Backend Laravel

Необходимо реализовать **REST API** интерфейс для работы с сущностью «оборудование».

#	Роут	Действие	Обязательно
1	`GET:/api/equipment`	Вывод пагинированного списка оборудования с возможностью поиска путем указания query параметров советующим ключам объекта, либо указанием параметра q	Да
2	`GET:/api/equipment/{id}`	Запрос данных по id	Да
3	`POST:/api/equipment`	Создание новой(ых) записи(ей)	Да
4	`PUT:/api/equipment/{id}`	Редактирование записи по id	Да
5	`DELETE:/api/equipment/{id}`	Удаление записи (мягкое удаление)	Да
6	`GET:/api/equipment-type`	Вывод пагинированного списка оборудования с возможностью поиска путем указания query параметров советующим ключам объекта, либо указанием параметра q	Да

Для хранения информации об оборудовании в базе данных используется 2 таблицы (использовать MySQL и работу с миграциями):

1. «Тип оборудования» поля: ID, наименование типа, маска серийного номера.
2. «Оборудование» поля: ID, id типа оборудования, серийный номер (уникальное поле в связке с типом оборудования), примечание/комментарий.

При создании новых записей интерфейс принимает массив объектов, каждый из которых валидируется. Ошибка валидации или сохранения одного объекта не должны приводить к остановке цикла обработки. По каждому из переданных объектов должен быть сформирован ответ (в случае успеха вернуть созданный объект, в случае ошибки массив сообщений)

id	Тип оборудования	Маска SN
1	TP-Link TL-WR74	XXAAAAAXAA
2	D-Link DIR-300	NXXAAXZXaa
…	…	…
100500	D-Link DIR-300 E	NAAAAXZXXX

Где:

- **N** – цифра от 0 до 9;
- **A** – прописная буква латинского алфавита;
- **a** – строчная буква латинского алфавита;
- **X** – прописная буква латинского алфавита либо цифра от 0 до 9;
- **Z** –символ из списка: “-“, “_”, “@”.

Обязательные требования:

1. использование фреймворка Laravel v.8+.
2. использование Form Request Validation для всех методов создания и обновления
3. все ответы API должны использовать API Resources & Resource Collections

Дополнительные требования:

1. наличие PHPDoc для всех методов
2. вся бизнес-логика манипуляции с данными должна быть инкапсулирована в моделях и сервисном слое приложения (контроллеры должны быть плоскими)

Приложение

Примеры форматов используемых в API

> Все форматы объектов могут быть изменены на усмотрение разработчика

Object EquipmentType

```
{
    "id": 0,
    "name": "...",
    "mask": "...",
    "created_at": "0000-00-00 00:00:00",
    "updated_at": "0000-00-00 00:00:00"
}
```

Object Equipment

```
{
    "id": 0,
    "equipment_type": {
        "id": 0,
        "name": "...",
        "mask": "..."
    },
    "serial_number": "...",
    "desc": "...",
    "created_at": "0000-00-00 00:00:00",
    "updated_at": "0000-00-00 00:00:00"
}
```

Формат body для создания Equipments

```
[
    {
        "equipment_type_id": 0,
        "serial_number": "...",
        "desc": "..."
    },
    {
        "equipment_type_id": 0,
        "serial_number": "...",
        "desc": "..."
    }
]
```

Формат ответа при создании Equipments

```
{
    "errors": {
        "0": [
            "Сообщение об ошибке"
        ]
    },
    "success": {
        "1": {
            "id": 0,
            "equipment_type": {
                "id": 0,
                "name": "...",
                "mask": "..."
            },
            "serial_number": "...",
            "desc": "...",
            "created_at": "0000-00-00 00:00:00",
            "updated_at": "0000-00-00 00:00:00"
        }
    }
}
```

> **errors** – содержет перечисление всех объектов с указанием сообщения об ошибках в случае не проходжения валидации или возникновения ошибки сохранения данных.
> **success** – содержет перечисление всех объектов которые были успешно созданы.
>
> В качестве ключей объектов **errors** & **success** используется индексы исходного массива.