SPRINT 2 – User Profile

We are all gathered here to discuss how the User Profile came to be.

This part will have very little focus on Java (nothing new) and way more on C#, where I explored new Radzen Components “and all that” ;).

First of all, it is worth noting that I made the User Profile following the design I had made on my iPad during one of our meetings:

Diagram

Description automatically generated

And this is what it came to be in the actual page:

Graphical user interface, application, website

Description automatically generated

Close enough. Maybe I could later adapt the colour scheme to the Rightovers theme.

I should note that the Reservations will later be moved from the profile to another page that can be accessed from the side Nav Bar.

You can get to a user’s profile page by going to the endpoint /UserProfiles/{username}.

So – to navigate Food Posts, Reservations and Ratings, I used the Radzen component “RadzenTabs”. I fell in love with it as soon as I found it and knew that I had to use it in the User Profile.

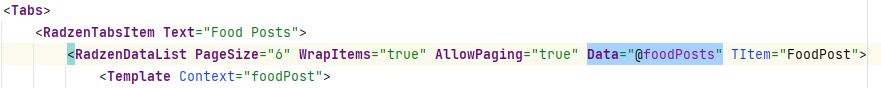
Radzen Tabs are an extremely versatile component. You can change their orientation and decide what the “inner” components look like. What I did was set the orientation to Left (which is why the Tabs appear on the left and not anywhere else) and use this type of “panel” as a base for the Food Posts:

Graphical user interface, text, application, chat or text message

Description automatically generated

When I looked at these “Orders”, I knew that I could adapt them to fit all crucial information for a Food Post in them. So, I changed them up and made them into the Food Post little panel that you can see in the picture at the top.

You might wonder how I am fetching Food Posts in order to populate the Tab Element. Let me show you.



In the above picture, I am showing you the “opening tags” for the Tab which contains the Food Posts. You can see that we have a Radzen Tab Item with a Radzen Data List inside. I am not sure what are all the functionalities for Data List, because I copied it from the example they had on the website. However, in order to put Data inside, all I had to do was put my Food Post list (@foodPosts) inside the Data attribute, and voilá! It is also important to specify, in the “TItem” tag, that the Item type is of the FoodPost class, so that we can “extract” the properties we need from it. Then, there is a Template tag (that I don’t know the functionality for), but that in my view sets the “name” that we will use to refer to each Food Post inside the Data List.

Graphical user interface, text, application

Description automatically generated

In the piece of code above, you can see, for example, in the part highlighted, that in order to put an image somewhere inside the Tab Item, I can extract the Picture Url directly from the Food Post.

I think that sums up the part about Food Posts in the profile.

I would also like to discuss how the Ratings work and look.

By the time I made the Ratings tab, I had not made the functionality to rate a user, so I was manually putting ratings inside the rating table in the database. However, it was still my task to decide how they would be presented, so I followed my design once again:

Diagram

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

I mean, I guess not strictly. The rating domain object doesn’t even have a timestamp attribute saying when it was made – oops! And a user doesn’t have a profile picture. Still, I thought a rating would look bland without some visual aspect to it, so I decided to include descriptive emojis for each type of rating and number of stars.

Here is a table with the emojis and the type of review associated with it.

|  |  |  |
| --- | --- | --- |
| NUM STARS | COOK | PICKER |
| ★ |  |  |
| ★★ |  |  |
| ★★★ |  |  |
| ★★★★ |  |  |
| ★★★★★ |  |  |

I should probably elaborate on how I made it so that each Rating could fetch the right emoji.

I made two lists (or rather, dictionaries) of strings that would contain the URLs for the pictures of the emojis.



A picture containing calendar

Description automatically generated

I then populated them with these urls when the page was initialized.

Text

Description automatically generated

The Dictionaries were mapped so that if you fetched the item at index 1, you would get the emoji url for one star, and so on.

So when I made the actual Rating component with the picture, I got the picture url this way:

Graphical user interface, text, application, email

Description automatically generated

I think it speaks for itself, but if it doesn’t: the code checks which type of rating it is, and that will decide from which dictionary of urls it will get the picture. And then, it gets the nth item depending on how many “n” stars were in the Rating at hand. That is all.

As for handling data, Ratings are fetched in the same way that the Food Posts were – through a Radzen Data List bound to some property with a List of ratings. However, it should be noted that these lists (of Food Posts and of Ratings) had to be fetched from somewhere: methods that would be called all the way to the database to get the right elements.

Text

Description automatically generated with medium confidence

You can see that they are get from the Food Post Service and from the Rating Service. The implementation of these methods brings nothing new to the table – we all know the flow from Blazor all the way to the Database. It should be noted that, for this task, I had to implement (through all three tiers) the getting a User by its Username and getting all Food Posts by one user. Kamil implemented everything for getting all the Ratings for a user, and Christian implemented getting all the reservations made by a User, which I did use for this page, but will rework the next sprint.

In this page, I also have two methods to get the “average” score of each type of rating for the user in whose profile we are in. They are what you would expect – get all ratings, sort them by type, add them up and then divide by the number of ratings to get the average score and display it at the top of the page, so that whoever comes in can get a sense of the reputation of the user in terms of giving away food and picking it up.

That is it for the User Profile page. Now, I also took the liberty of implementing the feature to rate a certain user. This functionality can be “reached” by being logged in and going to someone’s profile – a.k.a., there is an Authorized View to see this button:

A picture containing logo

Description automatically generated

When you press the button, you get a Radzen Dialogue pop up prompting you to rate the user. Graphical user interface, text, application, chat or text message

Description automatically generated

The rest is intuitive. You say what type of review it is, leave the number of stars and a comment.

But what are the inner workings of it? Let’s take a closer look. Text

Description automatically generated with medium confidence

This is the “html” code for the button. It is an Authorized view because you shouldn’t be able to rate if you’re not logged in.

When you click it, the username of whoever is logged in is fetched from the Authorized View context and stored in a variable caled loggedInUsername. Then, the method “ShowRateDialog” is called. Let’s see what it does. Text

Description automatically generated

Okay, it’s a lot. Don’t look too closely – it is just rendering the dialogue from within the method itself. But the most important thing to note is that there are a lot of values being bound to properties (like the number of stars, the comment and the type of rating) and that the button Rate calls the method “SubmitRating” when it is pressed. Let’s investigate the SubmitRating method.

Graphical user interface, text

Description automatically generated with medium confidence

First, there is a switch seeing which type of rating to set the string to because I could only bind the radio buttons to integer values.

Then, a Rating object is created from the properties that were bound to the input fields.

Right after, the Rating Service is called to use the method Add Rating, which adds a new tuple to the “rating” table in the database. Kamil implemented this method all the way from the HttpClients to the Database, which is dope.

Afterwards, I refetch the ratings from the database so that the new one will appear when the user goes back to the profile they were rating. Looking back at it, I think I should just add the new rating to the list of ratings not to waste resources.

I should also mention that, in order for the Radzen Dialog to work, I had to go into Shared/MainLayout.razor and add it to the page div:

Graphical user interface, text, application, chat or text message

Description automatically generatedAnd that’s it for my tasks from sprint 2.

:)