Міністерство освіти та науки України

Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

Лабораторна робота №2

З дисципліни: «Архітектура програмного забезпечення»

на тему: «Програмна система автоматизації ведення флористичного бізнесу магазинів»

       Виконала

ст. гр. ПЗПІ-18-5

       Борщова Олександра Вадимівна

          Перевірив

          ст. викл. каф. ПІ

       Сокорчук Ігор Петрович

Харків 2021

**Мета:** розробити серверну частину програмної системи, описати прийняті інженерні рішення, будову серверних компонентів, загальну структуру системи та структуру бази даних.

**Хід роботи:**

Каскад серверної частини проекту реалізований за допомогою фреймворку Spring Boot. Для написання контролерів, створення endpoints, реалізації REST API для взаємодії з клієнтами системи використовувався Spring MVC – фреймворк-реалізація шаблону проектування Model-View-Controller. Для доступу до даних та реалізації ORM використано фреймворк Spring Data JPA. Для захисту системи, реалізації рівнів доступу, JWT, авторизації використано фреймворк Spring Security. Система управління базами даних для проекту – MySQL.

Початком створення серверної частини системи було створення структури бази даних та моделювання зв’язків між сутностями. ER-діаграма для програмної системи зображена на рисунку 1.

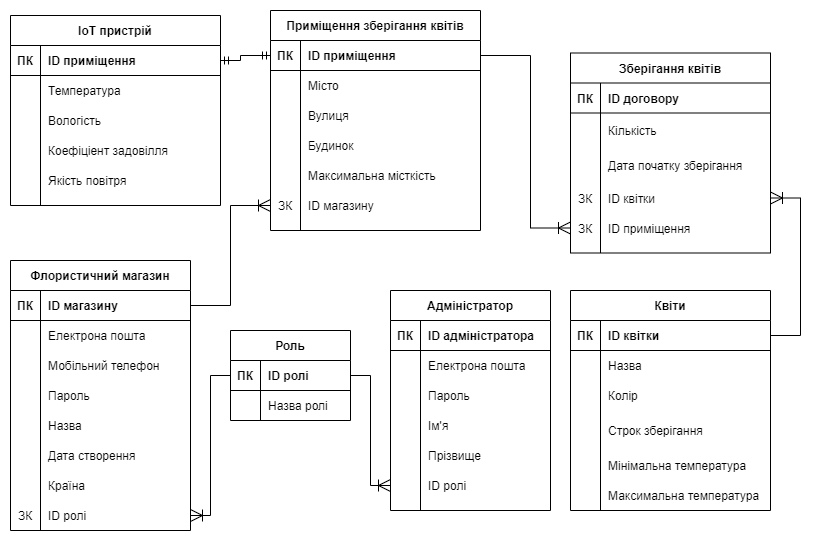


Рисунок 1 – ER-модель даних

Після моделювання структури бази даних було розпочато написання класів-сутностей у коді програмної системи с використанням JPA – Java Persistence Api, стандарту проектування ORM.

У підсумку отримано 7 класів сутностей, які відповідають таблицям у базі даних. Приклад програмної реалізації класу StorageRoom, який відповідає таблиці «Приміщення для зберігання»:

**@Entity**

**@Data**

**@NoArgsConstructor**

**@Accessors(chain = true)**

**public class StorageRoom {**

**@Id**

**@GeneratedValue(strategy = GenerationType.AUTO)**

**@Column(name = "storage\_room\_id")**

**private Long id;**

**@Column(name = "city")**

**private String city;**

**@Column(name = "street")**

**private String street;**

**@Column(name = "house")**

**private String house;**

**@Column(name = "max\_capacity")**

**private Integer maxCapacity;**

**@JsonIgnore**

**@ManyToOne(fetch = FetchType.EAGER)**

**@JoinColumn(name = "florist\_shop\_id")**

**protected FloristShop floristShop;**

**@OneToOne(mappedBy = "storageRoom", cascade = CascadeType.ALL)**

**@PrimaryKeyJoinColumn**

**private SmartSystem smartSystem;**

**@JsonIgnore**

**@OneToMany(mappedBy = "storageRoom", fetch = FetchType.EAGER)**

**Set<FlowerStorage> flowerStorages;**

**public void addStorage(FlowerStorage flowerStorage) {**

**flowerStorages.add(flowerStorage);**

**}**

**public void removeStorage(FlowerStorage flowerStorage) {**

**flowerStorages.remove(flowerStorage);**

**}**

**}**

Наступним кроком було створення інтерфейсів-репозиторіїв, які надають доступ до даних. Всі інтерфейси надають базовий доступ до даних та можливість написання власних методів вибірки даних для фільтрацій, статистик, пошуків тощо. Приклад програмної реалізації інтерфейсу FloristShopRepository:

**@Repository**

**public interface FloristShopRepository extends CrudRepository<FloristShop, Long> {**

**Optional<FloristShop> findByPhoneNumber(String phoneNumber);**

**Optional<FloristShop> findByEmail(String email);**

**List<FloristShop> findAll();**

**}**

Сукупність цих інтерфейсів являє собою перший шар серверного рівню системи – шар доступу до даних. В ньому зосереджена лише логіка роботи з базою даних.

Усі ці інтерфейси розташовані у пакеті «ua.nure.repository» та являють собою перший і найнижчий шар системи – шар доступу до даних. В ньому зосереджена лише логіка роботи з базою даних, збереження, редагування, видалення даних, надсилання до бази складних запитів на вибірку тощо. Ця логіка відокремлена від бізнес-логіки, що не перешкоджає масштабуванню та розширенню системи.

Далі було реалізовано сервісний шар системи – шар, у якому реалізується бізнес-логіка програми, а також використовуються класи, які знаходяться у рівні доступу до даних. Таке розподілення логіки доступу до даних від бізнес-логіки робить систему більш гнучкою для доповнення та масштабування. Класи-сервіси системи знаходяться у пакеті «ua.nure.service». Програмна реалізація логіки створення запиту на переміщення квітів на зберігання до приміщення наведена у додатку А. Програмна реалізація логіки автоматичного розподілу квітів до приміщень з задовільним для них мікрокліматом та місткістю наведена у додатку Б.

Наступний, останній шар розподілення програми - це шар контролерів. Класи-контролери приймають HTTP запити від клієнтів, оброблюють їх, викликають методи сервісного рівня, оброблюють помилки на рівні HTTP протоколу та надсилають клієнтам відповідь сервера. За допомогою класів контролерів було реалізовано мережеве REST API для взаємодії клієнтів, а саме веб-клієнта, мобільного застосунку та IoT пристрою з серверною частиною системи.

Кожен запит, окрім POST запитів на URL /auth/login та /auth/register/\*, надсилається із заголовком Authorization, у якому зберігається унікальний токен користувача, реалізований за допомогою JWT. Таким чином система стає більш захищеною і відповідає сучасним стандартам щодо захисту даних.

Отже, на основі реалізованих трьох рівнів системи, було створено діаграму компонентів та діаграму розгортання, які наведені у додатках В та Г відповідно.

На діаграмі компонентів вказані усі компоненти трьох шарів системи, компонент бази даних та компонент JWT Request Filter, який відносить до логіки захисту даних. Ці ж самі компоненти системи зображені в серверному вузлу на діаграмі розгортання, яка демонструє загальну архітектуру системи.

Після написання логіки та розподілення її на три шари: шар доступу до даних, шар бізнес-логіки та шар REST контролерів, було запроваджено захист системи за допомогою фреймворку Spring Security. Було реалізовано логіку створення, перевірки, збереження JWT токена. JWT токен зберігає у собі електрону пошту клієнта, яка є головним ідентифікатором, та права клієнта. На основі цих даних відбувається і аутентифікація, і авторизація з подальшим відкриттям доступу до захищених URL та закриттям доступу до тих URL, рівень доступу яких інакший від рівня доступу цього користувача. Конфігурація безпеки знаходиться у пакеті «ua.nure.config», інші класи, що надають безпеку систему – у пакеті «ua.nure.security».

У структурі проекту також наявні наступні пакети – «ua.nure.dto» та «ua.nure.validation». Перший пакет зберігає в собі DTO (Data Transfer Object) – об’єкти передачі даних, які приймаються у тілі запиту від клієнтів або надсилаються до них у тілі відповіді сервера. Зроблено це для того, щоб не передавати клієнтам весь об’єкт, який моделює сутність в БД, а лише ті дані, які потрібно.

У другому пакеті зосереджена логіка перевірки коректності надісланих від клієнтів даних. Ця перевірка могла б бути реалізована лише клієнтами, але вона реалізована і на стороні сервера для збільшення безпеки. Таким чином стають неможливими ситуації, коли до бази даних заносяться неправильні, небезпечні дані.

Після реалізації серверної частини проекту була створена загальна діаграма варіантів використання, що наведена у додатку Д. Cпецифікація REST у форматі OpenAPI Spacification (OAS) наведена у додатку Е.

Посилання на архів з програмним кодом та файлом контрольної суми:

<https://drive.google.com/drive/folders/1xpystslOeEiSFyRt_nHJ14FOfaNK7L2Y?usp=sharing>

Контрольна сума до архіву: 935d373221e39f3737cd3d6e9c0b6338

**Висновок:** у ході лабораторної роботи було створено серверну частину програмної системи. Було описано та обґрунтовано прийняті інженерні рішення, будову серверних компонентів, загальну структуру системи та структуру бази даних. Також у ході виконання було створено наступні діаграми: UML діаграму розгортання (Deployment Diagram), UML діаграму прецедентів (Use Case Diagram), ER-модель даних (Entity–Relationship Model) та UML діаграму компонентів (Component Diagram

ДОДАТОК А.

Програмна реалізація переміщення квітів на зберігання

|  |  |
| --- | --- |
|  | **@Override** |
|  | **public FlowerStorageResponseDto create(** |
|  | **FlowerStorageRequestDto flowerStorageRequestDto) throws Exception {** |
|  | **FlowerStorage flowerStorage = new FlowerStorage();** |
|  | **flowerStorage.setStartDate(new Date());** |
|  |  |
|  | **Optional<StorageRoom> roomById = storageRoomRepository** |
|  | **.findById(flowerStorageRequestDto.getStorageRoomId());** |
|  | **Optional<Flower> flowerById = flowerRepository** |
|  | **.findById(flowerStorageRequestDto.getFlowerId());** |
|  |  |
|  | **if (roomById.isPresent() && flowerById.isPresent()) {** |
|  | **StorageRoom storageRoom = roomById.get();** |
|  | **Flower flower = flowerById.get();** |
|  |  |
|  | **flowerStorage.setStorageRoom(storageRoom);** |
|  | **flowerStorage.setFlower(flower);** |
|  | **flowerStorage.setAmount(flowerStorageRequestDto.getAmount());** |
|  |  |
|  | **SmartSystem smartSystem = storageRoom.getSmartSystem();** |
|  |  |
|  | **String cause = "";** |
|  |  |
|  | **Double actualTemperature = smartSystem.getTemperature();** |
|  | **Integer minTemperature = flower.getMinTemperature();** |
|  | **Integer maxTemperature = flower.getMaxTemperature();** |
|  |  |
|  | **if (actualTemperature < minTemperature** |
|  | **|| actualTemperature > maxTemperature) {** |
|  | **cause = "Незадовільний мікроклімат для зберігання. "** |
|  | **+ "Температура в приміщенні: " + actualTemperature** |
|  | **+ " °С, діапазон зберігання квітки: " + minTemperature** |
|  | **+ "-" + maxTemperature + "°С.";** |
|  | **}** |
|  | **int actualAmount = storageRoom.getFlowerStorages().stream()** |
|  | **.mapToInt(FlowerStorage::getAmount).sum();** |
|  | **int newAmount = actualAmount + flowerStorage.getAmount();** |
|  | **int maxAmount = storageRoom.getMaxCapacity();** |
|  | **if (newAmount > maxAmount) {** |
|  | **cause = "Недостатня місткість приміщення. " + "Максимальна: "** |
|  | **+ maxAmount + ", теперішня: " + actualAmount** |
|  | **+ ", очікувана: " + newAmount;** |
|  | **}** |
|  | **if (!cause.isEmpty())** |
|  | **throw new Exception(cause);** |
|  | **return FlowerStorageMapper.toFlowerStorageResponseDto(** |
|  | **flowerStorageRepository.save(flowerStorage));** |
|  | **}** |
|  | **return null;** |
|  | **}** |

ДОДАТОК Б.

Програмна реалізація автоматичного перерозподілу квітів до приміщень з необхідним мікрокліматом та місткістю

|  |  |
| --- | --- |
|  | **public List<FlowerStorage> updateSmartSystem(** |
|  | **SmartSystemDto smartDeviceDto) {** |
|  | **Optional<StorageRoom> storageRoomOptional = storageRoomRepository** |
|  | **.findById(smartDeviceDto.getId());** |
|  | **if (storageRoomOptional.isPresent()) {** |
|  | **StorageRoom storageRoom = storageRoomOptional.get();** |
|  | **SmartSystem smartDevice = storageRoom.getSmartSystem();** |
|  | **ArrayList<FlowerStorage> updatedStorages = new ArrayList<>();** |
|  |  |
|  | **storageRoom.getFlowerStorages().forEach(flowerStorage -> {** |
|  | **Flower flower = flowerStorage.getFlower();** |
|  |  |
|  | **Integer minTemperature = flower.getMinTemperature();** |
|  | **Integer maxTemperature = flower.getMaxTemperature();** |
|  | **Double currentTemperature = smartDeviceDto.getTemperature();** |
|  |  |
|  | **if (currentTemperature < minTemperature** |
|  | **|| currentTemperature > maxTemperature) {** |
|  | **Set<StorageRoom> storageRooms = storageRoom** |
|  | **.getFloristShop().getStorageRooms();** |
|  |  |
|  | **storageRooms.stream().filter(room -> {** |
|  | **Double temperature = room.getSmartSystem()** |
|  | **.getTemperature();** |
|  | **return temperature > minTemperature** |
|  | **&& temperature < maxTemperature;** |
|  | **}).filter(room -> {** |
|  | **int actualAmount = room.getFlowerStorages().stream()** |
|  | **.mapToInt(FlowerStorage::getAmount).sum();** |
|  | **int newAmount =** |
|  | **actualAmount + flowerStorage.getAmount();** |
|  | **int maxAmount = room.getMaxCapacity();** |
|  | **return newAmount < maxAmount;** |
|  | **}).findAny().ifPresent(newStorageRoom -> {** |
|  | **flowerStorage.setStorageRoom(newStorageRoom);** |
|  | **updatedStorages.add(flowerStorageRepository** |
|  | **.save(flowerStorage));** |
|  | **});** |
|  | **}** |
|  | **});** |
|  | **smartDevice.setAirQuality(smartDeviceDto.getAirQuality())** |
|  | **.setTemperature(smartDeviceDto.getTemperature())** |
|  | **.setHumidity(smartDeviceDto.getHumidity())** |
|  | **.setSatisfactionFactor(** |
|  | **smartDeviceDto.getSatisfactionFactor());** |
|  | **storageRoom.setSmartSystem(smartDevice);** |
|  | **return updatedStorages;** |
|  | **}** |
|  | **return null;** |
|  | **}** |

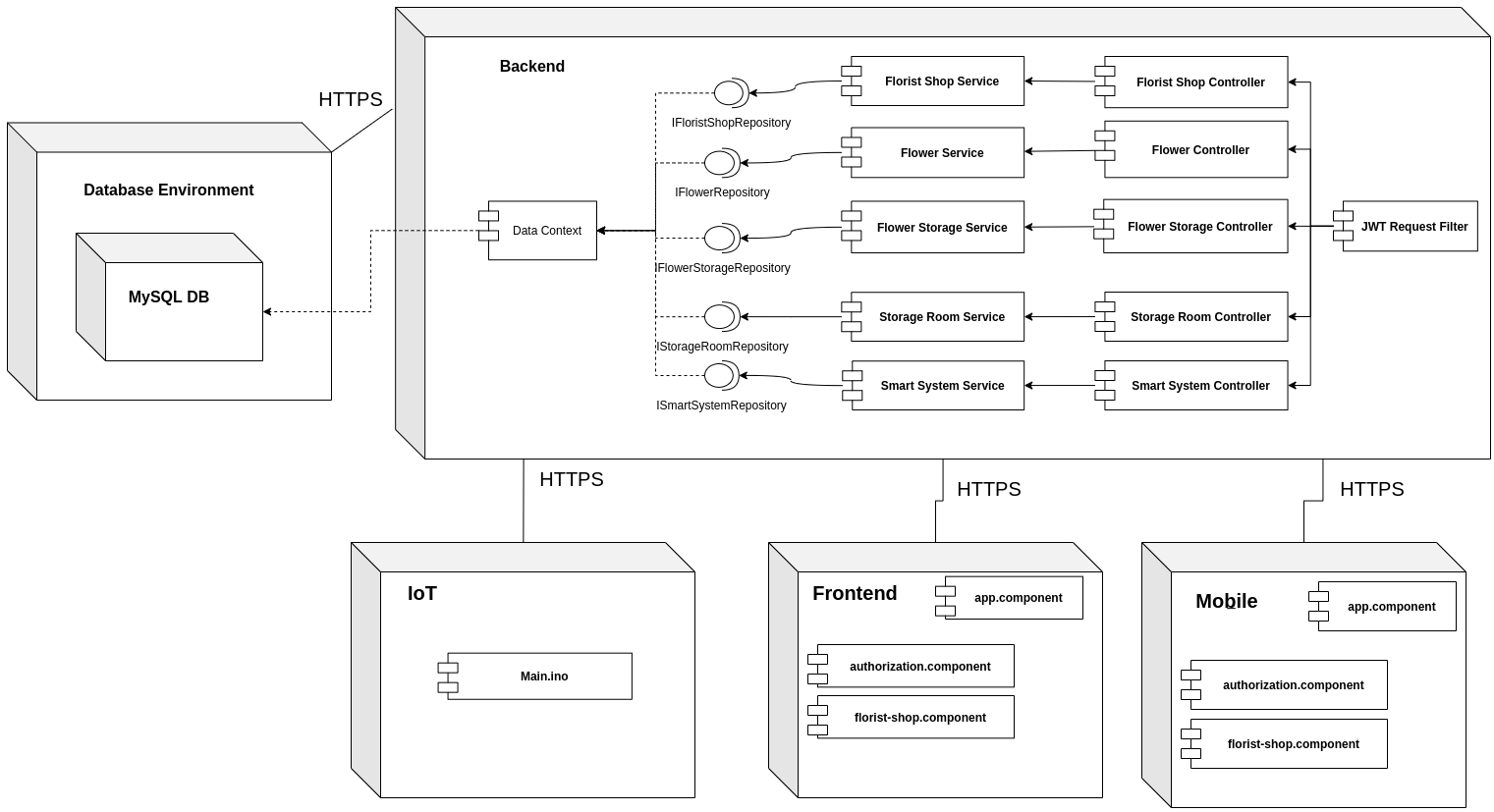
ДОДАТОК В.

Діаграма компонентів серверної частини системи



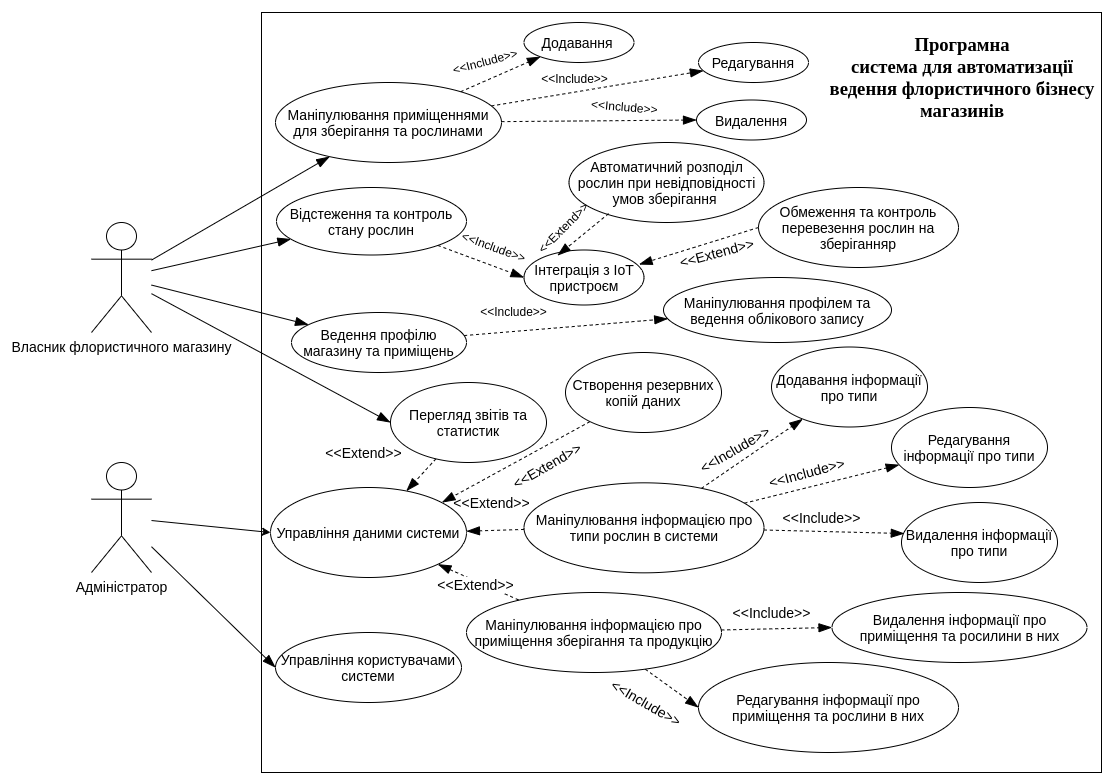
ДОДАТОК Г.

Діаграма розгортання системи



ДОДАТОК Д.

Діаграма варіантів використання системи



ДОДАТОК Е.

Специфікація REST у форматі Open API Specifiaction

**openapi: 3.0.0  
info:  
 title: FloristikUp system API  
 description: Documentation for FloristikUP REST API  
 contact: {}  
 version: '1.0.0'  
servers:  
- url: http://localhost:8080/  
 variables: {}  
paths:  
 /auth/login:  
 post:  
 tags:  
 - Authorization  
 summary: loginUser  
 description: Performs user login to the system  
 operationId: loginUser  
 parameters: []  
 requestBody:  
 description: loginDto  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/LoginDto'  
 required: true  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '201':  
 description: Created  
 headers: {}  
 content: {}  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 /auth/register/florist-shop:  
 post:  
 tags:  
 - Authorization  
 summary: registerFloristShop  
 description: Registers a florist shop  
 operationId: registerFloristShop  
 parameters: []  
 requestBody:  
 description: floristShopDto  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/FloristShopDto'  
 required: true  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '201':  
 description: Created  
 headers: {}  
 content: {}  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 /florist-shops:  
 get:  
 tags:  
 - Florist Shop  
 summary: getAllFloristShops  
 description: Returns a list of all florist shops  
 operationId: getAllFloristShops  
 parameters: []  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 post:  
 tags:  
 - Florist Shop  
 summary: addFloristShop  
 description: Adds new florist shop  
 operationId: addFloristShop  
 parameters: []  
 requestBody:  
 description: floristShopDto  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/FloristShopDto'  
 required: true  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '201':  
 description: Created  
 headers: {}  
 content: {}  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 put:  
 tags:  
 - Florist Shop  
 summary: updateFloristShop  
 description: Updates the florist shop  
 operationId: updateFloristShop  
 parameters: []  
 requestBody:  
 description: floristShopDto  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/FloristShopDto'  
 required: true  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '201':  
 description: Created  
 headers: {}  
 content: {}  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 /florist-shops/rooms/{id}:  
 get:  
 tags:  
 - Florist Shop  
 summary: getStorageRoomById  
 description: Finds storage room by id  
 operationId: getStorageRoomById  
 parameters:  
 - name: id  
 in: path  
 description: id  
 required: true  
 style: simple  
 schema:  
 type: integer  
 format: int64  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 delete:  
 tags:  
 - Florist Shop  
 summary: deleteStorageRoom  
 description: Deletes storage room by ID  
 operationId: deleteStorageRoom  
 parameters:  
 - name: id  
 in: path  
 description: id  
 required: true  
 style: simple  
 schema:  
 type: integer  
 format: int64  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content: {}  
 '204':  
 description: No Content  
 headers: {}  
 content: {}  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 /florist-shops/{email}:  
 get:  
 tags:  
 - Florist Shop  
 summary: getFloristShopByEmail  
 description: Finds florist shop by email  
 operationId: getFloristShopByEmail  
 parameters:  
 - name: email  
 in: path  
 description: email  
 required: true  
 style: simple  
 schema:  
 type: string  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 delete:  
 tags:  
 - Florist Shop  
 summary: deleteFloristShop  
 description: Deletes florist shop by email  
 operationId: deleteFloristShop  
 parameters:  
 - name: email  
 in: path  
 description: email  
 required: true  
 style: simple  
 schema:  
 type: string  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content: {}  
 '204':  
 description: No Content  
 headers: {}  
 content: {}  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 /florist-shops/{email}/rooms:  
 get:  
 tags:  
 - Florist Shop  
 summary: getAllStorageRooms  
 description: 'Returns all florist shop storage rooms '  
 operationId: getAllStorageRooms  
 parameters:  
 - name: email  
 in: path  
 description: email  
 required: true  
 style: simple  
 schema:  
 type: string  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 post:  
 tags:  
 - Florist Shop  
 summary: addStorageRoom  
 description: Adds new storage room for florist shop  
 operationId: addStorageRoom  
 parameters:  
 - name: email  
 in: path  
 description: email  
 required: true  
 style: simple  
 schema:  
 type: string  
 requestBody:  
 description: storageRoomDto  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/StorageRoomDto'  
 required: true  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '201':  
 description: Created  
 headers: {}  
 content: {}  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 put:  
 tags:  
 - Florist Shop  
 summary: updateStorageRoom  
 description: Updates storage room of florist shop (room id must be present)  
 operationId: updateStorageRoom  
 parameters:  
 - name: email  
 in: path  
 description: email  
 required: true  
 style: simple  
 schema:  
 type: string  
 requestBody:  
 description: storageRoomDto  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/StorageRoomDto'  
 required: true  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '201':  
 description: Created  
 headers: {}  
 content: {}  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 /flower-storages:  
 post:  
 tags:  
 - Flower Storage  
 summary: createStorage  
 description: Creates new flower storage in storage room  
 operationId: createStorage  
 parameters: []  
 requestBody:  
 description: flowerStorageRequestDto  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/FlowerStorageRequestDto'  
 required: true  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '201':  
 description: Created  
 headers: {}  
 content: {}  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 put:  
 tags:  
 - Flower Storage  
 summary: updateStorage  
 description: Updates storage (Flower Storage ID must be present, updates only amount)  
 operationId: updateStorage  
 parameters: []  
 requestBody:  
 description: flowerStorageRequestDto  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/FlowerStorageRequestDto'  
 required: true  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '201':  
 description: Created  
 headers: {}  
 content: {}  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 /flower-storages/{id}:  
 get:  
 tags:  
 - Flower Storage  
 summary: getFlowerStorageById  
 description: Finds flower storage by id  
 operationId: getFlowerStorageById  
 parameters:  
 - name: id  
 in: path  
 description: id  
 required: true  
 style: simple  
 schema:  
 type: integer  
 format: int64  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 delete:  
 tags:  
 - Flower Storage  
 summary: deleteStorage  
 description: Deletes storage by ID  
 operationId: deleteStorage  
 parameters:  
 - name: id  
 in: path  
 description: id  
 required: true  
 style: simple  
 schema:  
 type: integer  
 format: int64  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content: {}  
 '204':  
 description: No Content  
 headers: {}  
 content: {}  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 /flowers:  
 get:  
 tags:  
 - Flowers  
 summary: getAllFlowers  
 description: Returns a list of all flowers  
 operationId: getAllFlowers  
 parameters: []  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 post:  
 tags:  
 - Flowers  
 summary: addFlower  
 description: Adds new flower type  
 operationId: addFlower  
 parameters: []  
 requestBody:  
 description: FlowerDto  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/FlowerDto'  
 required: true  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '201':  
 description: Created  
 headers: {}  
 content: {}  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 put:  
 tags:  
 - Flowers  
 summary: updateFlower  
 description: Updates the flower type  
 operationId: updateFlower  
 parameters: []  
 requestBody:  
 description: FlowerDto  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/FlowerDto'  
 required: true  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '201':  
 description: Created  
 headers: {}  
 content: {}  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 /flowers/{id}:  
 get:  
 tags:  
 - Flowers  
 summary: getFlowerId  
 description: Finds flower by ID  
 operationId: getFlowerId  
 parameters:  
 - name: id  
 in: path  
 description: id  
 required: true  
 style: simple  
 schema:  
 type: integer  
 format: int64  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 delete:  
 tags:  
 - Flowers  
 summary: deleteFlower  
 description: Deletes flower type by ID  
 operationId: deleteFlower  
 parameters:  
 - name: id  
 in: path  
 description: id  
 required: true  
 style: simple  
 schema:  
 type: integer  
 format: int64  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content: {}  
 '204':  
 description: No Content  
 headers: {}  
 content: {}  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 /roles:  
 get:  
 tags:  
 - Role  
 summary: getAllRoles  
 description: Returns a list of all roles  
 operationId: getAllRoles  
 parameters: []  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
 /device:  
 post:  
 tags:  
 - Smart Device  
 summary: updateSmartDevice  
 description: Update smart device characteristics, endpoint for Arduino  
 operationId: updateSmartDevice  
 parameters: []  
 requestBody:  
 description: smartDeviceDto  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/SmartSystemDto'  
 required: true  
 responses:  
 '200':  
 description: OK  
 headers: {}  
 content:  
 '\*/\*':  
 schema:  
 type: object  
 '201':  
 description: Created  
 headers: {}  
 content: {}  
 '401':  
 description: Unauthorized  
 headers: {}  
 content: {}  
 '403':  
 description: Forbidden  
 headers: {}  
 content: {}  
 '404':  
 description: Not Found  
 headers: {}  
 content: {}  
 deprecated: false  
 security:  
 - Authorization:  
 - ''  
components:  
 schemas:  
 FloristShopDto:  
 title: FloristShopDto  
 type: object  
 properties:  
 creationDate:  
 type: string  
 format: date-time  
 email:  
 type: string  
 id:  
 type: integer  
 format: int64  
 name:  
 type: string  
 password:  
 type: string  
 phoneNumber:  
 type: string  
 role:  
 $ref: '#/components/schemas/Role2'  
 FlowerDto:  
 title: FlowerDto  
 type: object  
 properties:  
 color:  
 type: string  
 id:  
 type: integer  
 format: int64  
 maxTemperature:  
 type: integer  
 format: int32  
 minTemperature:  
 type: integer  
 format: int32  
 name:  
 type: string  
 shelfLife:  
 type: integer  
 format: int32  
 FlowerStorageRequestDto:  
 title: FlowerStorageRequestDto  
 type: object  
 properties:  
 amount:  
 type: integer  
 format: int32  
 flowerId:  
 type: integer  
 format: int64  
 id:  
 type: integer  
 format: int64  
 storageRoomId:  
 type: integer  
 format: int64  
 LoginDto:  
 title: LoginDto  
 type: object  
 properties:  
 email:  
 type: string  
 password:  
 type: string  
 Role:  
 title: Role  
 type: object  
 properties:  
 id:  
 type: integer  
 format: int64  
 name:  
 $ref: '#/components/schemas/Name'  
 SmartSystemDto:  
 title: SmartSystemDto  
 type: object  
 properties:  
 airQuality:  
 type: number  
 humidity:  
 type: number  
 id:  
 type: integer  
 format: int64  
 satisfactionFactor:  
 type: number  
 temperature:  
 type: number  
 StorageRoomDto:  
 title: StorageRoomDto  
 type: object  
 properties:  
 city:  
 type: string  
 house:  
 type: string  
 id:  
 type: integer  
 format: int64  
 maxCapacity:  
 type: integer  
 format: int32  
 smartDevice:  
 $ref: '#/components/schemas/SmartSystemDto'  
 street:  
 type: string  
 Name:  
 title: Name  
 enum:  
 - ADMIN  
 - USER  
 type: string  
 Role2:  
 title: Role2  
 enum:  
 - ADMIN  
 - USER  
 type: string  
 securitySchemes:  
 Authorization:  
 type: apiKey  
 description: ''  
 name: Authorization  
 in: header  
security:  
- Authorization: []  
tags:  
- name: Authorization  
 description: Auth Controller  
- name: Florist Shop  
 description: Florist Shop Controller  
- name: Flower Storage  
 description: Flower Storage Controller  
- name: Flowers  
 description: Flower Controller  
- name: Role  
 description: Role Controller  
- name: Smart Device  
 description: Smart System Controller**