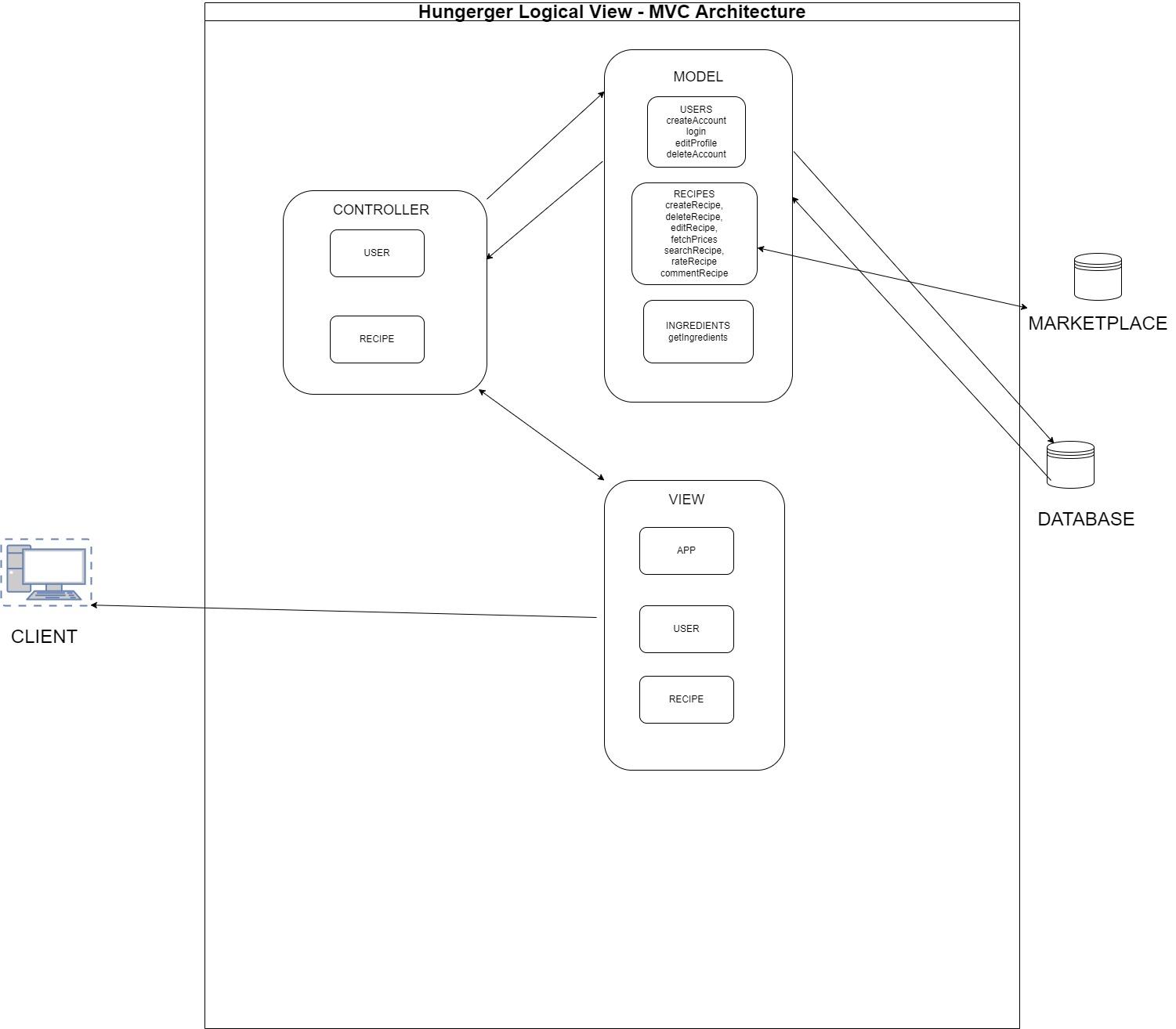
| **REVISIONS** | | | |
| --- | --- | --- | --- |
| **Rev. No** | **Description** | **Date** | **Person** |
| 0.1 | Creating the document according to the CreateAccount, CreateRecipe, and EditRecipe use cases. | 23.12.2023 | Aslı |
| 0.2 | Delete recipe, Class diagram,  MVC diagram, and navigation, database design were added  Use case realizations were corrected | 02.01.2024 | Aslı, Abdullah, Tarık, Merve |

**Design structure**

The design structure of the Hungerger Application is organized in an MVC architecture since it gives each individual within the development team a standard to follow while increasing the modularity throughout the project's development. This pattern separates concerns, dividing the application into three interconnected components: model, view, and controller**.**

****

**Figure:** MVC architecture of Hungerger

We use Flask to build the application backend, React for front-end development, and Flask templates for creating views. MySQL will serve as the application’s database. For caching, we use Redis. We use two Ubuntu servers; one is for the main server, and the other one is for backup. NGINX will be used to perform the load balancing. For security, we use the Bcrypt library to hash the passwords. AWS S3 bucket will be used as a cloud service to store and retrieve images uploaded by the users for profiles and recipes.

**Subsystems**

**Subsystem 1 - Admin Panel**

The admin panel subsystem is placed within the same server as the Hungerger application. This subsystem acts as a visualization of the database in order to give the admin users flexibility and control over the data.

**Patterns**

This system has no design pattern

**Requirement realizations**

This section introduces a class diagram to show the relationship of classes for realized use cases.

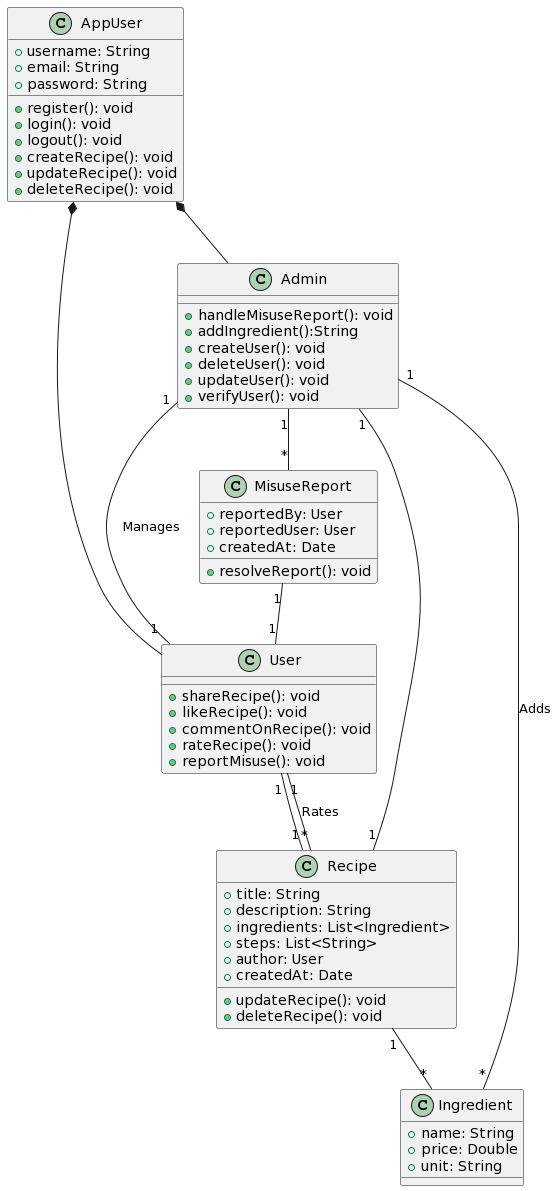


Figure: Class Diagram

**Realization1: Create Account**

This use case describes the account creation process on the Hungerger application. It can be initiated by visitors, people who are only allowed to see the main page. After the account creation, the visitor becomes a user of the system.

#### View of participants:

**Visitor**

* **Behavior:** Initiates account creation process, provides the necessary information and confirms their request.
* **Attributes:** username, email, password, and password confirmation.
* **Relationships:** Interacts with the user interface and triggers account creation.

**User**

* **Behavior:** Initiates login process, provides the necessary information, and confirms their request.
* **Attributes:** username and password
* **Relationships:** Interacts with the user interface and triggers the login process

**Basic scenario**

1. The Visitor has access to the webpage of the application.
2. The Visitor clicks on the “Create One” button.
3. The Visitor enters “Username”, which matches the standard.
4. The Visitor enters “Email”, which matches the standard.
5. The Visitor enters “Password”, which matches the standard.
6. The Visitor enters “Confirm Password”, which matches with the “Password”.
7. The Visitor clicks on the "Register" button, and the system directs the visitor to the login page.

**Realization2: Create Recipe**

This use case describes the recipe creation process on the Hungerger application. It can only be done by the users of the system. The prices of the ingredients are retrieved from a third-party marketplace.

**View of participants**

**User**

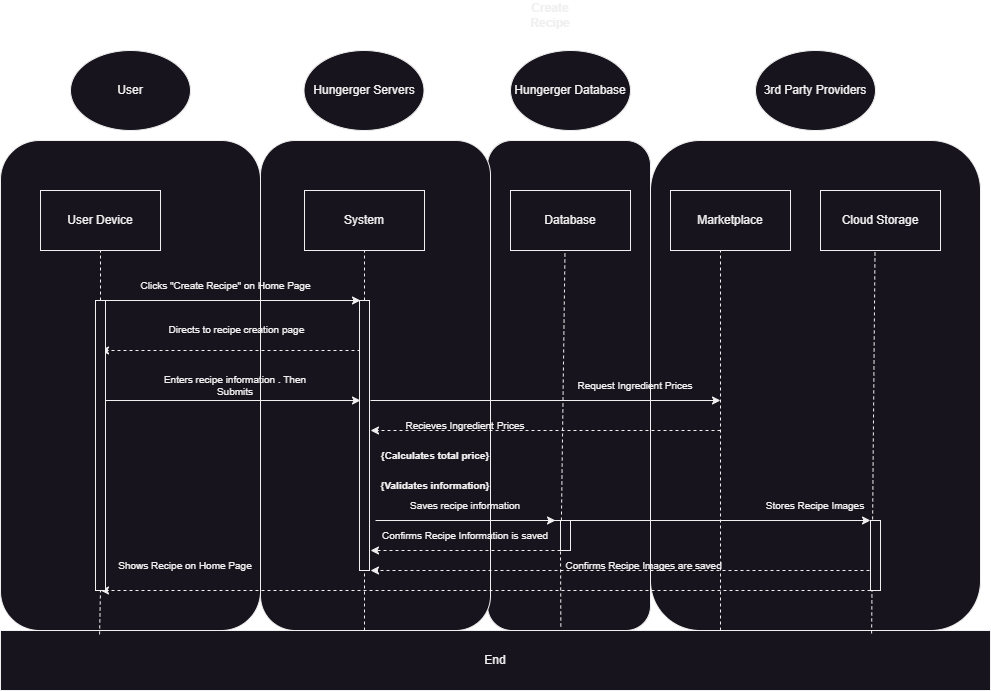
* **Behavior:** Initiates create recipe process, provides the necessary information, and confirms their request.
* **Attributes:** the recipe name, description, image, dietary type, ingredients, and the amounts of the ingredients.
* **Relationships:** Interacts with the user interface and triggers the creation recipe process and data retrieval from a marketplace process.

**Marketplace**

* **Behavior:** Provides the ingredient prices.
* **Attributes:** ingredients, prices.
* **Relationships:** Interact with the API to provide the necessary information.

**Basic scenario**

1. The User clicks on the “Create Recipe” button.
2. The User enters the recipe information (name, description, ingredients, image).
3. The User clicks on the “Post” button.
4. The system reflects the changes to the UI.



**Figure:** CreateRecipe Sequence Diagram

**Realization3: Edit Recipe**

This use case describes the recipe editing process on the Hungerger application. It can only be done by the users of the system. A recipe can be edited by only its creator. The prices of the ingredients are retrieved from a third-party marketplace.

**View of participants**

**User**

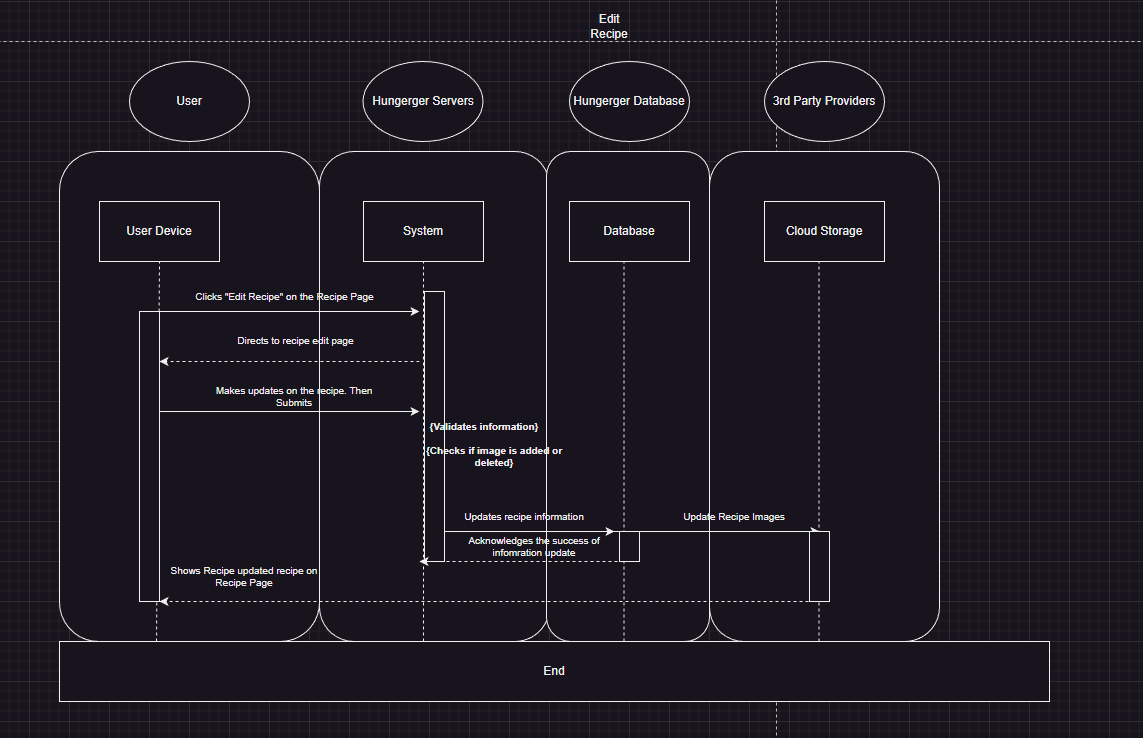
* **Behavior:** Initiates edit recipe process, provides the necessary information, and confirms their request.
* **Attributes:** the recipe name, description, image, dietary type, ingredients, and the amounts of the ingredients.
* **Relationships:** Interacts with the user interface and triggers the edit recipe process and data retrieval from a marketplace process.

**Marketplace**

* **Behavior:** Provides the ingredient prices.
* **Attributes:** ingredients, prices.
* **Relationships:** Interact with the API to provide the necessary information

**Basic scenario**

1. The User has at least one recipe to edit.
2. The User locates one of their recipes and clicks on it.
3. The system makes the recipe editable.
4. The User enters a new recipe information (name, ingredients, image).
5. The User clicks on the “Post” button.
6. The system reflects the changes to the UI.



**Figure:** EditRecipe Sequence Diagram

**Realization4: Delete Recipe**

This use case describes the recipe deletion process on the Hungerger application. It can only be done by the users of the system. A recipe can be deleted by only its creator, and the admin of the system. The prices of the ingredients are retrieved from a third-party marketplace.

**View of participants**

**User**

* **Behavior:** Initiates the delete recipe process and confirms their request.
* **Attributes:** None
* **Relationships:** Interacts with the user interface and triggers the delete recipe process.

**Admin**

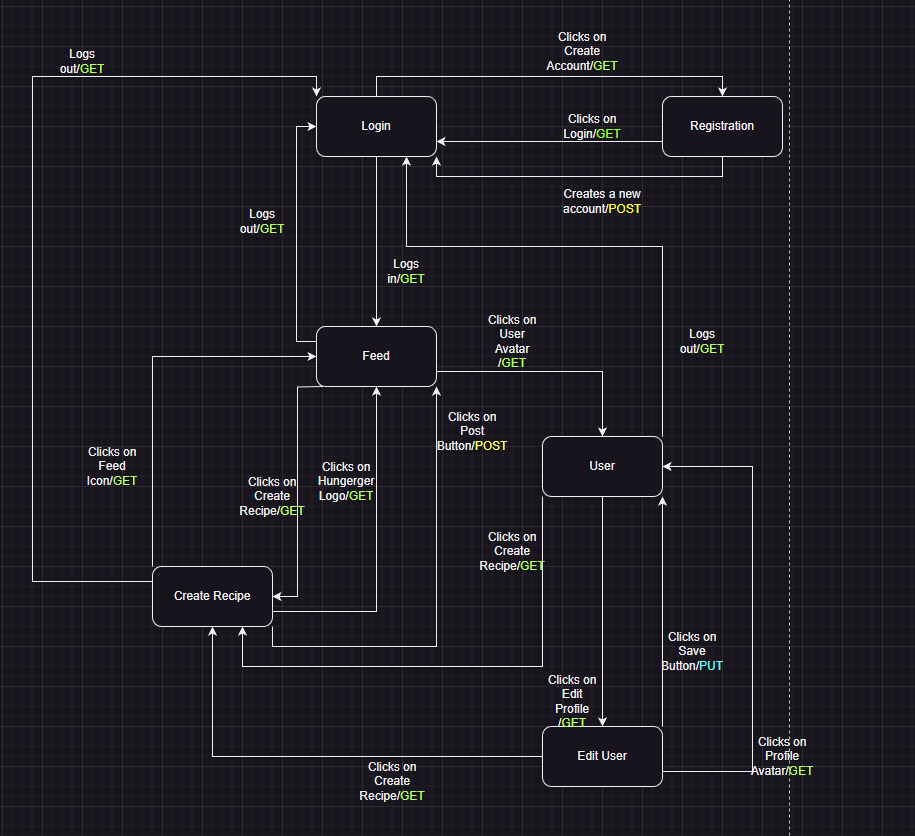
* **Behavior:** Initiates the delete recipe process and confirms their request.
* **Attributes:** None
* **Relationships:** Interacts with the user interface and triggers the delete recipe process.

**Basic scenario**

1. The User has at least one recipe to delete.
2. The User locates one of their recipes.
3. The User clicks on the “Delete Recipe” button.
4. The system removes the recipe and reflects the changes to the UI.

**Page Structure and Navigation:**

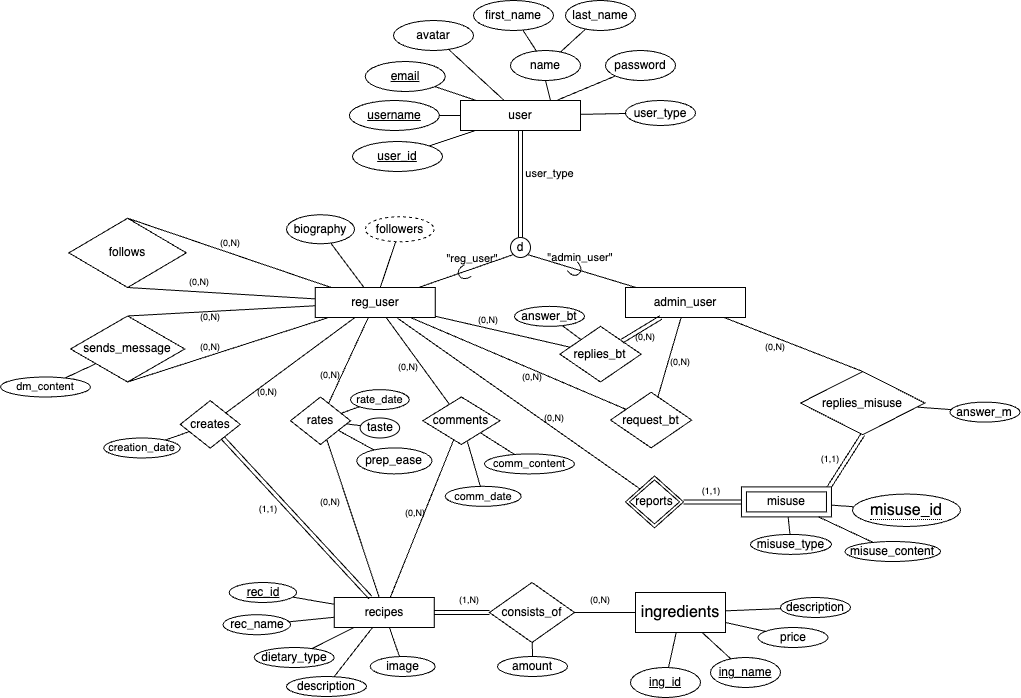
Figure 4 shows the different pages implemented in the Hungerger app and their navigation routes and triggers. You can also check the methods used to navigate the pages in the same figure.



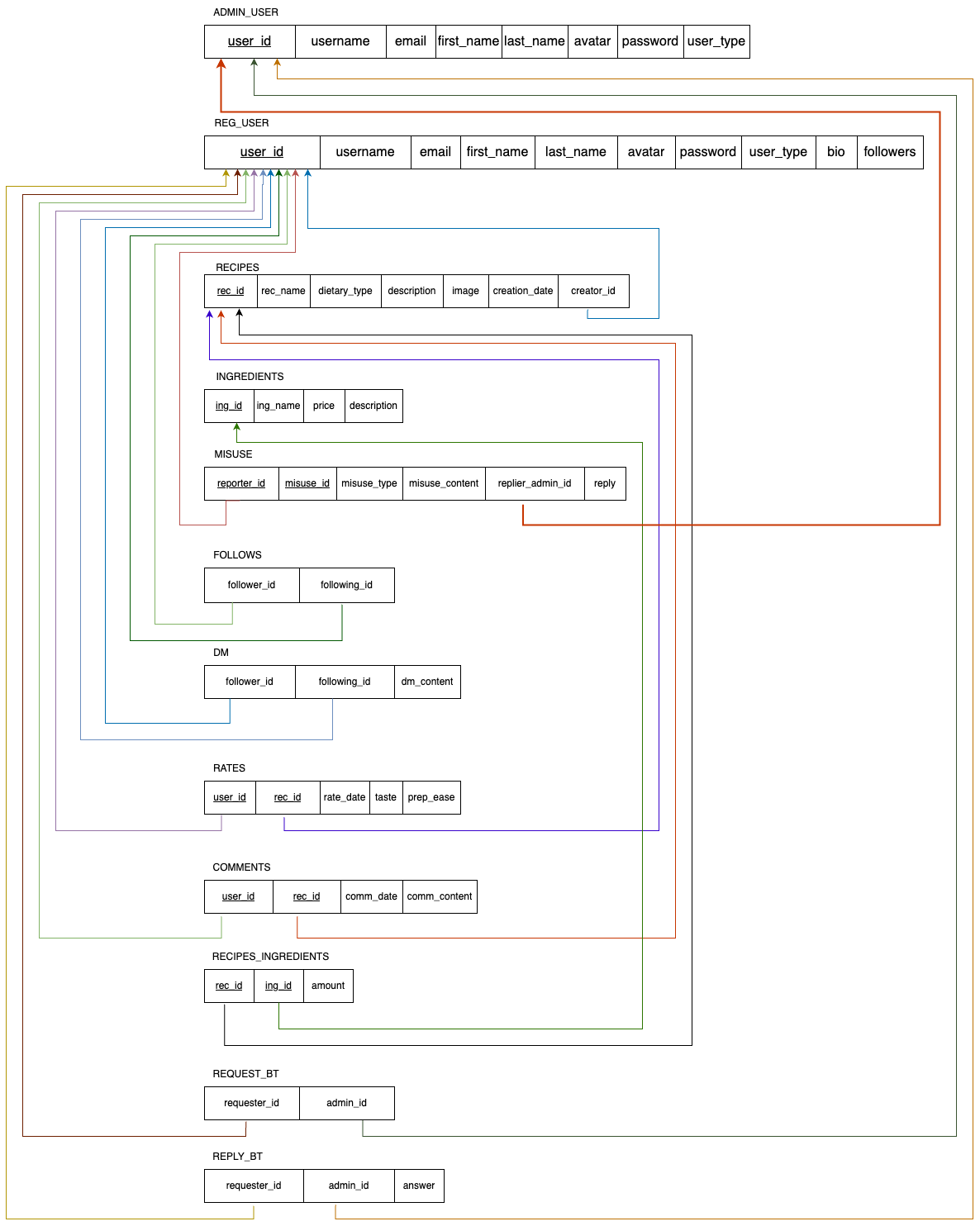
**Figure:** Navigation Diagram

**Database Design**

Please refer to the following images to see our database design.



**Figure:** ER Diagram



**Figure:** Relational Model