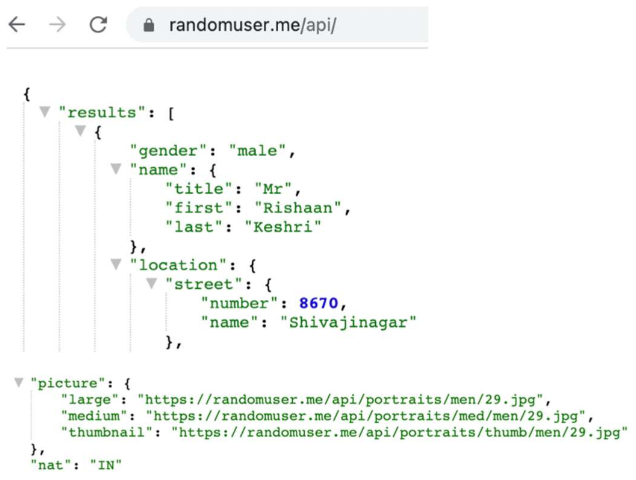
JAVA MINI Assignment 2

Step 1: Get a random user by calling the above API.

API 1: https://randomuser.me/api/



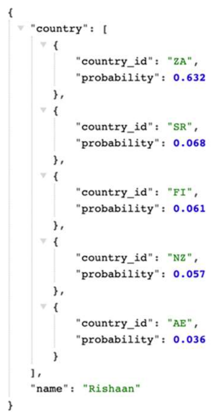
```
{
  "results": [
    {
      "gender": "male",
      "name": {
        "title": "Mr",
        "first": "Rishaan",
        "last": "Keshri"
      },
      "location": {
        "street": {
          "number": 8670,
          "name": "Shivajinagar"
        },

"picture": {
    "large": "https://randomuser.me/api/portraits/men/29.jpg",
    "medium": "https://randomuser.me/api/portraits/med/men/29.jpg",
    "thumbnail": "https://randomuser.me/api/portraits/thumb/men/29.jpg"
},
"nat": "IN"
```

Note: This API will have details like gender, nationality, name(combining first name and last name) and age.

Step 2: Get the nationality of the user by providing the name from step 1

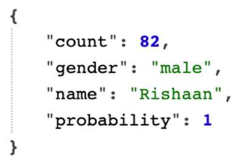
API 2: https://api.nationalize.io/?name=Rishaan



```
{
  "country": [
    {
      "country_id": "ZA",
      "probability": 0.632
    },
    {
      "country_id": "SR",
      "probability": 0.068
    },
    {
      "country_id": "FI",
      "probability": 0.061
    },
    {
      "country_id": "NZ",
      "probability": 0.057
    },
    {
      "country_id": "AE",
      "probability": 0.036
    }
  ],
  "name": "Rishaan"
}
```

Step 3: Get the gender of the user by providing the name from step 1

API 3: https://api.genderize.io/?name=Rishaan



```
{
    "count": 82,
    "gender": "male",
    "name": "Rishaan",
    "probability": 1
}
```

Note: API calls in Step2 & Step3 should be executed in parallel using executor framework

JAVA MINI Assignment 2

Step4:

- Validate if the Nationality ("nat": "IN") received in Step 1,
- Matches any of the list of nationalities received in Step 2, and the gender of the user received in Step 1 matches with the gender received in Step 3.
- If it matches, then the user will be marked as VERIFIED.
- If any of the two criteria nationality(from Step2) and gender(from Step3) does not match with the response in Step 1, the user should be marked as TO_BE_VERIFIED

Step5:

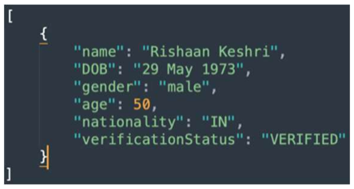
- User data should be saved in DB
  - user_id | name(combining first and last name) | age | gender | dob|nationality | verification_status | date_created | date_modified
  - Feel free to add more values in DB if you wish.Most the data you can get from Step1 except,
- verification_status -> from Step4
- date_created -> default user creation timestamp
- date_modified -> default user creation timestamp
- user_id -> should be auto incremented value

FUNCTIONAL REQUIREMENTS:

You need to implement the below 2 APIs.

1. **CREATE – POST http://localhost:8080/users**
   i. Should support 1 attribute in its request payload.
      1. size=1
      2. Size denotes the no of users that will be created once this API is executed. 5 is the max no.

   ii. Should return the list of saved users as a response.



```
[
    {
        "name": "Rishaan Keshri",
        "DOB": "29 May 1973",
        "gender": "male",
        "age": 50,
        "nationality": "IN",
        "verificationStatus": "VERIFIED"
    }
]
```

2. **GET - http://localhost:8080/users**
   i. Should support 4 query params.
      1. ?sortType=<Name/Age>
      2. ?sortOrder=<EVEN/ODD>
      3. ?limit=5
      4. ?offset=0

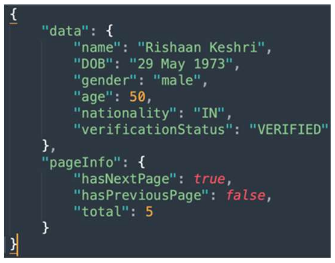
   Based on the sortType and sortOrder the list of users should be returned.

JAVA MINI Assignment 2

Guidelines:

- Code should contain 2 custom Validators to validate the input parameters. The input parameters can be either character like name and numeric like age – NumericValidator & EnglishAlphabetsValidator. Both of these should be created as a Singleton class. And should be instantiated using Factory design pattern based on the parameter type. If it is a limit/offset etc it should be validated through NumericValidator otherwise if it is name, it should be validated through EnglishAlphabetsValidator.
- Limit denotes no of users that should be returned in the list. Value can be from 1-5 (max 5, min 1).
- Offset denotes the number of users that should be skipped from first. Say suppose a database returns a query result with id 1, 2, 3, 4, 5 without limit and offset. Now if we pass limit=3 and offset=2, id 3,4,5 should be returned and id 1 & 2 will be skipped.
- SortType denotes either Name or Age
- SortOrder denotes the order by which the list of users should be arranged.
   i. example 1: sortType: Age & sortOrder: even
      1. all users whose age is even should come in the first irrespective of their names
   ii. example 2: sortType: Name & sortOrder: odd
      1. all users for whom the total characters count for name is odd should come in the first irrespective of age

- SAMPLE RESPONSE TYPE FOR: http://localhost:8080/listUser



```
{
    "data": {
        "name": "Rishaan Keshri",
        "DOB": "29 May 1973",
        "gender": "male",
        "age": 50,
        "nationality": "IN",
        "verificationStatus": "VERIFIED"
    },
    "pageInfo": {
        "hasNextPage": true,
        "hasPreviousPage": false,
        "total": 5
    }
}
```

   - *pageInfo* contains the value required for paginating. Though we are not using anyfrontend here but pageInfo with correct values should be present in response. All the values related to page info depends on total count of users in DB, offset and limit. *total* represents the total count of users present in the database.

- **NOTE: Write Junit Test Cases for below Endpoints, covering all the scenarios. Usage of Mockito for mocking the response is mandatory.**
   - POST: http://localhost:8080/users
   - GET : http://localhost:8080/users

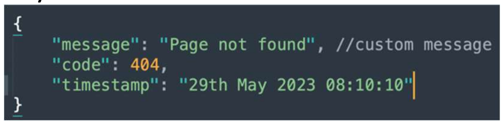
JAVA MINI Assignment 2

Coding practice:

1. To sort the List of users based on sortType and sortOrder use strategy design pattern
2. Use WebClient to call external APIs. 3 different webClient beans should be configured.
   For API1: web client configuration are as follows:
   connection_time_out = 2000
   read_time_out = 2000
   write_time_out = 2000

   For API2: web client configuration are as follows:
   connection_time_out = 1000
   read_time_out = 1000
   write_time_out = 1000
   For API3: web client configuration are as follows:
   connection_time_out = 1000
   read_time_out = 1000
   write_time_out = 1000

3. Error message should be customized. It should follow the below response type for any kind of error.



```
{
    "message": "Page not found", //custom message
    "code": 404,
    "timestamp": "29th May 2023 08:10:10"
}
```

4. Code should satisfy SOLID design pattern. Atleast 3

5. Use of Java 8+ functionalities like streams API is must wherever possible.

6. Package/class/method names should follow the proper naming conventions