**Блог**

**Задание**

Создать простой блог в виде REST сервера выполняющего следующие функции:

* Регистрация
* Авторизация
* Просмотр статей
* Добавление новой стати
* Редактирование статьи
* Удаление статьи
* Возможность изменять статус стати public/draft
* Добавление тегов
* Поиск по тегам и названию статьи
* Добавление комментариев
* Выдача списка статей и комментариев должна иметь возможность постраничного разбиения (skip, limit) и параметров фильтрации по полям

Все пользователи системы, включая незарегестрированных пользователей должны иметь возможность просмаривать публичные статьи.

**Регистрация и авторизация**

Регистрация должна происходить по следующему сценарию:

* пользователь вводить необходимые регистранционные данные
* система генерирует подтверждающую ссылку (код), добавляет ее в Redis и отправляет ее на введенный email
* Коды для email имеют срок жизни 24 часа
* переходя по ссылке, /auth/confirm/:hash\_code система ищет код в Redis и если есть активирует пользователя и дает возможность авторизоваться
* пока пользователь не подтвердил email, при попытке авторизации пользователю выдается соотвествующая ошибка.

**Пользователь должен иметь следующие поля:**

User {

id: Integer/UID,

first\_name: String,

last\_name: String,

password: String,

email: String,

created\_at: Date

}

Пользователь должен иметь возможность восстановить пароль по стандартному сценарию. /auth/forgot\_passwordPOST {email} отправлять письмо с кодом и в редис складывать этот код

/auth/reset POST {code, new\_password} брать из редиса код и если он верен, то устанавливать новый парль

/auth/check\_code GET {code} проверять актуальность кода сброса и возвращать в ответ

**Статьи**

**Статья должна иметь следующие поля:**

Article {

id: Integer/UID,

title: String,

text: Text,

status: Enum,

author\_id: Integer/UID

created\_at: Date,

updated\_at: Date

}

**REST**

PUT /articles/:id - Редактирование стати, может только создатель поста

POST /articles - Добавить пост

GET /articles - получить публичные посты

GET /my - получить список постов авторизованного пользователя

DELETE /articles/:id - удалить стаью, удалить может только автор поста

Фильтрация должна выглядеть следующим образом: /articles?skip=0&limit=10&q=post\_title&author=id&sort=field\_name&order=asc|desc

**Комментарии**

**Комментарий должен иметь следующие поля:**

Comment {

id: Integer/UID,

message: Text,

post\_id: Integer/UID,

author\_id: Integer/UID

created\_at: Date

}

**REST** POST /articles/:id/comments - добавить со ссылкой на пост и пользователя

GET /articles/:id/comments - список комментариев поста

GET /articles/:id/comments/:id - просмотр комментария

DELETE /articles/:id/comments/:id - удалить, удалить может только автор комментария или поста

Фильтрация должна выглядеть следующим образом: /articles/:id/comments?skip=0&limit=10&q=post\_title&author=id&sort=field\_name&order=asc|desc

**Теги**

**Теги должны иметь следующие поля:**

Tag {

id: Integer/UID,

name: String

}

Связь необходимо организовать многие ко многим, через таблицу связки. Теги передаются в виде массива при при создании или редактировании поста. Если переданный тег уже существует в базе не нужно дублировать, а нужно добавить ID уже существующего поста.

**REST**

GET /articles?tags=tag1,tag2 - в результате поиска должны быть посты которые содержат хотябы один искомый тег

GET /tags-cloud - Облако тегов. Где необходимо посчитать кол-во постов в которых встречается тег. Примерный результат: [{tag: 'tagName', post\_count: 10}]

**Тестирование**

В проекте необходимо настроить среду тестирования и создать 2 unit и 2 интеграционных теста.

**Рекомендации**

* Для авторизации необходимо использовать стандарт JWT ( [JSON Web Token](https://jwt.io/introduction/) )
* Для проверки API можно использовать [Postman](https://www.getpostman.com/)
* Проект нужно поднимать в Docker контейнере