МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ

УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Звіт

з лабораторної роботи №3

з дисципліни

«Аналіз та рефакторинг коду»

|  |  |
| --- | --- |
|  | Перевірила:  доц. каф. Програмної Інженерії  Побіженко І. О.  Виконав:  ст. гр. ПЗПІ-19-6  Бахмат Є. Ю. |

Харків 2021

**ВСТУП**

Темою даної лабораторної роботи є розробка клієнтської/front-end частини проекту за темою: «Програмна система універсального клімат-контролю».

Метою роботи є отримання навичок з проектування клієнтської частини програмної системи з використанням Use-Case діаграми, діаграми компонентів, діаграми станів та діаграми діяльності. У ході роботи повинна бути розроблена робоча клієнтська частина програмної системи та відповідна супроводжувальна документація.

Хід даної лабораторної роботи має такий вигляд:

1. Розробити Use Case діаграму з зображеними на ній акторами та їх взаємодією із системою;
2. Розробити UML діаграму компонентів;
3. Описати методи взаємодії програмних компонентів між собою та їх зв’язок із серверною частиною;
4. Розробити UML діаграму станів;
5. Розробити UML діаграму діяльності;
6. Зробити висновки по роботі.

**1 USE CASE**

Користувачами системи є 3 типи акторів: адміністратори системи, власники та співробітники. Власник та адміністратор взаємодіють із системою через веб-застосунок, а співробітники - за допомогою мобільного застосунку. Розумні пристрої контролюються як веб-застосунком так і мобільним застосунком. Виходячи з цього була створена UML діаграма прецедентів (див. додаток А. рис. А.1).

За допомогою  веб-застосунку адміністратори системи можуть отримати інформацію про стан усіх елементів системи, схеми систем контролю клімату підконтрольних будівель та їх складові частини, такі, як кімнати, пристрої, сенсори та їх дані, а також створювати резервні копії стану програмної платформи. Власник може використовувати веб-застосунок для створення, редагування та видалення організаційної інформації, наприклад: інформацію про будівлі, кімнати або співробітників, змінювати права доступу персоналу, реєструвати нових співробітників, тощо.

Крім того, власнику доступні перегляд статистики, усі CRUD операції системи, у рамках свого бізнесу, окрім зміни персональної інформації користувачів.

Усі операції потребують авторизації користувача. Усі користувачі мають можливість відновити пароль за допомогою електронної пошти. Авторизація через веб додаток доступна лише для робітників та адміністраторів, а для клієнтів – лише через мобільний додаток.

**2 СТРУКТУРА ПРОЕКТУ**

Для розробки клієнтської частини використовувалась мова TypeScript та фреймворк ReactJS. Для реалізації Flux архітектури, Детальніша структура проекту зображена у додатку Б (див. рис. Б.1).

Уся робота додатку розпочинається з файлу index.tsx, у якому написано головні інструкції React для правильної роботи програми.

ReactDOM.render(

  <Provider store={store}>

    <React.StrictMode>

      <BrowserRouter>

        <App />

        <ToastContainer

          position='top-center'

          autoClose={5000}

          hideProgressBar={false}

          newestOnTop={false}

          closeOnClick

          rtl={false}

          pauseOnFocusLoss

          draggable

          pauseOnHover

        />

      </BrowserRouter>

    </React.StrictMode>

  </Provider>,

  document.getElementById('root')

)

Файл App.tsx виконує роль роутера для переходу між сторінками розробленого застосунку. Код файлу наведено у додатку В

Для правильної роботи систему було запроваджено спеціальний сервіс (шар роботи з API) core.ts . Усі сервіси використовують core.ts для створення запитів до API. Таке рішення було прийнято для реалізації патерну проектування Singleton.

import axios from 'axios'

export default axios.create({

  baseURL: "http://192.168.0.106:3099/",

  headers: {

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

    'Access-Control-Allow-Origin': "\*"

  },

  timeout: 2000

})

Для роботи застосунку було створено динамічні компоненти, які дозволяють створювати форми (Form.tsx) та таблиці (Table.tsx) для основних CRUD операцій.Загальний вигляд компоненту Table наведено у додатку Г Загальний вигляд компоненту Form наведено нижче:

const Form = (props: Props) => {

  const reducer = (state: any, action: any) => {

    const { key, value } = action

    return { ...state, [key]: value }

  }

  const handleSubmit = (event: any) => {

    event.preventDefault()

    props.onSubmit(dataState)

  }

  const [dataState, dispatch] = useReducer(reducer, props.data)

  return (

    <form className={styles['form']} onSubmit={handleSubmit}>

      <h1>{props.title}</h1>

      {Object.entries(dataState).map((item: Array<any>) =>

        convertDataToJSX(item, props.modal, dispatch)

      )}

      <button className={styles['button-submit']} type='submit'>

        Submit

      </button>

    </form>

  )

}

Завдяки компоненту Table було створено наступні таблиці відображення інформації про доступ (див рис. 2.1), будівель (див рис. 2.2), запитів (див рис. 2.3).

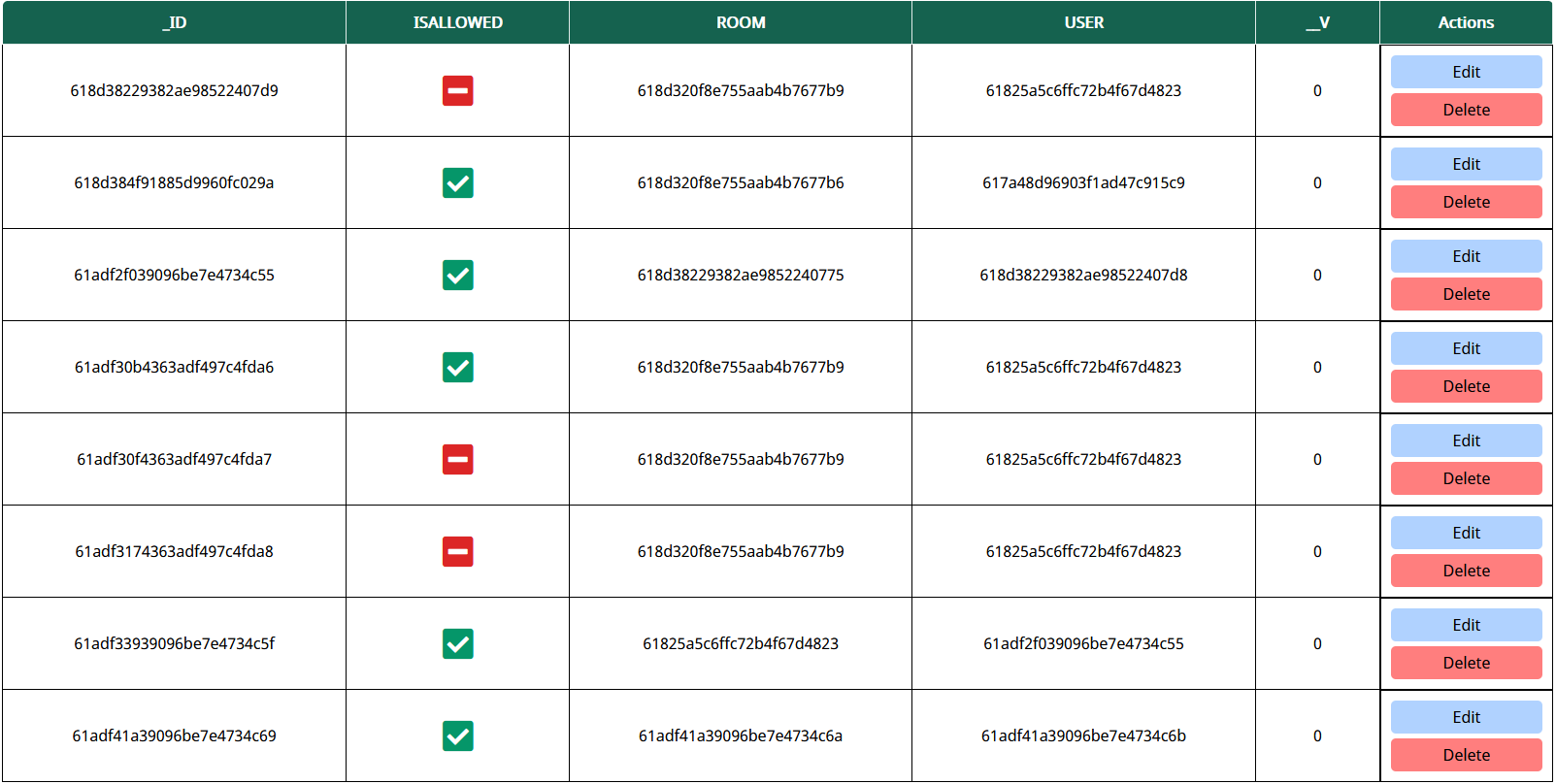


Рисунок 2.1 ‑ Відображення таблиці доступу

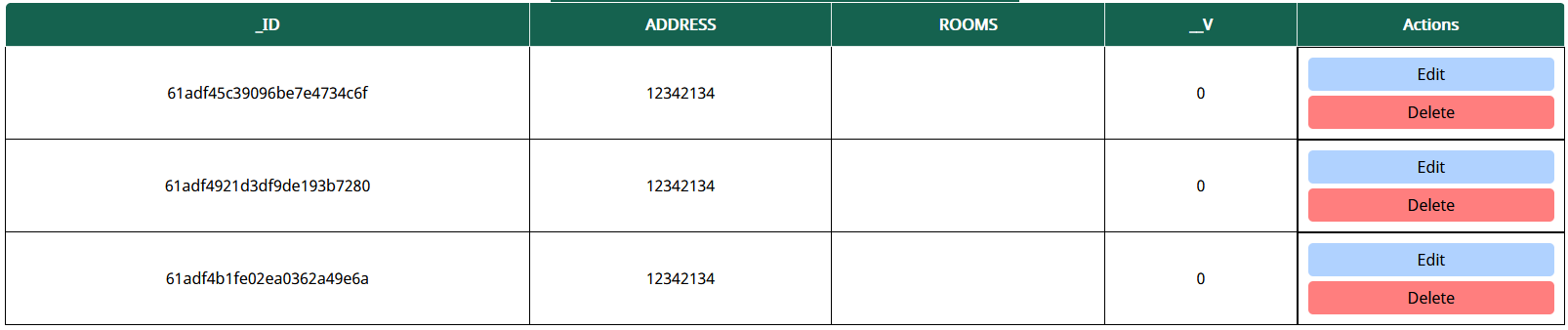


Рисунок 2.2 – Відображення таблиці будівель

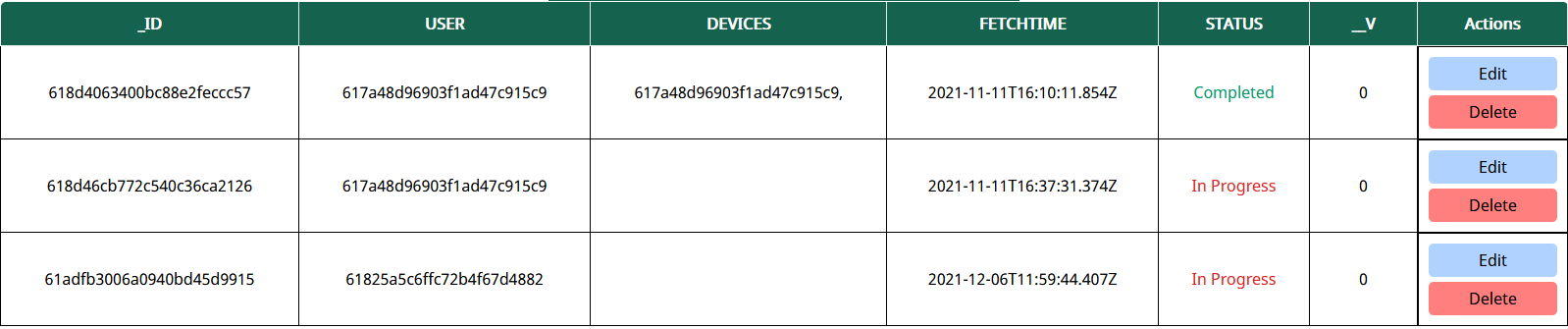


Рисунок 2.3 – Відображення таблиці запитів

Завдяки компоненту Form було створено наступні форми взаємодії із базою даних: форма створення доступу (див. рис. 2.4), форма редагування доступу (див. рис. 2.5), форма видалення доступу (див. рис. 2.6), тощо.

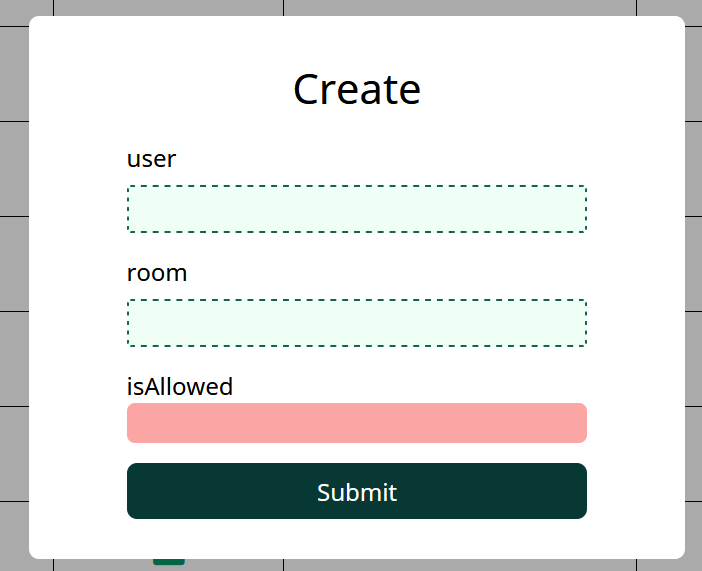


Рисунок 2.4 – Відображення форми створення доступу

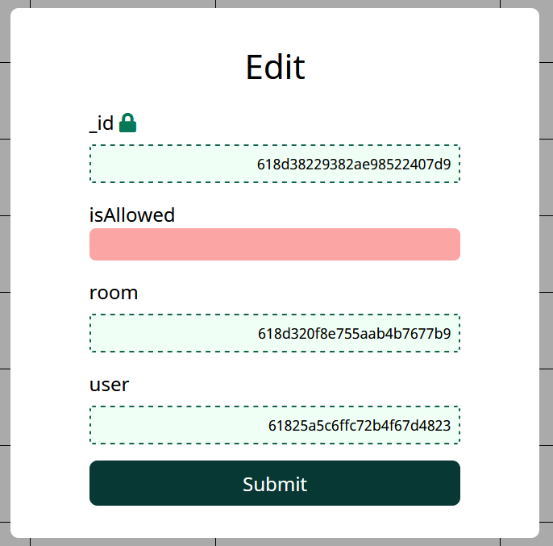


Рисунок 2.5 – Відображення форми редагування доступу

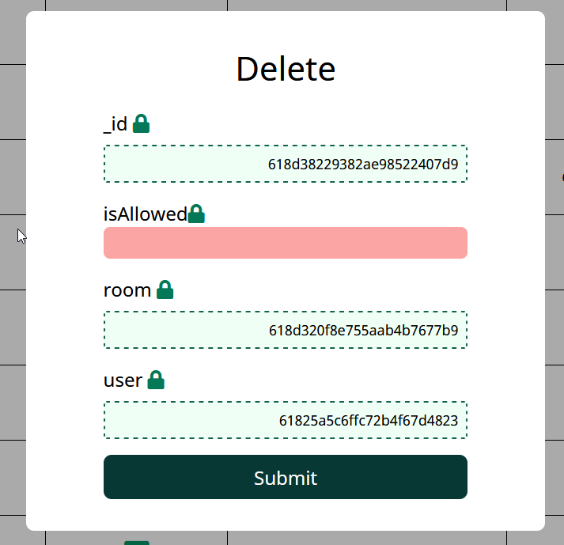


Рисунок 2.6 – Відображення форми створення доступу

Інтернаціоналізація додатку включає в себе локалізацію та зміну формату дати в залежності від обраної локалі. Для зміни та зберігання стану було створено зріз сховища, який зберігає інформацію про обрану локаль. Зріз допомагає передавати файл локалізації у будь-який компонент об’єктної моделі документу. Код зрізу наведено нижче:

import { createSlice } from '@reduxjs/toolkit'

import store from '.';

import EN\_US\_LANG from '../assets/lang/en\_us.json'

import UA\_UA\_LANG from '../assets/lang/ua\_ua.json'

import { Language } from '../models/Language';

const initialState: any = {

  language: UA\_UA\_LANG,

  code: Language.UA\_UA

}

const localization = createSlice({

  name: "lang",

  initialState,

  reducers: {

    switchLanguage: (state, action) => {

      switch (action.payload) {

        case Language.EN\_US:

          return { ...state, language: EN\_US\_LANG, code: action.payload }

        case Language.UA\_UA:

          return { ...state, language: UA\_UA\_LANG, code: action.payload }

      }

    }

  }

});

export const {

  switchLanguage

} = localization.actions

export const selectLanguagePack = (state: any) => state.lang.language

export const selectLanguageCode = (state: any) => state.lang.code

export const changeLanguagePack = (newPack: Language) => (dispatch: typeof store.dispatch) => {

  dispatch(switchLanguage(newPack));

}

export default localization.reducer

Основна бізнес логіка веб-застосунку закладена у визначенні статусу системи, бачити чи працюють певні пристрої чин і, а також підрахунок статистики системи.

**3 ПОВЕДІНКА СИСТЕМИ**

Для розробки функціоналу систему були створені UML діаграми діяльності та станів. Їх можна подивитися у додатку Д.

Діаграма діяльності відображає поведінку системи при різних діях користувача. Точкою старту на діаграмі є авторизації користувача. Далі відбуваються відгалуження, які описують поведінку системи в основних ситуаціях її роботи.

Діаграма станів починається з запуску сайту, закінчується в закритті сайту. Дана діаграма відображає основні стани системи, з якими взаємодіє користувач.

При авторизації адміністратор автоматично перенаправляється до сторінки адміністратора і працює тільки з нею та пов’язаними з нею сторінками.

Сторінка авторизації (див. рис. 3.1) дозволяє користувачу отримати доступ до системи шляхом введення коректних даних акаунту адміністратора. Зареєструвати адміністратора може лише адміністратор.



Рисунок 3.1 – Сторінка авторизації

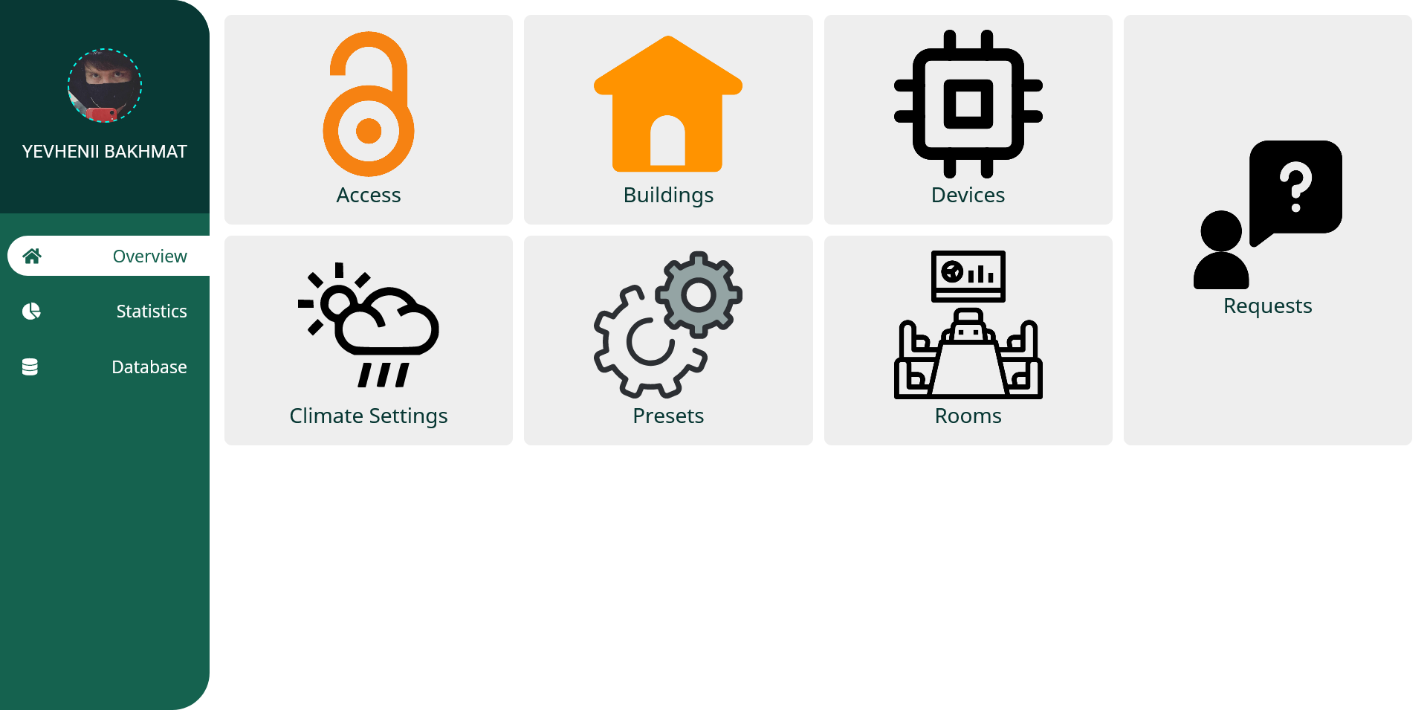
Після авторизації користувач потрапляє на головну сторінку адмінської панелі (див. рис. 3.2).

Рисунок 3.2 – Головна сторінка панелі адміністратора

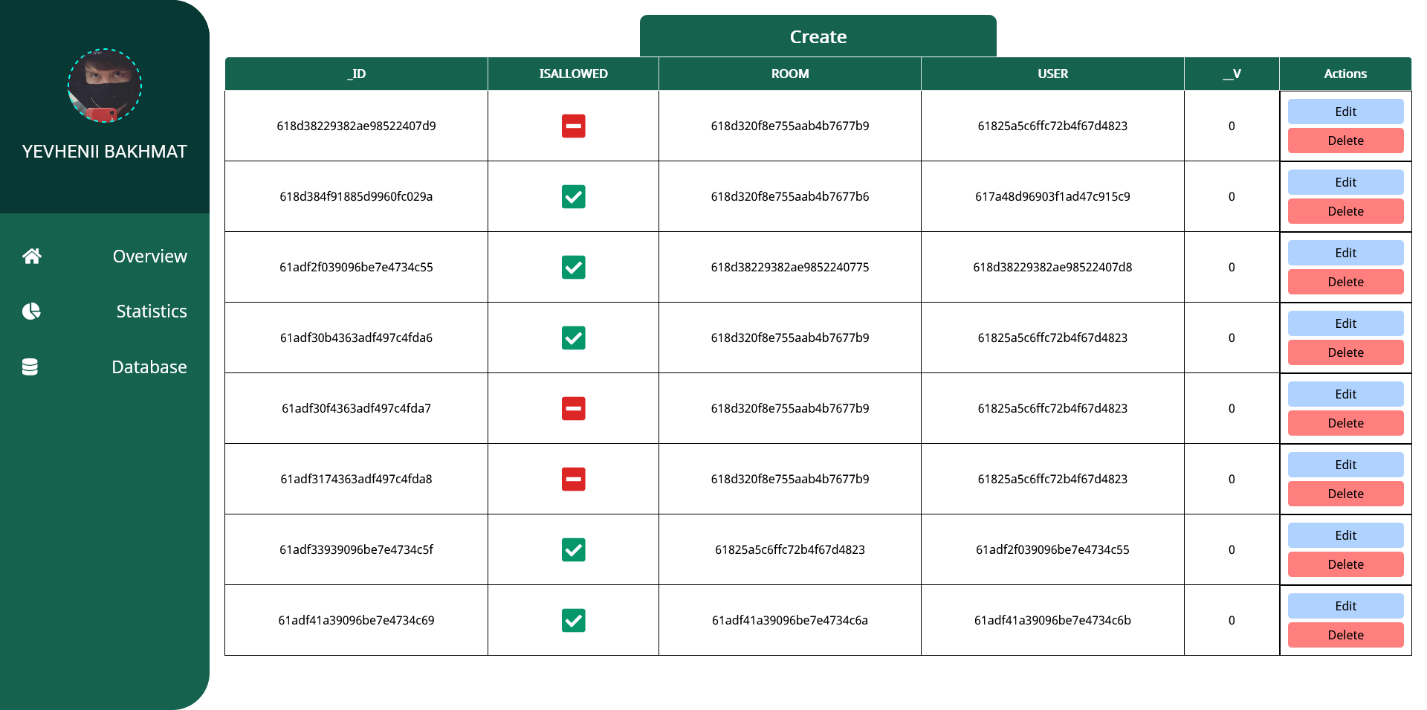
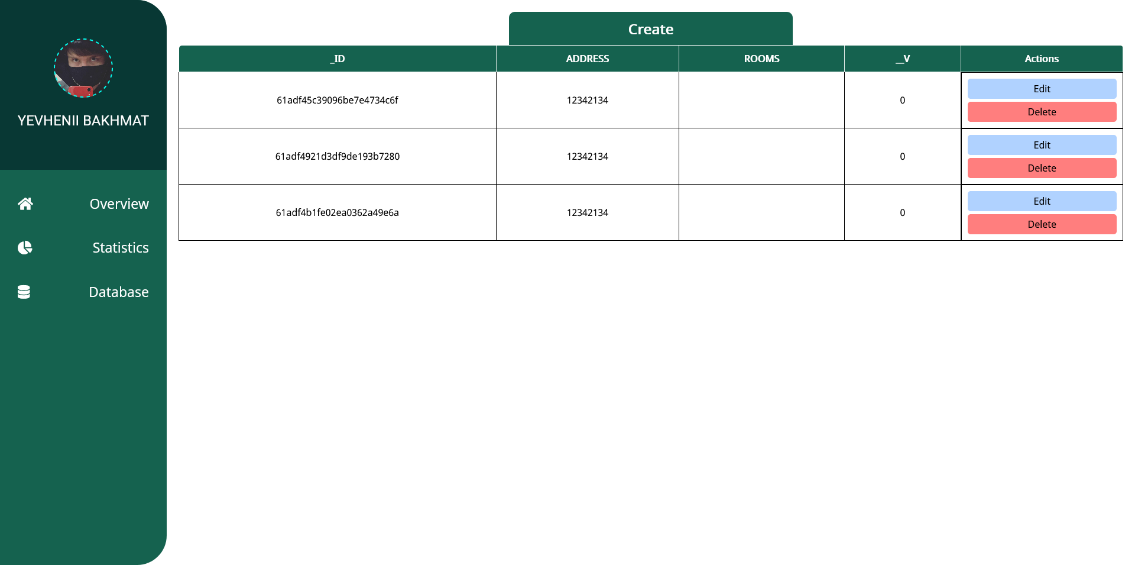
Далі адміністратор може переглянути сторінки з усима даними доступу (див. рис. 3.3), інформацію про будівлі (див. рис. 3.4), пристроїв (див. рис. 3.5), настройок клімату (див. рис. 3.6), пресетів (див. рис. 3.7), кімнат (див. рис. 3.8) та запитів зміни клімату (див. рис. 3.9).

Рисунок 3.3 – Сторінка перегляду даних доступу

Рисунок 3.4 – Сторінка перегляду інформації про будівлі

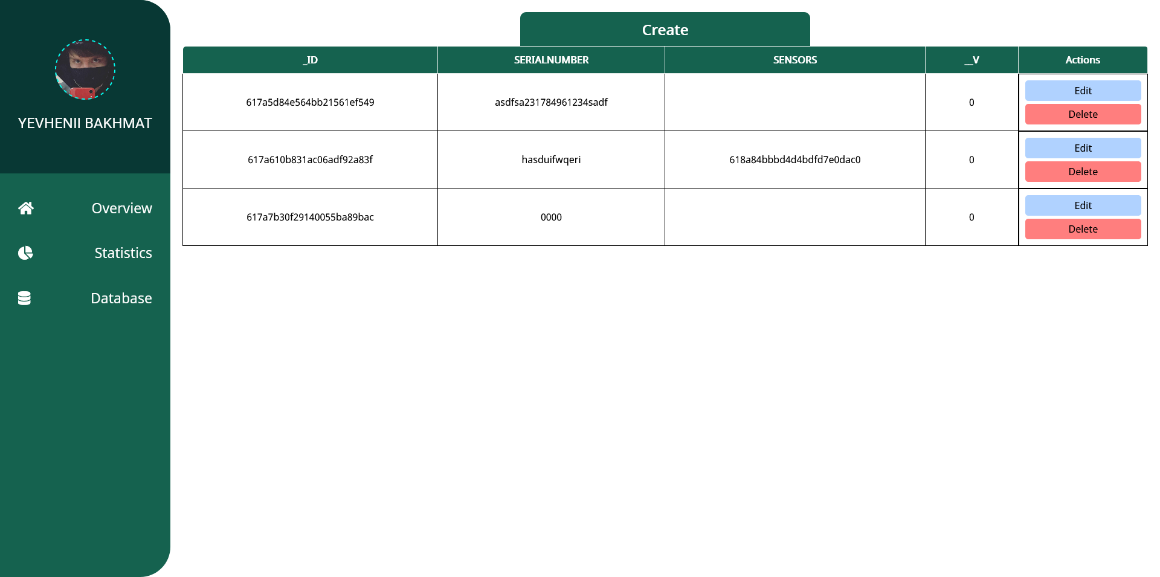


Рисунок 3.5 – Сторінка перегляду інформації про пристрої

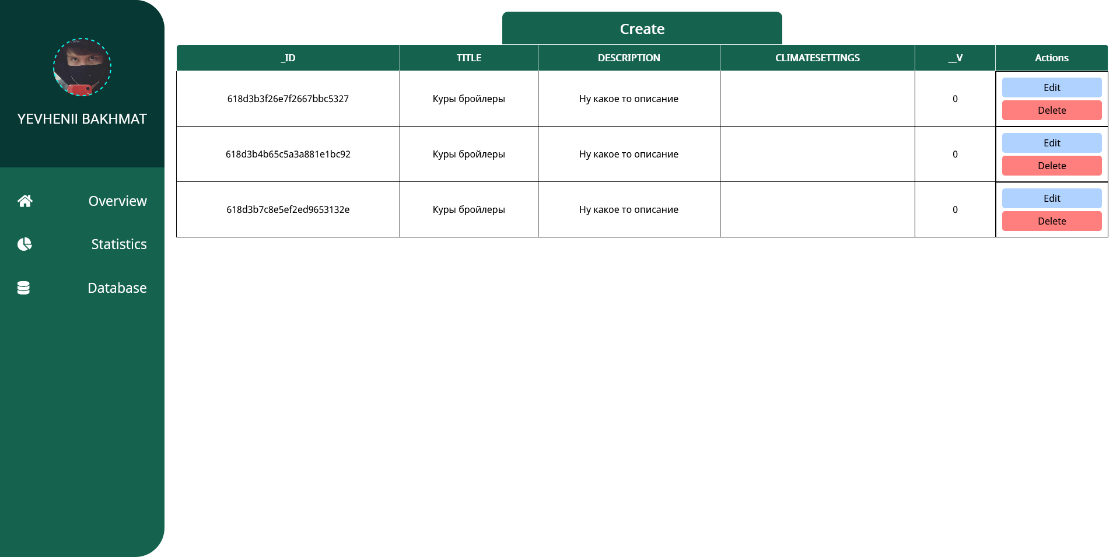


Рисунок 3.7 – Сторінка перегляду інформації про пресети

 Рисунок 3.8 – Сторінка перегляду інформації про кімнати

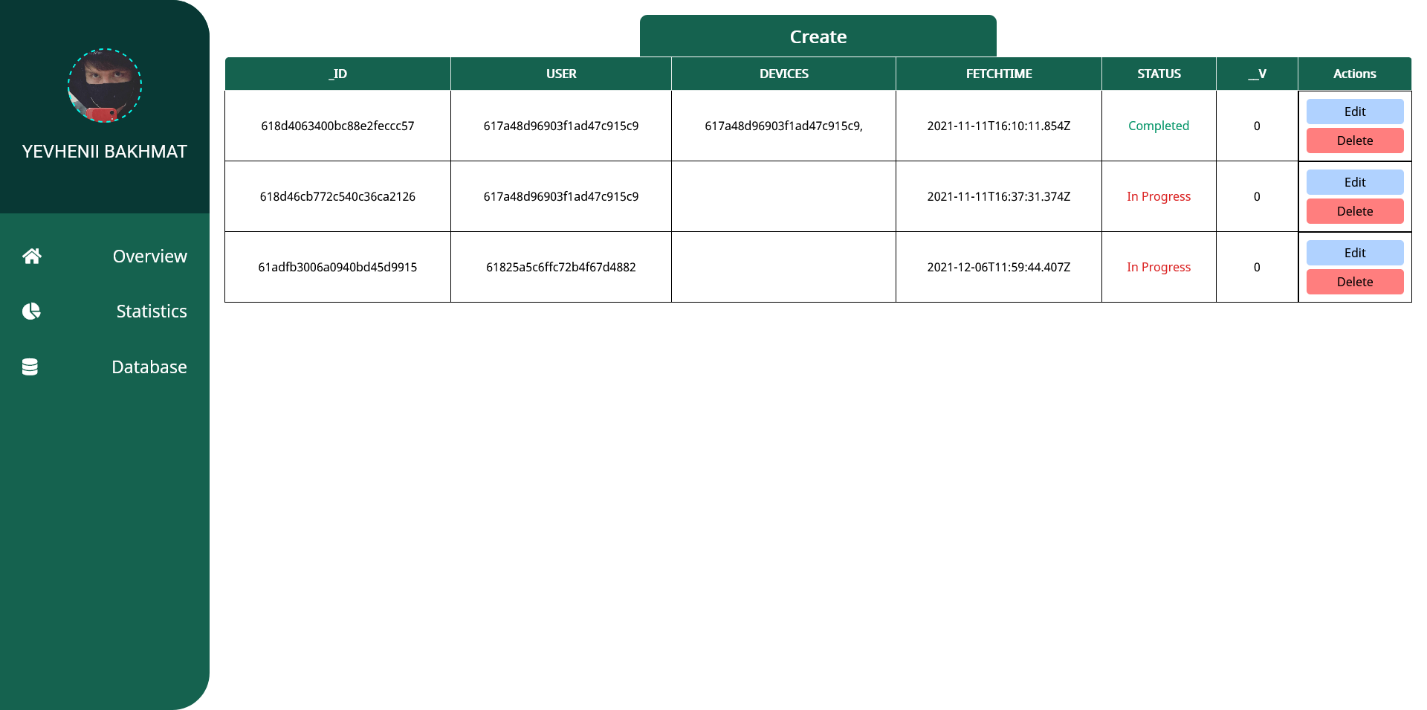


Рисунок 3.9 – Сторінка перегляду інформації про запити зміни клімату

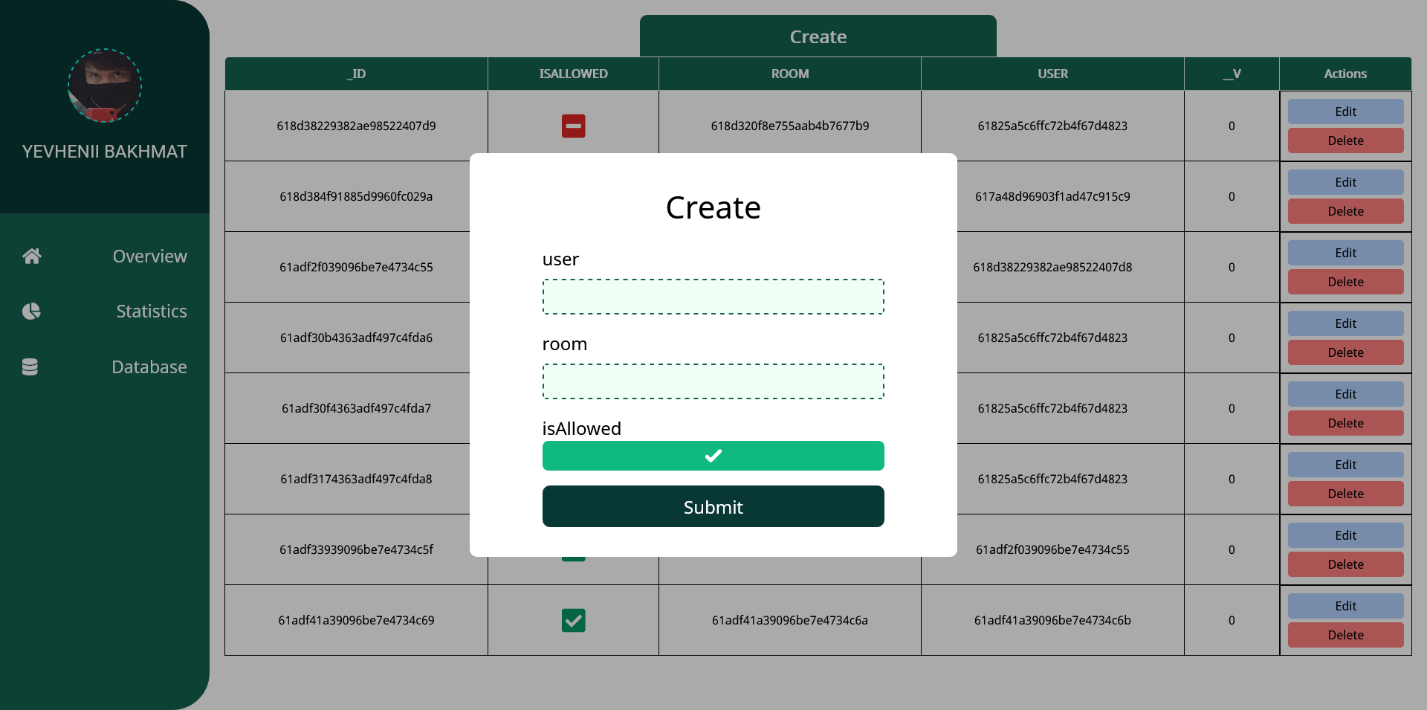
Далі у користувача є системні можливості виконання основних CRUD операцій пов’язаних із обраною моделлю системи (наприклад кімнати, доступ, пристрої). За допомогою кнопки «Create» користувач може створювати новий запис у базі даних за допомогою спеціальної форми (див. рис. 3.10). Кнопка «Edit» дозволяє редагувати інформацію за допомогою форми (див. рис 3.11). Кнопка «Delete» видаляє запис із системи,попередньо показуючи форму підтвердження видалення (див. рис. 3.12)

Рисунок 3.10 – Вигляд форми створення даних доступу

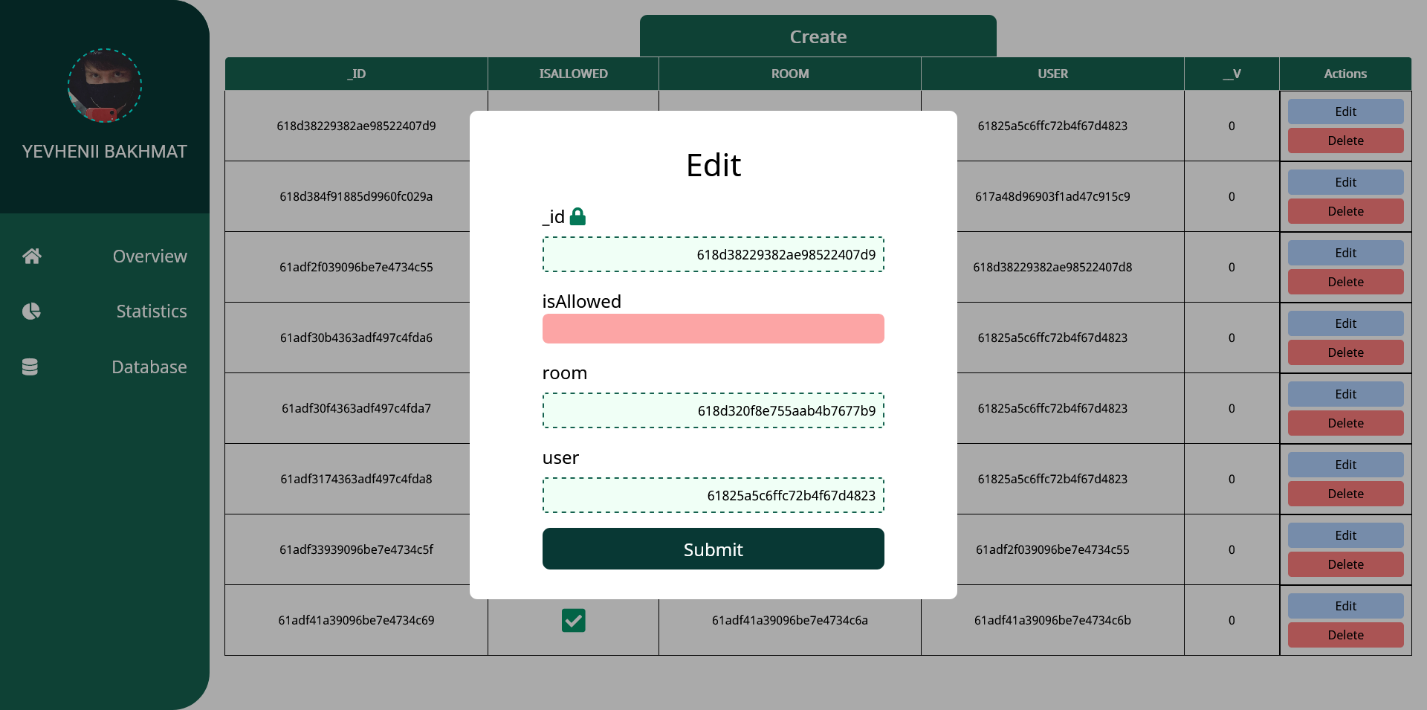


Рисунок 3.11 – Вигляд форми редагування даних доступу

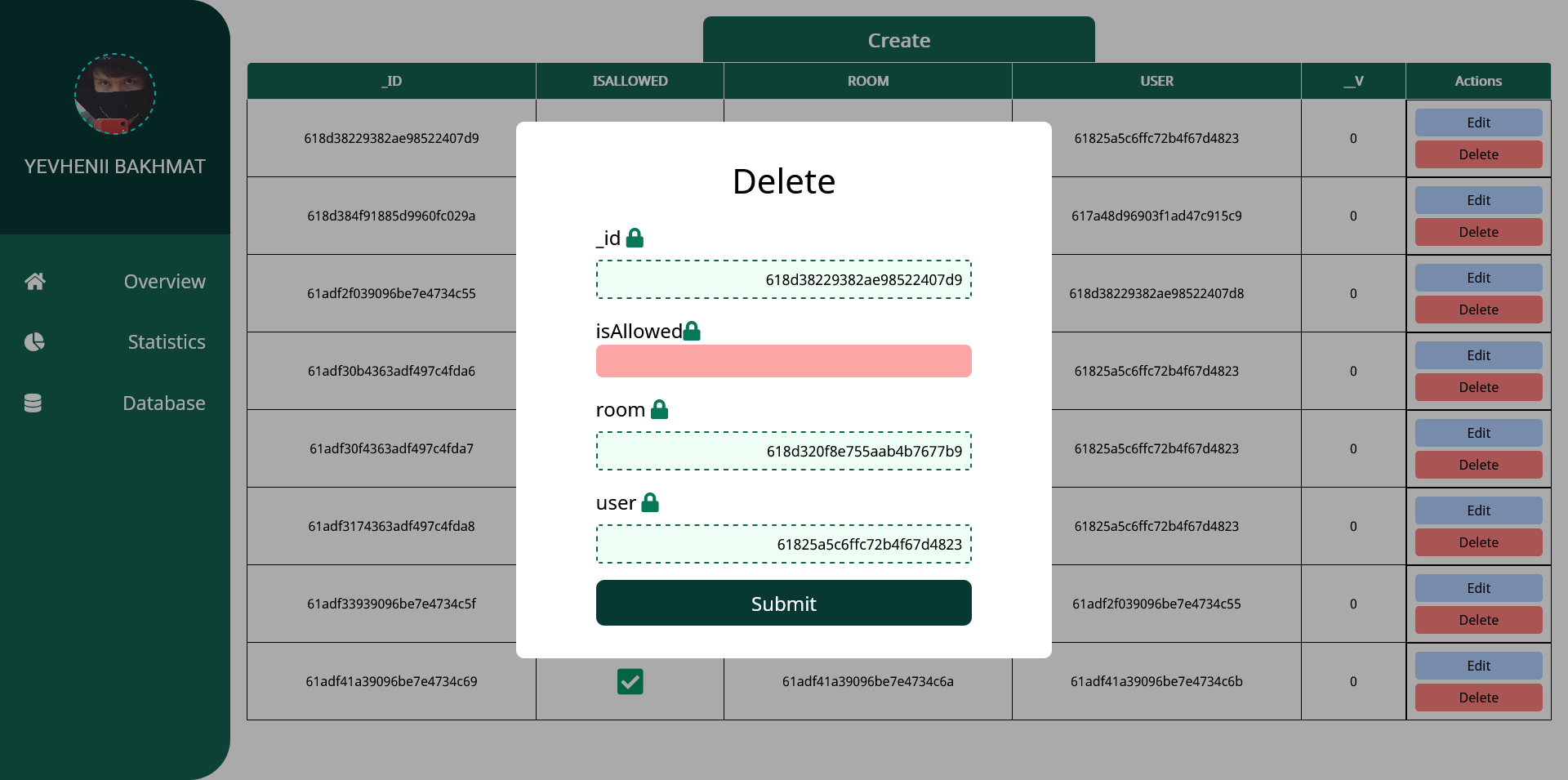


Рисунок 3.12 – Вигляд форми видалення даних доступу

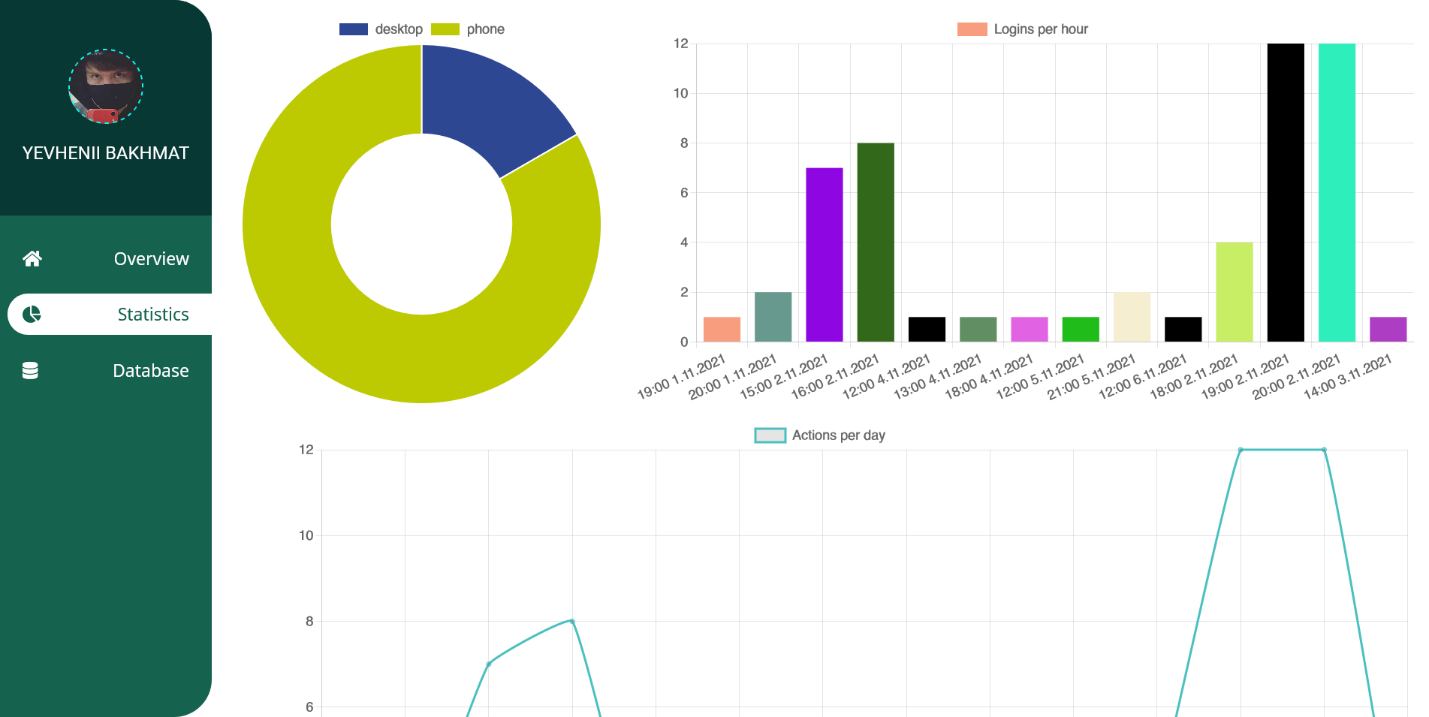
При натисканні кнопки «Statistics» користувач переходить на сторінку перегляду статистики (див. рис. 3.13), на якій представлені статистичні дані у вигляді діаграм та графіків. Сервіс збирає інформацію про кількість авторизацій у застосунках в день, кількість дій (створення, редагування, видалення або створеня запитів) в день та загальну кількість авторизацій з різних пристроїв.

Рисунок 3.13 – Сторінка перегляду статистики

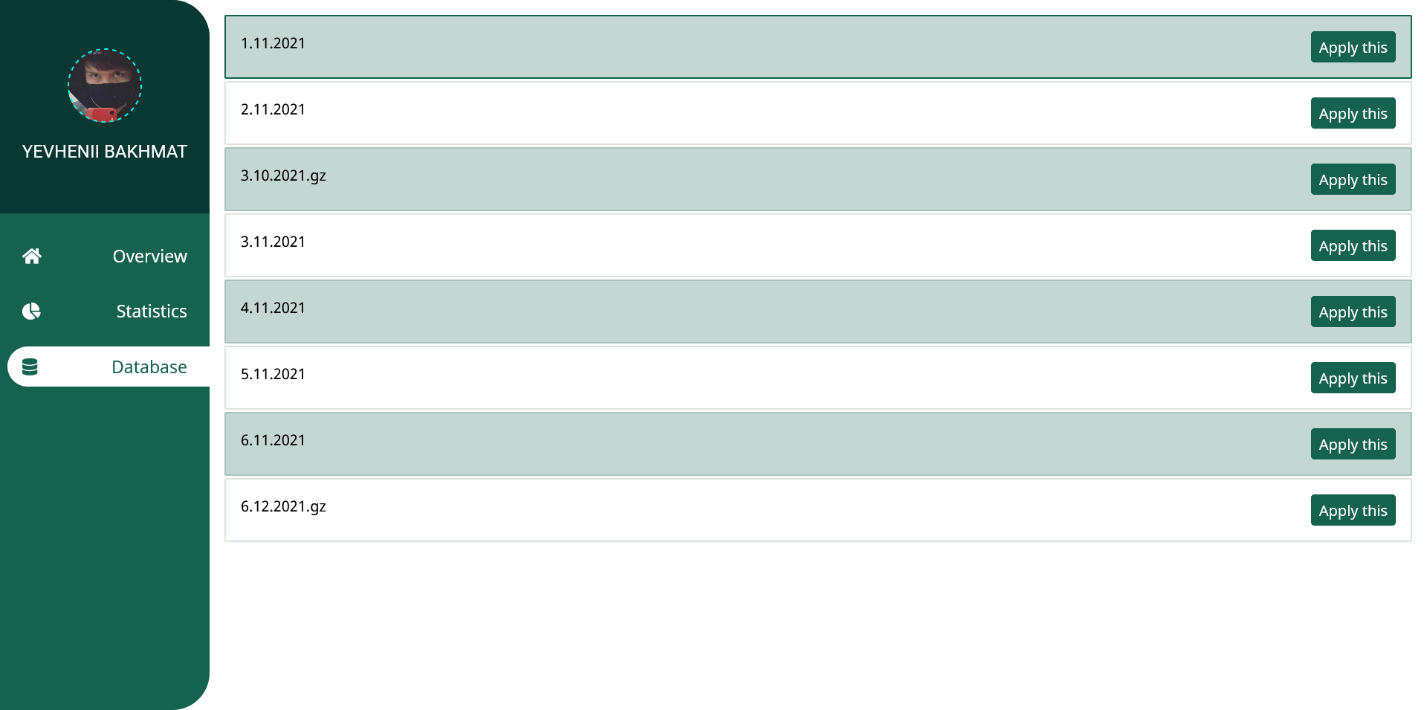
При натисканні кнопки «Database» користувач переходить на сторінку перегляду резервних копій бази даних сервісу (див. рис. 3.14). Натиснувши кнопку “Apply this” адміністратор змінить стан бази даних до обграної резервної копії.

Рисунок 3.14 – Сторінка перегляду резервних копій сервісу

**ВИСНОВКИ**

У ході виконання лабораторної роботи були отримані навички з проектування клієнтської частини програмної системи, розробки сценаріїв використання Use Case діаграми. Було проведене прогнозування поведінки системи (діаграми діяльності та станів), визначені відношення компонентів у системі (діаграма компонентів). Результатом роботи є робоча клієнтська частина програмної системи. При розробці продукту дотримувались вимоги до чистого коду та Code Convention мов програмування JavaScript та TypeScript.

Проект фронтенд частини знаходиться за посиланням:

https://drive.google.com/drive/folders/1c0VTNj6M1fXbTaRkJzkhDvJrZfA84aNa?usp=sharing

Контрольна хеш-сума архіву з проектом за алгоритмом md5: bf73c152b51295058b8cc90a31d1263b.

ДОДАТОК А

Use Case діаграма

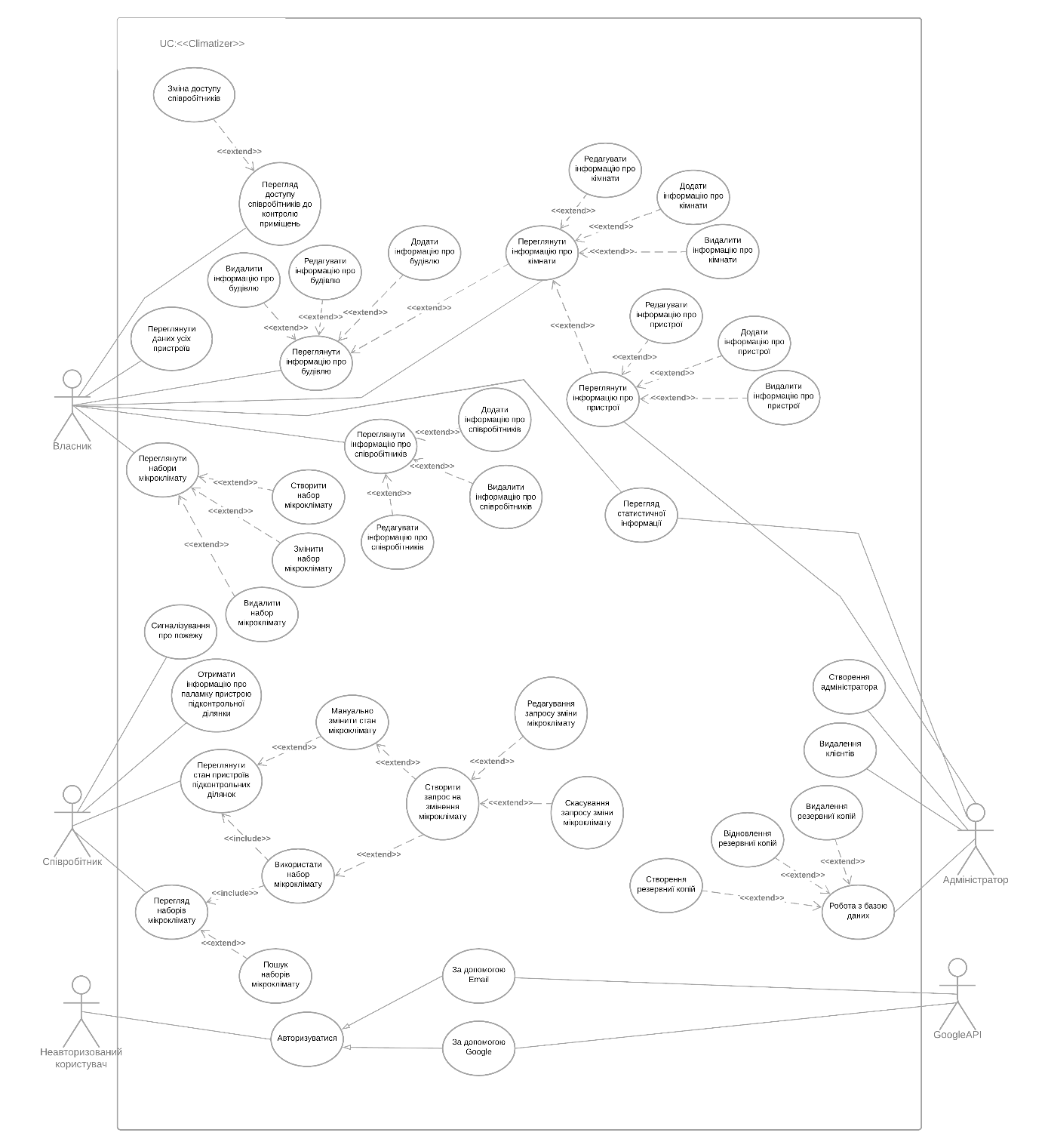


Рисунок А.1 – Use Case діаграма проекту «Програмна система універсального клімат-контролю»

ДОДАТОК Б

UML діаграма компонентів

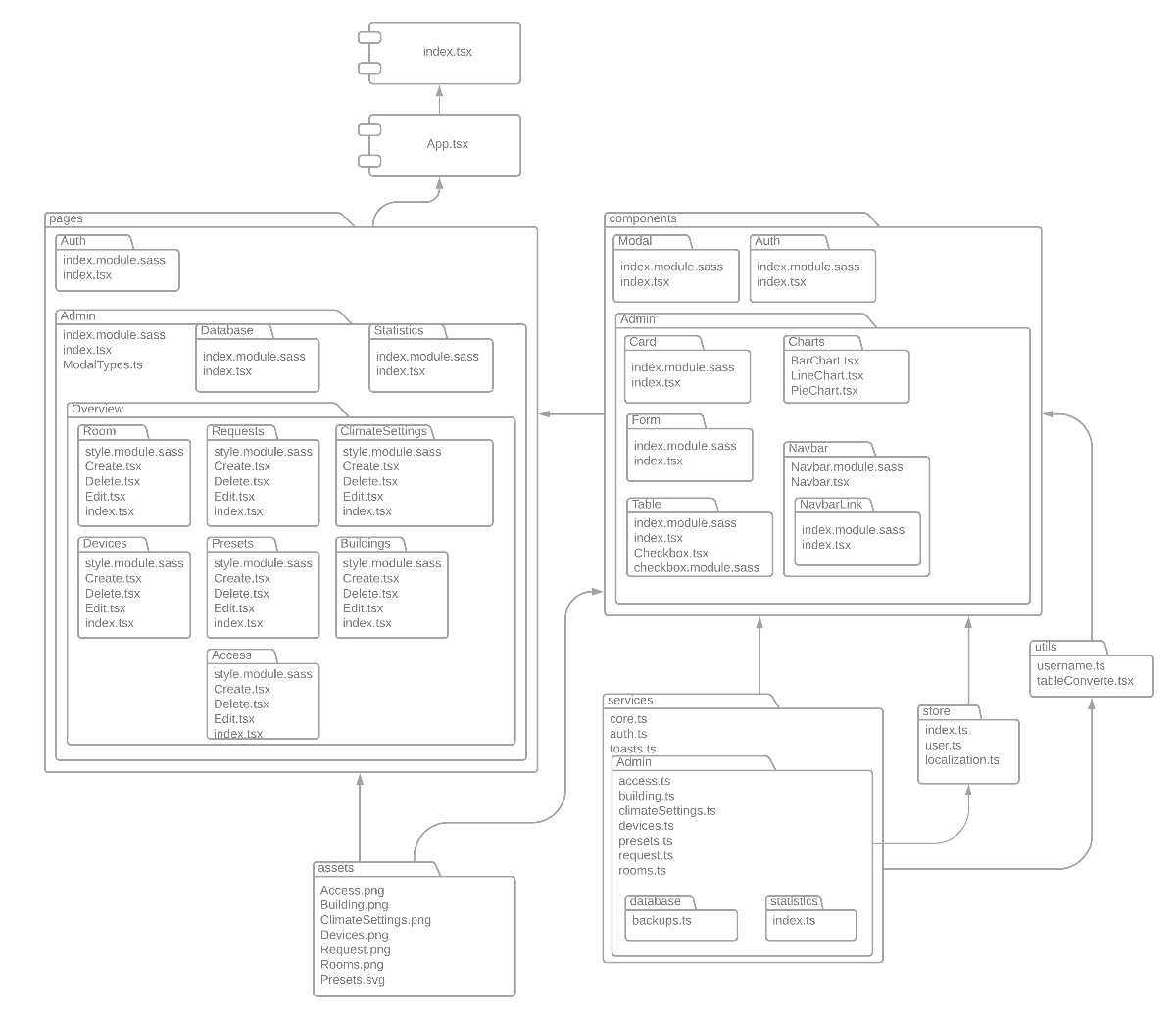


Рисунок Б.1 – UML діаграма компонентів проекту «Програмна система універсального клімат-контролю»

ДОДАТОК В

Код файлу App.tsx

import './App.css'

import { Route, Routes } from 'react-router-dom'

import Auth from './pages/Auth/'

import PrivateRoute from './PrivateRoute'

import Admin from './pages/Admin'

import Navbar from './components/Admin/Navbar/Navbar'

import Database from './pages/Admin/Database'

import Statistics from './pages/Admin/Statistics'

import Access from './pages/Admin/Overview/Access'

import Building from './pages/Admin/Overview/Buildings'

import Device from './pages/Admin/Overview/Devices'

import Request from './pages/Admin/Overview/Requests'

import Preset from './pages/Admin/Overview/Presets'

import ClimateSettings from './pages/Admin/Overview/ClimateSettings'

import Rooms from './pages/Admin/Overview/Rooms'

function App() {

  return (

    <>

      <Navbar />

      <Routes>

        <Route index element={<Auth />} />

        <Route path='/admin'>

          <Route

            index

            element={

              <PrivateRoute>

                <Admin />

              </PrivateRoute>

            }

          />

          <Route

            path='statistics'

            element={

              <PrivateRoute>

                <Statistics />

              </PrivateRoute>

            }

          />

          <Route

            path='database'

            element={

              <PrivateRoute>

                <Database />

              </PrivateRoute>

            }

          />

          <Route path='access/'>

            <Route

              index

              element={

                <PrivateRoute>

                  <Access />

                </PrivateRoute>

              }

            />

          </Route>

          <Route

            path='request'

            element={

              <PrivateRoute>

                <Request />

              </PrivateRoute>

            }

          />

          <Route

            path='building/'

            element={

              <PrivateRoute>

                <Building />

              </PrivateRoute>

            }

          />

          <Route

            path='device/'

            element={

              <PrivateRoute>

                <Device />

              </PrivateRoute>

            }

          />

          <Route

            path='climate-settings/'

            element={

              <PrivateRoute>

                <ClimateSettings />

              </PrivateRoute>

            }

          />

          <Route

            path='presets/'

            element={

              <PrivateRoute>

                <Preset />

              </PrivateRoute>

            }

          />

          <Route

            path='rooms/'

            element={

              <PrivateRoute>

                <Rooms />

              </PrivateRoute>

            }

          />

          <Route path='1' element={<PrivateRoute>13123</PrivateRoute>} />

        </Route>

      </Routes>

    </>

  )

}

export default App

ДОДАТОК Г

Код файлу Table.tsx

import React from 'react'

import { ModalTypes } from '../../../pages/Admin/ModalTypes'

import styles from './index.module.sass'

function stringify(data: any): any {

  if (/true/i.test(data))

    return <i className='fas fa-check-square text-green-600 text-4xl' />

  else if (/false/i.test(data))

    return <i className='fas fa-minus-square text-red-600 text-4xl' />

  else if (Array.isArray(data)) return data.join(', ')

  else if (/Completed/.test(data))

    return <p className='text-green-600'>{data}</p>

  else if (/In Progress/.test(data))

    return <p className='text-red-600'>{data}</p>

  return <p className={`truncate text-ellipsis`}>{data}</p>

}

const RenderRow = (props: any) =>

  props.keys.map((key: any, index: any) => (

    <td key={props.data[key]}>{stringify(props.data[key])}</td>

  ))

const Table = (props: any) => {

  const getKeys = () => Object.keys(props.data[0])

  const getHeader = () =>

    getKeys().map((key, index) => <th key={key}>{key.toUpperCase()}</th>)

  const handleButtonClick = (object: Object, modal: ModalTypes) => {

    props.onSelect(object, modal)

  }

  const getRowsData = () => {

    const items = props.data

    const keys = getKeys()

    return items.map((row: any, index: number) => (

      <tr className={styles['tr-body']} key={index}>

        <RenderRow key={index} data={row} keys={keys} />

        {!props.readOnly && (

          <td style={{ display: 'flex', flexDirection: 'column', gap: '5px' }}>

            <button

              onClick={() => handleButtonClick(row, ModalTypes.Edit)}

              className={`${styles['button']} ${styles['edit']}`}

            >

              Edit

            </button>

            <button

              onClick={() => handleButtonClick(row, ModalTypes.Delete)}

              className={`${styles['button']} ${styles['delete']}`}

            >

              Delete

            </button>

          </td>

        )}

      </tr>

    ))

  }

  return (

    <div className={styles['wrapper']}>

      <table className={styles['table']}>

        <thead className={styles['thead']}>

          <tr className={styles['tr-head']}>

            {getHeader()}

            {!props.readOnly && <th>Actions</th>}

          </tr>

        </thead>

        <tbody className={styles['tbody']}>{getRowsData()}</tbody>

      </table>

    </div>

  )

}

export default Table

ДОДАТОК Д

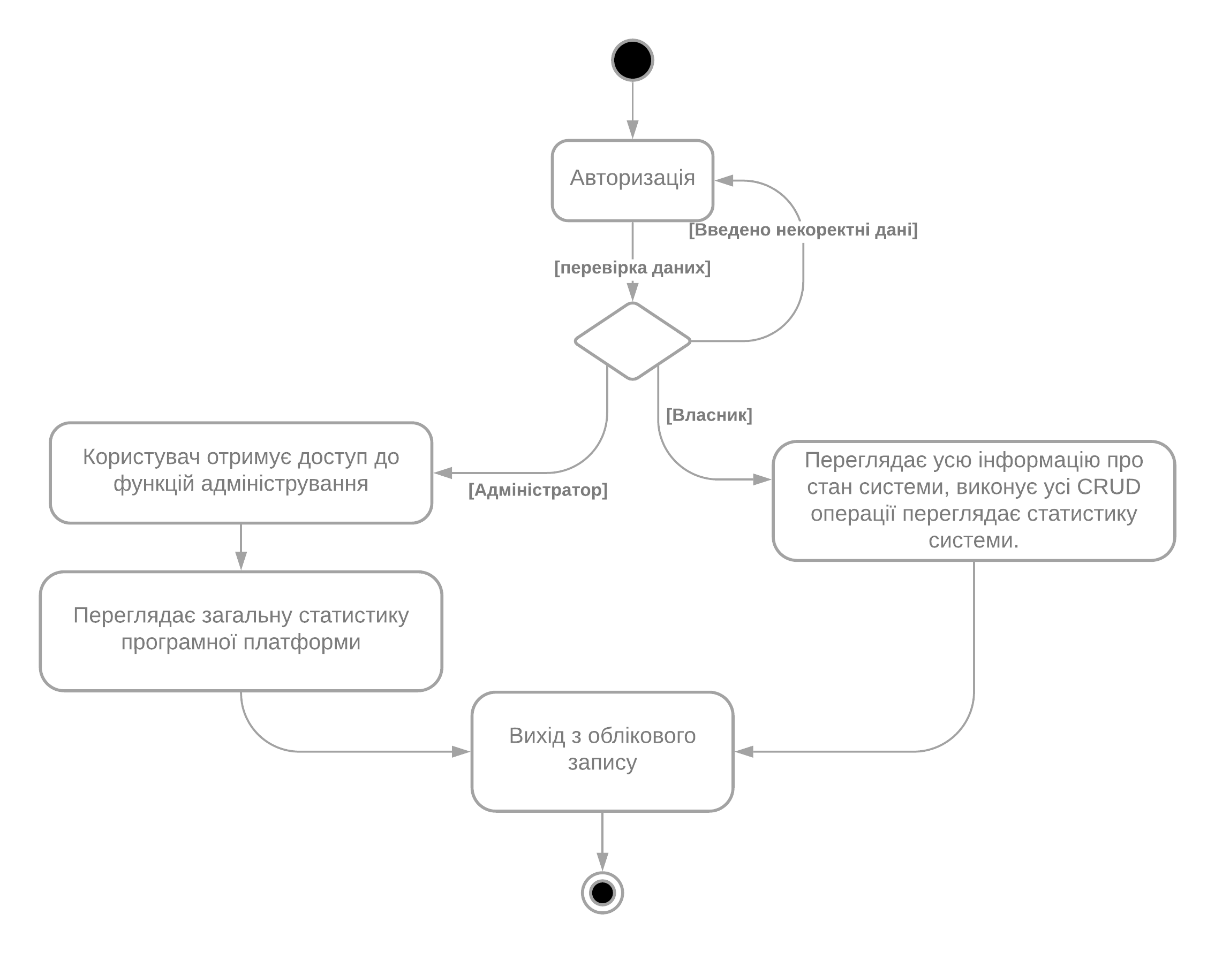
Діаграми для опису поведінки системи

Рисунок Е.1 – UML діаграма діяльності веб-частини проекту «Програмна система універсального клімат-контролю»

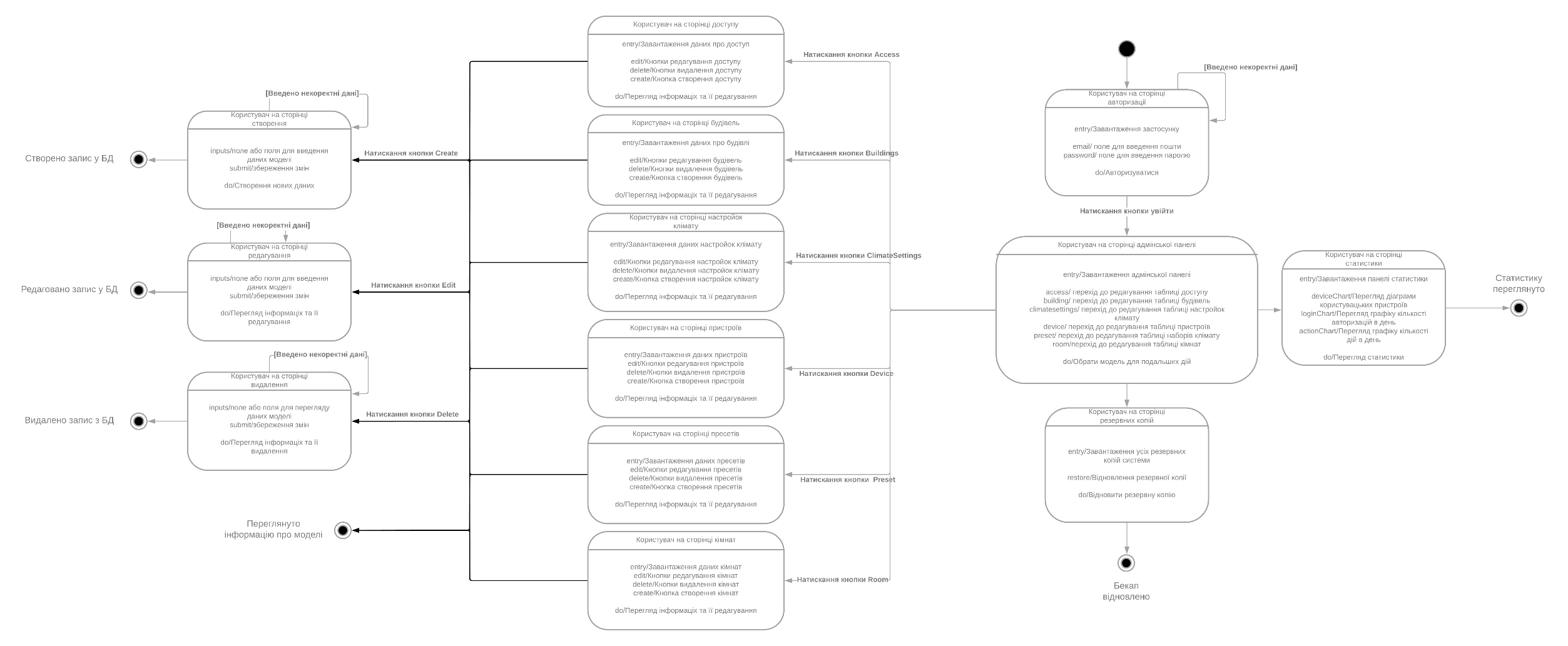


Рисунок Е.2 – UML діаграма станів веб-частини проекту «Програмна система універсального клімат-контролю»