**Внутри Route (в переменной типа Request)**

Всегда есть следующие переменные:

isAuthorized (boolean) – признак авторизован ли текущий http-запрос к серверу

user\_id (number) – идентификатор текущего авторизованного пользователя

**Формат ответа от сервера на не-Ajax-запрос**

Сервер генерирует шаблон со следующими полями (метод res.render(…)):

isAuthorized (boolean) (mandatory) – признак, авторизован ли текущий пользователь. Используется при генерации хэдера.

messages (array) – сообщения для пользователя

* type(string) – тип сообщения (пока не исп-ся)
* body (string) – текст сообщения

userId (number) (mandatory) - идентификатор текущего авторизованного пользователя. Используется при генерации хэдера.

**Формат ответа от сервера на Ajax-запрос**

JSON-объект со следующими полями (метод res.json(…))

* status (string) (mandatory) – результат выполнения запроса. Значения: ‘ok’, ‘not ok’.
* другие необязательные поля

**Замечание**

Замечание 1.

На странице Редактирования профиля в трех местах присутствуют чекбоксы: удобства квартиры (utility, flat\_utility), опции Кампуса500 (campus500, user\_campus500) и районы (district, user\_district). Первое хранится и обрабатывается по одной схеме, а второе и третье – по другой.

Схема1 (удобства квартиры)

Таблица utility хранит список всех возможных удобств. Flat\_utility является таблицей-пересечением между таблицами flat и utility, наличие записи в ней со ссылками на кваритру и удобство по первичному ключу является признаком наличия данного удобства в данной квартире.

В этой схеме есть минус. Когда нужно вывести удобства данной квартиры на веб-страницу, я делаю два запроса к БД (по одному на каждую таблицу). Далее в коде следует двойной цикл, в котором я формирую объект удобный для вставки в шаблон. Объект имеет следующий вид:

{

id: идентифкатор,

name: название,

isChecked: признак наличия

}

Схема2 (Кампус500 и Районы)

Например, возьмем Кампус500 потому, что Районы запрограммированы аналогично. Таблица campus500 является, как и в предыдущей схеме, таблицей всех возможных районов города. В таблице user\_campus500 хранятся не только внешние ключи, но также и названия, и признак выбран(checked). Это позволяет избавиться от минуса схемы1 при генерации объекта для вставки в шаблон. Это объект имеет аналогичную струтуру, как и в схеме1, однако в отличие от нее может быть получен одним запросом к базе данных. Это позволяет избавиться в коде от одного запроса к БД и от двойного цикла, алгоритмическую сложность которого я оцениваю как n^2/2, где n – количество записей в campus500.

На сервер приходят с клиента все опции кампуса, для каждой из которых выполняется отдельная команда UPDATE в БД. Количество этих команд равно количеству таких опций, и линейно растет с их количеством, что я считаю минусом. В схеме1 я обходил всего двумя командами – delete и insert. То есть я сначала все удалял, а затем вставлял нужные записи.

Однако есть слабое место схемы1, заключающееся в том, что между операциями delete и insert (которые обернуты в Promisы и выполняются асинхронно) данные отсутствуют, что потенциально может породить ошибку.