CS493
Final Project
Joseph Sullivan
ONID: sullijos

Final Project

API Description

My API represents a collection of recipes and ingredients. Both recipes and ingredients have an owner. A user must be logged in to create recipes and ingredients, and only that user/owner can update, delete, and create relationships for their own recipes and ingredients. Individual ingredients and recipes, an owner's lists of recipes and ingredients, and lists of all recipes and ingredients can be viewed by any user and can be viewed even when a user is not logged in.

Relationships can be created between a recipe and an ingredient that is owned by the same user. This relationship represents an ingredient being part of a recipe. This is a many to many relationship where an ingredient can be in multiple recipes and recipes can have multiple ingredients.

Auth0 Client Credentials Grant

For the final project, I registered an API and application with Auth0. To create new users, my application uses the Client Credentials Grant to obtain an access token which allows it to call Auth0 management API methods. This is how the flow is used with my Postman tests:

1. I send a POST /users request from Postman to my server that contains an email address and password in the request body.
2. My server sends its client id and client secret to Auth0, requesting an access token.
3. Auth0 sends back an access token.
4. My server sends the access token back as a Bearer token in the Authorization header of a POST request that also contains the new user's email address, password, and Auth0 connection group.
5. My server returns 201 to the postman request if the user was created, or 403 upon failure.

Auth0 Resource Owner Password Grant

I chose the Resource Owner Password Grant authentication flow for user authentication as it allows me to obtain a JWT for a user by submitting their username and password to my server which passes these credentials along to Auth0 in exchange for a JWT. While this is one of the

less secure flows available and should only be used when more secure flows are not an option, I chose it as it allows me to test all the required functionality of my API via Postman tests.

Typically, a user would submit their username and password to my application via a form, and I would then send these credentials to Auth0 in exchange for a JWT. For the purposes of testing via Postman, this is how the flow is implemented:

1. I send a user's username and password to my server in the body of a POST /users/login request in Postman.
2. My server then sends these credentials, my client id, and client secret to Auth0 with its own POST request.
3. If the credentials are valid, Auth0 returns a JWT.
4. This JWT gets returned to my Postman request by my server's response.

Example user


```
{
  "email": "user0@cs493-sullijos-finalproject.com",
  "email_verified": false,
  "updated_at": "2019-06-11T01:50:49.740Z",
  "user_id": "auth0|5cff08f9aa996b0e2d36e877",
  "name": "user0@cs493-sullijos-finalproject.com",
  "picture":
"https://s.gravatar.com/avatar/95aabf32cd9b6614ad42dde5e11663f7?s=480&r=pg&d=https%
3A%2F%2Fcdn.auth0.com%2Favatars%2Fus.png",
  "nickname": "user0",
  "identities": [
    {
      "connection": "Username-Password-Authentication",
      "user_id": "5cff08f9aa996b0e2d36e877",
      "provider": "auth0",
      "isSocial": false
    }
  ],
  "created_at": "2019-06-11T01:50:49.740Z"
}
```


Note: Only a username and password is submitted to Auth0. The remaining properties are created automatically by Auth0.

CS493
Final Project
Joseph Sullivan
ONID: sullijos

Example ingredient

```
{
  "units": “cups", // The unit of measure for the ingredient
  "quantity": 10, // The quantity of units of the ingredient needed
  "recipes": [ // List of recipes that the ingredient belongs to
    {
      "id": "5761901967441920", // The recipe’s id
      "self": "localhost:8080/recipes/5761901967441920" // A link to the recipe
    }
  ],
  "owner": "user0@cs493-sullijos-finalproject.com", // The ingredient’s owner
  "name": "milk", // The ingredient’s name
  "id": "5751016104394752", // The ingredient’s id
  "self": "localhost:8080/ingredients/5751016104394752" A link to the ingredient
}
```

Example recipe

```
{
  "servings": 10, // Number of servings that recipe yields
  "owner": "user0@cs493-sullijos-finalproject.com", // Owner of the recipe
  "name": "cookies", // name of the recipe
  "ingredients": [ // List of ingredients in the recipe
    {
      "id": "5751016104394752", // ingredients id
      "self": "localhost:8080/ingredients/5751016104394752" // Link to the ingredient
    }
  ],
  "prep_time": "01:23", // Estimated amount of time that it takes to make the recipe
  "id": "5761901967441920", // Recipe’s id
  "self": "localhost:8080/recipes/5761901967441920" // A link to the recipe
}
```

CS493
Final Project
Joseph Sullivan
ONID: sullijos

Request Examples

Note: All requests that have an Accept header that doesn't include application/json will be rejected with status code 406. Any HTTP methods that are attempted on routes that are not listed below will be rejected with a status code of 405.

POST /users

- Creates an Auth0 user with the email address and password sent in the request body.
- Request body:

```
{
  "email": "user0@cs493-sullijos-finalproject.com",
  "password": "user0Password"
}
```

- Response status codes:
  - 201 if successful
  - 403 if user already exists
- Example response body:

```
{
  "email": "user0@cs493-sullijos-finalproject.com",
  "email_verified": false,
  "updated_at": "2019-06-11T02:46:30.599Z",
  "user_id": "auth0|5cff16064870a60f098ac63e",
  "name": "user0@cs493-sullijos-finalproject.com",
  "picture":
"https://s.gravatar.com/avatar/95aabf32cd9b6614ad42dde5e11663f7?s=480&r=pg&d=
https%3A%2F%2Fcdn.auth0.com%2Favatars%2Fus.png",
  "nickname": "user0",
  "identities": [{
    "connection": "Username-Password-Authentication",
    "user_id": "5cff16064870a60f098ac63e",
    "provider": "auth0",
    "isSocial": false
  }],
  "created_at": "2019-06-11T02:46:30.599Z"
}
```

CS493
Final Project
Joseph Sullivan
ONID: sullijos

POST /users/login

- Sends a user's username and password in the request body to get a JWT for the user.
- Request body:

```
{
  "username":"user0@cs493-sullijos-finalproject.com",
  "password":"user0Password"
}
```

- Response statuses
  - 200 upon success
  - 403 if user doesn't exist or password is incorrect
- Example response body:


```
{
  "access_token": "t9LE7cV8-zMNcOKdk_u7BkxnankmBmiz",
  "id_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Ik5Ua3lOREk0TjBaRlFqbEVOMEUwT1R
aR05FTTVNVGt4UkRJMk1EZ3hRekEwUlVNeVJqTTNNQSJ9.eyJuaWNrbmFtZSI6InVzZXIwIi
wibmFtZSI6InVzZXIwQGNzNDkzLXN1bGxpam9zLWZpbmFscHJvamVjdC5jb20iLCJwaWN0
dXJlIjoiaHR0cHM6Ly9zLmdyYXZhdGFyLmNvbS9hdmF0YXIvOTVhYWJmMzJjZDliNjYxNGFk
NDJkZGU1ZTExNjYzZjc_cz00ODAmcj1wZyZkPWh0dHBzJTNBJTJGJTJGY2RuLmF1dGggwLm
NvbSUyRmF2YXRhcnMlMkZ1cy5wbmciLCJ1cGRhdGVkX2F0IjoiMjAxOS0wNi0xMVQwMjo
0NjozMi4xNTdaIiwiZW1haWwiOiJ1c2VyMEBjczQ5My1zdWxsaWpvcy1maW5hbHByb2pl
Y3QuY29tIiwiZW1haWxfdmVyaWZpZWQiOmZhbHNlLCJpc3MiOiJodHRwczovL2Rldi1jd3F
iNGdocS5hdXRoMC5jb20vIiwic3ViIjoiYXV0aDB8NWNmZjE2MDY0ODcwYTYwZjA5OGFjNj
NlIiwiYXVkIjoiSDk4TlNOU3YzY01SbjNKMTdrcjNDSjBldTNfenhrRWwiLCJpYXQiOjE1NjAyM
jExOTIsImV4cCI6MTU2MDI1NzE5Mn0.bZQ5eMYItC1tVwDvz12GtKkSUoclOjMo11egztc2c
L35e4RjjOyPULXvfHfOxoPIwKsWQqV6S3ZORQXx6cXOo-
Cu6ePi5p1_oV6KgENLY3CKn7uNSBTM2cNYCpEFP6X-Qeei9SnibOZOkvLe-
BZdo0rTB6RXoqEMbu65NuxfCOm8-4g5EpH6iA1r-dHhF7wajwjTxJ_a7-
g_dQgfaiL91P332xmzixRFJ-FuiQsWOFjehVej_zWEoH9SXdK1-W-
OazZy6F6WVWck8AWzgq2J9EQeS2OPed9mYUH9V-YWdPc92KlBIbI3szR-
ih3JhWrJclZVtz65P2HEtxB6_U-KBQ",
  "scope": "openid profile email address phone",
  "expires_in": 86400,
  "token_type": "Bearer"
```

}

GET /users/{user_id}/recipes

- Gets a list of recipes that belong to the specified user.
- No additional data needs to be sent.
- Response statuses
  - status 200 upon success
  - If a nonexistent user id is sent, an empty array is returned with status 200
- Example response body:

[{
"servings": 12,
"owner": "user1@cs493-sullijos-finalproject.com",
"name": "recipe2",
"ingredients": [],
"prep_time": "01:23",
"id": "5178081291534336",
"self": "localhost:8080/recipes/5178081291534336"
}, {
"owner": "user1@cs493-sullijos-finalproject.com",
"name": "recipe3",
"ingredients": [],
"prep_time": "01:23",
"servings": 13,
"id": "5658519688708096",
"self": "localhost:8080/recipes/5658519688708096"
}]

GET /users/{user_id}/ingredients

- Gets a list of ingredients that belong to the specified user.
- No additional data needs to be sent.
- Response statuses
  - status 200 upon success
  - If a nonexistent user id is sent, an empty array is returned with status 200
- Example response body:

[{
"units": "unitsUpdate",

```
    "quantity": 100,
    "recipes": [{
      "id": "6215736263442432",
      "self": "localhost:8080/recipes/6215736263442432"
    }],
    "owner": "user0@cs493-sullijos-finalproject.com",
    "name": "ingredientUpdate",
    "id": "5127307865882624",
    "self": "localhost:8080/ingredients/5127307865882624"
  }, {
    "owner": "user0@cs493-sullijos-finalproject.com",
    "name": "ingredient0",
    "units": "unit0",
    "quantity": 10,
    "recipes": [],
    "id": "5162453918810112",
    "self": "localhost:8080/ingredients/5162453918810112"
  }]
```


POST /recipes

- Creates a new recipe
- Authorization header must contain the id_token received from the response to a POST /user/login request formatted as a bearer token.
- Example Authorization header:
  Bearer
  eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Ik5Ua3lOREk0TjBaRlFqbEVOMEUwT1RaR05FTTVNVGt4UkRJMk1EZ3hRekEwUlVNeVJqTTNNQSJ9.eyJuaWNrbmFtZSI6InVzZXIwIiwibmFtZSI6InVzZXIwQGNzNDkzLXN1bGxpam9zLWZpbmFscHJvamVjdC5jb20iLCJwaWN0dXJlIjoiaHR0cHM6Ly9zLmdyYXZhdGFyLmNvbS9hdmF0YXIvOTVhYWJmMzJjZDliNjYxNGFkNDJkZGU1ZTExNjYzZjc_cz00ODAmcj1wZyZkPWh0dHBzJTNBJTJGJTJGY2RuLmF1dGggLmNvbSUyRmF2YXRhcnMlMkZ1cy5wbmciLCJ1cGRhdGVkX2F0IjoiMjAxOS0wNi0xMVQwMjo0NjozMi44ODBaIiwiZW1haWwiOiJ1c2VyMEBjczQ5My1zdWxsaWpvcy1maW5hbHByb2plY3QuY29tIiwiZW1haWxfdmVyaWZpZWQiOmZhbHNlLCJpc3MiOiJodHRwczovL2Rldi1jd3FiNGdocS5hdXRoMC5jb20vIiwic3ViIjoiYXV0aDB8NWNmZjE2MDY0ODcwYTYwZjA5OGFjNjNlIiwiYXVkIjoiSDk4TlNOU3YzY01SbjNKMTdrcjNDSjBldTNfenhrRWwiLCJpYXQiOjE1NjAyMjExOTIsImV4cCI6MTU2MDI1NzE5Mn0.l1L0_tN6-jznrWV97S5lSEWFvhULGDoL8KrRzVCWBOkPYOs0Iv68IYAMwQd7-RcRJW5KvS-ZD9n7aHjAtfPpMCNq_tue1rqhWeULXfW5Gxxf5IHWDlRdIk2FyHLeCUANQb8pmUgNfhN

QUdCC-36hUiG19mVEdkPHDzt_OgM_WvIhPJ8A-lbBFFlmrF2afUjENyGWJSkZG2iJaP_e_c7XDXdMy5swQ6MomORpnNw5b9Tf0lNhLoWgdl97svN96uUHPmakJCpzsmCeT6iwAPxMfbTu-TnxCMH_cLpw3v-pmsReJLhEljDPFzZJmNGSg8Fk-T4nfKscTc1pVkHc_CgBnQ

- The request body must contain a JSON object with a name property that must be a string with at least one non-white-space character, a prep_time property that must be a string duration that is formatted with minutes and hours like so: mm:hh, and a servings property that is a number greater than zero.
- Request body example:

```
{
  "name":"recipe0",
  "prep_time":"01:23",
  "servings":10
}
```

- Response statuses:
  - 201 upon successful create of the recipe.
  - 401 if the JWT is missing or invalid.
  - 422 if one or more recipe properties are invalid
- Response Location header contains link to new resource if status is 201
- Example response Location header: localhost:8080/recipes/5649050225344512

GET /recipes

- Gets all recipes regardless of user. Response includes a count of the number of recipes in the database and has a page limit of five. If more than 5 exist, a next page link is sent.
- No additional data sent.
- Response statuses
  - 200 upon success.
- Example response body:

```
{
  "count": 6,
  "recipes": [{
    "ingredients": [],
    "prep_time": "01:23",
    "servings": 15,
    "owner": "user1@cs493-sullijos-finalproject.com",
    "name": "recipe5",
    "id": "4808361379889152",
```

```
            "self": "localhost:8080/recipes/4808361379889152"
        }, {
            "ingredients": [],
            "prep_time": "01:23",
            "servings": 14,
            "owner": "user1@cs493-sullijos-finalproject.com",
            "name": "recipe4",
            "id": "5086100271923200",
            "self": "localhost:8080/recipes/5086100271923200"
        }, {
            "owner": "user1@cs493-sullijos-finalproject.com",
            "name": "recipe2",
            "ingredients": [],
            "prep_time": "01:23",
            "servings": 12,
            "id": "5178081291534336",
            "self": "localhost:8080/recipes/5178081291534336"
        }, {
            "ingredients": [],
            "prep_time": "11:11",
            "servings": 100,
            "owner": "user0@cs493-sullijos-finalproject.com",
            "name": "recipeUpdate",
            "id": "5649050225344512",
            "self": "localhost:8080/recipes/5649050225344512"
        }, {
            "ingredients": [],
            "prep_time": "01:23",
            "servings": 13,
            "owner": "user1@cs493-sullijos-finalproject.com",
            "name": "recipe3",
            "id": "5658519688708096",
            "self": "localhost:8080/recipes/5658519688708096"
        }],
        "next":
    "localhost:8080/recipes?cursor=CjsSNWodbX5jczQ5My1zdWxsaWpvcy1maW5hbHByb2
    plY3RyFAsSB1JlY2lwZXMYgICAwMzMhgoMGAAgAA=="
    }
```


GET /recipes?cursor={next_page_cursor}

- Returns the next page of the list of all recipes. If it is the last page, no next page link is provided.
- No additional data sent.
- Response statuses
    - 200 upon success.
- Example response body:

{
  "count": 6,
  "ingredients": [{
    "quantity": 12,
    "recipes": [],
    "owner": "user1@cs493-sullijos-finalproject.com",
    "name": "ingredient2",
    "units": "unit2",
    "id": "6319865665159168",
    "self": "localhost:8080/ingredients/6319865665159168"
  }]
}

GET /recipes/{recipe_id}

- Gets the recipe of the specified id.
- No additional data sent.
- Response statuses
    - 200 upon success
    - 404 if a bad id is sent
- Example response body:

{
  "servings": 10,
  "owner": "user0@cs493-sullijos-finalproject.com",
  "name": "recipe0",
  "ingredients": [],
  "prep_time": "01:23",
  "id": "5649050225344512",
  "self": "localhost:8080/recipes/5649050225344512"
}

PUT /recipes/{recipe_id}

- Updates a recipe with the new properties sent in the request body. The properties sent must be valid and must include name, prep_time, servings, and ingredients properties. The ingredients property must be an array of objects that each contain a valid ingredient id that belongs to the user. Any ingredients removed from the recipe's ingredient list as a result will have the recipe removed from their recipe list as well.
- Authorization header must contain the id_token received from the response to a POST /user/login request formatted as a bearer token.
- Example request body:
  ```
  {
    "name":"recipeUpdate",
    "prep_time":"11:11",
    "servings":100,
    "ingredients":[{"id":"5127307865882624"}]
  }
  ```
- Response statuses
  - 204 upon success
  - 401 if JWT is missing or invalid
  - 403 if user of JWT does not match owner of recipe.
  - 404 if recipe does not exist
  - 422 if user attempts to update recipe with bad properties
- If status is 204, Response Location header contains a link to the updated recipe
- Example Location header:
  localhost:8080/recipes/5649050225344512

DELETE /recipes/{recipe_id}

- Deletes the specified recipe from the database. Any ingredients that were in the recipe's list will have the recipe removed from their recipe list.
- Authorization header must contain the id_token received from the response to a POST /user/login request formatted as a bearer token.
- Response statuses
  - 204 upon success
  - 401 if JWT is missing or invalid
  - 403 if user of JWT does not match the owner of the recipe
  - 404 if recipe id is invalid

POST /ingredients

- Creates a new ingredient

- Authorization header must contain the id_token received from the response to a POST /user/login request formatted as a bearer token.
- The request body must contain a JSON object with a name property that must be a string with at least one non-white-space character, a units property that must be a string with at least one non-white-space character, and a quantity property that is a number greater than zero.
- Request body example:
- {
  "name":"ingredient0",
  "units":"unit0",
  "quantity":10
  }
- Response statuses:
  - 201 upon successful create of the ingredient.
  - 401 if the JWT is missing or invalid.
  - 422 if one or more ingredient properties are invalid
- Response Location header contains link to new resource if status is 201
- Example response Location header:
  localhost:8080/ingredients/5649050225344512

GET /ingredients

- Gets all ingredients regardless of user. Response includes a count of the number of ingredients in the database and has a page limit of five. If more than 5 exist, a next page link is sent.
- No additional data sent.
- Response statuses
  - 200 upon success.
- Example response body:
  {
  "count": 6,
  "ingredients": [{
  "units": "unit3",
  "quantity": 13,
  "recipes": [],
  "owner": "user1@cs493-sullijos-finalproject.com",
  "name": "ingredient3",
  "id": "5113583801204736",
  "self": "localhost:8080/ingredients/5113583801204736"

```
    }, {
      "units": "unitsUpdate",
      "quantity": 100,
      "recipes": [{
        "id": "6215736263442432",
        "self": "localhost:8080/recipes/6215736263442432"
      }],
      "owner": "user0@cs493-sullijos-finalproject.com",
      "name": "ingredientUpdate",
      "id": "5127307865882624",
      "self": "localhost:8080/ingredients/5127307865882624"
    }, {
      "owner": "user1@cs493-sullijos-finalproject.com",
      "name": "ingredient4",
      "units": "unit4",
      "quantity": 14,
      "recipes": [],
      "id": "5133118755307520",
      "self": "localhost:8080/ingredients/5133118755307520"
    }, {
      "units": "unit0",
      "quantity": 10,
      "recipes": [],
      "owner": "user0@cs493-sullijos-finalproject.com",
      "name": "ingredient0",
      "id": "5162453918810112",
      "self": "localhost:8080/ingredients/5162453918810112"
    }, {
      "units": "unit5",
      "quantity": 15,
      "recipes": [],
      "owner": "user1@cs493-sullijos-finalproject.com",
      "name": "ingredient5",
      "id": "6253207772725248",
      "self": "localhost:8080/ingredients/6253207772725248"
    }],
    "next":
"localhost:8080/ingredients?cursor=Cj8SOWodbX5jczQ5My1zdWxsaWpvcy1maW5hbH
Byb2plY3RyGAsSC0luZ3JlZGllbnRzGICAgKCm6I0LDBgAIAA="
```

}

GET /ingredients?cursor={next_page_cursor}

- Returns the next page of the list of all ingredients. If it is the last page, no next page link is provided.
- No additional data sent.
- Response statuses
  - 200 upon success.
- Example response body:

{
"count": 6,
"ingredients": [{
"quantity": 12,
"recipes": [],
"owner": "user1@cs493-sullijos-finalproject.com",
"name": "ingredient2",
"units": "unit2",
"id": "6319865665159168",
"self": "localhost:8080/ingredients/6319865665159168"
}]
}

GET /ingredients/{ingredients_id}

- Gets the ingredient of the specified id.
- No additional data sent.
- Response statuses
  - 200 upon success
  - 404 if a bad id is sent
- Example response body:

{
"units": "unit0",
"quantity": 10,
"recipes": [],
"owner": "user0@cs493-sullijos-finalproject.com",
"name": "ingredient0",
"id": "5162453918810112",

"self": "localhost:8080/ingredients/5162453918810112"
}

PUT /ingredients/{ingredients_id}

- Updates an ingredient with the new properties sent in the request body. The properties sent must be valid and must include name, units, quantity, and recipes properties. The recipes property must be an array of objects that each contain a valid recipe id that belongs to the user. Any ingredients removed from the ingredient's recipe list as a result will have the ingredient removed from their ingredient list as well.
- Authorization header must contain the id_token received from the response to a POST /user/login request formatted as a bearer token.
- Example request body:
- {
  "name":"ingredientUpdate",
  "units":"unitsUpdate",
  "quantity":100,
  "recipes":[{"id":"6215736263442432"}]
  }
- Response statuses:
  - 204 upon success
  - 401 if JWT is missing or invalid
  - 403 if user of JWT does not match owner of ingredient.
  - 404 if ingredient does not exist
  - 422 if user attempts to update ingredient with bad properties
- If status is 204, Response Location header contains a link to the updated ingredient
- Example Location header:
  localhost:8080/ingredients/5649050225344512

DELETE /ingredients/{ingredients_id}

- Deletes the specified ingredient from the database. Any recipes that were in the ingredient's list will have the ingredient removed from their ingredient list.
- Authorization header must contain the id_token received from the response to a POST /user/login request formatted as a bearer token.
- Response statuses
  - 204 upon success
  - 401 if JWT is missing or invalid
  - 403 if user of JWT does not match the owner of the ingredient

    - 404 if ingredient id is invalid

PUT /recipes/{recipe_id}/ingredients/{ingredient_id}

- Adds the specified recipe to the specified ingredient's recipe list and vice versa. Both entities must have the same owner.
- Authorization header must contain the id_token received from the response to a POST /user/login request formatted as a bearer token.
- Response statuses
    - 204 upon success
    - 401 if JWT is missing or invalid
    - 403 if user of JWT does not match owner of either entity
    - 404 if either id is invalid

DELETE /recipes/{recipe_id}/ingredients/{ingredient_id}

- Removes the specified recipe from the specified ingredient's recipe list and vice versa. Both entities must have the same owner.
- Authorization header must contain the id_token received from the response to a POST /user/login request formatted as a bearer token.
- Response statuses
    - 204 upon success
    - 401 if JWT is missing or invalid
    - 403 if user of JWT does not match owner of either entity
    - 404 if either id is invalid