Оглавление

[Введение 3](#_Toc500465679)

[1.Постановка задачи 4](#_Toc500465680)

[1.1 Обзор и анализ прототипов 4](#_Toc500465681)

[1.2 Актуальность решаемой задачи 6](#_Toc500465682)

[2. Разработка базы данных 7](#_Toc500465683)

[2.1 Проектирование базы данных 7](#_Toc500465684)

[2.2 Процедуры для решения поставленных задач 8](#_Toc500465685)

[2.2.1 Процедуры получения данных из таблиц 8](#_Toc500465686)

[2.2.2 Процедуры обновления данных 9](#_Toc500465687)

[2.2.3 Процедура вставки нового блюда 9](#_Toc500465688)

[2.2.4 Процедура чтения данных из XML 11](#_Toc500465689)

[2.2.5 Процедура записи данных в XML 11](#_Toc500465690)

[2.3 Привилегии 12](#_Toc500465691)

[2.4 Резервное копирование 13](#_Toc500465692)

[3. Технология SQL Server Reporting Services (SSRS) 18](#_Toc500465693)

[4. Руководство пользователя 26](#_Toc500465694)

[5. Тестирование 29](#_Toc500465695)

[Заключение 32](#_Toc500465696)

[Список использованных источников 33](#_Toc500465697)

[Приложение А 34](#_Toc500465698)

[Приложение Б 35](#_Toc500465699)

# Введение

Компьютеризация в наши дни затрагивает все больше и больше различных отраслей. Не так давно у нас в Беларуси вышел указ о компьютеризации библиотек учебных заведений. Да только государственные структуры оставили ни с чем в плане компьютеризации, рутинная работа осталась рутинной, а ошибки, основанные на человеческом факторе при учете данных, так и остались.

Кто начал создавать библиотечные базы данных на ПК при помощи офисных приложений (в Access или Excel), а кто стал искать «кряки» к разнообразным платным системам управления библиотеками. Поэтому темой данного курсового проекта стала разработка базы данных для учета библиотечных заказов.

Сегодня я познакомлю Вас с одной замечательной системой учета библиотеки. C ее помощью Вы сможете довольно легко описать все ваши книги, включая тэги, по которым можно найти эти книги, и жанры, к которым эта книг относится. Также можно удобно добавлять заказы, оперировать датами заказа, датой вероятного возвращения книги и датой возвращения книги.

Программа предназначена в основном для работников разного рода библиотек, будь то деревенская библиотека, библиотека школы, либо Национальная библиотека Беларуси. Она станет незаменимым помощником при автоматизации рутины библиотечной жизни, а также позволит использовать возможности компьютерных базы для надежного хранения информации, позволит проверять валидность вводимой информации и получать отчеты по заказам.

Данный проект основан на взаимодействии десктопного приложения с базой банных, где последняя является основной частью.

# 1.Постановка задачи

## 1.1 Обзор и анализ прототипов

На сегодняшний день существует небольшое число программных средств для учета книг в библиотеке. И даже из представленных в свободном доступе вариантов, многие работают либо некорректно, либо имеют ограниченный функционал.

На самых популярных мобильных платформах iOS и Android есть два «родных» приложения — «Здоровье» у Apple и Google Fit у Google, — которые могут собирать информацию из сторонних приложений, а также позволяют им обмениваться этой информацией между собой. Это идеальная опция для перфекционистов, так как она позволяет одновременно пользоваться несколькими приложениями, беря из каждого самое лучшее. Например, из одного — продвинутую систему тренировок, из другого — возможность отслеживать любые виды физической активности, из третьего — огромную базу продуктов питания и разные системы правильного питания.

**"Учет книг"**

"Учет книг" — программа для учета и каталогизации книг, журналов, документов, учет читателей, отслеживание должников, учет издательств, авторов, произведений и других объектов. "Учет книг" - удобное решение для автоматизации как небольших, домашних так и больших, промышленных библиотек. Программа поможет Вам быстро осуществлять поиск книги, отслеживать историю выдачи книги, своевременно узнавать о должниках, хранить необходимую информацию о читателях, вести читательский счет, контролировать даты выдачи и возврата книги. Ряд стандартных отчетов предоставит всю необходимую статистическую информацию.

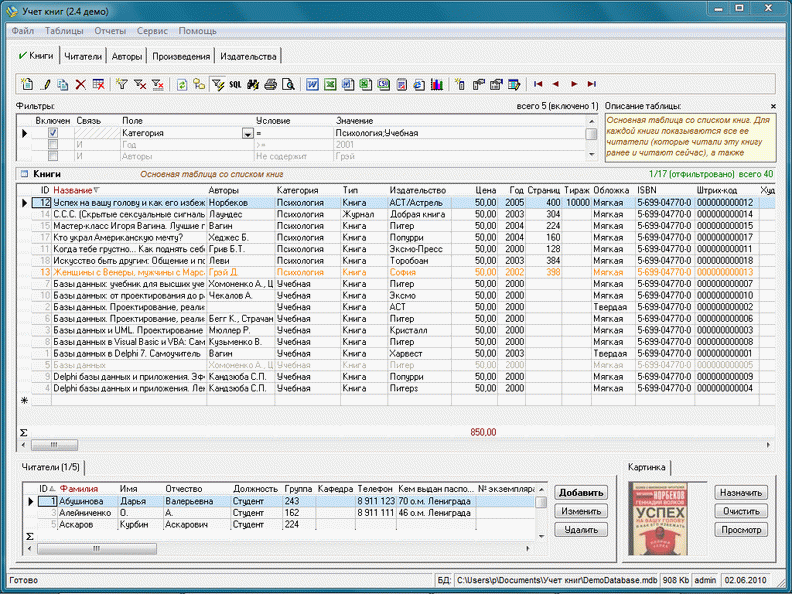


Рисунок 1.1 - Учет книг

**All My Books**

All My Books - одна из лучших программ для учета книг, ведь ручная работа по созданию каталога в программе сведена к минимуму. Выбираете из результатов поиска нужную книгу и найденная информация уже перед вами в удобном виде, включая картинку обложки! Таким образом вы сможете добавлять сотни книг в час!

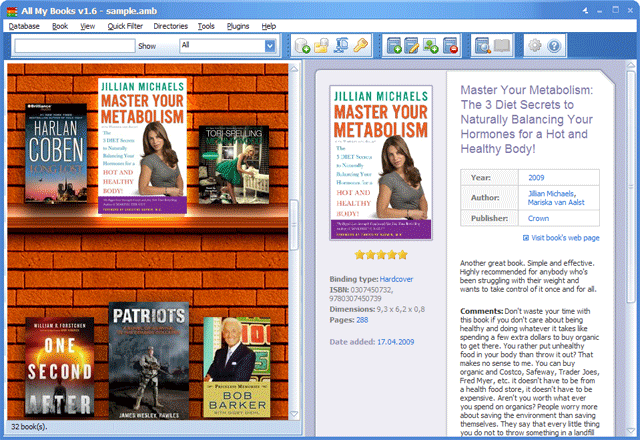


Рисунок 1.2 - All My Books

**Для учета книг в библиотеке 2.5**

Для учета книг в библиотеке - программа для учета книг или DVD материалов в личной библиотеке. Программа создана для библиотеки храма Св. Сильвестра Омско-Тарской Епархии.

В программе можно просматривать:

* Историю книги (кому и когда ее давали);
* Контактную информацию людей которые берут книгу;
* Автор и название книги;
* Местоположение книги (например номер полки).

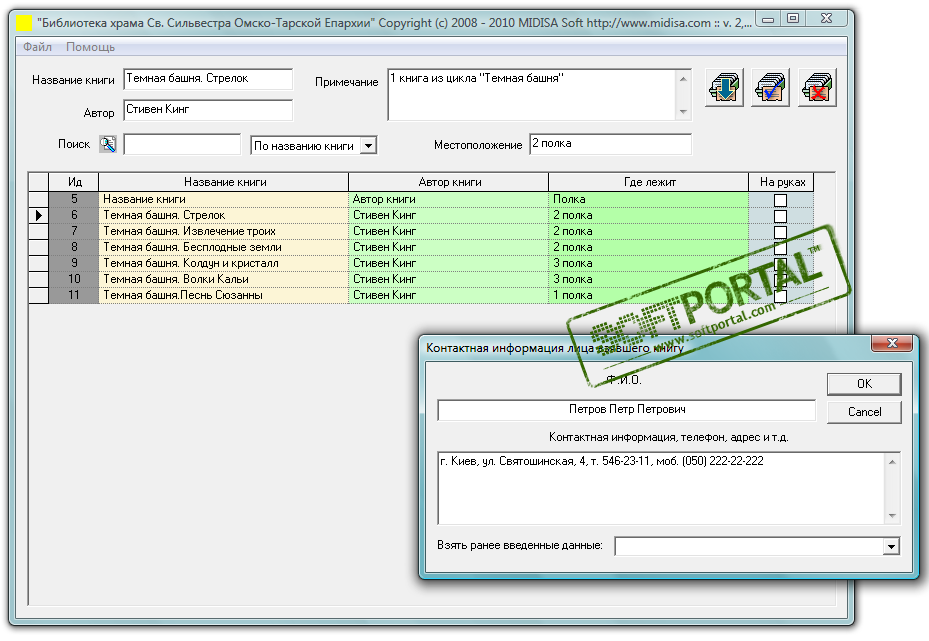


Рисунок 1.3 – Для учета книг в библиотеке 2.5

Если подвести итоги и выделить некоторую общую функциональность для всех приложений данной тематики, то она будет в себя включать следующее:

* учет заказов;
* большая база книг с их описанием;
* контроль заказов книг;
* отслеживание заказов.

Учитывая данную информацию и была спроектирована база данных для решения проблемы управления учетов заказов библиотеки и разработано приложение для работы с ней.

## 1.2 Актуальность решаемой задачи

Компьютеризация стала повседневным занятием для автоматизации различного рода занятий, где важна точность человеческих действий, где, возможно, бытует рутина и где важна корректность и актуальность данных.

Разработка базы данных для учета заказов в библиотеке стала с одной стороны рукой помощи для работников библиотек, а с другой символическим жестом внедрения компьютеров даже в такие отрасли как библиотеки с бумажными книгами.

# 2. Разработка базы данных

## 2.1 Проектирование базы данных

Для хранения и предоставления данных пользователю, была разработана база данных, которая содержит в себе восемь таблиц, 8 из которых связаны друг с другом внешними ключами, данная структура представлена на рисунке 2.1.

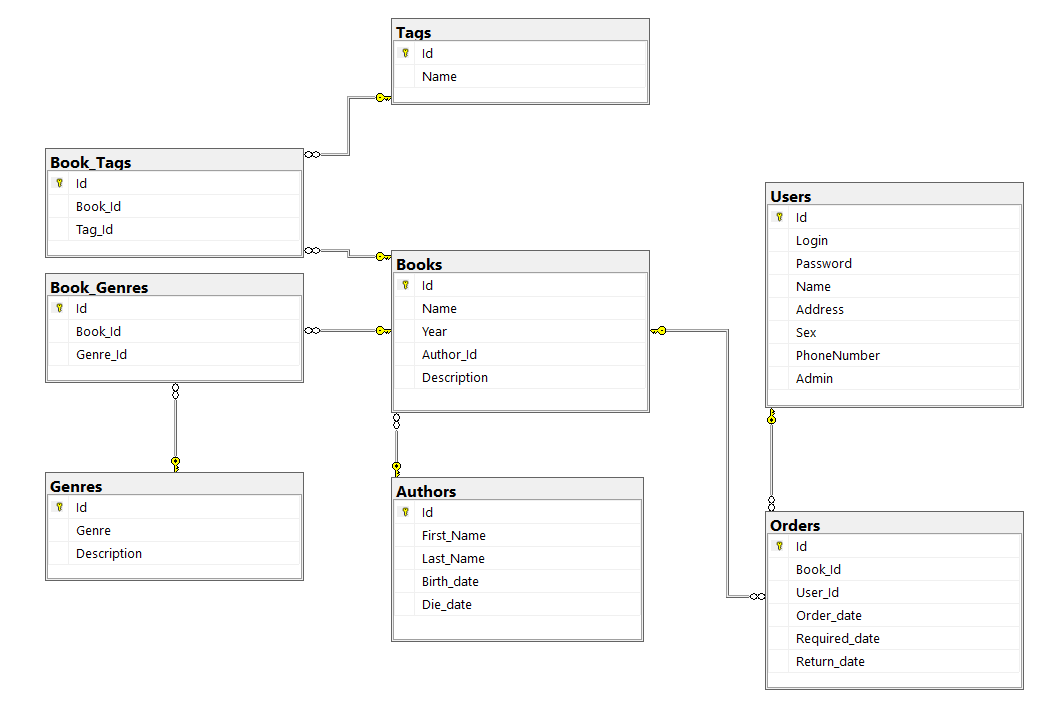


Рисунок 2.1 – Диаграмма базы данных

Таблица Books содержит информацию о книге, а именно: ее имя, год публикации книги, идентификатор автора из таблицы Authors и описание книги.

Таблица Authors хранит информацию об авторах, где хранится имя, фамилия, год рождения и год смерти, если автор умер.

Таблица Tags содержит тэги, ассоциированные с книгами. Тэги в принципе нужны для поиска книг и маркировки основных моментов, описываемых в книге. Таблица содержит только имя тэга.

Таблица Genres описывает жанры, которые описывают книгу. Содержит в себе описание жанра и его имя.

Таблицы Book\_Genres и Book\_Tags нужны для реализации связи многие-ко-многим таблиц Genres и Tags с таблицей Books соотвевественно.

Таблица Users содержит в себе информацию о пользователе. Поля, используемые в таблице: логин, пароль(нужны для аутентификации в приложении), поле, показывающее тип аккаунта(нужно для авторизации; нашей реализации имеет два значения: 1 – юзер, 2 - админ), который может использоваться в приложении, так же имеется имя пользователя, его пол(значения только ‘Female’ и ‘Male’), номер телефона и адрес проживания.

Таблица Orders содержит информацию о заказах. Является промежуточной между таблицами Books и Users, следовательно хранит поля с идентификаторами книги и пользователя соответственно. Поле Order\_date хранит значение даты заказа книги. Поле Require\_date хранит дату, когда пользователь должен был вернуть книгу. Поле Returned\_date(может быть NULL) хранит дату реального возврата книги. Данные поля реализуют удобный механизм для нахождения должников, просмотра текущих заказов, и заказов, где книги еще на руках.

## 2.2 Разработка необходимых объектов

### 2.2.1 Процедуры CRUD операций

Для всех из 8 таблиц были разработаны процедуры CRUD операций, а именно: выборка по Id, выборка из все таблицы, обновление всех полей в таблице, вставка в таблицу полей и удаление записи по Id. Данные операции позволяют реализовать принцип работы с БД посредством процедур. Имея пользователя с ролью в БД и логин для входа на сервер, можно управлять всем функционалом, абстрагируясь от конкретных реализаций операций CRUD посредством работы с операторами БД, используя только процедуры.

### 2.2.2 Настройка процедур

В каждой из процедур используется настройка процессом выполнения функции.



Рисунок 2.2 – Настройки работы фунции

Где **set nocount on** – отключает кол-во обработанных строк при работе с операторами DML внутри процедуры

И **set xart\_abort on** – инструкция языка Transact-SQL вызывает ошибку, вся транзакция завершается и выполняется ее откат.

Каждая процедура из CRUD списка содержит в себе транзакцию, которую мы явно открываем и коммитим, если все хорошо.

### 2.2.3 Исключительные ситуации

В процессе работы процедуры может возникнуть непредвиденная ситуация, которая в дальнейшем может привести к ошибки либо некорректной работе процедуры в целом.

Для решения этой проблемы были предприняты две меры: использования транзакций внутри процедур и настройки **set xart\_abort on**, а также использования оператора генерации исключений. Примет генерации исключении можем увидеть ниже. Данная процедура реализует вставку данных в таблицу Books, тут происходит проверка на корректность вводимых значений дат:

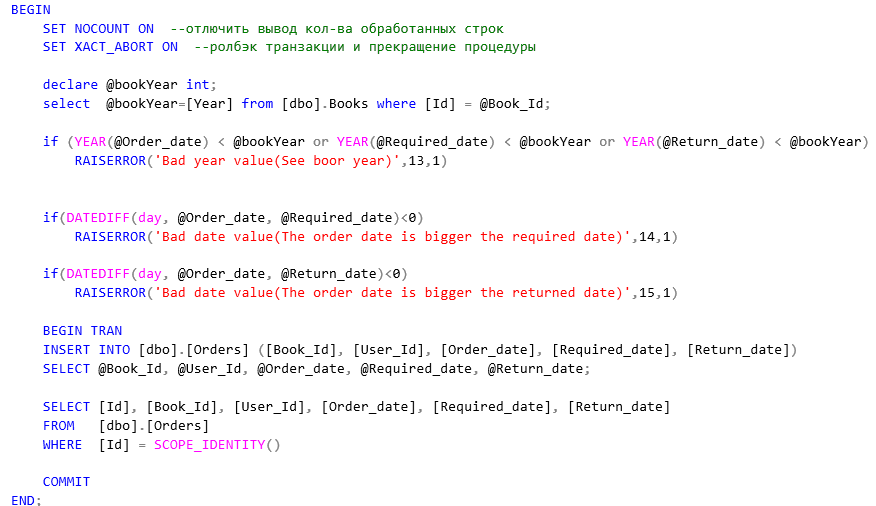


Рисунок 2.3 – Генераций исключений в процедуре

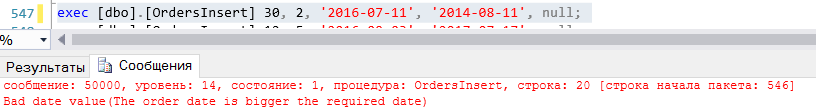
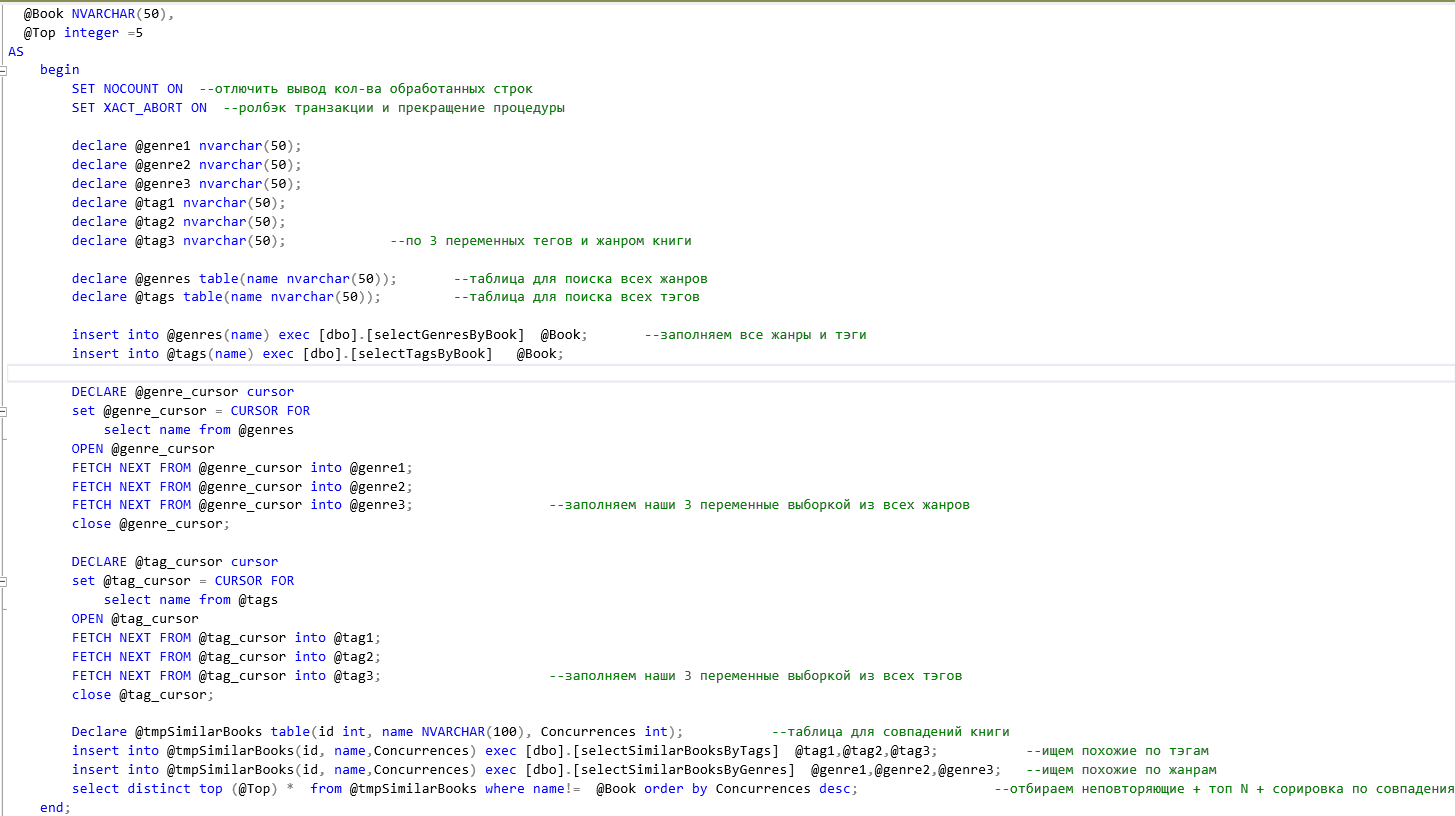


Рисунок 2.4 – Попытка вставить заказ с некорректными данными

Функция RAISERROR вызывает исключение с текстом сообщения и уровню важности. Исключения выбрасываются в основной поток выполнения стрипта и движок SQL Server выводит сообщения на консоль.

### 2.2.2 Процедура поиска похожих книг

Используя тэги и жанры, процедура подсчитывает книги с наибольшим совпадением тэгов или жанров исходя из данных книги, отправленной в процедуру.



Процедура сначала получает все тэги и жанры данной книги, потом ищем, самые распространенные в БД книги по жанрам и тэгам, подсчитывая их упоминания с помощью процедур selectSimilarBooksByTags и selectSimilarBooksByGenres, а далее на основе найденных данных строит результирующий набор, содержащий самые похожие книги на основе тэгов и жанров в порядке убывания совпадений. Листинги процедур selectSimilarBooksByTags и selectSimilarBooksByGenres представлены в приложении А.

!!!ТУТ закончил

### 2.2.3 Процедура вставки нового блюда

Данная процедура разработана для создания нового блюда питания. В нее отправляется имя, id продуктов питания в виде строки через пробелы, и таким же образом количество грамм этих продуктов. В ней используется функция для преобразования строковой переменной, содержащей int-значения, перечисленные через пробел, в таблицу, содержащую два столбца, а именно: первый – позицию числа в строке, второй – значение. Так же следует упомянуть, что сумма грамм продуктов в блюде должна равняться сотне грамм, так как нужно поддерживать процентное соотношение продуктов в блюде.

Листинг:

create procedure AddDish --создание процедуры

@Name varchar(30), -- имя блюда

@Products varchar(50), --список id продуктов питания

@Grams varchar(100) -- список грамм этих продуктов в блюде

as begin

begin tran -- начало транзакции

--проверяем, чтобы блюда с таким же именем не было в таблице

declare @a int = (select count(ID\_Dish) from Dishes where Name = @Name)

if(@a > 0) rollback tran; -- если такое блюдо есть - завершаем выполнение

else

begin

insert into Dishes ( Name) --иначе создаем блюдо с таким именем

values ( @Name);

-- получаем id данного блюда

declare @id\_dish int = (select ID\_Dish from Dishes where Name = @Name);

-- считаем сколько грамм суммарно образует список продуктов

declare @sumgram int = (select sum(number) from iter\_intlist\_to\_table(@Grams));

-- если данная сумма не равна 100, то прекращаем процедуру и откатываем транзакцию

if( @sumgram <> 100)

rollback tran;

else --иначе

begin

-- переменная для помещения значения курсора (граммы)

declare @gr int;

-- переменная для помещения значения курсора (id продукта)

declare @pr int;

--создаем курсор для получения всех грамм отдельными значениями

declare cursgr cursor for

select number

from iter\_intlist\_to\_table(@Grams);

--создаем курсор для получения всех id продуктов отдельными значениями

declare curspr cursor for

select number

from iter\_intlist\_to\_table(@Products);

--открываем курсоры

open cursgr

open curspr

--получаем значения из курсоров в переменные

fetch next from cursgr into @gr

fetch next from curspr into @pr

--пока получаем значения

while @@fetch\_status = 0

begin

--добавляем отношвения между блюдом и продуктами

--связываем блюдо с продуктом питания по id

-- а так же кол-во грамм этого продукта в блюде

insert into RelationDP(ID\_Dish,ID\_Product,Grams)

values (@id\_dish, @pr, @gr)

fetch next from cursgr into @gr

fetch next from curspr into @pr

end --закрытие курсоров и уночтожение курсоров

close curspr;

close cursgr;

DEALLOCATE curspr;

DEALLOCATE cursgr;

--получение нового блюда

select ID\_Dish from Dishes where Name = @Name;

commit tran; -- коммит транзакции

end;

end;

end;

### 2.2.4 Процедура чтения данных из XML

Данная процедура разработана для добавления новых данных в таблицу Product новых данных из xml-файла. Файл разработан заранее и путь к нему передается в качестве параметра данной процедуры.

Листинг:

CREATE PROC insertProductFromXML

@path nvarchar(256) --передаем путь

AS

begin

SET NOCOUNT ON --отклюаем вывод кол-ва обработанных строк

SET XACT\_ABORT ON --откат транзакции в случае ошибки

BEGIN TRAN

declare @results table (x xml) --объявляем таблицу с полем xml

--считывание данные с удаленного источника (xml-файла)

--BULK - загрузка большого набора данных,

--SINGLE\_BLOB - возвращает содержимое в виде 1строки 1столбца бинарных данных

declare @sql nvarchar(300)=

'SELECT

CAST(REPLACE(CAST(x AS VARCHAR(MAX)), ''encoding="utf-16"'', ''encoding="utf-8"'') AS XML)

FROM OPENROWSET(BULK '''+@path+''', SINGLE\_BLOB) AS T(x)';

INSERT INTO @results EXEC (@sql)--выполняем и вставлем во временную таблицу

declare @xml XML = (SELECT TOP 1 x from @results); -- записываем xml в xml-переменную

INSERT INTO Products( Name, Сategory, Calories, Proteins, Fats, Carbohydrates)

SELECT

C3.value('Name[1]', 'varchar(30)') AS Name, --создаем именование

C3.value('Сategory[1]', 'varchar(30)') AS Сategory,

C3.value('Calories[1]', 'float') AS Calories,

C3.value('Proteins[1]', 'float') AS Proteins,

C3.value('Fats[1]', 'float') AS Fats,

C3.value('Carbohydrates[1]', 'float') AS Carbohydrates

FROM @xml.nodes('products/product') AS T3(C3) --форирование результирующих строк из метода nodes

COMMIT;

end;

### 2.2.5 Процедура записи данных в XML

Данная процедура разработана для добавления данных о всех пользователях, зарегистрированных в приложении, xml-файла. Файл определен заранее и путь к нему передается в качестве параметра данной процедуры.

Листинг:

CREATE PROC exportUsersToXML

@path nvarchar(256) --передаем путь

AS

begin

BEGIN TRAN --начало транзакции

--bcp используется для массового копирования данных между экземпляром Microsoft SQL Server

-- и файлом данных в пользовательском формате

--queryout — копирует данные из запроса и используется при массовом копировании данных из запроса

-- -w - преобразует данные в символы UNICODE

-- -T - под пользователем системы (проверка подлинности windows)

declare @sql nvarchar(500)=

'bcp "SELECT UserName, Password, Sex, Age, Growth, Weight, PhysicalStress, Goal, DailyCaloriesNorm FROM '+

' Users where UserName <> ''admin'' FOR XML PATH(''User''), ROOT(''Users'')" queryout "'+

@path+'" -S HELEN -d ControlDiet -w -T ';

-- xp\_cmdshell оболочка над командной строкой

EXEC xp\_cmdshell @sql;

COMMIT;

end;

## 2.3 Привилегии

Для базы данных было разработано два типа пользователей. Один из них разработан для пользования под обычным пользователем, а второй для администраторов. Сперва были разработаны логины для подключения к базе данных.

Листинг:

--connection login to user

--задаем имя пользователя и пароль

CREATE LOGIN [usercd] WITH PASSWORD=N'1111',

--здаем базу данных и язык

DEFAULT\_DATABASE=[ControlDiet], DEFAULT\_LANGUAGE=[Русский],

--указываем политики для истечения время срока пароля и действия пароля

CHECK\_EXPIRATION=OFF, CHECK\_POLICY=ON

GO

ALTER LOGIN [usercd] ENABLE

GO

--connection login to admin

CREATE LOGIN [admincd] WITH PASSWORD=N'admincd',

DEFAULT\_DATABASE=[ControlDiet], DEFAULT\_LANGUAGE=[Русский],

CHECK\_EXPIRATION=OFF, CHECK\_POLICY=ON

GO

ALTER LOGIN [admincd] ENABLE

GO

Далее в рамках нашей базы данных – ControlDiet, создаем пользователей, использующих эти логины входа. После чего выдаем гранты для каждого из пользователей. Для обычного пользователя создается роль, после чего ею нагружается пользователь нашей бд, отвечающий за подключения обычных пользователей. Для администраторов мы даем грант на всю схему.

Листинг:

--создаем пользователей в рамках ControlDiet

CREATE USER [user] FOR LOGIN [usercd];

GO

CREATE USER [admin] FOR LOGIN [admincd];

GO

use ControlDiet;

DROP ROLE [ControlDiet\_user];

--создае роль для пользвоателей

CREATE ROLE [ControlDiet\_user];

--грант на выполнение [SelectUser]

GRANT EXECUTE ON [dbo].[SelectUser] TO [ControlDiet\_user];

--грант на выполнение [AddUser]

GRANT EXECUTE ON [dbo].[AddUser] TO [ControlDiet\_user];

--грант на выполнение [AddMeals]

GRANT EXECUTE ON [dbo].[AddMeals] TO [ControlDiet\_user];

--грант на выполнение [AddFood]

GRANT EXECUTE ON [dbo].[AddFood] TO [ControlDiet\_user];

--грант на выполнение [UpdateUser]

GRANT EXECUTE ON [dbo].[UpdateUser] TO [ControlDiet\_user];

--грант на выполнение [AddHistory]

GRANT EXECUTE ON [dbo].[AddHistory] TO [ControlDiet\_user];

--присваем данную роль пользователю

sp\_addrolemember 'ControlDiet\_user', 'user';

-- выдаем администарторам всю схему

GRANT EXECUTE ON SCHEMA::[dbo] TO [admin]

## 2.4 Резервное копирование

Компонент резервного копирования и восстановления SQL Server обеспечивает необходимую защиту важных данных, которые хранятся в базах данных SQL Server. Чтобы минимизировать риск необратимой потери данных, необходимо регулярно создавать резервные копии баз данных, в которых будут сохраняться производимые изменения данных. Резервная копия может использоваться для восстановления данных в случае возникновения ошибки, а также для восстановления копии базы данных в новом расположении.

Для создания регулярного резервного копирования можно использовать встроенные в SQL Server планировщик заданий - «SQL Server Agent». Данная служба позволяет автоматизировать некоторые административные задачи.

Для созданий расписания резервного копирования базы данных необходимо выполнить следующие шаги:

1. Запустить утилиту SQL Server Management Studio и подключиться к серверу под учетной записью администратора.

В левой части окна нажать правой кнопкой мыши на разделе «Объекты сервера/Устройства резервного копирования» и в контекстном меню выбрать пункт «Создать устройство резервного копирования», что вы можете увидеть на рисунке 2.2:

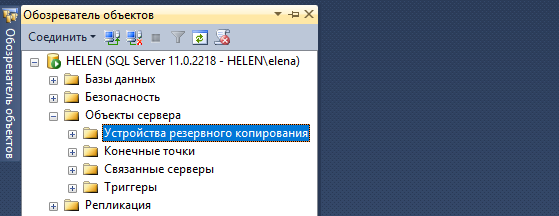


Рисунок 2.2 – Создание устройства резервного копирования

1. В поле «Имя устройства» ввести имя, которое будет ассоциироваться с файлом резервной копии БД, при необходимости изменить путь в поле «Файл» и нажать «ОК», пример на рисунке 2.3:

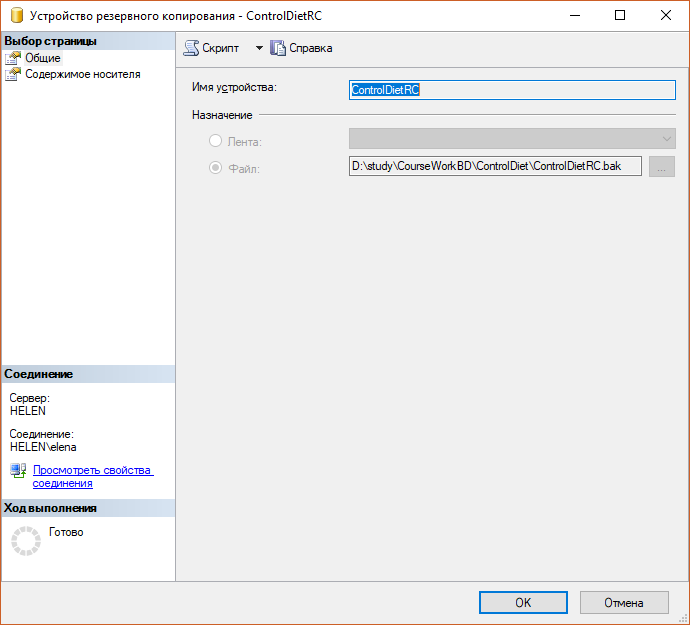


Рисунок 2.3 – Определение имени устройства

1. В левой части окна нажать правой кнопкой мыши на разделе «Агент SQL Server/Задания» и в контекстном меню выбрать пункт «Создать задание». В появившемся окне в поле «Имя» ввести имя задания, как на рисунке 2.4:

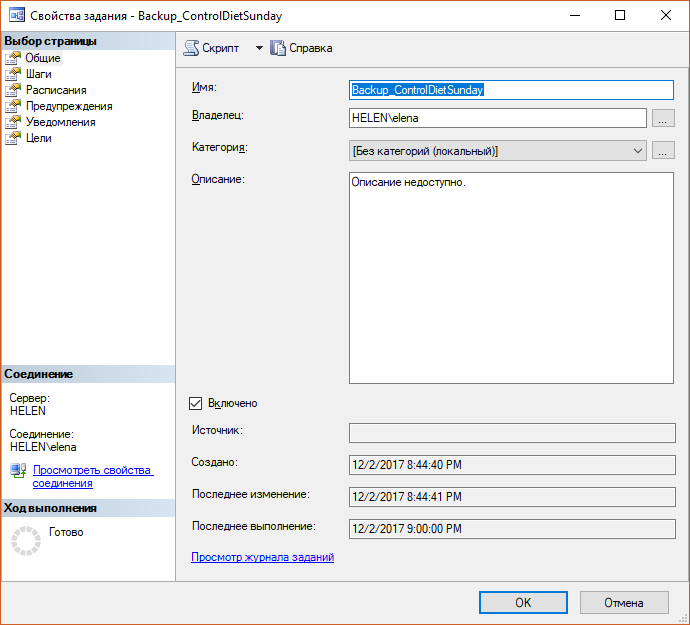


Рисунок 2.4 – Определение задания

1. На странице «Шаги» нажать кнопку «Создать». В появившемся окне ввести имя в поле «Имя шага», проверить, что в поле «Тип» выбрано «Сценарий Transact-SQL (T-SQL)», а в поле «Команда» ввести строку: «BACKUP DATABASE ControlDiet TO ControlDietRC WITH INIT, NOFORMAT, SKIP, NOUNLOAD», где «ControlDiet» – имя базы данных, «ControlDietRC» – имя устройства резервного копирования, созданного ранее, представлено на рисунке 2.5:

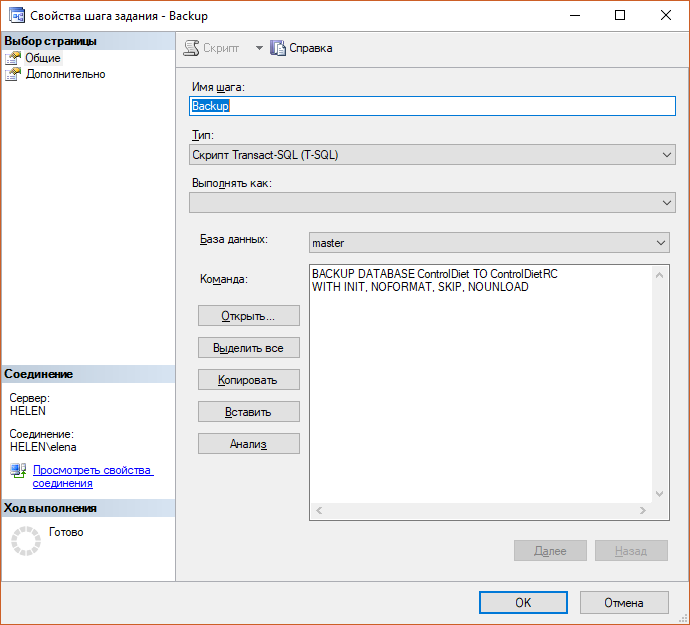


Рисунок 2.5 – Определение шага

1. На странице «Расписания» нажать кнопку «Создать». Ввести имя в поле «Имя», проверить, что в поле «Тип расписания» выбрано значение «Повторяющееся задание», а в поле «Выполняется» – «Еженедельно». Поставить галочку возле нужного дня недели (остальные снять), а в поле «Однократное задание» указать время, когда должен запускаться процесс резервного копирования (обычно это делается ночью), задание расписание представлено на рисунке 2.6:

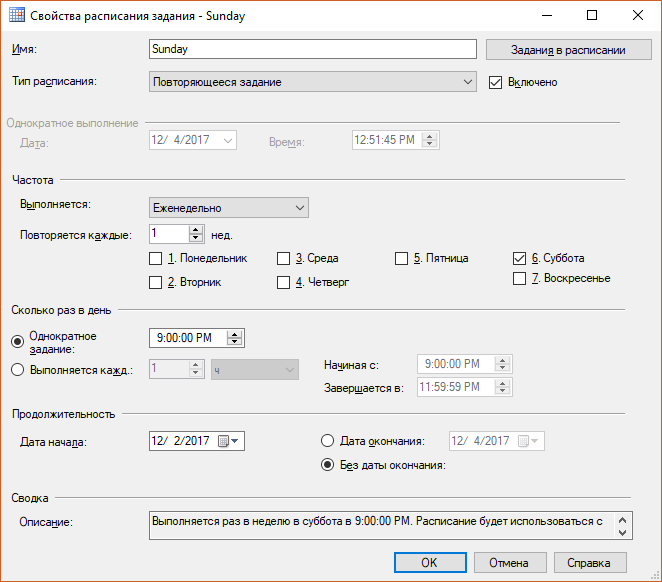


Рисунок 2.6 – Определение расписания

Нажать «ОК». Расписание создано.

Восстановление базы данных это многоэтапный процесс, в ходе которого все данные и страницы журнала копируются из указанной резервной копии SQL Server в определенную базу данных, а затем выполняется накат всех фиксированных транзакций, записанных в резервной копии журнала, путем внесения новых данных на основе зарегистрированных изменений.

Восстановление базы данных по резервной копии осуществляется следующим образом:

1. Запустить утилиту SQL Server Management Studio и подключиться к серверу под учетной записью администратора.

Раскройте узел Базы данных и выбирается нужная база данных. Щелкнув провой кнопкой мыши по пункту «Задачи» выбирается «Восстановить», далее «Базы данных». После откроется окно для восстановления базы дынных, представлено на рисунке 2.7:

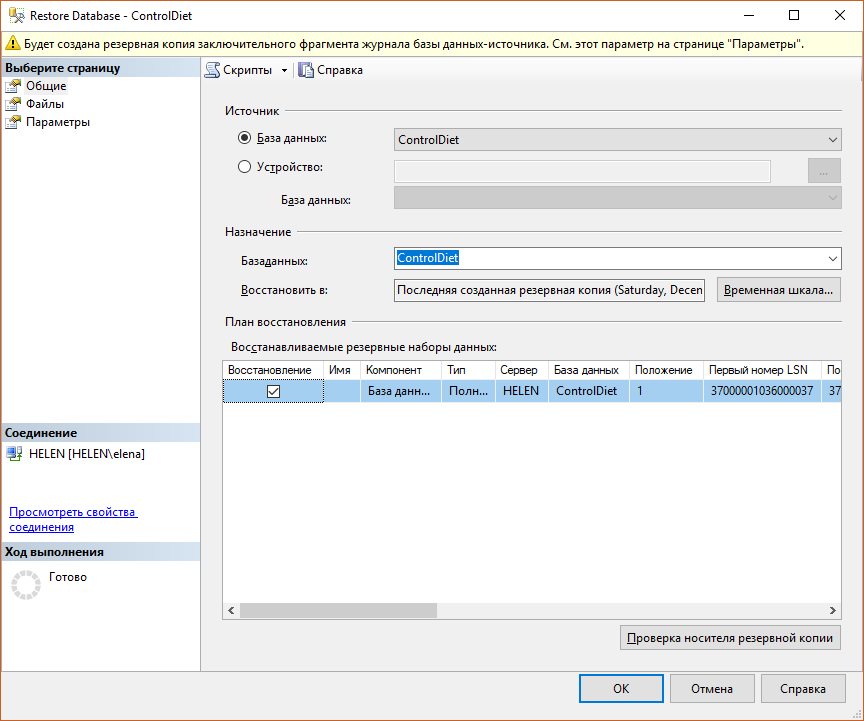


Рисунок 2.7 – Окно восстановления

Так как база данных ControlDiet не была потеряна в результате ошибки, сбоя, в примере приводится восстановление старой копии базы данных, перезапись данных.

1. На вкладке «Параметры» в «Параметрах восстановления» выбираем «Перезаписать существующую базы данных (WITH REPLACE)». И убираем галочку с «Создать резервную копию заключительного фрагмента журнала до восстановления».

После выполнения данных шагов на вкладке «Общее» нажимаем «Ок». В результате появляется окно, представленное на рисунке 2.8, и база данных считается откатившейся до состояния последней созданной резервной копии.

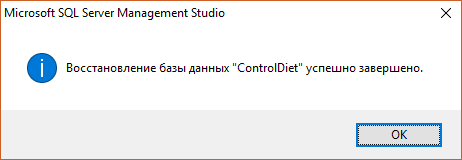


Рисунок 2.8 – Окно успешного восстановления

Базу данных и резервные копии следует размещать на разных устройствах. Чтобы в случае сбоя одного из устройств, второе осталось доступно.

# 3. Технология SQL Server Reporting Services (SSRS)

SQL Server Reporting Services (SSRS) – это службы для разработки, построения, доставки и просмотра отчетов. С помощью этих служб можно создавать табличные, интерактивные, графические и другие более сложные отчеты с использованием диаграмм и других отчетных элементов.

Данная служба была выбрана в качестве технологии для данной работы по причине того, что разрабатываемая база данных хранит статистические данные об изменении пользователей, их приемах питания, по которым удобнее всего создавать отчеты, чтобы просматривать процессы изменения, статистические данные, информацию о конкретных объектах и ее преобразование и обработку для более приятного и удобного восприятия.

На примере создания отчета, содержащего информацию о выбранном блюде, а именно: о его составе и соотношении белков, жиров и углеводов, будет разобрана работа с SSRS.

Данный отчет формируетсяя по переданному параметру: имени блюда. Далее будут описаны шаги созданий отчета «ReportDishByName»:

1. Для работы с Reporting Services необходимо в ссылки проекта подключить « Microsoft.ReportViewer.WinForms», после чего в оперделении окна MainWindow добавить:

- xmlns:rv="clr-namespace:Microsoft.Reporting.WinForms;assembly= Microsoft.ReportViewer.WinForms"

После чего в XAML создать объект WindowsFormsHost, в котором будет отображаться отчет.

1. Далее создаем событие на нажатие кнопки, для отправления имени блюда, веденного до этого в TextBox. Создадим функцию, которая будет возвращать нужный нам, для отчета, набор данных. Данная функция представлена на рисунке 3.1.

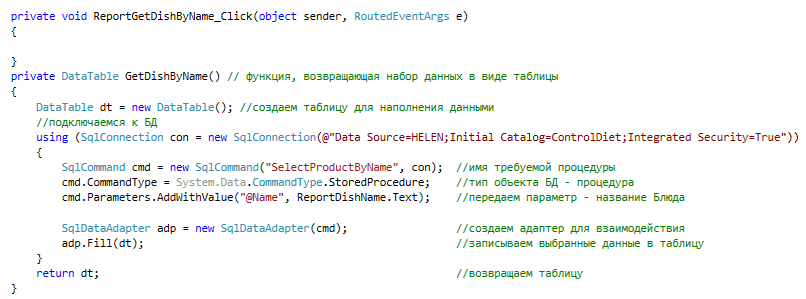


Рисунок 3.1 – Функция получения данных из БД

1. Далее создадим в нашем проекте набор данных под именем «DataSetForDishByName». В нем создаем объект DataTable, под именем «DataTableForDishByName», для хранения результирующего набора. Далее требуется добавить столбцы, они должны соответствовать результирующему набору полностью, а именно: имена столбцов результирующего набора соответствуют именам дстолбцов данной таблицы, а также типы данных. В итоге выходит DataSet, представленный на рисунке 3.2:

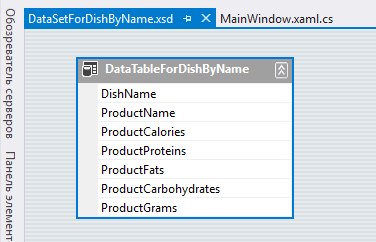


Рисунок 3.2 – DataSet для хранения результирующего набора

1. После этого в проект добавляем новый отчет, называя его «RoportForDishByName». В появившемся пустом отчете нажимаем правой клавишей мыши «Вставить -> Таблица». В появившемся окне задаем имя и выбираем наш источник данных, как изображено на рисунке 3.3:

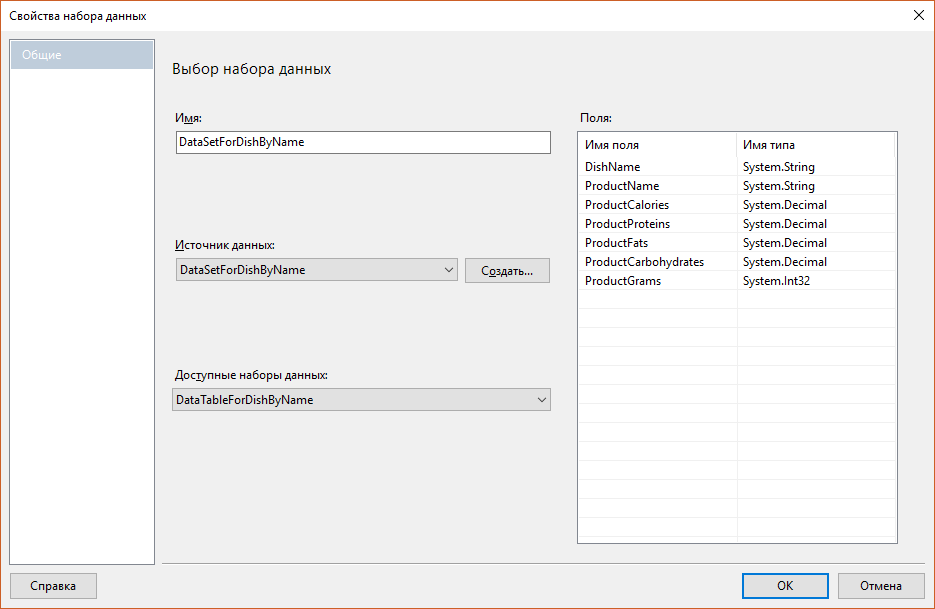


Рисунок 3.3 – Выбор набора данных

Далее нажимаем «ОК». Появляется пустая таблица, мы сами определяем данные, которые будут заноситься в каждый столбец.

Получается таблица, приведенная на рисунке 3.4:

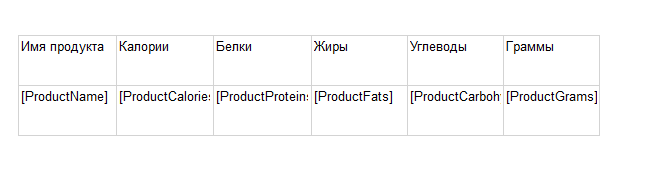


Рисунок 3.4 – Таблица

1. Далее форматируем отчет по нашей нужде, добавляем стили, диаграмму. Вставляем текстовое поле, нажимаем на него правой клавише мыши и выбираем «Выражение». В данном окошке в качестве выражения задаем следующее значение: «=First(Fields!DishName.Value, "DataSetForDishByName")», которое означает, что мы берем первое значение из столбца набора данных с именем блюда.

После этого к таблице мы добавляем еще одну строку, в которой будем подводить итоги. Под столбцом «Калории» в этой стоке нажимаем правой клавишей мыши на ячейку и выбираем пункт «Выражение», в которое вставляем следующее: «=Sum(Fields!ProductCalories.Value)», что означает подвести сумму по значениям столбца «Калории». По аналогии выполняем это с другими столбцами. После этих манипуляций наш отчет выглядит следующим образом, рисунок 3.5.

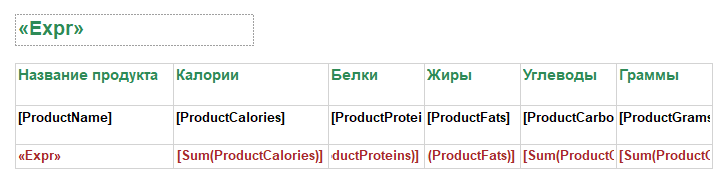


Рисунок 3.5 – Таблица отчета и заголовок

Далее нажимаем на незаполненное место в отчете правой кнопкой мыши и выбираем «Вставить -> Диаграмма». Выбираем тип диаграммы, и она отображается в нашем отчете. Задаем ее название. После этого требуется определить данные, которые будут отображаться на ней. Нажимаем на диаграмму и у нас появляется небольшое окошко под названием «Данные диаграммы». Т.к. мы хотим отобразить в этой диаграмме процентное содержание белков, жиров и углеводов в продукте, то возле надписи «Значения» нажимаем «плюсик» и выбираем поле, значения которого должны отображаться в нашей диаграмме. По дефолту это значение является суммой набора данных выбранного столбца. Так добавляем белки, жиры и углеводы. Это можно увидеть на рисунке 3.6.



Рисунок 3.6 – Созданная диаграмма

Далее требует отформатировать вывод данных. На каждом из созданных значений нажимаем правой клавишей мыши и выбираем «Отобразить метки данных». Теперь по очереди выбираем каждое из значений и заходим в свойства, где меняем Метки -> Lable ->Label на «#PERCENT{P0}», для того, чтобы данные на диаграмме отображались в процентном виде. А так же на каждом из созданных значений выбираем «Свойства ряда…-> Условные обозначения» и там в качестве пользовательского текста вводим желаемое отображение названия категории, например «Белки», вместо «ProductProteins».

1. Последним этапом является добавление данного отчета в WPF приложение. Для этого используется функция, изображенная на рисунке 3.7:

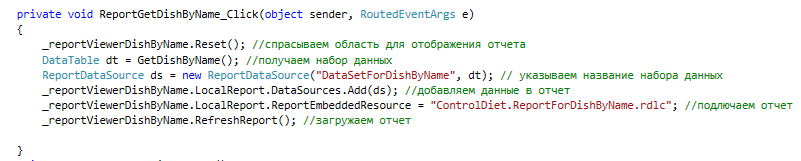


Рисунок 3.7 – Функция загрузки отчета

Итогом работы является отчет, представленный на рисунке 3.8:

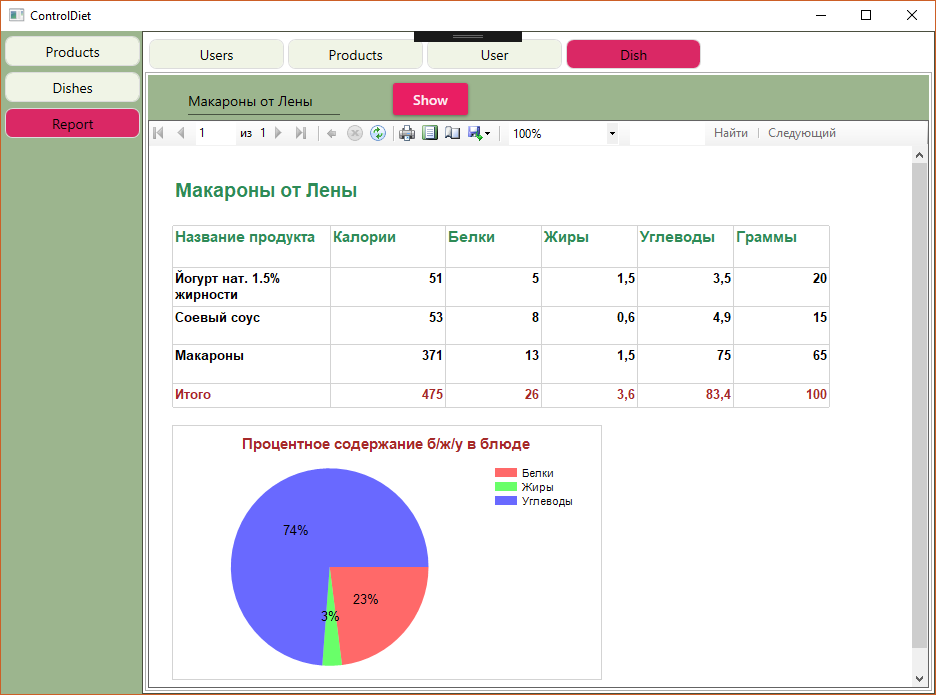


Рисунок 3.8 – Полученный отчет

Далее будут приведены отчеты, созданные в рамках данной курсовой работы, рисунок 3.9 – 3.15.

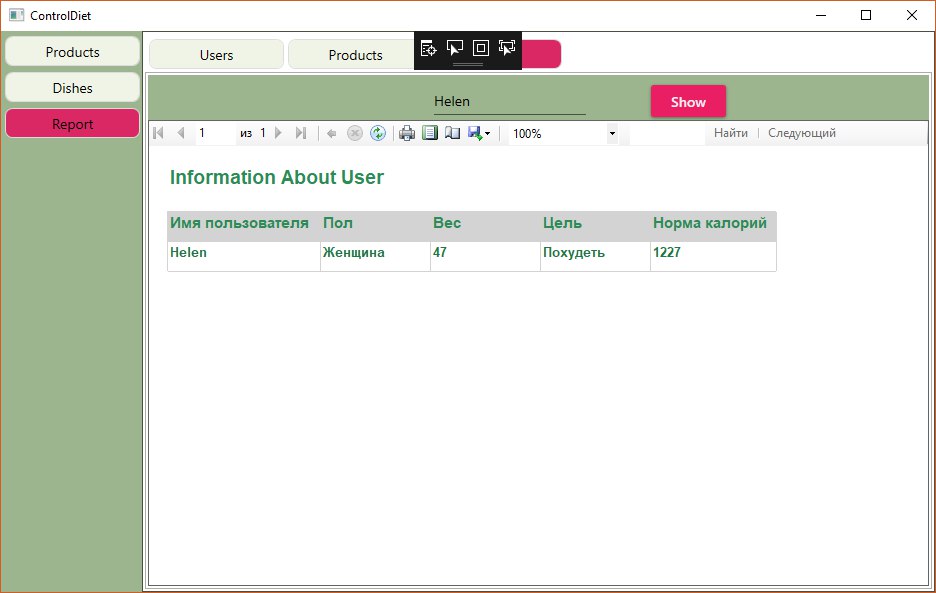


Рисунок 3.9 – Информация о выбранном пользователе

Для получения данного отчета требуется передать имя пользователя, по данному параметру осуществляется выборка данных из таблицы и составляется отчет в виде таблицы с информацией о данном пользователе. Поля данной таблицы проходят некоторую обработку для корректного отображения и понимания данных из таблиц.

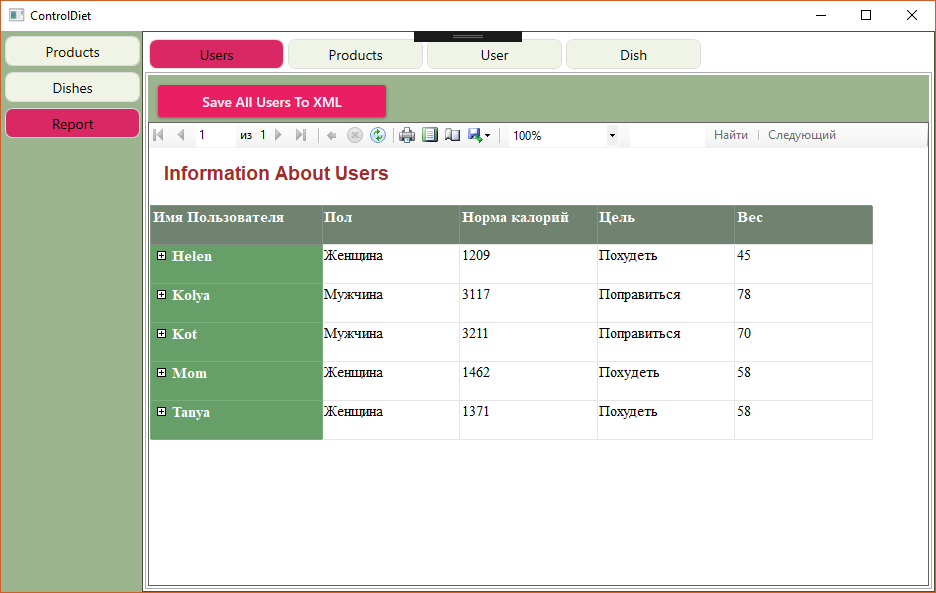


Рисунок 3.10 – Информация о всех пользователях

Данный отчет содержит общую информацию о текущем состоянии всех пользователей.

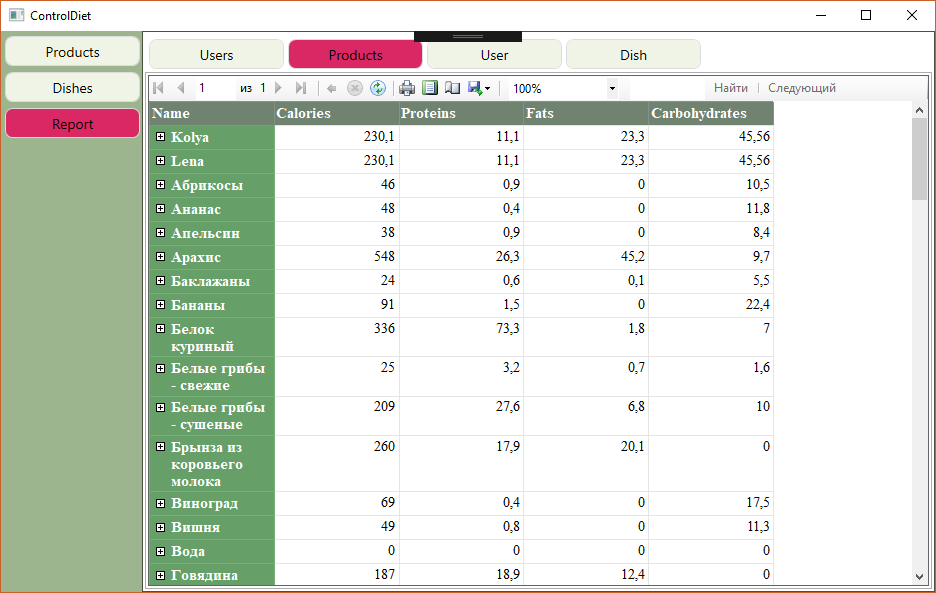


Рисунок 3.11 – Информация о всех пользователях

Отчет содержит информацию о всех продуктах питания.

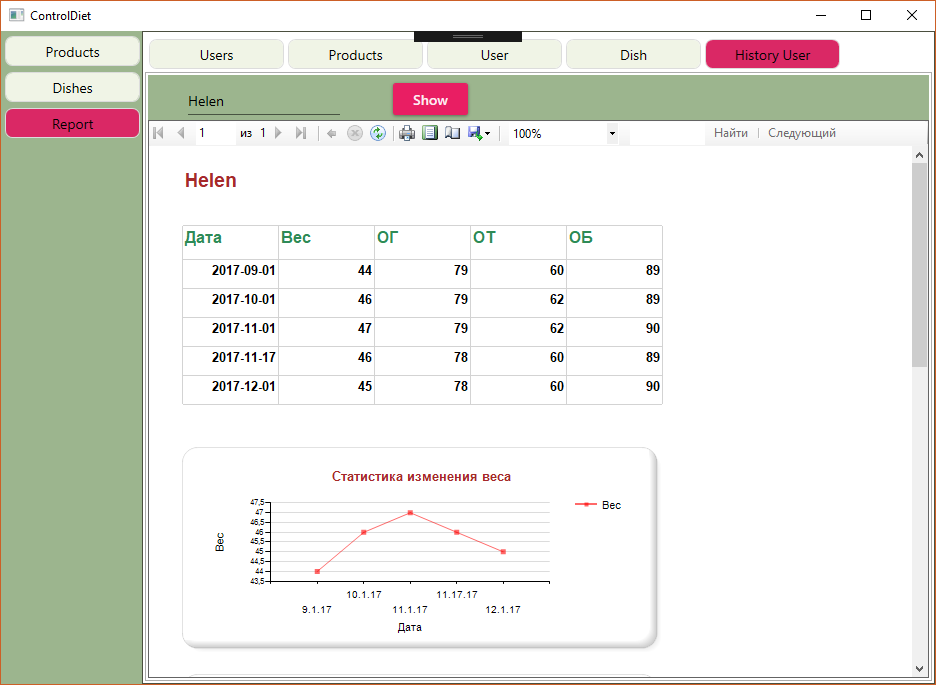


Рисунок 3.12 – Статистика изменения параметров пользователей часть 1

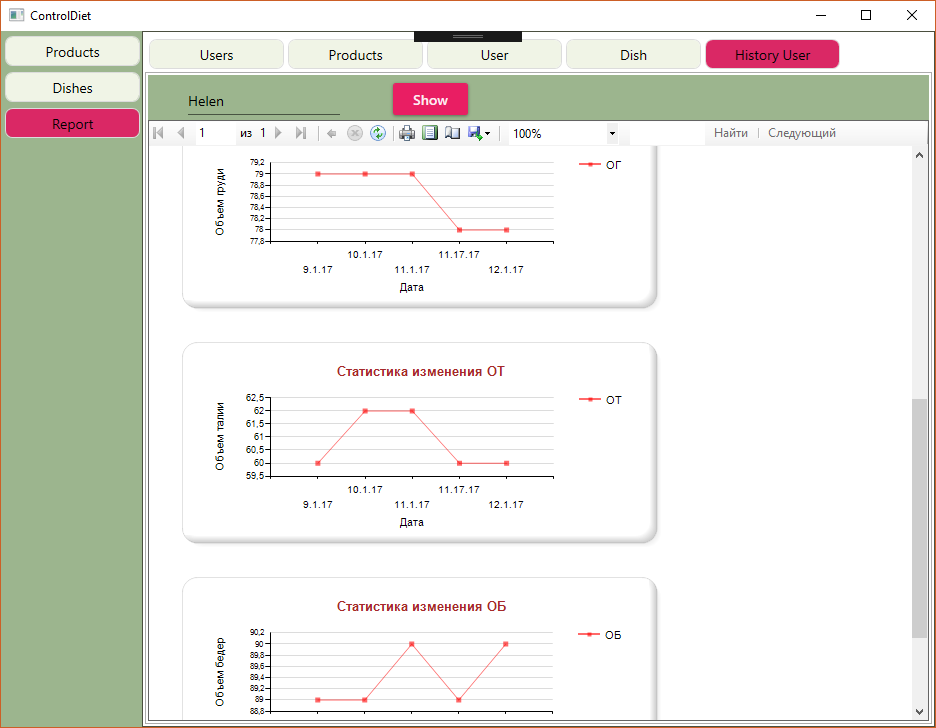


Рисунок 3.13 – Статистика изменения параметров пользователей часть 2

Благодаря данному отчету можно проследить статистику изменения веса пользователя, его объемов груди, талии и бедер, вносимых в базу данных.

Далее следует отчет, которые действует следующим образом: для получения анализа питания пользователя, передается его имя и дата дня, анализ которого следует провести. После чего осуществляется поиск из таблицы Audit нормы калорий в день, наиболее близкий к заданному. Так же из осуществляется поиск всех употреблённых продуктов, их количества, калорийности и б/ж/у в заданный день.

После получения этих данных в отчете производится их анализ. По норме калорий высчитываются нормальные количества грамм белков, жиров и углеводов на тот период, а также подводится итог на заданный день, а именно: суммируется вся калорийность и все б/ж/у. Далее, для наглядного примера, производится построение диаграмм для наглядного отображения в разнице требуемых калорий и б/ж/у к употребленным. Пример отчета представлен на рисунках 3.14 – 3.15.

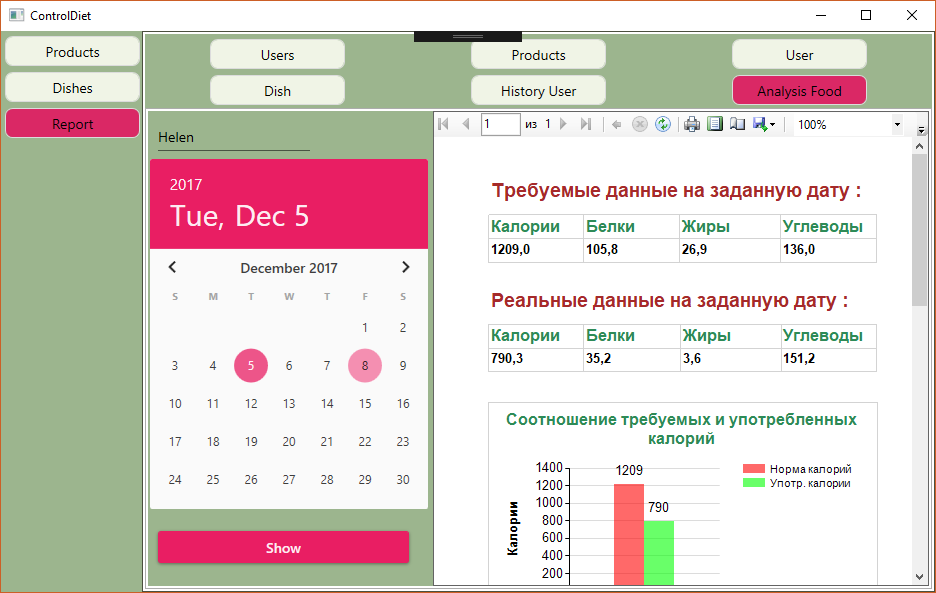


Рисунок 3.14 – Анализ питания часть 1

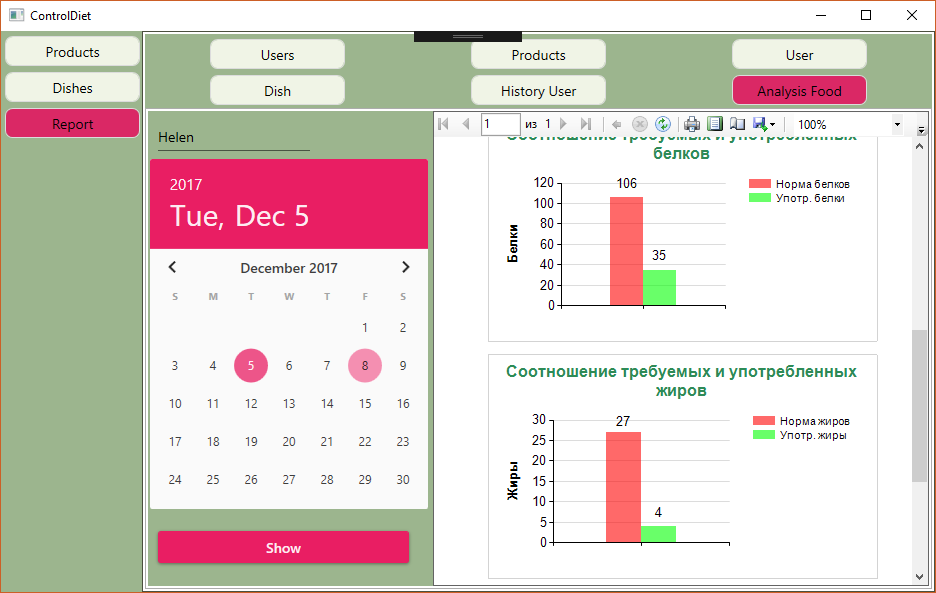
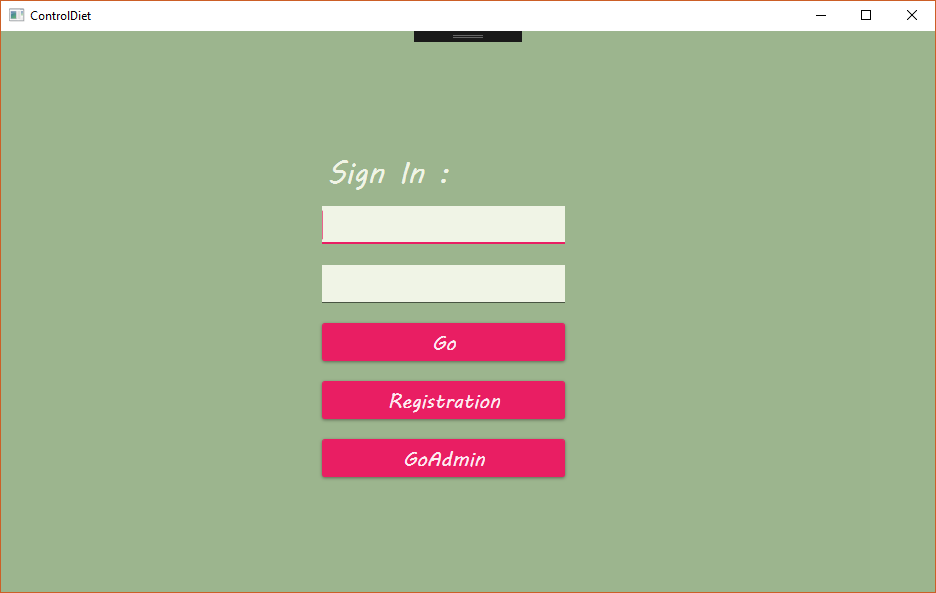
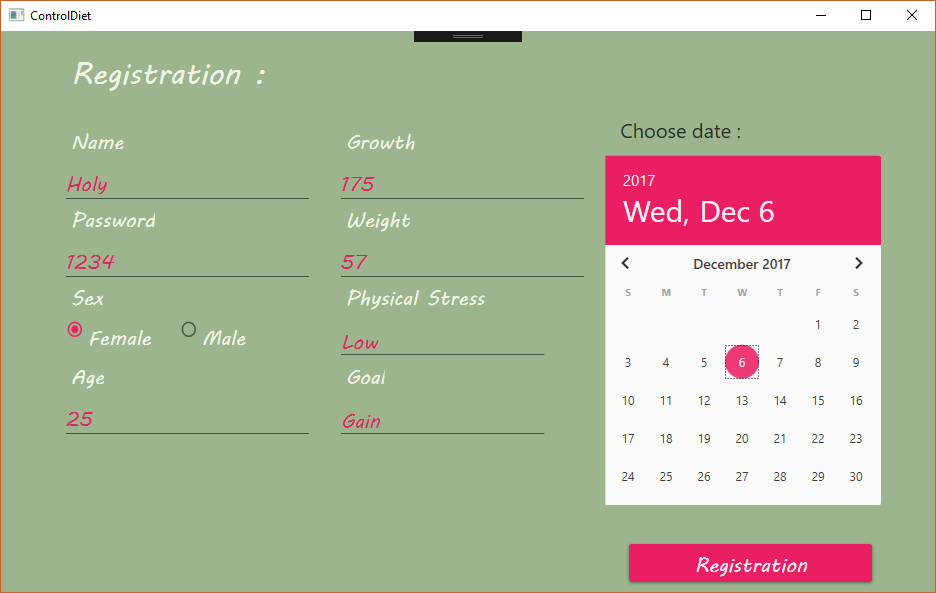


Рисунок 3.15 – Анализ питания часть 2

Технология SSRS оказалась очень удобной для отображения различных данных, проведения анализа данных и последующего его представления. В рамках темы курсового проекта она обеспечила наиболее важную функционала – анализ режима питания пользователей.

# Руководство пользователя

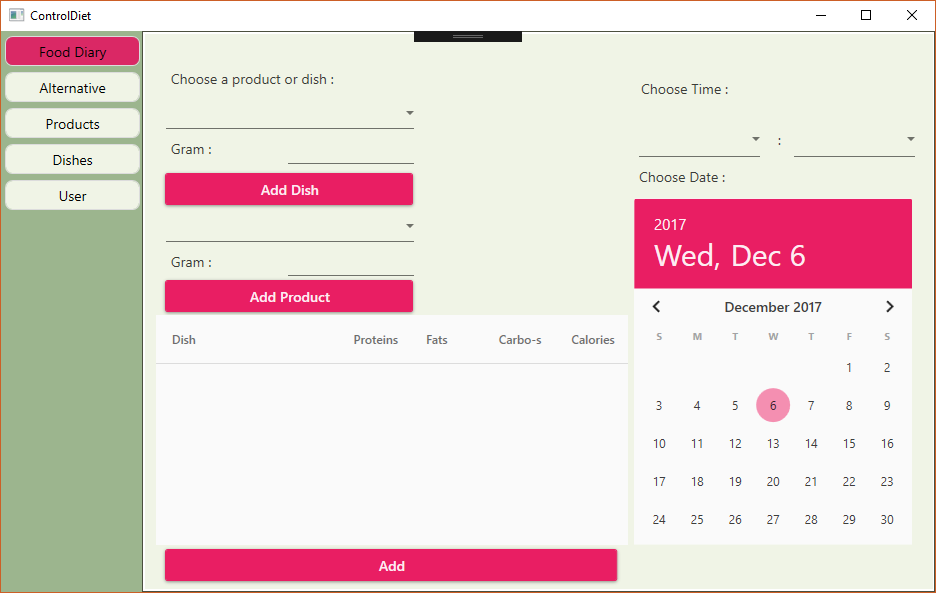
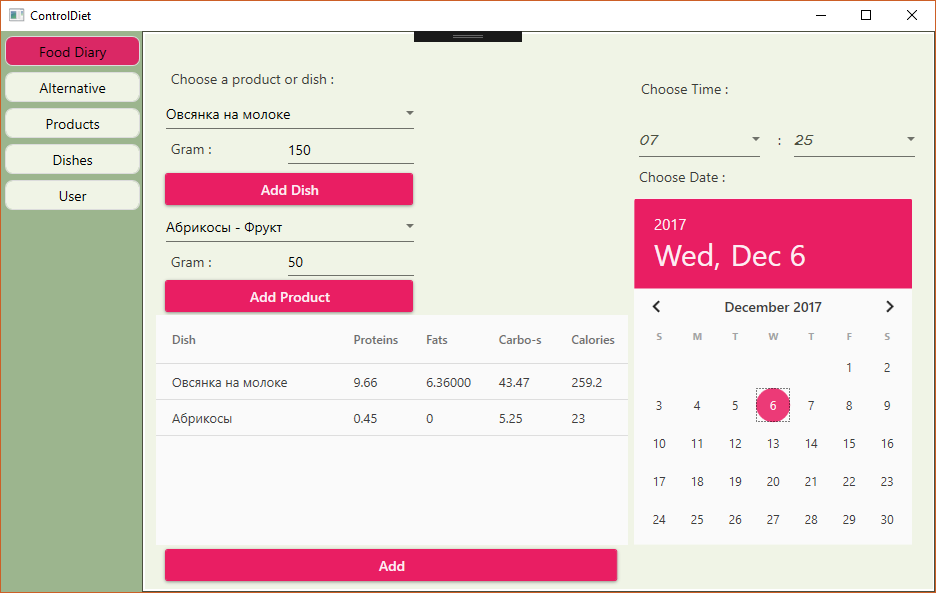
Начальной страницей приложения является вход в приложение. Здесь пользователь может ввести имя и пароль и войти в приложение, перейти к регистрации, либо войти администратор. Данная страница представлена на рисунке 4.1а, а страница регистрации на рисунке 4.1б.

а – Начальная страница б – Страница регистрации

Рисунок 4.1

После выхода, пользователь попадает на вкладку добавления приема пищи. Здесь он может составить из блюд и продуктов питания свой прием пищи и добавить его в базу данных, указав так же дату и время. На рисунках 4.2а и 4.2б представлен пример пользования данной вкладкой.

а – Пример пустой страницы б – Страница, после наполнения данными

Рисунок 4.2

Далее следует вкладка «Alternative». На данной странице пользователь может найти альтернативы выбранному продукту питания. Альтернативы ищется на основании категории продукта, а также калорийности. Пример поиска на данной вкладке представлен на рисунке 4.3.

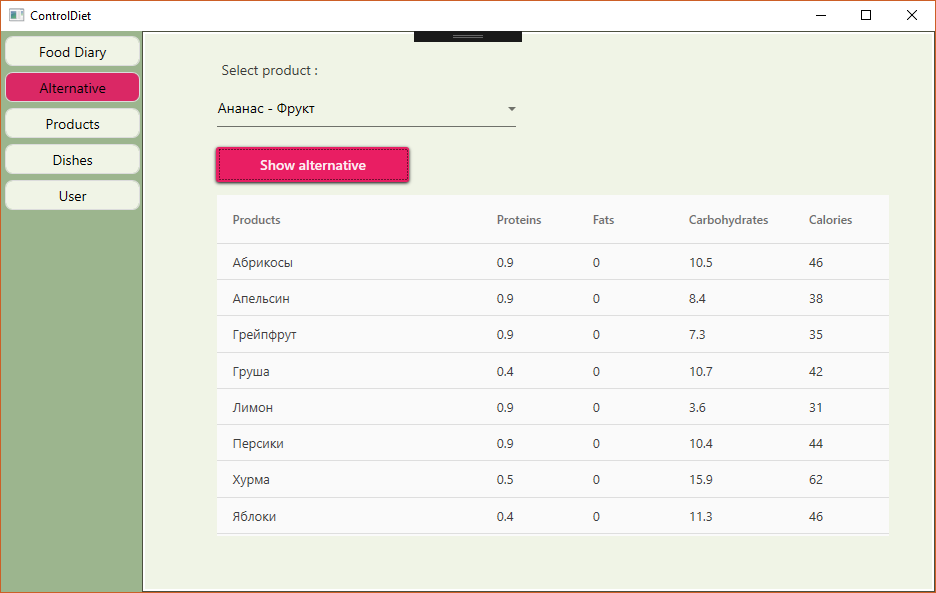
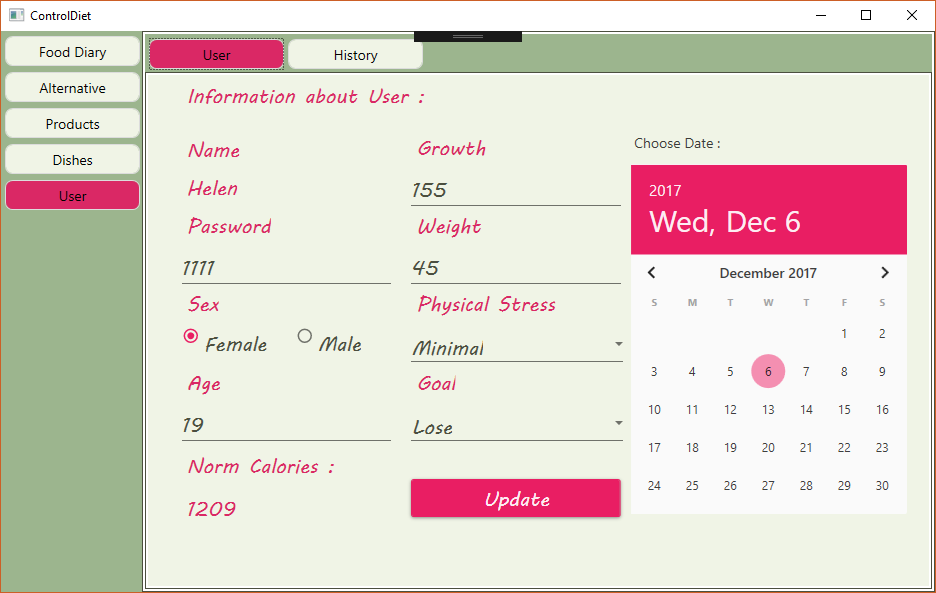
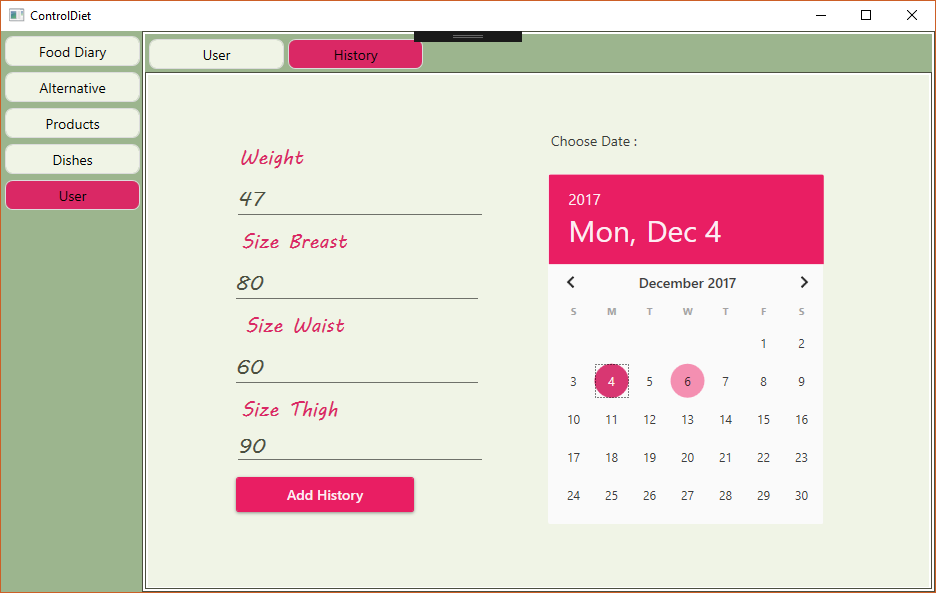


Рисунок 4.3 – Пример поиска на вкладке

Далее следую вкладки «Products» и «Dishes», на которых можно просмотреть информацию о продуктах питания и блюдах, выбрав их из списка.

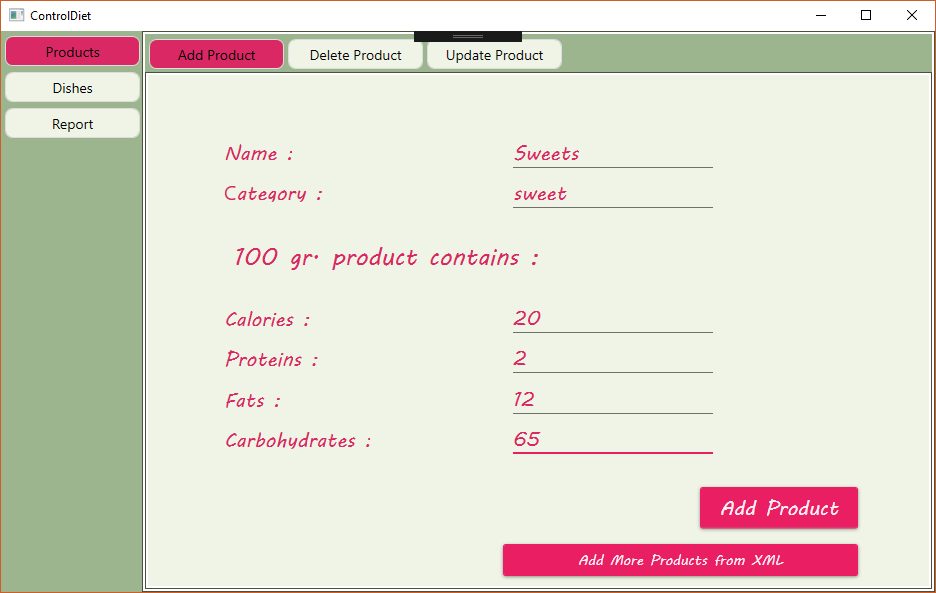
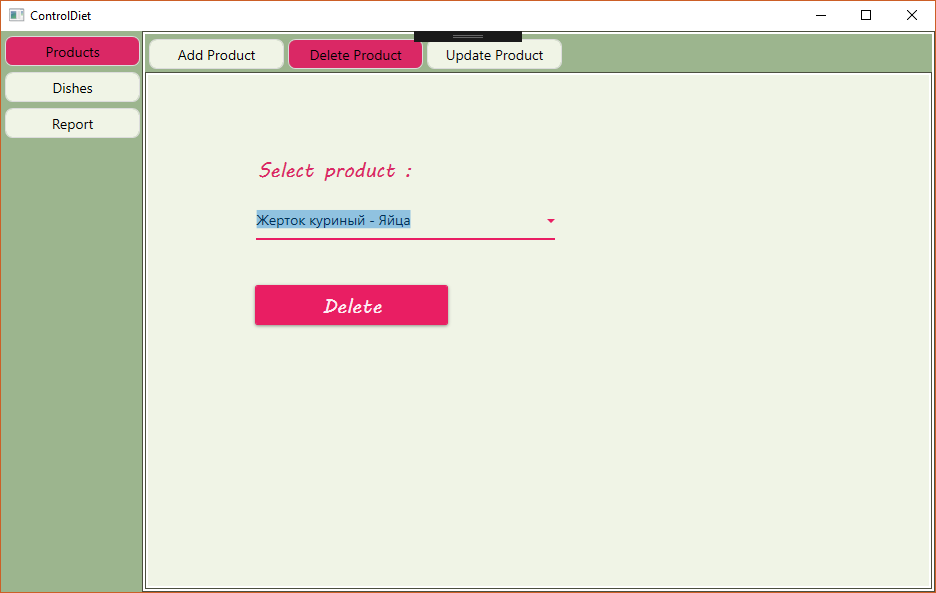
На вкладке «User» пользователь может просмотреть информацию о себе, а также изменить ее. Также там он может добавлять историю изменения своих параметров: веса, объемов груди, талии и бедер, указав день. Пример данной вкладки представлен на рисунках 4.4а и 4.4б.

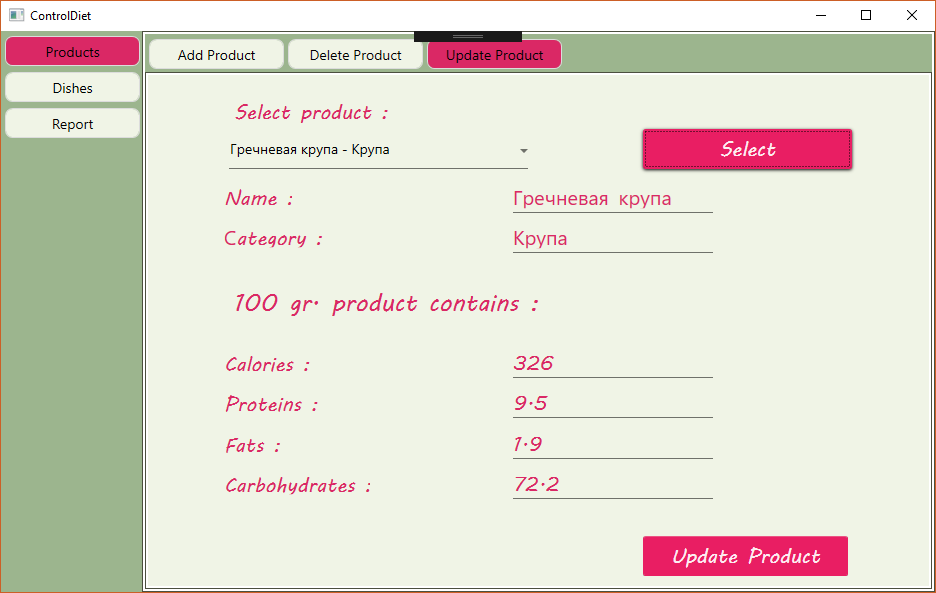
а – Данные пользователя б – История пользователя

Рисунок 4.4

Далее рассмотрим функционал, представленный для администратора приложения. Для того, чтобы его просмотреть, требуется при входе указать данные администратора и нажать кнопку «GoAdmin». Первая из вкладок предоставляет нам возможность работы с продуктами. На рисунках 4.5.а, 4.5б и 4.5в представлены возможности, такие как редактирование, удаление и добавление продуктов.

а – Добавление продукта б – Удаление продукта



в – Редактирование продукта

Рисунок 4.5

По аналогии с продуктами происходит работа с блюдами, а именно их удаление и добавление.

Также существует такая вкладка, как «Report», в ней представлена работа с технологией Reporting Services, описанной выше.

# Тестирование

Для тестирования базы данных были проведены запросы на основе разного объема набора данных, для определения разницы времени в обработке запроса. На объеме данных в 87 строк общий select-запрос на выборку всех данных из таблицы затратил ресурсы, представленные на рисунке 5.1.

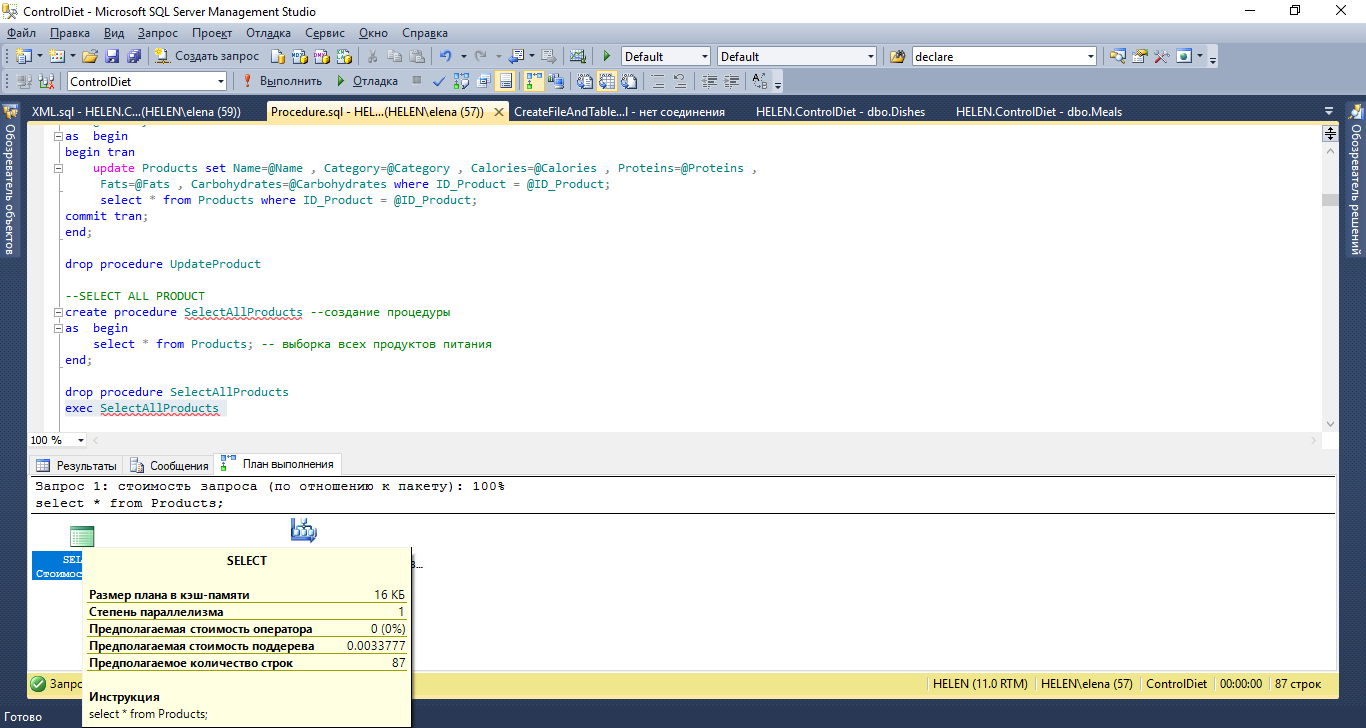


Рисунок 5.1 – Результат select-запроса с 87 строками

После добавления в таблицу Product еще порядка 10 000 данных, результаты уже оказались гораздо ниже, нежели при 87. Результаты такого-же запроса, но к 10 000 данных представлены на рисунке 5.2.

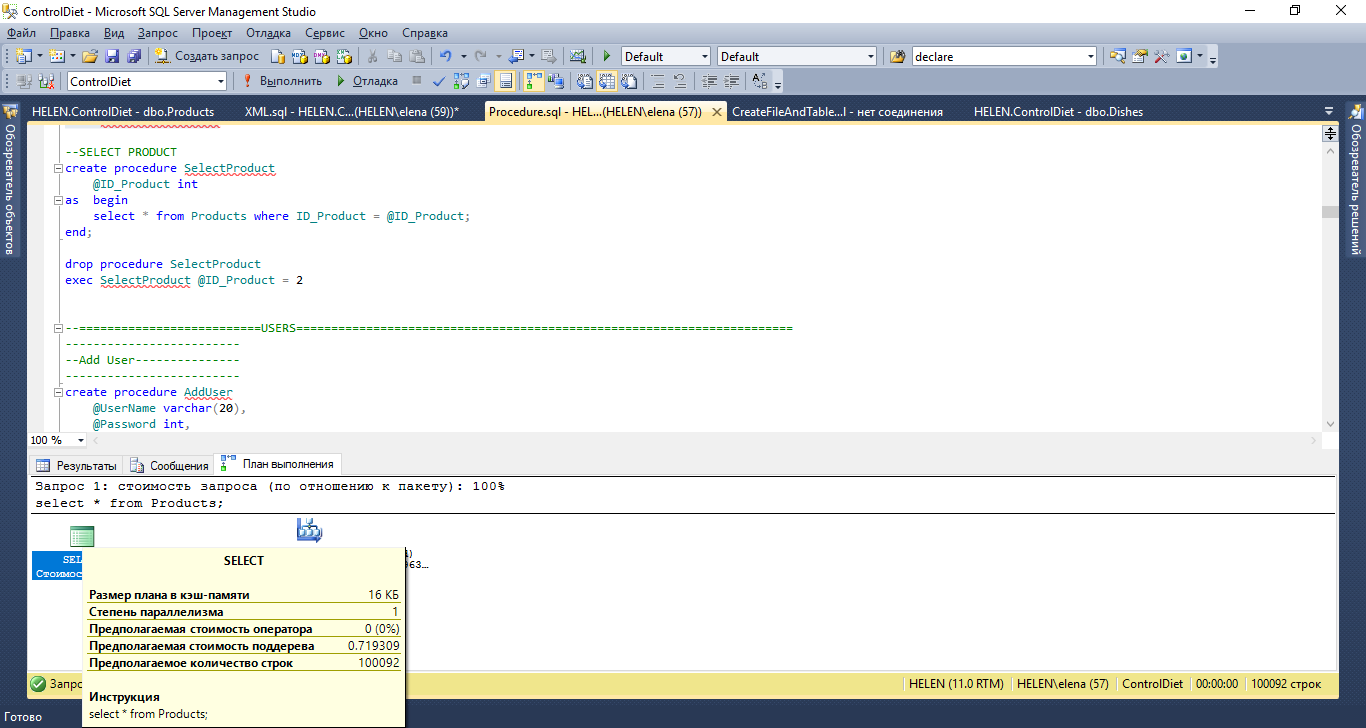


Рисунок 5.2 – Результат select-запроса с 10 000 строками

Далее проведем тестирования select-запроса на выборку имени и категории продуктов из 10 000 строк данных. Результаты представлены на рисунке 5.3.

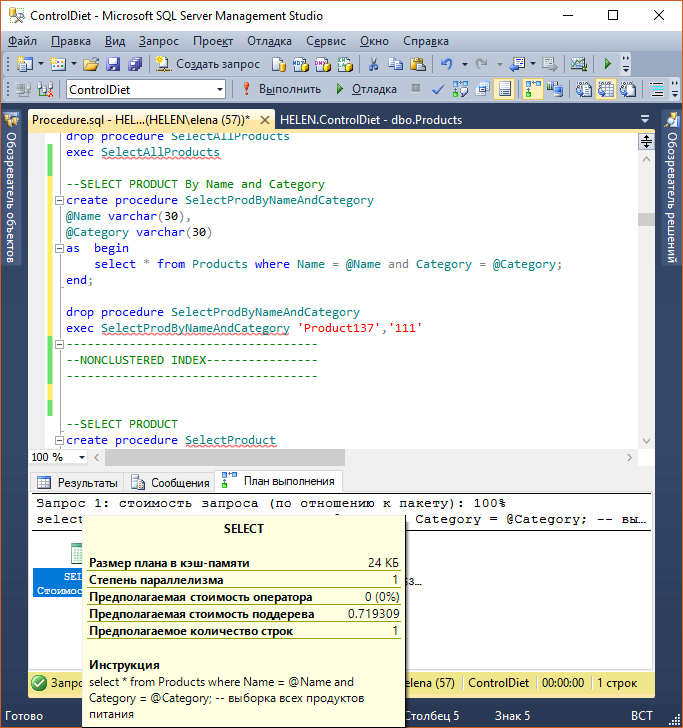


Рисунок 5.3 – Результат select-запроса

Здесь мы можем увидеть, что разницы в затраченном времени выборки по двум полям из таблицы продуктов или по всем – нет. Для оптимизации данного запроса был разработан некластеризованный индекс по таблице Products, поля - Name и Category. Благодаря данному индексу стоимость данного запроса уменьшила в десятки раз. Данное утверждение подтверждается рисунком 5.4.

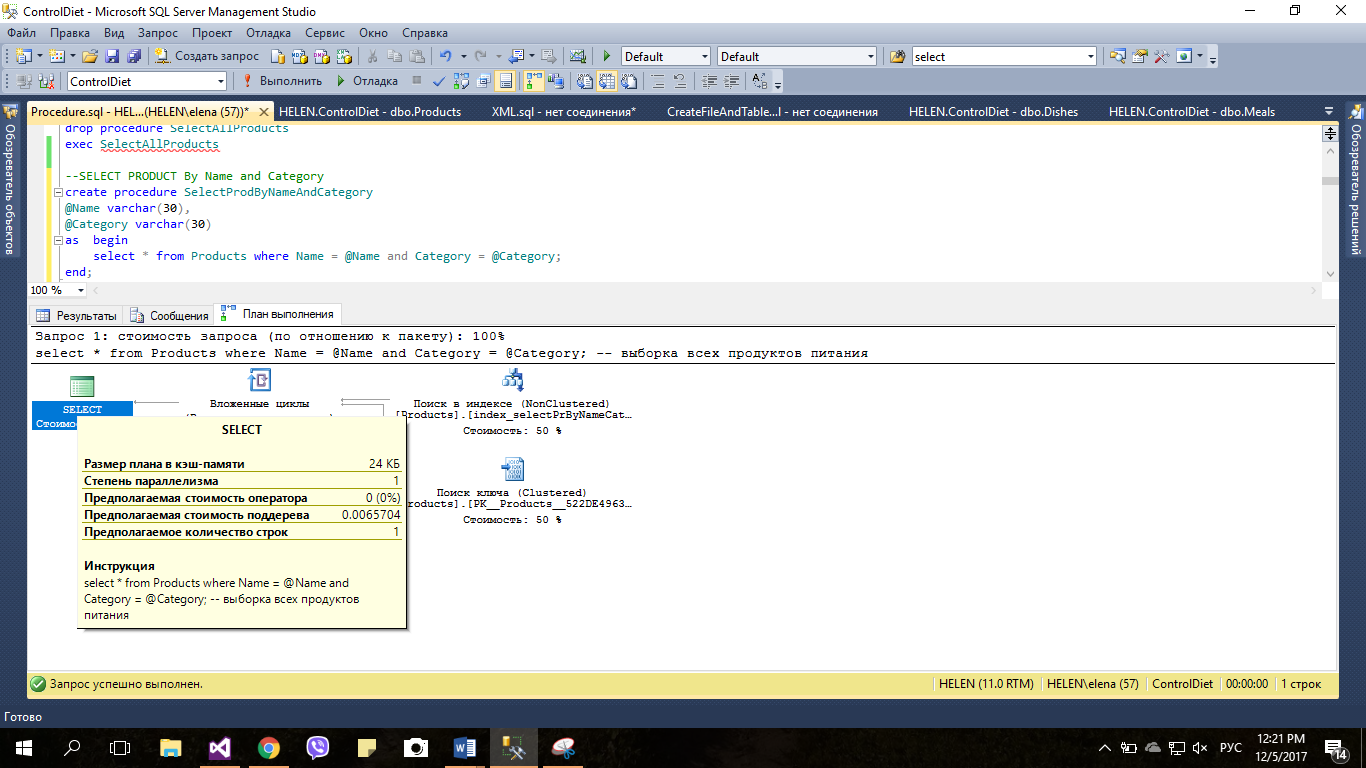


Рисунок 5.4 – Результат select-запроса с индексом

Так же в плане выполнения запроса мы можем увидеть, как распределилась стоимость запроса с использованием некластеризованного индекса, рисунок 5.5.

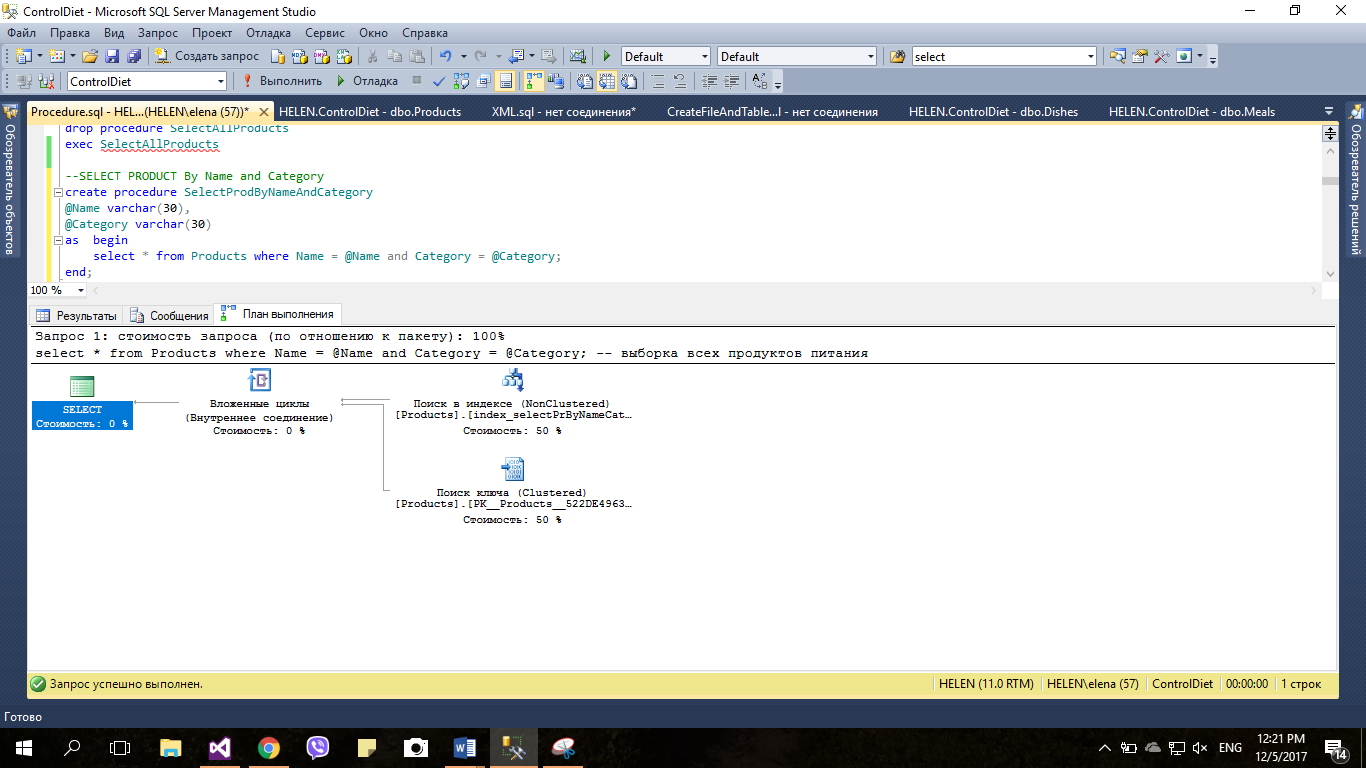


Рисунок 5.5 – План выполнения запроса

В ходе тестирования приложения были рассмотрены ситуации, при которых пользователь мог ввести данные неверно. Примером подобного действия будет неверный ввод имени и пароля для пользователя или админа. Предупреждение, возникающее на это действие, будет выглядеть следующим образом (рисунок 5.6).

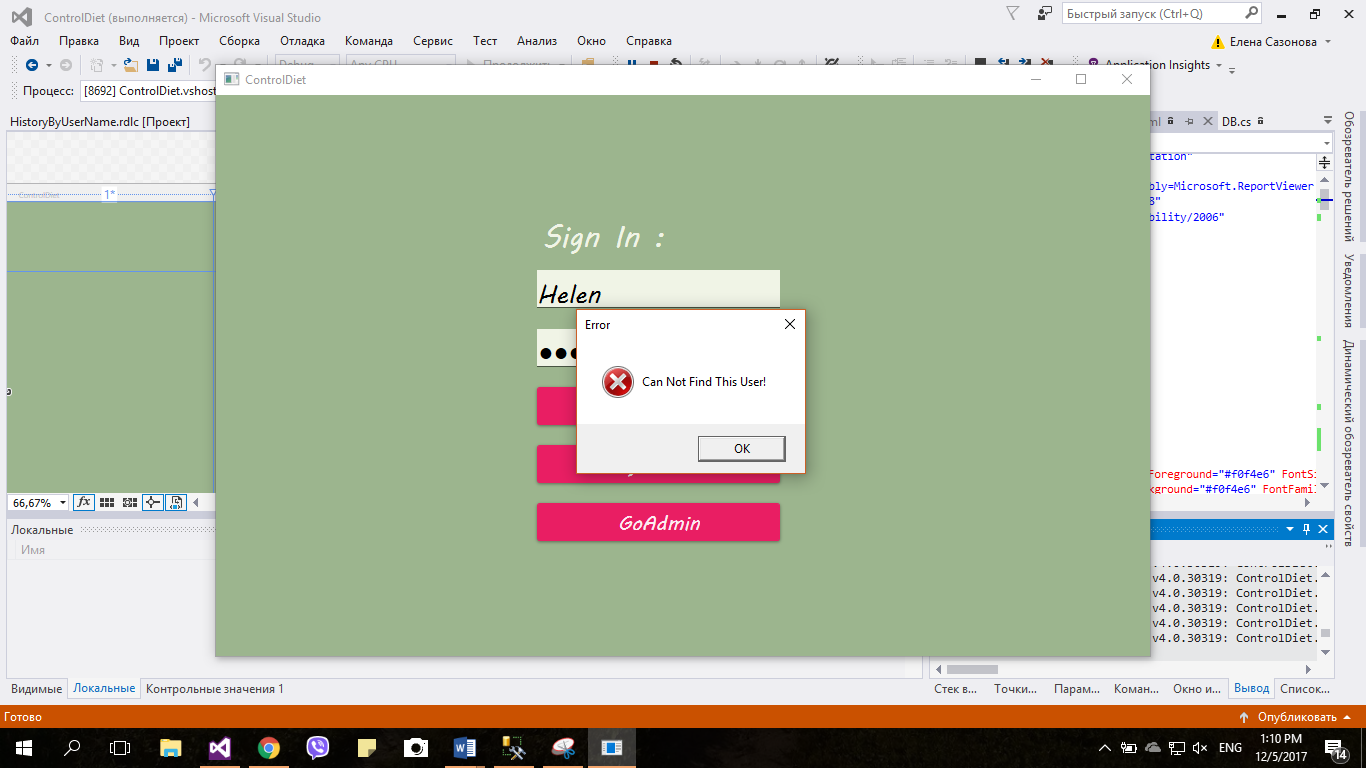


Рисунок 5.6 – Предупреждение при неверном входе

Аналогичным образом пользователь может получать предупреждения о неверном вводе данных и в других частях приложения.

После добавления 10 000 данных, было проведено тестирование приложения и сделаны следующие выводы: при таком большом объеме денных приложение нуждается в доработке, т.к. при выборке такого большого количества продуктов, оно становится не дееспособным. Один из вариантов решения данной проблемы – постраничная загрузка данных, примерно по 100, т.к. при таком объеме проблем не наблюдалось.

# Заключение

В процессе выполнения курсовой работы была спроектирована база данных, предметной областью которой является система управления режимом питания. Также было разработано приложение демонстрирующее работу с базой данных ControlDiet. База данных была разработана в рамках Microsoft SQL Server – система управления реляционными базами данных. Данная РСУБД используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия.

Одной из главных задач данной курсовой работы - освоение технологии SQL Server Reporting Services и данная цель была достигнута. В рамках курсовой работы было разработано порядка 6 различных отчетов, построенных на базе данных ControlDiet, а также было осуществлена работа по встраиванию технологии в приложения. Данная технология довольно удобна для построения различных отчетов на базах данных.

# Список использованных источников

1. Блинова Е.А. Курс лекций по базам данных / Е.А. Блинова

2. Microsoft OLE DB [Электронный ресурс]/ msdn.microsoft.com – Режим доступа: https://msdn.microsoft.com/en-us/library/ms722784(v=vs.85).aspx Дата доступа 20.11.2017.

3. Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с.

5. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: <https://professorweb.ru> – Дата доступа: 31.10.2017.

# Приложение А

Функция для перевода строки с int-значениями в таблицу вида позиция в строке – значение.

CREATE FUNCTION iter\_intlist\_to\_table (@list ntext)

RETURNS @tbl TABLE (listpos int IDENTITY(1, 1) NOT NULL,

number int NOT NULL) AS BEGIN

DECLARE @pos int,

@textpos int,

@chunklen smallint,

@str nvarchar(4000),

@tmpstr nvarchar(4000),

@leftover nvarchar(4000)

SET @textpos = 1

SET @leftover = ''

WHILE @textpos <= datalength(@list) / 2

BEGIN

SET @chunklen = 4000 - datalength(@leftover) / 2

SET @tmpstr = ltrim(@leftover + substring(@list, @textpos, @chunklen))

SET @textpos = @textpos + @chunklen

SET @pos = charindex(' ', @tmpstr)

WHILE @pos > 0

BEGIN

SET @str = substring(@tmpstr, 1, @pos - 1)

INSERT @tbl (number) VALUES(convert(int, @str))

SET @tmpstr = ltrim(substring(@tmpstr, @pos + 1, len(@tmpstr)))

SET @pos = charindex(' ', @tmpstr)

END

SET @leftover = @tmpstr

END

IF ltrim(rtrim(@leftover)) <> ''

INSERT @tbl (number) VALUES(convert(int, @leftover))

RETURN

END

# Приложение Б

Формула расчёта BMR – суточной нормы калорий для человека.

Коэффициент физической нагрузки (КФН):

1. минимальная: КФН = 1.2;
2. низкая: КФН = 1.375;
3. средняя: КФН = 1.55;
4. высокая: КФН = 1.725;
5. очень высокая: КФН = 1.9.

Коэффициент, зависящий от цели человека (КЦ):

1. похудание: КЦ = 0.8;
2. поддержка веса: КЦ = 1;
3. набор массы: КЦ = 1.2.

BMR для женщин:

BMR = (447.6 + (9.2 x вес, кг) + (3.1 х рост, cм) – (4.3 х возраст, лет)) \* КФН.

BMR для мужчин:

BMR = (88.36 + (13.4 x вес, кг) + (4.8 х рост, см) – (5.7 х возраст, лет)) \* КФН.