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

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

Progress Report

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

May 4, 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 intended to give an update on the project's progress made to this date.

Its main goal is to report that the project is progressing according to the initial plan, previously stated in the project's proposal.

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 accorded 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 where is disposed 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 fulfill almost all the objectives that were planned to be accomplished by this time - a relational database, a working HTTP server and a prototype Android aplication that can make requests to the server and use its information to display lists and detailed views.

However, as it will be detailed in the next section, the group faced some issues that caused slight drawbacks, such as the relational database conception, which had to be redesigned multiple times due to the project's requirements.

In this report's appendix there are disposed two project schedules: the initial one, also present in the project's proposal, and the actual one, which has the actual progress made to this date with the corrects starting and ending dates of each concluded milestone.

2.2 Issues encountered and updates

This section describes the issues found and the decisions made to overcome them, during the first project modules' development.

2.2.1 Relational database

The group had to redesign the database's conceptual model and relational model multiple times. One of the main issues that supported the main redesign was that the database only supported submissions from the users, while it had to support submissions both from users and APIs.

The other redesigns were related to the database normalization: it was normalized until the third normal form and had to be renormalized one more time because of the mentioned main redesign.

Because of this, the group invested a considerable amount of time on the database's conception, which was not considered in the initial plan. This invested time ended up colliding partially with other time periods that were reserved for other modules.

2.2.2 HTTP server

As mentioned in the project's proposal risks, the HTTP server is very similar to the DAW's project, which is an optional course inside the LEIC programme. Thus some improvements were made and errors were fixed because of the lectioned classes and the first project's delivery in DAW, which helped the group complete the tasks implied by the server's development.

However the server is lacking some functionalities as a result of the time invested in database conception, where the HTTP server depends strongly, causing some drawback to its development.

2.2.3 Android application

As a result of the time invested developing the database and the strong bond between the mobile application and the HTTP server and its completion, the Android app suffered a slight drawback, where only the essential fragments and requests work.

Althrough it was initially planned that the mobile application was just a prototype by this date, as it is now, the group could not implement more planned features because of the time invested in the other modules.

2.3 Group decisions

Given the previous issues, the group has the conditions to comply with the project's initial plan, with some minor changes.

As stated in the last period of the Android application's subsection, the group strongly agrees that the HTTP server has to be finished as soon as possible, to provide stability to the onward modules and their implementations, being considered a core module in this project.

Subsequently the group will have to invest a few more days in the server's development in order to fulfill the project's requirements.

Alongside with these fulfillments, the group will proceed to add more features to the mobile application, while starting the web application's development, which is compatible with the project's initial plan.

Chapter 3

Results

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

As previously written in the project's roadmap, this project has already a working relational database and a HTTP server, developed with postgreSQL and with kotlin and spring mvc, respectively. The group also managed to achieve, by this date as planned, its first working plaform client - the android mobile application, developed with kotlin. This last one, it is still a prototype, so it lacks some features that will be completed during the next weeks.

3.1 Relational database

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

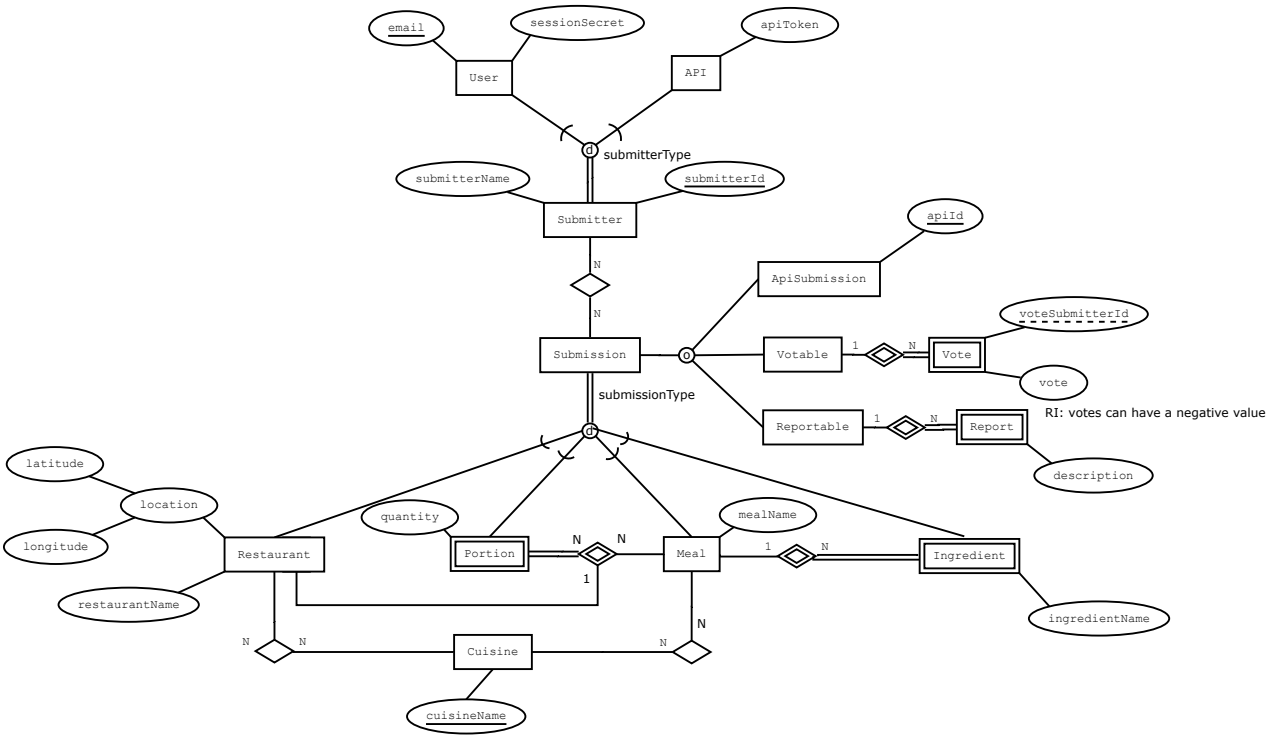

Figure 3.1: Database conceptual model

The database's relational model is present inside this report's appendix.

In the relational model there are tables which are not specified in the conceptual model, those tables are a product from associations between entities, which will, not only facilitate the access queries' construction, as provide useful data to the HTTP server's endpoints.

The best feature of this redesigned model is that the database can filter the user submissions from the API submissions. This is very useful, because when the user tries to associate a meal, that does not exist in the database, to a restaurant, that is already registered in the database, the system will have search for that meal in an external API. When the meal is found, it will be inserted in the database as an API submission, because the data came from an external API.

However if another user inserts this previously inserted meal to another registered restaurant, it will insert as an user's submission, because the data already existed in the database and the user just built the association between the specific meal and the specific restaurant.

This process, which resembles how memory pointers work, provides a high scalability and, cooperatively with the database normalization, lowers the memory usage.

3.2 HTTP server

As previously written in this report, the HTTP server was developed in Kotlin and Spring MVC.

Before starting the server's development, the group had to first define its endpoints and discuss what the client was allowed to request or receive from each one of them. The endpoint table that defines the endpoints and their tasks can be found in the appendix.

After defining the endpoints...

- hypermedia type / response type (siren and application/json)
- errors : application/problem+json

3.3 Android application

Being a prototype, the mobile application has only the core features to work and interact with the HTTP server.

3.3.1 Layout

The group chose to create an application that offers a simple layout to the user, providing accessible menus and a menu drawer, which implies the use of fragments instead of activities.

Each fragment, that has volatile information in it (live data), has a associated view model, that preserves the data while the application is running. As expected, the Android app also has static fragments - fragments which do not receive live data, those do not have any view model associated.

Many fragment that use live data have recycler view lists, which means that for each list there's the need to have an adapter and a view holder, called by the adapter, to bind the live data

to the respective layout elements.

So each list that uses a diferent model class, creates its own adapter and viewholder for its specific model extending from an abstract adapter and view holder.

3.3.2 HTTP requests

The volley framework was implemented to allow the application to make asynchronous server requests.

As most of the server endpoints send responses in JSON format, the application is using the Jackson library to deserialize and serialize JSON.

Chapter 4

Appendix

This section has disposed addicional information of the project, such as project's schedules and other references mentioned in this report.

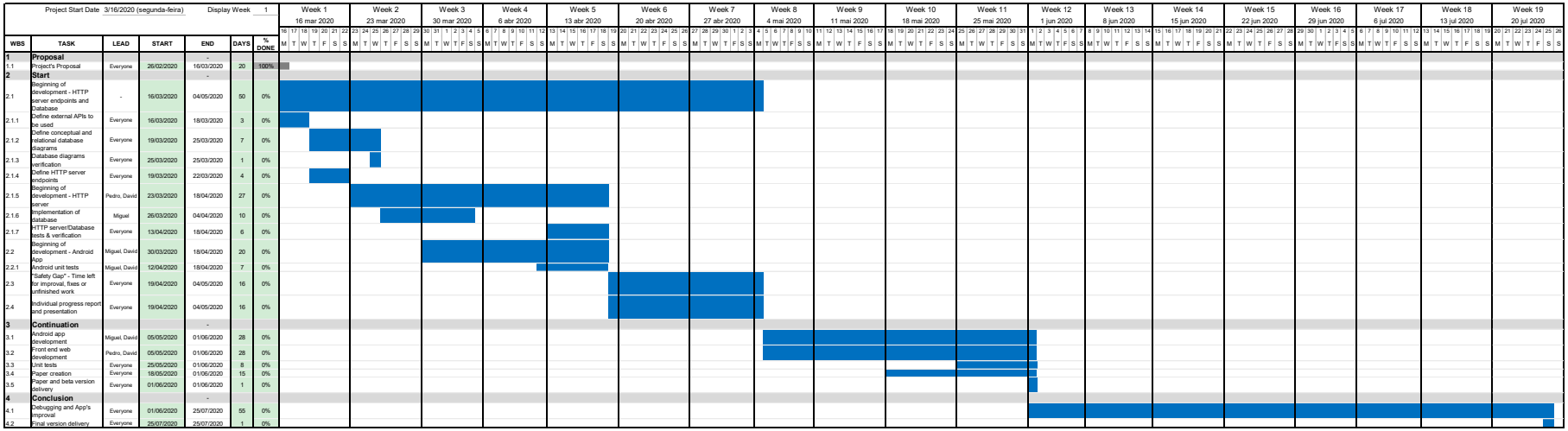

Figure 4.1: Initial plan accorded in the project's proposal

Nutr.io Project Schedule

Instituto Superior de Engenharia de Lisboa

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, Database and Android App prototype	-	16/03/2020	04/05/2020	50	100%
2.1.1	Define external APIs to be used	Everyone	16/03/2020	18/03/2020	3	100%
2.1.2	Define conceptual and relational database diagrams	Everyone	19/03/2020	23/04/2020	36	100%
2.1.3	Database diagrams verification	Everyone	25/03/2020	23/04/2020	30	100%
2.1.4	Define HTTP server endpoints	Everyone	19/03/2020	22/03/2020	4	100%
2.1.5	Beginning of development - HTTP server	Pedro, David	23/03/2020	04/05/2020	43	100%
2.1.6	Implementation of database	Miguel	26/03/2020	26/04/2020	32	100%
2.1.7	HTTP server/Database tests & verification	Everyone	13/04/2020	04/05/2020	22	100%
2.2	Beginning of development - Android App	Miguel, David	30/03/2020	04/05/2020	36	100%
2.2.1	Android unit tests	Miguel, David	12/04/2020	18/04/2020	7	100%
2.3	"Safety Gap" - Time left for improval, fixes or unfinished work	Everyone	19/04/2020	04/05/2020	16	100%
2.4	Individual progress report and presentation	Everyone	19/04/2020	04/05/2020	16	100%
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%

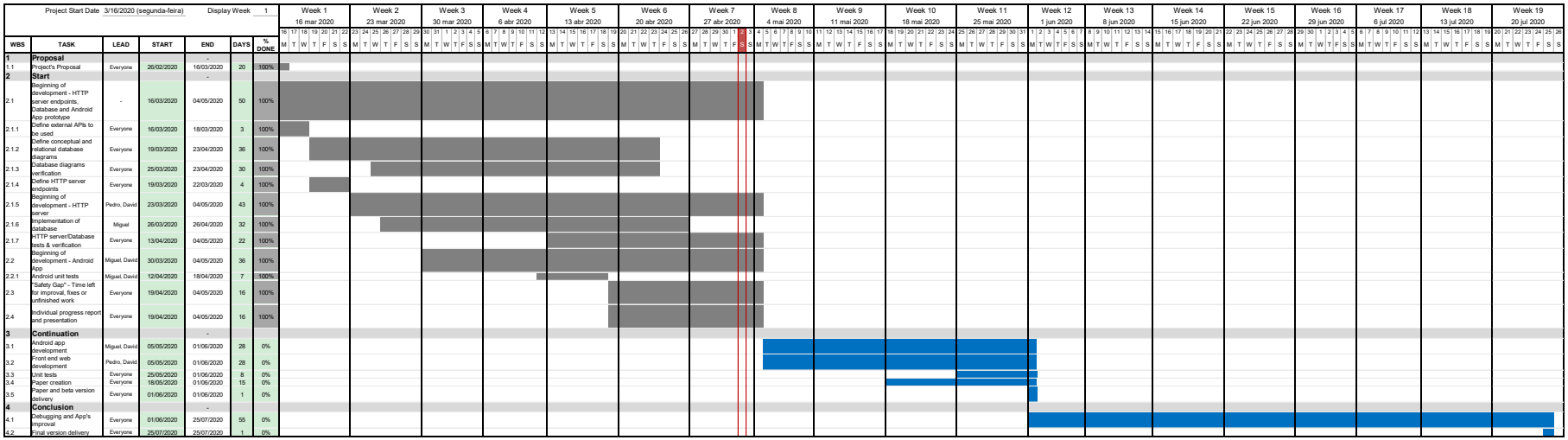

Figure 4.2: Actual plan with the progress updated as of today

Method	Path	Query String	Body parameters	Description
GET	restaurant	int lat, int lon, int skip, int count, String name		Search for restaurants and their cuisines, based on location or named search
GET	restaurant\:id			Restaurant's name and cuisines
GET	restaurant\:id\meal	int skip, int count		Restaurant's meals with votes
GET	cuisines	int skip, int count		List possible cuisines
GET	ingredients	int skip, int count		List all possible ingredients and it's id
GET	meal\:id			Meal's portions and ingredients
POST	restaurant		String name, int lat, int lon, String[] cuisines	Create a new restaurant
POST	restaurant\:id\meal		String name, Object portion, int[] ingredientIds	Create a new restaurant meal, allowing for multiple ingredients (optional), first portion submission (optional)
POST	restaurant\:id\meal\:mealId\portion		Object portion	Creates a restaurant's meal portion
POST	restaurant\:id\meal\:mealId\report		String reason	Create a report on a restaurant's meal
POST	restaurant\:id\meal\:mealId\vote		boolean isPositive	Create a vote on a restaurant's meal
POST	restaurant\:id\report		String reason	Create a report on a restaurant
POST	restaurant\:id\vote		boolean isPositive	Create a vote on a restaurant
PUT	restaurant\:id\meal\:mealId\portion		Object portion	Updates a restaurant's meal portion
PUT	restaurant\:id\meal\:mealId\vote		boolean isPositive	Update a vote on a restaurant's meal
DELETE	restaurant\:id			Delete user created restaurant, if it wasn't used by the community yet
DELETE	restaurant\:id\meal\:id			Delete user created restaurant meal, if it wasn't used by the community yet
DELETE	restaurant\:id\meal\:mealId\portion			Delete user's restaurant's meal portion
DELETE	restaurant\:id\meal\:mealId\vote			Delete user's restaurant meal vote
DELETE	restaurant\:id\vote			Delete user's restaurant

Database relational model

- **Submitter**
  - Attributes: submitterId, submitterName, submitterType
  - Primary Key(s): submitterId
  - Foreign Key(s): -
  - Not null: submitterName, submitterType
- **User**
  - Attributes: *submitterId*, email, sessionSecret
  - 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
  - Primary Key(s): submissionId
  - Not null: submissionType
- **ApiSubmission**
  - Attributes: *submissionId*, apiId, submissionType
  - Primary Key(s): *submissionId*, apiId
  - Foreign Key(s): *submissionId* references Submission(submissionId)
  - Not null: submissionType
- **SubmissionSubmitter**
  - Attributes: *submissionId*, *submitterId*, submissionDate
  - 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: *submitterId*, *submissionId*, description
  - Primary Key(s): *submitterId*, *submissionId*
  - Foreign Key(s):
    - *submissionId* references Submission(submissionId)
    - *submitterId* references Submitter(submitterId)
  - Not null: description
- **Vote**
  - Attributes: *submissionId*, *voteSubmitterId*, vote
  - Primary Key(s): *voteSubmitterId*, *submissionId*
  - Foreign Key(s):
    - *submissionId* references Submission(submissionId)
    - *voteSubmitterId* references Submitter(submitterId)
  - Not null: vote
- **Restaurant**
  - Attributes: *submissionId*, restaurantName, latitude, longitude
  - Primary Key(s): *submissionId*
  - Foreign Key(s): *submissionId* references Submission(submissionId)
  - Not null: restaurantName
- **Cuisine**
  - Attributes: cuisineName
  - Primary Key(s): cuisineName
- **Meal**
  - Attributes: *submissionId*, mealName
  - Primary Key(s): *submissionId*
  - Foreign Key(s): *submissionId* references Submission(submissionId)
  - Not null: mealName
- **Portion**
  - Attributes: *submissionId*, quantity
  - Primary Key(s): *submissionId*
  - Foreign Key(s): *submissionId* references Submission(submissionId)
  - Not null: quantity
- **Ingredient**
  - Attributes: *submissionId*, ingredientName
  - Primary Key(s): *submissionId*
  - Foreign Key(s): *submissionId* references Submission(submissionId)

  - Not null: ingredientName

- **MealIngredient**
  - Attributes: *mealSubmissionId*, *ingredientSubmissionId*
  - Primary Key(s): *mealSubmissionId*, *ingredientSubmissionId*
  - Foreign Key(s):
    - *mealSubmissionId* references Meal(submissionId)
    - *ingredientSubmissionId* references Ingredient(submissionId)

- **RestaurantMealPortion**
  - Attributes: *mealSubmissionId*, *portionSubmissionId*, *restaurantSubmissionId*
  - Primary Key(s): *mealSubmissionId*, *portionSubmissionId*
  - Foreign Key(s):
    - *mealSubmissionId* references Meal(submissionId)
    - *portionSubmissionId* references Portion(submissionId)
    - *restaurantSubmissionId* references Restaurant(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)