

Федеральное государственное автономное образовательное  
учреждение высшего образования

Университет ИТМО

Дисциплина: Сервис-ориентированная архитектура

## **Лабораторная работа 1**

Вариант 1407

**Выполнили:**

Кривоносов Егор Дмитриевич

**Группа:** Р34111

**Преподаватель:**

Райла Мартин

2022 г.

Санкт-Петербург

# Оглавление

Оглавление	2
Задание	3
Сгенерированный Swagger UI	3
Исходный код спецификации	4
Вывод	5

# Задание

Введите вариант: 1407

## Внимание! У разных вариантов разный текст задания!

Разработать спецификацию в формате OpenAPI для набора веб-сервисов, реализующего следующую функциональность:

**Первый веб-сервис** должен осуществлять управление коллекцией объектов. В коллекции необходимо хранить объекты класса **Flat**, описание которого приведено ниже:

```
public class Flat {
    private Long id; //Поле не может быть null, Значение поля должно быть больше 0, Значение этого поля должно быть уникальным, Значение этого поля должно генерироваться автоматически
    private String name; //Поле не может быть null, Строка не может быть пустой
    private Coordinates coordinates; //Поле не может быть null
    private java.time.LocalDateTime creationDate; //Поле не может быть null, Значение этого поля должно генерироваться автоматически
    private Integer area; //Поле может быть null, Значение поля должно быть больше 0
    private long numberOfRooms; //Значение поля должно быть больше 0
    private int floor; //Значение поля должно быть больше 0
    private int timeToMetroOnFoot; //Значение поля должно быть больше 0
    private View view; //Поле может быть null
    private House house; //Поле может быть null
}

public class Coordinates {
    private Integer x; //Поле не может быть null
    private Float y; //Поле не может быть null
}

public class House {
    private String name; //Поле может быть null
    private Long year; //Значение поля должно быть больше 0
    private int numberOfFloors; //Значение поля должно быть больше 0
}

public enum View {
    STREET,
    PARK,
    BAD,
    GOOD,
    TERRIBLE;
}
```

Веб-сервис должен удовлетворять следующим требованиям:

- API, реализуемый сервисом, должен соответствовать рекомендациям подхода RESTful.
- Необходимо реализовать следующий базовый набор операций с объектами коллекции: добавление нового элемента, получение элемента по ИД, обновление элемента, удаление элемента, получение массива элементов.
- Операция, выполняемая над объектом коллекции, должна определяться методом HTTP-запроса.
- Операция получения массива элементов должна поддерживать возможность сортировки и фильтрации по любой комбинации полей класса, а также возможность分页ного вывода результатов выборки с указанием размера и порядкового номера выводимой страницы.
- Все параметры, необходимые для выполнения операции, должны передаваться в URL запроса.
- Информация об объектах коллекции должна передаваться в формате **json**.
- В случае передачи сервису данных, нарушающих заданные на уровне класса ограничения целостности, сервис должен возвращать код ответа http, соответствующий произошедшей ошибке.

Помимо базового набора, веб-сервис должен поддерживать следующие операции над объектами коллекции:

- Удалить один (любой) объект, значение поля `view` которого эквивалентно заданному.
- Рассчитать среднее значение поля `timeToMetroOnFoot` для всех объектов.
- Вернуть массив уникальных значений поля `view` по всем объектам.

Эти операции должны размещаться на отдельных URL.

**Второй веб-сервис** должен располагаться на URL `/agency`, и реализовывать ряд дополнительных операций, связанных с вызовом API первого сервиса:


- `/find-with-balcony/{cheapest}/{with-balcony}` : найти самую дешёвую (или дорогую) квартиру с балконом (или без балкона)
- `/get-most-expensive/{id1}/{id2}/{id3}` : выбрать из трёх квартир с заданными id наиболее дорогую

Эти операции также должны размещаться на отдельных URL.

Для разработанной спецификации необходимо сгенерировать интерактивную веб-документацию с помощью Swagger UI. Документация должна содержать описание всех REST API обоих сервисов с текстовым описанием функциональности каждой операции. Созданную веб-документацию необходимо развернуть на сервере **helios**.

# Сгенерированный Swagger UI

[Swagger UI](#)


Swagger  
Powered by SWAGGER

Select a definition

Catalog
Catalog
Agency

## Flat service REST endpoints 1.0.0 OAS3

[/FlatService.yaml](#)

Flat service REST endpoints for SOALab\_1

Servers

https://localhost:8080/api/v1

### Catalog

GET /catalog/flats Get list of flats

POST /catalog/flats Add new flat

PUT /catalog/flats/{id} Update existing flat by id

DELETE /catalog/flats/{id} Delete existing flat by id

DELETE /catalog/flats/delete-one-by-view/{view} Delete one Flat whose value is equivalent to View

GET /catalog/flats/average-time-metro Get average time to metro by foot for all flats

GET /catalog/flats/unique-view Get a list of unique View values for all Flats

### Schemas

View >

House {
name string
example: My House
nullable: true
year integer(\$int64)
minimum: 0
exclusiveMinimum: true
example: 1999
numberOfFloors integer(\$int32)
minimum: 0
exclusiveMinimum: true
example: 8
}

Coordinates >

FlatExtraFieldSchema >

FlatAllFieldsSchema >

FlatsAllFieldsSchema >

Error >

VALID { }

## Исходный код спецификации

[FlatService](#)

[AgencyService](#)

## Вывод

В ходе выполнения лабораторной работы была разработана спецификация в формате OpenAPI для набора веб-сервисов.

Некст лаба:

Проверка на уровне контроллера с помощью аннотации @Valid

Если не возможно сделать - написать заранее

## Лабораторная работа #2

Введите вариант: 95313

### Внимание! У разных вариантов разный текст задания!

На основе разработанной в рамках лабораторной работы #1 спецификации реализовать два веб-сервиса и использующее их API клиентское приложение.

#### Требования к реализации и развёртыванию сервисов:

- Первый ("вызываемый") веб-сервис должен быть реализован на фреймворке Spring MVC REST и развёрнут в окружении под управлением сервера приложений WildFly.
- Второй веб-сервис должен быть реализован на фреймворке JAX-RS, развёрнут в окружении под управлением ещё одного экземпляра сервера приложений WildFly и вызывать REST API первого сервиса.
- Для обоих сервисов необходимо реализовать все функции, задокументированные в API, в строгом соответствии со спецификацией!**
- Доступ к обоим сервисам должен быть реализован с по протоколу https с самоподписанным сертификатом сервера. Доступ к сервисам посредством http без шифрования должен быть запрещён.

#### Требования к клиентскому приложению:

- Клиентское приложение может быть написано на любом веб-фреймворке, который можно запустить на сервере helios.
- Приложение должно обеспечить полный набор возможностей, предоставляемых API обоих сервисов -- включая сортировку, фильтрацию и страничный вывод элементов коллекции.
- Приложение должно преобразовывать передаваемые сервисами данные в человеко-читаемый вид -- параграф текста, таблицу и т.д.
- Клиентское приложение должно информировать пользователя об ошибках, возникающих на стороне сервисов, в частности, о том, что сервису были отправлены невалидные данные.

Оба веб-сервиса и клиентское приложение должны быть развёрнуты на сервере helios.

На 1 лабу:

- ☒ Расписать доп коды ошибок для каждого запроса.
- ☒ Найти (в спецификации например) где указано, что нужно использовать множественное число в объектах. [<https://restfulapi.net/resource-naming/>]
- ☒ PUT или Patch отличия

3) PUT - перезапишет полностью объект, если его не существует тогда его создаст.  
PATCH - перезапишет, только определенные поля или добавит их. Если объекта не существует тогда выдаст ошибку.