# Meal planner Web Application: generating meal plan option based on user’s nutritional constraints

Luiz Fernando Puttow Southier

Dennis Shasha

# Description of the problem

We describe the project and implementation of a diet web application that has two main features.

First, the user can input some demographic information (like weight, height and age) and the web application, based on this information, will generate recommended ranges of intake for each different nutrient that can satisfied the user daily nutritional needs.

Second, the user can classify the database’s foods in three categories, the foods that he/she likes, the foods that he/she dislikes and the foods that he/she does not like and does not dislike. Based on the food likability the web application will use linear programming to generate a meal plan that satisfies the user daily nutritional needs prioritizing the foods that he/she likes.

# Description of the data

## Data source

The data used to create the food database is the data provided by the Agricultural Research Service of the United States Department of Agriculture. The data consist in a slq file with description of 8,618 different foods arranged in 26 different food groups. Each food entry has specified the amount of each one of the 150 different nutrients in the database. The data can be accessed online [1] as well as the documentation describing the tables and theirs relations [2].

## Database used

The data obtained from USDA was used to create the database for the web application. The information used is the 8,618 different foods arranged in the 26 different food groups and the amount of 5 basic nutrients that each of these foods contain.



Figure - Database Schema

|  |  |
| --- | --- |
| Table | Structure |
| user | username - varchar(30) password - vachar(30) firstName - varchar(60) lastName - varchar(60) email - varchar(120) dateOfBirth - date dateOfRegister - date gender - varchar(10) height - double weight - double exerciseLevel - varchar(50) age - int(11)  *User account information (username, password, first name, last name, email, date of register) and demographic information (date of birth, gender, height, weight, exercise level, age)* |
| fd\_group | FdGrp\_CD - varchar(4) FdGrp\_Desc - vachar(60)  *Food group information (group code and group description)* |
| nutr\_def | Nutr\_No - varchar(3) Units - varchar(7) NutrDesc - varchar(60)  *Nutrients information (nutrient number, unit and description)* |
| food\_des | NDB\_No - int(5) FdGrp\_Cd - varchar(4) Long\_Desc - varchar(200) 203 - double 204 - double 205 - double 255 - double 291 - double  *Food information (food group, description, amount of protein, amount of fat, amount of carbohydrate, amount of water, amount of fiber)* |
| adm | username - varchar(30) password - vachar(30) firstName - varchar(60) lastName - varchar(60) email - varchar(120) dateOfRegister - date  *Administrator information (adm username, password, first name, last name, email and date of register)* |
| likes | username - varchar(30) NDB\_No - varchar(126) like\_id - int(11)  *Like information (username, food liked, and like id)* |
| dislikes | username - varchar(30) NDB\_No - varchar(126) dislike\_id - int(11)  *Dislike information (username, food disliked, and dislike id)* |
| meal\_plan | mealPlanId - int(11) username - varchar(30) dateOfCreation - date  *Meal Plan main information(meal plan id, username, date of creation)* |
| mealPlan\_food | mealPlanId - int(11) NDB\_No - int(5) quantity - int(11) food\_mealPlanID - int(11)  *Food in a specific meal plan (meal plan id, food id, quantity, food versus meal plan id)* |

# Description of the implementation.

The web application was developed in PHP language using Mysql to create and manage the database. It has been host by an Apache server and it uses some JavaScript functionalities. The Linear Programming package used was developed in C++.

## Diet planner websites review

The first step was to search the web for websites that already have similar goals. After some basic research was possible, verify the different aspects that involve diet planner websites. The results related to this research can be seen in the following tables:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Link | | | | | | | | | | Price per month | | | | |
| **Healthy Hand** | | | | | | | [**http://www.healthyhand.com/**](http://www.healthyhand.com/) | | **$ 0.00 (basic) $ 28.00 (premium)** | | | | | | |
| • users can specify their **traits,** **goals, diet and** **schedule • ordering** with the restaurant • meals **delivery**  Free version has fewer options of goals and traits, 3 kinds of diets and it cannot be customized.  Premium version has the Fully Customized Diet. Users can specify allergies & medical conditions and can upload their 23andMe genetics for gene-based nutrition.  Step 1: Users specify traits (Activity Levels, Allergens & Dislikes, Medical Conditions, Prescriptions & Supplements ...), goals (Mind, Body, Spirit, Medical ...) and meal schedule (Breakfast, Lunch, Dinner, Home, Office ...)  Step 2: Each day the users will receive an email with meal options, chosen according to their traits, goals, schedule & the latest research. They will select their meal. The website will place their order with the restaurant that will deliver the meal | | | | | | | | | | | | | | | |
| **Choose My Plate. Gov** | | | | | | | | [**http://www.choosemyplate.gov/**](http://www.choosemyplate.gov/myplate/index.aspx) | | | | | | **$ 0.00** | |
| • users can specify **physical information** • **mom** (pregnant or breastfeeding ) and **preschooler** versions • **tips** about how to eat the 'correctly'  No premium version  Step 1: Users complete a survey about their physical information (weight, height, age).  Step 2: Website calculates the quantity of Grains, Vegetables, Dairy, Fruits and Protein Foods that the user has to eat | | | | | | | | | | | | | | | |
| **Webmd** | | [**http://www.webmd.com/diet/food-fitness-planner/**](http://www.webmd.com/diet/food-fitness-planner/) | | | | | | | | | | | **$ 0.00** | | |
| • users can specify **physical information** • user can specify **healthy information**  No premium version  Step 1: Users complete a survey about their physical information (weight, height, age).  Step 2: Website calculates the quantity of calories that the user has to consume.  Step 3: Users can add more information about the dishes that they are consuming and the website calculates the calories | | | | | | | | | | | | | | | |
| **Eat This Much** | | | | | [**http://www.eatthismuch.com/**](http://www.eatthismuch.com/) | | | | | | | | | | **$ 0.00** |
| • user can specify the **amount of calories** that he/she wants to consume • user can **alter** the suggest dishes • protein/carbs/fat **statistics**  No premium version  Step 1: User specify the amount of calories that he/she wants to consume per day.  Step 2: Website generates a daily menu with options of dishes that contains the calorie amount specified  Step 3: User can choose new dishes and can verify the amount of protein, carbs and fat in those dishes | | | | | | | | | | | | | | | |
| **Swole .Me** | | | [**http://swole.me/**](http://swole.me/) | | | | | | | **$ 0.00** | | | | | |
| • user can specify the **amount of calories** that he/she wants to consume • user can **alter** the suggest dishes • protein/carbs/fat **statistics**  No premium version  Step 1: User specify the amount of calories that he/she wants to consume per day.  Step 2: Website generates a daily menu with options of dishes that contains the calorie amount specified  Step 3: User can choose new dishes and can verify the amount of protein, carbs and fat in those dishes | | | | | | | | | | | | | | | |
| **Eating Well** | | | | [**http://www.eatingwell.com/nutrition\_health/weight\_loss\_diet\_plans/diet\_meal\_plans**](http://www.eatingwell.com/nutrition_health/weight_loss_diet_plans/diet_meal_plans) | | | | | | | | | | | **$0.00** |
| • user can choose between Weight Loss, Diabetes, Vegetarian, Heart Healthy, Gluten Free or Healthy Aging meal plans. • recipe archive  No premium version  Step 1: Users specify the type of meal plan (Weight Loss, Diabetes, Vegetarian, Heart Healthy, Gluten Free or Healthy Aging) and physical information  Step 2: Website provides a Meal Plan with recipes suggestions | | | | | | | | | | | | | | | |
| **South Beach Diet** | | | | | | [**http://www.southbeachdiet.com/diet/recipes**](http://www.southbeachdiet.com/diet/recipes) | | | | | | **$ 20.00** | | | |
| • grocery-shopping **planner** • a guide to staying on the program when **dining out** • a recipe **archive** • users can specify their **weight loss goals**  No free version  Step 1: Users complete a survey about their physical information (weight, height, and age), free daily time.  Step 2: The website will suggest guides to help the user with the diet. | | | | | | | | | | | | | | | |
| **Diet.com** | | [**http://www.diet.com/**](http://www.diet.com/) | | | | | | | | | | **$ 40.00** | | | |
| • users can specify **physical information** • access to the **staff doctor** • live weekly chats with diet and fitness **experts** • message **boards** • **buddy system** to pair users with a fellow dieter for more motivation  No free version  Step 1: Users complete a survey about their physical information (weight, height, and age), free daily time.  Step 2: Website provides an eating style and personality profile | | | | | | | | | | | | | | | |

With this research it was possible to identify the diet planner websites that already exist, their main features and how the user interaction worked.

## Database creation

Using the data from USDA, it was possible to create the database for the web application. The selection of specifics tables from the USDA file and creation of the database for the web application was made using the phpmyadmin program and the result can be seen at Description of the Data section.

## Web application structure

The following table shows the files present in the web application and their main function.

|  |  |
| --- | --- |
| File | Description |
| register.php | Displays the form where the user can create an account and the login form. |
| loginADM.php | Displays the login form to administration area. |
| index.php | Displays the personal information form and the nutrient range results. |
| foods.php | Displays the different foods arranged in categories and allows the user to select the likes and dislikes. |
| mealPlan.php | Calculates and displays the Meal Plan. |
| LPScript.php | Creates the Linear Programming script and send the script to the Linear Programming package. |
| functions.php | Main functions used in the web application. |
| indexADM.php | Displays a form where the administrator can insert a new food in the database. |
| updateFood.php | Administrator can update a food in the database. |
| registerADM.php | Displays the form where the administrator can create an administrator account. |
| login.php | Connects the web application with the MySQL Database. |
| logout.php | Disconnects from the web application. |
| Format.css | Style Sheet file. |
| glpk.exe | Implements the Linear Programming Package. |

## register.php

This file has two main objectives: it allows the user to create a new account in the web application and it allows the user to login into the web application. The first form (registration form) has the inputs First Name, Last Name, Username, Password, Password confirmation and E-mail. The second form (login form) has the inputs Login and Password.

This page displays four buttons:

**Login**: gets the values in the login form (Login and Password) and check them with the JavaScript function validateForm2. If the function is true, it checks the values with the database. If the database check returns true, it creates a session and redirects the user to index.php; if it returns false, it displays an error message.

**Submit**: gets the values in the registration form (First Name, Last Name, Username, Password, Password confirmation and E-mail.) and checks them with the JavaScript function validateForm. If the function is true, it checks the username against the database. If the username does not exist, it creates a record in the table user with the values. In case of false, it displays an error message

**Reset**: resets the registration form.

**LoginADM**: redirects to loginADM.php.

This page has two functions:

**validateForm2**: verifies if the Login and Password inputs in the login form are non- NULL.

***Parameters*:** none

***Return*:** true, if login and password are non-null, false otherwise.

**validateForm**: verifies if the inputs in the registration form are non-NULL, if the Username and Password length is from six to 20 characters and if the Password and Password confirmation are the same.

***Parameters*:** none

***Return*:** true, if the inputs in the registration form are non-NULL, if the Username and Password length is from six to 20 characters and if the Password and Password confirmation are the same, false otherwise.

## loginADM.php

This file allows the administrator to login to the web application. The login administrator form has the inputs Login and Password.

This page displays one button:

**Login**: gets the values in the login form (Login and Password) and check them with the JavaScript function validateForm2. If the function returns true, it checks the values with the database. If the database check returns true, it creates a session and redirect the user to indexADM.php, if it returns false, it displays an error message.

This page has one function:

**validateForm2**: verifies if the Login and Password inputs in the login administrator form are non-NULL.

***Parameters*:** none

***Return*:** true, if login and password are non-null, false otherwise.

## index.php

This file allows the user to insert/update personal information and it calculates and displays the nutrient ranges based on that information. The personal information form has the inputs Gender, Date of Birth, Height, Weight and Exercise Level.

This page displays four buttons:

**Submit**: gets the values in the personal information form and check them with the JavaScript function *validateForm*. If the function is true, it update the values in the database. With these values, it calls the PHP function *getresults* that calculates and display the nutrient ranges and personal information.

**Next Page**: goes to food.php.

**Reset**: resets the personal information form.

**Logout**: redirects to logout.php.

This page has two functions:

**validateForm**: it verifies if the inputs in the registration form are different from NULL, if the Height and Weight are valid numbers and it calculates the user’s age based on the Date of Birth.

***Parameters*:** none

***Return*:** true, if the inputs in the registration form are different from NULL, if the Height and Weight are valid, false otherwise.

**getresults**: using the parameters (Gender, Age, Height, Weight and Exercise Level) this function will calculate the nutrient ranges. Its process is detailed in the functions.php section.

**createTable:** creates the html table to display the user's nutrients ranges. Its process is detailed in the functions.php section.

## food.php

This file allows the user to visualize the different options of food present in the database, to arrange them by categories, and to filter them using some specified string. With these results, the user can specify if he/she likes or dislikes the food. The user has an option to like/dislike all the results at once. The check boxes like and dislike cannot both be selected for a single food.

This page displays five buttons:

**Save**: gets the values in the results of food that the user likes or dislikes and saves them in the database.

**Filter**: filters the results using a specified string.

**Next Page**: goes to mealPlan.php.

**Previous Page**: goes to index.php.

**Logout**: redirects to logout.php.

This page has four functions:

**selectLike**: (JavaScript function) selects all the like checkboxes at once and deselects all the dislikes checkboxes.

***Parameters*:** none

***Return*:** none

**selectDislike**: (JavaScript function) selects all the dislike checkboxes at once and deselects all the likes checkboxes.

***Parameters*:** none

***Return*:** none

**optionLike**: (JavaScript function) is activated when a like check box is selected, and deselects the dislike checkbox of that food.

***Parameters*:** none

***Return*:** none

**optionDislike**: (JavaScript function) is activated when a dislike check box is selected, and deselects the like checkbox of that food

***Parameters*:** none

***Return*:** none

## mealPlan.php

First, the file will create a vector with the foods that the user likes using the database information (food\_likes vector). Second, the file will create a vector with the foods that the user likes and the foods that the user does not dislike (food\_likes\_neutral vector). Then the file will create a vector with all the foods using the database information (food\_all vector). In this file is defined the variable portion that will determine the maximum number of portions that a food item can be divided in.

The first vector and the variable portion are send to the PHP function lp that will calculate the meal plan using linear programming and will return a set of foods that satisfies the user’s nutrient ranges. This set of foods will be send to the PHP function check\_solution that will guarantee that it is a valid solution. This process is repeated for three times or until the check\_solution returns false. In each of this times, the first food of the set of solution is take off from the food\_likes vector before it is sent to the lp function so that we can get a different set of foods in each iteration.

If the first vector, food\_likes vector, does not generate a valid result, the process described above will be repeated with the second vector, food\_likes\_neutral vector. If even this vector does not generate a valid result, the process will be repeated with the third vector that contain all the foods. The results are displayed to the user.

This page displays seven buttons:

**Previous Page**: goes to food.php.

**Logout**: redirects to logout.php.

This page has four functions:

**lp**: gets a set of foods and a portion number and it tries to generate, using linear programming, a set of food that will satisfy the user’s nutrient ranges. Its process is detailed in the LPScript.php section.

**checkSolution**: gets the set of food returned by lp function and checks if this set of foods will satisfy the user’s nutrient ranges. Its process is detailed in the functions.php section.

## LPScript.php

This file defines the PHP function lp that creates and executes the linear programming script.

This page has two functions:

**lp**: receives as parameters a set of foods (*array\_food\_final*) and a portion number. For each of the foods it gets from the database the amount of each one of the nutrients and divides this amount for the portion number. The results are placed in an array for each of the nutrients (*water, carbohydrate, fiber, protein* and *fat*). Using the information from the database in the function *getresults*, it gets the user nutrient ranges and places them in the *definitions* array. Using the *definitions* array, the arrays of nutrients, and the *array\_food\_final*, the *lp* function creates a txt file with all this information that will be necessary to the linear programming package.

For example, let’s say that that the function *lp* receives a set of foods with two foods. It will generate the following vector for the problem:

*array\_food\_final*

|  |  |
| --- | --- |
| Food 1 | Food 2 |

*water*

|  |  |
| --- | --- |
| water1 | water2 |

*carbohydrate*

|  |  |
| --- | --- |
| carb1 | carb2 |

*fiber*

|  |  |
| --- | --- |
| fiber1 | fiber2 |

*protein*

|  |  |
| --- | --- |
| prot1 | prot2 |

*fat*

|  |  |
| --- | --- |
| fat1 | fat2 |

*definitions*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| maxWater | minWater | maxCarb | MinCarb | minFiber | maxProt | minProt | maxFat | minFat |

If the number of portions of food 1 is *x* and the number of portions of food 2 is *y*, then we have the following linear programing constraints:

Based on this system the following table is created in a txt file, using PHP. The PHP calls then the linear programming executable file that will read the txt file and return the values *x* and *y*.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

In a general case, the *lp* function will return a value for each one of the food that it received in its arguments. In the case of no solution, these values will be the values that most nearly obey the constraints above.

***Parameters*:** set of foods and a portion number

***Return*:** output of the Linear Programming package

**getresults**: using the parameters (Gender, Age, Height, Weight and Exercise Level) this function will calculate the nutrient ranges. Its process is detailed in the functions.php section.

This file also calls the glpk.exe file, the program that executes the linear programming calculation. More information in the glpk.exe section.

## functions.php

This file contain the functions *checkSolution* and *getresults* as the sub functions described as follows. All these functions use data and formulas from the Dietary Reference Intakes [1].

**getBMI:** calculates Body Mass Index using user’s weight and height.

***Parameters*:** user’sWeight and Height

***Return*:** user’s BMI

**getCalorie:** calculates the daily-recommended amount of calories using user’s exercise level, age, gender, height and weight.

***Parameters*:** user’s exercise level, age, gender, height and weight

***Return*:** user’s daily-recommended amount of calories

**getPA:** calculates the user’s Physical Activity Coefficient using user’s exercise level, age and gender.

***Parameters*:** user’s exercise level, age and gender

***Return*:** user’s Physical Activity Coefficient

**getWaterLower:** calculates the daily lower recommended amount of water, using user’s age and gender.

***Parameters*:** user’s age and gender

***Return*:** user’s daily lower recommended amount of water

**getWaterUpper:** calculates the daily upper recommended amount of water, using user’s age and gender.

***Parameters*:** user’s age and gender

***Return*:** user’s daily upper recommended amount of water

**getCarbohydrateLower:** calculates the daily lower recommended amount of carbohydrate using user’s daily-recommended amount of calories.

***Parameters*:** user’s daily recommended amount of calories

***Return*:** user’s daily lower recommended amount of carbohydrate

**getCarbohydrateUpper:** calculates the daily upper recommended amount of carbohydrate using user’s daily-recommended amount of calories.

***Parameters*:** user’s daily recommended amount of calories

***Return*:** user’s daily upper recommended amount of carbohydrate

**getFiberLower:** calculates the daily lower recommended amount of fiber, using user’s age and gender.

***Parameters*:** user’s age and gender

***Return*:** user’s daily lower recommended amount of fiber

**getProteinLower:** calculates the daily lower recommended amount of protein, using user’s daily-recommended amount of calories and age.

***Parameters*:** user’s daily-recommended amount of calories and age

***Return*:** user’s daily lower recommended amount of protein

**getProteinUpper:** calculates the daily upper recommended amount of protein, using user’s daily-recommended amount of calories and age.

***Parameters*:** user’s daily-recommended amount of calories and age

***Return*:** user’s daily upper recommended amount of protein

**getFatLower:** calculates the daily lower recommended amount of fat, using user’s daily-recommended amount of calories and age.

***Parameters*:** user’s daily-recommended amount of calories and age

***Return*:** user’s daily lower recommended amount of fat

**getFatUpper:** calculates the daily upper recommended amount of fat, using user’s daily recommended amount of calories and age.

***Parameters*:** user’s daily-recommended amount of calories and age

***Return*:** user’s daily upper recommended amount of fat

**getresults:** calculates the user’s nutrient ranges using the functions above. These functions were obtained based on Dietary Reference Intakes [3].

***Parameters*:** user’sAge, Weight, Height, Gender and Exercise Level

***Return*:** anarray with the user’s BMI, calories, water lower amount, water upper amount, carbohydrate lower amount, carbohydrate upper amount, fiber lower amount, protein lower amount, protein upper amount, fat lower amount, fat upper amount

**checkSolution:** gets the set of food returned by the *lp* function and checks if this set of foods will satisfy the user’s nutrient ranges, adding up all the amounts of a specific nutrient and comparing with the lower value and upper value.

***Parameters*:** quantity recommended for each different food, set of foods that are part of the solution, number of portion that a food can be split into and direction of linear programming

***Return*:** true if the set is a solution, false otherwise

**showingSolution:** gets the set of food returned by the *lp* function and shows the solution on the screen

***Parameters*:** quantity recommended for each different food, set of foods that are part of the solution, number of portion that a food can be split into and direction of linear programming

***Return*:** none

**createTable:** creates the html table to display the user's nutrients ranges

***Parameters*:** none

***Return*:** html table that displays the user's nutrients ranges

## indexADM.php

Displays a form where the administrator can insert a new food in the database. The insert food form has the inputs Food Description, Food Group, Protein, Fat, Water, Fiber and Carbohydrate.

This page displays five buttons:

**Submit**: gets the values in the insert food form and checks them with the JavaScript function validateForm. If the function is true, it insert the values in the database.

**Register New ADM**: goes to registerADM.php.

**Update Food Information**: goes to updateFood.php.

**Reset**: resets the insert food form.

**Logout**: redirects to logout.php.

This page has one functions:

**validateForm**: verifies if the inputs in the registration form are different than NULL and if the nutrient amounts are valid numbers.

***Parameters*:** none

***Return*:** true, if the inputs in the registration form are different than NULL and if the nutrient amounts are valid numbers, false otherwise.

## updateFood.php

Displays a form where the administrator can update a food in the database. The *update food form* has the inputs Food Description, Food Group, Protein, Fat, Water, Fiber and Carbohydrate.

This page displays four buttons:

**Save**: gets the values in the insert food form and checks them with the JavaScript function validateForm. If the function is true, it insert the values in the database.

**Register New ADM**: goes to registerADM.php.

**Register New Food**: goes to indexADM.php.

**Logout**: redirects to logout.php.

## registerADM.php

This file is for the administrator to create a new administrator account in the web application. The *register adm form* has the inputs First Name, Last Name, Username, Password, Password confirmation and E-mail.

This page displays three buttons:

**Submit**: gets the values in the registration form (First Name, Last Name, Username, Password, Password confirmation and E-mail.) and check them with the JavaScript function validateForm. If the function is true, it checks the username against the database. If the username does not exist, it creates a record in the table adm with the values. In case of false, it displays an error message

**Reset**: resets the form.

**Register New Food**: goes to indexADM.php.

**Logout**: redirects to logout.php.

This page has two functions:

**validateForm**: verifies if the inputs in the registration form are non-NULL, if the Username and Password length is from 6 to 20 characters and if the Password and Password confirmation are the same.

***Parameters*:** none

***Return*:** true, if the inputs in the registration form are non-NULL, if the Username and Password length is from six to 20 characters and if the Password and Password confirmation are the same, false otherwise.

## login.php

## Connects the web application with the MySQL Database. It is inserted in the files that need connection with the database.

## logout.php

## Disconnects from the web application and finishes the user’s section.

## Format.css

## Style Sheet file.

## glpk.exe

Implements the Linear Programming Package. This file is a compiled version of the GNU Linear Programming Kit (GLPK) [4], intended for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems. It is a set of routines written in ANSI C and organized in the form of a callable library. The php file LPscript creates a txt file that will be read by the Linear Programming package. The txt file is created by several values separated by spaces. The following table describes these values

|  |  |  |
| --- | --- | --- |
| **Index** | **Value** | **Description** |
| 1 | “glpk.exe” | Name of the executable file |
| 2 | “MIN” | Direction of the linear programming |
| 3 | X | Number of values in the txt file:  X = (N+1)\*C + 5+ N |
| 4 | N | Number of foods to try to generate the meal plan |
| 5 | C | Number of constraints for each food |
| 6 to 6+N-1 | “1” | Constant 1 to define the Linear Programming Problem |
| 6+N to X | Ci | Coefficients of the variables of the constraints |

This file is read by the glpk.exe file and the result of the calculation is returned to the php file LPScript. To create the glpk file, we have had to install the glpk package and compile the final file. The instructions and code to do it can be found on [4]

# References

[1] US Department of Agriculture - Agricultural Research Service. “USDA National Nutrient Database for Standard Reference." 08/01/2014 http://www.ars.usda.gov/Services/docs.htm?docid=8964

[2] US Department of Agriculture - Agricultural Research Service. "USDA National Nutrient Database for Standard Reference - Documentation and User Guide" 08/01/2014 http://www.ars.usda.gov/SP2UserFiles/Place/80400525/Data/SR27/sr27\_doc.pdf

[3] Jennifer J. Otten, Jennifer Pitzi Hellwig, Linda D. Meyers. 2006. *Dietary Reference Intake: the essential guide to nutrient requirements.* Washington, DC: National Academies Press.

[4] GNU project. “GLPK (GNU Linear Programming Kit).” 08/01/2014. https://www.gnu.org/software/glpk/