Задание

“Коллективная записная книга”

Суть задания: разработать серверное приложение, имеющее собственное REST API,

позволяющее приложению-клиенту работать с ним

Основа приложения:

* платформа Node(Node.JS)
* каркас веб-приложения Express.js с набором модулей
  + базовые модули:
    - body-parser
    - compression
    - express-session
    - cookie-session
    - express-uncapitalize
  + промежуточные обработчики, написанные по мере надобности

Использование шаблонизаторов опционально

Проверку работы API осуществлять через Postman

Использование базы данных MySQL для хранения записей

Использование промежуточного обработчика Passport.js по мере надобности

Основные требования:

1. Регистрация пользователей с разными уровнями доступа(именование остается за исполнителем)
   1. Администратор приложения - может просматривать список пользователей, книг, а также их содержимое, удалять/добавлять эти модели
   2. Владелец книги - может просматривать список записей, и осуществлять стандартный набор действий(удаление, создание, изменение), предоставлять доступ к книге пользователям
   3. Пользователь книги - может просматривать список записей в определенной книге, к которой он имеет доступ, добавлять записи в книгу
2. Просмотр записей в БД в виде JSON объектов, сформированными сервером и являющимися его ответом на определенный запрос
   1. Пример 1:
      1. GET “/books/” - возвращает список книг, созданных пользователями
      2. GET “/book/:id” - возвращает список записей, занесенных в книгу с определенным идентификатором
      3. GET “/book/:id/userlist/” - возвращает список пользователей, имеющих доступ к книге, включая владельца
3. Изменение записей в БД
   1. Пример 2:
      1. PUT “/book/:id/article/:id” - изменение содержания существующей записи в определенной существующей книге
      2. PUT “/book/:id/” - изменение информации о существующей книге
      3. DELETE “/book/:id/” - удаление книги
4. Создание записей в БД
   1. Пример 3:
      1. POST “/book/” - создание новой книги (ответ от сервера - идентификатор книги)
      2. POST “/userlist/” - добавление нового пользователя
      3. POST “/book/:id/article/” - создание записи в книге(ответ от сервера - идентификатор записи и книги)
5. Реагирование на попытки получить доступ к данным, которые доступны другой группе пользователей - точнее, установка ограничения на использование чужих ресурсов( ответ сервера должен содержать текст ошибки и код ошибки )
   1. Пример 4:
      1. Обычный пользователь не может обращаться к служебным ресурсам (список пользователей)
      2. Пользователь не может изменять книгу(кроме добавления новых записей), если он не является её владельцем/администратором
6. Реагирование на попытку получить доступ к отсутствующим данным (ответ - 404 - не найдено)
7. Ответы на запросы к API должны быть понятны и читаемы, т.е. если это ошибка - ответ сервера должен быть в виде кода ошибки и описания ошибки, если запрос выполнен успешно - результат выполнения запроса, если возвращается массив записей/пользователей, это должен быть пустой массив, в случае отсутствия записей и массив с данными в обратном случае, никакой отправки null или undefined или любой другой телеги не должно быть, что пользователь запросил - то и получил. Все поля JSON-ответа должны быть записаны в виде camelCase, это более понятно/читаемо чем snake-case или PascalCase.