Пояснительная записка

Основная идея: создание сайта, на котором пользователи могли бы оставлять свои заметки

Возможности: Регистрация, авторизация, добавление, изменение и удаление статей и комментариев к ним, добавление и удаление лайков на статьи

Возможности, предоставляемые API: функции, описанные выше + фильтрация пользователей по никнейму и некоторые другие возможности (см. ниже)

Особенности: возможность добавлять картинки к учётным записям, статьям и комментариям

Возможности API

Пользователи:

* Авторизация: POST /api/login (используйте cookie, полученные с запросом для подтверждения авторизации в других запросах)
  + JSON аргументы:
    - str: email
    - str: password
    - [bool: remember\_me(default=False)]
  + Возвращаемый JSON (в случае успешной операции):
    - {"success": "ok"}
* Выход из аккаунта: POST/api/logout login required
  + Возвращаемый JSON:
    - {"success": "ok"}
* Получение информации о пользователе: GET api/user/<int:user\_id>
  + Аргументы GET:
    - [\*get\_field(choices=[ "id", "name", "surname", "nickname", "email", "description", "avatar", "modified\_date"], default=["id", "nickname"])] (Отмеченные красным аргументы можно получить только при получении информации об аккаунте авторизованного на данный момент пользователя, в ином случае они будут проигнорированы, если все указанные поля будут заблокированными, вернётся только id. Аватар (как и все остальные изображения) возвращается в hex формате)
  + Пример: http://localhost:5000/api/user/1?get\_field=nickname&get\_field=description
  + Возвращаемый JSON:
    - {"user": {field.name: field.value for field in get\_field}}
* Изменение пользователя: PUT api/user/<int:user\_id> login required
  + JSON аргументы:
    - [str: name]
    - [str: surname]
    - [str: nickname]
    - [str: email]
    - [str: new\_password]
    - [str: new\_password\_again]
    - [str: description]
    - [str: avatar(hex format, \*.png/\*.jpg/\*.jpeg)]
    - str: password
  + Возвращаемый JSON:
    - {"success": "ok"}
* Удаление пользователя: DELETE api/user/<int:user\_id> login required
  + JSON аргументы:
    - str: password
  + Возвращаемый JSON:
    - {"success": "ok"}
* Регистрация: POST api/users
  + JSON аргументы:
    - str: name
    - str: surname
    - str: nickname
    - str: email
    - str: password
    - str: password\_again
    - [str: description]
    - [str: avatar(hex format, \*.png/\*.jpg/\*.jpeg)]
  + Возвращаемый JSON:
    - {"success": "ok"}
* Получение информации о списке пользователей: GET api/users
  + Аргументы GET:
    - [int: limit] Максимальное количество записей в ответе
    - [int: offset] Количество пропущенных записей в начале
    - [str: nickname\_search\_string] Поиск пользователя по никнейму
    - [str: nickname\_filter(choices=["equals", "starts", "ends", "contains", "equals\_case\_insensitive"], default="equals")] (При отсутствии аргумента nickname\_search\_string игнорируется. Выделенные красным варианты не учитывают регистр)
    - [\*get\_field(choices=[ "id, "nickname", "description", "avatar"], default=["id", "nickname"])]
  + Возвращаемый JSON:
    - {"users": [{field.name: field.value for field in get\_field} for user in users]}

Статьи:

* Получение информации о статье: GET api/article/<int:article\_id>
  + Аргументы GET:
    - [\*get\_field(choices=[ "id", "title", "content", "image", author", "likes\_count", "create\_date"], default=["id", "title"])]
  + Возвращаемый JSON:
    - {"article": {field.name: field.value for field in get\_field}}
* Изменение статьи: PUT api/article/<int:article\_id> login required
  + JSON аргументы:
    - [str: title]
    - [str: content]
    - [str: image(hex format, \*.png/\*.jpg/\*.jpeg)]
  + Возвращаемый JSON:
    - {"success": "ok"}
* Удаление статьи: DELETE api/article/<int:article\_id> login required
  + Возвращаемый JSON:
    - {"success": "ok"}
* Добавление статьи: POST api/articles login required
  + JSON аргументы:
    - str: title
    - str: content
    - [str: image(hex format, \*.png/\*.jpg/\*.jpeg)]
  + Возвращаемый JSON:
    - {"success": "ok"}
* Получение информации о списке статей: GET api/articles
  + Аргументы GET:
    - [int: limit] Максимальное количество записей в ответе
    - [int: offset] Количество пропущенных записей в начале
    - [str: sorted\_by(choices=["create\_date ", "likes\_count"], default="create\_date ")]
    - [int: author] Фильтрация по id автора
    - [\*get\_field(choices=[ "id", "title", "content", "image", author", "likes\_count", "create\_date"], default=["id", "title"])]
  + Возвращаемый JSON:
    - {"articles": [{field.name: field.value for field in get\_field} for article in articles]}

Комментарии:

* Получение информации о комментарии: GET api/comment/<int:comment\_id>
  + Аргументы GET:
    - [\*get\_field(choices=[ "id", "author", "article\_id", "image", "text", "create\_date"], default=["id", "author", "article\_id"])]
  + Возвращаемый JSON:
    - {"comment": {field.name: field.value for field in get\_field}}
* Изменение комментария: PUT api/comment/<int:comment\_id> login required
  + JSON аргументы:
    - [str: text]
    - [str: image(hex format, \*.png/\*.jpg/\*.jpeg)]
  + Возвращаемый JSON:
    - {"success": "ok"}
* Удаление комментария: DELETE api/comment/<int:comment\_id> login required
  + Возвращаемый JSON:
    - {"success": "ok"}
* Добавление комментария: POST api/comments login required
  + JSON аргументы:
    - str: text
    - int: article\_id
    - [str: image(hex format, \*.png/\*.jpg/\*.jpeg)]
  + Возвращаемый JSON:
    - {"success": "ok"}
* Получение информации о списке комментариев: GET api/comments
  + Аргументы GET:
    - [int: limit] Максимальное количество записей в ответе
    - [int: offset] Количество пропущенных записей в начале
    - [int: author] Фильтрация по id автора
    - [int: article\_id]
    - [\*get\_field(choices=[ "id", "author", "article\_id", "image", "text", "create\_date"], default=["id", "author", "article\_id"])]
  + Возвращаемый JSON:
    - {"comments": [{field.name: field.value for field in get\_field} for comment in comments]}

Лайки:

* Получение информации о том, поставлен ли лайк на запись: GET api/like/<int: article\_id> login required
  + Возвращаемый JSON:
    - { "like\_exist": (bool: текущий пользователь лайкнул запись)}
* Поставить лайк: POST api/like/<int: article\_id> login required
  + Возвращаемый JSON:
    - {"success": "ok"}
* Удалить лайк: DELETE api/like/<int: article\_id> login required
  + Возвращаемый JSON:
    - {"success": "ok"}