callbackUrl: `${window.location.origin}/register/confirm`})
76          await handleResponse(response, dispatch, setUpdated, setErrors)
77      }
78
79  export const rename = ({name}) => async dispatch => {
80      const response = await patch(`Account/ChangeName`, {name})
81      await handleResponse(response, dispatch, setUpdated, setErrors)
82  }
83  export const getCountries = () => commonGet('languages', null,
84  setCountries);
```

ДОДАТОК Г

Програмний код компоненту LoginPage.js

```
import React, {useCallback, memo} from 'react';
import {useFormik} from 'formik';
import {Navigate, NavLink} from "react-router-dom";
import {useDispatch, useSelector} from "react-redux";
import {emailValidation, passwordValidation} from
"../../utils/validation";
import * as yup from "yup";
import {selector, signIn} from "../../store/auth";
import {useTranslation} from "react-i18next";
import {Button, Container, Stack, SvgIcon, Typography} from
"@mui/material";
import {CustomTextField} from "../common/inputs/CustomTextField";
import {ReactComponent as GoogleIcon} from "../../img/google.svg"
import {baseUrl} from "../../api"
import {useReset} from "../../utils/hook/hooks";
import {ErrorsSnackbar} from "../common/ErrorsSnackbar";

export const LoginPage = memo(
    () => {
        const role = useSelector(selector("role"))
        const dispatch = useDispatch()
        const {t} = useTranslation()
        const initialValues = {email: '', password: ''};
        const validationSchema = yup.object({email: emailValidation(t),
password: passwordValidation(t)});
        const onSubmit = useCallback(
            values => {
                dispatch(signIn(values));
            },
            []
        )
        const formik = useFormik({initialValues, validationSchema,
onSubmit});
        useReset(open, formik.resetForm);
        return role
            ? <Navigate to={"/"}/>
            : <form onSubmit={formik.handleSubmit}>
                <Container maxWidth="sm">
                    <Stack spacing={4}>
                        <Typography variant="h3" align="center">
                            {t("Login")}
                        </Typography>
                        <ErrorsSnackbar/>
                        <CustomTextField name="email" formik={formik}
label={t("Email")}/>
```

```
                            <CustomTextField name="password" type="password"
formik={formik} label={t("Password")}/>
                            <Typography variant="body1" align="center">
                                <NavLink to="/forgot">
                                    {t("Forgot password?")}
                                </NavLink>
                            </Typography>
                            <Stack spacing={2}>
                                <Button variant="contained" type="submit">
                                    {t("Log in")}
                                </Button>
                                <Button variant="outlined"

href={`${baseUrl}Account/GoogleAuth?returnUrl=${window.location.href}`}
                                        startIcon={<SvgIcon
component={GoogleIcon} inheritViewBox/>}>
                                    {t("Continue with Google")}
                                </Button>
                            </Stack>
                            <Typography variant="body1" align="center">
                                {t("No account?")} 
                                <NavLink to="/register">
                                    {t("Create one")}
                                </NavLink>
                            </Typography>
                        </Stack>
                    </Container>
                </form>
        }
)

```