Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores

Nutr.io - Multi-platform application for diabetics' nutritional choices

Beta release

Authors:

Pedro Pires	Miguel Luís	David Albuquerque
42206	43504	43566
A42206@alunos.isel.pt	A43504@alunos.isel.pt	A43566@alunos.isel.pt

Tutor:
Fernando Miguel Gamboa de Carvalho
mcarvalho@cc.isel.pt

June 15, 2020

Instituto Superior de Engenharia de Lisboa

Nutr.io - Multi-platform application for diabetics' nutritional choices

42206 - Pedro Miguel Sequeira Pires

Signature: ______________________

43504 - Miguel Filipe Paiva Luís

Signature: ______________________

43566 - David Alexandre Sousa Gomes Albuquerque

Signature: ______________________

Tutor: Fernando Miguel Gamboa de Carvalho

Signature: ______________________

Abstract

The idea that every field of study can be digitalized in order to ease monotonous tasks is continuously growing in the modern world. Our project aims to tackle the field of Type 1 diabetes, given its growing prevalence in the world.

One of those monotonous tasks is the count and measurement of carbohydrates in meals used to administer the correspondent amount of insulin, along with their blood levels, to maintain a healthy lifestyle. A task that heavily relies on having access to food databases and realize of how many portions a meal has - usually by using a digital balance or doing estimations.

Eating in restaurants is the perfect example that showcases a gap in this field, that our project, Nutr.io, aims to fill. Most nutritional applications do not provide data for restaurants' meals, such as MyFitnessPal, nor does the user bring his digital balance from home - resulting in a faulty carbohydrate count and therefore the administration of an incorrect insulin dose.

The main goal of this project is to design a system that offers a way to facilitate difficult carbohydrate measurement situations, like in restaurants. To that end, a system that stores meals' nutritional information will be developed, where users can use and calibrate its data with their feedback.

Contents

Chapter 1

Introduction

This document is related to the project's beta release, mentioning all progress made up to this date.

The report will also state the issues encountered during this time period, mentioning the decisions the group made to solve them. This might also include changes in the initial plan, that the group found relevant for the project's progress efficiency.

The diagrams and schemas developed for this project are shown when approaching the respective topic, however there is an appendix which contains additional information about the project, having references pointing to it when necessary.

Chapter 2

Project development

2.1 Roadmap

According to the proposed plan, the group managed to accomplish a more complete Android aplication that can make requests to the server and use its information to display lists, detailed views and calculate user's insulin values as store local cache such as user profiles and search history.

The HTTP server also had great progress, having now most of its planned features working and properly tested. In order to make this possible, the database suffered some changes that will be detailed later in this report.

However, as it will be detailed in the next section, the group faced some issues that caused delays and made the web browser application development not possible in this delivery, mainly due to API withdrawals and data models changes.

2.2 Issues encountered and updates

This section describes the issues found and the decisions made to overcome them during development.

2.2.1 Relational database

The group had to redesign the database's models multiple times again, due to the previously mentioned API withdrawals and other encountered incoherences.

2.2.2 Food API's

2.2.3 HTTP server

After the progress report, the group continued developing the HTTP server and completing the endpoints that were previously lacking. However it was concluded that most APIs that were being used by the server didn't provide accurate nutritional information about meals and ingredients, which would void the main objective of our platform.

As a result, the group had to drop some meal APIs and introduce generic and hardcoded meal information inside the database as a replace.

This information will be progressively tuned along with the users inputs when they, for example, associate a meal with a restaurant, leaving their values about meals' quantities and portions.

2.2.4 Android client

The Android application's development progressed normally but it had to be put on hold sometimes, because of HTTP server's endpoints' completion, in which the application depends strongly. A major dto and model restructure had also to be made inside the mobile application in order to meet with the current HTTP responses.

2.3 Roadmap updates

Given that the previously mentioned issues, the group agrees that every mandatory requirement made in the initial draft and retified in the previous progress report can still be implemented until the project's final release, such as the web client apllication.

However the group will have to remove some optional features so the main ones can be delivered whole.

Chapter 3

Results

This chapter shows what has been achieved and developed to this date.

Overall, every initial mandatory goal stated in the project's proposal was met, with two key exceptions: a missing browser-based client and a missing authentication system. This was due to an ambitious number of mandatory requirements and encountered difficulties mentioned previously in the progress report.

However, the first optional objective was fulfilled. Meaning that a community based contribution system which allows users to create restaurants, meals and to vote on said submissions is present in the Beta phase.

3.1 Relational database

As a result of multiples redesigns, here is the database's conceptual model.

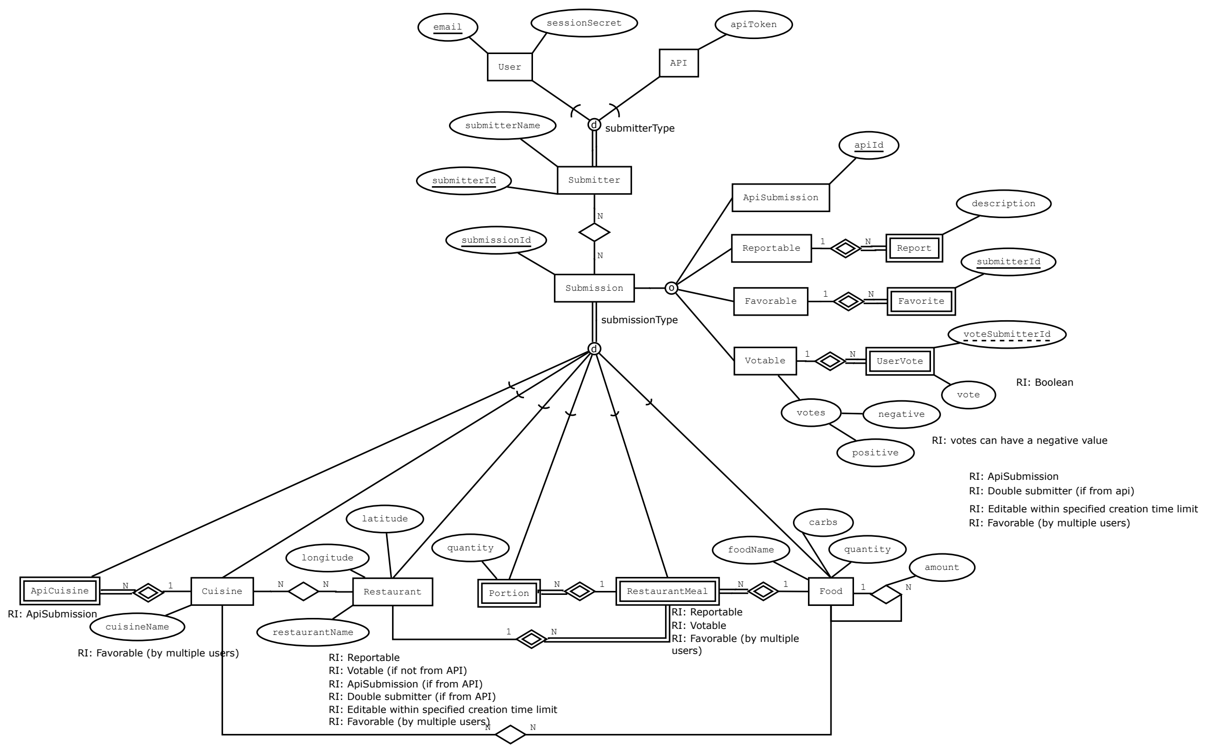


Figure 3.1: Database conceptual model

The database's relational model is present inside this report's appendix [Appendix B - Database relational model].

In the relational model there are tables which are not specified in the conceptual model. These are a product from associations between entities which will simplify queries' complexity.

Now the submission can fall into 4 categories: ApiSubmission, Reportable, Favorable and Votable, in order to disguish between submissions that are from the user or from APIs and to separate which ones can be reportable, favorable and votable by the user.

The cuisine entity has now an associated entity called ApiCuisine, to save cuisine information provided by the Here API.

Meals and ingredients were now condensed into one entity called food - now each meal can have meals inside it that can also be considered ingredients in other contexts.

Therefore each meal possesses nutritional information, which is essential to the user especially to the insulin calculations. That information is composed by 'carbs' - meal's carbohydrates; and quantity - meal's quantity.

3.2 HTTP server

Given that some mandatory goals were not met, the server endpoints appendix was changed [Appendix C - Endpoints' table] in order to be consistent with said goals.

Seeing as user authentication was not implemented, but some endpoints require a submitter, a placeholder form of authentication was created which relies on query parameter '?submitter=submitterId' being given.

Every 'POST' or 'PUT' endpoint requires authentication.

3.3 Geolocation

Given how all clients rely on obtaining nearby restaurants, there was a need to implement a geolocation function in the project's design.

Initial research showcased two possible solutions: Haversine distances and cartesian distances, where the latter returns a highly imprecise distances. As such, Haversine was selected.

The next step was to choose which system filters nearby restaurants: database or HTTP server. After some discussion, the group decided that database was the best option for two reasons:

- Given the large amount of existing restaurants, sending such data from the database to the HTTP server so that it could filter it would occupy too much memory;
- PostgreSQL already supplies extensions that add support for location queries, namely PostGIS.

3.4 Android application

As mentioned in the progress report, the group managed to implement in the mobile application a fragment that displays a map with a list of restaurants nearby the user.

The user can also search for restaurants, meals and cuisines providing the associated name or identifier.

The core feature of the apllication was also finished during this time period - the insulin calculator. This feature calculates how many insulin doses should be injected in order to maintain the blood glucose levels stabilized according to the user's planned glucose objective for that period of the day.

The blood glucose objective is set inside the user's profile, by creating multiples insulin profiles, each one has a limited time period to give the user freedom to map its own insulin routine throughout the day.

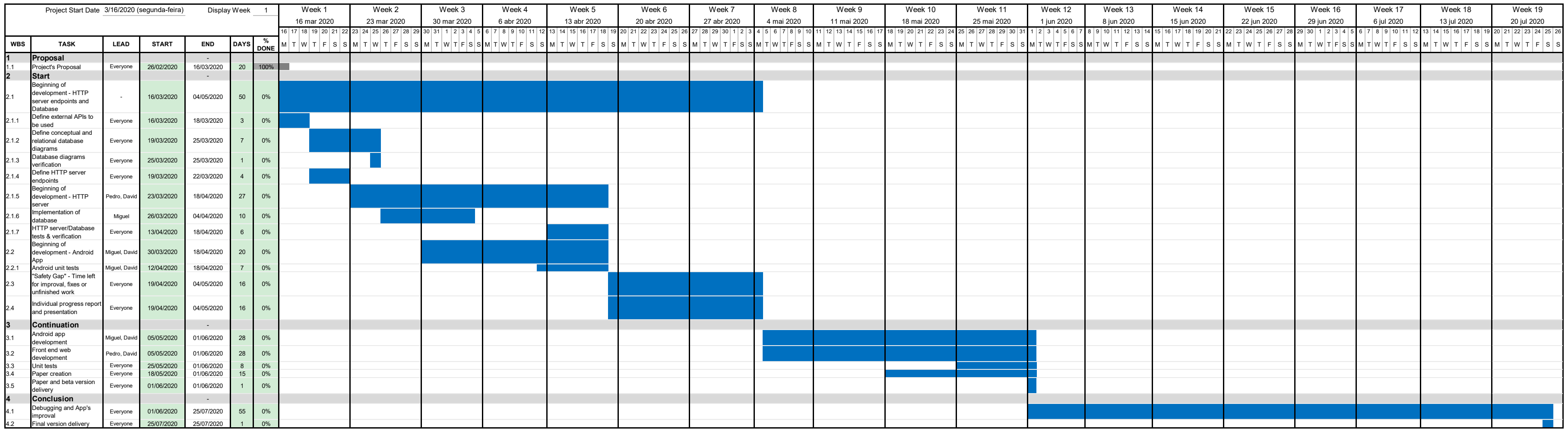
This is due to the fact that user's insulin sensitivity factor varies along the day, so the user has the ability to specify its own values in order to the calculator

Chapter 4

Appendices

This chapter displays all the appendices referenced in this report.

Appendix A - Initial plan



Project Start Date 3/16/2020 (segunda-feira) Display Week 1

WBS	TASK	LEAD	START	END	DAYS	% DONE
1	**Proposal**			-		
1.1	Project's Proposal	Everyone	26/02/2020	16/03/2020	20	100%
2	**Start**			-		
2.1	Beginning of development - HTTP server endpoints and Database	-	16/03/2020	04/05/2020	50	0%
2.1.1	Define external APIs to be used	Everyone	16/03/2020	18/03/2020	3	0%
2.1.2	Define conceptual and relational database diagrams	Everyone	19/03/2020	25/03/2020	7	0%
2.1.3	Database diagrams verification	Everyone	25/03/2020	25/03/2020	1	0%
2.1.4	Define HTTP server endpoints	Everyone	19/03/2020	22/03/2020	4	0%
2.1.5	Beginning of development - HTTP server	Pedro, David	23/03/2020	18/04/2020	27	0%
2.1.6	Implementation of database	Miguel	26/03/2020	04/04/2020	10	0%
2.1.7	HTTP server/Database tests & verification	Everyone	13/04/2020	18/04/2020	6	0%
2.2	Beginning of development - Android App	Miguel, David	30/03/2020	18/04/2020	20	0%
2.2.1	Android unit tests	Miguel, David	12/04/2020	18/04/2020	7	0%
2.3	"Safety Gap" - Time left for improval, fixes or unfinished work	Everyone	19/04/2020	04/05/2020	16	0%
2.4	Individual progress report and presentation	Everyone	19/04/2020	04/05/2020	16	0%
3	**Continuation**			-		
3.1	Android app development	Miguel, David	05/05/2020	01/06/2020	28	0%
3.2	Front end web development	Pedro, David	05/05/2020	01/06/2020	28	0%
3.3	Unit tests	Everyone	25/05/2020	01/06/2020	8	0%
3.4	Paper creation	Everyone	18/05/2020	01/06/2020	15	0%
3.5	Paper and beta version delivery	Everyone	01/06/2020	01/06/2020	1	0%
4	**Conclusion**			-		
4.1	Debugging and App's improval	Everyone	01/06/2020	25/07/2020	55	0%
4.2	Final version delivery	Everyone	25/07/2020	25/07/2020	1	0%

Figure 1: Initial plan accorded in the project's proposal

Appendix B - Database relational model

- **Submitter**
  - Attributes: submitterId, submitterName, submitterType
  - Primary Key(s): submitterId
  - Foreign Key(s): -
  - Not null: submitterName, submitterType
- **User**
  - Attributes: *submitterId*, email, sessionSecret, creationDate
  - Primary Key(s): *submitterId*, email
  - Foreign Key(s): *submitterId* references Submitter(submitterId)
  - Not null: sessionSecret
- **API**
  - Attributes: *submitterId*, apiToken
  - Primary Key(s): *submitterId*
  - Foreign Key(s): *submitterId* references Submitter(submitterId)
  - Not null: apiToken
- **Submission**
  - Attributes: submissionId, submissionType, submissionDate
  - Primary Key(s): submissionId
  - Not null: submissionType
- **ApiSubmission**
  - Attributes: *submissionId*, *apiId*
  - Primary Key(s): *submissionId*, *apiId*
  - Foreign Key(s): *submissionId* references Submission(submissionId)
  - Not null: submissionType
- **SubmissionSubmitter**
  - Attributes: *submissionId*, *submitterId*
  - Primary Key(s): *submissionId*, *submitterId*
  - Foreign Key(s):
    - *submissionId* references Submission(submissionId)
    - *submitterId* references Submitter(submitterId)
  - Not null: *submitterId*
- **SubmissionContract**
  - Attributes: submissionId, submissionContract
  - Primary Key(s): submissionId, submissionContract

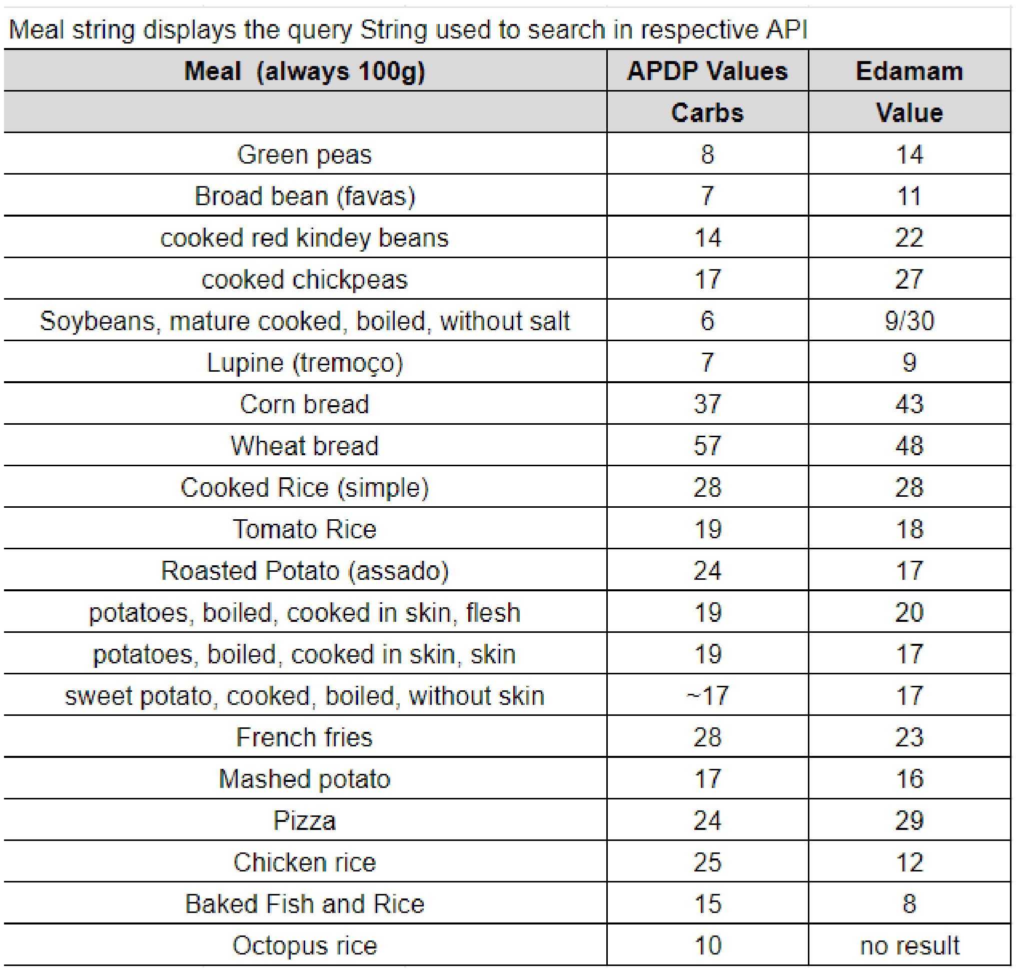
- **Report**
  - Attributes: *submissionId*, *submitterId*, description
  - Primary Key(s): *submissionId*, *submitterId*
  - Foreign Key(s):
  -
    - *submissionId* references Submission(submissionId)
    - submitterId references Submitter(submitterId)
  - Not null: description
- **Votes**
  - Attributes: *submissionId*, positiveCount, negativeCount
  - Primary Key(s): *submissionId*
  - Foreign Key(s): *submissionId* references Submission(submissionId)
  - Not null: submissionType
- **UserVote**
  - Attributes: *submissionId*, voteSubmitterId, vote
  - Primary Key(s): *submissionId*, voteSubmitterId
  - Foreign Key(s):
    - *submissionId* references Submission(submissionId)
    - voteSubmitterId references Submitter(submitterId)
- **Restaurant**
  - Attributes: *submissionId*, restaurantName, latitude, longitude
  - Primary Key(s): *submissionId*
  - Foreign Key(s): *submissionId* references Submission(submissionId)
  - Not null: restaurantName
- **Cuisine**
  - Attributes: submissionId, cuisineName
  - Primary Key(s): cuisineName
- **ApiCuisine**
  - Attributes: *submissionId*, *cuisineSubmissionId*
  - Primary Key(s): cuisineName
  - Foreign Key(s):
    - *submissionId* references Submission(submissionId)
    - *cuisineSubmissionId* references Cuisine(submissionId)
- **Meal**
  - Attributes: *submissionId*, mealName, carbs, quantity, unit
  - Primary Key(s): *submissionId*
  - Foreign Key(s): *submissionId* references Submission(submissionId)
  - Not null: mealName

- **RestaurantMeal**
  - Attributes: submissionId, *restaurantSubmissionId*, *mealSubmissionId*
  - Primary Key(s): submissionId, *restaurantSubmissionId*, *mealSubmissionId*
  - Foreign Key(s):
    - *restaurantSubmissionId* references Restaurant(submissionId)
    - *mealSubmissionId* references Meal(submissionId)
- **Favorite**
  - Attributes: *submissionId*, submitterId
  - Primary Key(s): *submissionId*, submitterId
  - Foreign Key(s): *submissionId* references Submission(submissionId)
- **Portion**
  - Attributes: *submissionId*, restaurantMealSubmissionId, quantity
  - Primary Key(s): *submissionId*
  - Foreign Key(s): *submissionId* references Submission(submissionId)
  - Not null: quantity
- **MealIngredient**
  - Attributes: *restaurantSubmissionId*, *ingredientSubmissionId*, quantity
  - Primary Key(s): *restaurantSubmissionId*, *ingredientSubmissionId*
  - Foreign Key(s):
    - *mealSubmissionId* references Meal(submissionId)
    - *ingredientSubmissionId* references Ingredient(submissionId)
- **RestaurantCuisine**
  - Attributes: *restaurantSubmissionId*, *cuisineName*
  - Primary Key(s): *restaurantSubmissionId*, *cuisineName*
  - Foreign Key(s):
    - *restaurantSubmissionId* references Restaurant(submissionId)
    - *cuisineName* references Cuisine(cuisineName)
- **MealCuisine**
  - Attributes: *mealSubmissionId*, *cuisineName*
  - Primary Key(s): *mealSubmissionId*, *cuisineName*
  - Foreign Key(s):
    - *mealSubmissionId* references Meal(submissionId)
    - *cuisineName* references Cuisine(cuisineName)

Appendix C - Endpoints' table

Method	Path	Query String	Body parameters	Description
GET	\restaurant	float latitude, float longitude, optional String name, optional int radius, optional String name		Search for restaurants and their cuisines, based on location or named search
GET	\restaurant\:restaurantId			Obtain specific restaurant's full information by given restaurantId
GET	\restaurant\:restaurantId\meal			Obtain all suggest and user inserted restaurant meals for given restaurant
GET	\restaurant\:restaurantId\meal\:mealId	int skip, int count		Obtain specific restaurant meal for given restaurantId and mealId
GET	\cuisines	optional int skip, optional int limit		List possible cuisines
GET	\ingredients	optional int skip, optional int limit		Get all possible ingredients
GET	\meal	optional string[] mealTypes, optional int skip, optional int count, optional string[] cuisines		Get all suggested meals
GET	\meal\:mealId			Obtain specific meal's full information by given mealId
POST	\restaurant		RestaurantInput	Create a new restaurant around given geolocation
POST	\meal		MealInput	Create a user meal with at least one ingredient
POST	\restaurant\:restaurantId\meal\:mealId		PortionInput	Insert a new portion for given restaurant meal
PUT	\restaurant\:restaurantId\vote		VoteInput	Add or update your vote on a user restaurant
PUT	\restaurant\:restaurantId\:mealId		VoteInput	Add or update your vote on a restaurant meal created by an user
PUT	\restaurant\:restaurantId\meal		RestaurantMealInput	Creates a restaurant meal from given user meal
DELETE	\restaurant\:restaurantId			Delete user created restaurant
DELETE	\restaurant\:restaurantId\vote			Delete user's vote on an user's restaurant
DELETE	\restaurant\:restaurantId\meal\:mealId			Delete user's portion submission for given restaurant meal
DELETE	\restaurant\:restaurantId\meal\:mealId\portion			Delete user's restaurant's meal portion
DELETE	\restaurant\:restaurantId\meal\:mealId\vote			Delete user's restaurant's meal vote
DELETE	\meal\:mealId			Delete an user created meal, along with any associations with a restaurant the meal might have

Appendix D - API nutritional accuracy sheet



Meal string displays the query String used to search in respective API

Meal (always 100g)	APDP Values	Edamam
	Carbs	Value
Green peas	8	14
Broad bean (favas)	7	11
cooked red kindey beans	14	22
cooked chickpeas	17	27
Soybeans, mature cooked, boiled, without salt	6	9/30
Lupine (tremoço)	7	9
Corn bread	37	43
Wheat bread	57	48
Cooked Rice (simple)	28	28
Tomato Rice	19	18
Roasted Potato (assado)	24	17
potatoes, boiled, cooked in skin, flesh	19	20
potatoes, boiled, cooked in skin, skin	19	17
sweet potato, cooked, boiled, without skin	~17	17
French fries	28	23
Mashed potato	17	16
Pizza	24	29
Chicken rice	25	12
Baked Fish and Rice	15	8
Octopus rice	10	no result

Figure 2: API nutritional accuracy sheet