МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Факультет: Інформаційних технологій

Кафедра: Програмного забезпечення комп’ютерних систем

(повна назва кафедри)

ЗВІТ

з дисципліни: **практика навчальна комп’ютерна**

(назва дисципліни)

Студента (ки) \_\_2\_\_ курсу, групи 122-20-2\_

Спеціальності Комп’ютерні науки

Півня І.С.

(прізвище та ініціали)

Керівник \_асистент Харь А.Т.\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Харь А.Т.

(підпис) (прізвище та ініціали)

м. Дніпро – 2022 рік

**Зміст**

|  |  |  |
| --- | --- | --- |
|  | Завдання на практику……………………………………………... | 3 |
| 1. | Створення сховища даних………………………………………... | 4 |
| 1.1. | Концептуальна модель предметної області……………………... | 5 |
| 1.2. | ER-діаграма бази даних…………………………………………... | 6 |
| 2. | Розробка веб-орієнтованого інтерфейсу………………………… | 7 |
| 2.1 | UML-діаграми застосунку……………………………………….. | 8 |
| 2.2 | Програмні засоби та технології, використані у процесі розробки…………………………………………………………….. | 9 |
| 2.3 | Вимоги до апаратного та програмного забезпечення…………… | 10 |
| 2.4 | Опис інтерфейсу користувача та порядку роботи з програмою... | 11 |
| 3. | Висновки…………………………………………………………….. | 19 |
|  | Список літератури………………………………………………….. | 20 |
|  | Додаток А. Скрипт бази даних…………………………………….. | 22 |
|  | Додаток Б. Код програми………………………………………….. | 23 |

**Завдання на практику**

Створіть інформаційну систему з використанням бази даних для місцевого підприємства, яке займається постачанням газу для населення, виконує розрахунки з клієнтами відповідно з наступними тарифами:

Дані мають наступну структуру: ім’я та прізвище, домашня адреса (назва вулиці, номер дома), місто і район, початкові показники лічильнику газу, кінцеві показники, початкова дата, кінцева дата, номер лічильника.

1. **Створення сховище даних**

Під час виконання навчальної практики мовою серверу була обрана мова JavaScript на серверній платформі Node.js.

SQLite – обрана система керування реляційними базами даними – SQLite. Драйвером бази даних став модуль об'єктно-реляційного відображення Sequelize.

На сервері був створений файл gas\_db.db та підключений до скрипту Node.js за допомогою створення нового екземпляра Sequelize(). (Додаток А. Скрипт бази даних)

* 1. **Концептуальна модель предметної області**

Концептуальна модель полягає в тому, що адміністратор підприємства, що займається постачанням газу має доступ до бази даних клієнтів та їх газових лічильників у вигляді таблиць. При цьому враховується, що кожен клієнт газового підприємства може мати декілька осель та декілька газових лічильників у них. Згідно поставленого завдання, відома така інформація про клієнта:

* ім’я клієнта;
* прізвище клієнта;
* назва вулиці;
* номер будинку;
* місто;
* район.

Про лічильник відомо наступне:

* початкова дата;
* кінцева дата;
* показання на початку;
* показання наприкінці;
* номер лічильника.

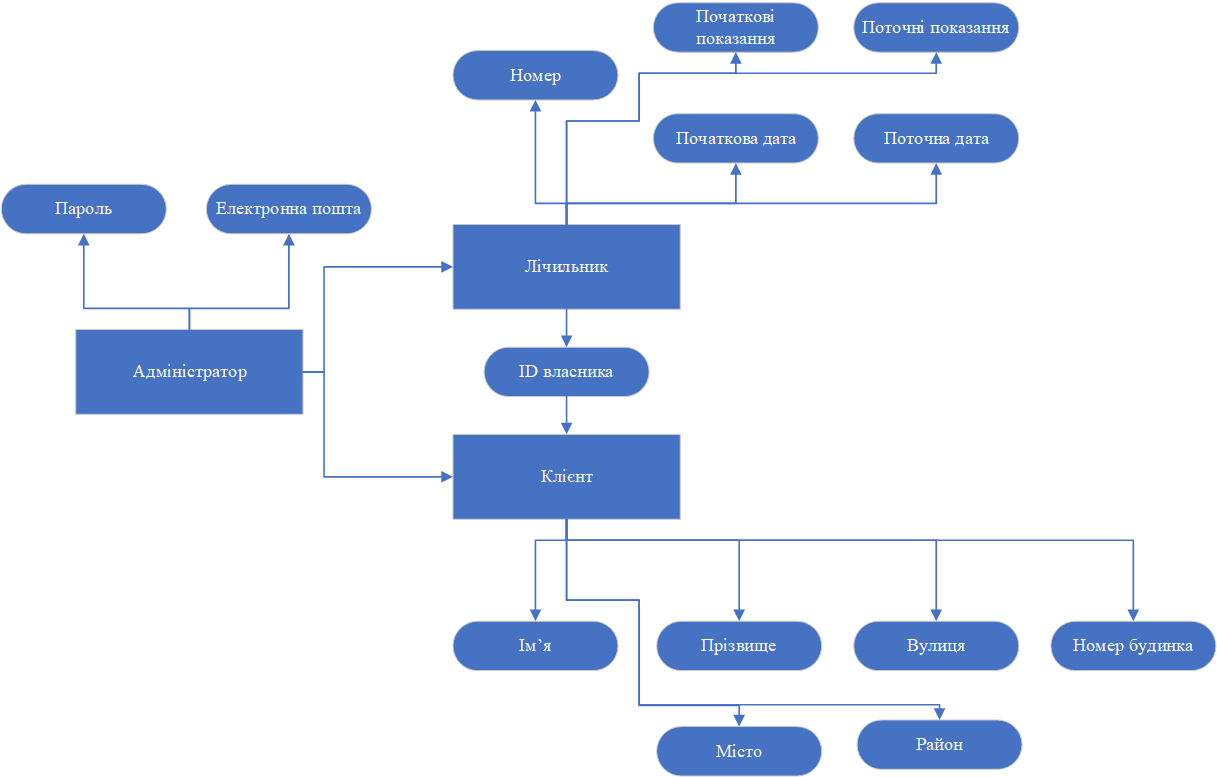


Рис. 1.1.1. Концептуальна модель

* 1. **ER-діаграма бази даних**

За завданням курсової роботи необхідно створити таблиці – регулярні структури, що складаються з кінцевих наборів полів по вертикалі й однотипних записах по горизонталі.

Отже, об’єкт КЛІЄНТ можна ідентифікувати за унікальним числовим ключем, який буде присвоєний кожному клієнту свій. Усі інші характеристики – залишаться атрибутами цього об’єкту: ім’я, прізвище, назва вулиці домашньої адреси, номер дому, місто, район проживання.

У об’єкті ЛІЧИЛЬНИК унікальним ключем може слугувати його номер, що зазначений на усіх лічильникам. За ним можна буде перебирати усі ці об’єкти таблиці. Інші характеристики лічильника будуть слугувати атрибутами: початкова дата, кінцева дата, дані на початку, дані наприкінці.

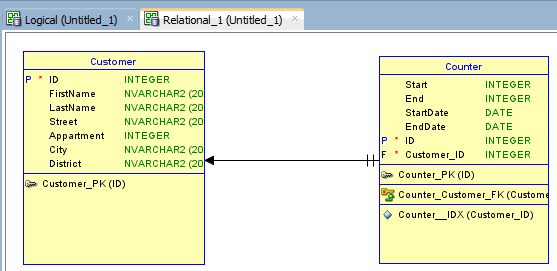


Рис. 1.2.1. ER-діаграма бази даних

1. **Розробка веб-орієнтованого інтерфейсу**

Для розробки застосунку був обраний підхід односторінкового застосунку (single-page application), який вміщується на одній сторінці, а потім оновлюється за допомогою JavaScript API за командами та бажаннями користувача. Для кожної взаємодії користувача з застосунком були створені компоненти (елементи сторінок), які рендеряться за викликом користувача за допомогою JavaScript.

Для зв’язку з сервером був обраний RESTful підхід, під час якого використовуються HTTP-запити для взаємодії з даними. Для цього було розроблено API: при взаємодії користувача з базою даних через веб-застосунок, застосунок посилає POST, GET, PATCH, DELTE HTTP-запити на адреси */customer*, */counter* з вказаним тілом запита або параметрами, в залежності від дії користувача. У відповідь сервер надсилає об’єкти JSON, які підхоплює веб-сервер та рендерить.

* 1. **UML-діаграми застосунку**

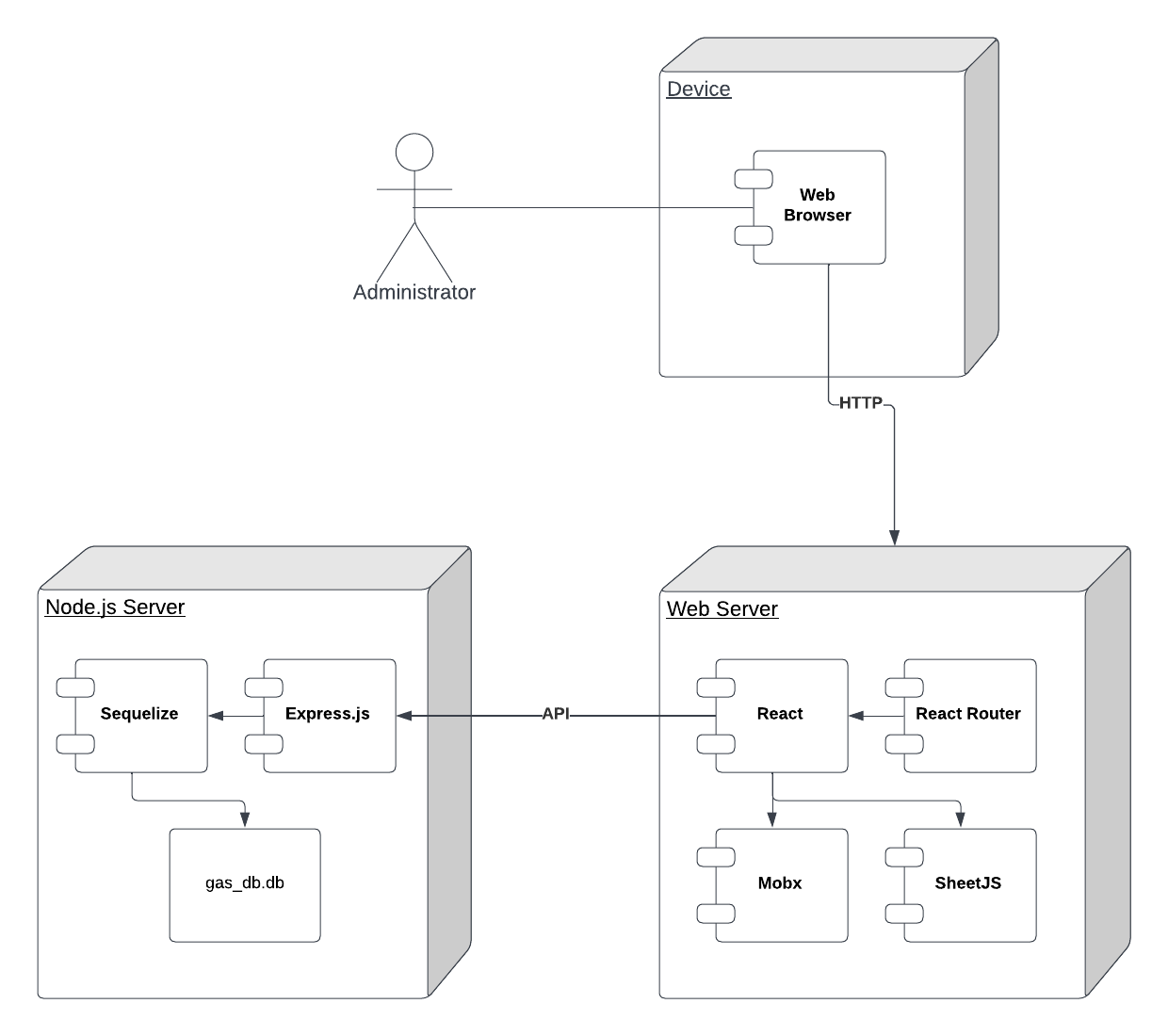


Рис. 2.1.1. UML-діаграма розгортання

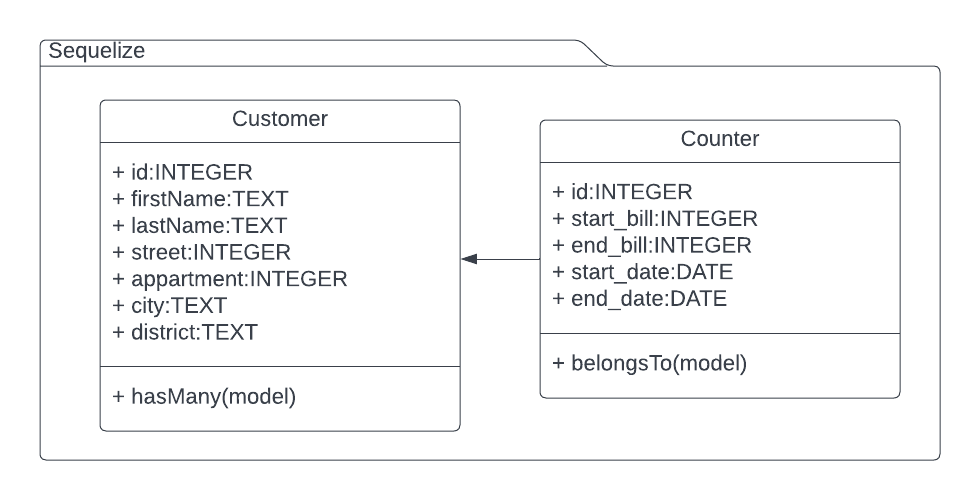


Рис. 2.1.2. UML-діаграма класів на серверній частині

* 1. **Програмні засоби та технології, використані у процесі розробки**

Для створення клієнт-серверного веб-застосунку був обраний такий стек технологій:

* Серверна платформа Node.js:
  + Express.js – веб-сервер для RESTful веб-API.
  + SQLite – система керування реляційними базами даних
  + Sequalize – модуль об'єктно-реляційного відображення для роботи з базами даних
* Frontend бібліотека React:
  + React Router – маршрутна бібліотека
  + Mobx – бібліотека-менеджер стану
  + SheetJS – пакет роботи з XLSX-файлами
  1. **Вимоги до апаратного та програмного забезпечення**

Мінімальні вимоги:

* Роздільна здатність екрану: 480p (854x480);
* Будь-який браузер з підтримкою HTML5, CSS3;
* Доступ до інтернету.

Рекомендовані вимоги:

* Роздільна здатність екрану: 1080p (1920x1080);
* Будь-який браузер з підтримкою HTML5, CSS3;
* Доступ до інтернету.
  1. **Опис інтерфейсу користувача та порядку роботи з програмою**

При заходженні на головну сторінку, адміністраторе перенаправляє до форми логіну у панель:

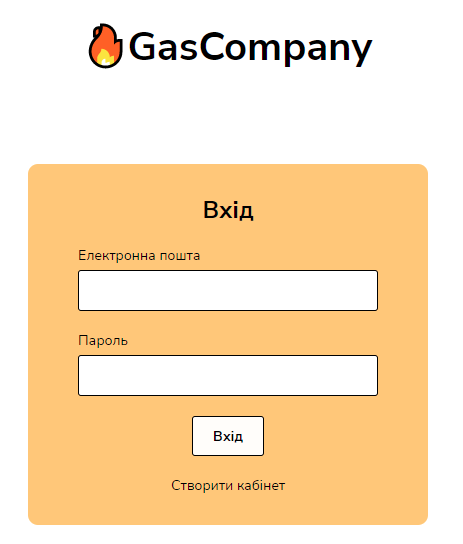


Рис. 2.4.1. Форма входу у панель

Після введені коректних даних користувача перенаправляє до кабінету. У верхній частині є змогу обрати дані для редагування – користувачів газової компанії, або їх лічильників.

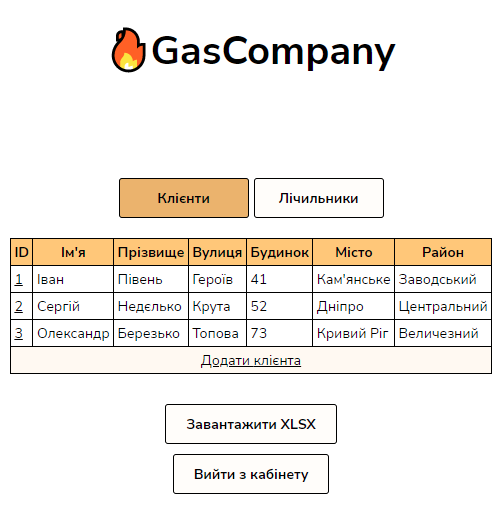


Рис. 2.4.2. Перелік клієнтів

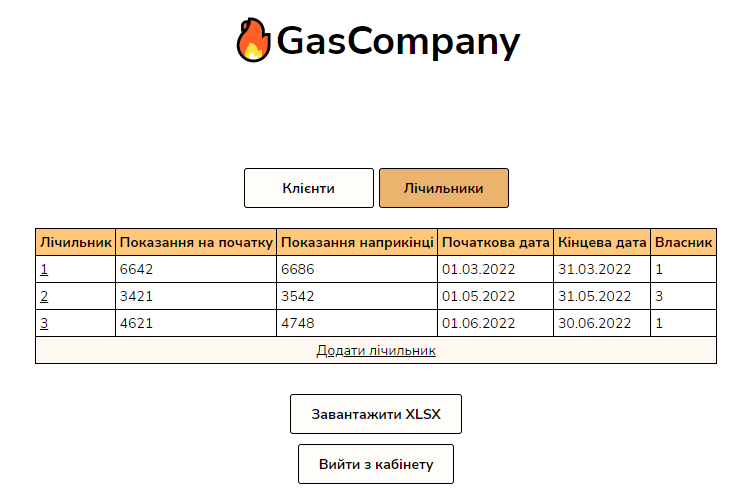


Рис. 2.4.3. Перелік лічильників

При натисканні на номер клієнта/лічильника, відкривається форма редагування з заповненими даними у полях про сутність. При натисканні на кнопку «Додати клієнта/лічильник» відкривається ця ж форма, але дані у полях – відсутні.

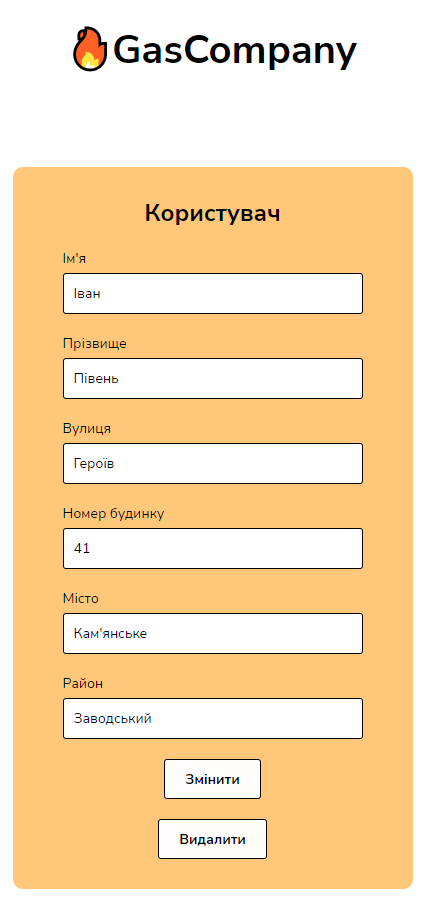


Рис. 2.4.4. Форма редагування клієнта

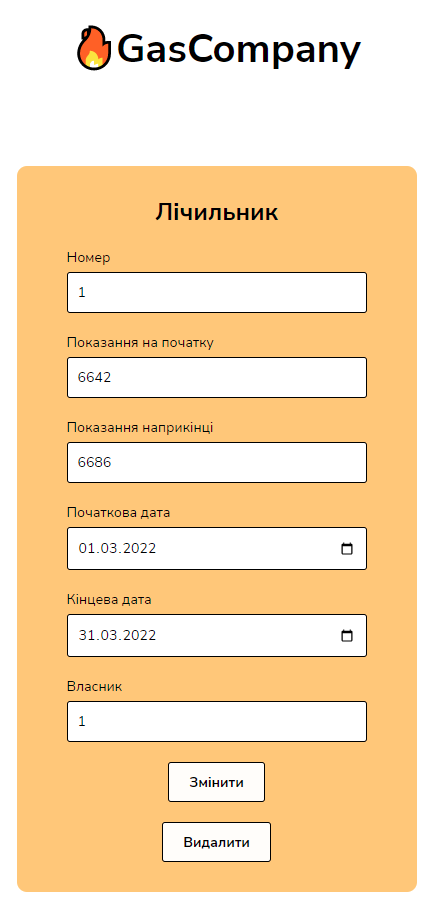


Рис. 2.4.5. Форма редагування лічильника

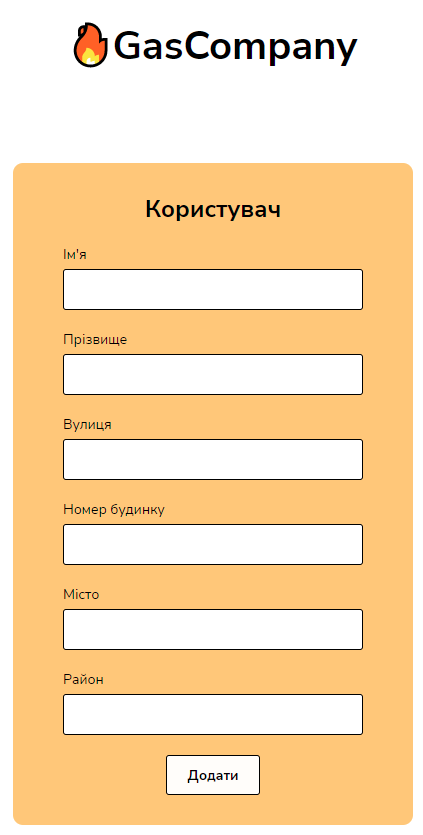


Рис. 2.4.6. Форма додання клієнта

Кожна форма редагування містить в кінці кнопку видалення запису. При натисненні на неї відображається вікно, у якому адміністратор повинен підтвердити свої наміри.

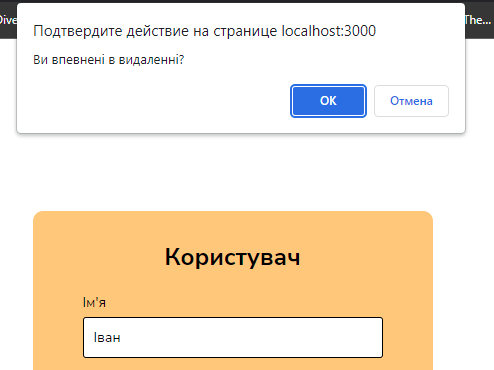


Рис. 2.4.7. Вікно попередження при видалення

У кожній формі виконується валідація даних у відповідні поля. Наприклад: заборонено вводити у текстові поля цифрові значення та навпаки, перевіряється актуальність введених дат та показань лічильників.

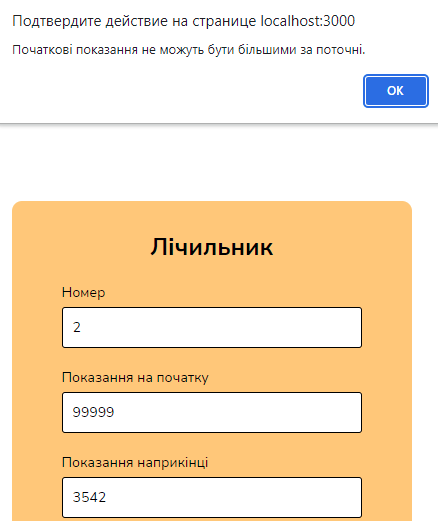


Рис. 2.4.8. Приклад валідації даних

При зберіганні будь-яких змін у формі адміністратора перенаправляє на сторінку кабінету. Під таблицею існує кнопка «Завантажити XLSX», що формує XLSX-таблицю та одразу завантажує її на пристрій користувача.

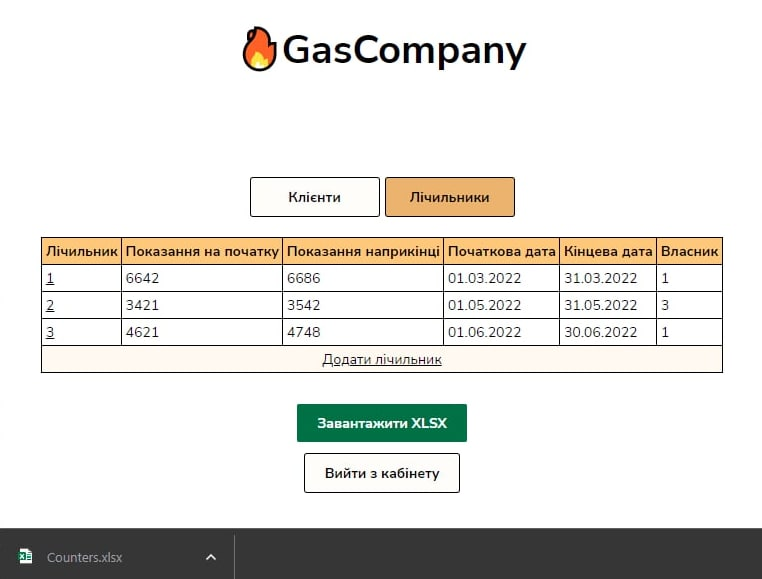


Рис. 2.4.9. Завантаження лічильників.

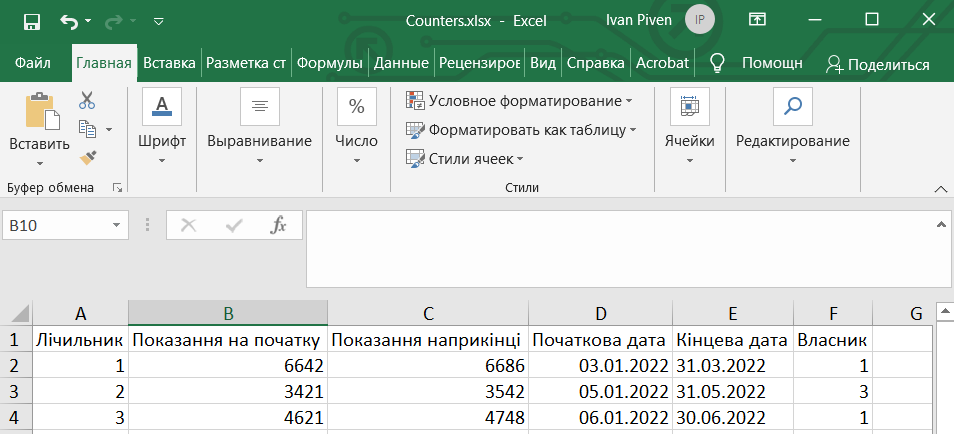


Рис. 2.4.10. Вміст файлу Counters.xlsx

Також, на сторінці присутня кнопка виходу з кабінету, після натискання якої, завершується сесія адміністратора та його перенаправляє на головну форму входу.

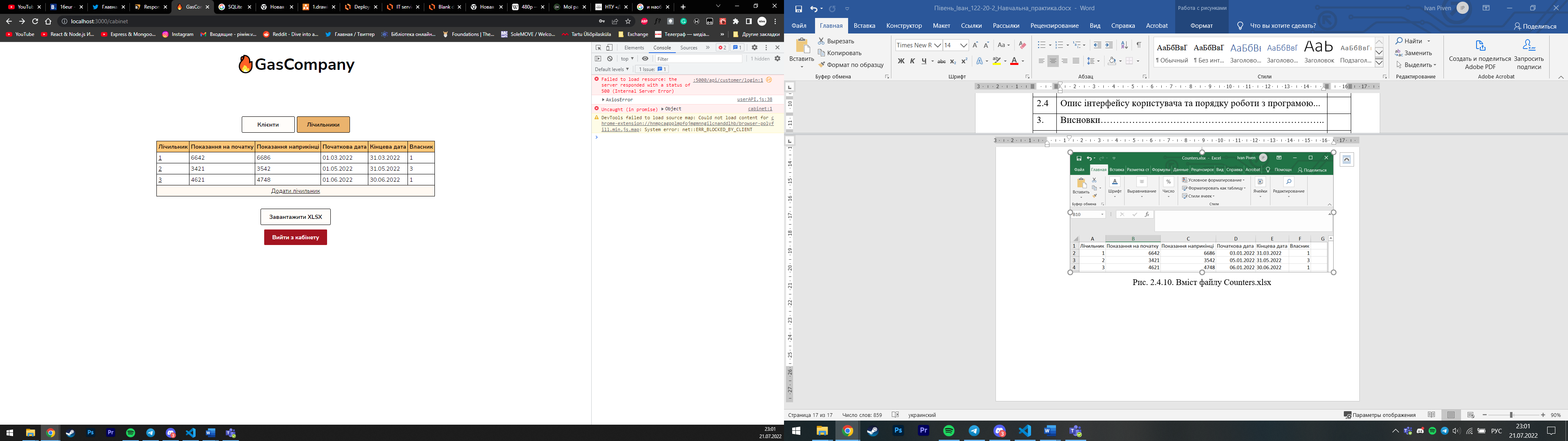


Рис. 2.4.11. Кнопка виходу з кабінету

**Висновки**

Під час виконання навчальної практики було вивчено роботу з системою керування реляційних баз даних SQLite 3, практично вивчено та закріплено навички створення RESTful API на базі бібліотеки Express.js з підключеною базою даних через Sequelize. Розроблено айдентику, дизайн та динамічний користувацький інтерфейс, побудовано односторінковий застосунок за допомогою бібліотеки React, налаштовано маршрути переходу по «сторінкам-компонентам» за допомогою React Router, отримано практичні навички по роботі з менеджером стану Mobx для керування даними на фронт-енді, ознайомлено з бібліотекою SheetJS для експорту XLSX-файлів.

**Список літератури**

1. Creating observable state – Mobx. URL: <https://mobx.js.org/observable-state.html> (дата звертання: 21.07.2022).
2. Eager Loading | Sequelize. URL: https://sequelize.org/docs/v7/advanced-association-concepts/eager-loading/ (дата звертання: 21.07.2022).
3. Error Handling. URL: <https://expressjs.com/en/guide/error-handling.html> (дата звертання: 21.07.2022).
4. Expess routing. URL: <https://expressjs.com/en/guide/routing.html> (дата звертання: 21.07.2022).
5. Getting Started | Sequelize. URL: <https://sequelize.org/docs/v7/getting-started/> (дата звертання: 21.07.2022).
6. Model Instances | Sequelize. URL: <https://sequelize.org/docs/v7/core-concepts/model-instances/> (дата звертання: 21.07.2022).
7. Model Querying – Basics | Sequelize. URL: https://sequelize.org/docs/v7/core-concepts/model-querying-basics/ (дата звертання: 21.07.2022).
8. Model Querying – Finders | Sequelize. URL: https://sequelize.org/docs/v7/core-concepts/model-querying-finders/ (дата звертання: 21.07.2022).
9. React Router | BrowserRouter. URL: <https://reactrouter.com/docs/en/v6/routers/browser-router> (дата звертання: 21.07.2022).
10. React Router | Link. URL: <https://reactrouter.com/docs/en/v6/components/link> (дата звертання: 21.07.2022).
11. React Router | Routes and Route. URL: https://reactrouter.com/docs/en/v6/components/route (дата звертання: 21.07.2022).
12. React Router | useNavigate. URL: https://reactrouter.com/docs/en/v6/hooks/use-navigate (дата звертання: 21.07.2022).
13. SheetJS CE | SheetJS Community Edition. URL: <https://docs.sheetjs.com/docs> (дата звертання: 21.07.2022).
14. Using middleware. URL: <https://expressjs.com/en/guide/using-middleware.html> (дата звертання: 21.07.2022).
15. Джей А. Крейбіх. Using SQLite. O'Reilly Media, Inc. 2010. – 241 с.
16. Компоненти і пропси – React. URL: <https://uk.reactjs.org/docs/components-and-props.html> (дата звертання: 21.07.2022).
17. Контекст – React. URL: <https://uk.reactjs.org/docs/context.html> (дата звертання: 21.07.2022).
18. Огляд хуків – React. URL: <https://uk.reactjs.org/docs/hooks-overview.html> (дата звертання: 21.07.2022).
19. Списки та ключі – React. URL: <https://uk.reactjs.org/docs/state-and-lifecycle.html> (дата звертання: 21.07.2022).
20. Стан та життєвий цикл – React. URL: <https://uk.reactjs.org/docs/state-and-lifecycle.html> (дата звертання: 21.07.2022).

**Додаток А. Скрипт бази даних**

const { Sequelize } = require('sequelize');

const { DataTypes } = require('sequelize');

const { model } = require('../db');

module.exports = new Sequelize({

dialect: 'sqlite',

storage: 'gas\_db.db'

});

const Customer = sequelize.define('user', {

id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true, },

firstName: { type: DataTypes.TEXT },

lastName: { type: DataTypes.TEXT },

street: { type: DataTypes.TEXT },

appartment: { type: DataTypes.INTEGER },

city: { type: DataTypes.TEXT },

district: { type: DataTypes.TEXT },

email: { type: DataTypes.TEXT, unique: true },

password: { type: DataTypes.TEXT },

role: { type: DataTypes.TEXT, defaultValue: 'USER' }

});

const Counter = sequelize.define('counter', {

id: { type: DataTypes.INTEGER, primaryKey: true },

start\_bill: { type: DataTypes.INTEGER, },

end\_bill: { type: DataTypes.INTEGER, },

start\_date: { type: DataTypes.DATE, },

end\_date: { type: DataTypes.DATE, },

})

Customer.hasMany(Counter);

Counter.belongsTo(Customer);

module.exports = {

Customer,

Counter

**Додаток Б. Код програми**

Контролер лічильника:

const { Counter } = require('../models/models')

const ApiError = require('../error/ApiError')

class CounterController {

async create(req, res, next) {

try {

const {

id,

start\_bill,

end\_bill,

start\_date,

end\_date,

userId } = req.body;

const counter = await Counter.create({

id,

start\_bill,

end\_bill,

start\_date,

end\_date,

userId

});

return res.json(counter);

} catch (error) {

next(ApiError.badRequest(error.message));

}

}

async getAll(req, res) {

const counters = await Counter.findAll();

return res.json(counters);

}

async getById(req, res) {

const { id } = req.params;

const counter = await Counter.findOne({ where: { id } });

return res.json(counter);

}

async updateCounter(req, res) {

const { id } = req.params;

const {

start\_bill,

end\_bill,

start\_date,

end\_date,

userId,

} = req.body;

try {

const counter = await Counter.findByPk(id);

Object.assign(counter, {

start\_bill,

end\_bill,

start\_date,

end\_date,

userId,

})

counter.save();

return res.json(await Counter.findByPk(id));

} catch (error) {

return (error);

}

}

async deleteCounter(req, res) {

const { id } = req.params;

try {

const counter = await Counter.findByPk(id);

await counter.destroy();

return res.json('Deleted');

} catch (error) {

return res.json(error);

}

}

}

module.exports = new CounterController();

Контролер клієнта:

const { Customer } = require('../models/models');

const ApiError = require('../error/ApiError');

const bcrypt = require('bcrypt');

const JWT = require('jsonwebtoken');

const { model } = require('../db');

const generateJwt = (id, email) => {

return JWT.sign(

{ id, email },

process.env.SECRET\_KEY,

{ expiresIn: '24h' });

}

class CustomerController {

async registration(req, res, next) {

const {

email,

password,

firstName,

lastName,

street,

appartment,

city,

district

} = req.body;

if (!email || !password) {

return next(ApiError.badRequest('Empty credentials not allowed.'))

}

const candidate = await Customer.findOne({ where: { email } });

if (candidate) {

return next(ApiError.badRequest('Customer already registered.'))

}

const hashPassword = await bcrypt.hash(password, 5);

const customer = await Customer.create({

email,

password: hashPassword,

firstName,

lastName,

street,

appartment,

city,

district

});

const token = generateJwt(customer.id, email);

return res.json({ token });

}

async login(req, res, next) {

const { email, password } = req.body;

const customer = await Customer.findOne({ where: { email } })

if (!customer) {

return next(ApiError.internal('Customer not found.'))

}

// let comparePassword = bcrypt.compareSync(password, Customer.password, (error, result) => {

// if (error) {

// return next(ApiError.badRequest('debix.'))

// }

// if (result) {

// const token = generateJwt(customer.id, customer.email);

// return res.json({ token });

// } else {

// // return res.json('bad');

// }

const validPass = await bcrypt.compare(password, customer.password);

if (validPass) {

const token = generateJwt(customer.id, customer.email);

return res.json({ token });

} else {

return next(ApiError.badRequest('Wrong password.'))

}

};

async check(req, res, next) {

const { email } = req.body;

const customerId = await Customer.findOne({ where: { email } });

const token = generateJwt(customerId, email);

return res.json({ token });

}

async getAllCustomers(req, res) {

const customers = await Customer.findAll();

return res.json(customers);

}

async getCustomer(req, res) {

const { id } = req.params;

const customer = await Customer.findByPk(id);

return res.json(customer);

}

async createCustomer(req, res) {

const {

firstName,

lastName,

street,

appartment,

city,

district

} = req.body;

const customer = await Customer.create({

firstName,

lastName,

street,

appartment,

city,

district

});

return res.json(customer);

}

async updateCustomer(req, res) {

const { id } = req.params;

const {

firstName,

lastName,

street,

appartment,

city,

district

} = req.body;

try {

const customer = await Customer.findByPk(id);

Object.assign(customer, {

firstName,

lastName,

street,

appartment,

city,

district

})

customer.save();

return res.json(await Customer.findByPk(id));

} catch (error) {

return (error);

}

}

async deleteCustomer(req, res) {

const { id } = req.params;

try {

const customer = await Customer.findByPk(id);

await customer.destroy();

return res.json('Deleted');

} catch (error) {

return res.json(error);

}

}

}

module.exports = new CustomerController();

Форма логіну:

import React from 'react';

import { useState, useContext } from 'react';

import { Link, useNavigate } from 'react-router-dom';

import './AuthForm.css';import { observer } from 'mobx-react-lite';

import { Context } from '../index';

export const AuthForm = observer((props) => {

const { user } = useContext(Context);

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

let navigate = useNavigate();

async function registerCabinet(email, pass) {

await axios.post('http://localhost:5000/api/customer/registration', {

email: email,

password: pass

})

.then((response) => {

if (response.data.token) {

localStorage.setItem('token', response.data.token);

// console.log(localStorage.getItem('token'));

const decodedJWT = jwt\_decode(response.data.token);

// console.log(decodedJWT);

return decodedJWT;

}

})

.catch((response) => {

console.log(response);

})

}

const btnClick = async (event) => {

try {

event.preventDefault();

props.wantSignIn

? await login(email, password)

: await registerCabinet(email, password);

user.setUser(user);

user.setIsAuth(true);

navigate('/cabinet');

} catch (error) {

alert(error);

}

};

async function login(email, pass) {

await axios.post('http://localhost:5000/api/customer/login', {

email: email,

password: pass

})

.then((response) => {

if (response.data.token) {

localStorage.setItem('token', response.data.token);

const decodedJWT = jwt\_decode(response.data.token);

return decodedJWT;

}

})

.catch((response) => {

console.log(response);

})

}

return (

<form className='form'>

<h3 className='register-form-title'>

{props.wantSignIn ? 'Вхід' : 'Реєстрація'}

</h3>

{props.wantSignIn ? (

''

) : (

<>

<div className='form-row'>

<label htmlFor='phone'>Ім'я</label>

<input id='name' type='text'></input>

</div>

<div className='form-row'>

<label htmlFor='phone'>Прізвище</label>

<input id='phone' type='text'></input>

</div>

<div className='form-row'>

<label htmlFor='street'>Вулиця</label>

<input id='street' type='text'></input>

</div>

<div className='form-row'>

<label htmlFor='appartment'>Номер будинку</label>

<input id='appartment' type='number'></input>

</div>

<div className='form-row'>

<label htmlFor='city'>Місто</label>

<input id='city' type='text'></input>

</div>

<div className='form-row'>

<label htmlFor='district'>Район</label>

<input id='district' type='text'></input>

</div>

</>

)}

<div className='form-row'>

<label htmlFor='email'>Електронна пошта</label>

<input

id='email'

type='email'

value={email}

onChange={(e) => setEmail(e.target.value)}

></input>

</div>

<div className='form-row'>

<label htmlFor='pass'>Пароль</label>

<input

id='pass'

type='password'

value={password}

onChange={(e) => setPassword(e.target.value)}

></input>

</div>

{props.wantSignIn ? (

''

) : (

<>

<div className='form-row'>

<label htmlFor='pass'>Підтвердження паролю</label>

<input id='check-pass' type='password'></input>

</div>

</>

)}

<button className='form-btn' type='submit' onClick={btnClick}>

{/\* <span className='form-btn-span'> \*/}

{props.wantSignIn ? 'Вхід' : 'Зареєструватись'}

{/\* </span> \*/}

</button>

<Link

className='form-proposal'

to={props.wantSignIn ? '/sign-up' : '/sign-in'}

>

{props.wantSignIn ? 'Створити кабінет' : 'Увійти в кабінет'}

</Link>

</form>

);

});

Сторінка кабінету:

import React from 'react';

import { Logo } from '../components/Logo';

import { Container } from '../components/Container';

import { LogOut } from '../components/LogOut';

import './cabinet.css';

import { Link } from 'react-router-dom';

import { useEffect } from 'react';

import axios from 'axios';

import { useState } from 'react';

const XLSX = require('xlsx');

export const Cabinet = () => {

const [counters, setCounters] = useState([]);

const [seeCustomers, setSeeCustomers] = useState(false);

useEffect(() => {

let url = seeCustomers

? 'http://localhost:5000/api/customer/'

: 'http://localhost:5000/api/counter/';

seeCustomers

? axios

.get('http://localhost:5000/api/customer/')

.then((res) => setCounters(res.data))

: axios

.get('http://localhost:5000/api/counter/')

.then((res) => setCounters(res.data));

}, [seeCustomers]);

const btnSwitch = async (event) => {

try {

event.preventDefault();

setSeeCustomers(!seeCustomers);

} catch (error) {

alert(error);

}

};

const getFormattedDate = (dateStr) => {

const date = new Date(dateStr);

return date.toLocaleDateString();

};

if (counters === undefined) {

return <>Wait</>;

}

return (

<Container>

<Logo></Logo>

<div className='switch'>

<button disabled={seeCustomers} onClick={btnSwitch}>

Клієнти

</button>

<button disabled={!seeCustomers} onClick={btnSwitch}>

Лічильники

</button>

</div>

<table id='maintable'>

<thead>

{seeCustomers ? (

<tr>

<th scope='col'>ID</th>

<th scope='col'>Ім'я</th>

<th scope='col'>Прізвище</th>

<th scope='col'>Вулиця</th>

<th scope='col'>Будинок</th>

<th scope='col'>Місто</th>

<th scope='col'>Район</th>

</tr>

) : (

<tr>

<th scope='col'>Лічильник</th>

<th scope='col'>Показання на початку</th>

<th scope='col'>Показання наприкінці</th>

<th scope='col'>Початкова дата</th>

<th scope='col'>Кінцева дата</th>

<th scope='col'>Власник</th>

</tr>

)}

</thead>

<tbody>

{counters.map((cell) =>

seeCustomers ? (

<tr key={cell.id}>

<td>

<Link to='/customer' state={{ id: cell.id }}>

{cell.id}

</Link>

</td>

<td>{cell.firstName}</td>

<td>{cell.lastName}</td>

<td>{cell.street}</td>

<td>{cell.appartment}</td>

<td>{cell.city}</td>

<td>{cell.district}</td>

</tr>

) : (

<tr key={cell.id}>

<td>

<Link to='/counter' state={{ id: cell.id }}>

{cell.id}

</Link>

</td>

<td>{cell.start\_bill}</td>

<td>{cell.end\_bill}</td>

<td>{getFormattedDate(cell.start\_date)}</td>

<td>{getFormattedDate(cell.end\_date)}</td>

<td>{cell.userId}</td>

</tr>

)

)}

</tbody>

<tfoot>

<tr>

<td colSpan='100%'>

<Link

to={seeCustomers ? '/customer' : '/counter'}

state={{ add: true }}

>

Додати {seeCustomers ? 'клієнта' : 'лічильник'}

</Link>

</td>

</tr>

</tfoot>

</table>

<div className='buttons'>

<button

id='sheetjsexport'

onClick={() => {

var wb = XLSX.utils.table\_to\_book(

document.getElementById('maintable')

);

XLSX.writeFile(

wb,

`${seeCustomers ? 'Customers' : 'Counters'}.xlsx`

);

}}

>

Завантажити XLSX

</button>

</div>

<LogOut />

</Container>

);

};

Форма додавання/редагування клієнта:

import React from 'react';

import { useState } from 'react';

import { useNavigate, useLocation } from 'react-router-dom';

import './CustomerForm.css';

import { useEffect } from 'react';

import axios from 'axios';

export const CustomerForm = (props) => {

const [customer, setCustomer] = useState({

firstName: '',

lastName: '',

street: '',

appartment: '',

city: '',

district: '',

});

let navigate = useNavigate();

const location = useLocation();

const { id } = location?.state;

const btnClick = async (event) => {

try {

event.preventDefault();

id

? axios.patch(`http://localhost:5000/api/customer/${id}`, customer)

: axios.post('http://localhost:5000/api/customer/', customer);

navigate('/cabinet');

} catch (error) {

alert(error);

}

};

const btnDelete = async (event) => {

try {

event.preventDefault();

if (window.confirm('Ви впевнені в видаленні?')) {

axios

.delete(`http://localhost:5000/api/customer/${id}`)

.then(navigate('/cabinet'));

}

} catch (error) {

alert(error);

}

};

useEffect(() => {

if (id) {

axios

.get(`http://localhost:5000/api/customer/${id}`)

.then((self) => setCustomer(self.data));

}

}, [id]);

if (customer === undefined) {

return <>Wait</>;

}

return (

<form className='form'>

<h3 className='register-form-title'>Користувач</h3>

{props.wantSignIn ? (

''

) : (

<>

<div className='form-row'>

<label htmlFor='firstName'>Ім'я</label>

<input

id='firstName'

type='text'

defaultValue={customer.firstName}

onChange={(e) =>

setCustomer({ ...customer, firstName: e.target.value })

}

></input>

</div>

<div className='form-row'>

<label htmlFor='lastName'>Прізвище</label>

<input

id='lastName'

type='text'

defaultValue={customer.lastName}

onChange={(e) =>

setCustomer({ ...customer, lastName: e.target.value })

}

></input>

</div>

<div className='form-row'>

<label htmlFor='street'>Вулиця</label>

<input

id='street'

type='text'

defaultValue={customer.street}

onChange={(e) =>

setCustomer({ ...customer, street: e.target.value })

}

></input>

</div>

<div className='form-row'>

<label htmlFor='appartment'>Номер будинку</label>

<input

id='appartment'

type='number'

defaultValue={customer.appartment}

onChange={(e) =>

setCustomer({ ...customer, appartment: e.target.value })

}

></input>

</div>

<div className='form-row'>

<label htmlFor='city'>Місто</label>

<input

id='city'

type='text'

defaultValue={customer.city}

onChange={(e) =>

setCustomer({ ...customer, city: e.target.value })

}

></input>

</div>

<div className='form-row'>

<label htmlFor='district'>Район</label>

<input

id='district'

type='text'

defaultValue={customer.district}

onChange={(e) =>

setCustomer({ ...customer, district: e.target.value })

}

></input>

</div>

</>

)}

<button className='form-btn' type='submit' onClick={btnClick}>

{id ? 'Змінити' : 'Додати'}

</button>

{id ? (

<button className='form-btn' onClick={btnDelete}>

Видалити

</button>

) : (

<></>

)}

</form>

);

};

Форма додавання/редагування лічильника:

import React from 'react';

import { useState } from 'react';

import { useNavigate, useLocation } from 'react-router-dom';

import './CustomerForm.css';

import { useEffect } from 'react';

import axios from 'axios';

export const CounterForm = (props) => {

const [counter, setCounter] = useState({

id: '',

start\_bill: '',

end\_bill: '',

start\_date: '',

end\_date: '',

userId: '',

});

let navigate = useNavigate();

const location = useLocation();

const { id } = location?.state;

const btnClick = async (event) => {

try {

event.preventDefault();

id

? axios.patch(`http://localhost:5000/api/counter/${id}`, counter)

: axios.post('http://localhost:5000/api/counter/', counter);

navigate('/cabinet');

} catch (error) {

alert(error);

}

};

const btnDelete = async (event) => {

try {

event.preventDefault();

if (window.confirm('Ви впевнені в видаленні?')) {

axios

.delete(`http://localhost:5000/api/counter/${id}`)

.then(navigate('/cabinet'));

}

} catch (error) {

alert(error);

}

};

useEffect(() => {

if (id) {

axios

.get(`http://localhost:5000/api/counter/${id}`)

.then((self) => setCounter(self.data))

.then(

setCounter({

...counter,

endta: counter.start\_date.split('T')[0],

end\_date: counter.end\_date.split('T')[0],

})

);

}

}, [id]);

if (counter === undefined) {

return <>Wait</>;

}

return (

<form className='form'>

<h3 className='register-form-title'>Лічильник</h3>

{props.wantSignIn ? (

''

) : (

<>

<div className='form-row'>

<label htmlFor='id'>Номер</label>

<input

id='id'

type='number'

defaultValue={counter.id}

onChange={(e) => setCounter({ ...counter, id: e.target.value })}

></input>

</div>

<div className='form-row'>

<label htmlFor='start\_bill'>Показання на початку</label>

<input

id='start\_bill'

type='number'

defaultValue={counter.start\_bill}

onChange={(e) =>

setCounter({ ...counter, start\_bill: e.target.value })

}

></input>

</div>

<div className='form-row'>

<label htmlFor='end\_bill'>Показання наприкінці</label>

<input

id='end\_bill'

type='number'

defaultValue={counter.end\_bill}

onChange={(e) =>

setCounter({ ...counter, end\_bill: e.target.value })

}

></input>

</div>

<div className='form-row'>

<label htmlFor='start\_date'>Початкова дата</label>

<input

id='start\_date'

type='date'

defaultValue={counter.start\_date.split('T')[0]}

onChange={(e) =>

setCounter({ ...counter, start\_date: e.target.value })

}

></input>

</div>

<div className='form-row'>

<label htmlFor='end\_date'>Кінцева дата</label>

<input

id='end\_date'

type='date'

defaultValue={counter.end\_date.split('T')[0]}

onChange={(e) =>

setCounter({ ...counter, end\_date: e.target.value })

}

></input>

</div>

<div className='form-row'>

<label htmlFor='userId'>Власник</label>

<input

id='userId'

type='number'

defaultValue={counter.userId}

onChange={(e) =>

setCounter({ ...counter, userId: e.target.value })

}

></input>

</div>

</>

)}

<button className='form-btn' type='submit' onClick={btnClick}>

{id ? 'Змінити' : 'Додати'}

</button>

{id ? (

<button className='form-btn' onClick={btnDelete}>

Видалити

</button>

) : (

<></>

)}

</form>

);

};