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

Лабораторна робота №3
з дисципліни: «Архітектура програмного забезпечення»
на тему: «Програмна система автоматизації ведення флористичного бізнесу»

Виконала
ст. гр. ПЗПІ-18-5
Борщова О. В.

Перевірив
ст. викл. каф. ПІ
Сокорчук І. П.

Харків 2021

Мета: розробити front-end частину для програмної системи автоматизації ведення флористичного бізнесу (зберігання, відстеження стану, перевезення рослин та квіткової продукції).

Хід роботи:

Для розробки front-end частини застосунку використано JavaScript-бібліотеку React з відкритим вихідним кодом. Для взаємодії з серверною частиною системи використовується HTTPS протокол, JSON формат транспортування даних, що у HTTP термінології позначається зазвичай як application/json, та JavaScript функція fetch() для посилання запитів.

Перед тим, як програмно реалізувати front-end частину, було проаналізовано предметну область та встановлено всі основні способи використання клієнтського веб-застосунку, створено UseCase діаграму, що описує сценарій поведінки застосунку у процесі взаємодії з його користувачами. UseCase діаграма наведена у додатку А.

Front-end частина програмної системи має декілька видів акторів: людина, що є власником або представником флористичного магазину, та адміністратор системи («Адміністратор»), що може керувати користувачами системи, створювати нові облікові записи, блокувати та розблоковувати існуючі акаунти.

Власник флористичного магазину має змогу редагувати свій профіль, маніпулювати приміщеннями, де зберігаються квіти та їх смарт-пристроями, типами квітів, створювати нові зберігання квітів, редагувати їх, переглядати їх, виконувати запит на автоматичний перерозподіл квітів, для яких мікроклімат є незадовільним, до інших приміщень зберігання, відстежувати стан квітів та мікроклімат приміщень зберігання.

Для відображення робочих компонентів клієнтської частини системи та відображення логіки їх взаємодії та інженерних рішень під час проектування було створено діаграму компонентів web-клієнту програмної системи. Діаграма компонентів для програмної системи автоматизації ведення флористичного бізнесу зображена на рисунку 1.

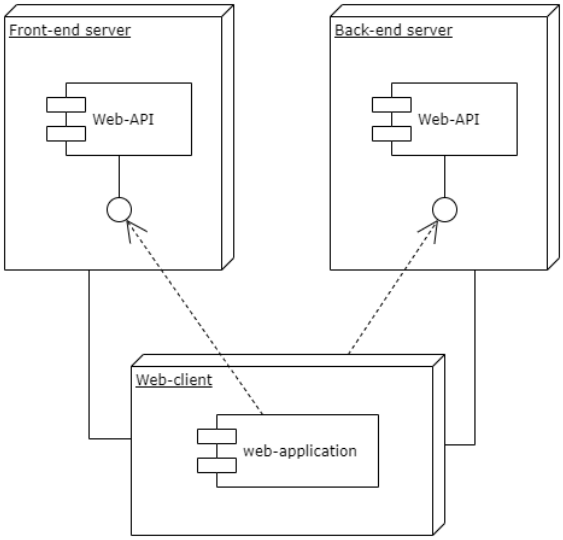

Рисунок 1 – Діаграма компонентів для веб-клієнту програмної системи «FloristikUp»

З іншого боку було побудовано діаграму, що визначає зміну станів об'єкту у часі – діаграму станів, зображену на рисунку 2.

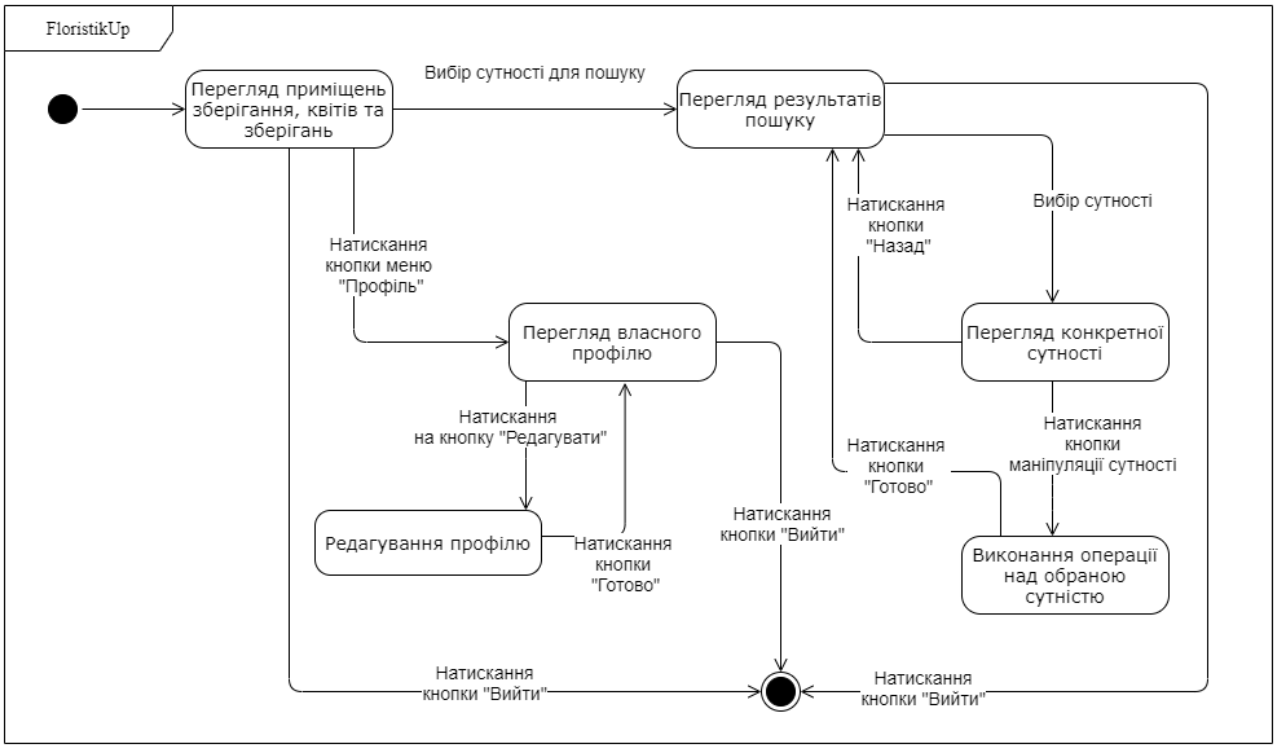

Рисунок 2 – Діаграма станів та переходів веб-клієнту програмної системи «FloristikUp»

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

Так як клієнт розроблений з використанням фреймворку React.js, в основі його архітектури знаходиться компонентний підхід, тобто вона повністю написана за допомогою розширення React компонентів. У додатку В наведено приклад програмної реалізації компоненту маніпулювання зберіганнями квітів.

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/1lw_bujv-mYKeCdR15anRcJ-AO0vsnHEB?usp=sharing

Контрольна сума до архіву: e38122579b2f884f5295bbe7df9d9510

Висновок: під час виконання лабораторної роботи було розроблено front-end частину для програмної системи автоматизації ведення флористичного бізнесу.

ДОДАТОК А
UseCase діаграма системи

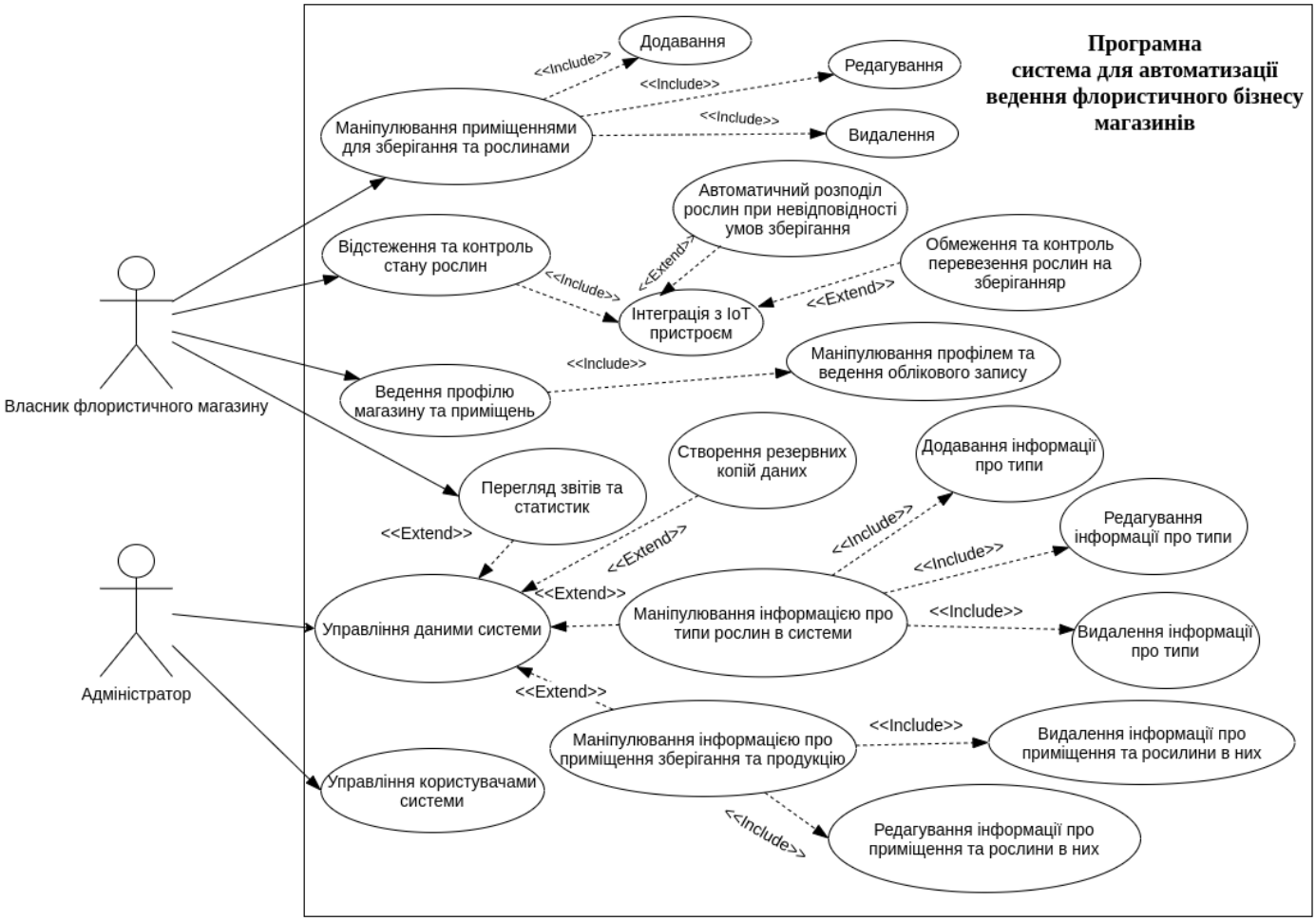

Рисунок А.1 – Діаграма варіантів використання для програмної системи «FloristikUp»

ДОДАТОК Б
Діаграма діяльності системи

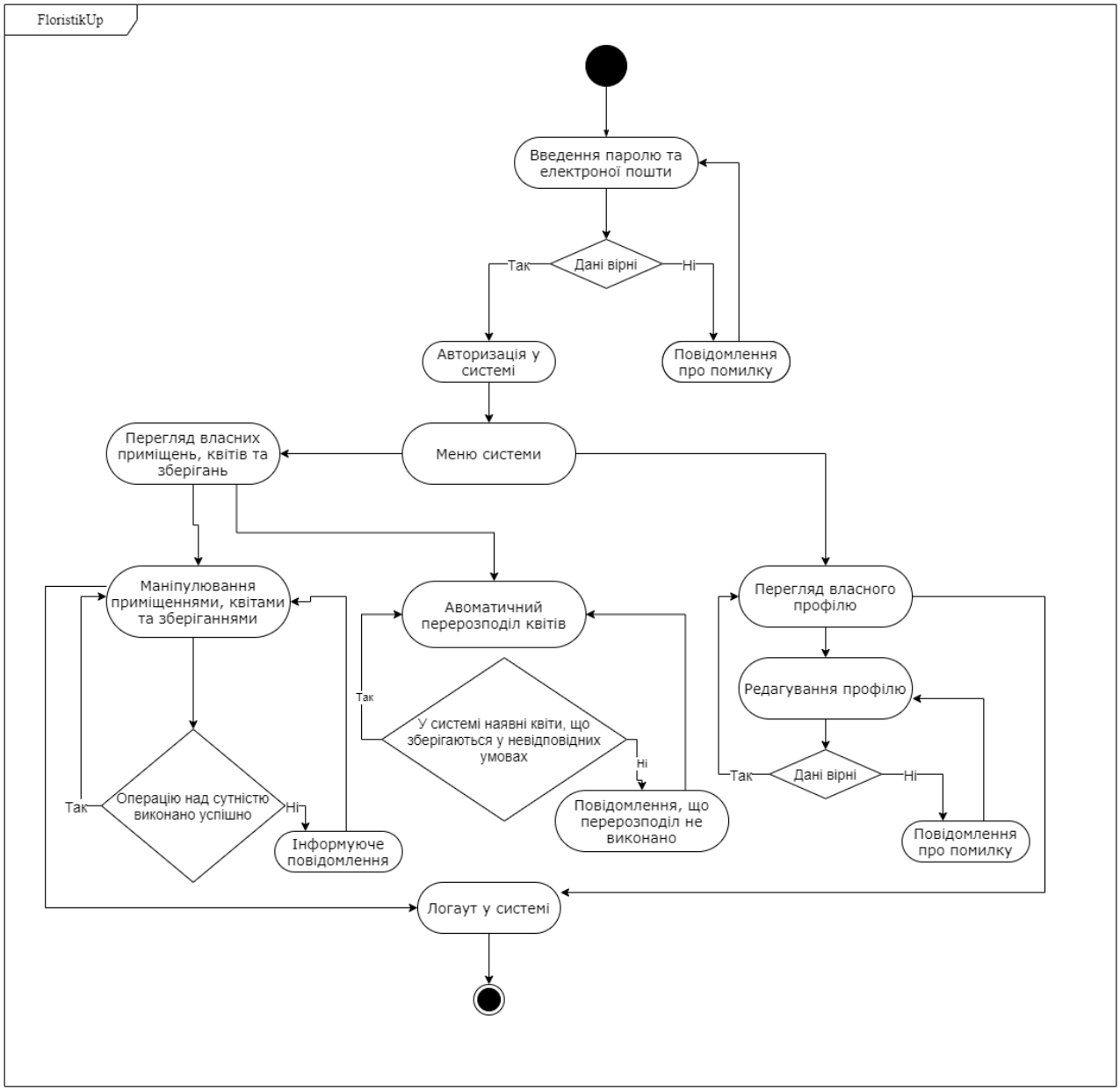

Рисунок Б.1 – Діаграма діяльності для програмної системи «FloristikUp»

ДОДАТОК В

Програмна реалізація компоненту маніпулювання зберіганнями квітів

```
class PickStorageCard extends React.Component {

  constructor(props) {
    super(props);
    this.state = {
      error: null,
      isLoaded: false,
      storages: []
    };

    Date.prototype.addDays = function (days) {
      var date = new Date(this.valueOf());
      date.setDate(date.getDate() + days);
      return date;
    }
  }

  componentDidMount() {
    const res =
    `${url}/flower-storages/storage-room/${localStorage.getItem('RId')}`;
    fetch(res, {
      method: "get",
      headers: {
        Accept: "application/json",
        "Content-Type": "application/json",
        Authorization: "Bearer " + token,
      },
    })
      .then((res) => res.json())
      .then(
        (result) => {
          this.setState({
            isLoaded: true,
            storages: result,
          });
        },
        (error) => {
          this.setState({
            isLoaded: true,
            error,
          });
        }
      );
  }

  render() {
    const {t} = this.props;

    const {error, isLoaded, storages} = this.state;
    if (error) {
      return (
        <div className="additional">
          {t("Failiture")}: {error.message}
```

```
54            </div>
55          );
56        } else if (!isLoaded) {
57          return <div className="centered">
58            <Loader
59              type="BallTriangle"
60              color="seagreen"
61              height={400}
62              width={400}
63              timeout={30000}
64            />
65          </div>;
66        } else if (storages.length === 0) {
67          return <div className="centered">
68            <h1>{t("NoStorages")}</h1>
69          </div>
70        } else {
71          const storage = storages[0]
72
73          return <div>
74            <div className="rooms_back">
75              <p>
76                {t("RoomNumber")} {localStorage.getItem('RId')} &#124;
77                 {t("City")} {storage.city}, {t("Street")}
78                {storage.street}, {t("House")} {storage.house} &#124;
79                 {t("Fullness")}
80                {storage.actualCapacity}/{storage.maxCapacity} &#124;
81                 {t("Temp")}: {storage.temperature}&deg;C
82                / {t("Hum")}: {storage.humidity}%
83              </p>
84              <Button
85                text={t("redistribution")}
86                onClick={(e) => {
87                  this.submitRedistribution(storage)
88                }}
89              />
90              <Button
91                text={t("AddStorage")}
92                onClick={(e) => {
93                  localStorage.setItem("RId", storage.storageRoomId);
94                  localStorage.setItem("Address",
95                  storage.city + ', ' + storage.street + ' ' + storage.house)
96                  window.location.href = "./add_storage";
97                }}
98              />
99            </div>
100           <div className="grid">{storages.sort((a, b) => {
101             return (a.formattedDate > b.formattedDate) ? -1
102               : ((b.formattedDate > a.formattedDate) ? 1 : 0)
103           }).map(this.renderCard)}</div>
104         </div>;
105       }
106     }
107
108     renderCard = (storage) => {
109       const {t} = this.props;
110
111       const startDate = new Date(Date.parse(storage.startDate));
```

```
112       const lastStorageDate = Date.parse(
113         startDate.addDays(
114           Number.parseInt(storage.flowerShelfLife)).toDateString())
115       const actualDate = Date.parse(new Date().toDateString())
116
117       return (
118         <div className="card text-center">
119           <div className="crd-body text-dark" id={storage.id}>
120             <h2 className="card-title">{storage.flowerName}</h2>
121             <hr/>
122             {actualDate > lastStorageDate ?
123               <p className="card-text text-secondary text-danger">
124                 {t("ExpiredShelfLife")}
125               </p> :
126               <p className="card-text text-secondary text-success">
127                 {t("NormalShelfLife")}
128               </p>
129             }
130             {(storage.temperature < storage.minTemperature ||
131               storage.temperature > storage.maxTemperature) ?
132               <p className="card-text text-secondary text-danger">
133                 {t("AbnormalClimate")}
134               </p> :
135               <p className="card-text text-secondary text-success">
136                 {t("NormalClimate")}
137               </p>
138             }
139             <p className="card-text text-secondary">
140               {t("FColor")}: {storage.flowerColor}
141             </p>
142             <p className="card-text text-secondary">
143               {t("StartDate")}: {storage.formattedDate}
144             </p>
145             <p className="card-text text-secondary">
146               {t("FShelfLife")}: {storage.flowerShelfLife}
147             </p>
148             <p className="card-text text-secondary">
149               {t("Amount")}: {storage.amount}
150             </p>
151             <p className="card-text text-secondary">
152               {t("TempInterval")}:
153               {storage.minTemperature}-{storage.maxTemperature}&deg;C
154             </p>
155             <Button
156               text={t("Edit")}
157               onClick={(e) => {
158                 localStorage.setItem("SId", storage.id);
159                 window.location.href = "./edit_storage";
160               }}
161             />
162             <Button
163               text={t("Delete")}
164               onClick={() => this.submitDelete(storage.id)}
165             />
166
167           </div>
168         </div>
169       );
```

```
170     }
171
172     submitDelete = (storageId) => {
173       const {t} = this.props;
174
175       confirmAlert({
176         title: t("Delete"),
177         message: t("areYouSure"),
178         buttons: [
179           {
180             label: t("yes"),
181             onClick: () => this.deleteStorage(storageId)
182           },
183           {
184             label: t("no")
185           }
186         ],
187         closeOnEscape: true,
188         closeOnClickOutside: true,
189       });
190     };
191
192     deleteStorage(id) {
193       const {t} = this.props;
194       fetch(`${url}/flower-storages/${id}`, {
195         method: "delete",
196         headers: {
197           Accept: "application/json",
198           "Content-Type": "application/json",
199           Authorization: "Bearer " + localStorage.getItem("Token"),
200         },
201       }).then(
202         (result) => {
203           this.setState({
204             storages: this.state.storages.filter(storage => {
205                 if (storage.id === id) {
206                 let actualCapacity = localStorage.getItem('actualCapacity')
207                   localStorage.setItem('actualCapacity',
208                     (actualCapacity - storage.amount).toString())
209                 }
210                 return storage.id !== id
211               }
212             )
213           });
214           window.location.reload();
215         },
216         (error) => {
217           this.setState({
218             isLoaded: true,
219             error,
220           });
221         }
222       );
223     }
224
225     submitRedistribution = (storage) => {
226       const {t} = this.props;
227
```

```
228       confirmAlert({
229         title: t("redistribution"),
230         message: t("areYouSureRedistribute"),
231         buttons: [
232           {
233             label: t("yes"),
234             onClick: () => this.redistribute(storage)
235           },
236           {
237             label: t("no")
238           }
239         ],
240         closeOnEscape: true,
241         closeOnClickOutside: true
242       });
243     };
244
245     redistribute(storage) {
246       const {t} = this.props;
247       this.setState({isLoaded: false})
248       fetch(`${url}/device`, {
249         method: "post",
250         headers: {
251           Accept: "application/json",
252           "Content-Type": "application/json",
253           Authorization: "Bearer " + localStorage.getItem("Token"),
254         },
255         body: JSON.stringify({
256           id: storage.storageRoomId,
257           airQuality: storage.airQuality,
258           humidity: storage.humidity,
259           temperature: storage.temperature,
260           satisfactionFactor: storage.satisfactionFactor
261         })
262       }).then((res) => res.json())
263         .then(result => {
264             console.log(result)
265             confirmAlert({
266               title: t("redistribution"),
267               message: this.createRedistributionMessage(result),
268               buttons: [
269                 {
270                   label: "Ok",
271                   onClick: () => window.location.reload()
272                 }
273               ],
274               closeOnEscape: false,
275               closeOnClickOutside: false
276             });
277           },
278           (error) => {
279             console.log(error)
280           }
281         );
282     }
283
284     createRedistributionMessage(json) {
285       const {t} = this.props;
```

```
286
287       let resultMessage = ""
288       let flower, room
289
290       json.forEach(storage => {
291         flower = storage.flower
292         room = storage.storageRoom
293         resultMessage += `${t("flower")} ${flower.name}
294         (${flower.color}) ${t("inCount")} ${storage.amount}
295         ${t("movedTo")} ${room.id} (${room.city},
296         ${room.street} ${room.house}).\r\n`
297       })
298
299       if (resultMessage === "") {
300         resultMessage = t("noRedistributionPerformed")
301       }
302
303       return resultMessage
304     }
305   }
```