API Reference

RecipEase.Server

API Version: 1.0

INDEX

API

1. CUSTOMER

1.1 GET /api/Customer

Returns the Customer with the given username.

Retrieves the object with the given username value, in the username column, from the `Customer` table, if it exists.

A 'select*' query with a 'where' clause to find the username
and its associated attributes.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
`userId`	`string`	The Username of the Customer to retrieve.

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
{
  userId        string
  customerName  string
  age           integer
  weight        number
  favMeal       enum     ALLOWED:0, 1, 2
}
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

1.2 PUT /api/Customer/{id}

Update the information of an existing customer

Updates information/attributes of an existing user of type customer,
if the user exists in the Customer table of the database.
The authenticated user must be the user to be updated.

An Update operation is used to update the Customer in the database if
the user exists.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	string	The user id of the Customer to update.

REQUEST BODY - application/json

```
{
  userId       string
  customerName string
  age          integer
  weight       number
  favMeal      enum     ALLOWED:0, 1, 2
}
```

RESPONSE

STATUS CODE - 204: Success

STATUS CODE - 400: Bad Request

RESPONSE MODEL - application/json

```
{
  type     string
  title    string
  status   integer
  detail   string
  instance string
}
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type     string
  title    string
  status   integer
  detail   string
  instance string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type     string
  title    string
```

```
  status   integer
  detail   string
  instance string
}
```

---

2. INGR

2.1 GET /api/Ingr

Returns all the ingredients in Ingredient
functionalities : Retrieves all ingredients.

database: Ingredient

constraints: no constraints

query: select * in Ingredient

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
[{
Array of object:
  name*            string
  rarity           enum    ALLOWED:0, 1, 2
  weightToVolRatio number
}]
```

2.2 GET /api/Ingr/{id}

Returns the ingredients in Ingredient with the given id
functionalities : Retrieves 1 row of ingredient

database: Ingredient

constraints: no constraints

query: select * in Ingredient where name=id

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	string	

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
{
```

```
  name*            string
  rarity           enum    ALLOWED:0, 1, 2
  weightToVolRatio number
}
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type     string
  title    string
  status   integer
  detail   string
  instance string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type     string
  title    string
  status   integer
  detail   string
  instance string
}
```

3. INGRINSHOPPINGLIST

3.1 PUT /api/IngrInShoppingList

Edit the specific item in the IngrInShoppingList

functionalities : Edit an item ingredient with a specific userid ,ingredient and unit.

database: IngrInShoppingList, Customer

constraints: The authenticated user making this request must be the owner of the shopping list.

query: update * IngrInShoppingList set (some instance) where UserId = id and IngrName = iname and UnitName = uname

REQUEST

REQUEST BODY - application/json

```
{
  userId*    string
  unitName*  string
  ingrName*  string
  quantity   number
}
```

RESPONSE

STATUS CODE - 204: Success

STATUS CODE - 401: Unauthorized

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
```

```
  status    integer
  detail    string
  instance  string
}
```

3.2 POST /api/IngrInShoppingList

Create new row in IngrInShoppingList

functionalities : Insert a row with a specific userid ,ingredient and unit.

database: IngrInShoppingList, Customer

constraints: The authenticated user making this request must be the owner of the shopping list.

query: insert into IngrInShoppingList

REQUEST

REQUEST BODY - application/json

```
{
  userId*    string
  unitName*  string
  ingrName*  string
  quantity   number
}
```

RESPONSE

STATUS CODE - 201: Success

RESPONSE MODEL - application/json

```
{
  userId*    string
  unitName*  string
  ingrName*  string
  quantity   number
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
```

```
  detail    string
  instance  string
}
```

3.3 DELETE /api/IngrInShoppingList

Delete an ingredient in an user's shopping list

functionalities : Retrieves ingredient in a user's shopping list with according unit.

database: IngrInShoppingList, Customer

constraints: The authenticated user making this request must be the owner of the shopping list.

query: delete from IngrInShoppingList where userId = userId, ingrName = ingrName, unitName = unitName

REQUEST

REQUEST BODY - application/json

```
{
  userId*    string
  unitName*  string
  ingrName*  string
  quantity   number
}
```

REQUEST BODY - text/json

```
{
  userId*    string
  unitName*  string
  ingrName*  string
  quantity   number
}
```

REQUEST BODY - application/*+json

```
{
  userId*    string
  unitName*  string
  ingrName*  string
  quantity   number
}
```

RESPONSE

STATUS CODE - 204: Success

STATUS CODE - 401: Unauthorized

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

---

4. OIDCCONFIGURATION

4.1 GET /_configuration/{clientId}

Authentication endpoint.
This endpoint is necessary for .NET Identity to work.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*clientId	string	

RESPONSE

STATUS CODE - 200: Success

5. RECIPE

5.1 GET /api/Recipe/{id}

Get a recipe.

Returns the recipe with the given `id`, or gives a 404 status code if it doesn't exist in the database. The recipe will include it's average rating if it has been rated.

This endpoint interacts with all attributes from the `recipe` table, and with the `RecipeId` and `Rating` attributes from the `reciperating` table.

The endpoint performs a `select *` query with a `where` clause to find the specified recipe. If the recipe was found, the `reciperating` table is queried for rows with `RecipeId` matching the found id, and the `Rating` attribute is collected.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	int32	The id of the recipe to return.

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
{
  id*           integer
  name          string
  steps         string
  cholesterol   number
  fat           number
  sodium        number
  protein       number
  carbs         number
  calories      number
  authorId*     string
  averageRating number
}
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type     string
  title    string
  status   integer
  detail   string
  instance string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

5.2 PUT /api/Recipe/{id}

Edit a recipe.

Updates the given recipe in the database. The `id` in the url must match the `id` in the recipe.

The customer specified by `authorId` must be the authenticated user making this request.

This endpoint interacts with the `recipe` and `customer` tables. The `UserId` attribute on the `customer` table will be checked against the `authorId`.

The endpoint will perform an `update` command on the `recipe` table to update the recipe, and foreign key constraints will be relied upon to validate the `authorId`.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	int32	The id of the recipe to update.

REQUEST BODY - application/json

```
{
  id              integer
  name*           string
  steps           string
  cholesterol     number
  fat             number
  sodium          number
  protein         number
  carbs           number
  calories        number
  authorId        string
  author          {recursive}     Customer
  categories      [{recursive}]   RecipeInCategory
  usesIngredients [{recursive}]   Uses
  ratings         [{recursive}]   RecipeRating
}
```

RESPONSE

STATUS CODE - 204: Success

STATUS CODE - 400: Bad Request

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

5.3 DELETE /api/Recipe/{id}

Delete a recipe.

Deletes the given recipe in the database.

If the recipe does not exist in the database, a 404 status code is returned. The customer specified by `AuthorId` in the found recipe must be the authenticated user making this request.

This endpoint interacts with the `recipe` and `customer` tables. The `UserId` attribute on the `customer` table will be checked against the `AuthorId` from the `recipe` table.

The endpoint will perform a `select` query on the `customer` table to validate the `authorId`, and a `delete` command on the `recipe` table to delete the recipe.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
*id	int32	The id of the recipe to update.

RESPONSE

STATUS CODE - 200: Success

STATUS CODE - 400: Bad Request

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

5.4 GET /api/Recipe

Get a list of recipes.

Returns a list of recipes from the database, optionally filtered by title, category, and/or user id.

This endpoint interacts with all attributes in the `recipe` and `recipeincategory` tables, and the `UserId` attribute in the `Customer` table.

The endpoint performs a `select *` query, with a `where` clause included when necessary to apply the specified filters. The `recipe` table is optionally joined with `recipeincategory` to filter by category. The `recipe` table is optionally joined with the `customer` table to filter by customer.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
`titleMatch`	`string`	String to filter recipes by title. If provided, only recipes with the filter string in their title (case insensitive) will be returned.
`categoryName`	`string`	String to filter recipes by category. If provided, only recipes in the given category will be returned. If there is no category with the given name in the database, no recipes will be returned.
`userId`	`string`	Get only recipes authored by the customer with this id.

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
[{
Array of object:
  id*            integer
  name           string
  steps          string
  cholesterol    number
  fat            number
  sodium         number
  protein        number
  carbs          number
  calories       number
  authorId*      string
  averageRating  number
}]
```

5.5 POST /api/Recipe

Create a recipe.

Adds the given recipe to the database, and returns it on success. If the `id` is specified for the recipe, the endpoint will attempt to add the recipe to the database with that `id`. If a recipe with the given `id` already exists, an error code is returned. If the `id` is not specified, the `id` is automatically generated for the given recipe.

The customer specified by `authorId` must be the authenticated user making this request.

This endpoint interacts with the `recipe` and `customer` tables. The `UserId` attribute on the `customer` table will be checked against the `authorId`.

The endpoint will perform an `insert` command on the `recipe` table to add the recipe, and foreign key constraints will be relied upon to validate the `authorId`.

REQUEST

REQUEST BODY - application/json

```
{
  id              integer
  name*           string
  steps           string
  cholesterol     number
  fat             number
  sodium          number
  protein         number
  carbs           number
  calories        number
  authorId        string
  author          {recursive}     Customer
  categories      [{recursive}]   RecipeInCategory
  usesIngredients [{recursive}]   Uses
  ratings         [{recursive}]   RecipeRating
}
```

RESPONSE

STATUS CODE - 201: Success

RESPONSE MODEL - application/json

```
{
  id              integer
  name*           string
  steps           string
  cholesterol     number
  fat             number
  sodium          number
  protein         number
  carbs           number
  calories        number
  authorId        string
  author          {recursive}     Customer
  categories      [{recursive}]   RecipeInCategory
  usesIngredients [{recursive}]   Uses
  ratings         [{recursive}]   RecipeRating
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
```

```
  title     string
  status    integer
  detail    string
  instance  string
}
```

---

6. RECIPECATEGORY

6.1 GET /api/RecipeCategory

Get all recipe categories

Returns a list of all recipe categories in the database.

This endpoint interacts with all attributes in the `recipecategory` table.

The endpoint performs a `select *` query.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
[{
Array of object:
  name                 string
  totalInCatg          integer
  averageCaloriesCatg  integer
}]
```

7. RECIPECOLLECTION

7.1 GET /api/RecipeCollection

Get recipe collections.

Returns the recipe collections of the user with given `userId`.

This endpoint interacts with the `recipecollection` table.

The endpoint performs a `select *` query with a `where` clause to find the specified recipe collections.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
`userId`	`string`	The id of the customer whose recipe collections should be returned.

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
[{
Array of object:
  userId*      string
  title*       string
  description  string
  visibility*  enum    ALLOWED:0, 1
}]
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

7.2 POST /api/RecipeCollection

Create a recipe collection.

Adds the given recipe collection to the database, and returns it on success. The `title` must be unique across all recipe collections for the given user; if it isn't an error code will be returned.

The customer specified by `userId` must be the authenticated user making this request.

This endpoint interacts with the `recipecollection` and `customer` tables. The `UserId` attribute on the `customer` table will be checked against the `userId`.

The endpoint will perform an `insert` command on the `recipecollection` table to add the recipe collection, and foreign key constraints will be relied upon to validate the `userId`.

REQUEST

REQUEST BODY - application/json

```
{
  userId*      string
  title*       string
  description  string
  visibility*  enum    ALLOWED:0, 1
}
```

RESPONSE

STATUS CODE - 201: Success

RESPONSE MODEL - application/json

```
{
  userId*      string
  title*       string
  description  string
  visibility*  enum    ALLOWED:0, 1
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
```

```
  title     string
  status    integer
  detail    string
  instance  string
}
```

7.3 DELETE /api/RecipeCollection

Delete a recipe collection.

Deletes the given recipe collection in the database.

If the recipe collection does not exist in the database, a 404 status code is returned. The customer specified by `UserId` in the found recipe collection must be the authenticated user making this request.

This endpoint interacts with the `recipecollection` and `customer` tables. The `UserId` attribute on the `customer` table will be checked against the `UserId` from the `recipecollection` table.

The endpoint will perform a `select` query on the `customer` table to validate the `UserId`, and a `delete` command on the `recipecollection` table to delete the recipe collection.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
`title`	`string`	The title of the recipe collection to delete.

RESPONSE

STATUS CODE - 200: Success

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

8. RECIPEINCOLLECTION

8.1 GET /api/RecipeInCollection

Get a list of all recipes in a given recipe collection.

Returns a list of recipes from the database which are in the given recipe collection.

This endpoint interacts with all attributes in the `recipeincollection` table.

The endpoint performs a `select *` query, with a `where` clause included to filter for the given recipe collection.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
`userId`	`string`	The userId of the customer who owns the collections to be retrieved.
`collectionTitle`	`string`	The title of the collection whose recipes should be retrieved.

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
[{
Array of object:
   recipeId*          integer
   collectionUserId*  string
   collectionTitle*   string
}]
```

8.2 POST /api/RecipeInCollection

Add a recipe to a collection.

Adds the given recipe to the given recipe collection in the database, and returns it on success. If the given recipe or recipe collection do not exist, an error code is returned.

The authenticated user making this request must be the owner of the collection.

This endpoint interacts with the `recipe`, `customer`, `recipeincollection`, and `recipecollection` tables. The keys in the payload will be checked against the rows in `recipe` and `recipeincollection`.

The endpoint will perform an `insert` command on the `recipeincollection` table to add the recipe to the collection, and foreign key constraints will be relied upon to validate recipe

collection and recipe keys.

REQUEST

REQUEST BODY - application/json

```
{
  recipeId*          integer
  collectionUserId*  string
  collectionTitle*   string
}
```

REQUEST BODY - text/json

```
{
  recipeId*          integer
  collectionUserId*  string
  collectionTitle*   string
}
```

REQUEST BODY - application/*+json

```
{
  recipeId*          integer
  collectionUserId*  string
  collectionTitle*   string
}
```

RESPONSE

STATUS CODE - 201: Success

RESPONSE MODEL - application/json

```
{
  recipeId*          integer
  collectionUserId*  string
  collectionTitle*   string
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

8.3 DELETE /api/RecipeInCollection

Remove a recipe from a collection.

Adds the given recipe to the given recipe collection in the database, and returns it on success. If the given recipe or recipe collection do not exist, an error code is returned.

The authenticated user making this request must be the owner of the collection.

This endpoint interacts with the `recipe`, `customer`, `recipeincollection`, and `recipecollection` tables. The keys in the payload will be checked against the rows in `recipe` and `recipeincollection`.

The endpoint will perform a `delete` command on the `recipeincollection` table to remove the recipe from the collection, and foreign key constraints will be relied upon to validate recipe collection and recipe keys.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
`recipeId`	`int32`	The recipe to remove from the collection.
`collectionTitle`	`string`	The collection to delete from.

RESPONSE

STATUS CODE - 200: Success

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

9. RECIPERATING

9.1 GET /api/RecipeRating

Returns the (user) rating of the recipe under consideration

In the case of a specific customers rating, it
retrieves the object with the given recipeID value and customer userID,
from the recipeID and userID columns, from the
`RecipeRating` table, if the entry exists. In this case the following query applies:
A 'select rating' query with a 'where userId=x and recipeId=y' clause to find a users rating from the RecipeRating table

In the case of a recipe's rating, it
retrieves all objects with the given recipeID value
from the recipeID column, from the
`RecipeRating` table, if the entry exists. In this case the following query applies:

A 'select rating' query with a 'where' clause to find the receipeID's ratings from the RecipeRating table

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
`userId`	`string`	The Username of the Customer to retrieve.
`recipeId`	`int32`	The recipeID of the recipe to retrieve.

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
[{
Array of object:
  userId*    string
  recipeId*  integer
  rating     integer  between 1 and  5
}]
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
```

```
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

9.2 PUT /api/RecipeRating

Update the information of an existing rating entry

Updates the rating information of an existing rating entry
in the RecipeRating table of the database.
The authenticated user must be the supplier whose ingredient is to be updated.

An Update operation is used to update the rating entry in the database, if the entry exists. The query to find the entry relies on a 'select *' and 'where' using the recipeId and userId foreign keys.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
userId	string	The Id of the user who is updating the rating.
RecipeId	int32	The Id for the Recipe whose rating will be updated.

REQUEST BODY - application/json

```
{
  userId*    string
  recipeId*  integer
  rating     integer  between 1 and  5
}
```

RESPONSE

STATUS CODE - 204: Success

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
```

```
}
```

9.3 POST /api/RecipeRating

Create a recipe rating.

Adds the given recipe rating to the database, and returns it on success.

The customer specified by `userId` must be the authenticated user making this request. The recipe specified by `recipeId` must exist in the database.

This endpoint interacts with the `reciperating`, `recipe`, and `customer` tables. The `UserId` attribute on the `customer` table will be checked against the `userId`. The `Id` attribute on the `recipe` table will be checked against the `recipeId`.

The endpoint will perform an `insert` command on the `reciperating` table to add the recipe rating, and the foreign key constraints will be relied upon to validate the `recipeId` and `userId`.

REQUEST

REQUEST BODY - application/json

```
{
  userId*    string
  recipeId*  integer
  rating     integer  between 1 and  5
}
```

RESPONSE

STATUS CODE - 201: Success

RESPONSE MODEL - application/json

```
{
  userId*    string
  recipeId*  integer
  rating     integer  between 1 and  5
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
```

```
  title     string
  status    integer
  detail    string
  instance  string
}
```

9.4 DELETE /api/RecipeRating

Delete an existing rating entry

Delete a rating entry from the RecipeRating relation,
if the entry exists in the RecipeRating relation of the database.

The authenticated user must be the supplier of the item to be deleted.

A Delete operation to delete a Rating entry is performed. The query involves a 'select *'
and a 'where' on the userId and recipeId

REQUEST

REQUEST BODY - application/json

```
{
  userId*    string
  recipeId*  integer
  rating     integer  between 1 and  5
}
```

REQUEST BODY - text/json

```
{
  userId*    string
  recipeId*  integer
  rating     integer  between 1 and  5
}
```

REQUEST BODY - application/*+json

```
{
  userId*    string
  recipeId*  integer
  rating     integer  between 1 and  5
}
```

RESPONSE

STATUS CODE - 204: Success

STATUS CODE - 401: Unauthorized

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

---

10. SHOPPINGLIST

10.1 GET /api/ShoppingList

Returns a user's shopping list

functionalities : Retrieves the shopping list of the currently logged in user. Creates the shopping list of the user if it does not exist.

database: ShoppingList, User, Ingredient, IngrInShoppingList, Unit

constraints: The authenticated user making this request must be the owner of the shopping list.

query: select * ShoppingList with UserId = userId, select * ingredients where ingredient is in shopping list based on ingrinshoppinglist table

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
{
  _shoppingList {
    userId*         string
    name*           string
    lastUpdate      string
    numIngredients  integer
  }
  shoppingList {
    userId*         string
    name*           string
    lastUpdate      string
    numIngredients  integer
  }
  ingredients [{
  Array of object:
    ingredient {
      name*             string
      rarity            enum    ALLOWED:0, 1, 2
      weightToVolRatio  number
    }
    unit {
      name*     string
      unitType  enum    ALLOWED:0, 1
      symbol    string
    }
    quantity          number
  }]
}
```

STATUS CODE - 401: Unauthorized

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

11. SUPPLIER

11.1 GET /api/Supplier

Returns the Supplier with the given username.

Retrieves the object with the given username value, in the username column, from the `Supplier` table, if it exists.

A 'select*' query with a 'where' clause to find the username
and its associated attributes.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
`userId`	`string`	The username of the Supplier to retrieve.

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
{
  userId          string
  email           string
  phoneNo         string
  website         string
  supplierName    string
  storeVisitCount integer  >=0
                           DEFAULT:0
}
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type     string
  title    string
  status   integer
  detail   string
  instance string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type     string
  title    string
  status   integer
  detail   string
  instance string
}
```

11.2 PUT /api/Supplier/{id}

Update the information of an existing supplier user

Updates information of existing supplier,
if the supplier exists in the Supplier table of the database.
The authenticated user must be the user to be updated.

An Update operation is used to update the Supplier in the database if the user exists.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	string	The user id of the Supplier to update.

REQUEST BODY - application/json

```
{
  userId          string
  email           string
  phoneNo         string
  website         string
  supplierName    string
  storeVisitCount integer  >=0
                           DEFAULT:0
}
```

RESPONSE

STATUS CODE - 204: Success

STATUS CODE - 400: Bad Request

RESPONSE MODEL - application/json

```
{
  type     string
  title    string
  status   integer
  detail   string
  instance string
}
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type     string
  title    string
  status   integer
  detail   string
  instance string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
```

```
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

12. SUPPLIES

12.1 GET /api/Supplies/{ingredientName}

Returns suppliers that supply an ingredient.

The suppliers are retrieved from the `Supplies` and `Supplier` tables using `select` queries.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*ingredientName	string	The name of the ingredient whose suppliers should be retrieved.

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
[{
Array of object:
  userId          string
  email           string
  phoneNo         string
  website         string
  supplierName    string
  storeVisitCount integer  >=0
                           DEFAULT:0
}]
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type     string
  title    string
  status   integer
  detail   string
  instance string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type     string
  title    string
  status   integer
  detail   string
  instance string
}
```

12.2 GET /api/Supplies

Returns ingredients supplied by a supplier.

The ingredients, units, and quantities are retrieved from the `Supplies`, `Ingredient`, and `Unit` tables using `select` queries. The supplier with the given username is retrieved from the `Users` table as necessary using a `select query.`

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
`userId`	`string`	The user id of the supplier whose supplied ingredients should be retrieved.

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
[{
Array of object:
  ingredient {
    name*             string
    rarity            enum    ALLOWED:0, 1, 2
    weightToVolRatio  number
  }
  unit {
    name*     string
    unitType  enum    ALLOWED:0, 1
    symbol    string
  }
  quantity          number
}]
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

12.3 PUT /api/Supplies

Update the information of an existing supplies entry

Updates the quantity information of an existing supplies entry
in the supplies table of the database.
The authenticated user must be the supplier whose ingredient is to be updated.

An Update operation is used to update the supplies entry in the database, if
the entry exists.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
userId	string	The supplier who supplies the ingredient to be updated.

REQUEST BODY - application/json

```
{
  ingredient {
    name*             string
    rarity            enum    ALLOWED:0, 1, 2
    weightToVolRatio  number
  }
  unit {
    name*     string
    unitType  enum    ALLOWED:0, 1
    symbol    string
  }
  quantity          number
}
```

RESPONSE

STATUS CODE - 204: Success

STATUS CODE - 400: Bad Request

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

12.4 POST /api/Supplies

Add a new supplies entry

Add a new supplies entry to the Supplies relation,
if the entry does not exist in the Supplies relation of the database.

The authenticated user must be the supplier of the new item.

An Insert operation to insert a new supplies entry is performed.

REQUEST

REQUEST BODY - application/json

```
{
  ingrName*  string
  unitName*  string
  userId*    string
  quantity   number
}
```

RESPONSE

STATUS CODE - 201: Success

RESPONSE MODEL - application/json

```
{
  ingrName*  string
  unitName*  string
  userId*    string
  quantity   number
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

12.5 DELETE /api/Supplies

Delete an existing supplies entry

Delete a supplies entry from the Supplies relation,
if the entry exists in the Supplies relation of the database.

The authenticated user must be the supplier of the item to be deleted.

A Delete operation to delete a supplies entry is performed.

REQUEST

REQUEST BODY - application/json

```
{
  ingrName*  string
  unitName*  string
  userId*    string
  quantity   number
}
```

REQUEST BODY - text/json

```
{
  ingrName*  string
  unitName*  string
  userId*    string
  quantity   number
}
```

REQUEST BODY - application/*+json

```
{
  ingrName*  string
  unitName*  string
  userId*    string
  quantity   number
}
```

RESPONSE

STATUS CODE - 204: Success

STATUS CODE - 401: Unauthorized

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
```

```
}
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

---

13. UNIT

13.1 GET /api/Unit

Retrieves every unit

functionalities : retrieve all unit in Unit table

database: Unit

constraints: no constraints

query: Select * from Unit

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
[{
Array of object:
  name*     string
  unitType  enum    ALLOWED:0, 1
  symbol    string
}]
```

13.2 GET /api/Unit/{unitName}

Get a unit with the specified name

functionalities : retrieve the unit with the specified id

database: Unit

constraints: no constraints

query: select * from Unit where Name = id

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*unitName	string	name of the unit to be retrieved.

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
{
```

```
  name*     string
  unitType  enum    ALLOWED:0, 1
  symbol    string
}
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

14. UNITCONVERSION

14.1 GET /api/UnitConversion

Get unit conversion from one unit to another

functionalities : retrieve a unitconversion matching the given keys.

database: UnitConversion

query: select * from UnitConversion where ConvertsToUnitName = id1 and ConvertsFromUnitName = id2

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
convertsToUnitName	string	A unit name to find conversions to.
convertsFromUnitName	string	A unit name to find conversions from.

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
{
  convertsToUnitName*   string
  convertsFromUnitName* string
  ratio*                number
}
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type     string
  title    string
  status   integer
  detail   string
  instance string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type     string
  title    string
  status   integer
  detail   string
  instance string
}
```

15. USES

15.1 GET /api/Uses/{id}

Returns list of all ingredients used by a given recipe

Retrieves all ingredients that a given recipe uses,
and their associated units attribute, from the `Uses` table.

A 'select*' query with a 'where' clause to find the list of ingredients
used by a recipe,
and their associated attributes, is performed.
<param name="id">The id of the recipe who's ingredients should be retrieved.</param>

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	int32	

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
[{
Array of object:
   recipeId* integer
   unitName* string
   ingrName* string
   quantity  number
}]
```

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
   type     string
   title    string
   status   integer
   detail   string
   instance string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
   type     string
   title    string
   status   integer
   detail   string
   instance string
}
```

15.2 PUT /api/Uses

Update ingredients of a recipe

Retrieves the object with the given recipe id, from the
`Uses` table, if it exists.
The recipe owner must be the user to update.

An update query is performed using the recipe id, to update the ingredients
and their associated units.

REQUEST

REQUEST BODY - application/json

```
{
   recipeId* integer
   unitName* string
   ingrName* string
   quantity  number
}
```

RESPONSE

STATUS CODE - 204: Success

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
   type     string
   title    string
   status   integer
   detail   string
   instance string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
   type     string
   title    string
   status   integer
   detail   string
   instance string
}
```

15.3 POST /api/Uses

Add a new Uses entry

Add a new Uses entry to the Uses relation,
if the entry does not exist in the Uses relation of the database.

The recipe owner must be the user to add a new entry.
An Insert operation to insert a new Uses entry is performed.

REQUEST

REQUEST BODY - application/json

```
{
  recipeId            integer
  unitName            string
  ingrName            string
  quantity            number
  recipe {
    id              integer
    name*           string
    steps           string
    cholesterol     number
    fat             number
    sodium          number
    protein         number
    carbs           number
    calories        number
    authorId        string
    author          {recursive}    Customer
    categories      [{recursive}]  RecipeInCategory
    usesIngredients [{recursive}]  Uses
    ratings         [{recursive}]  RecipeRating
  }
  unit {
    name      string
    unitType* enum    ALLOWED:0, 1
    symbol*   string
  }
  ingredient {
    name             string
    rarity           enum          ALLOWED:0, 1, 2
    weightToVolRatio number
    suppliedBy       [{recursive}] Supplies
  }
}
```

REQUEST BODY - text/json

```
{
  recipeId            integer
  unitName            string
  ingrName            string
  quantity            number
  recipe {
    id              integer
    name*           string
    steps           string
    cholesterol     number
    fat             number
    sodium          number
    protein         number
    carbs           number
    calories        number
    authorId        string
    author          {recursive}    Customer
    categories      [{recursive}]  RecipeInCategory
    usesIngredients [{recursive}]  Uses
    ratings         [{recursive}]  RecipeRating
  }
  unit {
    name      string
```

```
    unitType* enum    ALLOWED:0, 1
    symbol*   string
  }
  ingredient {
    name             string
    rarity           enum           ALLOWED:0, 1, 2
    weightToVolRatio number
    suppliedBy       [{recursive}]  Supplies
  }
}
```

REQUEST BODY - application/*+json

```
{
  recipeId           integer
  unitName           string
  ingrName           string
  quantity           number
  recipe {
    id               integer
    name*            string
    steps            string
    cholesterol      number
    fat              number
    sodium           number
    protein          number
    carbs            number
    calories         number
    authorId         string
    author           {recursive}    Customer
    categories       [{recursive}]  RecipeInCategory
    usesIngredients  [{recursive}]  Uses
    ratings          [{recursive}]  RecipeRating
  }
  unit {
    name      string
    unitType* enum    ALLOWED:0, 1
    symbol*   string
  }
  ingredient {
    name             string
    rarity           enum           ALLOWED:0, 1, 2
    weightToVolRatio number
    suppliedBy       [{recursive}]  Supplies
  }
}
```

RESPONSE

STATUS CODE - 201: Success

RESPONSE MODEL - application/json

```
{
  recipeId* integer
  unitName* string
  ingrName* string
  quantity  number
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

15.4 DELETE /api/Uses

Delete an existing Uses entry

Delete a Uses entry from the Uses relation based on recipe_id,
if the entry exists in the Uses relation of the database.

A Delete operation to delete a Uses entry is performed.

REQUEST

REQUEST BODY - application/json

```
{
  recipeId* integer
  unitName* string
  ingrName* string
  quantity  number
}
```

REQUEST BODY - text/json

```
{
  recipeId* integer
  unitName* string
  ingrName* string
  quantity  number
}
```

REQUEST BODY - application/*+json

```
{
  recipeId* integer
  unitName* string
  ingrName* string
  quantity  number
}
```

RESPONSE

STATUS CODE - 204: Success

STATUS CODE - 404: Not Found

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  type      string
  title     string
  status    integer
  detail    string
  instance  string
}
```
