

API Reference

RecipEase.Server

API Version: 1.0

INDEX

1. CUSTOMER	4
1.1 GET /api/Customer	4
1.2 POST /api/Customer	4
1.3 GET /api/Customer/{id}	5
1.4 PUT /api/Customer/{id}	6
1.5 DELETE /api/Customer/{id}	7
2. INGR	9
2.1 GET /api/Ingr	9
2.2 GET /api/Ingr/{id}	9
3. INGRINSHOPPINGLIST	11
3.1 GET /api/IngrInShoppingList/{id}	11
3.2 GET /api/IngrInShoppingList	11
3.3 PUT /api/IngrInShoppingList	12
3.4 POST /api/IngrInShoppingList	12
3.5 DELETE /api/IngrInShoppingList	13
4. OIDCCONFIGURATION	15
4.1 GET /_configuration/{clientId}	15
5. RECIPE	16
5.1 GET /api/Recipe/{id}	16
5.2 PUT /api/Recipe/{id}	17
5.3 DELETE /api/Recipe/{id}	18
5.4 GET /api/Recipe	19
5.5 POST /api/Recipe	20
6. RECIPECOLLECTION	22
6.1 GET /api/RecipeCollection	22
6.2 POST /api/RecipeCollection	23
6.3 DELETE /api/RecipeCollection/{title}	24
7. RECIPEINCOLLECTION	25
7.1 GET /api/RecipeInCollection	25
7.2 POST /api/RecipeInCollection	25
7.3 DELETE /api/RecipeInCollection	27
8. RECIPERATING	28
8.1 POST /api/RecipeRating	28
9. SHOPPINGLIST	30
9.1 GET /api/ShoppingList/{userId}	30
9.2 PUT /api/ShoppingList/{userId}	30

10. SUPPLIER	32
10.1 GET /api/Supplier	32
10.2 POST /api/Supplier	32
10.3 GET /api/Supplier/{id}	33
10.4 PUT /api/Supplier/{id}	34
10.5 DELETE /api/Supplier/{id}	35
11. SUPPLIES	37
11.1 GET /api/Supplies	37
11.2 POST /api/Supplies	37
11.3 GET /api/Supplies/{id}	38
11.4 PUT /api/Supplies/{id}	39
11.5 DELETE /api/Supplies/{id}	40
12. UNIT	42
12.1 GET /api/Unit	42
12.2 GET /api/Unit/{id}	42
13. UNITCONVERSE	44
13.1 GET /api/UnitConverse/all	44
13.2 GET /api/UnitConverse	44
14. USER	46
14.1 GET /api/User	46
14.2 POST /api/User	46
14.3 GET /api/User/{id}	48
14.4 PUT /api/User/{id}	49
14.5 DELETE /api/User/{id}	50
15. USES	52
15.1 GET /api/Uses	52
15.2 POST /api/Uses	52
15.3 GET /api/Uses/{id}	53
15.4 PUT /api/Uses/{id}	54
15.5 DELETE /api/Uses/{id}	55

API

1. CUSTOMER

1.1 GET /api/Customer

Returns all Customer credential information.

Retrieves all items and all attributes from the 'customer' table.

A 'select*' query with a 'where' clause to find the list of usernames and their associated attributes.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
[{  
  Array of object:  
    userId      string  
    customerName string  
    age         integer  
    weight      number  
    favMeal     enum    ALLOWED:0, 1, 2  
}]
```

1.2 POST /api/Customer

Add a new Customer

Add a new Customer to the Customer relation, if the username does not exist in the Customer relation of the database.

An Insert operation to insert a new Customer is performed.

REQUEST

REQUEST BODY - application/json

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

RESPONSE

STATUS CODE - 201: Success

RESPONSE MODEL - application/json

```
{  
  userId      string
```

```

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

```

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
}

```

1.3 GET /api/Customer/{id}

Returns the Customer with the given username id.

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

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

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	string	The Username ID 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
}
```

1.4 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 username 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
}

```

1.5 DELETE /api/Customer/{id}

Delete a Customer user

Delete a Customer from the database,
if the customer exists in the Customer relation of the database.
The authenticated user must be the user to be deleted.

A Delete operation to delete a Customer is performed.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	string	The username of the Customer to delete.

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  
}
```

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
}
```

3. INGRINSHOPPINGLIST

3.1 GET /api/IngrInShoppingList/{id}

Returns all the items 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: select * IngrInShoppingList with UserId = id

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	string	id of the user who have the shopping list to converse to.

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

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

3.2 GET /api/IngrInShoppingList

Returns the specific item in the IngrInShoppingList

functionalities : Retrieves 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: select * IngrInShoppingList with UserId = id and IngrName = iname and UnitName = uname

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
id	string	
uname	string	

NAME	TYPE	DESCRIPTION
iname	string	

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

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

3.3 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

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
id	string	

REQUEST BODY - application/json

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

RESPONSE

STATUS CODE - 200: Success

3.4 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  integer
}
```

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

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

3.5 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  integer
}
```

REQUEST BODY - text/json

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

REQUEST BODY - application/*+json

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

RESPONSE

STATUS CODE - 200: Success

4. OIDC CONFIGURATION

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
  averageRating 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
}
```

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
*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
  averageRating number
}
```

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
  averageRating 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
}
```

6. RECIPECOLLECTION

6.1 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  
}
```

6.2 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
}
```

6.3 DELETE /api/RecipeCollection/{title}

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

PATH PARAMETERS

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

RESPONSE

STATUS CODE - 200: Success

7. RECIPEINCOLLECTION

7.1 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

REQUEST BODY - application/json

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

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

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

7.2 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
}
```

7.3 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

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 - 200: Success

8. RECIPERATING

8.1 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. SHOPPINGLIST

9.1 GET /api/ShoppingList/{userId}

Returns an user's shopping list

functionalities : Retrieves an user's shopping list with accordinging id.

database: ShoppingList, User

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

query: select * ShoppingList with UserId = userId

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*userId	string	id of the user who have the shopping list.

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
{
  userId*      string
  name*        string
  lastUpdate   string
  numIngredients integer
}
```

9.2 PUT /api/ShoppingList/{userId}

Edit an user's shopping list

functionalities : Edit an user's shopping list with accordinging id.

database: ShoppingList, User

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

query: update ShoppingList Set (some values) where UserId = userId

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*userId	string	id of the user who have the shopping list.

REQUEST BODY - application/json

```
{
  userId*      string
  name*        string
  lastUpdate   string
  numIngredients integer
}
```

RESPONSE

STATUS CODE - 200: Success

10. SUPPLIER

10.1 GET /api/Supplier

REQUEST

No request parameters

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  
  } ]
```

10.2 POST /api/Supplier

Add a new Supplier user

Add a new Supplier to the Supplier relation,
if the username does not exist in the Supplier relation of the database.

An Insert operation to insert a new Supplier user is performed.

REQUEST

REQUEST BODY - application/json

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

RESPONSE

STATUS CODE - 201: Success

RESPONSE MODEL - application/json

```
{  
  userId          string  
  email           string  
  phoneNo        string
```



```

website*      string
supplierName* string
storeVisitCount integer >=0
                                DEFAULT:0
}

```

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
}

```

10.3 GET /api/Supplier/{id}

Returns the Supplier with the given username id.

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

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

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	string	The Username ID 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
}
```

10.4 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 username 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
}
```

10.5 DELETE /api/Supplier/{id}

Delete a Supplier user

Delete a Supplier from the database,
if the user exists in the Supplier relation of the database.
The authenticated user must be the user to be deleted.

A Delete operation to delete a Supplier is performed.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	string	The username of the Supplier to delete.

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
}
```

11. SUPPLIES

11.1 GET /api/Supplies

Returns list of Suppliers' Ingredient stock

Retrieves all ingredients that all suppliers supply, and their associated attributes, from the `supplies` table.

A 'select*' query with a 'where' clause to find the list of ingredients and their associated attributes is performed.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

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

11.2 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.

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

REQUEST

REQUEST BODY - application/json

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

RESPONSE

STATUS CODE - 201: Success

RESPONSE MODEL - application/json

```
{  
  ingrName*  string  
  unitName*  string  
  userId*    string
```

```
    quantity integer
}
```

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
}
```

11.3 GET /api/Supplies/{id}

Returns the list of suppliers of an ingredient

Retrieves the object with the given ingredient name, from the `Supplies` table, if it exists.

A 'select user,ingredient' query with a 'where' clause to find the ingredient and its associated suppliers.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	string	The ingredient name in the Supplies table.

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

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

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
}

```

11.4 PUT /api/Supplies/{id}

Update the information of an existing supplies entry

Updates information of an existing supplies entry in the supplies table of the database.
The authenticated user must be the user to update the entry.

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

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	string	The ingredient to update.

REQUEST BODY - application/json

```

{
  ingrName* string
  unitName* string
  userId*   string
  quantity  integer
}

```

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
}
```

11.5 DELETE /api/Supplies/{id}

Delete an existing supplies entry

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

A Delete operation to delete a supplies entry is performed.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	string	The supplies entry to delete.

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
}
```

12. UNIT

12.1 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  
  } ]
```

12.2 GET /api/Unit/{id}

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
*id	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
}
```

13. UNITCONVERSE

13.1 GET /api/UnitConverse/all

Get all unit conversion

functionalities : retrieve all unitconversion in UnitConversion

database: UnitConversion

constraints: no constraints

query: select * from UnitConversion

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
[ {
  Array of object:
    convertsToUnitName*  string
    convertsFromUnitName* string
    ratio*               integer
}]
```

13.2 GET /api/UnitConverse

Get unit conversion from one unit to another

functionalities : retrieve all unitconversion in UnitConversion

database: UnitConversion, Unit

constraints: two unit must be the same unit type

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

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
id1	string	
id2	string	

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

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

14. USER

14.1 GET /api/User

Returns all User credential information.

Retrieves all items and all attributes from the `user` table.

A 'select*' query with a 'where' clause to find the list of usernames and their associated attributes.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
[ {  
  Array of object:  
    id                string  
    userName          string  
    normalizedUserName string  
    email             string  
    normalizedEmail   string  
    emailConfirmed    boolean  
    passwordHash      string  
    securityStamp     string  
    concurrencyStamp  string  
    phoneNumber       string  
    phoneNumberConfirmed boolean  
    twoFactorEnabled  boolean  
    lockoutEnd        string  
    lockoutEnabled    boolean  
    accessFailedCount integer  
    loginCount        integer DEFAULT:0  
  } ]
```

14.2 POST /api/User

Add a new User

Add a new User to the User relation,
if the username does not exist in the User relation of the database.

An Insert operation to insert a new User is performed.

REQUEST

REQUEST BODY - application/json

```
{  
  id                string  
  userName          string  
  normalizedUserName string  
  email             string
```

```

normalizedEmail    string
emailConfirmed     boolean
passwordHash       string
securityStamp      string
concurrencyStamp   string
phoneNumber        string
phoneNumberConfirmed boolean
twoFactorEnabled   boolean
lockoutEnd         string
lockoutEnabled     boolean
accessFailedCount  integer
loginCount         integer DEFAULT:0
}

```

RESPONSE

STATUS CODE - 201: Success

RESPONSE MODEL - application/json

```

{
  id            string
  userName      string
  normalizedUserName string
  email        string
  normalizedEmail string
  emailConfirmed boolean
  passwordHash  string
  securityStamp string
  concurrencyStamp string
  phoneNumber   string
  phoneNumberConfirmed boolean
  twoFactorEnabled boolean
  lockoutEnd    string
  lockoutEnabled boolean
  accessFailedCount integer
  loginCount    integer DEFAULT:0
}

```

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
}

```

}

14.3 GET /api/User/{id}

Returns the User with the given username id.

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

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

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	string	The Username ID of the User to retrieve.

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
{
  id                string
  userName          string
  normalizedUserName string
  email             string
  normalizedEmail   string
  emailConfirmed    boolean
  passwordHash      string
  securityStamp     string
  concurrencyStamp  string
  phoneNumber       string
  phoneNumberConfirmed boolean
  twoFactorEnabled  boolean
  lockoutEnd        string
  lockoutEnabled    boolean
  accessFailedCount integer
  loginCount        integer 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
}
```

14.4 PUT /api/User/{id}

Update the information of an existing user

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

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

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	string	The username of the User to update.

REQUEST BODY - application/json

```
{
  id                string
  userName          string
  normalizedUserName string
  email            string
  normalizedEmail   string
  emailConfirmed    boolean
  passwordHash      string
  securityStamp     string
  concurrencyStamp  string
  phoneNumber       string
  phoneNumberConfirmed boolean
  twoFactorEnabled  boolean
  lockoutEnd        string
  lockoutEnabled    boolean
  accessFailedCount integer
  loginCount        integer 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
}
```

14.5 DELETE /api/User/{id}

Delete aUser

Delete a User from the database,
if the user exists in the User relation of the database.
The authenticated user must be the user to be deleted.

A Delete operation to delete a User is performed.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	string	The username of the User to delete.

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
}
```

15. USES

15.1 GET /api/Uses

Returns list of all ingredients used by all recipes

Retrieves all ingredients that all recipes use, and their associated units, from the `Uses` table.

A 'select*' query with a 'where' clause to find the list of ingredients used by all recipes, and their associated attributes, is performed.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

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

15.2 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 integer  
}
```

REQUEST BODY - text/json

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

REQUEST BODY - application/*+json

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

RESPONSE

STATUS CODE - 201: Success

RESPONSE MODEL - application/json

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

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
}
```

15.3 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.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	int32	

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

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

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
}
```

15.4 PUT /api/Uses/{id}

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

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	int32	The recipe id in the Uses table.

REQUEST BODY - application/json

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

```
    quantity  integer
}
```

REQUEST BODY - text/json

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

REQUEST BODY - application/*+json

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

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
}
```

15.5 DELETE /api/Uses/{id}

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

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	int32	The recipe_id to delete.

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
}
```


