ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Название проекта: Portal

База данных: postgresql 10.\* и выше.

СТРУКТУРА САЙТА

* Управление пользователями
* Курсы
* Статистика

# УПРАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯМИ

## Авторизация

Должна проходить по полям username и password. Авторизованный пользователь получает токен, который используется в дальнейших запросах.

## Аутентификация

Должна проходить по токену с ограниченным сроком жизни (3 часа).

КУРСЫ

Внутри данного раздела присутствуют папки, внутри которых находятся документы.

Папка имеет следующие поля:

* название,
* превью,
* видимость (да/нет).

Документ имеет следующие поля:

* папка, в которой он находится,
* название,
* описание,
* превью,
* дата создания,
* видимость (да/нет).

Кроме того, к документу крепятся блоки с контентом. Каждый блок имеет следующие поля:

* документ, которому он принадлежит,
* тип (один из вариантов: html, изображение, файл),
* текст,
* порядок следования,
* файл.

# Задачи

29.07.2020

1. Создать пустой проект. В settings.py настроить доступ к БД, а также к файлам контента и статики. (<https://docs.djangoproject.com/en/2.2/ref/settings/>, параметры STATUC\_ROOT, MEDIA\_ROOT)
2. Создать суперпользователя, чтобы можно было работать с сайтом Django Admin.
3. Создать приложение courses. В models.py определить модели папок и документов, а в admin.py оформить доступ к ним через Django Admin. (<https://docs.djangoproject.com/en/2.2/ref/models/fields/>, поля моделей)

31.07.2020

1. Удалить модель UIDModel, а модель Element наследовать от models.Model.
2. Сделать поля моделей по ТЗ.
3. В admin.py настроить поиск по таблицам.
4. Исключить из индекса папки migrations, .idea и файл db.sqlite3 и удалить их из репозитория. db.sqlite3 удалить вообще.

20.08.2020

В приложение courses добавить следующие запросы:

1. folder/get/

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| **Вход** | | |
| folder\_id | int or None | id папки или None для корня |
| **Выход** | | |
| folders | list of dicts | массив папок с полями id, title, preview |
| documents | list of dicts | массив документов с полями id, title, description, preview, created (timestamp) |

1. document/get/

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| **Вход** | | |
| doc\_id | int | id документа |
| **Выход** | | |
| title | str | название |
| content | list of dicts | массив элементов с полями text, type, file в порядке возрастания поля order |

21.08.2020

1. Создать приложение authentication, в котором определить модель CustomUser, унаследованную от django.contrib.auth.models.User. В модель CustomUser добавить два поля:
   1. token типа models.UUIDField(default=uuid.uuid1),
   2. date\_token\_renewed типа models.DateTimeField(default=timezone.now).
2. В файле settings.py указать
   1. модель CustomUser в качестве модели авторизации,
   2. параметр TOKEN\_EXPIRE = 3, обозначающий количество часов валидности токена.
3. В приложение authentication добавить запрос signin/ для авторизации.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| **Вход** | | |
| username | str | username пользователя |
| password | str | пароль пользователя |
| **Выход** | | |
| token | str | временный токен пользователя |

Этот запрос находит пользователя по username, проверяет активен ли он и если да, то создает новый токен, сохраняет его в поле user.token и дату его создания. Если пользователь с таким username не существует или заблокирован, то возвращать пустое тело и статус 401.

25.08.2020

В приложении authentication создать модуль decorators.py, а в нем определить декоратор auth\_and\_parse, который будет навешиваться на представления в views, требующие аутентификации.

Декоратор выполняет следующие функции:

* Аутентификация пользователя по токену. Токен будет приходить в заголовке Authorization в формате «Token токен», например «Token 693f8d98-e69f-11ea-a955-00505603340c». По присланному токену нужно найти пользователя в БД и проверить возраст этого токена. Если токен устарел или пользователь заблокирован, то вернуть пустое тело со статусом 401.
* Распределение входящих параметров в request. Входящие параметры должны быть распределены по двум словарям:
  + system – словарь, в который попадают query параметры, а также объект пользователя под ключом «\_\_user»;
  + data – словарь в который попадает содержимое тела запроса.
* Декорируемая функция должна принимать system и data, а возвращать словарь с ответом и статус в виде числа.

В итоге после применения декоратора представления будут иметь вид, аналогичный рис. 1.

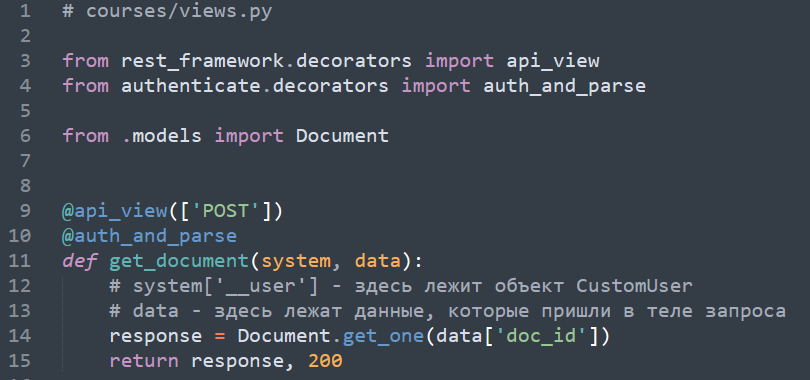


Рис.