Backend

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 ответа	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	Да
5	**DELETE**:/api/equipment/{id}	Удаление записи (мягкое удаление)	Да
6	**GET**:/api/equipment-type	Вывод пагинированного списка типов оборудования с возможностью поиска путем указания query параметров советующим ключам ответа	Да
7	**POST**:/api/user/login	Авторизация пользователя (выпуск токена)	Нет

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

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

При создании новых записей в поле «серийный номер» интерфейс принимает массив номеров каждый из которых валидируется на соответствие маске и отсутствие в таблице «оборудование» в качестве уникального ключа. Создание может быть как по одному объекту, так и коллекция (на входе json-массив).

Если «серийный номер» не проходит валидацию, по нему формируется соответствующие сообщение, которое возвращает интерфейс, после обработки всего массива номеров.

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

id	Тип оборудования	Маска SN
…	…	…
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 CherryPy, Django
2. использование методов валидации данных для всех методов создания и обновления
3. все ответы API должны использовать API Resources/Collections (DTO Pattern)

Дополнительные требования:

1. все запросы API должны быть авторизованы (проверка Bearer Authentication)
2. наличие Docstring для всех методов
3. вся бизнес-логика манипуляции с данными должна быть инкапсулирована в моделях и сервисном слое приложения (контроллеры должны быть плоскими)

Материалы для решения задачи:

- python: https://www.python.org
- CherryPy https://cherrypy.dev / Django https://www.djangoproject.com
- MySQL Server (http://www.mysql.com/downloads/mysql)
- regex101 https://regex101.com

Frontend

Для отображения сделать минимальное **SPA-приложение** с использованием **Vue.js/Nuxt JS**, работающее с разработанным API.

Форма добавления записей, содержащая:

1. Список «Тип оборудования» (id типа) select.
2. поле «Серийные номера» input-text/textarea.
3. Поле «Примечание» textarea
4. Кнопка «Добавить».

Форма(ы) поиска, редактирования и удаления записей:

1. Поисковая строка input-text, для поиска по серийному номеру/примечанию.
2. Поле «Тип оборудования» (показать наименование) — input-text/select.
3. Поле «Серийный номер» — input-text.
4. Поле «Примечание» — input-text/textarea.
5. Кнопки редактировать/сохранить/удалить в зависимости от выдранного режима (просмотр/редактирование).

В качестве **ui kit** можно использовать любой готовый фреймворк/библиотеку.

Редактирование может быть реализовано как в целом всех полей, так и отдельными полями.

Для всех форм обязательно наличие минимальной валидации данных на стороне клиента.