

# Java Web Application Assignment

## PURPOSE

---

This assignment is a chance for you to showcase your skills and will hopefully form a good base for discussion further down the process. Here in European Dynamics, we believe that it's for the mutual benefit of both the team and the candidate to get a good taste of one another's way of work and thinking. This assignment is designed to evaluate the candidate skills regarding:

- Web application development
- RESTful web services
- Development in Java
- Enterprise application development (using JavaEE or Spring or anything else)
- Persist data in SQL or NoSQL database
- Building a Java project with Maven

## NOTES

---

- Java version to be used: 8+
- We mostly use JVM, Java and JavaEE(currently rebranded as JakartaEE), but any other framework or library is more than ok
- Even if you don't manage to implement the whole assignment, please do send your work as described in the [Deliverable](#) section.
- You can use Postman ([www.getpostman.com](http://www.getpostman.com)) or any other API client to test your REST API.
- Don't hesitate to contact us for any clarifications.
- You have **7 days** to complete the assignment.

## DELIVERABLE

---

The deliverable should be the project (folder). You can send it as:

- a zip file *or*
- a GitHub (or any other online code repository) link

## EVALUATION

---

The assignment's evaluation will be based on:

- Implementation of core functionality
- Quality engineering
- Code organisation
- Implementation of the *optional* requirements

## OVERVIEW

A customer organisation has requested the creation of a discussion forum for its employees.

The basic front-end has been implemented and there is a need for a basic back-end that will expose a REST API with the basic functionality.

## INSTRUCTIONS

Create a web application that will expose a REST Endpoint that will satisfy the wireframes provided below:

### 1. GET LIST OF USERS

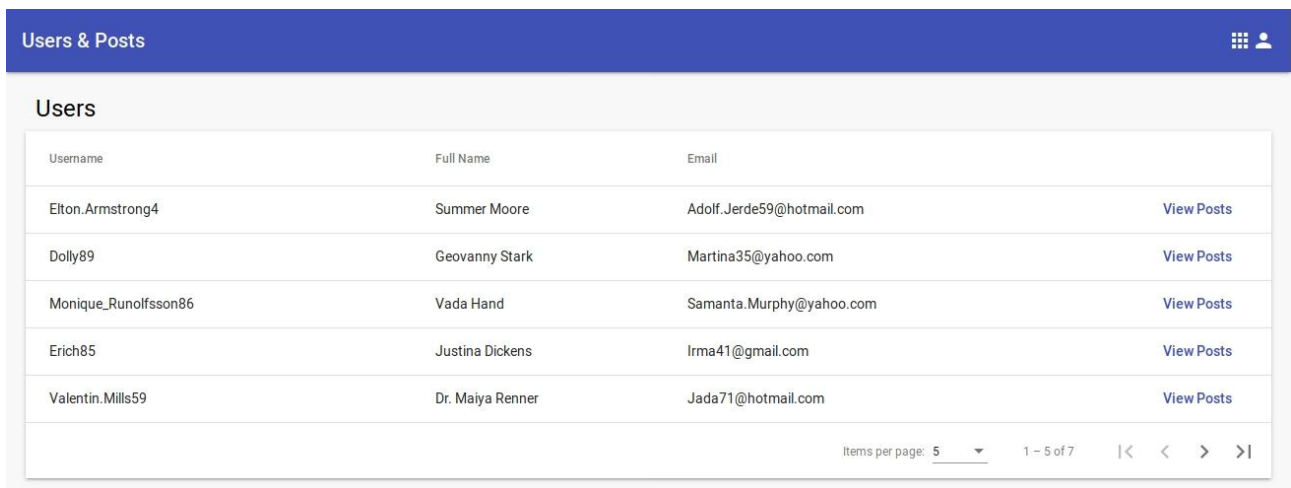
This method should expose users' data that are stored in the database of your choice. In order to fetch all the users' the front-end should make a REST call to the following URL:

**GET** /users

The table in the front-end has the following columns, so be sure to include the following fields in your model:

- Username
- Full Name
- Email

The wireframe that is provided below should be satisfied by your method.



The wireframe shows a web application titled 'Users & Posts'. It features a table with the following data:

Username	Full Name	Email	
Elton.Armstrong4	Summer Moore	Adolf.Jerde59@hotmail.com	<a href="#">View Posts</a>
Dolly89	Geovanny Stark	Martina35@yahoo.com	<a href="#">View Posts</a>
Monique_Runolfsson86	Vada Hand	Samanta.Murphy@yahoo.com	<a href="#">View Posts</a>
Erich85	Justina Dickens	Irma41@gmail.com	<a href="#">View Posts</a>
Valentin.Mills59	Dr. Maiya Renner	Jada71@hotmail.com	<a href="#">View Posts</a>

At the bottom right of the table, there is a pagination control showing 'Items per page: 5', '1 - 5 of 7', and navigation arrows.

Figure 1: List of Users

Note: The [Action](#) column should not be provided by the backend. It should contain a [link](#) with label [View Posts](#) that will navigate the user to the `users/{userId}` route that will be explained later. This is already handled by the front-end.

## OPTIONAL REQUIREMENT

You might also implement [pagination](#) functionality on server-side. In the server-side pagination, the front-end requests only a portion of the data by including the page index and the page size as URL parameters to its request to the REST API (provided that the REST API supports this functionality).

The implementation of **server-side** pagination is *optional*.

In case you wish to implement server-side pagination you can pass the following URL parameters to the previous endpoint:

- [page](#): The page index
- [limit](#): The page size

E.g. [http://{server\\_ip}/users?page=2&limit=5](http://{server_ip}/users?page=2&limit=5)

The data received (payload) from this endpoint will be in the following format:

```
{
  items: [
    {
      id: number,
      name: string,
      email: string,
      avatar: string, // Link to an image stored in your DB
      username: string
    }
  ],
  total: number
};
```

## 2. GET USER WITH HER POSTS

In this task we want to expose 2 methods:

- Get [user](#) details (Name, email, avatar)  
To get the [user](#) data the following URL should be called by the front-end:  
**GET /users/{userId}**

- Get all the posts that user with given `userId` has created  
**GET /users/{userId}/posts**

The [posts](#) can be exposed as a list of items. For each post you should expose its [title](#) and its [creation date](#).

The response of this endpoint will be an array of [post](#) objects. The [post](#) object format is:

```
{
  id: number,
  userId: number,
  createdAt: Date,
  title: string,
  body: string,
}
```

The wireframe that is provided below should be satisfied by the 2 methods described above. One call is made to get the user with his/her details and one second call takes place to get the list of user's posts.

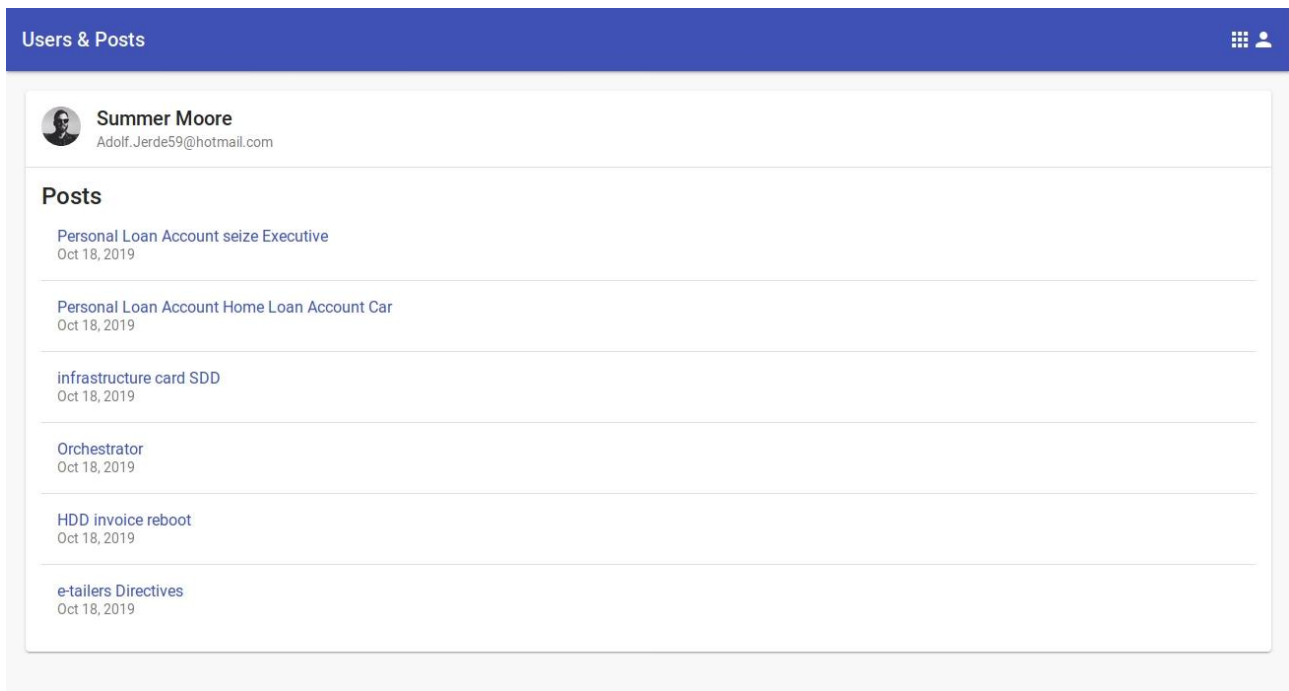


Figure 2: User Posts

### 3. GET A USER'S POST WITH ITS COMMENTS

In this task we want to expose 2 methods:

- Get a **post** (Title, creation date, body) by Id  
**GET /users/{userId}/posts/{postId}**
- Get all the comments related to this post  
**GET /users/{userId}/posts/{postId}/comments**

The **comments** can be displayed as a list of items. For each comment you should display the author's **name** and **avatar**, the comment's **creation date** and the comment's **body**.

The response of this endpoint will be an array of **comment** objects. The **comment** object format is:

```
{
  id: number,
  postId: number,
  createdAt: Date,
  email: string,
  avatar: string,    // Link to an image stored in your DB
  name: string,
  body: string,
}
```

The wireframe that is provided below should be satisfied by the 2 methods described above. One call is made to get the post and one second call takes place to get the list of comments for this post.

Users & Posts

Post

Personal Loan Account seize Executive

Oct 18, 2019

archive Investment Account

Comments (4)

Creola Wilkinson

Oct 17, 2019

integrate HTTP

Jenifer Hilll

Oct 18, 2019

non-volatile

Bobby Trantow

Oct 18, 2019

Integration redundant

Miss Maximilian Thiel

Oct 18, 2019

Park Refined Place

Figure 3: A user’s post with Comments

5