

## Introduction

Collected survey data on people's attitudes towards their own healthy diet across countries in Europe reveals that on average 50-60% are unhappy about the contents of their plates and would like to reduce their calorie intake [1-4]. In addition to governmental initiatives to foster a more balanced diet among citizens [5,6], many industries have been attempting to influence people's attitudes through various digital lifestyle assistants. Some of the more popular and sophisticated applications include MyNetDiary and MyFitnessPal and it has been shown that these apps can have an impact on eating behaviour when used carefully and avoiding obsessive use [7]. Major critique towards such applications comes from the fact that there is no 'one-size-fits-all' solution when talking about individual diets and very often, unfortunately, users opt for default application settings [8], which does not utilise full application capabilities. Therefore, we will try to focus on individual user dietary requirements and wishes in our solution.

We will use an API provided by the Edamam company established in 2011 [9] that has a goal of capturing the world's food knowledge and distilling it to help data users make informed choices at the store and in the kitchen [10]. Before we begin, we have to create a Recipe Search API Application ID and Application Key. First, create an account at <https://developer.edamam.com/login> by specifying Kristiania as the name of your organisation and selecting Recipe Search API (Developer). After you have registered, navigate to <https://developer.edamam.com/admin> and login using your email and password after which you should be able to see the following (Dashboard) page

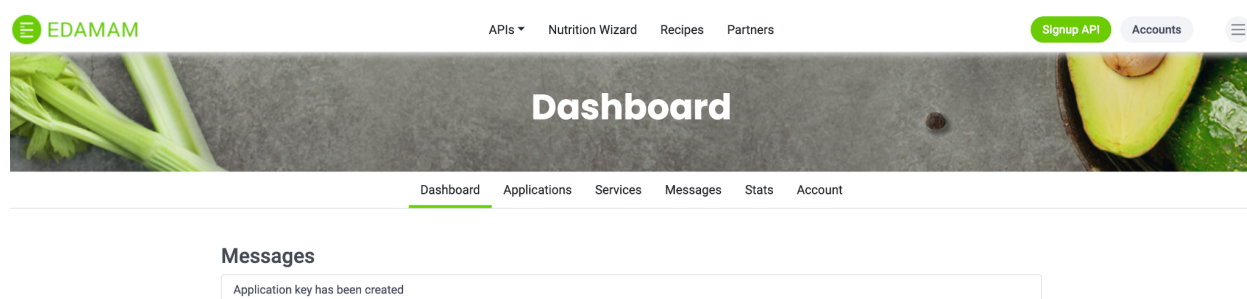


Figure 1: Dashboard page after logging into the Recipe Search API by Edamam

In Applications in Figure 1, you should be able to view your Recipe Search API, which will give you Application ID and Application Keys. You should make a note of them.

## API Response Structure

The Recipe Search API with access point <https://api.edamam.com/api/recipes/v2> has the structure specified in [Appendix A](#), where 'from, to, and count' indicate that this API uses pagination. This means that in order to load the next page with API results, it is required to follow URL under `_links` → `next` → `href`. Note that the next page URL already contains `app_id` and `app_key`.

We will only use <https://api.edamam.com/api/recipes/v2> end-point, where in addition to `app_id` and `app_key` we will have to specify 'q' parameter, which will tell the API what recipe we are searching for (e.g. Frog Eye Salad). Response 'hits' key contains a list of JSON objects with each object starting with a 'recipe' key. Default setting is to load 20 recipes per page with total recipes indicated under 'count' key.

Each recipe in the response contains:

- label or a name of the recipe
- four dish images in different sizes
- source of the recipe (e.g. Serious Eats, Martha Stewart, etc.)
- URL for the recipe (where recipe preparation instructions could be found). Note that Edamam does not provide recipe instructions through the API
- concise web-based info about the recipe under `shareAs` key
- yield that specifies how many portions are expected with this recipe using the amount of ingredients specified in the response. Note that you will have to divide, for example, calories by the number found under the `yield` key
- diet labels
- health labels
- cautionary labels
- short ingredient list and long ingredient list with images for each ingredient and other extra information about the ingredient
- amount of calories for all the portions. Note that you have to divide this number by the number found under `yield` key
- total weight
- cooking time
- cuisine type
- meal type
- dish type
- a list of nutrients
- percentage of daily norm for each nutrient
- chemical compounds for each recipe along with the daily norm

Example of a request looking for 'lapskaus' using working app\_key and app\_id:

[https://api.edamam.com/api/recipes/v2?app\\_key=d3338ea03fdf3626c7343299dd51c6b0&app\\_id=cd83fc5d&type=public&q=lapskaus](https://api.edamam.com/api/recipes/v2?app_key=d3338ea03fdf3626c7343299dd51c6b0&app_id=cd83fc5d&type=public&q=lapskaus) provides following information about a found similar dish

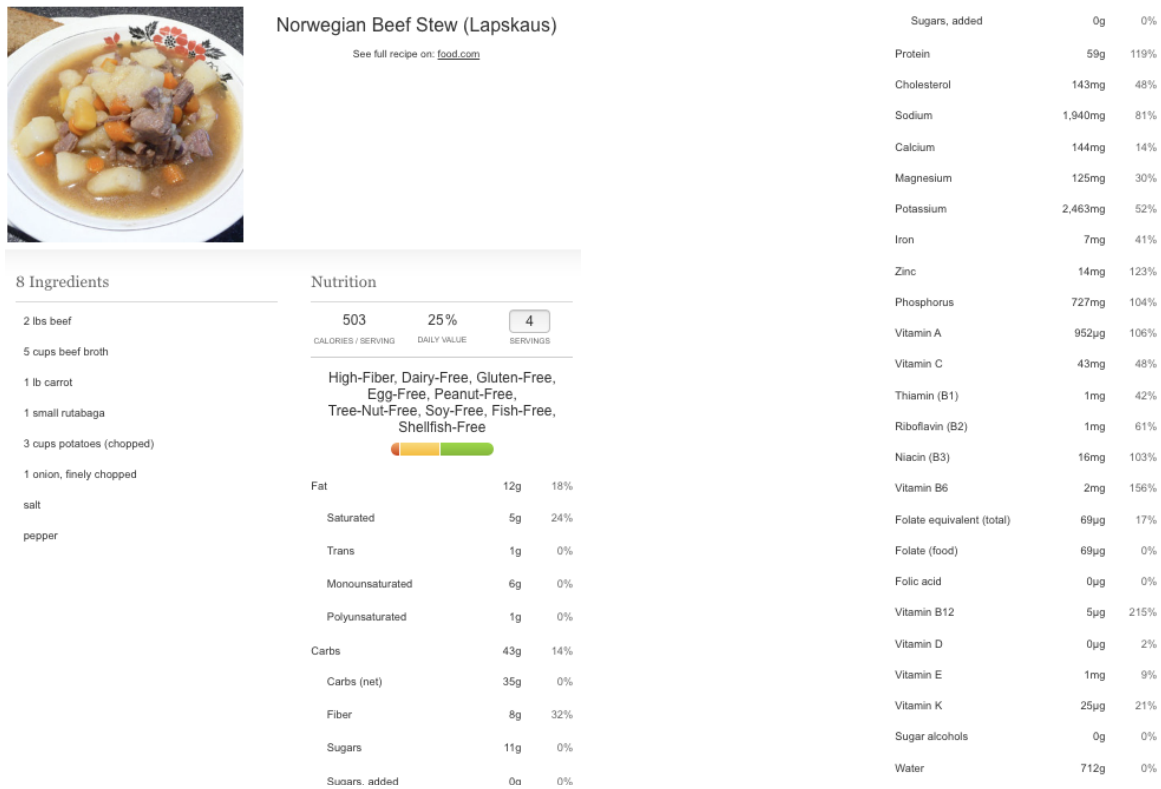


Figure 2: Example of data available for each recipe from Edamam's Recipe Search API (URL under shareAs key)

Documentation for the API can be found at <https://developer.edamam.com/edamam-docs-recipe-api>

You can test the API using web interface at <https://developer.edamam.com/recipe-demo>

## Suggested Screen Diagrams


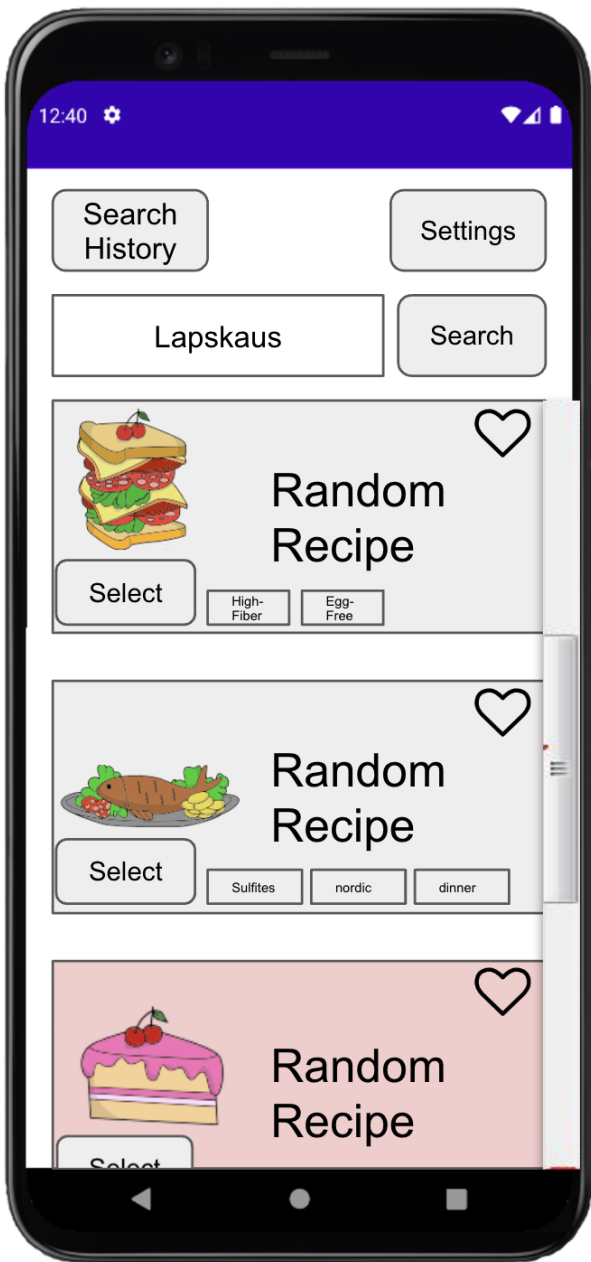
Table 1 below shows screen suggestions (students can come up with their own design). Rectangles without rounded corners with grey background indicate an information box (e.g. LinearLayour or TextView). Rectangles without rounded corners and without grey background indicate an EditText where a user should be able to write text. Rectangles with rounded corners indicate a button. ♥-icon indicates that a recipe has not been added to favourites, whereas the same icon with red background indicates that a recipe has been added to favourites. ▼-icon indicates some form of drop-down list. -icon indicates horizontal scrolling. The scrolling could also be vertical with the icon rotated 90°

Table 1: Suggested Screen Diagrams

UI Design Suggestion	Explanation
	<p>UI to the left shows an example of how the application's main screen could look like. In this example, an activity contains:</p> <ul style="list-style-type: none"> <li>• Three buttons for search history, settings, and search</li> <li>• EditText for specifying which recipe the user would like to search for</li> <li>• RecyclerView with each row containing an ImageView that displays recipe image, TextView with the name of the recipe, multiple TextViews for different labels (e.g. diet labels, health labels, cautionary labels, etc), and an ImageView which shows whether a recipe is added to favourites or not</li> </ul> <p>When the application is opened, random recipes are shown depending on the time of the day (e.g. Breakfast at 8AM and Dinner at 6PM).</p> <p>The user should be able to search for recipes and the list should be updated based on the search query obtained from the user (e.g. Lapskaus). Moreover, searched items should be appended to the search history list</p> <p>The user should be able to click on a specific recipe (e.g. clicking on text or image of the recipe) and navigate to a webpage with instructions for this recipe.</p> <p>Once the user has decided to prepare a specific recipe, they should click on "Select" this recipe button and the amount of calories for this recipe should be deducted from the daily calorie allowance specified in settings. If the recipe has more calories than the amount of calories left for the day, then the recipe's layout should be marked using, for example, red colour.</p> <p>User should be able to add a recipe to favourites and remove them by clicking on the same button / icon</p>



UI to the left shows what happens when a participant selects “Search History” from the main screen. This view contains

- TextView used as a header
- One RecyclerView that is split into breakfast, lunch, and dinner with the results from user searches performed in the main screen. RecyclerView rows have similar layout to the RecyclerView on the main screen (see above)

All the searched items should be listed in the search history, but the maximum number of search history items from settings should be obeyed. This means that only a certain number of latest searched items should be displayed in the search history.

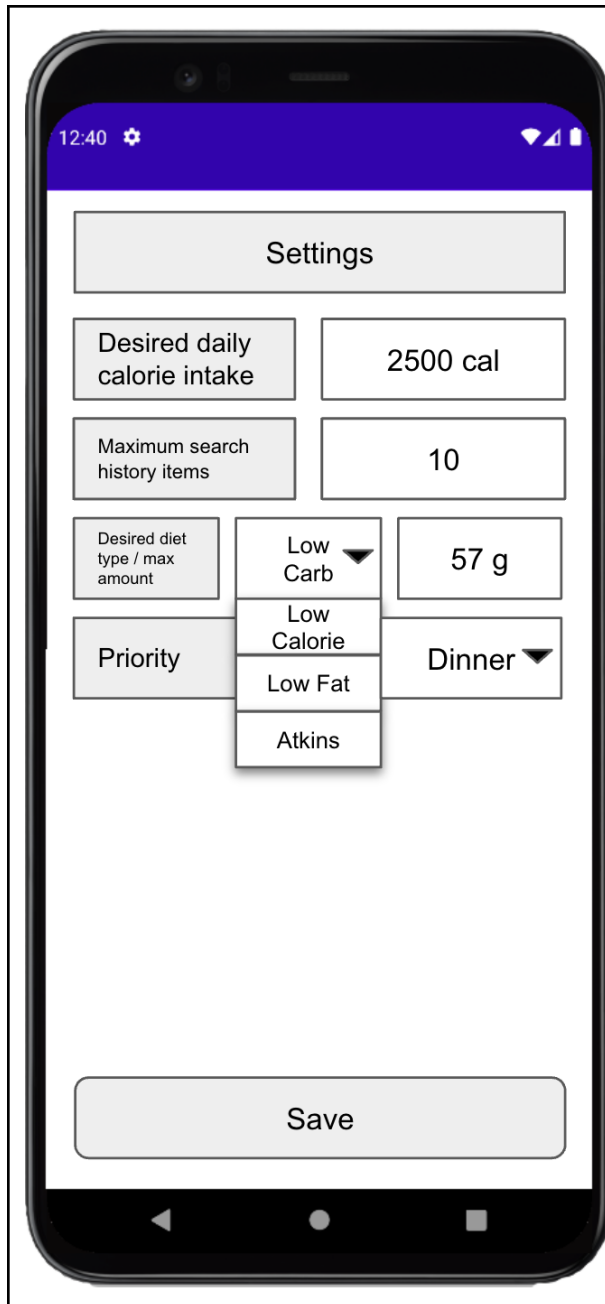
Just as in the main screen, the user should be able to favourite a recipe either from main screen or from search history screen

The user should be able to pick a recipe from the search history screen by clicking “Select” on any recipe.

Recipes that contain more calories than the remaining daily allowance should be marked, for example, with red colour

User should be able to add a recipe to favourites and remove them by clicking on the same button / icon

The user should be able to navigate to an external URL to read the recipe instructions when selecting either the image or the text of each recipe



UI to the left shows what happens when a participant selects the “Settings” button from the main screen. This screen contains

- TextView used as a header
- TextView on the left hand side for each user setting label
- EditText or a Spinner on the right hand side for each user setting value
- Save settings button

This screen should set global parameters for the application. For example, once the desired daily calorie intake is updated, it should also update which recipes are marked with red in both the main and the search history screens.

User should be able to set the maximum search history items number, which would be reflected in the search history screen

User should be able to set a desired diet (if wanted) and only recipes from that diet should be visible to the user when searching for recipes

The user should be able to set priorities for different meals of the day. For example, if the user wants to prioritise dinner, then more recipes should be shown for that meal of the day (e.g. dinner). Similarly, if the user prioritises dinner, then the user should be able to select a recipe that is more than the remaining daily calorie allowance

# Application Requirements

The list below specifies criteria when evaluating each submission. This means that not meeting these requirements will have a high negative impact on the final grade. [Table 2](#) gives more details into each of the requirements given below.

1. Activity implementation of each screen
2. At least one RecyclerView in the application
3. Store in the database (or similar) all search history items, favourites, settings, and daily calorie allowance
4. Use Kotlin as main programming language
5. Use threads with callbacks / coroutines for performing network and resource-heavy operations
6. Use at least one custom view class (e.g. RecyclerView row) that extends either a View class (including TextView, ImageView, Button, etc)
7. Use the Edamam API using your own api key and api id
8. The application you are developing should not crash or lose data when it is put to sleep if the user is switching between applications

## Optional

The list below specifies optional criteria when evaluating each submission. This means that not meeting these requirements will have no impact on the final grade. However, meeting these requirements will have a positive impact on the final grade.

1. Fragments for different screens instead of Activities
2. Use internal / external storage to cache images
3. Use at least one 3rd party library in your project that is not from androidx, google, coroutine, or JetBrains
4. Inspect and report network and resource usage statistics with caching and without caching
5. Lambdas and higher order functions

## Report

Each group is expected to submit a concise report of 3 pages (excluding images/diagrams and code), where you will present your project and explain your way of thinking and assumptions you made when interpreting this exam handout. For the technical part of the report, you will have to document the programming choices you made (e.g. libraries used, splitting functionality into classes, class and function names). In addition, you will have to provide a discussion for why certain UI components, programming patterns, frameworks have been used, their pros and



cons and what are the possible alternatives. This should provide an insight into how attentive you are when reading Android and Kotlin official documentation.

## Submission

Each student is expected to submit source code and a report in English in one .zip file. Students are expected to link their code and report. This means that the report should explicitly state lines of code along with discussions. Similarly, you should write comments in your code, where you will reference the report.

## Evaluation

Table 2 gives additional insight into what each submission will be evaluated on. The report is weighted the most as your discussion and reflection are paramount.

Table 2: List of sub-requirements

#	Sub-Requirement	Points (Total 100)
1	Show that you understand when it is appropriate to use the elvis operator by pointing to parts of code where it was used and discussing what edge cases it is meant to handle (e.g. data not available and why the data would not be available). Use as few !! as possible attempting to bypass null pointer exception checks	7
2	Implement your own RecyclerView adapter and discuss the view you used for each row in detail in the report. How fast is it when scrolling through the list. Is it user-friendly	17
3	Pass the data in an intent between Activities	5
4	Store data in some way in the application so it does not get removed from the memory if you restart the application. Describe and discuss the storage you have chosen in detail. What are the pros / cons / alternatives	17
5	Create callbacks (if needed) across the application to let the parent class / view know when something (e.g. data) is changing. Write how you handled asynchronous methods and what are the alternatives	8
6	Report with the following structure:  1. Introduction - introduce the application in your report. 2. Overall code structure (classes, interfaces, abstracts, xml, etc.) 3. Implementation, design decisions, and discussion	36

	a) what you did and what are the alternatives (pros/cons) b) what you did not do c) what you did not do (but wanted to do) and why d) what you would have done differently if you had the same task again 5. Conclusion 6. References	
7	How well you followed the guidelines and how attentive you were when reading this task description	10
8	Optional requirements (on top of 100% from sub-requirements 1-7)	8 points: Fragments 10 points: Cache images and report statistics of the benefit of caching 8 points: 3rd party libraries 7 points: Other than default colours / layouts / fonts 7 points: Lambdas and higher order functions

## References

- [1] <https://www.food.gov.uk/sites/default/files/media/document/healthy-and-sustainable-diets-consumer-poll.pdf>
- [2] <https://www.helsedirektoratet.no/rapporter/sektorrapport-om-folkehelse/sunne-valg/kosthold>
- [3] <https://www.ipsos.com/fr-fr/76-des-francais-considerent-que-nous-ne-sommes-pas-egaux-en-termes-dalimentation>
- [4] [https://www.aesan.gob.es/AECOSAN/web/seguridad\\_alimentaria/subdetalle/enalia.htm](https://www.aesan.gob.es/AECOSAN/web/seguridad_alimentaria/subdetalle/enalia.htm)
- [5] <https://www.local.gov.uk/parliament/briefings-and-responses/national-food-strategy-and-public-health-house-commons-15>
- [6] [https://www.regjeringen.no/contentassets/fab53cd681b247bfa8c03a3767c75e66/handlingsplan\\_kosthold\\_2017-2021.pdf](https://www.regjeringen.no/contentassets/fab53cd681b247bfa8c03a3767c75e66/handlingsplan_kosthold_2017-2021.pdf)
- [7] <https://dl.acm.org/doi/abs/10.1145/3308558.3313432>
- [8] <https://bmcmmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-020-01294-9>
- [9] <https://www.crunchbase.com/organization/edamam>
- [10] <https://www.edamam.com/about/company/>

## Appendix A - Response Structure from <https://api.edamam.com/api/recipes/v2>

```
{
  "from": 1,
  "to": 20,
  "count": 10000,
  "_links": {
    "next": {
      "href": "link_to_next_page_with_app_key_and_app_id",
      "title": "Next page"
    }
  },
  "hits": [
    {
      "recipe": {
        "uri": "uri",
        "label": "recipe_name",
        "image": "recipe_image",
        "images": {
          "THUMBNAIL": {
            "url": "recipe_image_thumbnail",
            "width": 100,
            "height": 100
          }, ...
        },
        "source": "Serious Eats",
        "url": "url_to_the_external_recipe",
        "shareAs": "share_page",
        "yield": 4.0,
        "dietLabels": ["Low-Carb"],
        "healthLabels": ["Mediterranean", "Dairy-Free"],
        "cautions": ["Sulfites"],
        "ingredientLines": ["1/2 cup olive oil", "5 cloves garlic, peeled"],
        "ingredients": [
          {
            "text": "1/2 cup olive oil",
            "quantity": 0.5,
            "measure": "cup",
            "food": "olive oil",
            "weight": 108.0,
            "foodCategory": "Oils",
            "foodId": "food_b1d1icua3iktrbqby0hiagafaz7",
            "image": "ingredient_image_url.jpg"
          },
          {
            "text": "5 cloves garlic, peeled",
            "quantity": 5.0,
            "measure": "clove",
            "food": "garlic",
            "weight": 15.0,

```

```

        "foodCategory": "vegetables",
        "foodId": "food_avtcmx6bgjv1jvay6s6stan8dnyp",
        "image": "ingredient_image_url.jpg"
    }, ...
],
"calories": 4228.043058200812,
"totalWeight": 2976.8664549004047,
"totalTime": 60.0,
"cuisineType": ["italian"],
"mealType": ["lunch/dinner"],
"dishType": ["main course"],
"totalNutrients": {
    "ENERC_KCAL": {
        "label": "Energy",
        "quantity": 4228.043058200812,
        "unit": "kcal"
    },
    "FAT": {
        "label": "Fat",
        "quantity": 274.4489059026023,
        "unit": "g"
    },
    ...
},
"totalDaily": {
    "ENERC_KCAL": {
        "label": "Energy",
        "quantity": 211.4021529100406,
        "unit": "%"
    },
    ...
},
"digest": [
    {
        "label": "Fat",
        "tag": "FAT",
        "schemaOrgTag": "fatContent",
        "total": 274.4489059026023,
        "hasRDI": true,
        "daily": 422.2290860040035,
        "unit": "g",
        "sub": [
            {
                "label": "Saturated",
                "tag": "FASAT",
                "schemaOrgTag": "saturatedFatContent",
                "total": 62.497618998656044,
                "hasRDI": true,
                "daily": 312.48809499328024,
                "unit": "g"
            },
            {
                "label": "Trans",
                "tag": "FATRN",
                "schemaOrgTag": "transFatContent",

```

```

        "total": 1.047163345382,
        "hasRDI": false,
        "daily": 0.0,
        "unit": "g"
      }, ...
    ]
  },
  {
    "label": "Carbs",
    "tag": "CHOCDF",
    "schemaOrgTag": "carbohydrateContent",
    "total": 175.96206666631727,
    "hasRDI": true,
    "daily": 58.65402222210575,
    "unit": "g",
    "sub": [
      {
        "label": "Carbs (net)",
        "tag": "CHOCDF.net",
        "schemaOrgTag": null,
        "total": 156.13025633549864,
        "hasRDI": false,
        "daily": 0.0,
        "unit": "g"
      },
      {
        "label": "Fiber",
        "tag": "FIBTG",
        "schemaOrgTag": "fiberContent",
        "total": 19.83181033081862,
        "hasRDI": true,
        "daily": 79.32724132327448,
        "unit": "g"
      }, ...
    ]
  }, ...
]
},
"_links": {
  "self": {
    "title": "Self",
    "href": "json_url"
  }
}
}
]
}

```