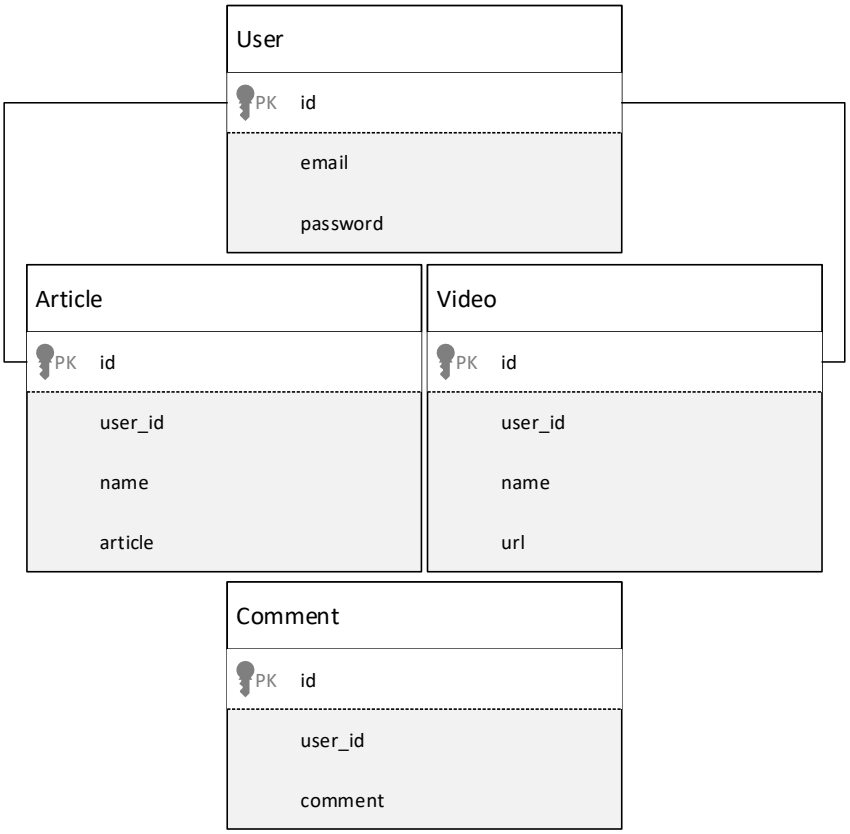
Техническое задание

Основные положения

Требуется разработать REST API на любом удобном языке программирования и фреймворке.

Примерная UML сущностей изображена на следующем рисунке:



Основные требования

Для работы с данными разрешается выбрать любую реляционную базу данных – PostgreSQL, MySQL, SQLite и т.д.

Ответы от API должны быть в формате JSON.

В случае отправки запросов с наличием тела (POST, PUT и т.д.), тело должно иметь формат JSON.

Строгое соблюдение принципов REST не обязательно. Необходимо предоставить для каждой сущности следующий набор возможных действий:

- Создание сущности – подразумевает процедуру генерации строки в СУБД;

- Показ сущности по идентификатору – подразумевает вывод строки из СУБД путём обращения к разработанному приложению с указанием идентификатора сущности;
- Показ коллекции сущностей – подразумевает вывод списка строк из СУБД с возможностями ограничения количества сущностей на странице и постраничной навигации;
- Изменение сущности – подразумевает процедуру изменения созданной ранее сущности путём обращения по идентификатору;
- Удаление сущности – подразумевает процедуру удаления созданной ранее сущности.

Пример маршрутизации запросов к API:

- [GET] /api/model?limit=10&page=1 – показ коллекции сущностей;
- [POST] /api/model – создание сущности;
- [GET] /api/model/:id – показ сущности по идентификатору;
- [PUT] /api/model/:id – изменение сущности;
- [DELETE] /api/model/:id – удаление сущности.

Где GET/POST/PUT/DELETE – тип HTTP запроса, model – название сущности во множественном числе, :id – идентификатор сущности.

Сущность Comment должна иметь возможность быть соединена с сущностями Article или Video, т.е требуется добавить дополнительные поля для указания принадлежности комментария к родительской сущности.

Пример REST API для сущности Article

Для примера рассмотрим перечень маршрутов, тел запроса и тел ответа при работе с REST API для сущности Article.

Показ коллекции сущностей

Запрос: [GET] /api/articles?limit=2&page=1

В данном запросе переданы query параметры, определяющие количество возвращаемых из СУБД строк и номер страницы. Механизм постраничной навигации можно реализовать вручную или с использованием средств фремворка.

Пример ответа API:

```
{
  "data": [
    {
      "id": 1,
```

```
      "user_id": 1,
      "name": "Зачем я это делаю?",
      "article": "Вроде бы и так всё понятно, что за детский садик"
    },
    {
      "id": 2,
      "user_id": 1,
      "name": "Продолжаю зачем-то это делать..",
      "article": "Как же скучно быть бэкендером..."
    }
  ],
  "meta": {
    "current_page": 1,
    "last_page": 3,
    "per_page": 2,
    "total": 5
  }
}
```


Ответ является лишь примером необходимого объёма информации. Обязательным является массив с сущностями типа Article и раздел с результатами постраничной навигации, где указаны текущая страница, последняя доступная страница, количество элементов на странице и общее количество элементов.

Создание сущности

Запрос: [POST] /api/articles

Пример тела запроса:


```
{
  "user_id": 1,
  "name": "Почему текст в Times New Roman?",
  "article": "Чёт как-то госами попахивает, куда я попал.."
}
```


Пример ответа API:


```
{
  "id": 6,
  "user_id": 1,
  "name": "Почему текст в Times New Roman?",
  "article": "Чёт как-то госами попахивает, куда я попал.."
}
```

Обрати внимание, что ID генерируется самостоятельно.

Показ сущности по идентификатору

Запрос: [GET] /api/articles/6

Пример ответа API:


```
{
  "id": 6,
  "user_id": 1,
  "name": "Почему текст в Times New Roman?",
  "article": "Чёт как-то госами попахивает, куда я попал.."
}
```


Изменение сущности

Запрос: [PUT] /api/articles /6

Пример тела запроса:


```
{
  "article": "Даже JSON в Times New Roman, кошмар.."
}
```


Пример ответа API:


```
{
  "id": 6,
  "user_id": 1,
  "name": "Почему текст в Times New Roman?",
  "article": "Чёт как-то госами попахивает, куда я попал.."
}
```


Удаление сущности

Запрос: [DELETE] /api/articles/6

Пример ответа API:


```
{
  "data": "ok"
}
```

Задания со звёздочкой

В качестве демонстрации своих навыков, ты можешь реализовать следующий дополнительный функционал:

- Аутентификация и авторизация. Регистрация должна стать единственной публичной частью API, дальнейшие манипуляции над сущностями должны быть доступны только авторизованному пользователю. При этом свойство user_id должно подставляться в соответствии с идентификатором авторизованного пользователя. В случае редактирования чужой сущности (чужой статьи, видео или комментария) должна выполняться проверка прав – только администратор может менять чужие сущности, иначе пользователь имеет право редактировать только свои сущности или создавать новые. При этим просматривать список пользователей может только администратор, в то время как к остальным сущностям доступ имеют все авторизованные пользователи.
- Добавить механизм поиска и фильтрации в методе вывода коллекции. Предоставить возможность потребителю API указывать дополнительные параметры, позволяющие вернуть только те сущности в коллекции, которые подходят. Например, вернуть только те статьи, которые написал пользователь с id=1. Или вернуть только те статьи, чей текст содержит фразу «JSON»
- Сменить реляционную SQL базу на NoSQL – MongoDB.
- Добавить механизм отправки Email сообщений.
- Добавить контроль отправляемых данных – валидацию. Например, название статьи не может быть меньше 10 символов и длиннее 255.