КУРСОВ ПРОЕКТ

по дисциплина „Основи на обектно ориентираното програмиране“

ТЕМА: Програма за генериране на хранителен режим

Студент:Никола Иванов

Специалност:ИКТ

фак.№:12220124

гр.Варна

2022г.

Съдържание

[1.Увод 3](#_Toc103791679)

[2.Аналитична част 4](#_Toc103791680)

[2.1. База от данни 4](#_Toc103791681)

[2.2. Промени в базата от данни 5](#_Toc103791682)

[2.3 Работа на приложението 6](#_Toc103791683)

[3. Описание на кода 9](#_Toc103791684)

[3.2 Описание на обектите 9](#_Toc103791685)

[3.2 Промени в кода и поправяне на грешки 11](#_Toc103791686)

[4. Заключение 17](#_Toc103791687)

[5. Литература 18](#_Toc103791688)

**1.Увод**

Правилното хранене е много важен аспект от нашия живот. То може да подобри нашето настроение, да повиши енергията ни, да намали стреса, имат главна роля в заздравяването на костите и раните ни, и много други. Тази тема бива пренебрегната в днешно време и по тази причина беше решено да се направи курсовия проект за нея.

Правилното хранене е тясно свързано с витамините от които се нуждаем всеки ден. Приемането на достатъчно количество от всеки един витамин, може да допринесе за нашето здраве и като цяло да ни накара да се чувстваме по-добре. Основните витамини на които ще се обърне внимание в това приложение са: А, D, C, E, B12 и други. Всеки един витамин играе важна роля в човешкия организъм. Например Една от най-важните функции на витамин A е подкрепата на растежа на клетките и клетъчното делене. По този начин той се оказва жизнено значим за организма като цяло. Това му е спечелило името „витамин на растежа”. Витамин C играе роля в имунната система, подобрявайки абсорбирането на нехемово желязо. Нужен е за синтеза на колаген, Л-карнитин и други невространсмитери. Участва и в метаболизма на протеини. Само един пример за важен ген, който витамин D регулира, е способността на организма да се бори с инфекции и хронично възпаление. Той също така произвежда над 200 антимикробни пептида.

Това приложение успява да генерира хранителен режим на потребителя според това какви витамини му липсват. Менютата, които биват извадени не са задължителни да се спазват безусловно. Използването на само едно или две от показаните храни може да допринесе за по-здравословен начин на живот.

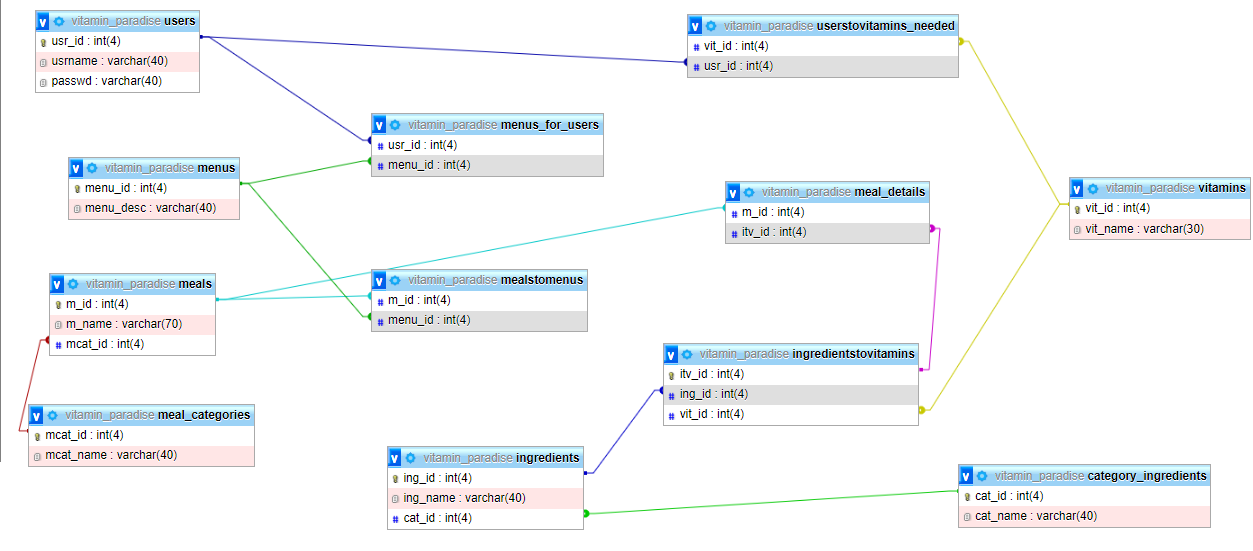
# **2.Аналитична част**

## **2.1. База от данни**

За да е възможен достъпа на приложението до всички храни бе използван mysql сървър, на който бе създадена база от данни, включваща дванадесет таблици, които включват всяка нужна информация за да бъде създадено това приложение и да се осигури оптимален хранителен план. Тези таблици биват свързани една с друга чрез външни ключове по такъв начин че да може да се достъпва лесно всяка една информация нужна на потребителя.

В случая имаме една таблица, която съдържа всички витамини, като всеки един от тях притежава уникален номер, с който се отбелязват в другите таблици. По същия начин имаме таблица за продуктите, които ще използваме за съставянето на менютата . Тези две таблици са свързани чрез трета таблица. По този начин можем да видим всеки един продукт колко и какви витамини има. Няколко продукта образуват едно ястие, а от своя страна ястията биват разделени на категории според времето на консумация и образуват менюта. Тези менюта биват записани на един потребител в отделна таблица, за да може да му се намери перфектното меню според нуждите му. Накрая между таблицата с потребители и таблицата с витамини е направена трета таблица, която ги свързва, за да бъде запазена информацията на за това кой витамин е нужен на даден потребител.

Базата от данни можеше да се структурира по много различни начини, но те не бяха толкова ефикасни. Първоначално имаше възможност продуктите да се разделят на 5 различни вида и от там да се направят 5 таблици за всеки вид след което тези 5 таблици да се свържат към шеста таблица като по тази начин да се формира ястие. Този подход обаче имаше 2 проблема. Първият беше, че към ястията може да се добавя само един продукт от дадена група, което би намалило вариантите за възможни ястия и би ни ограничило в избора ни. Вторият проблем бе това че таблиците ставаха твърде много, което ми забавило процеса на извеждане на дадено ястие и всички негови продукти. В сегашния модел има само една таблица за продукти и една таблица за категории като тези две таблици са свързани и може да разберем към коя категория принадлежи всеки един продукт, както и да добавим множество продукти от една категория към едно ястия.



## **2.2. Промени в базата от данни**

беше замислено да се добави още една таблица, наречена „DEFICIENCIES“ в която да се съдържа видовете липси, които може да притежава един потребител както и таблица “Users\_To\_ DEFICIENCIES”, която да ни насочва каква липса има всеки един потребител. В крайна сметка се разбра че таблицата „DEFICIENCIES“ и тази която съдържа всички витамини е идентична, затова беше взето решението едната таблица да се премахне, тъй като другата бе излишна и в крайна сметка просто беше създадена таблицата “Userstovitamins\_needed”. По този начин се спести създаването на още една таблица.

Първоначално беше замислено яденията и менютата да се попълнят автоматично, като комбинираме всеки един продукт с друг от друга категория. Така едно ядене щеше да съставя продукти от всяка една категория. Този метод обаче не изглеждаше ефикасен, тъй като потребителя ще вижда единствено и само продукти, докато при статичното добавяне както е направено в случая, има различни методи за приготвяне на яденето. Обмисляше се идеята да се добави още една таблица, свързана с таблицата ingredients, която показва как всеки един продукт може да бъде сготвен. Тази идея обаче не се реализира.

За да се създаде връзката между нашето приложение и базата данни са добавени специални библиотеки, чрез които успяхме да направим конекцията. Първоначално, за да се осъществи връзка трябва да добавим потребителско име и парола на някой от потребителите в базата както и наименованието на сървъра след което вече имаме възможност да въвеждаме mysql команди в нашето приложение.

## **2.3 Работа на приложението**

Посредством езика C++ е възможно да се създаде приложение за генериране на хранителен режим.

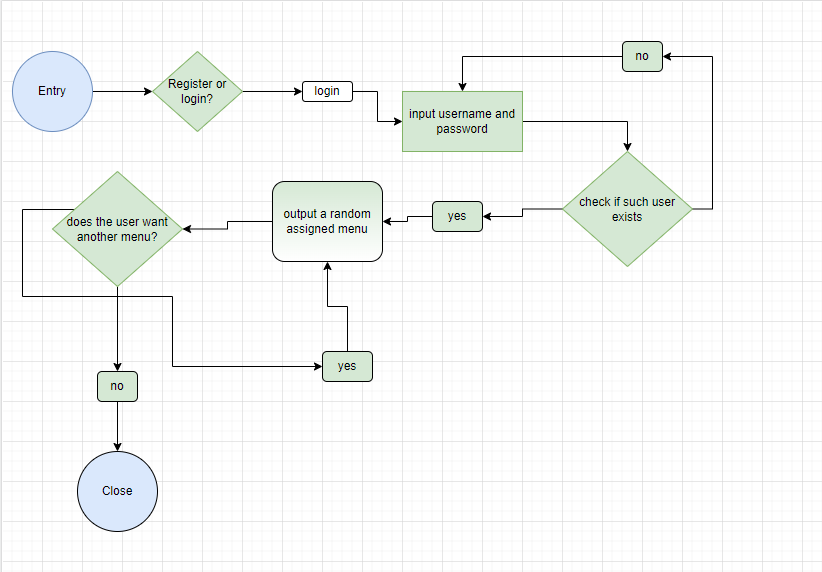
Целта на това приложение е да се изведат най-подходящите менюта според нуждите на потребителя. За да се осъществи желаната работа бе нужно добавяне на допълнителни библиотеки, за да могат да се правят заявки към сървъра.

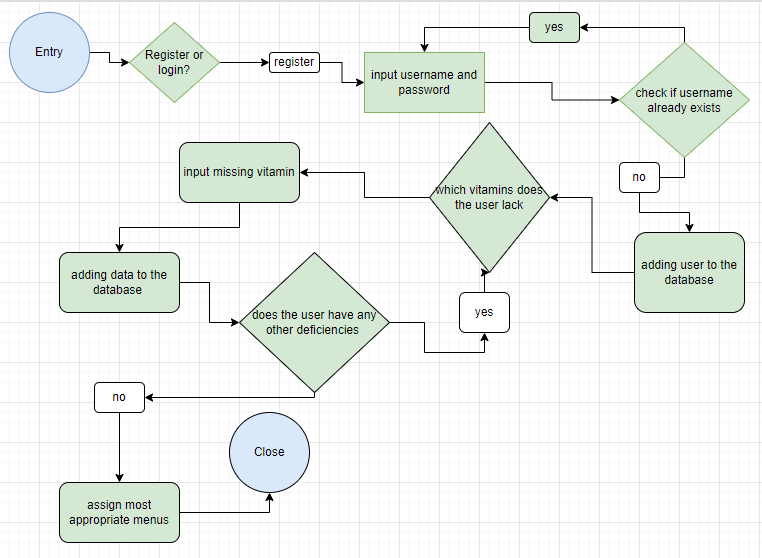
Метода на работа на приложението е следния: В начало се задава въпрос на потребителя дали иска да влезе в профила си или да създаде свой ако вече няма създаден. При избор за регистрация, потребителя въвежда желаните име и парола след което се прави проверка дали въведеното потребителско име вече съществува. Ако името вече го има, то приложението подканва потребителя да въведе друго. След въвеждането на уникално потребителско име, данните на клиента се записват в таблицата users, а той трябва да въведе витамините, които му липсват. След въвеждането на един витамин се прави проверка дали този витамин съществува в таблицата “vitamins”. Ако няма такъв витамин, програмата кара потребителя да въведе витамина още веднъж. След това се задава въпрос дали има липсва на още витамини и ако потребителя отговори положително на този въпрос, програмата подканва потребителя да въведе следващия витамин. Всеки пък когато той въведе един витамин, тези данни се записват в таблицата „userstovitamins\_needed“, като на всеки нов ред се добавя номера на потребителя и номера на витамина. По този начин ще разполагаме с информацията за всеки липсващ витамин на потребителя. Според това кои витамини е въвел потребителят се избират менюта от таблицата „menus“ , които предварително са създадени. Всяко избрано меню се добавя на нов ред на таблицата “menus\_for\_users” срещу номера на потребителя. По този начин лесно може да се достъпват менютата за избран потребител и да се извеждат в приложението.

Пример за това е когато потребителя се регистрира и въведе, че му липсва витамин D. При влизане на потребителя ще му се изведе едно от подходящите менюта по следния начин:

* закуска: Сандвич с авокадо и яйце
* обяд: телешко с ориз
* междинно хранене: ягоди
* вечеря: Печени картофи

След което, потребителят ще бъде попитан дали иска да пробва друго меню и ако отговори положително да се появи друго такова.





Ako клиентът избере опцията да влезе в своя профил, тогава той бива подканен да въведе своето име и парола след което се прави проверка в таблицата “users” дали той съществува. При несъответствие, потребителят въвежда име и парола отново. След влизането се извежда произволен план от вече генерираните такива при регистрацията и се извеждат идеи за закуска, обяд, следобед и вечеря.

# **3. Описание на кода**

## **3.2 Описание на обектите**

За програмна среда на разработка бе използван Visual Studio 2022.

В приложението са създадени няколко обекти, а функциите са поставени в съответния обект. Така всяка една функция може да се намери по-лесно в кода вместо всяка една функция да е отделна и да няма подредба.

Програмата използва 3 класа.

1. USR

Този клас бе създаден с цел да събере всички данни на потребителя в един обект. Класът USR се състои от следните функции:

* bool checkExist(string usrname, string passwd, bool reg) - Прави проверка на всеки ред от таблицата users дали въведеното потребителско име и/или парола съществуват.
* void reg(string usrname, string passwd) – добавя въведените потребителско име и парола в таблицата Users
* int getusrid(string usrname) – според даденото потребителско име, от таблицата се изважда съответния му номер

1. VIT

Класът VIT успява да побере всички витамини и техните детайли в един обект. Състои се от следните функции:

* bool checkvit(string vitname) – Според въведения витамин, функцията връща истина или не, ако той съществува в таблицата vitamins
* int getvid(string vitname) – Изважда се номера на въведения витамин от таблицата Vitamins
* void insertvitotusr(int vitid, int usrid) – Чрез номера на потребителя и номера на липсващия витамин се добавя нов ред в таблицата users\_to\_vitamins\_needed със съответно въведените данни от потребителя.
* vector<int> getvitids(int usrid) – връща вектор който е пълен с номерата на липсващите витамини на потребителя
* bool existCheck(int usrid, int vitid) – връща булева стойност според това дали въведения витамин от потребителя не е вече записан в таблицата userstovitamins\_needed.

1. MENUS

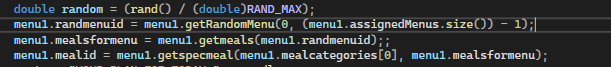
Този обект бе създаден с целта, всички менюта и техните ядения, принадлежащи на един потребител да бъдат записани в един обект. Състои се от следните функции:

* vector<int> getmenus(vector<int> vits) – връща се вектор, който е пълен с номерата на всички менюта имащи съответните витамини, от които се нуждае потребителя. Данните се извличат от множество таблици присъединени чрез join.
* void assign(vector<int> menus, int usrid) – Чрез вектора от предходната функция се възлагат менютата от получения вектор на съответния потребител в таблицата menus\_for\_users.
* int getRandomMenu(int low, int high) – връща произволна стойност, която се намира между двете граници. В нашия случай се използва, за да се изведе номера на едно произволно меню.
* vector<int> getAssignedMenus(int usr\_id) – Връща вектор, който е пълен с номерата на възложените менюта на дадения потребител от таблицата menus\_for\_users.
* vector<int> getmeals(int menuid) – тази функция връща вектор, който съдържа номерата на всяко ядене от избраното меню. От таблицата mealstomenus се взима всеки един номер на ядене, срещу което стои съответния номер на меню.
* string getspecmeal(string mealcat, vector<int> assignedMeals) – Връща номер на яденето, което е от специфична категория и принадлежи на вектора, който се получава от предходната функция. Данните се извличат като се прави съединение между двете таблици meal\_categories и meals.

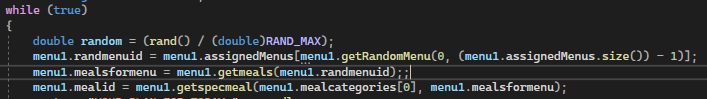
## **3.2 Промени в кода и поправяне на грешки**

При първото пускане се установи, че яденията, което се извеждат всъщност изобщо не принадлежат на вектора който съдържа номерата на яденията. В крайна сметка се разбра, че вместо номера на произволно ядене се извежда номера на произволна позиция на вектора, което в случая е грешно. За да се разреши проблема трябваше просто полученото число да се постави в квадратни скоби и да се сложи името на вектора преди него. По този начин се селектира частта от съдържанието на вектора с този номер.

Преди:



След:



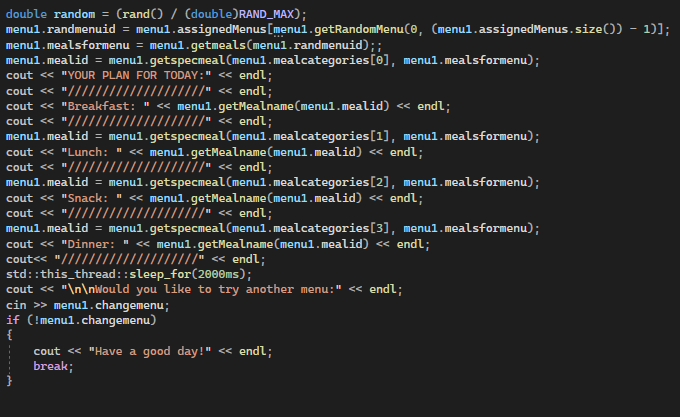
При преглед на кода се установи, че има две почти идентични функции - Функцията, която проверява дали сме въвели правилно име и парола при влизане, и тази която проверява, при регистрация, дали въведеното потребителско име вече съществува. Единствената разлика е, че при опит за влизане със създаден потребител се проверява и паролата, докато в другата функция се проверява само името. Затова беше взето решението да се премахне функцията за проверка при регистрация и просто да се използва функцията за влизане. Така при регистрация се проверява и паролата, но се спестява една цяла функция. Това обаче доведе до още един проблем, а именно това че когато потребителят се опита да се регистрира със съществуващо име, но с различна парола, неговото действие ще се позволи от програмата. За да се разреши този проблем бе добавена още една променлива във функцията от булев тип. В случай, че нейната стойност е положителна, се оказва на програмата, че потребителя се опитва да се регистрира и че е нужно да се провери съществуването само и единствено на въведеното потребителско име.

Когато беше нужно да се пита потребителя дали би желал да му се покаже друго меню или дали има друг липсващ витамин докато се регистрира, трябваше той да въвежда 1 или 0, след което тази стойност се запазваше в булева променлива и се използваше в различни “if” условия. В следствие, булевата променлива бе сменена от променлива тип “string”, а if условията бяха променени. Пример за това би било, когато се задава въпрос дали потребителят иска да му се изведе ново меню. След което се проверява дали той е въвел “no”,”n” или -, при което се дава край на програмата.

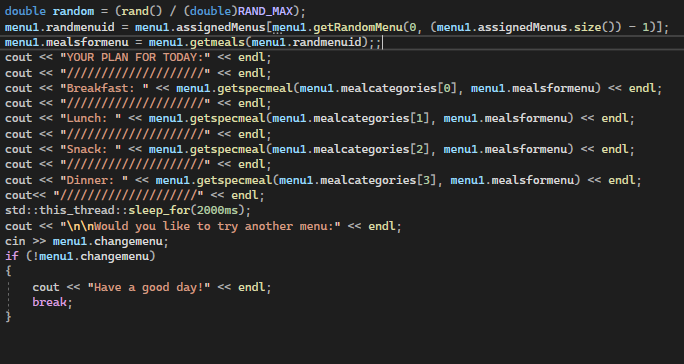
Един от проблемите, които се появи във заявките към сървъра беше че не можеха да се приемат целочислени променливи, когато се правеше опит да се добавят в условието на определената заявка. Проблемът се получаваше при опит за използване на “Select” и “Insert”. За да се разреши проблемът при “Select” беше нужно единствено, стойността на целочислената променлива да се зададе на друга променлива от типа string, но с добавена функцията to\_string пред нея. По този начин, когато добавим тази променлива от типа string вместо целочислената променлива, грешката не се появява. При използването на “Insert” е възможно да се използва допълнителна функция „pstmt” или “Prepare statement”, която позволява добавянето на нужните променливи след самата заявка. Метода, който се използва за справяне с грешката при използването на “select”, е възможен да се използва и при “Insert”, но този начин не се използва в приложението.

Друга важна промяна бе оставането в програмата след самата регистрация. Тъй като потребителя, най вероятно ще желае да влезе в своя профил след като се регистрира, нямаше нужда програмата да приключва. По тази причина беше направено така че на потребителя да се извежда същия въпрос, който се задава в началото, а именно да влезе в своя профил или да се регистрира.

След допълнителен преглед на кода беше взето решение да се слеят някои от функциите, за да се спести извикването на всяка една от тях. Преди:

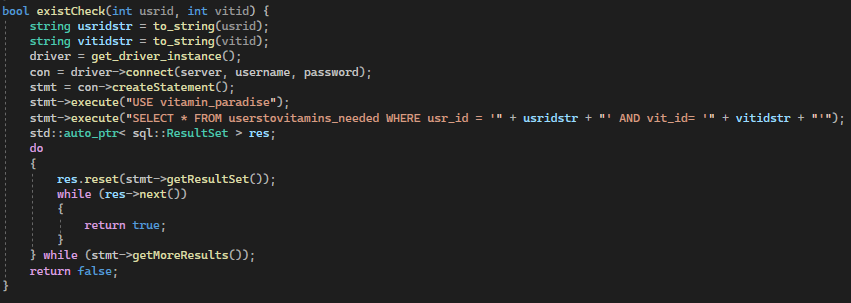


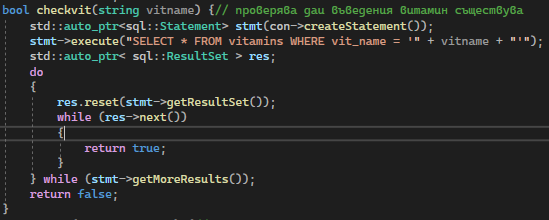
След:



Първоначално имаше две отделни функции за да се намери номера на яденето и да се изведе името на избраното ядене според неговия номер. Нямаше смисъл от това, двете функции да са отделни затова те бяха обединени в една.

След много тестове се установи, че при регистрация, портебителяt има възможността да въведе един и същ витамин няколко пъти, което би довело до повтарящи се редове в таблицата userstovitamins\_needed, както и дублиращи се менюта в таблицата menus\_for\_users. По тази причина беше решено да се направи нова функция, която да проверява дали въведения витамин вече не е бил попълнен в таблицата срещу името на потребителя. Подобен проблем бе това, че трябваше да се намери начин да се проверява витамина, който потребителя въвежда в случай че той изобщо не съществува. Това бе причината за създаването на още една функция, която проверява дали въведения символ присъства в таблицата vitamins.





Първоначално беше замислено, потребителят да получава едно произволно меню при влизане в профила си и да няма възможност да получава други менюта, освен ако не влезе отново. Този метод обаче изглеждаше непрактичен, затова беше решено да се добави опция. Щом произволното меню се изведе бива зададен въпрос към потребителя дали иска да види друго меню и ако той отговори положително, то ще се изведат яденията за друго произволно меню. При отрицателен отговор програмата приключва.

За увеличаване на сигурността беше променен кода по такъв, начин че когато потребителят се регистрира или влиза в профила си и се опита да въведе своята парола, тя няма да се изведе на екрана, а вместо това ще се показва символа „\*“ всеки път когато потребителя въведе някакъв символ. За да бъде успешен този метод беше нужно добавянето на още една библиотека наречена “conio.h”. При всеки един въведен символ се проверява дали той е “Enter”. В противей случай се записва в променливата която пази паролата. Проблем който се появи беше че когато потребителя се опита да изтрие предния символ, знакът „\*“ се запазва, което доведе до използването на функцията erase() с която се премахва стойност от дадена променлива.

# **4. Заключение**

Възможно е да се създаде програма на езика C++, която да генерира хранителен режим според липсата на витамини. Създадената програма следва да се използва като потребителят, първоначално се регистрира и след това въведе витамините, които му липсват.

За в бъдеще би било възможно да се добавят много различни допълнения към приложението. Пример за това е добавянето на точно количество от всеки витамин, което е нужно да се спазва, защото прекаляването с някой витамин може да доведе до лоши последици. Така потребителя ще може безусловно да следва някои от плановете, а не както тук просто да получава идеи. Разбира се това би изисквало значително увеличаване на базата от данни, тъй като ще ни е нужно точното количество от всеки витамин, което се съдържа в един продукт, както и нова таблица, която да съдържа минималното количество витамини. Това действие също така било усложнило сегашните функции и най-вероятно би увеличило броя им. Пример за друга функция е такава която може да изчислява колко точно количество от всеки витамин се съдържа в дадено ядене или меню и да го извежда на потребителя.

В бъдеще е възможно и добавяне на графичен интерфейс, което би позволило на приложението да бъде използвано от реални потребители с истинска нужда от хранителен режим за техните липси на витамини.

Има вариант приложението да се промени така че на един клиент да се поставя едно меню и то да се променя единствено, след приключване на деня. По този начин ще се следи часовника и след приключването на всеки ден, потребителя ще получава ново меню.

Друго допълнение би било, след дадените ядения да се дават съвети как потребителя да подобри своето ежедневие, така че начина му на живот да допринесе максимално за по-здравословен живот, като това може да включва разходки, разтягания, упражнения и други.

Възможно е и да се добави информация за калориите на всеки един продукт. Така освен витамините, потребителите ще могат и да следят своите калории и едновременно да поддържат добра форма. В този случай ще бъде нужно отново да се допълни базата от данни, както и функциите, а ако бъде взето решение в приложението да се включват и калории, и количество витамини, тогава ще трябва да се пишат допълнително функции, за да има баланс между двете или по избор на клиента.

В бъдещето би било възможно да се задават въпроси към потребителя ако той все още не си е направил изследвания, за да се направи диагноза и да се генерира план, без нужните изследвания.

# **5. Литература**

1. Атанас Джананов - Културизъм за всички - София - 1985