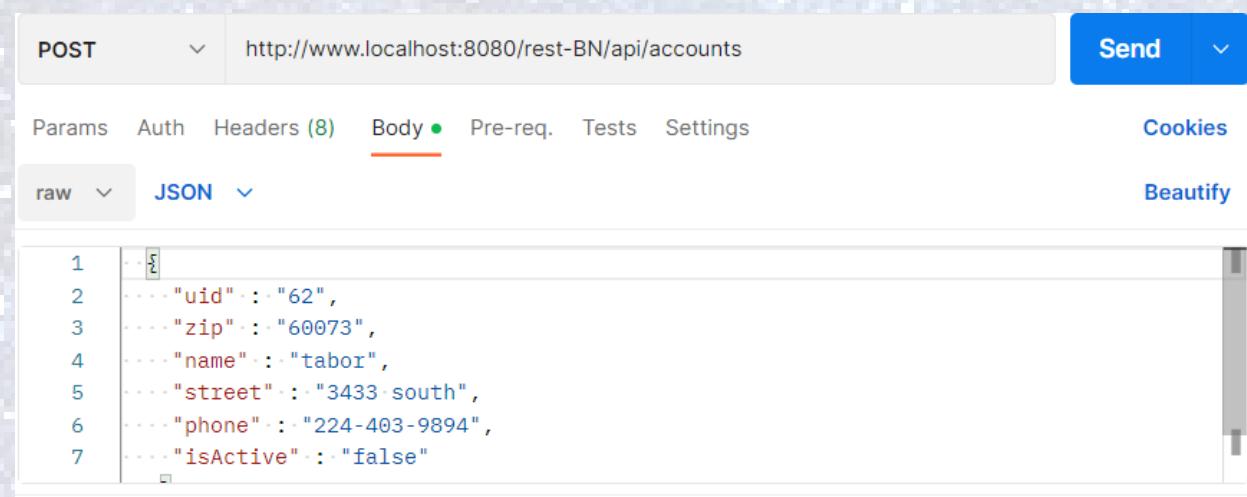


BuyNothing is an application which allows users to transact gifts, send asks and give thanks to other users on the platform. It essentially works like a CashApp or Venmo, the only difference is money doesn't get traded. My task was to build the backend server that will support the application infrastructure. This will consist of creating, updating, and deleting accounts on the app, generating gift transactions that can be seen from both users, allowing users to make ask requests, creating thank you messages, and notes that can be stored for the users convenience. Additionally, there will be an admin point of view of all the data involved. As admin, you can see all the accounts, gifts, thanks, and notes.

The framework of the application is built on RESTful APIs. When the server is up, users can complete actions through http requests with additional information such as path and body parameters. The API was built using the Java programming language on the Eclipse IDE. Here are some things one can do with this backend server:

ACCOUNTS:

Creating Accounts:



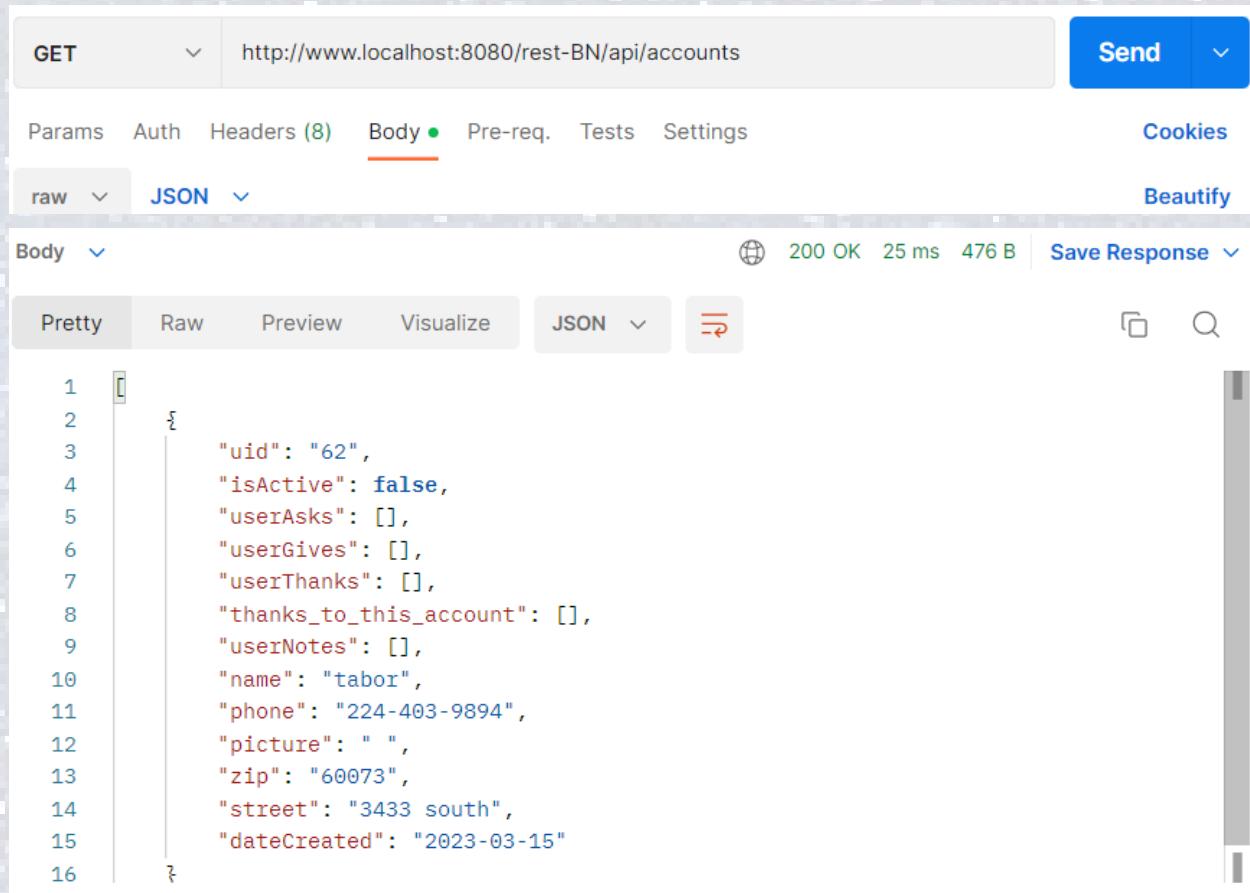
POST http://www.localhost:8080/rest-BN/api/accounts

Params Auth Headers (8) **Body** ● Pre-req. Tests Settings Cookies Beautify

raw JSON

```
1 {  
2   "uid": "62",  
3   "zip": "60073",  
4   "name": "tabor",  
5   "street": "3433 south",  
6   "phone": "224-403-9894",  
7   "isActive": "false"  
}
```

Getting Accounts:



The screenshot shows the Postman application interface. At the top, there is a header bar with 'GET' selected as the method, the URL 'http://www.localhost:8080/rest-BN/api/accounts', and a 'Send' button. Below the header, there are tabs for 'Params', 'Auth', 'Headers (8)', 'Body' (which is currently active), 'Pre-req.', 'Tests', and 'Settings'. To the right of these tabs are 'Cookies' and 'Beautify' buttons. Under the 'Body' tab, there are dropdowns for 'raw' and 'JSON', and a 'Beautify' button. Below these dropdowns, there are buttons for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON'. On the far right of this row are 'Copy' and 'Search' icons. The main content area displays a JSON response with line numbers on the left. The JSON object contains fields such as uid, isActive, userAsks, userGives, userThanks, thanks_to_this_account, userNotes, name, phone, picture, zip, street, and dateCreated.

```
1 {  
2     "uid": "62",  
3     "isActive": false,  
4     "userAsks": [],  
5     "userGives": [],  
6     "userThanks": [],  
7     "thanks_to_this_account": [],  
8     "userNotes": [],  
9     "name": "tabor",  
10    "phone": "224-403-9894",  
11    "picture": " ",  
12    "zip": "60073",  
13    "street": "3433 south",  
14    "dateCreated": "2023-03-15"  
15 }  
16
```

Activate Account:

The screenshot shows three requests in the Postman interface, each with a status of 200 OK.

- Request 1 (Top):** Method: PUT, URL: http://www.localhost:8080/rest-BN/api/accounts/62/activate. Body tab is selected. Response Body: "1".
- Request 2 (Middle):** Method: PUT, URL: http://www.localhost:8080/rest-BN/api/accounts/62/activate. Body tab is selected. Response Body: JSON response (Pretty view) showing account details with "isActive": true. A black arrow points from the text "Admin can deactivate accounts by using "deactivate" in the path params instead of "activate"" to the "isActive": true line in the JSON response.
- Request 3 (Bottom):** Method: PUT, URL: http://www.localhost:8080/rest-BN/api/accounts/62/deactivate. Body tab is selected. Response Body: JSON response (Pretty view) showing account details with "isActive": false.

- Admin can deactivate accounts by using “deactivate” in the path params instead of “activate”

Delete Account:

The screenshot shows the Postman interface with two requests. The top request is a GET to `http://www.localhost:8080/rest-BN/api/accounts/62`. The bottom request is also a GET to the same URL. Both requests have 'Headers (8)' selected. The bottom response shows a JSON body with the message 'Account is Gone!' and an array containing a single element: `[{"id": 62}]`. A black arrow points from the text 'Account is Gone!' to the array element.

All accounts in current environment:

GET http://www.localhost:8080/rest-BN/api/accounts/62

Params Auth Headers (8) Body **Pre-req.** Tests Settings Cookies Beautify

raw JSON

Account is Gone!

Body **Pretty** **Raw** Preview Visualize JSON **Save Response**

1 []

200 OK 6 ms 157 B

Updating Account:

The screenshot shows a POST request in Postman. The URL is `http://www.localhost:8080/rest-BN/api/accounts/62`. The "Body" tab is selected, and the content type is set to "JSON". The JSON payload is as follows:

```
1 {  
2   "uid": "62",  
3   "zip": "60616",  
4   "name": "tabor",  
5   "street": "123 first street",  
6   "phone": "224-403-9894",  
7   "isActive": "false"  
}
```

- Changing zip and street

API response:

The screenshot shows the API response in Postman. The status code is 204 No Content, and the response body is empty.

Update:

The screenshot shows the updated account details in Postman. The status code is 200 OK, and the response body is a JSON object containing all account fields, including the updated "zip" and "street" values.

```
1 {  
2   "uid": "62",  
3   "isActive": false,  
4   "userAsks": [],  
5   "userGives": [],  
6   "userThanks": [],  
7   "thanks_to_this_account": [],  
8   "userNotes": [],  
9   "name": "tabor",  
10  "phone": "224-403-9894",  
11  "picture": " ",  
12  "zip": "60616",  
13  "street": "123 first street",  
14  "dateCreated": "2023-03-16"  
15  "  
16 }
```

ASKS:

- This allows a user to send an ask request for a favor/gift
- Consists of a uid (user id) of the user sending the ask, type (type of the message), and the description of the ask

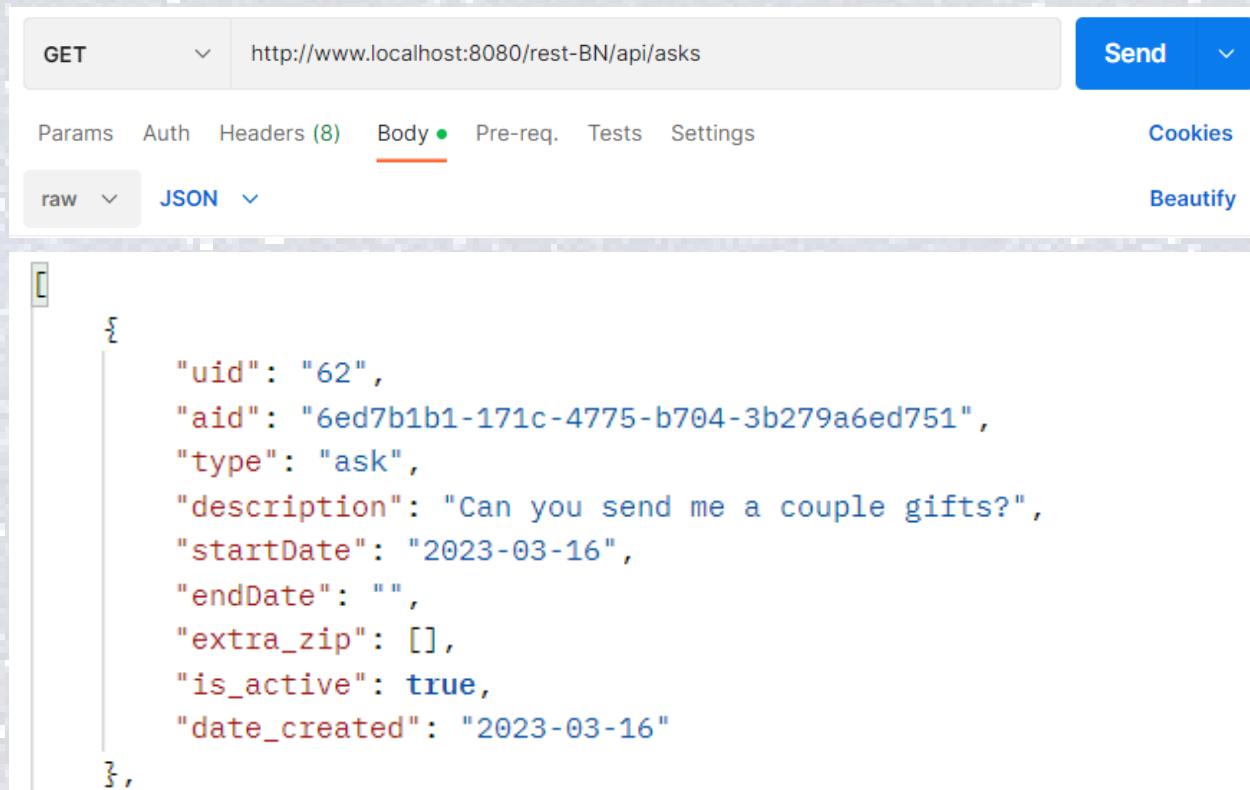
Create Ask:

The screenshot shows a POST request in the Postman application. The URL is `http://www.localhost:8080/rest-BN/api/accounts/62/asks`. The Body tab is selected, showing the following JSON payload:

```
6  {
7    "uid": "62",
8    "type": "ask",
9    "description": "Can you send me a couple gifts?"
10   }
11
12
```

Getting User Asks:

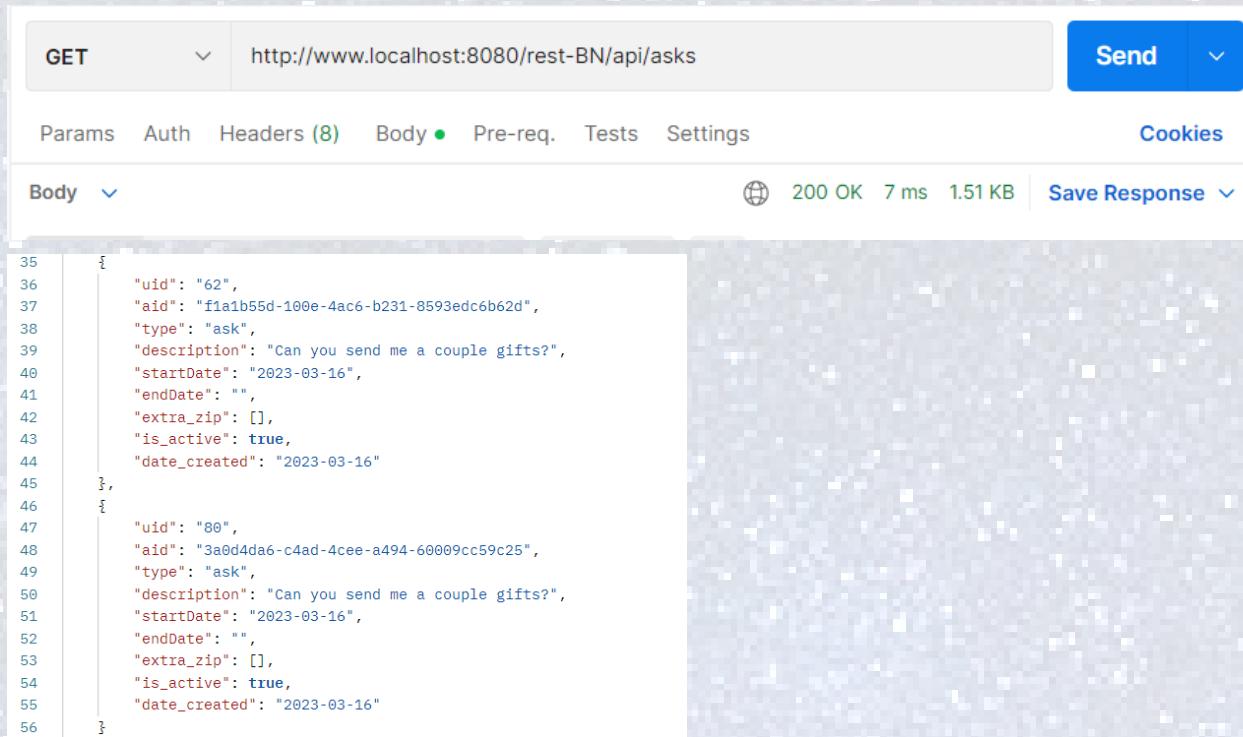
- aid is a unique ask id for every ask
- Start date is when the ask request is active (default is current date) and date created is when the ask was constructed



The screenshot shows the Postman application interface. A GET request is being made to the URL `http://www.localhost:8080/rest-BN/api/asks`. The "Body" tab is selected, displaying the following JSON payload:

```
{  
    "uid": "62",  
    "aid": "6ed7b1b1-171c-4775-b704-3b279a6ed751",  
    "type": "ask",  
    "description": "Can you send me a couple gifts?",  
    "startDate": "2023-03-16",  
    "endDate": "",  
    "extra_zip": [],  
    "is_active": true,  
    "date_created": "2023-03-16"  
},
```

Getting All Asks in Application:



GET http://www.localhost:8080/rest-BN/api/asks

Params Auth Headers (8) Body ● Pre-req. Tests Settings Cookies

Body ▾

200 OK 7 ms 1.51 KB Save Response ▾

```
35  {
36      "uid": "62",
37      "aid": "f1a1b55d-100e-4ac6-b231-8593edc6b62d",
38      "type": "ask",
39      "description": "Can you send me a couple gifts?",
40      "startDate": "2023-03-16",
41      "endDate": "",
42      "extra_zip": [],
43      "is_active": true,
44      "date_created": "2023-03-16"
45  },
46  {
47      "uid": "80",
48      "aid": "3a0d4da6-c4ad-4cee-a494-60009cc59c25",
49      "type": "ask",
50      "description": "Can you send me a couple gifts?",
51      "startDate": "2023-03-16",
52      "endDate": "",
53      "extra_zip": [],
54      "is_active": true,
55      "date_created": "2023-03-16"
56  }
```

- Asks are now stored in the user accounts information ("userAsks")!



```
{
  "uid": "80",
  "isActive": false,
  "userAsks": [
    {
      "uid": "80",
      "aid": "3a0d4da6-c4ad-4cee-a494-60009cc59c25",
      "type": "ask",
      "description": "Can you send me a couple gifts?",
      "startDate": "2023-03-16",
      "endDate": "",
      "extra_zip": [],
      "is_active": true,
      "date_created": "2023-03-16"
    }
  ]
}
```

For more functions and information check out the complete API:

<http://www.cs.iit.edu/~virgil/cs445/mail.spring2022/project-api.html>