

КОНКУРСНОЕ ЗАДАНИЕ
Модуль С
Серверное программирование REST API

ВВЕДЕНИЕ

Технологии этого модуля: Серверное программирование REST API.

Время на выполнение: 3 часа.

К вам обратился застройщик, которому необходимо разработать инструмент продажи квартир для внутреннего использования в компании.

Заказчик предоставляет вам готовую базу данных. Вам необходимо использовать все имеющиеся навыки в серверной разработке для создания REST API.

Заказчик хочет, чтобы API можно было легко поддерживать, поэтому использование фреймворков будет плюсом.

ОПИСАНИЕ ПРОЕКТА И ЗАДАЧ

Ваша задача – реализовать REST API, которое будет отвечать требованиям заказчика.

Для вашего удобства, во всех URL будет использоваться переменная {host} которая обозначает хост адрес API: <http://xxxxxx-m3.wsr.ru>, где xxxxxx – логин участника.

В случае ошибок связанных с валидацией данных во всех запросах необходимо возвращать следующее тело ответа:

```

{
  "error": {
    "code": <code>,
    "message": <message>,
    "errors": {
      <key>: [ <error message>]
    }
  }
}

```

Обратите внимание, что вместо <code> и <message> необходимо указывать соответствующее значение, определенное в описании ответа на соответствующий запрос. В свойстве error.errors необходимо перечислить те свойства, которые не прошли валидацию, а в их значениях указать массив с ошибками валидации.

Например если отправить пустой запрос на сервер, где проверяется следующая валидация:

- phone – обязательно поле;
- password – обязательное поле,

то тело ответа должно быть следующим

```

{
  "error": {
    "code": 422,
    "message": "Validation error",

```

```

“errors”: {
  phone: [ “field phone can not be blank” ],
  password: [ “field password can not be blank” ]
}
}
}

```

Учтите, что `code` и `message` могут быть определены иначе, если в запросе указано иное. В значениях свойств `errors` вы можете использовать любые сообщения об ошибках (если не указана конкретная ошибка), но они должны описывать возникшую проблему.

Авторизация

Запрос для авторизации пользователя в системе. При отправке запроса необходимо передать объект с логином и паролем. Если клиент отправил корректные данные, то необходимо вернуть сгенерированный токен, а иначе сообщение об ошибке.

В базу данных уже внесён пользователь с логином admin и паролем admin, но авторизация должна работать и для других пользователей, которые в дальнейшем появятся в системе.

Request	Response
URL: {host}/api/login Method: POST Headers - Content-Type: application/json	<p>----- Successful -----</p> Status: 200 Content-Type: application/json Body: {

Body: <pre>{ "login": "admin", "password": "admin" }</pre>	<pre>"data": { "token": <сгенерированный token> }</pre> <p>----- Validation error -----</p> <p>Status: 422 Content-Type: application/json Body:</p> <pre>{ "error": { "code": 422, "message": "Validation error", "errors": { <key>: <массив ошибок> } } }</pre> <p>----- Unauthorized -----</p> <p>Status: 401 Content-Type: application/json Body:</p> <pre>{ "error": { "code": 401, "message": "Unauthorized", "errors": { "login": ["phone or password incorrect"] } } }</pre>
--	---

Все последующие запросы требуют авторизации с использованием токена. Токен должен быть получен через заголовок Authorization.

При принятии запроса без заголовка авторизации сервер, должен вернуть следующий ответ:

```

Status: 403
Content-Type: application/json
Body:
{
  "error": {
    "code": 403,
    "message": "Вы должны авторизоваться",
  }
}

```

Получение информации об объектах

При отправке данного запроса, с сервера должен вернуться список всех проектов с информацией о них:

- id – ID проекта;
- name – название проекта;
- flat_statuses – количество квартир:
 - free – свободных;
 - reserved – забронированных;
 - sold – проданных.

Запрос на получение списка объектов:

Request	Response
URL: {host}/api/project Method: GET	<p>----- Successful -----</p> <p>Status: 200 Content-Type: application/json Body:</p> <pre> { "data": { "projects": [{ "id": 1, "name": "ЖК Новый", "flat_statuses": { </pre>

	<pre> "free": 19, "reserved": 17, "sold": 64, }, { "id": 2, "name": "ЖК Новый 2", "flat_statuses": { "free": 19, "reserved": 17, "sold": 64, }, }] } </pre>
--	--

Создание объекта

При создании объекта пользователь должен передать на сервер следующие данные:

- name – название объекта (обязательное, строка);
- coords – координаты объекта (обязательное, два десятичных числа через запятую);
- district – район города (обязательное, строка);
- website – сайт объекта (обязательное, URL);

В случае успешного создания, с сервера должен вернуться ID созданного объекта, а иначе список ошибок.

Запрос на создание объекта:

Request	Response
URL: {host}/api/project Method: POST Body: {	----- Successful ----- Status: 201 Content-Type: application/json

<pre>"name": "ЖК Вересаево", "coords": "47.241768, 39.800856", "district": "Александровка", "website": "https://veresaevo.ru/", }</pre>	<p>Body:</p> <pre>{ "data": { "id": 1 } }</pre> <p>----- Validation error -----</p> <p>Status: 422</p> <p>Content-Type: application/json</p> <p>Body:</p> <pre>{ "error": { "code": 422, "message": "Validation error", "errors": { <key>: <массив ошибок> } } }</pre>
---	--

Получение информации об объекте

При отправке данного запроса, с сервера должна вернуться информация об объекте и домах, которые находятся в нём.

Получить информацию об объекте можно по ID объекта:

Request	Response
<p>URL: {host}/api/project/{project_id}</p> <p>Method: GET</p>	<p>----- Successful -----</p> <p>Status: 200</p> <p>Content-Type: application/json</p> <p>Body:</p> <pre>{ "data": { "id": 1, "name": "ЖК Вересаево", "coords": "47.241768, 39.800856", "district": "Александровка", "website": "https://veresaevo.ru/", "houses": [</pre>

	<pre> { "id": 1, "name": "Литер 1", }, { "id": 2, "name": "Литер 2", },] } </pre>
--	--

Обновление информации об объекте

Запрос для обновления информации об объекте:

Request	Response
URL: {host}/api/project/{project_id} Method: PATCH Headers - Content-Type: application/json Body: <pre> { "name": "Эко-квартал Вересаево", "coords": "47.241768, 39.800856", "district": "Александровка", "website": "https://veresaevo.ru/", } </pre>	<p>----- Successful -----</p> <p>Status: 204</p> <p>----- Validation error -----</p> <p>Status: 422 Content-Type: application/json Body:</p> <pre> { "error": { "code": 422, "message": "Validation error", "errors": { <key>: <массив ошибок> } } } </pre>

Создание дома

При создании дома пользователь должен передать на сервер следующие данные:

- `project_id` – id объекта (обязательное, ID существующее в таблице `projects`);
- `name` – название дома (обязательное, строка);
- `address` – адрес объекта (обязательное, строка);
- `built_year` – год сдачи (обязательное, четырёхзначное число);
- `built_quarter` – квартал сдачи (обязательное, целое число от 1 до 4);

В случае успешного создания, с сервера должен вернуться ID созданного дома, а иначе список ошибок.

Запрос на создание дома:

Request	Response
URL: {host}/api/house Method: POST Body: <pre>{ "project_id": 1, "name": "Литер 1", "address": "г. Ростов-на-Дону, ул.Вересаева, 101/3 стр.1", "built_year": 2024, "built_quarter": 2, }</pre>	<p>----- Successful -----</p> <p>Status: 201 Content-Type: application/json Body:</p> <pre>{ "data": { "id": 1 } }</pre> <p>----- Validation error -----</p> <p>Status: 422 Content-Type: application/json Body:</p> <pre>{ "error": { "code": 422, "message": "Validation error", "errors": { <key>: <массив ошибок> } } }</pre>

Получение информации о доме

При отправке данного запроса, с сервера должна вернуться информация о доме и подъездах, которые находятся в нём.

Получить информацию о доме можно по ID дома:

Request	Response
URL: {host}/api/house/{house_id} Method: GET	<p>----- Successful -----</p> <p>Status: 200</p> <p>Content-Type: application/json</p> <p>Body:</p> <pre>{ "data": { "id": 1, "name": "Литер 1", "address": "г. Ростов-на-Дону, ул.Вересаева, 101/3 стр.1", "built_year": 2024, "built_quarter": 2, "sections": [{ "id": 1, "number": 1, }, { "id": 2, "number": 2, },] } }</pre>

Обновление информации о доме

Запрос для обновления информации о доме:

Request	Response
URL: {host}/api/house/{house_id} Method: PATCH	<p>----- Successful -----</p> <p>Status: 204</p>

Headers - Content-Type: application/json Body: <pre>{ "name": "Литер 1", "address": "г. Ростов-на-Дону, ул.Вересаева, 101/3 стр.1", "built_year": 2024, "built_quarter": 4, }</pre>	<p>----- Validation error -----</p> Status: 422 Content-Type: application/json Body: <pre>{ "error": { "code": 422, "message": "Validation error", "errors": { <key>: <массив ошибок> } } }</pre>
---	---

Создание подъезда

При создании подъезда пользователь должен передать на сервер следующие данные:

- `house_id` – id дома (обязательное, ID существующее в таблице `houses`);
- `number` – номер подъезда (обязательное, целое число);
- `floors` – количество этажей (обязательное, целое число);
- `flats_on_floor` – количество квартир на этаже (обязательное, целое число);
- `starting_flat_number` – номер с которого начинается нумерации квартир (обязательное, целое число);

Обратите внимание, что одновременно с созданием подъезда, в базу данных должны быть добавлены и квартиры этого подъезда с данными:

- `id` – ID квартиры;
- `section_id` – ID подъезда, в котором находится квартира;
- `floor` – этаж, на котором находится квартира;

- flat_number – номер квартиры;
- status – статус квартиры, по-умолчанию “free”.

Массив квартир для добавления вы формируете самостоятельно на основе данных о подъезде.

В случае успешного создания, с сервера должен вернуться ID созданного подъезда, а иначе список ошибок.

Запрос на создание подъезда:

Request	Response
URL: {host}/api/section Method: POST Body: <pre>{ "house_id": 1, "number": 2, "floors": 20, "flats_on_floor": 12, "starting_flat_number": 241, }</pre> Создание квартир: После отправки этого запроса будет создано 20 (floors) * 12 (flats_on_floor) = 240 квартир с номерами от 241 (starting_flat_number) до 480.	<p>----- Successful -----</p> <p>Status: 201 Content-Type: application/json Body:</p> <pre>{ "data": { "id": 1 } }</pre> <p>----- Validation error -----</p> <p>Status: 422 Content-Type: application/json Body:</p> <pre>{ "error": { "code": 422, "message": "Validation error", "errors": { "<key>": "<массив ошибок>" } } }</pre>

Получение информации о подъезде

При отправке данного запроса, с сервера должна вернуться информация о подъезде и квартирах, которые находятся в нём.

Получить информацию о подъезде можно по ID подъезда:

Request	Response
URL: {host}/api/section/{section_id} Method: GET	<p>----- Successful -----</p> <p>Status: 200</p> <p>Content-Type: application/json</p> <p>Body:</p> <pre>{ "data": { "id": 1, "number": 2, "flats": [{ "id": 241, "flat_number": 241, "floor": 1, "size": 86.2, "rooms": 3, "price": 5000000, "status": "free", }, { "id": 242, "flat_number": 242, "floor": 1, "size": 36.8, "rooms": 1, "price": 2500000, "status": "reserved", }, { "id": 243, "flat_number": 243, "floor": 1, "size": 57.2, "rooms": 2, "price": 4000000, }] } }</pre>

	<pre> "status": "sold", } ...] } }</pre>
--	---

Обновление информации о квартирах

Для изменения информации о квартирах в подъезде, пользователь должен передать на сервер следующие данные:

- flats – массив ID квартир (обязательное);
- size – площадь (обязательное, десятичное число);
- rooms – количество комнат (обязательное, целое число от 0 до 6);
- price – стоимость квартиры (обязательное, целое число от 0);

Запрос для массового обновления информации о квартирах:

Request	Response
URL: {host}/api/flat Method: PATCH Headers - Content-Type: application/json Body: <pre> { "flats": [241, 253, 265, 277, 289, 301, 313, 325, 337, 349, 361, 373, 385, 397, 409, 421, 433, 445, 457, 469], "size": 86.2, "rooms": 3, "price": 5600000 }</pre>	<p>----- Successful -----</p> <p>Status: 204</p> <p>----- Validation error -----</p> <p>Status: 422 Content-Type: application/json Body:</p> <pre> { "error": { "code": 422, "message": "Validation error", "errors": { <key>: <массив ошибок> } } }</pre>

Получение квартир дома

Квартиры, которые находятся в конкретном доме, можно получить по ID дома:

Request	Response
URL: {host}/api/house/{house_id}/flats Method: GET	<p>----- Successful -----</p> <p>Status: 200 Content-Type: application/json Body:</p> <pre>{ "data": { "flats": [{ "id": 241, "flat_number": 241, "floor": 1, "size": 86.2, "rooms": 3, "price": 5000000, "status": "free", "section": { "id": 1, "number": 2, "floors": 20, "flats_on_floor": 12, }, }, { "id": 242, "flat_number": 242, "floor": 1, "size": 36.8, "rooms": 1, "price": 2500000, "status": "reserved", "section": { "id": 1, "number": 2,</pre>

	<pre> "floors": 20, "flats_on_floor": 12, }, }, { "id": 243, "flat_number": 243, "floor": 1, "size": 57.2, "rooms": 2, "price": 4000000, "status": "sold", "section": { "id": 1, "number": 2, "floors": 20, "flats_on_floor": 12, }, }, ...] } </pre>
--	--

Обновление статуса квартиры

Данный запрос принимает обязательный параметр **status**, который может иметь следующие значения:

- “free” – свободна;
- “reserved” – забронирована;
- “sold” – продана.

Запрос для обновления статуса квартиры:

Request	Response
URL: {host}/api/flat/{flat_id} Method: PATCH Headers	<p>----- Successful -----</p> <p>Status: 204</p> <p>----- Validation error -----</p> <p>Status: 422</p> <p>Content-Type: application/json</p>

- Content-Type: application/json Body: <pre>{ "status": "reserved", }</pre>	Body: <pre>{ "error": { "code": 422, "message": "Validation error", "errors": { <key>: <массив ошибок> } } }</pre>
---	--

ИНСТРУКЦИЯ ДЛЯ КОНКУРСАНТА

В медиафайлах вам предоставляется SQL дамп с готовой базой данных. Структуру БД менять запрещается. Добавлять новые записи в существующие таблицы разрешается.

Заказчик допускает возможность изменения базы данных в будущем, поэтому вам необходимо подготовить свой вариант схемы базы данных и сохранить его в корне с модулем. Сохраните файл с названием DB.png.

Форматы запросов и ответов должны соответствовать примерам из задания.

Разработанное приложение должно быть доступно по адресу <http://xxxxxx-m3.wsr.ru/>, где xxxxxx - логин участника (указан на индивидуальной карточке).

Проверяются только работы, загруженные на сервер!

СИСТЕМА ОЦЕНКИ

Секция	Критерий	Сумма
A	Организация работы и управление	1,50
B	Коммуникация и навыки межличностного общения	1,50
C	Графический дизайн	0,00
D	Верстка	0,00
E	Программирование на стороне клиента	0,00
F	Программирование на стороне сервера	17,00
G	CMS	0,00
Всего		20,00