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. Архітектура побудована на базі моделі MVC (Model-View-Controller). 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, яка бажає маніпулювати своїми приміщеннями зберігання квітів, власне квітами, їх зберіганнями у приміщеннях, 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.

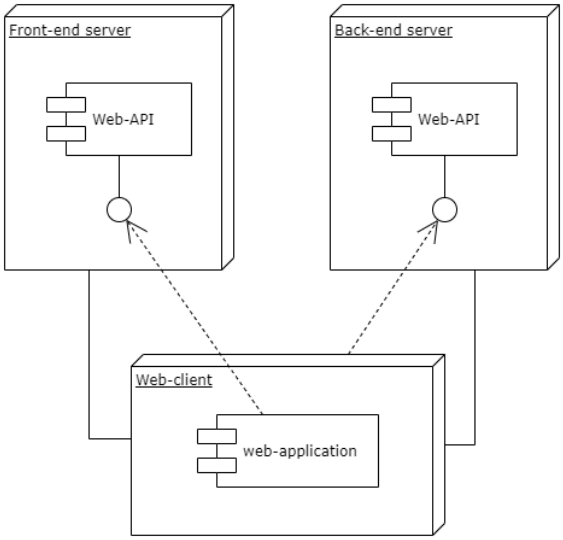

Рисунок 1 – Діаграма компонентів

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

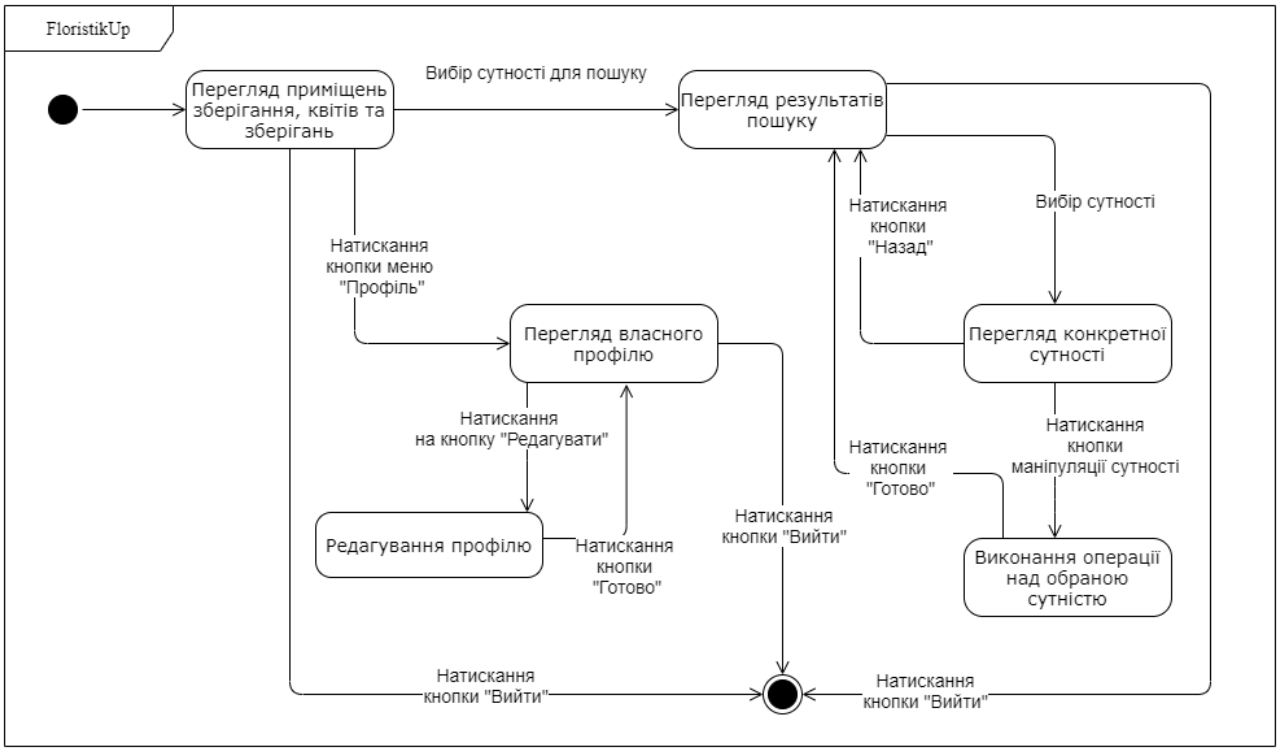

Рисунок 2 – Діаграма станів та переходів

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 квітами.

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

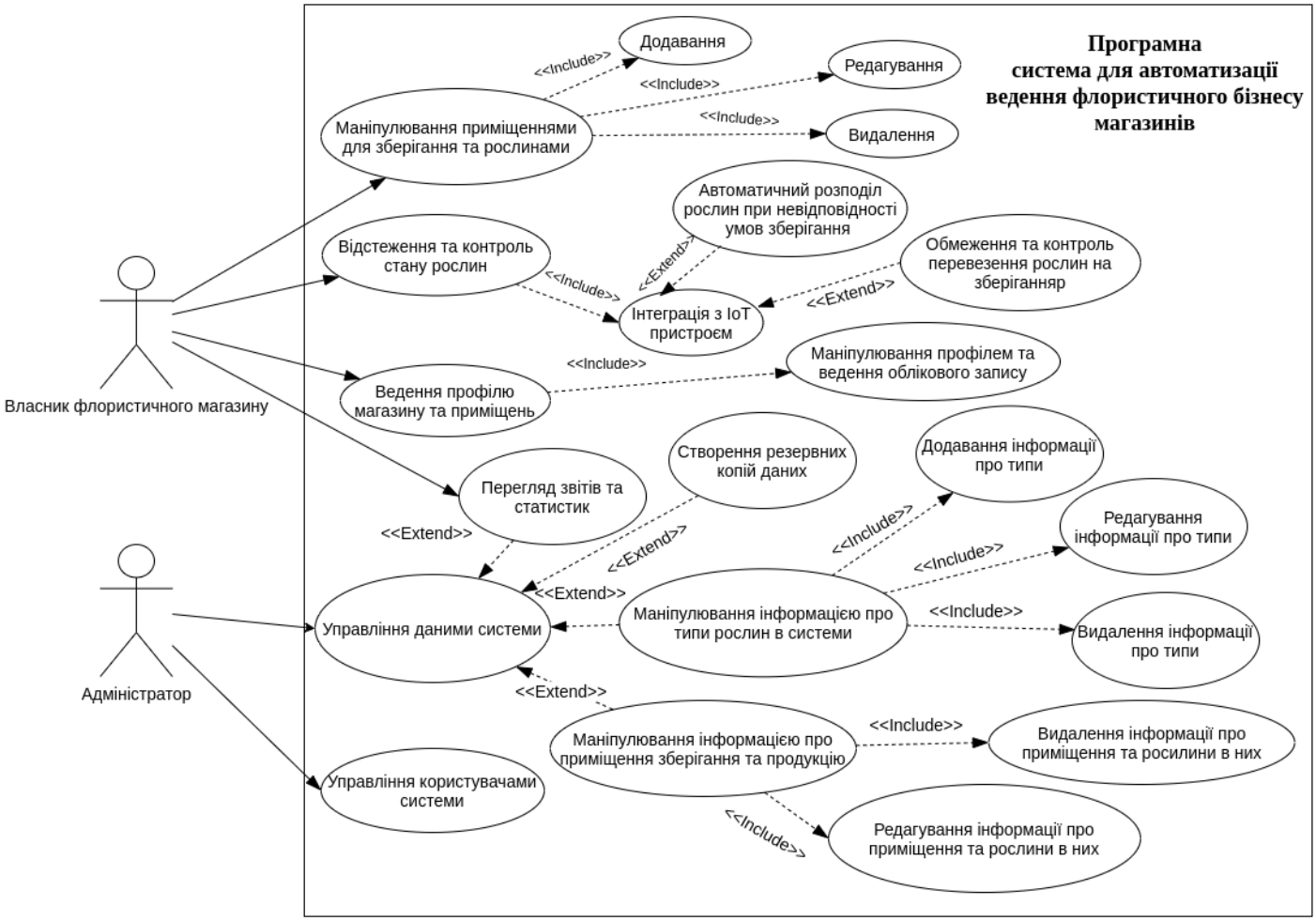

Рисунок А.1 – Діаграма варіантів використання для програмної системи «FloristikUp»

ДОДАТОК Б.
Діаграма діяльності системи

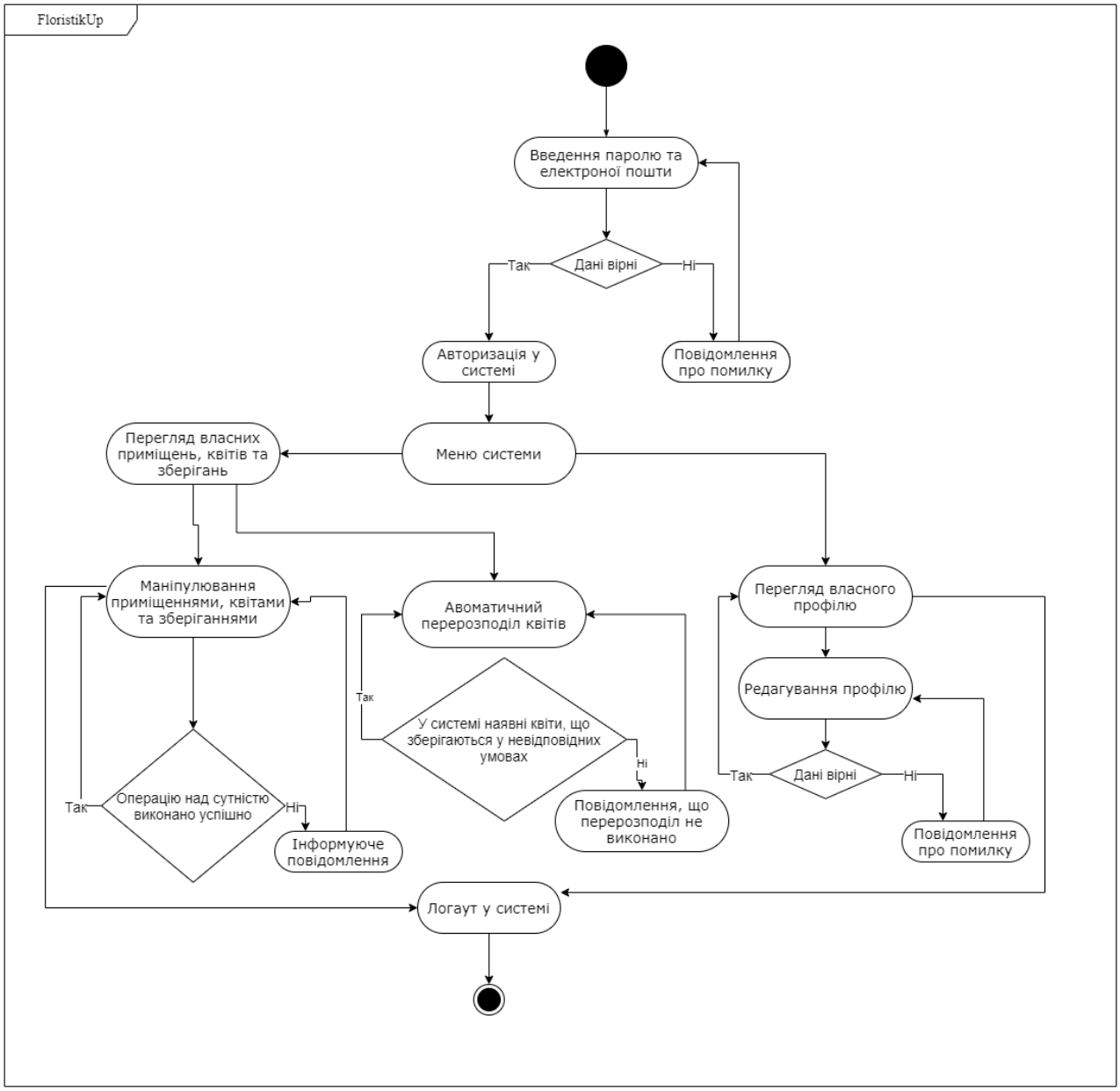

Рисунок Б.1 – Діаграма діяльності для програмної системи «FloristikUp»

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}
        </div>
      );
```

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

```
        Number.parseInt(storage.flowerShelfLife)).toDateString())
    const actualDate = Date.parse(new Date().toDateString())

    return (
      <div className="card text-center">
        <div className="crd-body text-dark" id={storage.id}>
          <h2 className="card-title">{storage.flowerName}</h2>
          <hr/>
          {actualDate > lastStorageDate ?
            <p className="card-text text-secondary text-danger">
              {t("ExpiredShelfLife")}
            </p> :
            <p className="card-text text-secondary text-success">
              {t("NormalShelfLife")}
            </p>
          }
          {(storage.temperature < storage.minTemperature ||
            storage.temperature > storage.maxTemperature) ?
            <p className="card-text text-secondary text-danger">
              {t("AbnormalClimate")}
            </p> :
            <p className="card-text text-secondary text-success">
              {t("NormalClimate")}
            </p>
          }
          <p className="card-text text-secondary">
            {t("FColor")}: {storage.flowerColor}
          </p>
          <p className="card-text text-secondary">
            {t("StartDate")}: {storage.formattedDate}
          </p>
          <p className="card-text text-secondary">
            {t("FShelfLife")}: {storage.flowerShelfLife}
          </p>
          <p className="card-text text-secondary">
            {t("Amount")}: {storage.amount}
          </p>
          <p className="card-text text-secondary">
            {t("TempInterval")}:
            {storage.minTemperature}-{storage.maxTemperature}&deg;C
          </p>
          <Button
            text={t("Edit")}
            onClick={(e) => {
              localStorage.setItem("SId", storage.id);
              window.location.href = "./edit_storage";
            }}
          />
          <Button
            text={t("Delete")}
            onClick={() => this.submitDelete(storage.id)}
          />

        </div>
      </div>
    );
  }

```

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

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

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