Міністерство освіти та науки України
Харківський національний університет радіоелектроніки
Кафедра програмної інженерії

Лабораторна робота №3

З дисципліни: «Архітектура програмного забезпечення»

на тему: «Програмна система для автоматизації процесів прибирання приміщень»

Виконав
ст. гр. ПЗПІ-18-2
Кузнецов Роман Олександрович

Перевірив
ст. викл. каф. ПІ
Сокорчук Ігор Петрович

Харків 2021

Мета: розробити front-end частину для програмної системи для автоматизації процесів прибирання приміщень.

Хід роботи:

Для розробки front-end частини застосунку використано бібліотеку React, допоміжні бібліотеки mobX.js, jwt-decode.js, js-download тощо. Архітектура побудована на базі моделі MVC (Model-View-Controller), де Controller та Model розташовані в серверній частині системи, а front-end частина системи виконує роль View. Для взаємодії з серверною частиною системи використовується HTTPS протокол та JSON формат даних.

Перед реалізацією front-end частини було проаналізовано предметну область програмної системи. Було встановлено всі можливості для використання застосунку користувачами, створено діаграму варіантів використання, що описує сценарій поведінки системи при взаємодії з її користувачами. UseCase діаграма наведена у додатку А.

Front-end частина програмної системи має декілька видів акторів: підрядні працівники, які надають послуги та виконують прибирання приміщень замовників, якими виступають власники приміщень, що можуть маніпулювати своїми приміщеннями, переглядати їх стан та рівень забрудненості і складати угоди на прибирання в них, обираючи приміщення та бажаний сервіс з різними характеристиками. Також в системі наявні базові функції адміністратора, а саме маніпулювання обліковими записами в системі, управління даними, створення резервних копій.

В UML діаграма, на якій відображаються компоненти, залежності та зв'язки між ними називається діаграмою компонентів. Діаграма компонентів відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти та компоненти, що можуть виконуватись. Діаграма компонентів для клієнтської частини системи зображена на рисунку 1.

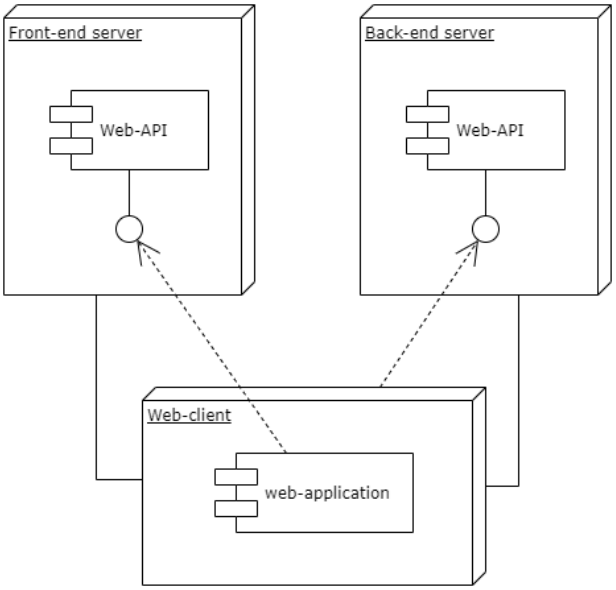

Рисунок 1 – Діаграма компонентів

З іншого боку було побудовано діаграму станів, що представляє об'єкт як автомат з теорії автоматів зі стандартизованими умовними позначеннями, зображену на рисунку 2. На діаграмі зображені стани та переходи користувача у ролі власника приміщень.

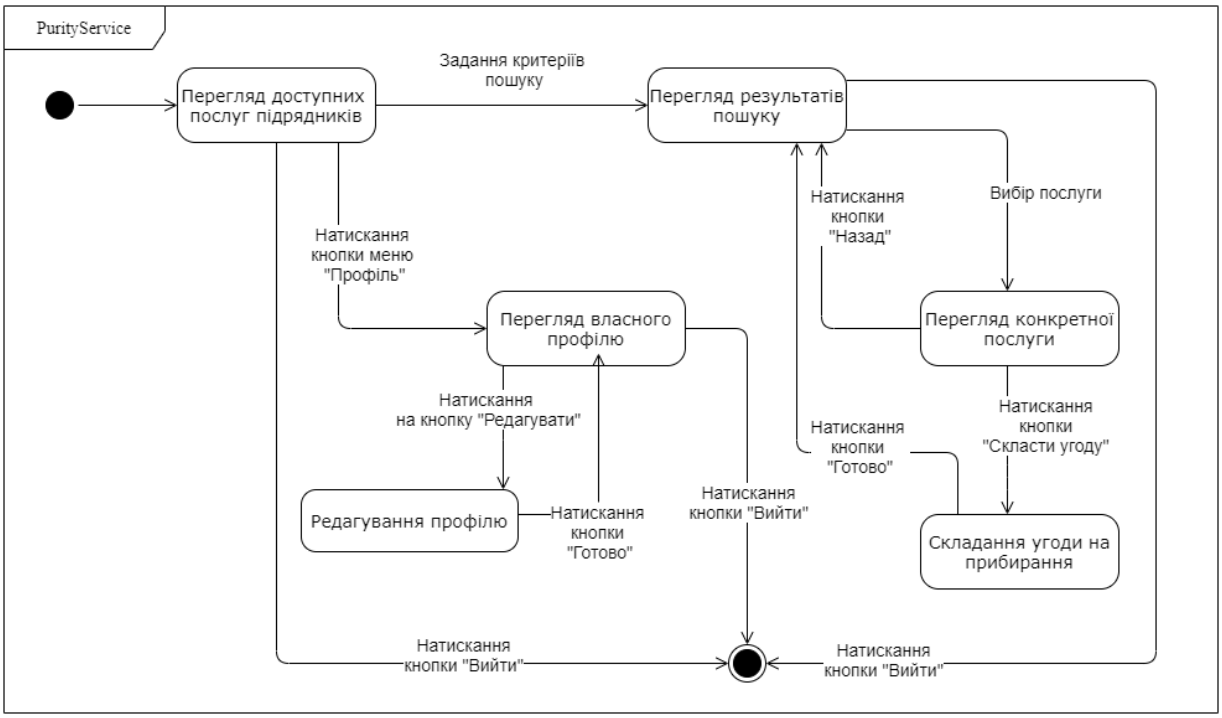

Рисунок 2 – Діаграма станів та переходів

Для більш детального опису умов переходів системи з одного стану в інший також побудовано діаграму діяльності, що наведена у додатку Б.

Архітектура клієнтського програмного застосунку побудована на компонентно-орієнтованому підході з використанням React.JS та React компонентів з управлінням їх станом за допомогою mobX.js. Програмна реалізація компоненту складання угоди на прибирання наведена у додатку В.

Локалізація системи була реалізована за допомогою технології i18n.js.

Посилання на архів з програмним кодом та файлом контрольної суми: https://drive.google.com/drive/folders/1CFmpXzByEwwNiSZ99idp06B7Tn3l_9g4?usp=sharing

Контрольна сума до архіву: 7c7fbbfb8ff55b9766fbcbabb72285ba

Висновок: під час виконання лабораторної роботи було розроблено front-end частину для програмної системи автоматизації прибирання приміщень.

ДОДАТОК А.
UseCase діаграма системи

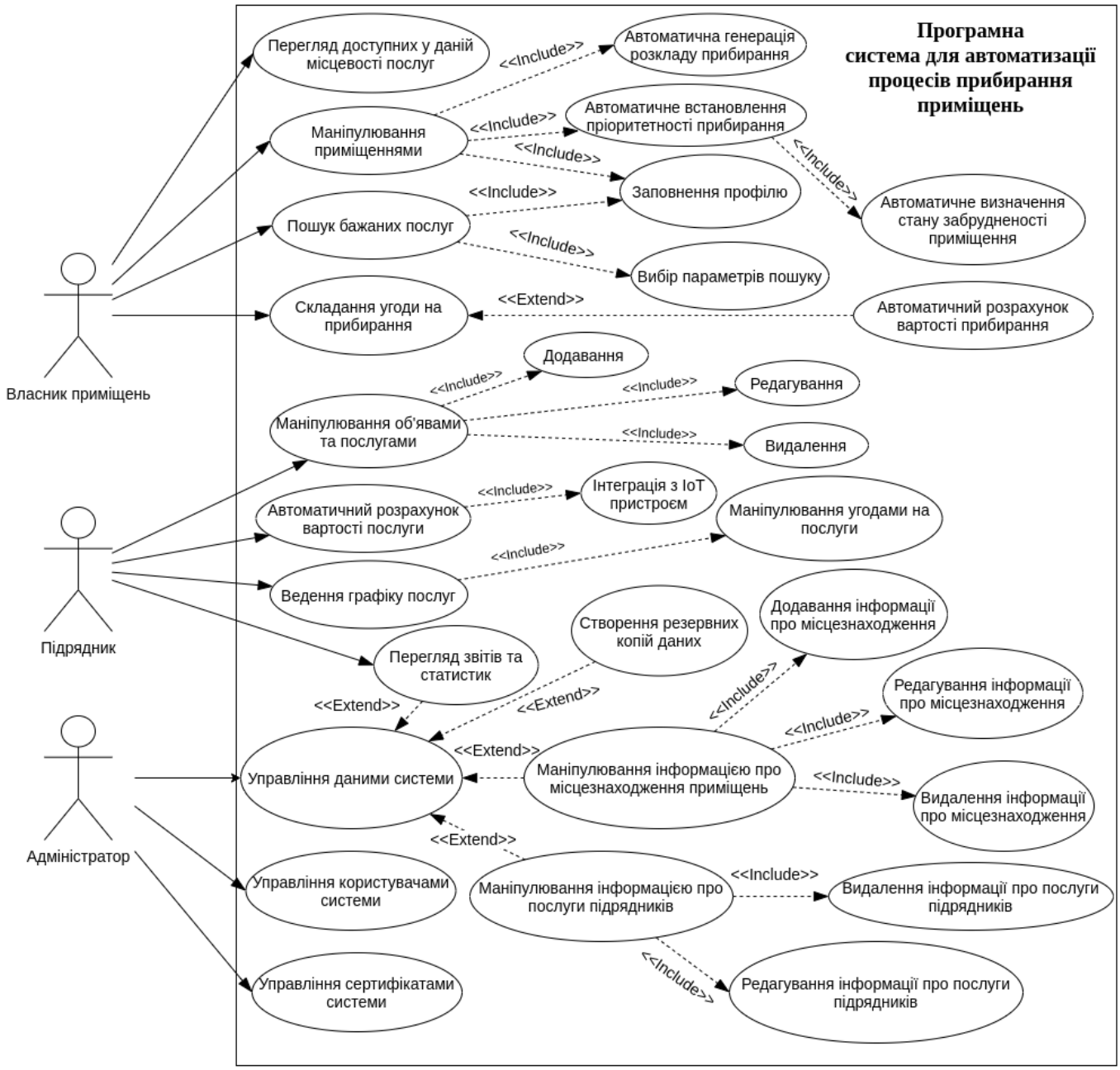

Рисунок А.1 – Діаграма варіантів використання

ДОДАТОК Б.
Діаграма діяльності системи

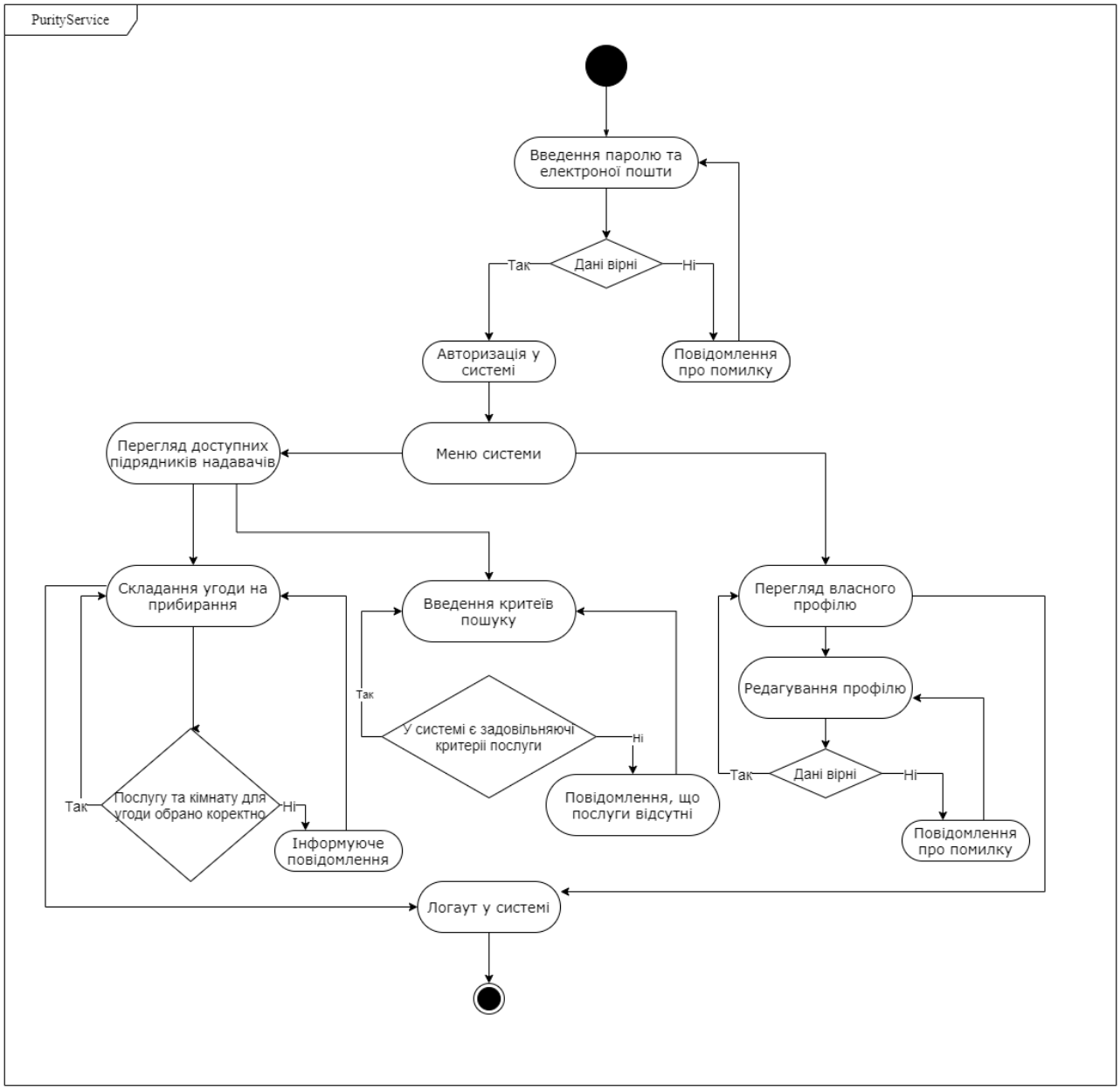

Рисунок Б.1 – Діаграма діяльності

ДОДАТОК В.

Програмна реалізація компоненту складання угоди на прибирання

```
1   class SignContract extends React.Component {
2       constructor(props) {
3           super(props)
4           this.state = {
5               rooms: '',
6               roomId: 0,
7               roomsId: [],
8               services: [],
9               serviceId: 0,
10              servicesId: [],
11              buttonDisabled: false
12          }
13      }
14
15      componentDidMount() {
16          fetch(`${url}/placement-owners/${decoded.email}/placements`,
17              {
18                  method: 'get',
19                  headers: {
20                      'Accept': 'application/json',
21                      'Content-Type': 'application/json',
22                      'Authorization':
23                          'Bearer ' + localStorage.getItem('Token')
24                  }
25              }
26          )
27              .then(res => res.json())
28              .then(result => {
29                      this.setState({
30                          isLoaded: true,
31                          rooms: result
32                      });
33                      this.state.rooms.forEach(element => {
34                          this.state.roomsId.push(element.id)
35                      });
36                  },
37                  (error) => {
38                      this.setState({
39                          isLoaded: true,
40                          error
41                      });
42                  }
43              );
44
45          fetch(`${url}/cleaning-
    providers/${localStorage.getItem("cleaningMail")}/services`, {
46                  method: 'get',
47                  headers: {
48                      'Accept': 'application/json',
49                      'Content-Type': 'application/json',
50                      'Authorization':
51                          'Bearer ' + localStorage.getItem('Token')
52                  }
```

```
            }
        )
            .then(res => res.json())
            .then(
                (result) => {
                    this.setState({
                        isLoaded: true,
                        services: result
                    });
                    console.log(this.state.services)
                    this.state.services.forEach(element => {
                        this.state.servicesId.push(element.id)
                    });
                },
                (error) => {
                    this.setState({
                        isLoaded: true,
                        error
                    });
                }
            )
    }

    async signContract() {
        try {
            let res = await fetch(`${url}/contracts`, {
                    method: 'post',
                    headers: {
                        'Accept': 'application/json',
                        'Content-Type': 'application/json',
                        'Authorization':
                            'Bearer ' + localStorage.getItem('Token')
                    },
                    body: JSON.stringify({
                        providerServiceId: this.state.serviceId,
                        date: new Date(),
                        placementId: this.state.roomId
                    })
                }
            )
            let result = await res.json()
            if (result) {
                window.location.href = './contracts';
            }
        } catch (e) {
            console.log(e)
            this.resetForm()
        }
    }

    handleRChange = (event) => {
        this.setState({
            roomId: event.value
        });
    }

    handleSChange = (event) => {
        this.setState({
```

```
            serviceId: event.value
        });
    }

    render() {
        const {t} = this.props
        return (
            <div className="signIn">
                <Header/>
                <div className="container">
                    <div className="signInForm">
                        <div className='signInContainer'>
                            <h1>{t('SContract')}</h1>
                            <div>
                                <p>{t("pickSId")}</p>
                                <Dropdown options={this.state.servicesId}
                                          onChange={this.handleSChange}
                                          value={this.state.value}
                                          placeholder={t("Select an id")}/>
                                <p>{t("pickRId")}</p>
                                <Dropdown options={this.state.roomsId}
                                          onChange={this.handleRChange}
                                          value={this.state.value}
                                          placeholder={t("Select an id")}/>
                            </div>
                            <Button
                                text={t('SContract')}
                                disabled={this.state.buttonDisabled}
                                onClick={() => this.signContract()}
                            />
                        </div>
                    </div>
                </div>
            </div>
        )
    }
}

export default withTranslation()(SignContract);
```