Select a Topic ⌄

Private Messages

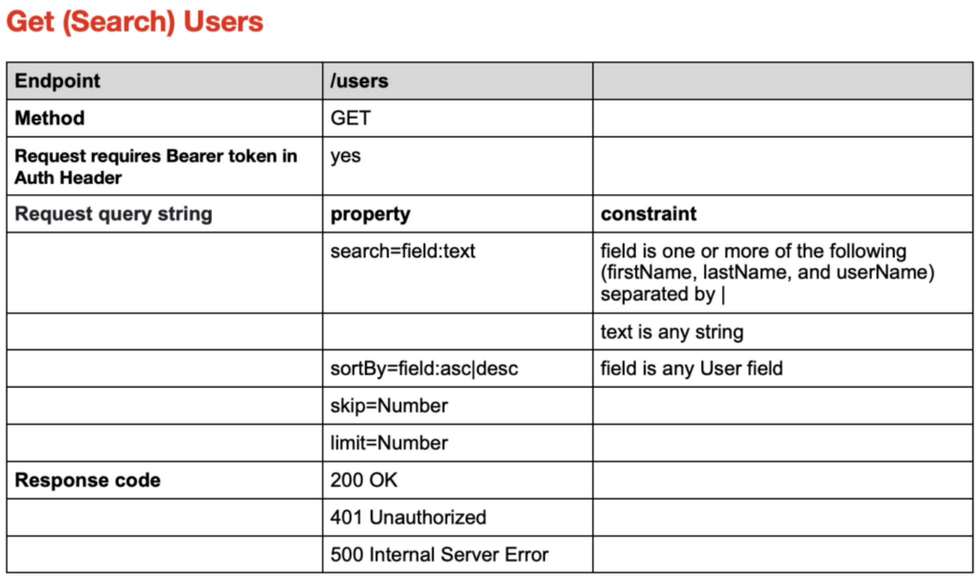
20.2.2025

For your next assignment you will modify your app so that users can send and display private text messages. In this tutorial, let's refer to the user who's logged in as Alice and call the user whom Alice wishes to send a private message, Jack.

Create a Search Bar

Create a search bar and button that allows Alice to search for other users by first name, last name or username. Place the search bar somewhere in your Main view.

To perform the search, use the **GET /users** endpoint which includes **search**, **sortBy**, **skip**, and **limit** query properties.



Get (Search) Users

Endpoint	**/users**	
Method	GET	
Request requires Bearer token in Auth Header	yes	
Request query string	**property**	**constraint**
	search=field:text	field is one or more of the following (firstName, lastName, and userName) separated by |
		text is any string
	sortBy=field:asc|desc	field is any User field
	skip=Number	
	limit=Number	
Response code	200 OK	
	401 Unauthorized	
	500 Internal Server Error	

This endpoint will return an array of objects, with each object containing a user's id, name, and username; looking something like this.

```
[
    {
        "_id": "67ad530e8c010766d55b5e17",
        "firstName": "joe",
        "userName": "joey",
        "lastName": "doe"
    },
    {
        "_id": "67a9e5170bfef7a9c86e8568",
        "firstName": "Joe",
        "userName": "joe",
        "lastName": "Smith"
    }
]
```

Create a List of RouterLinks

We need to display the list of users on the screen so Alice can choose a user to send a private message to. When Alice selects, say Jack, we want a new component to be displayed that shows Alice the private messages that have been sent between her and Jack and allows her to send a new private message to Jack.

One way to do this is to create a list of `<RouterLink>` elements, one for each user in the array. Each <RouterLink> element will then redirect Alice to a path dedicated to the particular user she's chosen.

For example, if Jack's user id is 67a9e5170bfef7a9c86e8568, when Alice clicks on the <RouterLink> associated with Jack, then Alice can be redirect to a path like `'/user/67a9e5170bfef7a9c86e8568'.`

You can utilize Vue's `v-for` directive to quickly generate a list of `<RouterLink>` elements using the data in the array. Since the `v-for` directive provides you access to the fields of the objects in the array, you can use a template literal to set the <RouterLink>'s `to` property. You may also want to include user's username or first and last name in the path as query strings.

```
<RouterLink v-for="user in users" :to="`/user/${user._id}?name=${user.userName}`">
    {{ user.firstName }} {{ user.lastName }}
</RouterLink>
```

Create a Route Record

Inside your *router/index.js* file, create a route record that displays a new component when the user is redirected by the <RouterLink> elements to `"/user/:userId"`.

Your router record will undoubtably be different than the one below, but should have similar `path` and `props` properties. In the next step you will create a component that display private messages and allow the user to send private messages. Load this new component using this new route record. For example, in the record below, I call the new component PrivateMessage and mount it in my layout view's <RouterView> named *middlePanel*.

```
{
    path:'/user/:userId',
    components: {
        topPanel:Menu,
        middlePanel:PrivateMessage,
    },
    props:true
}
```

Note: the `props` property instructs the router to pass the `userId` parameter into the components as a prop.

Create a Private Messages Component

Create a new component that displays the private messages between the user who is logged in and another user and allow the user who's logged in to send a private message to the other user.

Since this component will be loaded by the route record you just created, it will have at its disposal a **prop** named `userId` as well as any query strings that are on the `route` object. Note that you must define a **prop** named `userId` in order to have access to the prop value. You also have to import and call `useRoute()` in order to get the query strings from the route.

To obtain the private messages between the user who's logged in and another users, use the **GET /messages/:userId** endpoint.

In the example below, the URL to retrieve all of the messages between Alice (who's logged in) and Jack (who's id is 67a9e5170bfef7a9c86e8568) is

`https://hap-app-api.azurewebsites.net/messages/67a9e5170bfef7a9c86e8568`

Note that when constructing the URL for this endpoint you **do not** include a **:** before the user's id.

Get Private Messages

Endpoint	**/messages/:userId**	
Method	GET	
Request requires Bearer token in Auth Header	yes	
Parameter	userId	User._id
Query string	**property**	**constraint**
	after=DateTime	ISO 8601 string
	before=DateTime	ISO 8601 string
	limit=Number	
Response code	200 OK	[ ]
	401 Unauthorized	
	500 Internal Server Error	

The endpoint will return an array of objects similar to the ones below.

```
[
    {
        "text": "Not much Joe.  What's up with you?",
        "updatedAt": "2025-02-20T20:30:07.861Z",
        "messageId": "67b790cfef3c5e3af0ac3b9d",
        "senderId": "67b78ffbef3c5e3af0ac3b80",
        "senderName": "Jack Smith",
        "receiverId": "67a9e5170bfef7a9c86e8568",
        "receiverName": "Joe Smith"
    },
    {
        "text": "What's up Jack?",
        "updatedAt": "2025-02-20T20:29:24.244Z",
        "messageId": "67b790a4ef3c5e3af0ac3b8e",
        "senderId": "67a9e5170bfef7a9c86e8568",
        "senderName": "Joe Smith",
        "receiverId": "67b78ffbef3c5e3af0ac3b80",
        "receiverName": "Jack Smith"
    }
]
```

For bonus points, when displaying the private messages, only fetch enough messages to fill the viewable area of the scrollable container (perhaps a little more) and when the user scrolls up or down, fetch accordingly. Like the **GET /messages** endpoint, the **GET /messages/:userId** endpoint includes **before**, **after**, and **limit** query strings.

In addition, a new **GET /messages/:userId/count** endpoint returns the number of private messages that have been sent to the user whose _id is **userId**. The optional query string **after=DateTime** returns the number of private messages sent after the specified ISO 8601 DateTime string.

Get Private Message Count

Endpoint	/messages/:userId/count	
Method	GET	
Request requires Bearer token in Auth Header	yes	
Parameter	userId	User._id
Query string	**property**	**constraint**
	after=DateTime	ISO 8601 string
Response code	200 OK	{ total }
	401 Unauthorized	
	500 Internal Server Error	

Create a Post Private Message Form

Create a form (or component if you wish) that allows the user who's logged into to send a new private message. The **POST /message/:userId** endpoint allows the current user to send a private message to the user whose id is **userId**.

Post Private Message

Endpoint	/message/:userId			
Method	POST			
Request requires Bearer token in Auth Header	yes			
Parameter	userId	User._id		
Request body	**property**	**type**	**required**	**constraint**
	text	string	yes	max length: 280
Response code	201 Created	Message { }		
	400 Bad request			
	401 Unauthorized			
	500 Internal Server Error			

Search...