МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-40 05 01-03 Программное обеспечение информационных технологий

Специализация Программное обеспечение информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

База данных для программного средства «Дневник питания для диабетиков» с технологией мультимедийных типов данных в БД

Выполнил студент Трошко Валерия Николаевна

(Ф.И.О.)

Руководитель проекта ассист. Колмаков Михаил Владимирович

(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: ассист. Колмаков Михаил Владимирович

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: ассист. Колмаков Михаил Владимирович

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2021

**Содержание**

[Введение 3](#_Toc90883776)

[1. Постановка задачи 4](#_Toc90883777)

[1.1. Обзор прототипов и анализ 4](#_Toc90883778)

[1.1.1. Glucose Buddy Diabetes Tracker 4](#_Toc90883779)

[1.1.2. Диабет 5](#_Toc90883780)

[1.1.3. Diabetes Tracker 6](#_Toc90883781)

[1.2. Аналогичность и различие программных средств 7](#_Toc90883782)

[2. Разработка модели базы данных 8](#_Toc90883783)

[3. Разработка необходимых объектов 10](#_Toc90883784)

[3.1. Проектирование базы данных 10](#_Toc90883785)

[3.2. Процедуры для решения поставленных задач 11](#_Toc90883786)

[3.2.1. Выборка данных из таблиц 11](#_Toc90883787)

[3.2.2. Добавление данных в таблицу 11](#_Toc90883788)

[3.2.3. Изменение данных в таблице 13](#_Toc90883789)

[3.2.4. Удаление данных из таблицы 13](#_Toc90883790)

[3.3. Триггеры 14](#_Toc90883791)

[3.4. Пользователи 15](#_Toc90883792)

[4. Описание процедур экспорта и импорта 17](#_Toc90883793)

[5. Описание и реализация технологии 19](#_Toc90883794)

[6. Тестирование производительности 22](#_Toc90883799)

[7. Руководство пользователя 24](#_Toc90883800)

[7.1. Установка программного средства 24](#_Toc90883801)

[7.2. Руководство пользователя по работе с программным средством 25](#_Toc90883802)

[Заключение 29](#_Toc90883803)

[Список используемых источников 30](#_Toc90883804)

[Приложение А 31](#_Toc90883805)

[Приложение Б 32](#_Toc90883806)

[Приложение В 35](#_Toc90883807)

[Приложение Г 38](#_Toc90883808)

## **Введение**

В наше время всё больше информации хранится в цифровом виде и для хранения такой информации отлично подходят различные базы данных. Использование базы данных упрощает управление данными и их хранение. Одним из примеров успешного внедрения баз данных является использование их различными приложениями для отслеживания питания, веса и уровня сахара в крови.

Сейчас есть множество удобных приложений, позволяющих без особых усилий сохранять всю необходимую информацию, не выходя из дома, с помощью таких приложений обычно можно контролировать всё и всегда, начиная от контроля, съеденного за день и заканчивая историей уровня сахара в крови и веса.

В ходе выполнения данной курсовой работы было разработано приложение «Дневник питания для диабетиков».

Цель данного курсового проекта – создание базы данных для хранения пользовательской информации, продуктов питания и их БЖУ, ознакомление с технологией мультимедийных типов данных в БД и её применение в базе данных, а также разработка удобного программного средства для обработки этой информации. Готовое программное средство должно предоставлять возможность пользователям следить за ежедневной нормой калорий, весом и уровнем сахара в крови, а также просматривать рецепты.

Основными задачами курсовой работы являются:

* + - провести аналитический обзор прототипов;
    - спроектировать базу данных;
    - реализовать функциональность базы данных;
    - провести тестирование используемой технологии в базе данных;
    - разработать приложение для работы с базой данных;
    - написать руководство пользователя.

Для проектирования базы данных используется СУБД «Microsoft SQL Server 2019». В базе данных применяется технология мультимедийных типов данных. Приложение для работы с базой данных было создано на платформе Windows Presentation Foundation (WPF). Для работы с WPF использовался объектно-ориентированный язык программирования С#, а также декларативный язык разметки XAML. Для осуществления связи между базой данных и приложением, используется технология ADO.

## **Постановка задачи**

Основной задачей курсового проекта является разработка программного средства, позволяющего пользователю фиксировать личную информацию об уровне сахара в крови, весе, а также записывать съеденное в течении дня.

## **Обзор прототипов и анализ**

Одним из ключевых моментов в разработке программного обеспечения является просмотр и изучения различных аналогов, и поиск в них недостатков и достоинств.

* + 1. **G****lucose Buddy Diabetes Tracker**

Данное приложение было разработано под мобильные операционные системы IOS и Android.

Интерфейс приложения представлен на рисунке 1.1.

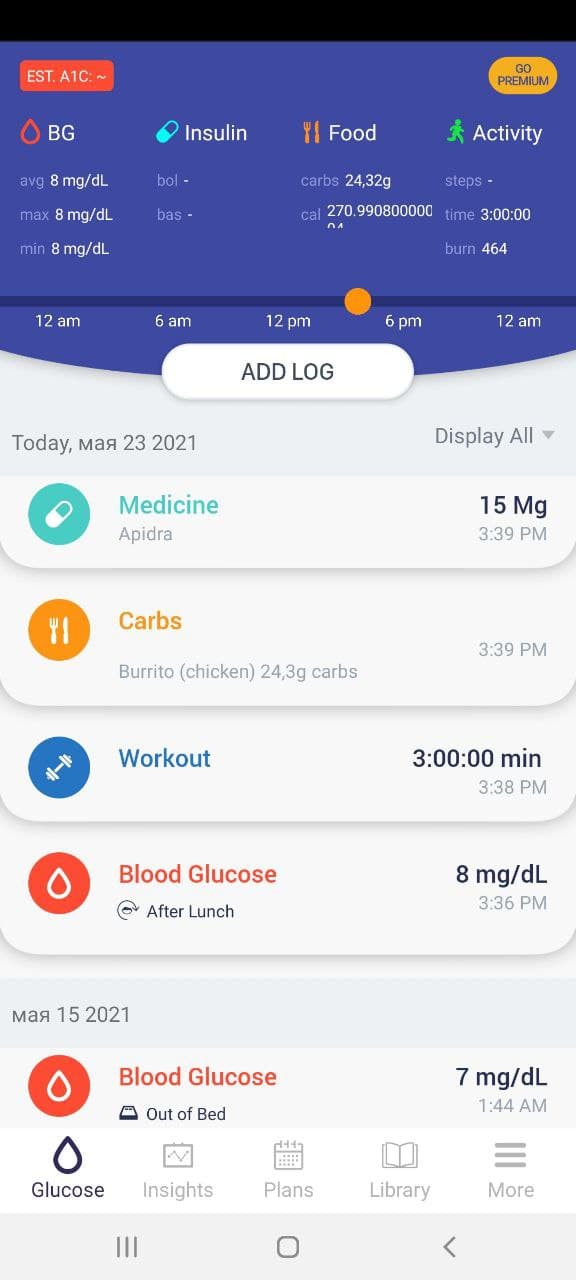


Рисунок 1.1 − Интерфейс приложения Glucose Buddy Diabetes Tracker

Имеет следующие возможности:

* + - легко регистрировать уровень глюкозы крови, лекарства, приемы пищи, в одной записи;
    - отслеживание изменения уровня сахара в крови, инсулина, веса, артериального давления, A1C и многого другого;
    - добавление заметок к вашим записям для дальнейшего использования;
    - запись своих блюд, используя нашу обширную базу данных продуктов питания.

Из недостатков необходимо отметить не самый приятный и интуитивно непонятный дизайн программного средства. Большинство возможностей доступны только в платной версии приложения. Отсутствие контроля водного баланса.

* + 1. **Диабет**

Данное приложение так же было разработано под мобильные операционные системы IOS и Android.

Внешний вид программы представлен на рисунке 1.2.

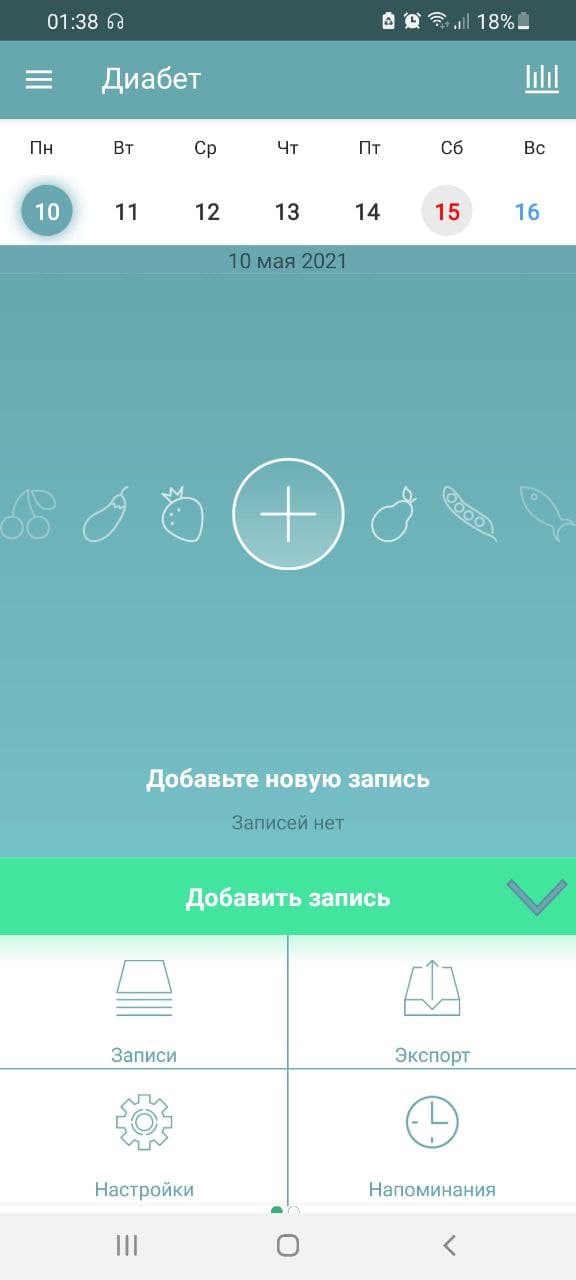


Рисунок 1.2 — Интерфейс приложения Диабет

Приложение Диабет имеет следующие возможности:

* + - Ведение дневника контроля глюкозы
    - Ведение дневника питания
    - Точные расчеты ХЕ на еду (самая большая база продуктов + возможность добавлять свои продукты)
    - Расчет доз инсулина (в том числе дробный инсулин для помпы)
    - Экспорт дневника в .pdf и .xls, печать дневника, отправка прямо из приложения в любой меседжер или на почту.

Диабет, как и приложение, рассмотренное ранее, не отличается удобством в использовании. Его интерфейс выглядит устаревшим и однозначно будет не очень приятен пользователю. В нём так же отсутствует функция записи употреблённых лекарств и функция контроля водного баланса.

* + 1. **Diabetes Tracker**

Дневник контроля глюкозы крови и питания – важная вещь в жизни диабетика. Приложение Diabetes Tracker создано в помощь как тем, кто впервые столкнулся с этим диагнозом, так и давно болеющим людям.

Главное окно программы представлено на рисунке 1.3.



Рисунок 1.3 — Интерфейс приложения Diabetes Tracker

Функционал приложения:

* + - удобный ежедневник с графами для внесения результатов многократных измерений;
    - заметки, куда можно вписывать любую другую важную информацию: объем текста – 140 знаков;
    - архив, который поможет при необходимости получить давние сведения, сохраненные ранее трех месяцев назад;
    - статистика – раздел, где анализируются и интерпретируются результаты контроля сахара;
    - цветные графики - "сахарная кривая" визуального контроля измерений сахара крови за разные промежутки времени: 2 недели, месяц, 3 месяца. Пользователям также доступен график изменений HbA1c за год;
    - интеграция с Google Drive - способ быстро и надёжно разместить в облачном хранилище все результаты измерений сахара крови;
    - настройки интерфейса.

К недостаткам данного приложения можно отнести:

* + - отсутствие контроля водного баланса;
    - приложение поддерживает только один язык;
    - скудная база продуктов;
    - отсутствует ежедневное обнуление главной страницы.

## **Аналогичность и различие программных средств**

Схожесть рассмотренных программных средств заключается в том, что все они, в первую очередь, позволяют следить за нормой КБЖУ (КБЖУ — это аббревиатура, прижившаяся в кругах людей, придерживающихся правильного питания, худеющих или же, наоборот, набирающих вес: К — калории; Б — белки; Ж — жиры; У — углеводы;). Так же они позволяют в различной степени следить за потреблением медицинских препаратов и инсулина.

Главные отличия программ кроются в их функциональности. Так, например, в приложениях Diabetes Tracker и Glucose Buddy Diabetes Tracker доступны заметки, которые позволяют пользователю облегчить просмотр его истории. В приложении Diabetes Tracker пользователь может сам регулировать внешний вид приложения, что очень манит современных людей: стремление к красоте, а в приложении Диабет существует возможность сохранения в .pdf и .xls, что так же, несомненно, очень облегчает просмотр истории, особенно тем, кто вынужден на постоянной основе посещать врача и отчитываться ему.

Таким образом, можно сделать вывод о том, что в каждом приложении есть свои плюсы и минусы. Но, разумеется, каждому разработчику хочется, чтобы его приложение не имело отрицательных качеств, т.к. это сильно влияет рейтинг приложения, от чего напрямую зависит прибыль и статус самого разработчика.

## **Разработка модели базы данных**

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Цель анализа требований в проектах — получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Определение и описание требований — шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

Анализируя ранее приведённые примеры, я составила основные требования. Различают три уровня требований к проекту:

* бизнес-требования;
* пользовательские требования;
* функциональные требования.

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга. Курсовой проект не подразумевает наличие заказчика, который мог бы выдвинуть бизнес-требования, поэтому в качестве таких высокоуровневых требований можно рассматривать общие требования к разрабатываемому средству. К их числу относятся:

* простота и лёгкость интерфейса;
* использование принципов объектно-ориентированного программирования;
* использование архитектурных шаблонов проектирования;
* использование системы управления базами данных (СУБД);

Весь дальнейший процесс проектирования и разработки программного средства должен находиться в очерченных бизнес-требованиями границах.

Следующими требованиями являются требования пользователей. Данные требования описывают цели и задачи, которые пользователям позволит решить система. Таким образом, в пользовательских требованиях указано, что клиенты смогут делать с помощью системы.

Неавторизованный пользователь данного программного решения должен иметь возможность:

* регистрировать себя в системе;
* авторизоваться;
* выходить из приложения.

Авторизованный пользователь может:

* просматривать и изменять личные данные;
* просматривать рецепты и всю информацию о них;
* отмечать съеденные продукты;
* предлагать новые продукты;
* вносить выпитые медикаменты, воду и сделанные уколы;
* просматривать статистику данных.

Администратор имеет возможность:

* просматривать уже существующие продукты/рецепты;
* удалять продукты/рецепты в которых нет надобности;
* создавать новые продукты/рецепты;
* одобрять или отклонять новые, предложенные клиентами, продукты.

Более наглядно это видно на UML-схеме (Приложение А).

После проведения анализа были выявлены следующие функциональные требования:

* архитектура приложения должна соответствовать шаблонам проектирования, таким как MVVM, Command;
* вся информация должна храниться в базе данных;
* приложение должно производить валидацию вводимых пользователем данных;
* приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;
* приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;
* приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным пользовательским именем;

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированными списком.

## **Разработка необходимых объектов**

## **Проектирование базы данных**

Для реализации поставленной задачи была создана база данных Diabet.net. Для её создания использовалась система управления реляционными базами данных Microsoft SQL Server 2019.

Прежде всего, необходимо было спроектировать корректную базу данных для работы. Моя база данных содержит следующие таблицы (рисунок 3.1):

* личные данные пользователей (Users);
* информация о продуктах (Products);
* информация о рецептах (Recipe);
* данные ингредиентов в рецептах (Prod\_ Rec);
* информация о типах приема пищи (Type\_of\_Food);
* данные о съеденной пищи пользователями в течении дня (Daily\_Food);
* информация о типах инсулина (Type\_of\_Insulin);
* данные о сделанных уколах инсулина в течении дня (Daily\_Insulin);
* данные об оставшихся калорий, выпитых воды и медикаментов пользователя в течении дня (Daily\_Cal\_Water\_Pill);
* добавленные пользователями новые продукты, ожидающие одобрение администратора (Product\_Awaiting\_Approval);
* история веса пользователей (History);
* история уровня сахара в крови пользователей (History\_Blood\_Sugar).

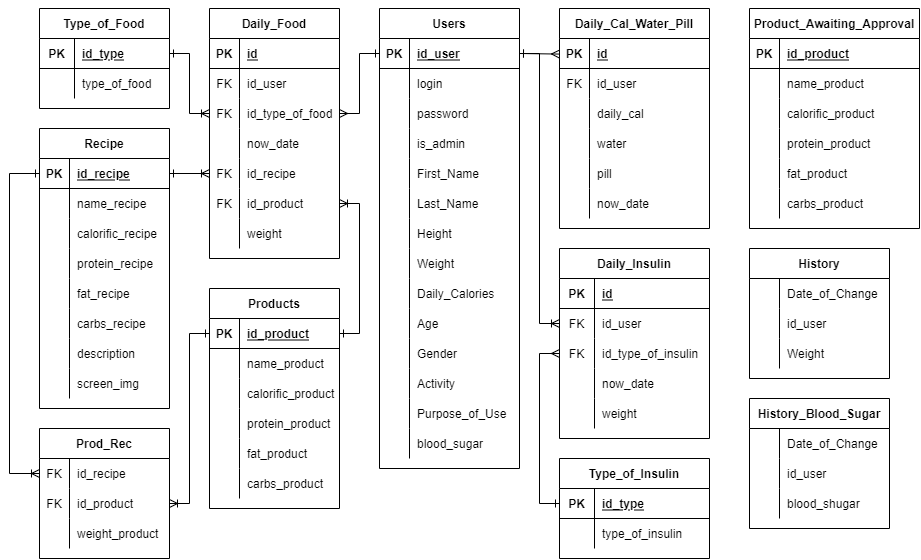


Рисунок 3.1 – Диаграмма базы данных

Листинг создания базы данных представлен в приложении Б.

## **Процедуры для решения поставленных задач**

Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

Использование хранимых процедур позволяет ограничить либо вообще исключить непосредственный доступ пользователей к таблицам базы данных, оставив пользователям только разрешения на выполнение хранимых процедур, обеспечивающих косвенный и строго регламентированный доступ к данным.

При разработке курсового проекта было создано 55 процедур для следующих целей:

1. Выборка данных из таблиц по различным полям;
2. Добавление пользователя/ проверка при авторизации;
3. Удаление данных из таблиц;
4. Добавление данных в таблицы;
5. Изменение данных в таблицах;
6. Экспорт и импорт таблиц в формат xml;
7. Заполнение таблицы на 100 000 строк.

Весь перечень созданных процедур будет представлен в Приложении В.

## **Выборка данных из таблиц**

Для выборки данных из таблиц были написаны следующие процедуры: GetProduct, GetRecipe, GetPill, GetSugar и другие. Поскольку в приложении мы часто обращаемся с поиском к таблицам Users, Product, Recipe и Products\_Awaiting\_Approval, которые содержат большое количество данных было принято решение создать индексы, тем самым сократив время поиска по этим таблицам.

## **Добавление данных в таблицу**

При создании пользователем нового продукта с помощью процедуры AddProductInApproval ему необходимо заполнить все поля формы (название, калорийность, белки, жиры, углеводы). Прежде создать его, все данные проверяются на корректность (длину, повтор и т.д.). Процедура представлена ниже в листинге 3.1:

Create or alter procedure dbo.AddProductInApproval

@name\_product nvarchar(20),

@calorific\_product int,

@protein\_product real,

@fat\_product real,

Продолжение листинга 3.1

@carbs\_product real

as

INSERT INTO Products\_Awaiting\_Approval (name\_product, calorific\_product, protein\_product, fat\_product, carbs\_product )

VALUES (@name\_product,@calorific\_product,@protein\_product, @fat\_product, @carbs\_product)

Select SCOPE\_IDENTITY()

go

Листинг 3.1 – Описание процедуры AddProductInApproval

При регистрации нового пользователя с помощью процедуры AddUser, ему необходимо заполнить необходимые поля: login, password, First\_Name, Last\_Name, Height, Weight, Age, Gender, Activity, Purpose\_of\_Use, Sugar. Для увеличения безопасности учетной записи пользователя его пароль шифруется с помощью алгоритма MD5. Также проверяется корректность ввода логина на повторение. Данная процедура представлена ниже в листинге 3.2:

Create or alter procedure dbo.AddUser

@login nvarchar(20),

@password nvarchar(100),

@is\_admin bit,

@First\_Name nvarchar(20),

@Last\_Name nvarchar(20),

@Height real,

@Weight real,

@Daily\_Calories smallint,

@Age int,

@Gender nvarchar(5),

@Activity float,

@Purpose\_of\_Use smallint,

@Sugar real

as

INSERT INTO Users (login, password, is\_admin, First\_Name, Last\_Name, Height, Weight, Daily\_Calories, Age, Gender, Activity, Purpose\_of\_Use, blood\_sugar)

VALUES (@login,@password,@is\_admin,@First\_Name,@Last\_Name, @Height, @Weight, @Daily\_Calories, @Age, @Gender, @Activity, @Purpose\_of\_Use, @Sugar)

Select SCOPE\_IDENTITY()

go

Листинг 3.2 – Описание процедуры AddUser

Есди же все поля заполнены корректно, то процедура создаст нового пользователя с ролью User.

## **Изменение данных в таблице**

Для управления базой данных были созданы процедуры обновления: UpdateWater, UpdateSugarUser, UpdatePurposeUser, UpdatePill и другие. Создание одной из таких процедур представлена ниже в листинге 3.3:

Create or alter procedure dbo.UpdateAgeUser

@id\_user int,

@age int

as

Update Users Set Age = @age Where id\_user = @id\_user

Select SCOPE\_IDENTITY()

go

Листинг 3.3 – Создание процедуры UpdateAgeUser

Все подобные процедуры были разработаны для обновления данных в соответствующих таблицах.

## **Удаление данных из таблицы**

Процедура, разработанная в рамках курсового проекта, выполняющая удаление данных: DeleteFromApproveProduct, она позволяет администратору удалять из таблицы ожидания продукты, которые ему не интересны. Скрипт создания этой процедуры приведён ниже в листинге 3.4:

Create or alter procedure dbo.DeleteFromApproveProduct

@id int

as

Delete From Products\_Awaiting\_Approval Where id\_product = @id

Select SCOPE\_IDENTITY()

go

Листинг 3.4 – Описание процедуры DeleteFromApproveProduct

Для некоторых элементов удаление ведёт за собой ряд проблем, связанное с внешними ключами, но тут нам помогают транзакции, выполнение которых позволяет нам контролировать, чтобы происходило только того, что нам надо. Таким образом построена процедура DeleteRecipe (листинг 3.5).

Create or alter procedure dbo.DeleteRecipe

@id int

as

begin tran;

begin try

Delete From Prod\_Rec Where id\_recipe = @id

Delete From Recipe Where id\_recipe = @id

commit;

end try

begin catch

rollback;

end catch

select count(\*) [count] from Recipe where id\_recipe = @id

Select SCOPE\_IDENTITY()

go

Листинг 3.5 – Скрипт процедуры DeleteRecipe

Иначе же, в противном случае, программа может откатить до начального состояния, не потеряв никакие данные.

## **Триггеры**

Для ведения статистики веса пользователя нам необходимо контролировать и фиксировать изменения веса, чтобы это делать быстро и без особых усилий, удобнее всего использовать триггер.

Триггер – это особый вид хранимой процедуры, предназначенной для обработки событий в БД, в нашем случае конкретно изменение поля веса.

Для этого в проекте был разработан триггер Users\_History, который фиксирует в отдельную таблицу все изменения.

Пример создания триггера приводится в листинге 3.6:

create or alter trigger Users\_History

on Users

after update, insert

as

begin

DECLARE @oldWeight real

SELECT @oldWeight = (SELECT Weight FROM deleted)

DECLARE @newWeight real

SELECT @newWeight = (SELECT Weight FROM inserted)

DECLARE @id int

SELECT @id = (SELECT id\_user FROM inserted)

IF @oldWeight != @newWeight

Продолжение листинга 3.6

begin

INSERT INTO History(Date\_of\_Change, id\_user, Weight ) VALUES( GETDATE(), @id, @newWeight)

end

end

Листинг 3.6 – Скрипт создания процедуры Users\_History

Таким образом, благодаря триггерам мы можем не переживать, что данные изменятся, но не зафиксируются в истории, что в дальнейшем позволит пользователю получить достоверную статистику.

## **Пользователи**

Пользователь базы данных – это физическое или юридическое лицо, которое имеет доступ к БД и пользуется услугами информационной системы для получения информации.

При проектировании базы данных было использовано 3 пользователя.

Первый пользователь – Unauthuser – неавторизованный пользователь, может только зарегистрироваться, авторизоваться или выйти из приложения.

Второй пользователь – User – зарегистрированный пользователь, имеет доступ к своим личным данным, может просматривать таблицы продуктов и рецептов, предлагать для одобрения новые продукты, может менять личную информацию, отмечать сделанные уколы, выпитые воду и таблетки, просматривать свои статистики.

Третий пользователь – Admin – обладает иным перечнем прав. Имеет права для чтения и создания новых продуктов и рецептов, может одобрять либо отклонять предложенные зарегистрированным пользователем продукты (листинг 3.7).

USE master

CREATE LOGIN [User] WITH PASSWORD = 'User', DEFAULT\_DATABASE = [Diabet.net];

CREATE LOGIN [Admin] WITH PASSWORD = 'Admin', DEFAULT\_DATABASE = [Diabet.net];

CREATE LOGIN [Unauthuser] WITH PASSWORD = 'Unauthuser', DEFAULT\_DATABASE = [Diabet.net];

USE [Diabet.net]

CREATE USER [User] FOR LOGIN [User]

WITH DEFAULT\_SCHEMA = dbo;

Продолжение листинга 3.7

CREATE USER [Admin] FOR LOGIN [Admin]

WITH DEFAULT\_SCHEMA = dbo;

CREATE USER [Unauthuser] FOR LOGIN [Unauthuser]

WITH DEFAULT\_SCHEMA = dbo;

CREATE ROLE User\_role AUTHORIZATION [User];

CREATE ROLE Admin\_role AUTHORIZATION [Admin];

CREATE ROLE Unauthuser\_role AUTHORIZATION [Unauthuser];

Листинг 3.7 – Скрипт создания ролей

Для корректной обработки данных и ограничений доступа у пользователя, все их действия прикреплены к процедурам. Каждый пользователь имеет доступ только к своим (разрешённым) процедурам. Пример выдачи прав неавторизованному пользователю приведён ниже в листинге 3.8:

--Unauthuser

GRANT SELECT ON OBJECT::Users TO Unauthuser\_role;

GRANT INSERT ON OBJECT::Users TO Unauthuser\_role;

GRANT EXECUTE ON OBJECT::[dbo].[GiveUserByLoginAndPassword] TO Unauthuser\_role;

GRANT EXECUTE ON OBJECT::[dbo].[AddUser] TO Unauthuser\_role;

GRANT EXECUTE ON OBJECT::[dbo].[GetIdUserByLogin] TO Unauthuser\_role;

GRANT EXECUTE ON OBJECT::[dbo].[GetIsAdminUser] TO Unauthuser\_role;

Листинг 3.8 – Скрипт выдачи прав неавторизованному пользователю

Скрипт выдачи прав на процедуры в приложении Г.

## **Описание процедур экспорта и импорта**

Для экспорта таблицы Products в формате xml была разработана процедура ProductToXml. Для формирования xml в select запросе используется конструкция FOR XML. Для вывода в файл была использована расширенная хранимая процедура xp\_cmdshell. Процедура ProductToXml представлена ниже в листинге 4.1.

create or alter procedure ProductToXml

as

begin

select name\_product, calorific\_product, protein\_product, fat\_product, carbs\_product

from Products

for xml path('product'), root('products');

exec master.dbo.sp\_configure 'show advanced options', 1

reconfigure with override

exec master.dbo.sp\_configure 'xp\_cmdshell', 1

reconfigure with override;

declare @cmd nvarchar(255);

select @cmd = 'bcp "use [Diabet.net]; select name\_product, calorific\_product, protein\_product, fat\_product, carbs\_product from Products for xml path(''product''), root(''products'')" ' +

'queryout "C:\test\Export.xml" -S .\SQLEXPRESS -T -w -r -t';

exec xp\_cmdshell @cmd;

end;

Листинг 4.1 – Скрипт создания процедуры ProductToXml

Для импорта данных в таблицу Products, из файла формата xml, была разработана процедура XmlToProduct. Составляющая файла представлена в листинге 4.2:

<products>

<product>

<name\_product>Помидор Черри</name\_product>

<calorific\_product>15</calorific\_product>

<protein\_product>8.0000001e-001</protein\_product>

<fat\_product>1.0000000e-001</fat\_product>

<carbs\_product>2.8000000e+000</carbs\_product>

</product>

<product>

<name\_product>Лук Репчатый</name\_product>

<calorific\_product>41</calorific\_product>

<protein\_product>1.4000000e+000</protein\_product>

<fat\_product>2.0000000e-001</fat\_product>

Продолжение листинга 4.2

<carbs\_product>8.1999998e+000</carbs\_product>

</product>

<product>

<name\_product>Морковь корейская</name\_product>

<calorific\_product>65</calorific\_product>

<protein\_product>1.4000000e+000</protein\_product>

<fat\_product>2.8900000e+001</fat\_product>

<carbs\_product>8.6000004e+000</carbs\_product>

</product>

</products>

Листинг 4.2 − Данные XML для импорта в базу данных

Для получения xml файла и последующего разбора со вставкой используется конструкция FROM OPENROWSET совместно с параметром BULK. Процедура XmlToProduct представлена ниже в листинге 4.3.

create or alter procedure XmlToProduct

as begin

DECLARE @xml XML;

SELECT @xml = CONVERT(xml, BulkColumn, 2) FROM OPENROWSET(BULK 'C:\test\Export.xml', SINGLE\_BLOB) AS x

INSERT INTO Products(name\_product, calorific\_product, protein\_product, fat\_product, carbs\_product)

SELECT

t.x.query('name\_product').value('.', 'nvarchar(20)'),

t.x.query('calorific\_product').value('.', 'INT'),

t.x.query('protein\_product').value('.', 'real') ,

t.x.query('fat\_product').value('.', 'real'),

t.x.query('carbs\_product').value('.', 'real')

FROM @xml.nodes('//products/product') t(x)

end

Листинг 4.3 – Скрипт создания процедуры XmlToProduct

Таким образом мы можем быстро и удобно импортировать данные в нашу таблицу.

## **Описание и реализация технологии**

В разработке своего проекта я использовала технологию хранения мультимедийных типов данных.

Современные базы данных могут хранить самые различные виды информации. В том числе, целые файлы.

Обычно хранения файлов непосредственно в базе данных стараются избегать, так как это приводит к усложнению процесса разработки, как самой базы данных, так и клиентского приложения. А, также к увеличению размера базы данных. Однако в целом ряде случаев именно такой подход становится наилучшим решением

Суть этой технологии предельно проста. Создаётся поле с типом данных VARBINARY(MAX) (именно этот тип данных может сохранять в себе файлы, в том числе, весьма внушительных размеров) и в него из клиентской программы загружаются файлы в виде двоичных данных.

При загрузке необходимо прочитать файл в массив байтов фрагмент кода на языке C# представлен в листинге 5.1.

private byte[] OpenImageDialog()

{

var openFileDialog = new OpenFileDialog { Filter = @"Image files (\*.jpg,\*.png)|\*.jpg;\*.png;" };

byte[] binArray = null;

if (openFileDialog.ShowDialog() == true)

{

binArray = System.IO.File.ReadAllBytes(openFileDialog.FileName);

}

else

{

return null;

}

return binArray;

}

Листинг 5.1 – Добавление изображения на С#

Далее следует подключиться к базе данных, создать запрос на добавление, присвоить значения параметров и выполнить запрос. Для этого происходит через процедуру AddRecipe, скрипт создания которой представлен ниже в листинге 5.2:

Create or alter procedure dbo.AddRecipe

@name\_recipe nvarchar(50),

@calorific\_recipe int,

@protein\_recipe real,

Продолжение листинга 5.2

@fat\_recipe real,

@carbs\_recipe real,

@description nvarchar(500),

@screen\_img varbinary(max)

as

INSERT INTO Recipe (name\_recipe, calorific\_recipe, protein\_recipe, fat\_recipe, carbs\_recipe, description, screen\_img )

VALUES (@name\_recipe,@calorific\_recipe,@protein\_recipe, @fat\_recipe, @carbs\_recipe, @description, @screen\_img)

Select SCOPE\_IDENTITY()

go

Листинг 5.2 – Скрипт создания процедуры AddRecipe

При выгрузке вначале получаем выборку с рецептом и его содержимым через процедуру GetRecipe.

Create or alter procedure dbo.GetRecipe

as

Select id\_recipe, name\_recipe, calorific\_recipe,protein\_recipe,

fat\_recipe,carbs\_recipe, description, screen\_img From Recipe

Select SCOPE\_IDENTITY()

go

Листинг 5.3 – Скрипт cоздания процедуры GetRecipe

Далее читаем содержимое файла из базы в массив байт (листинг 5.4).

SqlDataReader info = command.ExecuteReader();

object p = -1, cal\_food = -1, f\_food = -1, c\_food = 1, p\_food = -1, i = -1, d = -1, s = -1;

while (info.Read())

{

p = info["name\_recipe"]; cal\_food = info["calorific\_recipe"];

p\_food = info["protein\_recipe"]; f\_food = info["fat\_recipe"];

c\_food = info["carbs\_recipe"]; s = info["screen\_img"];

i = info["id\_recipe"]; d = info["description"];

spam.Add(new Product()

{

ID = Convert.ToInt32(i),

Name = Convert.ToString(p),

Calorific = Convert.ToString(cal\_food + "ккал"),

Protein = Convert.ToString(p\_food + "г"),

Продолжение листинга 5.4

Fat = Convert.ToString(f\_food + "г"),

Carbs = Convert.ToString(c\_food + "г"),

Description = Convert.ToString(d),

Screenimg = (byte[])s

}

);

}

Листинг 5.4 – Фрагмент кода для чтения содержимого файла в массив байт

После этого остаётся только записать байты из массива в специальный объект класса ByteToBitmapImage

public static BitmapImage ByteToBitmapImage(byte[] imageData)

{

if (imageData == null || imageData.Length == 0) return null;

var image = new BitmapImage();

using (var mem = new MemoryStream(imageData))

{

mem.Position = 0;

image.BeginInit();

image.CreateOptions = BitmapCreateOptions.PreservePixelFormat;

image.CacheOption = BitmapCacheOption.OnLoad;

image.UriSource = null;

image.StreamSource = mem;

image.EndInit();

}

image.Freeze();

return image;

}

Листинг 5.5 – Конвертация из байт массива в изображение

Для вывода картинки в приложение, в WPF есть специальный тег <Image> (листинг 5.6).

<Image VerticalAlignment="Top" Margin="385,90,10,280" Source="{Binding \_Screenimg, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"/>

Листинг 5.6 – Вывод изображения (WPF)

Таким образом, мы получаем готовое изображение в своём приложении.



## **Тестирование производительности**

К таблице ProductAwaitingApproval было разработано несколько процедур для заполнения таблицы строками в количестве 100000.

Например, одной из таких процедур была процедура InsertAppProd, представленная ниже в листинге 6.1:

Create Procedure InsertAppProd AS

Begin

DECLARE @NUMBER int, @Random\_Name nvarchar(20),

@Random\_Cal int, @Random\_Protein real,

@Random\_Fat real, @Random\_Carb real,

@set\_prod nvarchar(20);

SET @number = 1;

While @number <= 100000

BEGIN

exec CreateRandomWord 4, @Random\_Name OUTPUT;

set @Random\_Cal = floor(1000\*RAND());

set @Random\_Protein = floor(300.0\*RAND());

set @Random\_Fat = floor(300.0\*RAND());

set @Random\_Carb = floor(300.0\*RAND());

set @set\_prod = (select name\_product from Products\_Awaiting\_Approval where name\_product = @Random\_Name);

insert into Products\_Awaiting\_Approval(name\_product, calorific\_product, protein\_product, fat\_product, carbs\_product)

values (@Random\_Name, @Random\_Cal, @Random\_Protein, @Random\_Fat, @Random\_Carb);

SET @number = @number + 1;

END;

End;

Листинг 6.1 – Описание процедуры InsertAppProd

При работе InsertAppProd так же вызывается процедура CreateRandomWord, задача которой состоит в том, чтобы создавать слово заданной длины из заранее заданных символов (листинг 6.2)

Create Procedure CreateRandomWord

@size integer,

@Name char(50) OUTPUT

AS

Begin

SET @Name = (

SELECT

c1 AS [text()]

Продолжение листинга 6.2

FROM

(

SELECT TOP (@size) c1

FROM

(

VALUES

('A'), ('B'), ('C'), ('D'), ('E'), ('F'), ('G'), ('H'), ('I'), ('J'), ('K'), ('L'), ('M'), ('N'), ('O'), ('P'), ('Q'), ('R'), ('S'), ('T'), ('U'), ('V'), ('W'), ('X'), ('Y'), ('Z')

) AS T1(c1)

ORDER BY ABS(CHECKSUM(NEWID()))

) AS T2

FOR XML PATH(''));

End;

go

Листинг 6.2 – Описание процедуры InsertAppProd

Тем самым мы создаём разнообразие при случайном заполнении таблицы на довольно-таки большое количество строк.

В качестве тестирования производительности было протестировано заполнение одной из таблиц 250000 строк. Запрос на вывод всех данных из таблицы Products\_Awaiting\_Approval по итогу обрабатывался 1.8 секунд (результат на рис. 6.1).

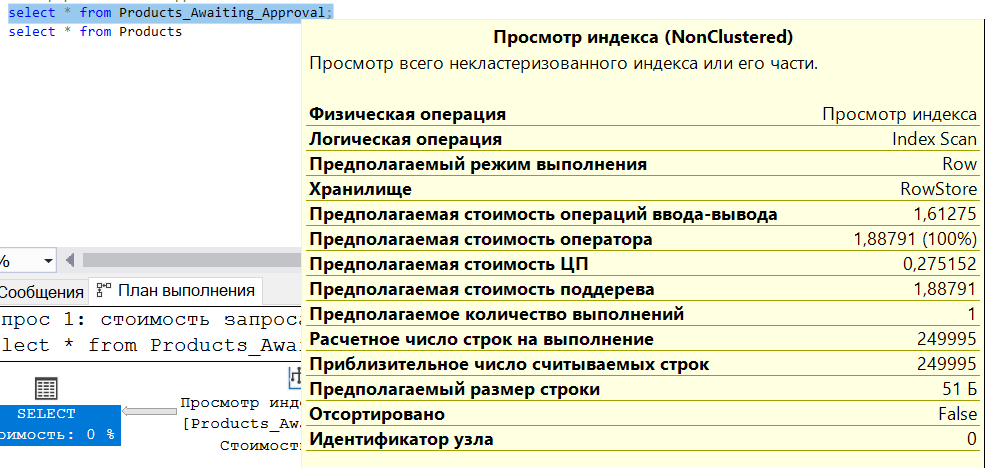


Рисунок 6.1 − Стоимость запроса на вытягивание 249995 строк

Такого быстрого результата нам помог достичь некластеризованный индекс в таблице.

## **Руководство пользователя**

## **Установка программного средства**

Для работы программного средства необходима первоначальная настройка. Для начала, необходимо наличие установленной на компьютере системы управления базами данных Microsoft SQL Server, а также утилиты SQL Server Management Studio для управления и администрирования компонентов Microsoft SQL Server.

Необходимо войти в SQL Server Management Studio, где в обозревателе объектов раскрыть список текущего подключения и нажать правой кнопкой мыши на вкладке «Базы данных». В контекстном меню требуется выбрать пункт «Присоединить», как на рисунке 7.1.

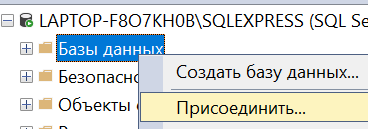


Рисунок 7.1 − Выбор пункта «Присоединить»

В открывшемся окне на панели «Базы данных для присоединения» необходимо нажать кнопку «Добавить» и указать физический путь к базе данных проекта Diabet.net (файл MDF). После этого нажать на кнопку «ОК».

После всех проделанных шагов в раскрывающемся списке «Базы данных» обозревателя объектов должна появиться база данных Diabet.net (рис. 7.2).

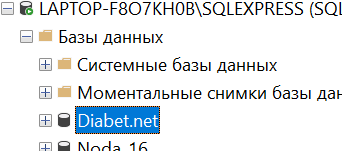


Рисунок 7.2 − Появление базы данных Diabet.net

Далее необходимо запустить установщик приложения и установить на свой компьютер. На рисунке 7.3 представлен установщик. После чего на рабочем столе будет создан ярлык для быстрого запуска приложения.



Рисунок 7.3 − Установщик приложения Diabet.net

Теперь программное средство готово к запуску.

## **Руководство пользователя по работе с программным средством**

После успешного присоединения базы данных и установки приложения пользователю необходимо запустить файл Diabet.net.exe. После открытия окна авторизации новому пользователю требуется перейти к окну регистрации, в котором правильно заполнить все поля. Условия валидности вводимых данных можно прочитать, если навести курсор мыши на название поля.

Так, например, пароль не может быть длиннее 8 символов, а в поля возраста, веса, роста, возраста и желательного сахара нельзя вводить ничего кроме цифр. Пример правильного заполнения полей показан на рисунке 7.4.

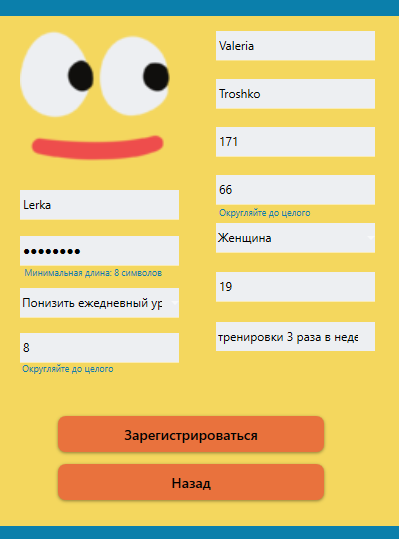


Рисунок 7.4 − Пример правильного заполнения полей окна регистрации

После успешной регистрации, пользователь авторизуется и попадает на главную страницу программного средства.

На главной странице пользователь сразу может отметить выпитые им лекарства и стаканы воды, отметить сделанные уколы инсулина, а также отметить съеденные им продукты, блюда (рис. 7.5).



Рисунок 7.5 − Пример главного меню пользователя

Выбрав тип приёма пищи, пользователь выбирает что конкретно и сколько он съел (рис. 7.6).

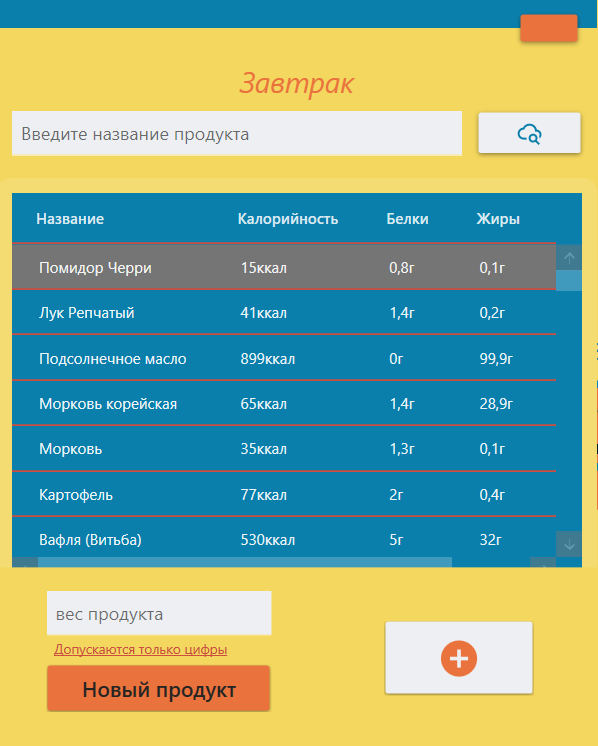


Рисунок 7.6 − Окно выбора съеденного

В случае, если в базе нет какого-то продукта/блюда, а пользователь желает его в дальнейшем видеть, он может сам создать его, заполнив специальную форму и ждать одобрения администратора (рис. 7.7).

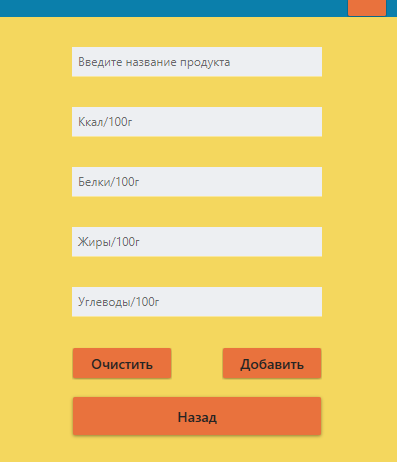


Рисунок 7.7 − Форма для создания нового блюда

На странице рецептов, пользователю представляются все рецепты, которые есть в базе в удобном формате (рис. 7.8).

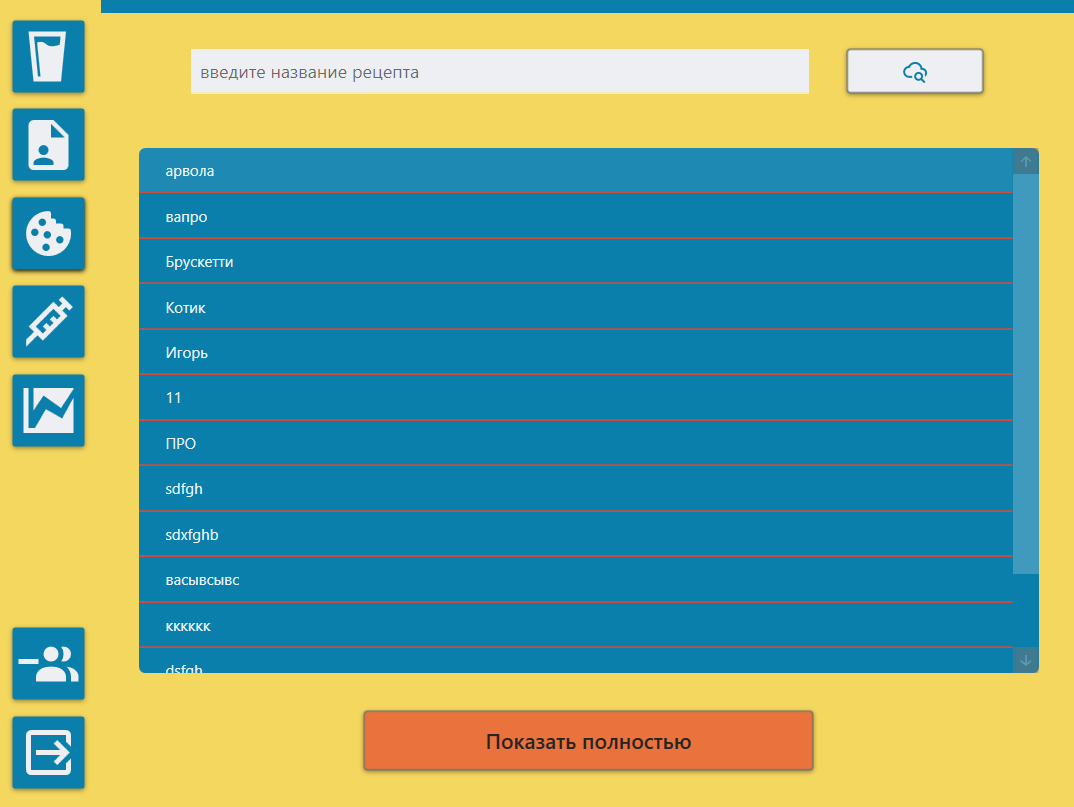


Рисунок 7.8 − Список рецептов, имеющихся в базе

По желанию пользователь может выбрать интересующее его блюдо и просмотреть информацию о нём (рис. 7.9).

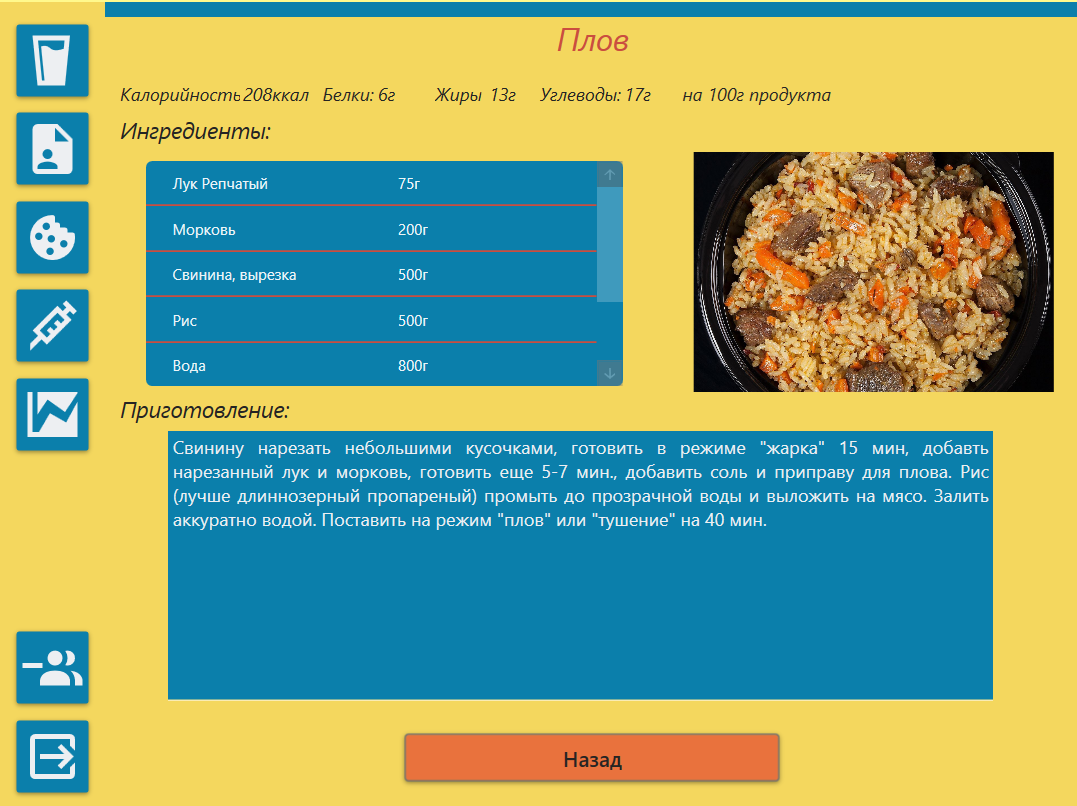


Рисунок 7.9 − Информация о выбранном рецепте

Для просмотра своей истории веса или уровня сахара в крови, пользователь может зайти на страницы статистики (рис 7.10).

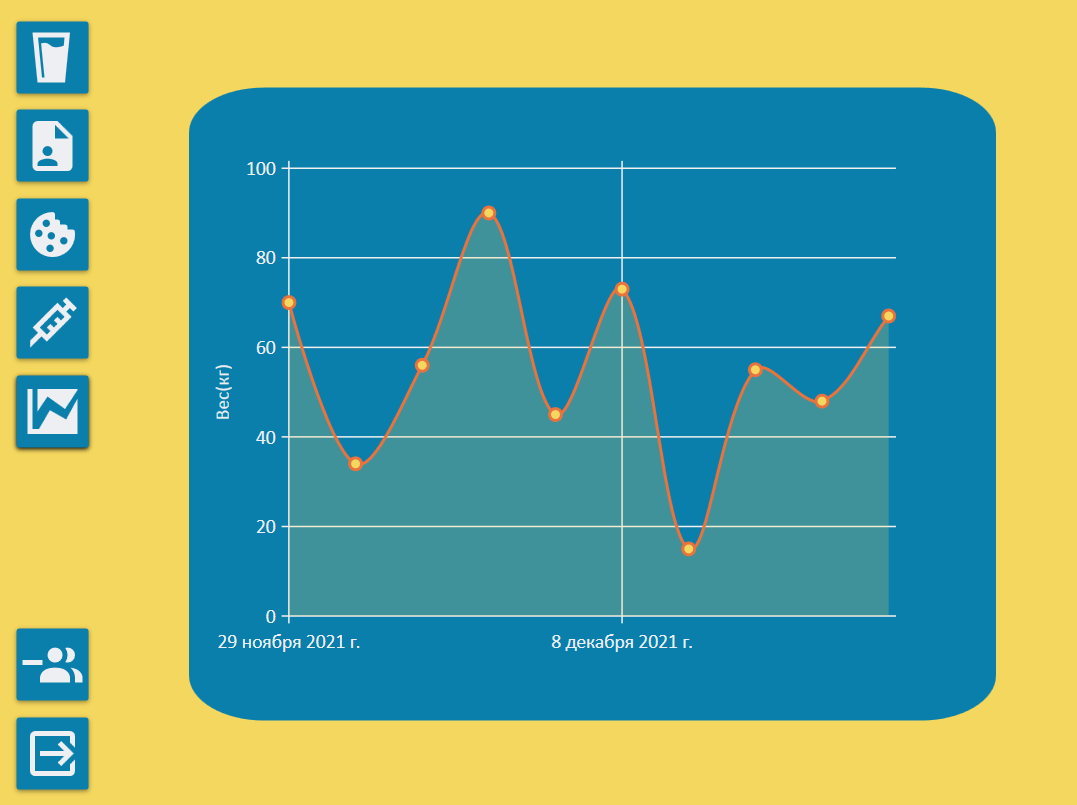


Рисунок 7.10 − Статистика веса пользователя

Статистика пользователя строится на истории изменений веси и содержания сахара в крови, соответственно.

## **Заключение**

В ходе выполнения курсовой работы был разработан проект, являющийся помощником в управлении режимом питания и контролем веса, с уклоном для людей с диагнозом «Сахарный диабет». Во время выполнения данной курсовой работы было изучено немало теоретического материала, а также просмотрено и разобрано большое количество уже готовых решений тех или иных задач. Были разработаны, описаны и применены на практике процедуры экспорта и импорта данных; разработаны процедуры резервного копирования и восстановления базы данных; было описано руководство пользователя для приложения.

Функционально были выполнены следующие задачи:

* + - администратор одобряет, удаляет новые продукты;
    - администратор добавляет продукты и рецепты;
    - клиент просматривает и изменяет личные данные;
    - клиент просматривает рецепты;
    - клиент добавляет съеденные продукты.

Приложение разработано на платформе .NET Framework с использованием технологий WPF и ADO, реализацией технологии мультимедийные типы данных в БД, а также объектно-ориентированного языка программирования C#.

Данное программное средство имеет удобный и понятный пользователю интерфейс, корректно обрабатывает ошибочный ввод данных (например, ввод данных некорректного формата, попытка оставить поля пустыми), а также некоторые внутренние ошибки.

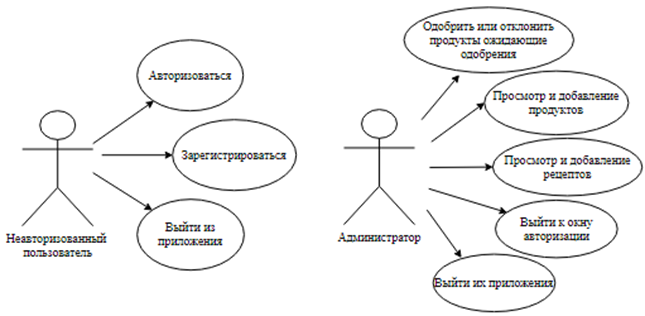
В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объёме.

Естественно, существует ряд приложений, имеющих функционал и дизайн в разы лучше. Подводя итог всей курсовой работе, можно сделать выводы, что в разработке приложений мне предстоит еще долгий путь.

## **Список используемых источников**

1. Garofallo, R. Building Enterprise Applications with Windows Presentation Foundation and the Model View ViewModel Pattern / R. Garofallo, California: O’Reilly Media, 2011, 220 p.
2. Microsoft Справочник по Transact-SQL [Электронный ресурс]/ docs.microsoft.com– Режим доступа: https://docs.microsoft.com/ru-ru/tsql/language-reference?view=sql-server-ver15 – Дата доступа 02.12.2021
3. Блинова Е.А. Курс лекций по базам данных / Е.А. Блинова
4. Интернет-портал [Электронный ресурс] / [https://docs.microsoft.com](https://vk.com/away.php?to=https%3A%2F%2Fdocs.microsoft.com&cc_key=) – SQL Server Documentation: [https://docs.microsoft.com/en-us/sql/sql-server/sql-s..](https://vk.com/away.php?to=https%3A%2F%2Fdocs.microsoft.com%2Fen-us%2Fsql%2Fsql-server%2Fsql-server-technical-documentation%3Fview%3Dsql-server-2017&cc_key=) – Дата доступа: 27.10.2021
5. Информационный сайт: Diabetes [Электронный ресурс]. –Режим доступа: https://play.google.com/store/apps/details?id=com.axel\_stein.glucose\_tracker&hl=ru&gl=US. Дата доступа: 24.05.2021.
6. Информационный сайт: GlucoseBuddy [Электронный ресурс]. Режим доступа:https://play.google.com/store/apps/details?id=com.skyhealth.glucosebuddyfree&hl=ru&gl=US. Дата доступа: 24.05.2021.
7. Информационный сайт: базы данных [Электронный ресурс]. Режим доступа:inform.sch117nn.edusite.ru/DswMedia/klyaks\_netuchitelyutestyibazyidannyix.htm – Дата доступа: 24.05.2021.
8. Информационный сайт: Диабет [Электронный ресурс]. Режим доступа: https://play.google.com/store/apps/details?id=ru.hintsolutions.diabets&hl=ru&gl=US. Дата доступа: 24.05.2021.
9. Информационный сайт: Организация хранения файлов в базе данных Microsoft SQL Server [Электронный ресурс]. –Режим доступа: <https://streletzcoder.ru/organizatsiya-hraneniya-faylov-v-baze-dannyih-microsoft-sql-server-universalnyiy-sposob>. Дата доступа: 24.11.2021.
10. Пацей, Н.В. Курс лекций по языку программирования C# / Н.В. Пацей. – Минск: БГТУ, 2016. – 175 с.
11. Сайт о программировании Metanit: Введение в ADO.NET [Электронный ресурс]– Режим доступа: https://metanit.com/sharp/adonet/1.1.php – Дата доступа: 24.05.2021.

## **Приложение А**



## **Приложение Б**

|  |
| --- |
| use master;  create database [Diabet.net];  use [Diabet.net]  create table Users  (  id\_user int Identity (1,1) primary key,  login nvarchar(20) not null unique,  password nvarchar(100) not null,  is\_admin bit not null, --0-user, 1-admin  First\_Name nvarchar(20) not null,  Last\_Name nvarchar(20) not null,  Height real ,  Weight real,  Daily\_Calories smallint,  Age int,  Gender nvarchar(5), --0-M, 1-Ж,  Activity float ,  Purpose\_of\_Use smallint,  blood\_sugar real null  );  insert into Users ( login, password, is\_admin, First\_Name, Last\_Name )  values ( 'Kisa', 'ccefd1bf9e84154d3f11b6745eb1cfe3', 1,'Админ', 'Админский' );  create table Products(  id\_product int identity(1,1) primary key,  name\_product nvarchar(20) unique,  calorific\_product int,  protein\_product real,  fat\_product real,  carbs\_product real  );  create table Recipe(  id\_recipe int identity(1,1) primary key,  name\_recipe nvarchar(50) not null,  calorific\_recipe int,  protein\_recipe real,  fat\_recipe real,  carbs\_recipe real,  description nvarchar(500) null,  screen\_img varbinary(max) null  );  create table Prod\_Rec  (  id\_recipe int foreign key (id\_recipe) references Recipe(id\_recipe),  id\_product int foreign key (id\_product) references Products(id\_product),  weight\_product smallint  );  create table Products\_Awaiting\_Approval  (  id\_product int identity(1,1) primary key,  name\_product nvarchar(20),  calorific\_product int,  protein\_product real,  fat\_product real,  carbs\_product real  );  create table Type\_of\_Food  (  id\_type int identity(1,1) primary key,  type\_of\_food nvarchar(20)  );  insert into Type\_of\_Food (type\_of\_food) values ('Завтрак'),('Обед'),('Ужин'), ('Перекус');  create table Daily\_Food  (  id int identity(1,1) primary key,  id\_user int foreign key (id\_user) references Users(id\_user),  id\_type\_of\_food int foreign key (id\_type\_of\_food) references Type\_of\_Food(id\_type),  now\_date date default CONVERT (date, SYSDATETIME()) not null,  id\_product int foreign key (id\_product) references Products(id\_product),  id\_recipe int foreign key (id\_recipe) references Recipe(id\_recipe),  weight real not null  );  create table Daily\_Cal\_Water\_Pill  (  id int identity(1,1) primary key,  id\_user int foreign key (id\_user) references Users(id\_user),  daily\_cal int,  water real,  pill real,  now\_date date default CONVERT (date, SYSDATETIME()) not null  )  create table Type\_of\_Insulin  (  id\_type int identity(1,1) primary key,  type\_of\_Insulin nvarchar(40)  );  insert into Type\_of\_Insulin (type\_of\_insulin) values ('Дневной (быстрого действия)'),('Ночной (длительного действия)');  create table Daily\_Insulin  (  id int identity(1,1) primary key,  id\_user int foreign key (id\_user) references Users(id\_user),  id\_type\_of\_insulin int foreign key (id\_type\_of\_insulin) references Type\_of\_Insulin(id\_type),  now\_date date default CONVERT (date, SYSDATETIME()) not null,  weight real not null  );  create table History  (  Date\_of\_Change datetime,  id\_user int,  Weight real  );  create table History\_Blood\_Sugar  (  Date\_of\_Change datetime,  id\_user int,  blood\_shugar real  ); |

## **Приложение В**

|  |  |  |
| --- | --- | --- |
| Название процедуры | Описание | Принимаемые параметры |
| AddApprovalMln | Процедура заполнения таблицы на 100000 строк (вместо имени числа) |  |
| AddDailyCal | Вычет калорий из дневной нормы | id\_user, daily\_cal |
| AddIngred | Добавление ингредиентов в рецепт | id\_recipe, id\_product, weight\_product |
| AddInsulin | Добавление инсулина | id\_user, weight, id\_type\_of\_insulin, now\_date |
| AddProductInApproval | Добавление продукта в таблицу ожидания | name\_product, calorific\_product, protein\_product, fat\_product, carbs\_product |
| AddProductInDailyFood | Добавление продукта в рацион по типу приёма пищи | id\_user, id\_product, weight, id\_type\_of\_food, now\_date |
| AddProductInProduct | Добавление продукта сразу в базу продуктов | name\_product, calorific\_product, protein\_product, fat\_product, carbs\_product |
| AddRecipe | Процедура добавления рецепта | name\_recipe, calorific\_recipe, protein\_recipe, fat\_recipe, carbs\_recipe, description, screen\_img(varbinary(max)) |
| AddRecipeInDailyFood | Добавление рецепта в рацион по типу приёма пищи | id\_user, id\_recipe, weight, id\_type\_of\_food, now\_date |
| AddUser | Процедура создания пользователя | login, password, is\_admin bit, First\_Name, Last\_Name, Height, Daile\_Calories, Age, Gender, Activity, Purpose\_of\_Use, Sugar |
| ApprovalProduct | Процедура одобрения продукта | Id |
| CreateRandomWord | Процедура для создания рандомного строкового значения | size, Name |
| DeleteFromApprove-Product | Процедура удаления из таблицы ожидания | Id |
| DeleteProduct | Удаление продукта из таблицы продуктов | Id |
| DeleteRecipe | Удаление рецепта из таблицы рецептов | Id |
| GetAllApproveProduct | Получение всех продуктов, ожидающих одобрения |  |
| GetCalProductByID | Процедура поиска калорийности продукта по ID | id\_product |
| GetCalRecipeByID | Процедура поиска калорийности рецепта по ID | id\_recipe |
| GetDailyCal | Проверка на наличие ежедневной нормы калорий | id\_user, now\_date |
| GetDailyCalInTableUser | Проверка на наличие нормы калорий | id\_user |
| GetDateForDailyCal | Проверка на наличие потраченных калорий | id\_user, now\_date |
| GetDateForInsulin | Проверка на наличие уколотого инсулина | id\_user, now\_date |
| GetDateForPill | Проверка на наличие выпитых медикаментов | id\_user, now\_date |
| GetDateForWater | Проверка на наличие выпитой воды | id\_user, now\_date |
| GetDateFromHistory | Получение даты из истории | id\_user, type |
| GetIdProductByName | Получение ID продукта по названию | name\_product |
| GetIdRecipeByName | Получение ID рецепта по названию | name\_recipe |
| GetIdUserByLogin | Получения ID пользователя по логину | Login |
| GetInfoFromHistory | Получение данных из истории | id\_user, type |
| GetIngtedients | Процедура поиска ингредиентов в рецепте | id\_recipe |
| GetInsulin | Получение данных по инсулину по ID, дате и типу | Id\_user, now\_date, id\_type\_of\_insulin |
| GetIsAdminUser | Процедура проверки для определения роля | id\_user |
| GetNameFoodByIdType | Получение всего съеденного по ID приёма пищи | id\_user, id\_type\_of\_food |
| GetNameProductById | Получение названия продукта по ID | id\_product |
| GetNameRecipeById | Получение названия рецепта по ID | id\_recipe |
| GetPill | Получение выпитых медикаментов по ID и дате | id\_user, now\_date |
| GetProduct | Получение всех продуктов |  |
| GetRecipe | Получение всех рецептов |  |
| GetSearchProduct | Поиск продукта | search\_TextBox |
| GetSearchRecipe | Поиск рецепта | search\_TextBox |
| GetSugar | Получение данных уровня сахара по ID и дате | id\_user |
| GetTypeOfFoodById | Получение типа приёма пищи по ID | id\_type |
| GetUserInfo | Получение личной информации пользователя | id\_user |
| GetWater | Получение количества выпитой по ID и дате | id\_user, now\_date |
| GiveUserByLogin-AndPassword | Авторизация пользователя | login, password |
| InsertAppProd | Процедура заполнения таблицы на 100000 строк (вместо имени рандомно заполненное слово) |  |
| ProductToXml | Процедура экспорта в XML |  |
| UpdateAgeUser | Обновление возраста пользователя | id\_user, age |
| UpdateDailyCal | Обновление нормы калорий пользователя (после каждого приёма пищи) | id\_user, daily\_cal, now\_date |
| UpdateDailyCalUser | Обновление нормы калорий пользователя (пересчёт нормы) | id\_user, daily |
| UpdateMassUser | Обновление веса пользователя | id\_user, weight |
| UpdatePill | Обновление выпитых медикаментов пользователя | id\_user, pill, now\_date |
| UpdatePurposeUser | Обновление цели пользователя | id\_user, Purpose\_of\_Use smallint |
| UpdateSugarUser | Обновление уровня сахара в крови пользователя | id\_user, blood\_sugar |
| UpdateWater | Обновление выпитой воды пользователя | id\_user, water, now\_date |

## **Приложение Г**

Скрипт распределения доступа по ролям

|  |
| --- |
| --User  GRANT SELECT ON DATABASE::[Diabet.net] TO User\_role;  GRANT INSERT ON DATABASE::[Diabet.net] TO User\_role;  GRANT UPDATE ON OBJECT::Daily\_Cal\_Water\_Pill TO User\_role;  GRANT UPDATE ON OBJECT::Daily\_Food TO User\_role;  GRANT UPDATE ON OBJECT::Daily\_Insulin TO User\_role;  GRANT UPDATE ON OBJECT::Users TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[AddDailyCal] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[AddInsulin] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[AddPill] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[AddProductInApproval] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[AddProductInDailyFood] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[AddRecipeInDailyFood] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[AddWater] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetCalProductByID] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetCalRecipeByID] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetDailyCal] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetDailyCalInTableUser] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetDateForDailyCal] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetDateForInsulin] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetDateForPill] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetDateForWater] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetDateFromHistory] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetIdProductByName] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetIdRecipeByName] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetIdUserByLogin] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetInfoFromHistory] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetIngtedients] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetInsulin] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetIsAdminUser] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetNameFoodByIdType] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetNameProductById] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetNameRecipeById] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetPill] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetProduct] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetRecipe] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetSearchProduct] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetSearchRecipe] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetSugar] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetTypeOfFoodById] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetUserInfo] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetWater] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GiveUserByLoginAndPassword] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[UpdateAgeUser] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[UpdateDailyCal] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[UpdateDailyCalUser] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[UpdateMassUser] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[UpdatePill] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[UpdatePurposeUser] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[UpdateSugarUser] TO User\_role;  GRANT EXECUTE ON OBJECT::[dbo].[UpdateWater] TO User\_role;  --Admin  GRANT SELECT ON DATABASE::[Diabet.net] TO Admin\_role;  GRANT INSERT ON DATABASE::[Diabet.net] TO Admin\_role;  GRANT UPDATE ON OBJECT::[dbo].[Prod\_Rec] TO Admin\_role;  GRANT UPDATE ON OBJECT::[dbo].[Products] TO Admin\_role;  GRANT UPDATE ON OBJECT::[dbo].[Recipe] TO Admin\_role;  GRANT UPDATE ON OBJECT::[dbo].[Products\_Awaiting\_Approval] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[AddIngred] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[AddProductInProduct] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[AddRecipe] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[ApprovalProduct] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[DeleteFromApproveProduct] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[DeleteProduct] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[DeleteRecipe] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetAllApproveProduct] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetCalProductByID] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetCalRecipeByID] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetIdProductByName] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetIdRecipeByName] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetIdUserByLogin] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetIngtedients] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetIsAdminUser] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetNameFoodByIdType] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetNameProductById] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetNameRecipeById] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetProduct] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetRecipe] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetSearchProduct] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GetSearchRecipe] TO Admin\_role;  GRANT EXECUTE ON OBJECT::[dbo].[GiveUserByLoginAndPassword] TO Admin\_role; |