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

Лабораторна робота №2

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 І. П.

Харків 2021

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

Хід роботи:

Серверна частина проекту повністю реалізована в екосистемі фреймворку Spring. Каскад проекту реалізований за допомогою фреймворку Spring Boot. Для написання контролерів, створення endpoints, реалізації REST API для взаємодії з клієнтами системи використовувався Spring WebFlux – фреймворк для реалізації web-додатків на мовах, що використовують JVM. Для доступу до даних та реалізації ORM використано фреймворк Spring Data JPA та Hibernate. 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.

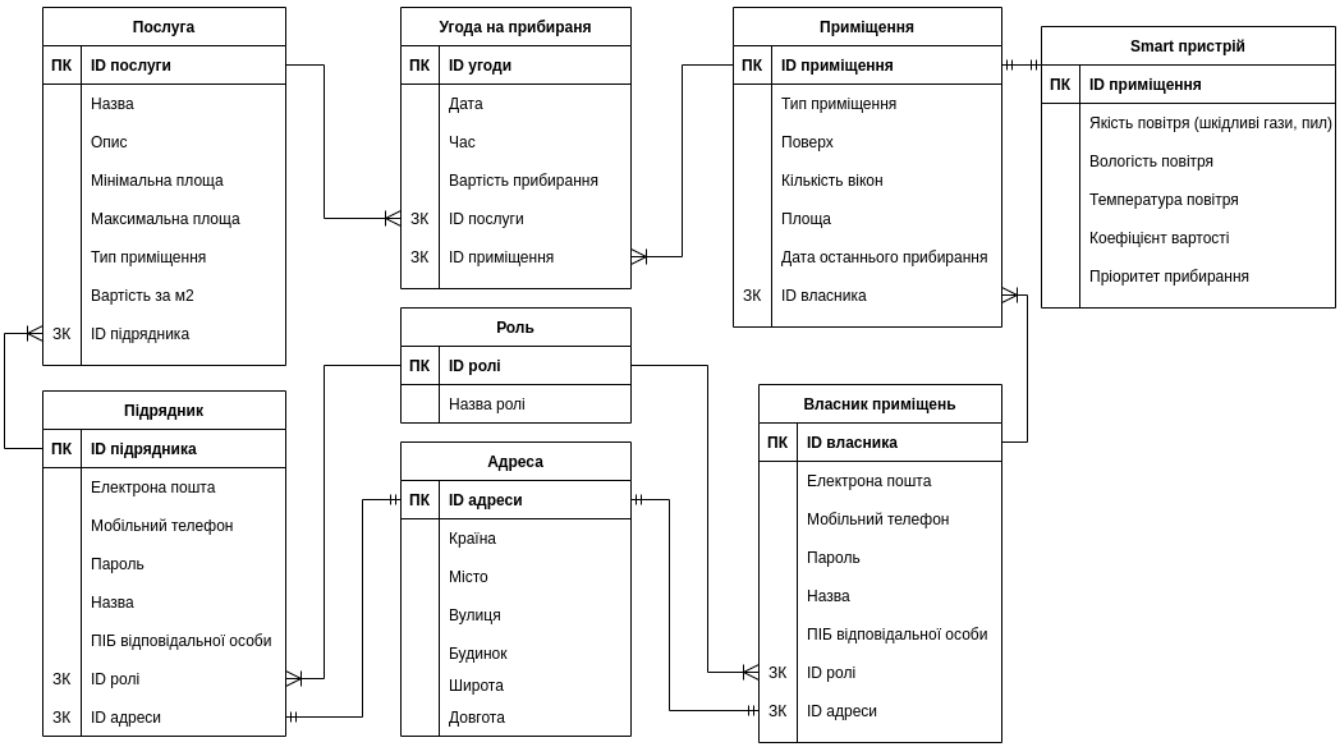


Рисунок 1 – ER-модель даних

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

У підсумку отримано 9 класів сутностей, які відповідають таблицям у базі даних. Приклад програмної реалізації класу PlacementOwner, який відповідає таблиці «Власник приміщень»:

```
@Entity
@Data
@NoArgsConstructor
@EqualsAndHashCode(callSuper = true)
@ToString(callSuper = true)
public class PlacementOwner extends User {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "placement_owner_id")
    private Long id;

    @Column(name = "phone_number", unique = true)
    private String phoneNumber;

    @Column(name = "name")
    private String name;

    @Column(name = "creation_date")
    private Date creationDate;

    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "address_id",
        referencedColumnName = "address_id")
    private Address address;

    @OneToMany(mappedBy = "placementOwner", fetch = FetchType.EAGER)
    @OnDelete(action = OnDeleteAction.CASCADE)
    private Set<Placement> placements;

}
```

Наступним кроком було створення інтерфейсів-репозиторіїв, які надають API для доступу та маніпулювання даними програмної системи, які зберігаються у СКБД MySQL. Всі інтерфейси розширюють функціонал інтерфейсу з параметрами CrudRepository фреймворку Spring Data JPA, надають базовий доступ до даних, Spring-компонент «Repository» для впровадження у сервісний шар та можливість написання власних методів вибірки даних для фільтрацій, статистик, пошуків тощо.

Приклад програмної реалізації інтерфейсу AddressRepository:

```
@Repository
public interface AddressRepository extends CrudRepository<Address, Long>
{
    List<Address> findAllByCountry(String country);

    List<Address> findAllByCountryAndCity(String country, String city);

    List<Address> findAllByCountryAndCityAndStreet(String country,
String city, String address);
}
```

Усі ці інтерфейси являють собою перший і найнижчий шар серверного рівню 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 – 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. Приклад програмної реалізації логіки регулювання вартості складених угод на прибирання приміщення наведено у додатку А. Приклад програмної реалізації бізнес-логіки автоматичного розрахунку вартості праці підрядного працівника наведено у додатку Б.

Наступний, останній рівень розподілення програми, це рівень 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 з заголовком Authorization, у якому зберігається унікальний токен користувача, реалізований за допомогою Json Web Token. Таким чином система стає більш захищеною і відповідає сучасним вимогам до захисту програмних систем. Докладніше засоби встановлення безпеки у системі будуть описані далі.

Отже, на основі реалізованих трьох головних 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 відноситься до логіки захисту даних. Ці ж самі компоненти системи зображені в серверному вузлу на діаграмі розгортання, яка демонструє загальну архітектуру системи.

Після написання логіки 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.

У структурі проекту також наявні два допоміжні пакети класів. 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, наведена у додатку Д. Специфікація REST у форматі OpenAPI Specification (OAS) наведена у додатку Е.

Посилання на архів з програмним кодом та файлом контрольної суми: https://drive.google.com/drive/folders/1TpkrPkFM1t04XNKByazjfc-D0a9uWiut?usp=sharing

Контрольна сума до архіву: 69642801caacae9c70efeca63070276b

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).

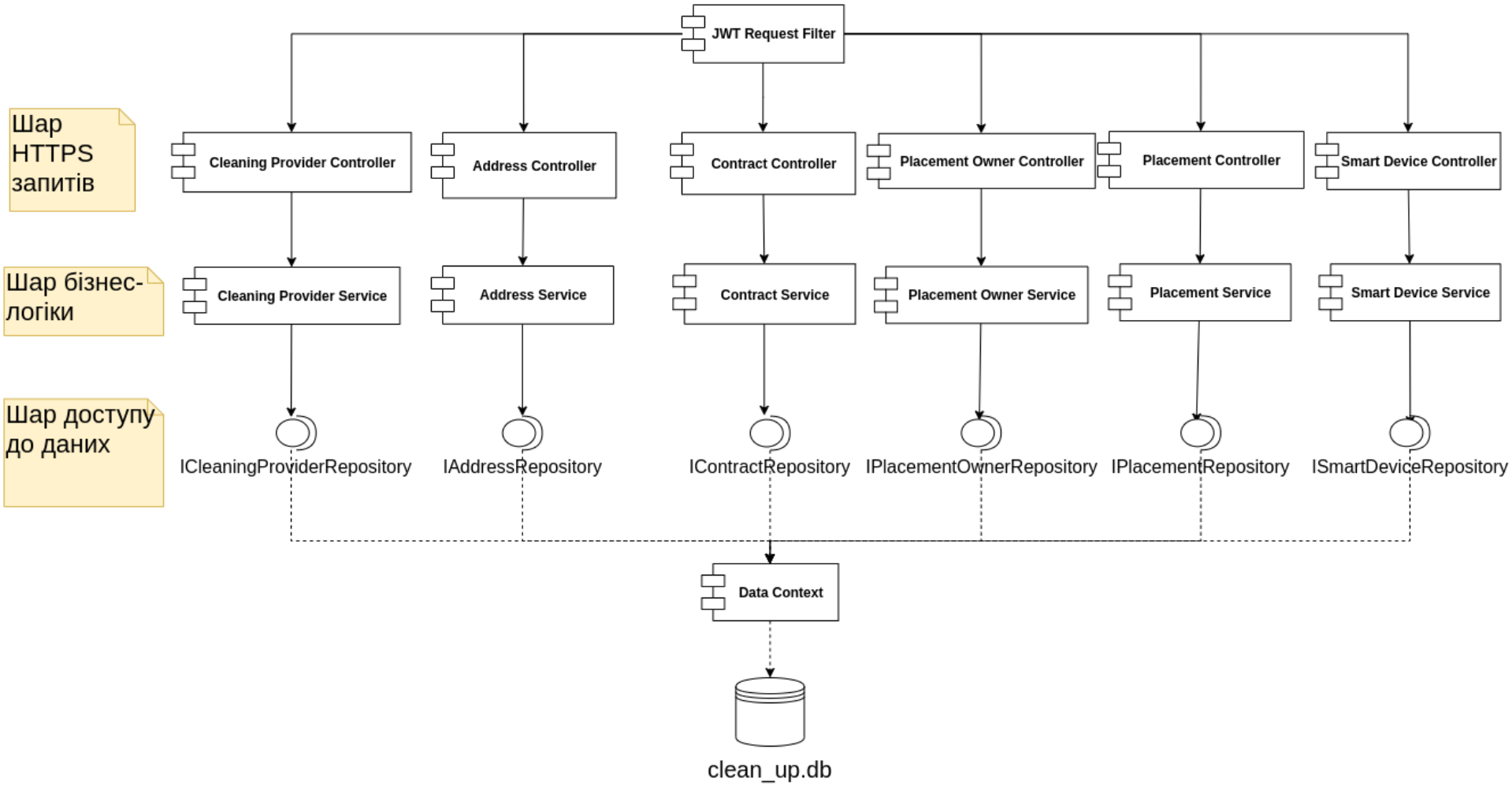
ДОДАТОК А. Програмна реалізація регулювання вартості складених угод

```
@Override
public PlacementDto updateSmartDevice(SmartDeviceDto smartDeviceDto) {
    Optional<Placement> placementDto = placementRepository
            .findById(smartDeviceDto.getId());

    if (placementDto.isPresent()) {
        Placement placement = placementDto.get();

        SmartDevice smartDevice = placement.getSmartDevice();
        double previousAdjustmentFactor =
                smartDevice.getAdjustmentFactor();
        double adjustmentFactor =
                round(smartDeviceDto.getAdjustmentFactor());

        Date currentDate = new Date();
        placement.getContracts().stream()
                .filter(
                        contract -> contract.getDate().after(currentDate)
                )
                .forEach(contract -> {
                    double price = contract.getPrice();

                    if (previousAdjustmentFactor != 0) {
                        price /= previousAdjustmentFactor;
                    }

                    if (adjustmentFactor != 0) {
                        price *= adjustmentFactor;
                    }

                    contract.setPrice(round(price));
                });

        smartDevice
                .setAirQuality(smartDeviceDto.getAirQuality())
                .setTemperature(smartDeviceDto.getTemperature())
                .setHumidity(smartDeviceDto.getHumidity())
                .setAdjustmentFactor(adjustmentFactor)
                .setDirtinessFactor(smartDeviceDto.getDirtinessFactor())
                .setPriority(smartDeviceDto.getPriority());
        placement.setSmartDevice(smartDevice);

        return PlacementMapper
                .toPlacementDto(placementRepository.save(placement));
    }

    return null;
}
```

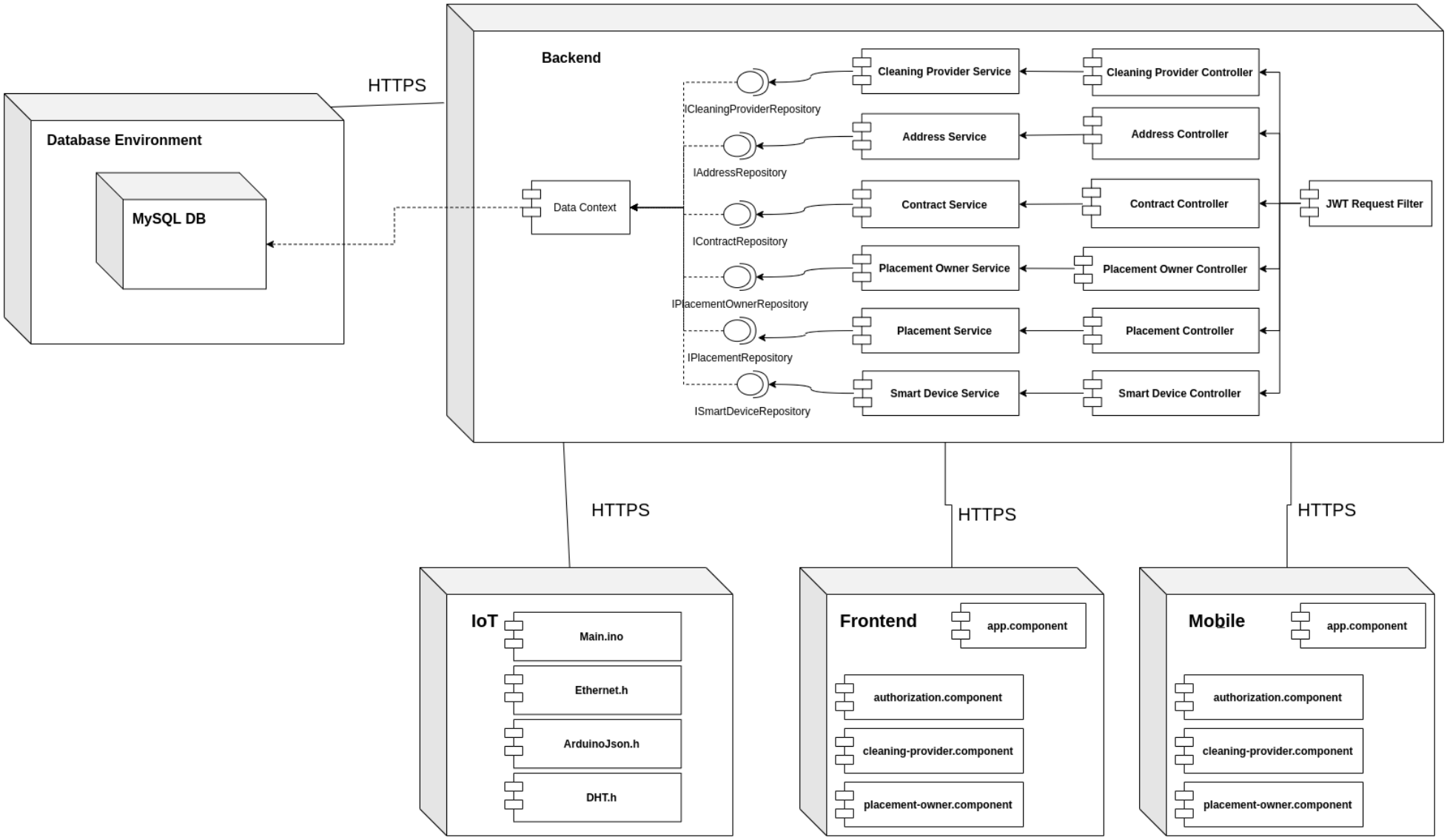
Додаток Б. Програмна реалізація розрахунку вартості праці підрядника

```
1   @Override
2   public ContractResponseDto create(PriceDto priceDto) {
3       Contract contract = new Contract();
4       contract.setDate(priceDto.getDate());
5
6       Optional<Placement> placementById = placementRepository
7               .findById(priceDto.getPlacementId());
8       Optional<ProviderService> providerServiceById =
9               providerServiceRepository.findById(
10                      priceDto.getProviderServiceId()
11              );
12
13      if (placementById.isPresent() && providerServiceById.isPresent()) {
14          Placement placement = placementById.get();
15          ProviderService providerService = providerServiceById.get();
16
17          contract.setPlacement(placement);
18          contract.setProviderService(providerService);
19
20          double price =
21                  providerService.getPricePerMeter() * placement.getArea();
22          Double priceFactor = null;
23
24          SmartDevice smartDevice = placement.getSmartDevice();
25
26          if (isPresent(smartDevice))
27              priceFactor = smartDevice.getAdjustmentFactor();
28
29          if (isPresent(priceFactor)) {
30              if (priceFactor != 0) {
31                  price = price * priceFactor;
32              }
33          }
34          contract.setPrice(price);
35
36          return ContractMapper.toContractResponseDto(
37                  contractRepository.save(contract)
38          );
39      }
40
41      return null;
42  }
```

Додаток В. Діаграма компонентів системи



Додаток Г. Діаграма розгортання системи



Додаток Д. Діаграма варіантів використання системи

ДОДАТОК Е. Специфікація REST у форматі OpenAPI Specification


```
{
  "swagger": "2.0",
  "info": {
    "description": "Documentation for REST API endpoints",
    "version": "1.0.0",
    "title": "Backend system API"
  },
  "host": "localhost:8080",
  "basePath": "/",
  "tags": [
    {
      "name": "Admin",
      "description": "Admin Controller"
    },
    {
      "name": "Authorization",
      "description": "Auth Controller"
    },
    {
      "name": "Cleaning Provider",
      "description": "Cleaning Provider Controller"
    },
    {
      "name": "Contract",
      "description": "Contract Controller"
    },
    {
      "name": "Placement Owner",
      "description": "Placement Owner Controller"
    },
    {
      "name": "Role",
      "description": "Role Controller"
    },
    {
      "name": "Smart Device",
      "description": "Smart Device Controller"
    }
  ],
  "paths": {
    "/admin/backup": {
      "get": {
        "tags": [
          "Admin"
        ],
        "summary": "Performs data backup and returns mysql dump file",
        "operationId": "getBackupData",
        "produces": [
          "*/*"
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "type": "object"
            }
          },
          "401": {
            "description": "Unauthorized"
```

```
      },
      "403": {
        "description": "Forbidden"
      },
      "404": {
        "description": "Not Found"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
  }
},
"/auth/register/cleaning-provider": {
  "post": {
    "tags": [
      "Authorization"
    ],
    "summary": "Registers a new cleaning provider",
    "operationId": "registerCleaningProvider",
    "consumes": [
      "application/json"
    ],
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "in": "body",
        "name": "cleaningProviderDto",
        "description": "cleaningProviderDto",
        "required": true,
        "schema": {
          "$ref": "#/definitions/CleaningProviderDto"
        }
      }
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "type": "object"
        }
      },
      "201": {
        "description": "Created"
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      },
      "404": {
        "description": "Not Found"
      }
    },
    "security": [
```

```
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
  }
},
"/auth/register/placement-owner": {
  "post": {
    "tags": [
      "Authorization"
    ],
    "summary": "Registers a new placement owner",
    "operationId": "registerPlacementOwner",
    "consumes": [
      "application/json"
    ],
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "in": "body",
        "name": "placementOwnerDto",
        "description": "placementOwnerDto",
        "required": true,
        "schema": {
          "$ref": "#/definitions/PlacementOwnerDto"
        }
      }
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "type": "object"
        }
      },
      "201": {
        "description": "Created"
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      },
      "404": {
        "description": "Not Found"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
  }
},
```

```
"/cleaning-providers": {
  "get": {
    "tags": [
      "Cleaning Provider"
    ],
    "summary": "Returns a list of all cleaning providers",
    "operationId": "getAllCleaningProviders",
    "produces": [
      "*/*"
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/CleaningProviderDto"
          }
        }
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      },
      "404": {
        "description": "Not Found"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
  },
  "post": {
    "tags": [
      "Cleaning Provider"
    ],
    "summary": "Adds new cleaning provider",
    "operationId": "addCleaningProvider",
    "consumes": [
      "application/json"
    ],
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "in": "body",
        "name": "cleaningProviderDto",
        "description": "cleaningProviderDto",
        "required": true,
        "schema": {
          "$ref": "#/definitions/CleaningProviderDto"
        }
      }
    ],
    "responses": {
```

```
      "200": {
        "description": "OK",
        "schema": {
          "type": "object"
        }
      },
      "201": {
        "description": "Created"
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      },
      "404": {
        "description": "Not Found"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
  },
  "put": {
    "tags": [
      "Cleaning Provider"
    ],
    "summary": "Updates the cleaning provider",
    "operationId": "updateCleaningProvider",
    "consumes": [
      "application/json"
    ],
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "in": "body",
        "name": "cleaningProviderDto",
        "description": "cleaningProviderDto",
        "required": true,
        "schema": {
          "$ref": "#/definitions/CleaningProviderDto"
        }
      }
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "type": "object"
        }
      },
      "201": {
        "description": "Created"
      },
      "401": {
        "description": "Unauthorized"
```

```
      },
      "403": {
        "description": "Forbidden"
      },
      "404": {
        "description": "Not Found"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
  }
},
"/cleaning-providers/services/{id}": {
  "get": {
    "tags": [
      "Cleaning Provider"
    ],
    "summary": "Finds provider service by id",
    "operationId": "getProviderServiceById",
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "name": "id",
        "in": "path",
        "description": "id",
        "required": true,
        "type": "integer",
        "format": "int64"
      }
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "type": "object"
        }
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      },
      "404": {
        "description": "Not Found"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
```

```
  },
  "delete": {
    "tags": [
      "Cleaning Provider"
    ],
    "summary": "Deletes provider service by ID",
    "operationId": "deleteProviderService",
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "name": "id",
        "in": "path",
        "description": "id",
        "required": true,
        "type": "integer",
        "format": "int64"
      }
    ],
    "responses": {
      "200": {
        "description": "OK"
      },
      "204": {
        "description": "No Content"
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
  }
},
"/cleaning-providers/{email}": {
  "get": {
    "tags": [
      "Cleaning Provider"
    ],
    "summary": "Finds cleaning provider by email",
    "operationId": "getCleaningProviderByEmail",
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "name": "email",
        "in": "path",
        "description": "email",
        "required": true,
        "type": "string"
      }
    ],
```

```
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "type": "object"
        }
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      },
      "404": {
        "description": "Not Found"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
  },
  "delete": {
    "tags": [
      "Cleaning Provider"
    ],
    "summary": "Deletes cleaning provider by email",
    "operationId": "deleteCleaningProvider",
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "name": "email",
        "in": "path",
        "description": "email",
        "required": true,
        "type": "string"
      }
    ],
    "responses": {
      "200": {
        "description": "OK"
      },
      "204": {
        "description": "No Content"
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
```

```
      }
    ],
    "deprecated": false
  }
},
"/cleaning-providers/{email}/services": {
  "get": {
    "tags": [
      "Cleaning Provider"
    ],
    "summary": "Returns all cleaning provider services (offers)",
    "operationId": "getAllProviderServices",
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "name": "email",
        "in": "path",
        "description": "email",
        "required": true,
        "type": "string"
      }
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "type": "object"
        }
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      },
      "404": {
        "description": "Not Found"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
  },
  "post": {
    "tags": [
      "Cleaning Provider"
    ],
    "summary": "Adds new service for cleaning provider",
    "operationId": "addProviderService",
    "consumes": [
      "application/json"
    ],
    "produces": [
      "*/*"
    ],
    "parameters": [
```

```
          {
            "name": "email",
            "in": "path",
            "description": "email",
            "required": true,
            "type": "string"
          },
          {
            "in": "body",
            "name": "providerServiceDto",
            "description": "providerServiceDto",
            "required": true,
            "schema": {
              "$ref": "#/definitions/ProviderServiceDto"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "type": "object"
            }
          },
          "201": {
            "description": "Created"
          },
          "401": {
            "description": "Unauthorized"
          },
          "403": {
            "description": "Forbidden"
          },
          "404": {
            "description": "Not Found"
          }
        },
        "security": [
          {
            "JWT": [
              "global"
            ]
          }
        ],
        "deprecated": false
      },
      "put": {
        "tags": [
          "Cleaning Provider"
        ],
        "summary": "Updates service of cleaning provider (service ID must be
present!)",
        "operationId": "updateProviderService",
        "consumes": [
          "application/json"
        ],
        "produces": [
          "*/*"
        ],
        "parameters": [
          {
            "name": "email",
            "in": "path",
```

```
        "description": "email",
        "required": true,
        "type": "string"
      },
      {
        "in": "body",
        "name": "providerServiceDto",
        "description": "providerServiceDto",
        "required": true,
        "schema": {
          "$ref": "#/definitions/ProviderServiceDto"
        }
      }
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "type": "object"
        }
      },
      "201": {
        "description": "Created"
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      },
      "404": {
        "description": "Not Found"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
  }
},
"/device": {
  "post": {
    "tags": [
      "Smart Device"
    ],
    "summary": "Update smart device characteristics, endpoint for Arduino",
    "operationId": "updateSmartDevice",
    "consumes": [
      "application/json"
    ],
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "in": "body",
        "name": "smartDeviceDto",
        "description": "smartDeviceDto",
        "required": true,
```

```
        "schema": {
          "$ref": "#/definitions/SmartDeviceDto"
        }
      }
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "type": "object"
        }
      },
      "201": {
        "description": "Created"
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      },
      "404": {
        "description": "Not Found"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
  }
},
"/placement-owners": {
  "get": {
    "tags": [
      "Placement Owner"
    ],
    "summary": "Returns a list of all placement owners",
    "operationId": "getAllPlacementOwners",
    "produces": [
      "*/*"
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/PlacementOwnerDto"
          }
        }
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      },
      "404": {
        "description": "Not Found"
```

```
    }
  },
  "security": [
    {
      "JWT": [
        "global"
      ]
    }
  ],
  "deprecated": false
},
"post": {
  "tags": [
    "Placement Owner"
  ],
  "summary": "Adds new placement owner",
  "operationId": "addPlacementOwner",
  "consumes": [
    "application/json"
  ],
  "produces": [
    "*/*"
  ],
  "parameters": [
    {
      "in": "body",
      "name": "placementOwnerDto",
      "description": "placementOwnerDto",
      "required": true,
      "schema": {
        "$ref": "#/definitions/PlacementOwnerDto"
      }
    }
  ],
  "responses": {
    "200": {
      "description": "OK",
      "schema": {
        "type": "object"
      }
    },
    "201": {
      "description": "Created"
    },
    "401": {
      "description": "Unauthorized"
    },
    "403": {
      "description": "Forbidden"
    },
    "404": {
      "description": "Not Found"
    }
  },
  "security": [
    {
      "JWT": [
        "global"
      ]
    }
  ],
  "deprecated": false
},
```

```
  "put": {
    "tags": [
      "Placement Owner"
    ],
    "summary": "Updates the placement owner",
    "operationId": "updatePlacementOwner",
    "consumes": [
      "application/json"
    ],
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "in": "body",
        "name": "placementOwnerDto",
        "description": "placementOwnerDto",
        "required": true,
        "schema": {
          "$ref": "#/definitions/PlacementOwnerDto"
        }
      }
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "type": "object"
        }
      },
      "201": {
        "description": "Created"
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      },
      "404": {
        "description": "Not Found"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
  }
},
"/placement-owners/placements/{id}": {
  "get": {
    "tags": [
      "Placement Owner"
    ],
    "summary": "Finds placement by id",
    "operationId": "getPlacementById",
    "produces": [
      "*/*"
    ],
```

```
    "parameters": [
      {
        "name": "id",
        "in": "path",
        "description": "id",
        "required": true,
        "type": "integer",
        "format": "int64"
      }
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "type": "object"
        }
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      },
      "404": {
        "description": "Not Found"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
  },
  "delete": {
    "tags": [
      "Placement Owner"
    ],
    "summary": "Deletes placement by ID",
    "operationId": "deletePlacement",
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "name": "id",
        "in": "path",
        "description": "id",
        "required": true,
        "type": "integer",
        "format": "int64"
      }
    ],
    "responses": {
      "200": {
        "description": "OK"
      },
      "204": {
        "description": "No Content"
      },
      "401": {
```

```
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
  }
},
"/placement-owners/{email}": {
  "get": {
    "tags": [
      "Placement Owner"
    ],
    "summary": "Finds placement owner by email",
    "operationId": "getPlacementOwnerByEmail",
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "name": "email",
        "in": "path",
        "description": "email",
        "required": true,
        "type": "string"
      }
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "type": "object"
        }
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      },
      "404": {
        "description": "Not Found"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
  },
  "delete": {
    "tags": [
```

```
      "Placement Owner"
    ],
    "summary": "Deletes placement owner by email",
    "operationId": "deletePlacementOwner",
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "name": "email",
        "in": "path",
        "description": "email",
        "required": true,
        "type": "string"
      }
    ],
    "responses": {
      "200": {
        "description": "OK"
      },
      "204": {
        "description": "No Content"
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      }
    },
    "security": [
      {
        "JWT": [
          "global"
        ]
      }
    ],
    "deprecated": false
  }
},
"/placement-owners/{email}/placements": {
  "get": {
    "tags": [
      "Placement Owner"
    ],
    "summary": "Returns all placements",
    "operationId": "getAllPlacements",
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "name": "email",
        "in": "path",
        "description": "email",
        "required": true,
        "type": "string"
      }
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
```

```
        "type": "object"
      }
    },
    "401": {
      "description": "Unauthorized"
    },
    "403": {
      "description": "Forbidden"
    },
    "404": {
      "description": "Not Found"
    }
  },
  "security": [
    {
      "JWT": [
        "global"
      ]
    }
  ],
  "deprecated": false
},
"post": {
  "tags": [
    "Placement Owner"
  ],
  "summary": "Adds new placement for owner",
  "operationId": "addPlacement",
  "consumes": [
    "application/json"
  ],
  "produces": [
    "*/*"
  ],
  "parameters": [
    {
      "name": "email",
      "in": "path",
      "description": "email",
      "required": true,
      "type": "string"
    },
    {
      "in": "body",
      "name": "placementDto",
      "description": "placementDto",
      "required": true,
      "schema": {
        "$ref": "#/definitions/PlacementDto"
      }
    }
  ],
  "responses": {
    "200": {
      "description": "OK",
      "schema": {
        "type": "object"
      }
    },
    "201": {
      "description": "Created"
    },
    "401": {
```

```
      "description": "Unauthorized"
    },
    "403": {
      "description": "Forbidden"
    },
    "404": {
      "description": "Not Found"
    }
  },
  "security": [
    {
      "JWT": [
        "global"
      ]
    }
  ],
  "deprecated": false
},
"put": {
  "tags": [
    "Placement Owner"
  ],
  "summary": "Updates placement owner (placement id must be present)",
  "operationId": "updatePlacement",
  "consumes": [
    "application/json"
  ],
  "produces": [
    "*/*"
  ],
  "parameters": [
    {
      "name": "email",
      "in": "path",
      "description": "email",
      "required": true,
      "type": "string"
    },
    {
      "in": "body",
      "name": "placementDto",
      "description": "placementDto",
      "required": true,
      "schema": {
        "$ref": "#/definitions/PlacementDto"
      }
    }
  ],
  "responses": {
    "200": {
      "description": "OK",
      "schema": {
        "type": "object"
      }
    },
    "201": {
      "description": "Created"
    },
    "401": {
      "description": "Unauthorized"
    },
    "403": {
      "description": "Forbidden"
```

```
        },
        "404": {
          "description": "Not Found"
        }
      },
      "security": [
        {
          "JWT": [
            "global"
          ]
        }
      ],
      "deprecated": false
    }
  },
  "/roles": {
    "get": {
      "tags": [
        "Role"
      ],
      "summary": "Returns a list of all roles",
      "operationId": "getAllRoles",
      "produces": [
        "*/*"
      ],
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Role"
            }
          }
        },
        "401": {
          "description": "Unauthorized"
        },
        "403": {
          "description": "Forbidden"
        },
        "404": {
          "description": "Not Found"
        }
      },
      "security": [
        {
          "JWT": [
            "global"
          ]
        }
      ],
      "deprecated": false
    }
  }
},
"securityDefinitions": {
  "JWT": {
    "type": "apiKey",
    "name": "Authorization",
    "in": "header"
  }
},
```

```
"definitions": {
  "AddressDto": {
    "type": "object",
    "properties": {
      "city": {
        "type": "string"
      },
      "country": {
        "type": "string"
      },
      "houseNumber": {
        "type": "string"
      },
      "latitude": {
        "type": "string"
      },
      "longitude": {
        "type": "string"
      },
      "street": {
        "type": "string"
      }
    },
    "title": "AddressDto"
  },
  "CleaningProviderDto": {
    "type": "object",
    "properties": {
      "address": {
        "$ref": "#/definitions/AddressDto"
      },
      "creationDate": {
        "type": "string",
        "format": "date-time"
      },
      "email": {
        "type": "string"
      },
      "id": {
        "type": "integer",
        "format": "int64"
      },
      "name": {
        "type": "string"
      },
      "password": {
        "type": "string"
      },
      "phoneNumber": {
        "type": "string"
      },
      "role": {
        "type": "string",
        "enum": [
          "ADMIN",
          "PLACEMENT_OWNER",
          "CLEANING_PROVIDER"
        ]
      }
    },
    "title": "CleaningProviderDto"
  },
  "ContractRequestDto": {
```

```
    "type": "object",
    "properties": {
      "date": {
        "type": "string",
        "format": "date-time"
      },
      "id": {
        "type": "integer",
        "format": "int64"
      },
      "placementId": {
        "type": "integer",
        "format": "int64"
      },
      "providerServiceId": {
        "type": "integer",
        "format": "int64"
      }
    },
    "title": "ContractRequestDto"
  },
  "LoginDto": {
    "type": "object",
    "properties": {
      "email": {
        "type": "string"
      },
      "password": {
        "type": "string"
      }
    },
    "title": "LoginDto"
  },
  "PlacementDto": {
    "type": "object",
    "properties": {
      "area": {
        "type": "number",
        "format": "double"
      },
      "floor": {
        "type": "integer",
        "format": "int32"
      },
      "id": {
        "type": "integer",
        "format": "int64"
      },
      "lastCleaning": {
        "type": "string",
        "format": "date-time"
      },
      "placementType": {
        "type": "string"
      },
      "smartDevice": {
        "$ref": "#/definitions/SmartDeviceDto"
      },
      "windowsCount": {
        "type": "integer",
        "format": "int32"
      }
    },
```

```
  "title": "PlacementDto"
},
"PlacementOwnerDto": {
  "type": "object",
  "properties": {
    "address": {
      "$ref": "#/definitions/AddressDto"
    },
    "creationDate": {
      "type": "string",
      "format": "date-time"
    },
    "email": {
      "type": "string"
    },
    "id": {
      "type": "integer",
      "format": "int64"
    },
    "name": {
      "type": "string"
    },
    "password": {
      "type": "string"
    },
    "phoneNumber": {
      "type": "string"
    },
    "role": {
      "type": "string",
      "enum": [
        "ADMIN",
        "PLACEMENT_OWNER",
        "CLEANING_PROVIDER"
      ]
    }
  },
  "title": "PlacementOwnerDto"
},
"ProviderServiceDto": {
  "type": "object",
  "properties": {
    "description": {
      "type": "string"
    },
    "id": {
      "type": "integer",
      "format": "int64"
    },
    "maxArea": {
      "type": "integer",
      "format": "int32"
    },
    "minArea": {
      "type": "integer",
      "format": "int32"
    },
    "name": {
      "type": "string"
    },
    "placementType": {
      "type": "string"
    },
```

```
        "pricePerMeter": {
          "type": "number",
          "format": "double"
        }
      },
      "title": "ProviderServiceDto"
    },
    "Role": {
      "type": "object",
      "properties": {
        "id": {
          "type": "integer",
          "format": "int64"
        },
        "name": {
          "type": "string",
          "enum": [
            "ADMIN",
            "PLACEMENT_OWNER",
            "CLEANING_PROVIDER"
          ]
        }
      },
      "title": "Role"
    },
    "SmartDeviceDto": {
      "type": "object",
      "properties": {
        "adjustmentFactor": {
          "type": "number",
          "format": "double"
        },
        "airQuality": {
          "type": "number",
          "format": "double"
        },
        "dirtinessFactor": {
          "type": "number",
          "format": "double"
        },
        "humidity": {
          "type": "number",
          "format": "double"
        },
        "id": {
          "type": "integer",
          "format": "int64"
        },
        "priority": {
          "type": "string"
        },
        "temperature": {
          "type": "number",
          "format": "double"
        }
      },
      "title": "SmartDeviceDto"
    }
  }
}
```