Условия

1. Дана модель предметной области с сущностями:
    1. Страница (Page)
    2. Контент типа видео (Video)
        1. специфичные атрибуты: ссылка на видеофайл, ссылка на файл субтитров
    3. Контент типа аудио(Audio)
        1. специфичные атрибуты: битрейт в количестве бит в секунду
    4. Контент типа текст (Text)
        1. специфичные атрибуты: поле для хранение текста произвольной длинны
2. Нужно учитывать, что специфичные атрибуты разных видов контента существенно различаются.
3. У всех видов контента присутствует атрибут “счетчик просмотров” (counter).
4. У всех видов контента и страниц есть атрибут “заголовок” (title).
5. Со страницей может быть связан любой вид контента в любом количестве. Семантика такая: “на страницу можно выложить любой вид контента в любом количестве”. Например: на странице может быть 5 видео, 3 аудио и 7 текстов в любом порядке и в перемешку.
    1. Следует учитывать, что в будущем виды контента могут добавляться и функционал должен легко расширяться.

Функциональные требования

1. Сделать API для получения списка всех страниц.
    1. Должна поддерживаться пагинация (постраничная выдача результатов)
    2. В ответе должен содержаться URL на API с детальной информацией о странице (пункт №2).
2. Сделать API для получения детальной информации о странице
    1. Помимо атрибутов страницы в ответе должны содержаться все привязанные к странице объекты контента в виде вложенной структуры – упорядоченного списка привязанных к странице объектов контента со всеми атрибутами, включая специфичные.
3. При обращении к API с деталями о странице счетчик просмотров каждого объекта контента, привязанного к странице должен увеличиваться на единицу.
    1. Нагрузка на данное API предполагается существенная, поэтому желательно непосредственно изменение данных в БД реализовать в фоновой задаче.
    2. Важно обратить внимание, что увеличение счетчика должно происходить строго атомарно. То есть, если две задачи параллельно обновляют счетчик одного объекта, то на выходе всегда должно получаться ”+2”.
4. Заведение страниц и привязка к ним контента должна выполняться через админку.
    1. Должен поддерживаться поиск по заголовку (title) страниц и контента (по частичному совпадению от начала).
    2. Желательно для удобства реализовать привязку и управление контентом на странице в виде inline-блоков в разделе управления страницей (Page) в админке.
    3. Желательно, чтобы была возможность задавать порядок выдачи в API объектов, привязанных к странице.
5. Для каждой API должно быть минимум по одному положительному автотесту.

Нефункциональные требования

1. Обязательный технологический стек:
    1. Python 3.6+
    2. Django 2+

    3. Django Rest Framework
    4. Celery
    5. PostgreSQL
    6. Docker + docker-compose
2. Код должен быть выложен на https://github.com или https://gitlab.com
3. Должны быть миграции для наполнения БД данными, необходимыми для демонстрации.
4. README.md в корне должен содержать примеры вызовов API.
5. По команде docker-compose up autotests должны запускать и корректно проходить автотесты.
6. По команде docker-compose up runserver должен запускаться Django runserver для демонстрации работы кода.
    1. Необходимые миграции должны применяться автоматически.
    2. Сервер после запуска должен отвечать по адресу http://0.0.0.0:8000

Условия выполнения задания:

- Ожидаемое время выполнения задания в полном объеме 8 часов.
- Допускается отказ от выполнения некоторых обязательных пунктов требований с подробным обоснованием.
- Выполнение пунктов с формулировкой “желательно” будет плюсом.