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.

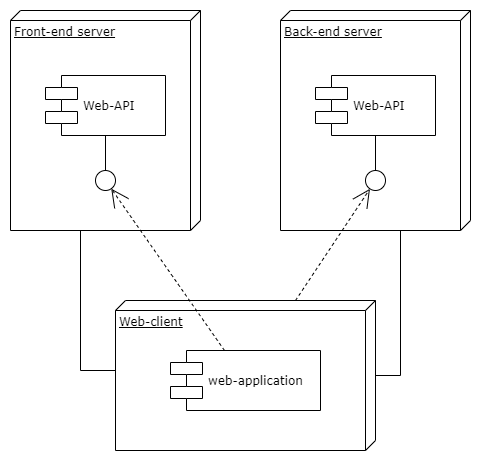


Рисунок 1 – Діаграма компонентів

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

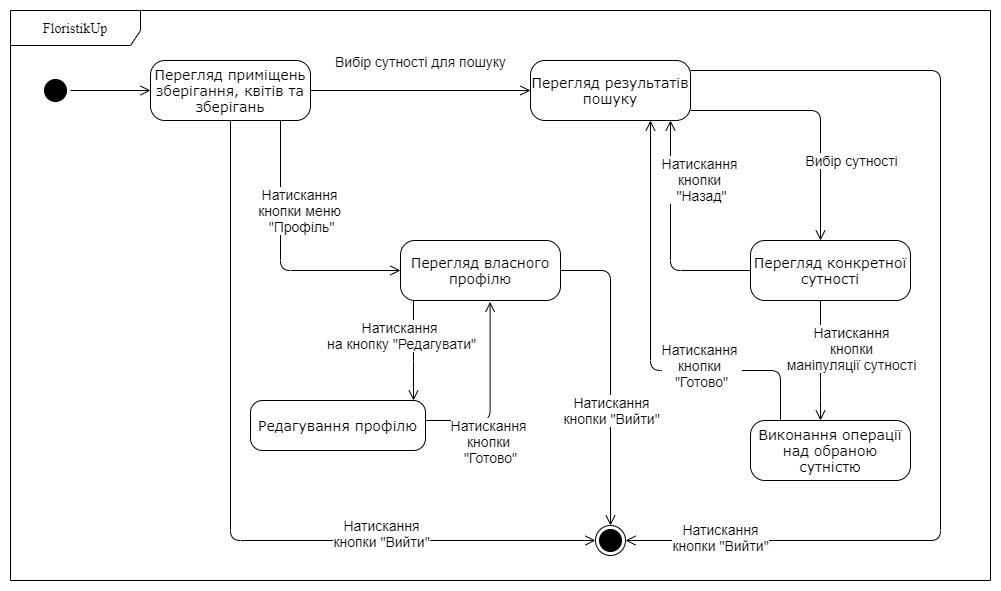


Рисунок 2 – Діаграма станів та переходів

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

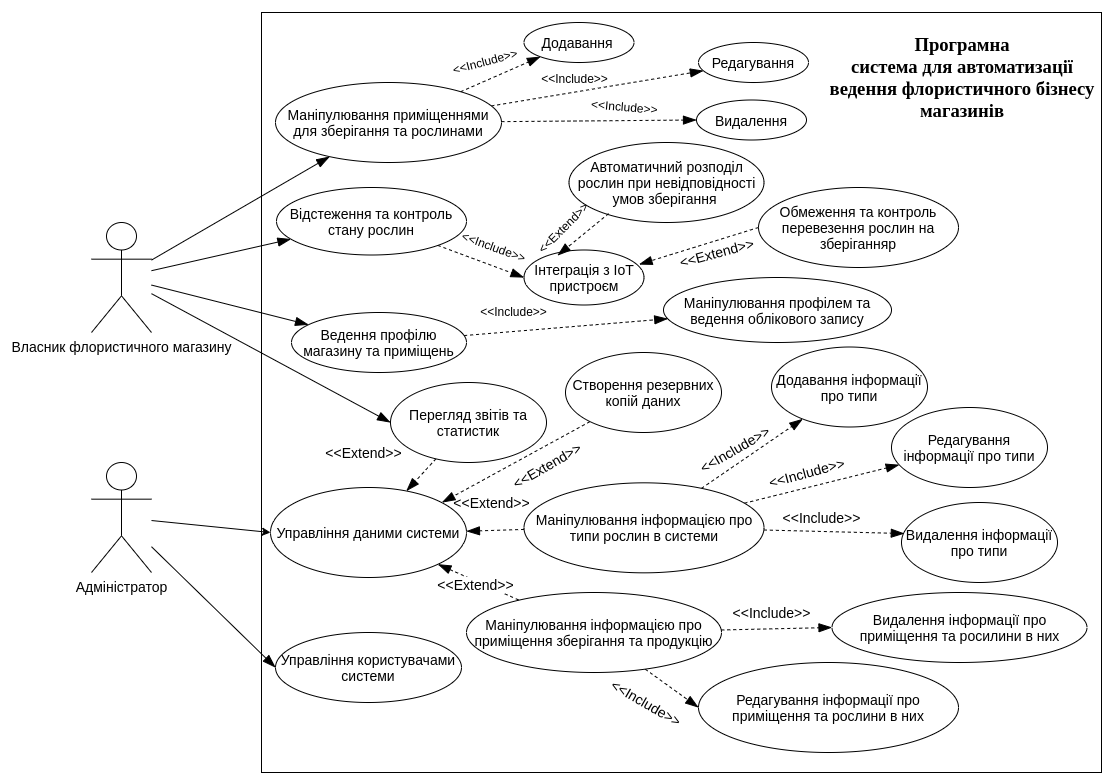


Рисунок А.1 – Діаграма варіантів використання для програмної системи «FloristikUp»

ДОДАТОК Б.

Діаграма діяльності системи

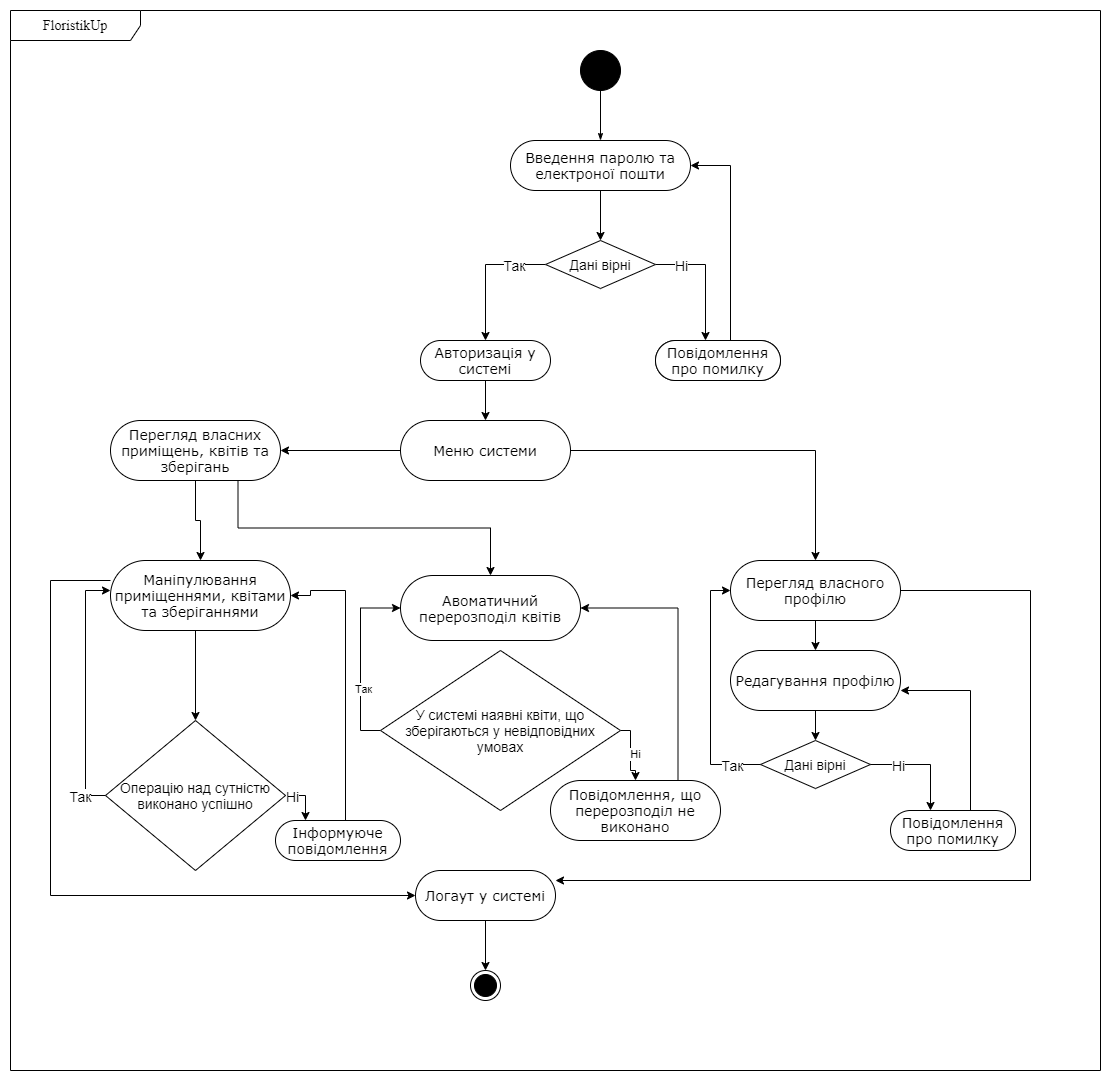


Рисунок Б.1 – Діаграма діяльності для програмної системи «FloristikUp»

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>** |
|  | **);** |
|  | **} else if (!isLoaded) {** |
|  | **return <div className="centered">** |
|  | **<Loader** |
|  | **type="BallTriangle"** |
|  | **color="seagreen"** |
|  | **height={400}** |
|  | **width={400}** |
|  | **timeout={30000}** |
|  | **/>** |
|  | **</div>;** |
|  | **} else if (storages.length === 0) {** |
|  | **return <div className="centered">** |
|  | **<h1>{t("NoStorages")}</h1>** |
|  | **</div>** |
|  | **} else {** |
|  | **const storage = storages[0]** |
|  |  |
|  | **return <div>** |
|  | **<div className="rooms\_back">** |
|  | **<p>** |
|  | **{t("RoomNumber")} {localStorage.getItem('RId')} &#124;** |
|  | **&nbsp;{t("City")} {storage.city}, {t("Street")}** |
|  | **{storage.street}, {t("House")} {storage.house} &#124;** |
|  | **&nbsp;{t("Fullness")}** |
|  | **{storage.actualCapacity}/{storage.maxCapacity} &#124;** |
|  | **&nbsp;{t("Temp")}: {storage.temperature}&deg;C** |
|  | **/ {t("Hum")}: {storage.humidity}%** |
|  | **</p>** |
|  | **<Button** |
|  | **text={t("redistribution")}** |
|  | **onClick={(e) => {** |
|  | **this.submitRedistribution(storage)** |
|  | **}}** |
|  | **/>** |
|  | **<Button** |
|  | **text={t("AddStorage")}** |
|  | **onClick={(e) => {** |
|  | **localStorage.setItem("RId", storage.storageRoomId);** |
|  | **localStorage.setItem("Address",** |
|  | **storage.city + ', ' + storage.street + ' ' + storage.house)** |
|  | **window.location.href = "./add\_storage";** |
|  | **}}** |
|  | **/>** |
|  | **</div>** |
|  | **<div className="grid">{storages.sort((a, b) => {** |
|  | **return (a.formattedDate > b.formattedDate) ? -1** |
|  | **: ((b.formattedDate > a.formattedDate) ? 1 : 0)** |
|  | **}).map(this.renderCard)}</div>** |
|  | **</div>;** |
|  | **}** |
|  | **}** |
|  |  |
|  | **renderCard = (storage) => {** |
|  | **const {t} = this.props;** |
|  |  |
|  | **const startDate = new Date(Date.parse(storage.startDate));** |
|  | **const lastStorageDate = Date.parse(** |
|  | **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>** |
|  | **);** |
|  | **}** |
|  |  |
|  | **submitDelete = (storageId) => {** |
|  | **const {t} = this.props;** |
|  |  |
|  | **confirmAlert({** |
|  | **title: t("Delete"),** |
|  | **message: t("areYouSure"),** |
|  | **buttons: [** |
|  | **{** |
|  | **label: t("yes"),** |
|  | **onClick: () => this.deleteStorage(storageId)** |
|  | **},** |
|  | **{** |
|  | **label: t("no")** |
|  | **}** |
|  | **],** |
|  | **closeOnEscape: true,** |
|  | **closeOnClickOutside: true,** |
|  | **});** |
|  | **};** |
|  |  |
|  | **deleteStorage(id) {** |
|  | **const {t} = this.props;** |
|  | **fetch(`${url}/flower-storages/${id}`, {** |
|  | **method: "delete",** |
|  | **headers: {** |
|  | **Accept: "application/json",** |
|  | **"Content-Type": "application/json",** |
|  | **Authorization: "Bearer " + localStorage.getItem("Token"),** |
|  | **},** |
|  | **}).then(** |
|  | **(result) => {** |
|  | **this.setState({** |
|  | **storages: this.state.storages.filter(storage => {** |
|  | **if (storage.id === id) {** |
|  | **let actualCapacity = localStorage.getItem('actualCapacity')** |
|  | **localStorage.setItem('actualCapacity',** |
|  | **(actualCapacity - storage.amount).toString())** |
|  | **}** |
|  | **return storage.id !== id** |
|  | **}** |
|  | **)** |
|  | **});** |
|  | **window.location.reload();** |
|  | **},** |
|  | **(error) => {** |
|  | **this.setState({** |
|  | **isLoaded: true,** |
|  | **error,** |
|  | **});** |
|  | **}** |
|  | **);** |
|  | **}** |
|  |  |
|  | **submitRedistribution = (storage) => {** |
|  | **const {t} = this.props;** |
|  |  |
|  | **confirmAlert({** |
|  | **title: t("redistribution"),** |
|  | **message: t("areYouSureRedistribute"),** |
|  | **buttons: [** |
|  | **{** |
|  | **label: t("yes"),** |
|  | **onClick: () => this.redistribute(storage)** |
|  | **},** |
|  | **{** |
|  | **label: t("no")** |
|  | **}** |
|  | **],** |
|  | **closeOnEscape: true,** |
|  | **closeOnClickOutside: true** |
|  | **});** |
|  | **};** |
|  |  |
|  | **redistribute(storage) {** |
|  | **const {t} = this.props;** |
|  | **this.setState({isLoaded: false})** |
|  | **fetch(`${url}/device`, {** |
|  | **method: "post",** |
|  | **headers: {** |
|  | **Accept: "application/json",** |
|  | **"Content-Type": "application/json",** |
|  | **Authorization: "Bearer " + localStorage.getItem("Token"),** |
|  | **},** |
|  | **body: JSON.stringify({** |
|  | **id: storage.storageRoomId,** |
|  | **airQuality: storage.airQuality,** |
|  | **humidity: storage.humidity,** |
|  | **temperature: storage.temperature,** |
|  | **satisfactionFactor: storage.satisfactionFactor** |
|  | **})** |
|  | **}).then((res) => res.json())** |
|  | **.then(result => {** |
|  | **console.log(result)** |
|  | **confirmAlert({** |
|  | **title: t("redistribution"),** |
|  | **message: this.createRedistributionMessage(result),** |
|  | **buttons: [** |
|  | **{** |
|  | **label: "Ok",** |
|  | **onClick: () => window.location.reload()** |
|  | **}** |
|  | **],** |
|  | **closeOnEscape: false,** |
|  | **closeOnClickOutside: false** |
|  | **});** |
|  | **},** |
|  | **(error) => {** |
|  | **console.log(error)** |
|  | **}** |
|  | **);** |
|  | **}** |
|  |  |
|  | **createRedistributionMessage(json) {** |
|  | **const {t} = this.props;** |
|  |  |
|  | **let resultMessage = ""** |
|  | **let flower, room** |
|  |  |
|  | **json.forEach(storage => {** |
|  | **flower = storage.flower** |
|  | **room = storage.storageRoom** |
|  | **resultMessage += `${t("flower")} ${flower.name}** |
|  | **(${flower.color}) ${t("inCount")} ${storage.amount}** |
|  | **${t("movedTo")} ${room.id} (${room.city},** |
|  | **${room.street} ${room.house}).\r\n`** |
|  | **})** |
|  |  |
|  | **if (resultMessage === "") {** |
|  | **resultMessage = t("noRedistributionPerformed")** |
|  | **}** |
|  |  |
|  | **return resultMessage** |
|  | **}** |
|  | **}** |