

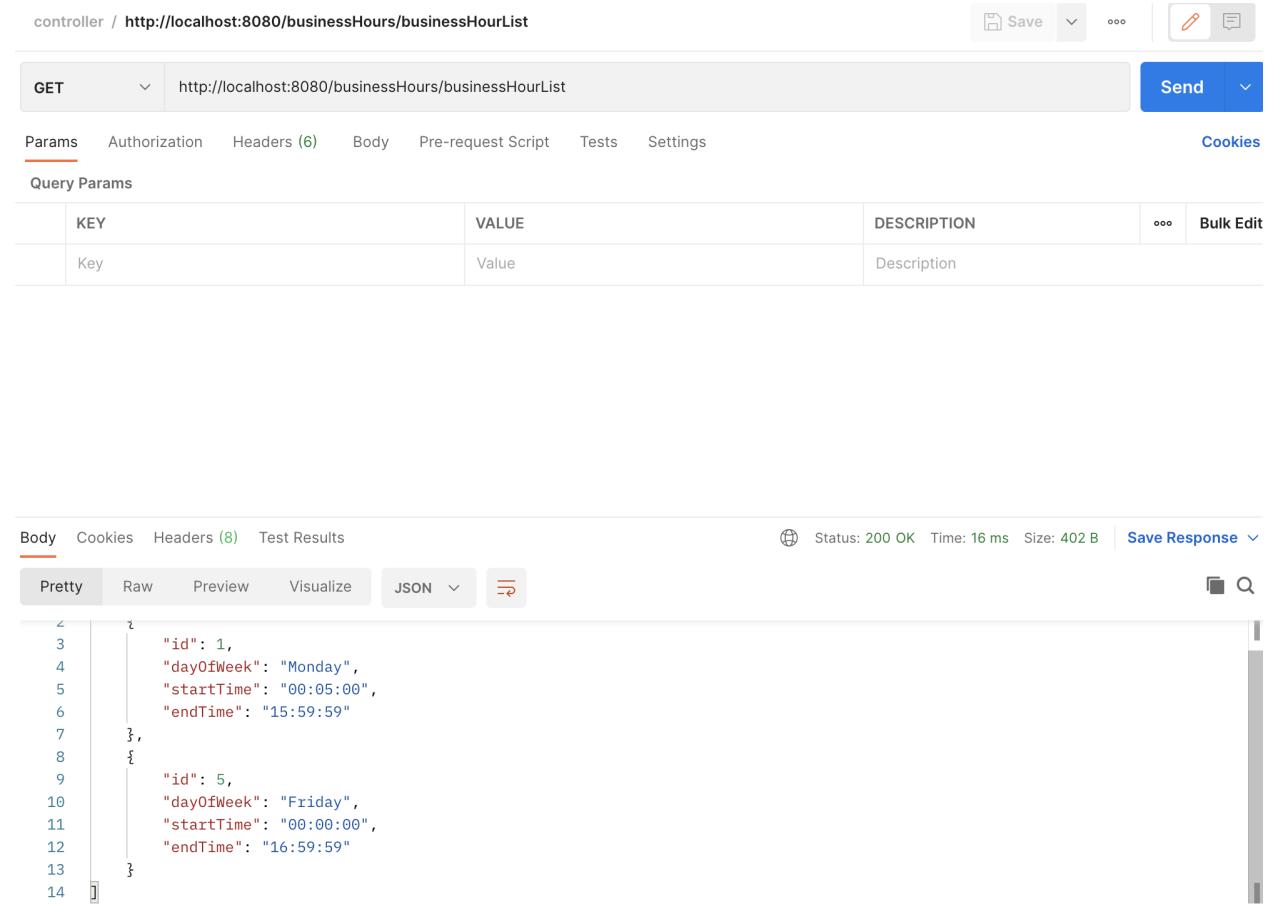
Integration tests for RESTful services

Abstract:

This document records tests mocking a normal use process in reality. Results are presented using screenshots of Postman.

REST Services:

1. getAllBusinessHours



The screenshot shows a Postman request for the endpoint `http://localhost:8080/businessHours/businessHourList`. The request method is `GET`. The response status is `200 OK`, time `16 ms`, size `402 B`. The response body is displayed in Pretty JSON format, showing two objects representing business hours:

```
3
4     "id": 1,
5     "dayOfWeek": "Monday",
6     "startTime": "00:05:00",
7     "endTime": "15:59:59"
8   },
9   {
10    "id": 5,
11    "dayOfWeek": "Friday",
12    "startTime": "00:00:00",
13    "endTime": "16:59:59"
14 }
```

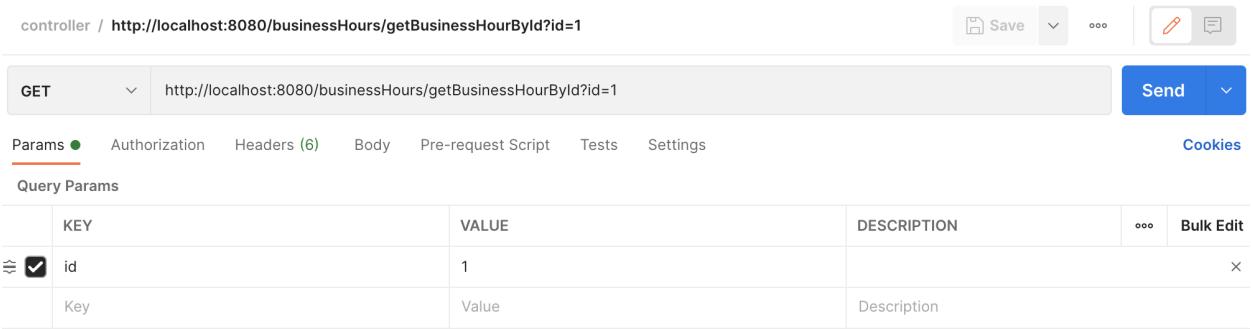
2. createBusinessHour

The screenshot shows the Postman interface with the following details:

- Request URL:** POST <http://localhost:8080/businessHours/createBusinessHour?dayOfWeek=5&startTime=00:00:00&endTime=23:59:59>
- Params:** dayOfWeek (Value: 5), startTime (Value: 00:00:00), endTime (Value: 23:59:59)
- Response Status:** Status: 200 OK Time: 204 ms Size: 326 B
- Response Body (Pretty JSON):**

```
1  {
2   "id": 5,
3   "dayOfWeek": "Friday",
4   "startTime": "00:00:00",
5   "endTime": "23:59:59"
```

3. getBusinessHourById



controller / <http://localhost:8080/businessHours/getBusinessHourById?id=1>

GET <http://localhost:8080/businessHours/getBusinessHourById?id=1> Send

Params ● Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> id	1			X
Key	Value	Description		

Body Cookies Headers (8) Test Results Status: 200 OK Time: 128 ms Size: 326 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {  
2   "id": 1,  
3   "dayOfWeek": "Monday",  
4   "startTime": "00:05:00",  
5   "endTime": "15:59:59"  
6 }
```

4. getBusinessHourByDayOfWeek

controller / http://localhost:8080/businessHours/getBusinessHourByDayOfWeek?dayOfWeek=1

GET http://localhost:8080/businessHours/getBusinessHourByDayOfWeek?dayOfWeek=1 Send

Params ● Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	dayOfWeek	1			
	Key	Value	Description		

Body Cookies Headers (8) Test Results Status: 200 OK Time: 15 ms Size: 326 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {  
2   "id": 1,  
3   "dayOfWeek": "Monday",  
4   "startTime": "00:05:00",  
5   "endTime": "15:59:59"  
6 }
```

5. updateBusinessHourTime

The screenshot shows the Postman interface for a PUT request to `http://localhost:8080/businessHours/updateBusinessHourTime?dayOfWeek=5&startTime=00:00:00&endTime=16:59:59`. The 'Params' tab is selected, displaying the following query parameters:

KEY	VALUE	DESCRIPTION	...	Bulk Edit
dayOfWeek	5			
startTime	00:00:00			
endTime	16:59:59			

The response body shows the updated business hour time:

```
1 {  
2   "id": 5,  
3   "dayOfWeek": "Friday",  
4   "startTime": "00:00:00",  
5   "endTime": "16:59:59"  
6 }
```

6. getAllEvents

The screenshot shows the Postman application interface. At the top, the URL `http://localhost:8080/events/eventList` is entered into the address bar. Below the address bar, there are buttons for `Save`, `Send`, and other options. The main area shows a `GET` request being sent to the specified URL. The `Params` tab is selected, showing a single entry: `Key` with a value of `Value`. Other tabs include `Authorization`, `Headers (6)`, `Body`, `Pre-request Script`, `Tests`, and `Settings`. To the right, there is a `Cookies` section. Below the params table, there is a large empty space. At the bottom, there are tabs for `Body`, `Cookies`, `Headers (8)`, and `Test Results`. The `Test Results` tab is selected. It displays the response status as `Status: 200 OK`, time as `Time: 282 ms`, and size as `Size: 255 B`. There are also `Save Response` and search/filter buttons. The response body is shown as a JSON array: `[]`.

7. createEvent

The screenshot shows the Postman interface with a successful API call to create an event.

Request URL: `http://localhost:8080/events/createEvent?name=party1&startDate=2031-09-06&endDate=2031-09-06&startTime=00:00:00&endTime=00:00:01`

Method: POST

Params: (selected) Authorization, Headers (7), Body, Pre-request Script, Tests, Settings, Cookies

Query Params:

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> name	party1			
<input checked="" type="checkbox"/> startDate	2031-09-06			X
<input checked="" type="checkbox"/> endDate	2031-09-06			
<input checked="" type="checkbox"/> startTime	00:00:00			
<input checked="" type="checkbox"/> endTime	00:00:01			
Key	Value	Description		

Body: (Pretty, Raw, Preview, Visualize, JSON, `curl`)

Status: 200 OK Time: 19 ms Size: 409 B Save Response

```
1 "id": 1053747058,
2 "name": "party1",
3 "timeSlotDto": {
4     "id": 261737600,
5     "startDate": "2031-09-06",
6     "endDate": "2031-09-06",
7     "startTime": "00:00:00",
8     "endTime": "00:00:01"
9 }
10 }
11 }
```

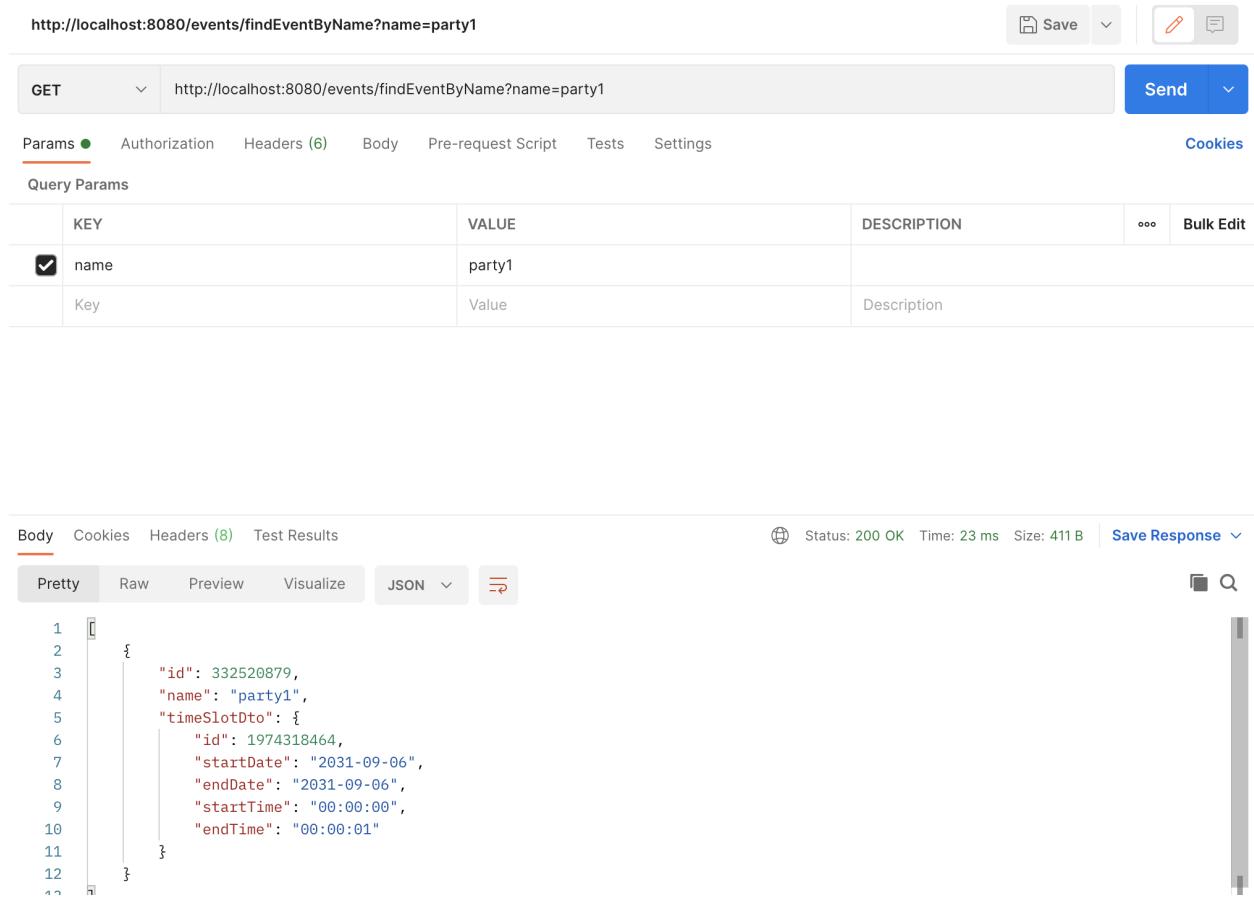
8. getEventById

The screenshot shows the Postman interface with the following details:

- URL:** `http://localhost:8080/events/findEventById?id=332520879`
- Method:** GET
- Params:** A table showing a single query parameter `id` with value `332520879`.
- Body:** JSON response (Pretty) showing the event details:

```
1  {
2      "id": 332520879,
3      "name": "party1",
4      "timeSlotDto": {
5          "id": 1974318464,
6          "startDate": "2031-09-06",
7          "endDate": "2031-09-06",
8          "startTime": "00:00:00",
9          "endTime": "00:00:01"
10     }
11 }
```
- Headers:** (8)
- Test Results:** Status: 200 OK, Time: 14 ms, Size: 409 B
- Save Response:** Available

9. `getEventByName`



The screenshot shows the Postman interface for a GET request to `http://localhost:8080/events/findEventByName?name=party1`. The 'Params' tab is selected, showing a single parameter `name` with value `party1`. The response body is displayed in Pretty mode, showing a JSON object with an ID and a name.

```
1 {  
2   "id": 332520879,  
3   "name": "party1",  
4   "timeSlotDto": {  
5     "id": 1974318464,  
6     "startDate": "2031-09-06",  
7     "endDate": "2031-09-06",  
8     "startTime": "00:00:00",  
9     "endTime": "00:00:01"  
10   }  
11 }  
12 }
```

10. updateEventName

The screenshot shows the Postman interface for a PUT request to update an event name. The URL is `http://localhost:8080/events/updateEventName?id=332520879&name=party2`. The request method is PUT, and the body contains the following JSON:

```
1 {
2     "id": 332520879,
3     "name": "party2",
4     "timeSlotDto": {
5         "id": 1974318464,
6         "startDate": "2031-09-06",
7         "endDate": "2031-09-06",
8         "startTime": "00:00:00",
9         "endTime": "00:00:01"
10    }
11 }
```

The response status is 200 OK, with a time of 27 ms and a size of 409 B.

11. updateEventTimeSlot

The screenshot shows a POST request in Postman to `http://localhost:8080/events/updateEventTimeSlot?id=332520879&startDate=2031-09-07&endDate=2031-09-07&startTime=12:0...`. The request body contains the following JSON:

```
1 {  
2   "id": 332520879,  
3   "name": "party2",  
4   "timeSlotDto": {  
5     "id": 1974318464,  
6     "startDate": "2031-09-07",  
7     "endDate": "2031-09-07",  
8     "startTime": "12:00:00",  
9     "endTime": "19:00:00"  
10   }  
11 }
```

The response status is 200 OK, with a time of 111 ms and a size of 409 B.

12. deleteEvent

The screenshot shows the Postman interface for a DELETE request. The URL is `http://localhost:8080/events/deleteEvent?id=332520879`. The request method is set to `DELETE`. The query parameters are listed as follows:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> id	332520879	
Key	Value	Description

The response body is displayed in Pretty format, showing the deleted event details:

```
1 {  
2   "id": 332520879,  
3   "name": "party2",  
4   "timeSlotDto": {  
5     "id": 1974318464,  
6     "startDate": "2031-09-07",  
7     "endDate": "2031-09-07",  
8     "startTime": "12:00:00",  
9     "endTime": "19:00:00"  
10   }  
11 }
```

The status bar at the bottom indicates: Status: 200 OK Time: 31 ms Size: 409 B Save Response ▾

13. getAllEventRegistration

The screenshot shows the Postman interface with the following details:

- URL:** controller / <http://localhost:8080/eventRegistrations/getEventRegistrationList>
- Method:** GET
- Headers:** (6) (highlighted)
- Body:** (Empty)
- Tests:** (Empty)
- Settings:** (Empty)
- Cookies:** (Empty)
- Query Params:** (Empty)
- Body (JSON Response):**

```
1 [  
2 {  
3   "id": 1572874424,  
4   "personDto": {  
5     "id": -66194669,  
6     "name": "luna",  
7     "address": "joe"  
8   },  
9   "eventDto": {  
10     "id": 1053747058,  
11     "name": "party1",  
12     "timeSlotDto": {  
13       "id": 261727600  
14     }  
15   }  
16 }]
```
- Status:** 200 OK
- Time:** 9 ms
- Size:** 499 B
- Save Response:** (button)

14.attendEventRegistration

The screenshot shows the Postman interface with the following details:

- URL:** `http://localhost:8080/eventRegistrations/attend?pid=-66194669&eid=1053747058`
- Method:** POST
- Params:** pid: -66194669, eid: 1053747058
- Body:** (Pretty) JSON response:

```
1  {
2     "id": 1572874424,
3     "personDto": {
4         "id": -66194669,
5         "name": "luna",
6         "address": "joe"
7     },
8     "eventDto": {
9         "id": 1053747058,
10        "name": "party1",
11        "timeSlotDto": {
12            "id": 261737600,
13            "cetar+Dota" : "2021-09-06"
```
- Headers:** (8)
- Tests:** (None)
- Settings:** (None)
- Cookies:** (None)
- Status:** 200 OK | Time: 33 ms | Size: 497 B | Save Response

15. cancelEventRegistration

The screenshot shows the Postman interface for a DELETE request. The URL is `http://localhost:8080/eventRegistrations/cancel?pid=-66194669&eid=1053747058`. The request method is set to `DELETE`. The query parameters are `pid` (value: -66194669) and `eid` (value: 1053747058). The response status is 200 OK, and the body contains the value `true`.

HTTP Request:

```
DELETE http://localhost:8080/eventRegistrations/cancel?pid=-66194669&eid=1053747058
```

Query Params:

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> pid	-66194669			
<input checked="" type="checkbox"/> eid	1053747058			
Key	Value	Description		

Response:

```
1 true
```

Status: 200 OK | Time: 197 ms | Size: 257 B | Save Response

16.getPersonByEvents

The screenshot shows the Postman application interface. At the top, the URL is set to `http://localhost:8080/eventRegistrations/getPersonByEvent?eid=1053747058`. Below the URL, there are tabs for `GET`, `POST`, `PUT`, and `DELETE`, with `GET` selected. The `Send` button is highlighted in blue. The `Params` tab is active, showing a single query parameter `eid` with the value `1053747058`. Other tabs include `Authorization`, `Headers (6)`, `Body`, `Pre-request Script`, `Tests`, and `Settings`. A `Cookies` tab is also present. The `Query Params` table has two rows: one for the `eid` parameter and another for a general `Key` and `Value`. The `Body` tab shows the JSON response, which is a single object with the following structure:

```
1 [ {  
2   "id": -66194669,  
3   "name": "luna",  
4   "address": "joe"  
5 } ]  
6  
7 ]
```

The `Headers (8)` tab shows the following headers:

- Content-Type: application/json
- Content-Length: 131
- Accept: */*
- Host: localhost:8080
- User-Agent: Postman/7.2.0
- Connection: keep-alive
- Cache-Control: no-cache
- Postman-Token: 3a2a2a2a-2a2a-2a2a-2a2a-2a2a2a2a2a2a

The `Test Results` tab shows a success message: "Success! Response received". The status bar at the bottom indicates: Status: 200 OK Time: 16 ms Size: 301 B Save Response.

17. getEventsByPerson

The screenshot shows the Postman interface for a GET request to `http://localhost:8080/eventRegistrations/getEventsByPerson?pid=-66194669`. The request is successful, returning a status of 200 OK with a response size of 411 B.

Params (1): pid = -66194669

Body (Pretty):

```
1 [ {  
2   "id": 1053747058,  
3   "name": "party1",  
4   "timeSlotDto": {  
5     "id": 261737600,  
6     "startDate": "2031-09-06",  
7     "endDate": "2031-09-06",  
8     "startTime": "00:00:00",  
9     "endTime": "00:00:01"  
10   }  
11 }]  
12  
13 }
```

18. getEventRegistrationByPersonAndEvent

http://localhost:8080/eventRegistrations/getEventRegistrationByPersonAndEvent?pid=-66194669&eid=1053747058

Save |

GET http://localhost:8080/eventRegistrations/getEventRegistrationByPersonAndEvent?pid=-66194669&eid=1053747058 Send

Params ● Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> pid	-66194669			
<input checked="" type="checkbox"/> eid	1053747058			
Key	Value	Description		

Body Cookies Headers (8) Test Results

Status: 200 OK Time: 15 ms Size: 497 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2     "id": -487584312,
3     "personDto": {
4         "id": -66194669,
5         "name": "luna",
6         "address": "joe"
7     },
8     "eventDto": {
9         "id": 1053747058,
10        "name": "party1",
11        "timeSlotDto": {
12            "id": 261737600,
13            "start": "2021-09-06"
```

19. `findParticipantsNumber`

The screenshot shows the Postman application interface. At the top, the URL is set to `http://localhost:8080/eventRegistrations/getParticipantsNumber?eid=1053747058`. Below the URL, the method is selected as `GET`. The `Send` button is visible on the right. The `Params` tab is active, showing a table with one row: `eid` with value `1053747058`. Other tabs like `Authorization`, `Headers (6)`, `Body`, `Pre-request Script`, `Tests`, and `Settings` are also present. The `Query Params` section is empty. On the far right, there are buttons for `Save`, `Edit`, and `Comment`. The `Cookies` tab is also visible.

Below the request configuration, the response details are shown. The `Body` tab is active, displaying the JSON response: `1`. Other tabs include `Cookies`, `Headers (8)`, and `Test Results`. The response status is listed as `Status: 200 OK Time: 13 ms Size: 254 B`. A `Save Response` button is available. The `JSON` tab is selected, and there are buttons for `Pretty`, `Raw`, `Preview`, and `Visualize`. On the right side, there are icons for copy and search.

20. getAllItems

The screenshot shows the Postman application interface. At the top, the URL `http://localhost:8080/items/itemList` is entered. Below the URL, there are buttons for `Save`, `Send`, and other options. The main area shows a `GET` request with the URL `http://localhost:8080/items/itemList`. The `Params` tab is selected, showing a table for `Query Params` with one entry: `Value`. The `Body` tab is selected, showing the response body in `Pretty` format:

```
1 [  
2   {  
3     "id": -1552496352,  
4     "itemCategory": "Book",  
5     "name": "Book",  
6     "inLibrary": true  
7   }  
8 ]
```

The response status is `200 OK` with a time of `11 ms` and a size of `326 B`.

21. **browse**

The screenshot shows the Postman application interface. At the top, the URL `http://localhost:8080/items/browse?pid=-66194669` is entered in the address bar, along with standard save and export buttons.

The request method is set to `GET`. The request URL is displayed as `http://localhost:8080/items/browse?pid=-66194669`.

The `Params` tab is selected, showing a table with one row:

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	pid	-66194669			

The `Headers` section contains 6 items.

The `Body` tab is selected, showing the response body:

```
1 [  
2   "threeCountries"  
3 ]
```

The status bar at the bottom indicates: Status: 200 OK Time: 59 ms Size: 271 B Save Response ▾

22. createItem

http://localhost:8080/items/createItem?name=Book&itemCategory=Book

Save |

POST http://localhost:8080/items/createItem?name=Book&itemCategory=Book

Send

Params ● Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	name	Book			
<input checked="" type="checkbox"/>	itemCategory	Book			
	Key	Value	Description		

Body Cookies Headers (8) Test Results

Status: 200 OK Time: 44 ms Size: 324 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {  
2   "id": -1552496352,  
3   "itemCategory": "Book",  
4   "name": "Book",  
5   "inLibrary": true  
6 }
```

23. deleteItem

The screenshot shows the Postman interface for a DELETE request. The URL is `http://localhost:8080/items/deleteItem?id=-1552496352`. The request method is set to `DELETE`. The response status is `200 OK` with a time of `37 ms` and a size of `334 B`. The response body is displayed in Pretty format:

```
1 {  
2   "id": -1552496352,  
3   "itemCategory": "Book",  
4   "name": "threeCountries",  
5   "inLibrary": true  
6 }
```

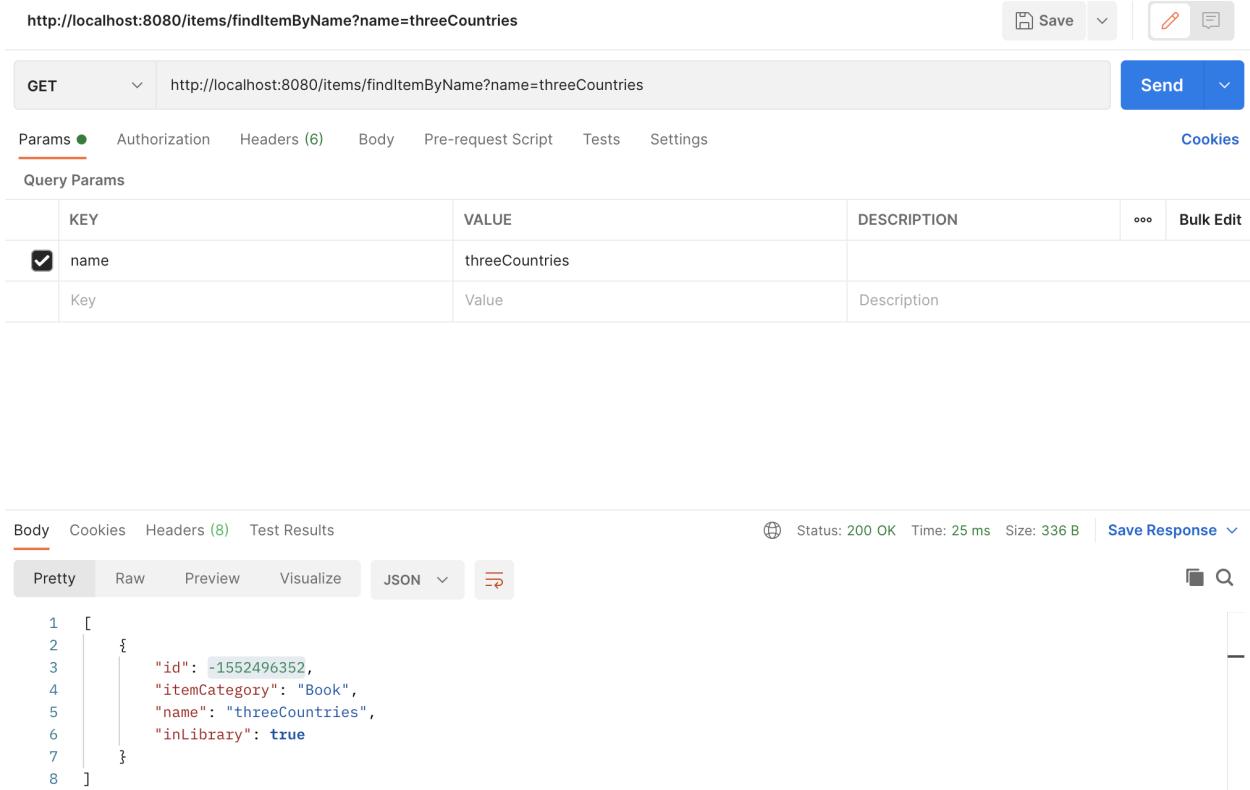
24. updateItem

The screenshot shows the Postman interface for a PUT request to update an item. The URL is `http://localhost:8080/items/updateItem?id=-1552496352&name=threeCountries`. The request method is PUT, and the URL is `http://localhost:8080/items/updateItem?id=-1552496352&name=threeCountries`. The 'Params' tab is selected, showing two parameters: 'id' with value '-1552496352' and 'name' with value 'threeCountries'. The 'Body' tab shows a JSON response with the following structure:

```
1 {  
2   "id": -1552496352,  
3   "itemCategory": "Book",  
4   "name": "threeCountries",  
5   "inLibrary": true  
6 }
```

The status bar at the bottom indicates: Status: 200 OK Time: 18 ms Size: 334 B Save Response.

25. findItemByName



http://localhost:8080/items/findItemByName?name=threeCountries

GET http://localhost:8080/items/findItemByName?name=threeCountries Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	name	threeCountries			
	Key	Value	Description		

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ↻

Status: 200 OK Time: 25 ms Size: 336 B Save Response ↻

```
1 [  
2 {  
3   "id": -1552496352,  
4   "itemCategory": "Book",  
5   "name": "threeCountries",  
6   "inLibrary": true  
7 }  
8 ]
```

26. `findItemById`

The screenshot shows the Postman interface with the following details:

- URL:** `http://localhost:8080/items/findItemById?id=-1552496352`
- Method:** GET
- Params:** A table showing a single parameter `id` with value `-1552496352`.
- Body:** JSON response (Pretty) showing the item details:

```
1 "id": -1552496352,
2 "itemCategory": "Book",
3 "name": "threeCountries",
4 "inLibrary": true
```
- Headers:** (8)
- Status:** 200 OK
- Time:** 10 ms
- Size:** 334 B

27. getAllItemReservations

The screenshot shows the Postman interface for a GET request to `http://localhost:8080/itemReservations/getItemReservationList`. The 'Params' tab is selected, showing a single query parameter 'Key' with a value of 'Value'. The response body is displayed in a pretty-printed JSON format, showing a list of reservations. Each reservation contains an item and a person.

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 [  
2 {  
3   "id": 774674153,  
4   "personDto": {  
5     "id": -66194669,  
6     "name": "luna",  
7     "address": "joe"  
8   },  
9   "itemDto": {  
10    "id": -2009700544,  
11    "itemCategory": "Book",  
12    "name": "Book",  
13    "...": "..."  
14  }  
15 }  
16 ]
```

28. checkout

http://localhost:8080/itemReservations/checkout?pid=-66194669&itemId=-2009700544&startDate=2022-09-01&startTime... Save Send

POST http://localhost:8080/itemReservations/checkout?pid=-66194669&itemId=-2009700544&startDate=2022-09-01&startTime... Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
pid	-66194669			X
itemId	-2009700544			X
startDate	2022-09-01			X
startTime	00:00:00			X
Key	Value	Description		

Body Cookies Headers (8) Test Results Status: 200 OK Time: 221 ms Size: 534 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 774674153,
3   "personDto": {
4     "id": -66194669,
5     "name": "luna",
6     "address": "joe"
7   },
8   "itemDto": {
9     "id": -2009700544,
10    "itemCategory": "Book",
11    "name": "Book",
12    "inLibrary": false
}
```

29. reserve

http://localhost:8080/itemReservations/reserve?pid=460223120&itemId=-2009700544&startDate=2025-09-01&startTime=... Save Send

POST http://localhost:8080/itemReservations/reserve?pid=460223120&itemId=-2009700544&startDate=2025-09-01&startTime=... Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> pid	460223120			
<input checked="" type="checkbox"/> itemId	-2009700544			
<input checked="" type="checkbox"/> startDate	2025-09-01			
<input checked="" type="checkbox"/> startTime	00:00:00			
<input checked="" type="checkbox"/> endDate	2025-09-02			X
<input checked="" type="checkbox"/> endTime	00:15:00			
Key	Value	Description		

Body Cookies Headers (8) Test Results Status: 200 OK Time: 52 ms Size: 537 B Save Response

Pretty Raw Preview Visualize JSON JSON

```
1 {  
2   "id": -370429415,  
3   "personDto": {  
4     "id": 460223120,  
5     "name": "luna",  
6     "address": "jack"  
7   },  
8   "itemDto": {  
9     "id": -2009700544,  
10    "itemCategory": "Book",  
11    "name": "Book",  
12    "inLibrary": true  
13  }  
}
```

30. renew

The screenshot shows a POST request in Postman to the URL `http://localhost:8080/itemReservations/renew?pid=-66194669&itemId=-2009700544&itemReservationId=774674153&endDate=2022-09-15&endTime=00:00:00`. The request body contains the following JSON:

```
1 {
2     "id": 774674153,
3     "personDto": {
4         "id": -66194669,
5         "name": "luna",
6         "address": "joe"
7     },
8     "itemDto": {
9         "id": -2009700544,
10        "itemCategory": "Book",
11        "name": "Book",
12        "inLibrary": false
13    }
14 }
```

31. returnItem

The screenshot shows the Postman interface with the following details:

- Method:** PUT
- URL:** <http://localhost:8080/itemReservations/return?pid=-66194669&itemId=-2009700544&itemReservationId=774674153&endD...>
- Params:** pid: -66194669, itemId: -2009700544, itemReservationId: 774674153, endDate: 2022-09-12, endTime: 00:00:00
- Body:** (Pretty) JSON response:

```
1 {
2     "id": 774674153,
3     "personDto": {
4         "id": -66194669,
5         "name": "luna",
6         "address": "joe"
7     },
8     "itemDto": {
9         "id": -2009700544,
10        "itemCategory": "Book",
11        "name": "Book",
12        "inLibrary": true
13    }
}
```

32. getItemReservedByPerson

http://localhost:8080/itemReservations/getItemReservedByPerson?pid=-66194669

Save  

GET <http://localhost:8080/itemReservations/getItemReservedByPerson?pid=-66194669> Send

Params  Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> pid	-66194669			
Key	Value	Description		

Body Cookies Headers (8) Test Results Status: 200 OK Time: 20 ms Size: 327 B Save Response  

Pretty Raw Preview Visualize JSON 

```
1 [ ]  
2 {  
3   "id": -2009700544,  
4   "itemCategory": "Book",  
5   "name": "Book",  
6   "inLibrary": false  
7 }  
8 ]
```

33.getPersonsReserveItem

The screenshot shows the Postman application interface. At the top, there is a header bar with a save button and other icons. Below it, the URL `http://localhost:8080/itemReservations/getPersonsReserveItem?itemId=-2009700544` is entered into the address bar, along with a `GET` method. To the right of the URL is a `Send` button.

The main area has tabs for `Params`, `Authorization`, `Headers (6)`, `Body`, `Pre-request Script`, `Tests`, and `Settings`. The `Params` tab is currently selected. It shows a table with a single row for the `itemId` parameter, which is checked and has a value of `-2009700544`. There is also a row for a `Key` with a `Value` and a `Description` column.

Below the table, there are tabs for `Body`, `Cookies`, `Headers (8)`, and `Test Results`. The `Body` tab is selected and displays the JSON response. The response is shown in a pretty-printed format:

```
1 [  
2 {  
3   "id": -66194669,  
4   "name": "luna",  
5   "address": "joe"  
6 }  
7 ]
```

At the bottom right of the body panel, there are status details: `Status: 200 OK`, `Time: 13 ms`, `Size: 301 B`, and a `Save Response` button.

34. getItemReservationsByPersonAndItem

http://localhost:8080/itemReservations/getItemReservationsByPersonAndItem?pid=-66194669&itemId=-2009700544

Save  

Send 

GET <http://localhost:8080/itemReservations/getItemReservationsByPersonAndItem?pid=-66194669&itemId=-2009700544>

Params  Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> pid	-66194669			
<input checked="" type="checkbox"/> itemId	-2009700544			
Key	Value	Description		

Body Cookies Headers (8) Test Results Status: 200 OK Time: 16 ms Size: 536 B Save Response  

Pretty Raw Preview Visualize JSON 

```
1 [ {  
2   "id": 774674153,  
3   "personDto": {  
4     "id": -66194669,  
5     "name": "luna",  
6     "address": "joe"  
7   },  
8   "itemDto": {  
9     "id": -2009700544,  
10    "itemCategory": "Book",  
11    "name": "Book",  
12  }  
}
```

35. cancelReservation

The screenshot shows the Postman interface for a DELETE request to cancel a reservation. The URL is `http://localhost:8080/itemReservations/cancelReservation?pid=460223120&itemId=-2009700544&itemReservationId=-370...`. The request method is set to `DELETE`. The `Params` tab is selected, showing three query parameters: `pid` (value: 460223120), `itemId` (value: -2009700544), and `itemReservationId` (value: -370429415). The `Body` tab is visible but empty. The response status is 200 OK, time: 35 ms, size: 257 B, and the response body is `true`.

DELETE http://localhost:8080/itemReservations/cancelReservation?pid=460223120&itemId=-2009700544&itemReservationId=-370...

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
pid	460223120			
itemId	-2009700544			
itemReservationId	-370429415			
Key	Value	Description		

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

Status: 200 OK Time: 35 ms Size: 257 B Save Response

1 true

36. getAllLibrarian

The screenshot shows the Postman interface with the following details:

- Request URL:** `http://localhost:8080/librarians/librarianList`
- Method:** GET
- Headers:** Authorization, Headers (6), Body, Pre-request Script, Tests, Settings
- Query Params:** Key: Value, Description
- Body:** Status: 200 OK, Time: 85 ms, Size: 357 B
- Response:** A JSON array containing one element, representing a librarian object with fields: id, name, address, businessHourDtos, and headLibrarian.

```
1 [  
2   {  
3     "id": -2053678927,  
4     "name": "HEAD",  
5     "address": "LibraryAddress",  
6     "businessHourDtos": [],  
7     "headLibrarian": true  
8   }  
9 ]
```

37. `createLibrarian`

The screenshot shows the Postman interface for a POST request to `http://localhost:8080/librarians/createLibrarian?headLibrarianId=-2053678927&name=joe&address=luna&isHead=false`. The request is set to `POST` and has `Params` selected in the header. The query parameters are listed below:

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> headLibrarianId	-2053678927			
<input checked="" type="checkbox"/> name	joe			
<input checked="" type="checkbox"/> address	luna			X
<input checked="" type="checkbox"/> isHead	false			
Key	Value	Description		

In the response section, the status is `200 OK`, time is `42 ms`, size is `346 B`, and the response body is displayed in Pretty format:

```
1 {  
2   "id": 1611049724,  
3   "name": "joe",  
4   "address": "luna",  
5   "businessHourDtos": null,  
6   "headLibrarian": false  
7 }
```

38. getLibrarianById

The screenshot shows the Postman interface for a GET request to `http://localhost:8080/librarians/getLibrarianById?id=1611049724`. The request parameters are listed in the 'Params' tab, with 'id' set to '1611049724'. The response body is displayed in 'Pretty' JSON format:

```
1 {  
2   "id": 1611049724,  
3   "name": "joe",  
4   "address": "luna",  
5   "businessHourDtos": [],  
6   "headLibrarian": false  
7 }
```

The status of the response is 200 OK, with a time of 13 ms and a size of 344 B.

39. assignBusinessHour

The screenshot shows a POST request in Postman to the URL `http://localhost:8080/librarians/assignBusinessHour?headLibrarianId=-2053678927&librarianId=1611049724&dayOfWeek=5`. The request method is `PUT`. The `Params` tab is selected, showing three query parameters: `headLibrarianId` (value: `-2053678927`), `librarianId` (value: `1611049724`), and `dayOfWeek` (value: `5`). The `Body` tab is active, displaying the JSON response:

```
2 {"id": 1611049724, "name": "joe", "address": "luna", "businessHourDtos": [  { "id": 5, "dayOfWeek": "Friday", "startTime": "00:00:00", "endTime": "23:59:59" } ], "headLibrarian": false ..}
```

40.unassignBusinessHour

The screenshot shows the Postman interface with the following details:

- URL:** `http://localhost:8080/librarians/unassignBusinessHour?headLibrarianId=-2053678927&librarianId=1611049724&dayOfWeek=5`
- Method:** PUT
- Headers:** (7)
- Body:** (empty)
- Tests:** (empty)
- Settings:** (empty)
- Query Params:**

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> headLibrarianId	-2053678927			
<input checked="" type="checkbox"/> librarianId	1611049724		X	
<input checked="" type="checkbox"/> dayOfWeek	5			
- Body:** (Pretty, Raw, Preview, Visualize, JSON, `curl`)
Status: 200 OK Time: 25 ms Size: 344 B Save Response
- JSON Response:**

```
1 {  
2   "id": 1611049724,  
3   "name": "joe",  
4   "address": "luna",  
5   "businessHourDtos": [],  
6   "headLibrarian": false  
7 }
```

41. updateIsHeadLibrarian

The screenshot shows a POST request in the Postman interface. The URL is `http://localhost:8080/librarians/updateIsHeadLibrarian?headLibrarianId=1611049724&librarianId=-2053678927`. The request body contains the following JSON:

```
1 "id": -2053678927,
2 "name": "HEAD",
3 "address": "LibraryAddress",
4 "businessHourDtos": [],
5 "headLibrarian": true
```

The response status is 200 OK, with a time of 19 ms and a size of 355 B.

42. fireLibrarian

The screenshot shows the Postman application interface. At the top, the URL is `http://localhost:8080/librarians/fireLibrarian?headLibrarianId=-2053678927&librarianId=1611049724`. Below the URL, there are buttons for Save, Edit, and Delete. The method is set to **DELETE**, and the target URL is the same as the header. The **Params** tab is selected, showing two query parameters: `headLibrarianId` with value `-2053678927` and `librarianId` with value `1611049724`. There are also sections for Headers, Body, Pre-request Script, Tests, Settings, Cookies, and a Bulk Edit table.

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> headLibrarianId	-2053678927			
<input checked="" type="checkbox"/> librarianId	1611049724			
Key	Value	Description		

Below the Params section, there are tabs for Body, Cookies, Headers (8), and Test Results. The Headers tab is selected, showing 8 items. The Test Results tab is selected, showing a status of 200 OK, time of 62 ms, size of 344 B, and a **Save Response** button. The Body tab is selected, displaying a JSON response:

```
1
2   "id": 1611049724,
3   "name": "joe",
4   "address": "luna",
5   "businessHourDtos": [],
6   "headLibrarian": false
7
```

43. deleteUserById

The screenshot shows the Postman interface for a DELETE request to the endpoint `http://localhost:8080/librarians/deleteUser?id=-2053678927&uid=-1346660120`. The request method is set to `DELETE`. The query parameters `lid` and `uid` are checked and have values `-2053678927` and `-1346660120` respectively. The response status is `200 OK` with a size of `257 B`.

Request URL: `http://localhost:8080/librarians/deleteUser?id=-2053678927&uid=-1346660120`

Method: `DELETE`

Query Params:

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> lid	-2053678927			
<input checked="" type="checkbox"/> uid	-1346660120			
Key	Value	Description		

Response Status: `200 OK` Time: `21 ms` Size: `257 B`

Body: `1 true`

44. createUsers

The screenshot shows the Postman interface for a POST request to `http://localhost:8080/users/createUser?name=joe&address=luna&isLocal=true`. The 'Params' tab is selected, showing query parameters: name (joe), address (luna), and isLocal (true). The response body is a JSON object with fields id, name, address, onlineAccountDto, and local.

POST http://localhost:8080/users/createUser?name=joe&address=luna&isLocal=true

Params ● Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	name	joe			
<input checked="" type="checkbox"/>	address	luna			
<input checked="" type="checkbox"/>	isLocal	true			X
	Key	Value	Description		

Body Cookies Headers (8) Test Results

Status: 200 OK Time: 494 ms Size: 338 B Save Response

Pretty Raw Preview Visualize JSON ↻

```
1 "id": -1346660120,
2 "name": "luna",
3 "address": "joe",
4 "onlineAccountDto": null,
5 "local": true
```

45. getAllUsers

The screenshot shows the Postman application interface. At the top, the URL `http://localhost:8080/users/userList` is entered. Below the URL, there are tabs for `GET`, `POST`, and `PUT`. The `GET` tab is selected. To the right of the URL, there are buttons for `Save`, `Send`, and `Cancel`.

The main area shows the request configuration. Under the `Params` tab, there is a table for `Query Params`:

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description	...	

Below the table, there are tabs for `Body`, `Cookies`, `Headers (8)`, and `Test Results`. The `Body` tab is selected. On the right, the status bar shows `Status: 200 OK Time: 62 ms Size: 340 B` and a `Save Response` button.

The `Body` section displays the JSON response:

```
1 [  
2   {  
3     "id": -1346660120,  
4     "name": "luna",  
5     "address": "joe",  
6     "onlineAccountDto": null,  
7     "local": true  
8   }  
9 ]
```

46. updateIsLocal

The screenshot shows the Postman interface for a PUT request to `http://localhost:8080/users/updateIsLocal?uid=-1346660120&isLocal=false`. The request is set to `PUT` and the URL is specified. The `Params` tab is selected, showing query parameters: `uid` with value `-1346660120` and `isLocal` with value `false`. A table below lists these parameters. The response tab shows a status of `200 OK` with a size of `339 B`.

PUT http://localhost:8080/users/updateIsLocal?uid=-1346660120&isLocal=false

Params ● Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	uid	-1346660120			
<input checked="" type="checkbox"/>	isLocal	false			
	Key	Value	Description		

Body Cookies Headers (8) Test Results

Status: 200 OK Time: 20 ms Size: 339 B Save Response ▾

Pretty Raw Preview Visualize JSON ↻

```
1 "id": -1346660120,
2 "name": "luna",
3 "address": "joe",
4 "onlineAccountDto": null,
5 "local": false
6
7
```

47. updateAddress

The screenshot shows the Postman interface for a PUT request to update a user's address. The URL is `http://localhost:8080/users/updateAddress?uid=-1346660120&address=J street`. The request method is PUT, and the body contains the following JSON:

```
1 {
2   "id": -1346660120,
3   "name": "luna",
4   "address": "J street",
5   "onlineAccountDto": null,
6   "local": false
7 }
```

The response status is 200 OK, with a time of 59 ms and a size of 344 B.

48. updateOnlineAccount

The screenshot shows a POST request in Postman to the URL `http://localhost:8080/users/updateOnlineAccount?uid=-1346660120&username=jack&password=114514&email=qqmail`. The request body contains the following JSON:

```
1 {
2     "id": -1346660120,
3     "name": "luna",
4     "address": "J street",
5     "onlineAccountDto": {
6         "id": -1584444229,
7         "username": "jack",
8         "email": "qqmail",
9         "userId": -1346660120
10    },
11    "local": false
12 }
```

The response status is 200 OK, with a time of 69 ms and a size of 414 B.

49. getUserById

The screenshot shows the Postman interface with the following details:

- URL:** `http://localhost:8080/users/getUserById?uid=-1346660120`
- Method:** GET
- Params:** uid: -1346660120
- Body:** (Pretty) JSON response:

```
1 {
2     "id": -1346660120,
3     "name": "luna",
4     "address": "J street",
5     "onlineAccountDto": {
6         "id": -158444229,
7         "username": "jack",
8         "email": "qqmail",
9         "userId": -1346660120
10    },
11    "local": false
12 }
```
- Headers:** (8)
- Test Results:** Status: 200 OK, Time: 63 ms, Size: 414 B

50.login

The screenshot shows the Postman application interface for a login request.

Request URL: `http://localhost:8080/users/login?username=jack&password=114514&email=qqmail`

Method: POST

Params: (selected)

KEY	VALUE	DESCRIPTION	...	Bulk Edit
username	jack			
password	114514			
email	qqmail			

Body: (Pretty) `success:-1346660120`

Response Headers: Status: 200 OK, Time: 31 ms, Size: 272 B

Response Body: (Pretty) `success:-1346660120`

51. checkUsernameExistence

http://localhost:8080/users/checkUsernameExistence?username=jack

Save Send

POST http://localhost:8080/users/checkUsernameExistence?username=jack

Params ● Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> username	jack			
Key	Value	Description		

Body Cookies Headers (8) Test Results

Status: 200 OK Time: 14 ms Size: 258 B Save Response

Pretty Raw Preview Visualize JSON

1 false

52. signUp

The screenshot shows a POST request to `http://localhost:8080/users/signUpOnlineAccount?uid=-1346660120&username=jack&password=11111&email=qqmail`. The request includes the following parameters:

KEY	VALUE	DESCRIPTION
uid	-1346660120	
username	jack	
password	11111	
email	qqmail	
Key	Value	Description

The response status is 200 OK with a message: "1 username already exist".

53. initialize

The screenshot shows a GET request to `http://localhost:8080/init`. The request includes the following parameters:

KEY	VALUE	DESCRIPTION
Key	Value	Description

The response status is 200 OK with a message: "1 Success!".