

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий
Кафедра Информационные системы и технологии
Специальность 1–40 01 01 Программное обеспечение информационных технологий
Специализация 1–40 01 01 10 Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

База данных «Food»

Выполнил студент Шкода Кристина Михайловна
(Ф.И.О.)

Руководитель проекта Копыток Д.В.
(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .
(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: Копыток Д.В.
(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: Копыток Д.В.
(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой _____

Содержание

Содержание	2
ВВЕДЕНИЕ	3
1. Постановка задачи	4
1.1 Обзор прототипов	4
1.2 Анализ требований к программному средству	6
2 Разработка модели базы данных	8
3. Разработка необходимых объектов	11
3.1 Таблицы	11
3.2 Процедуры	12
3.3 Индексы	14
4. Описание процедур импорта и экспорта данных	14
4.1 Описание процедуры импорта данных	14
4.2 Описание процедуры экспорта данных	15
5. Тестирование производительности	15
6. Описание технологий	17
6.1 Резервное копирование и восстановление базы данных	17
7. Руководство пользователя	19
ЗАКЛЮЧЕНИЕ	26
Список использованных литературных источников	27
ПРИЛОЖЕНИЕ А	28
ПРИЛОЖЕНИЕ Б	30
ПРИЛОЖЕНИЕ В	33

ВВЕДЕНИЕ

В наше время технологии играют важную роль в жизни и словно по геометрической прогрессии развиваются. С развитием технологий человеку становятся доступны все новые и новые возможности, которые раньше нельзя было даже представить.

Люди стали все больше следить за здоровьем, самодисциплиной, планированием. Современный человек сегодня – это человек, стремящийся контролировать и анализировать свою жизнь. И, безусловно, такой человек не откажется от удобного приложения, которое позволит контролировать дневной рацион питания и норму калорий. Это можно сделать с помощью приложения Personal Nutritionist, в котором подключена база данных food.

Для чего нужно знать, сколько энергии человек получает с пищей и сколько расходует его организм. Количество энергии исчисляется в калориях. Так устроен мир, что люди в нем разные. Профессии некоторых людей предполагают постоянный контроль массы тела: спортсмены, танцоры и т.д. Это одни из немногих причин, по которым люди сталкиваются с необходимостью подсчитывать количество калорий, потребляемых за день.

Раньше для этого пользовались специальными таблицами, где искали продукты, смотрели количество калорий продукта и пересчитывали под нужный вес. На сегодняшний день данный процесс заметно упрощается с помощью калькулятора калорий. Не нужно самостоятельно пересчитывать калорийность под вес, считать их суммарное количество.

В соответствии с заданием курсового проекта для проектирования базы данных используется система управления базами данных Microsoft SQL Server.

В качестве интерфейса прикладного программирования был выбран обширный API-интерфейс — Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом.

Для работы с WPF использовался объектно-ориентированный язык программирования с C-подобным синтаксисом — C#, разработанный для создания приложений на платформе Microsoft .NET Framework.

1. Постановка задачи

В соответствии с заданием курсового проекта следует не только создать базу данных, но и разработать программное средство, которое должно в полной мере показать возможности базы данных. Для того чтобы сформировать окончательные требования к проектируемому программному средству сначала рассмотрим прототипы из той же области.

1.1 Обзор прототипов

Немаловажным этапом в разработке программного продукта является аналитический обзор прототипов. Аналогов десктопного программного средства «Персональный диетолог» мной найдено не было. Но были найдены сайты и мобильные приложения, осуществляющие ту же функцию, что и мое программное средство.

Например, сайт <https://allcalc.ru> помогает рассчитать суточную норму калорий. Интерфейс «Allcalc» представлен на рисунке 1.1.

The screenshot shows the 'Allcalc' website interface for calculating daily calorie needs. The header is dark purple with the 'allcalc' logo and navigation links: 'Все калькуляторы', 'Конвертеры', 'Обратная связь', and 'Приложения'. A search bar is on the right. The main content area is white and titled 'Суточная норма калорий'. It contains input fields for 'Возраст' (Age), 'Пол' (Gender) with radio buttons for 'Мужской' (Male) and 'Женский' (Female), 'Вес' (Weight) with a value of 90 and radio buttons for 'в килограммах' (in kilograms) and 'в фунтах' (in pounds), 'Рост' (Height) with a value of 186 and a unit of 'в сантиметрах' (in centimeters), and a dropdown for 'Степень физической активности' (Degree of physical activity) set to '3 раза в неделю'. Below these is a section for 'Результат' (Result) with radio buttons for 'ккал' (kcal) and 'килоджоулях' (kilojoules), and a section for 'Формула' (Formula) with radio buttons for 'Мифлина - Сан Жеора' (Mifflin - San Jeor) and 'Харриса-Бенедикта' (Harris-Benedict). An orange 'Вычислить' (Calculate) button is at the bottom.

Рисунок 1.1 – Интерфейс «Allcalc»

На этом сайте можно рассчитать по введенным пользователем данным (возраст, пол, вес, рост, степень физической нагрузки) необходимое количество калорий на день для разных целей: остаться в том же весе или похудеть. Так же можно выбрать по какой формуле пользователь хочет, чтобы были рассчитаны калории, и в какой единице измерения были выведены.

Проанализировав «Allcalc», можно выделить некоторые плюсы и минусы программного средства.

Основные плюсы:

- возможность выбрать желаемую формулу;

- возможность ввести степень физической активности;
 - возможность рассчитать калории для быстрого похудения;
- Основные минусы:
- повсеместная реклама;
 - нет возможности посчитать количество калорий для набора веса;

«Lifesum» — приложение, которое помогает следить за питанием.

Основные плюсы приложения:

- возможность подсчета съеденных калорий;
- возможность добавлять объем выпитой воды;
- возможность добавлять завтрак, обед и ужин;

Основные минусы:

- постоянно предлагает перейти на премиум;
- недоступны многие функции без покупки премиум-аккаунта;

Интерфейс «Lifesum» представлен на рисунке 1.2.

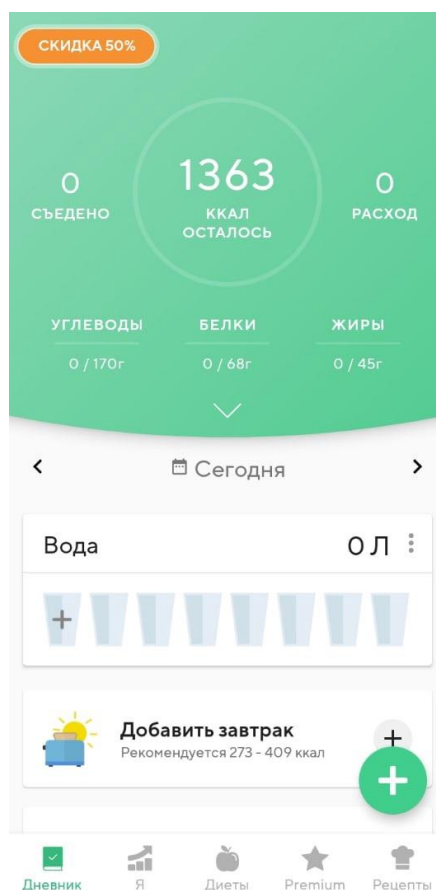


Рисунок 1.2 – Интерфейс «Lifesum»

1.2 Анализ требований к программному средству

Анализ требований — это сбор требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, незавершенности и разрешения конфликтов в процессе разработки программного обеспечения.

Различают три уровня требований к проекту:

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

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. К их числу относятся:

- простота интерфейса;
- использование принципов объектно-ориентированного программирования;
- использование системы управления базами данных.

Следующими требованиями являются требования пользователей. Данные требования описывают цели и задачи, которые пользователям позволит решить система. В данном приложении пользователи разделены на три группы: пользователь и администратор.

Пользователь должен иметь возможность:

- регистрировать себя в системе;
- входить в приложение, после ввода данных, необходимых для аутентификации;
- добавлять продукты;
- добавлять рецепты;
- добавлять понравившееся в избранное;
- заполнять завтрак, обед, ужин;
- удалять из избранного;
- выходить из профиля.

Администратор должен иметь возможность:

- выходить из профиля;
- смотреть список пользователей;
- доступ к данным пользователя;
- удалять рецепты/продукты;
- редактировать съеденное количество калорий пользователя.

Данные требования обобщены в виде диаграммы вариантов использования разрабатываемого программного средства, которая приведена в приложении А. Она отражает функциональность программного средства с точки зрения получения значимого результата для пользователя.

Функциональные требования определяют функциональность ПО, которую разработчики должны построить, чтобы пользователи смогли выполнить свои

задачи в рамках бизнес-требований. После проведения анализа были выявлены следующие функциональные требования:

- архитектура приложения должна соответствовать шаблонам проектирования, таким как MVVM и Repository.
- вся информация должна храниться в базе данных;
- приложение должно предоставлять пользователям возможность создания нового аккаунта в виде формы регистрации;
- приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим логином;
- приложение должно производить валидацию вводимых данных при регистрации, авторизации;
- приложение должно корректным образом обрабатывать возникающие исключительные ситуации;
- приложение должно предоставлять возможность пользователю добавлять/изменять завтрак, обед и ужин;
- приложение должно предоставлять возможность пользователю просматривать список рецептов и продуктов;
- приложение должно предоставлять возможность пользователю просматривать описание рецепта;
- приложение должно предоставлять возможность пользователю просматривать историю питания;
- приложение должно предоставлять возможность пользователю добавлять рецепт/продукт;
- приложение должно предоставлять возможность пользователю добавлять рецепты в избранное или же удалять из избранного;
- приложение должно предоставлять возможность администратору просматривать список пользователей;
- приложение должно предоставлять возможность администратору просматривать данные о пользователе;
- приложение должно предоставлять возможность администратору добавлять рецепты и продукты;
- приложение должно предоставлять возможность администратору удалять рецепты и продукты;
- приложение должно предоставлять возможность администратору просматривать описание рецепта;

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 сформированным списком.

2 Разработка модели базы данных

Проектирование баз данных — процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования базы данных:

- обеспечение хранения в БД всей необходимой информации;
- обеспечение возможности получения данных по всем необходимым запросам;
- сокращение избыточности и дублирования данных;
- обеспечение целостности базы данных.

Проектирование базы данных проводится в два этапа: концептуальное и логическое проектирование.

Концептуальное проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных. Основными понятиями модели являются: сущность, связь и атрибут

Сущность – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна.

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация обычно является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

Атрибут сущности – это любая деталь, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности.

В рамках этого этапа была создана модель, которая включает 8 сущностей:

- пользователь;
- избранное;
- история питания;
- продукты;
- рецепты;
- прием пищи;
- роли;
- подсчет калорий админом.

Также в модели были определены необходимые связи. Например, между сущностями роли и пользователь была установлена связь один-ко-многим. Для каждой сущности были выделены атрибуты. Например, для пользователя в качестве атрибутов были выделены такие характеристики, как идентификатор и имя.

Логическое проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для

реляционной модели данных логическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Логическая модель базы данных представлена на рисунке 3.3

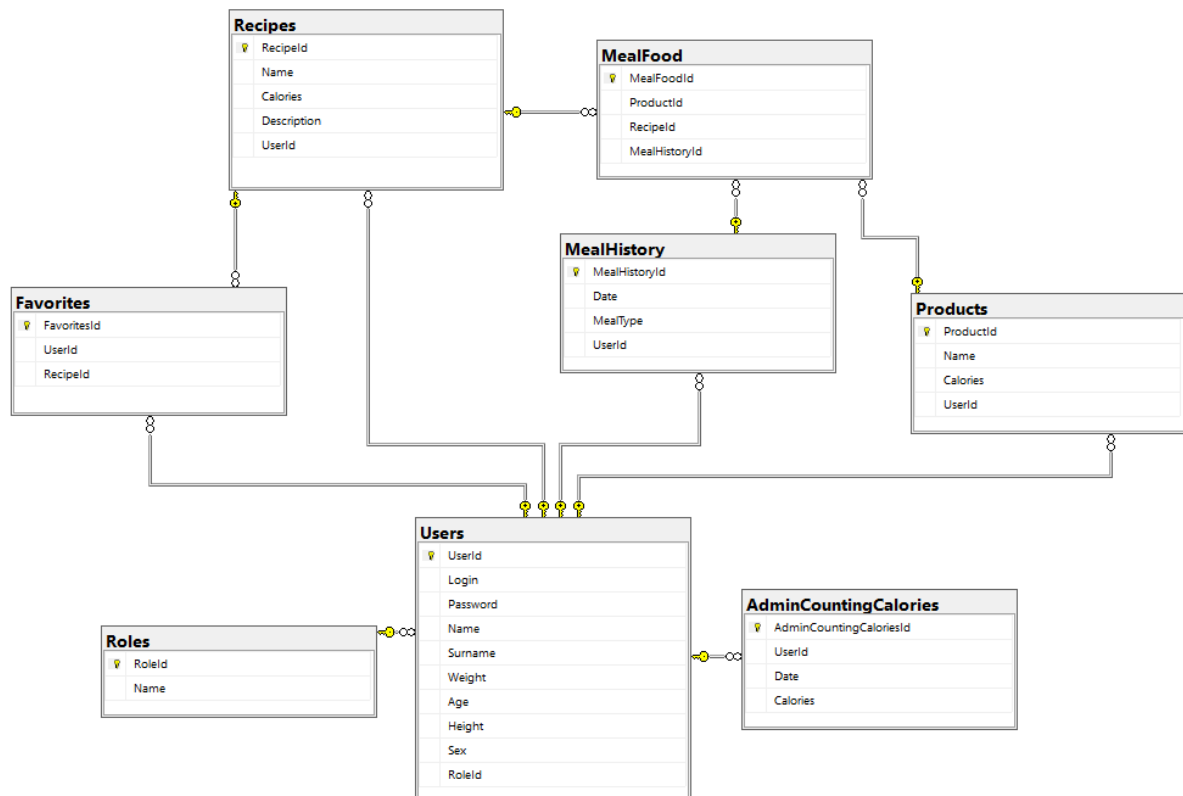


Рисунок 2.1 – Логическая модель базы данных

Всего в базе данных содержится 8 таблиц. В таблице Users хранятся все пользователи, зарегистрированные в приложении. Таблица Roles содержит роли, которые могут иметь пользователи. Таблица AdminCountingCalories хранит посчитанное администратором количество калорий за данный день. Таблица Products хранит продукты, созданные либо администратором, либо пользователем. Таблица Recipes хранит рецепты, добавленные либо администратором, либо пользователем. Таблица Favorites содержит рецепты, которые понравились пользователю. Таблица MealHistory хранит историю приема пищи. В таблице MealFood хранится еда, которую пользователь потребил на завтрак, обед и ужин.

Таблица Users состоит из 10 столбцов:

- UserId;
- Login;
- Password;
- Name;

- Surname;
- Weight;
- Age;
- Height;
- Sex;
- RoleId.

В столбцах Name и Surname хранится информация о имени и фамилии зарегистрированного пользователя соответственно, в столбце Weight – вес пользователя, Age – возраст, Height – рост, Sex – пол. UserId – идентификатор пользователя, RoleId – идентификатор роли пользователя. Login – логин пользователя, Password – пароль пользователя.

Таблица Roles состоит из 2 столбцов:

- RoleId;
- Name;

В столбце Name хранится информация о названии роли пользователя, в столбце RoleId – идентификатор.

Таблица AdminCountingCalories состоит из 4 столбцов:

- AdminCountingCaloriesId;
- UserId;
- Date;
- Calories.

В столбце Calories хранится количество калорий, посчитанные администратором, в столбце Date – даты, когда было съедено данное количество калорий. UserId – идентификатор пользователя, который съел столько калорий. AdminCountingCaloriesId – идентификатор числа калорий, посчитанных администратором.

Таблица Products состоит из 4 столбцов:

- ProductId;
- Name;
- Calories;
- UserId.

В столбце ProductId хранится идентификатор продукта питания, UserId – идентификатор пользователя, который создал данный продукт. Name – название продукта питания. Calories – это количество калорий в данном продукте питания.

Таблица Recipes состоит из 5 столбцов:

- RecipeId;
- Name;
- Calories;
- Description;
- UserId.

В столбце `RecipeId` – идентификатор рецепта, `UserId` – идентификатор пользователя, который создал данный рецепт. `Name` – название рецепта. `Calories` – количество калорий в данном рецепте, `Description` – описание рецепта.

Таблица `Favorites` состоит из 3 столбцов:

- `FavoriteId`;
- `UserId`;
- `RecipeId`.

В столбце `FavoriteId` хранится идентификатор рецепта, находящегося в избранном. `UserId` – идентификатор пользователя, которому понравился рецепт. `RecipeId` – идентификатор рецепта из таблицы рецептов.

Таблица `MealFood` состоит из 4 столбцов:

- `MealFoodId`;
- `ProductId`;
- `RecipeId`;
- `MealHistoryId`.

В столбце `MealFoodId` хранится идентификатор приема пищи, характеризующий потребление продукта питания или же блюда по рецепту, в столбце `ProductId` – идентификатор продукта питания, который был потреблен за данный прием пищи, столбце `RecipeId` – идентификатор рецепта, который был потреблен за данный прием пищи. `MealHistoryId` – идентификатор истории приема пищи.

Таблица `MealHistory` состоит из 4 столбцов:

- `MealHistoryId`;
- `Date`;
- `MaelType`;
- `UserId`.

В столбце `MealHistoryId` хранится идентификатор истории приема пищи. В столбце `Date` – дата, когда был прием пищи, в столбце `MaelType` – тип приема пищи (0 – завтрак, 1 – обед, 2 – ужин). `UserId` – идентификатор пользователя, который ел.

3. Разработка необходимых объектов

3.1 Таблицы

Таблица – это совокупность связанных данных, хранящихся в структурированном виде в базе данных. Она состоит из столбцов и строк.

Столбцы таблицы называют полями; каждое поле характеризуется своим именем (названием соответствующего свойства) и типом данных, отражающих значения данного свойства. Каждое поле обладает определенным набором свойств (размер, формат и др.). Поле базы данных – это столбец таблицы, включающий в себя значения определенного свойства.

В каждой таблице должно быть, по крайней мере, одно ключевое поле, содержимое которого уникально для любой записи в этой таблице. Значения ключевого поля однозначно определяют каждую запись в таблице

База данных данного курсового проекта содержит 8 таблиц, которые описаны в главе 2.

Листинг SQL-кода для создания таблиц находится в приложении А.

3.2 Процедуры

Хранимая процедура – это поименованный код на языке T-SQL. Хранимая процедура может быть создана с помощью CREATE, изменена с помощью ALTER и удалена с помощью оператора DROP. Процедура может принимать входные и формировать выходные параметры. Результатом ее выполнения может быть целочисленное значение, которое возвращается к точке вызова оператором RETURN, либо один или более результирующих наборов, сформированных операторами SELECT, либо содержимое стандартного выходного потока, полученного при выполнении операторов PRINT. Вызов процедуры осуществляется оператором EXECUTE (EXEC). В хранимых процедурах допускается применение основных DDL, DML и TCL-операторов, конструкций TRY/CATCH, курсоров, временных таблиц.

Листинг SQL-кода для создания процедур, выполняющих вставку информации, находится в приложении Б.

Процедуры, разработанные в рамках курсового проекта, выполняющие чтение информации:

- GetUsers– получение всех пользователей;
- GetWithUserRole– получение всех пользователей, у кого роль User;
- GetUserByLogin– получение пользователя по логину;
- GetUserById – получение пользователя по его идентификатору;
- GetRoleById – получение роли по ее идентификатору;
- GetRoleByName – получение роли по ее названию;
- GetMealHistory– получение истории питания за день;
- GetMealHistoryByType – получение истории питания за день по ее типу;
- GetMealFoodByMealHistoryId – получение потребленной пищи по идентификатору истории питания;
- GetMealFoodByProductId – получение потребленных продуктов;
- GetMealFoodByRecipeId – получение потребленных блюд;
- GetProductById – получение продукта питания по его идентификатору;
- GetProducts – получение всех продуктов питания, доступных пользователю;
- GetAllProducts – получение всех продуктов питания;
- GetRecipeById – получение рецепта по его идентификатору;
- GetRecipes – получение всех рецептов, доступных пользователю;

- GetAllRecipes – получение всех рецептов;
- GetRecipesForRecommendations – получение рецептов для рекомендации пользователю;
- GetFavorites – получение избранных рецептов пользователя;
- GetFavoritesByRecipe – получение избранных рецептов по идентификатору рецепта;
- GetFavoritesByRecipeAndUser – получение избранных рецептов пользователя;
- GetRecommendation – получение рецептов, рекомендованных администратором пользователю;
- GetRecommendation – получение рекомендованного рецепта по его идентификатору;
- GetAdminCountingCalories – получение количества калорий, установленных администратором.

Процедуры, разработанные в рамках курсового проекта, выполняющие вставку информации:

- AddNewRole – добавление новой роли;
- AddNewUser – добавление нового пользователя;
- AddNewProduct – добавление нового продукта питания;
- AddNewRecipe – добавление нового рецепта;
- AddNewFavorite – добавление в избранное;
- AddNewMealHistory – добавление в историю питания;
- AddNewMealFoodProduct – добавление продукта в съеденное;
- AddNewMealFoodRecipe – добавление рецепта в съеденное;
- AddNewAdminRecommendation – добавление рекомендаций;
- AddAdminCountingCalorie – добавление количества калорий.

Процедуры, разработанные в рамках курсового проекта, выполняющие изменение информации в базе данных:

- UpdateAdminCountingCalorie – изменение количества калорий;
- UpdateUser – процедура для изменения информации о пользователе.

Процедуры, разработанные в рамках курсового проекта, выполняющие удаление записей из базы данных:

- DeleteProduct – удаление продукта;
- DeleteRecipe – удаление рецепта;
- DeleteFavorite – удаление из избранного;
- DeleteAdminRecommendation – удаление рекомендаций администратора;
- DeleteByProductId – удаление продукта по его идентификатору;
- DeleteByRecipeId – удаление рецепта по его идентификатору;

– DeleteMealFood – удаление потребленной пищи.

3.3 Индексы

Индекс — объект базы данных, создаваемый с целью повышения производительности поиска данных. Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путём последовательного просмотра таблицы строка за строкой может занимать много времени.

Процедуры, разработанные в рамках курсового проекта:

- #User_index;
- #Products_index;
- #Recipes_index.

Листинг SQL-кода для создания индексов находится в приложении В.

4. Описание процедур импорта и экспорта данных

В данной курсовой работе реализованы процедуры экспорта и импорта данных из XML файла в базу данных таблицы Products и наоборот. При данных операциях работает с файловой системой приложение, а разбором и генерацией XML занимается MS SQL Server.

4.1 Описание процедуры импорта данных

BULK –Инструкция T-SQL, импортирующая данные непосредственно из файла данных в таблицу базы данных или несекционированное представление.

Код процедура импорта данных в формат XML представлена на рисунке 4.1.

```

go
;Create or alter Procedure ImProductsfromXml
AS
;Begin
;
;    INSERT INTO Products(Name, Calories, UserId)
;    SELECT
;        MY_XML.PRODUCT.query('Name').value('.', 'nvarchar(300)'),
;        MY_XML.PRODUCT.query('Calories').value('.', 'int'),
;        MY_XML.PRODUCT.query('UserId').value('.', 'int')
;    FROM (SELECT CAST(MY_XML AS xml)
;          FROM OPENROWSET(BULK 'C:\univer\DB\products.xml', SINGLE_BLOB) AS T(MY_XML)) AS T(MY_XML)
;          CROSS APPLY MY_XML.nodes('PRODUCTS/PRODUCT') AS MY_XML (PRODUCT);
;
;End;

exec ImProductsfromXml
drop procedure ImProductsfromXml

```

Рисунок 4.1 – Процедура импорта данных из xml в таблицу Products

4.2 Описание процедуры экспорта данных

Для преобразования результата SELECT-запроса в формат XML в операторе SELECT применяется секция FOR XML. При этом могут использоваться режимы RAW, AUTO, PATH. Код процедуры экспорта представлена на рисунке 4.2.

```
Create or ALTER Procedure ExProductstoXml
AS
Begin
    select * from Products
        for xml PATH('PRODUCT'), root('PRODUCTS');

    --to use xp_cmdshell
    EXEC master.dbo.sp_configure 'show advanced options', 1
        RECONFIGURE WITH OVERRIDE
    EXEC master.dbo.sp_configure 'xp_cmdshell', 1
        RECONFIGURE WITH OVERRIDE;
    -- Save XML records to a file
    DECLARE @fileName nVARCHAR(100)
    DECLARE @sqlStr VARCHAR(1000)
    DECLARE @sqlCmd VARCHAR(1000)

    SET @fileName = 'C:\univer\DB\products.xml'
    SET @sqlStr = 'USE food; select * from Products for xml path(''PRODUCT''), root(''PRODUCTS'') '
    SET @sqlCmd = 'bcp "' + @sqlStr + '" queryout ' + @fileName + ' -w -T'
    EXEC xp_cmdshell @sqlCmd;
End;

exec ExProductstoXml
```

Рисунок 4.2 – Процедура экспорта

5. Тестирование производительности

Производительность БД является решающим фактором эффективности управленческих и коммерческих приложений. Если поиск или запись данных выполняется медленно – способность к нормальной работе приложения падает. Существует единственный путь выяснить причину плохой производительности – выполнить количественные измерения и определить, что является причиной проблемы производительности.

Проблемы выявления узких мест производительности баз данных напрямую связаны с метриками, методами измерений производительности и технологией их выполнения.

Для того чтобы правильно организовать процесс тестирования БД, тестировщики должны обладать хорошими знаниями SQL и DML и иметь ясное представление о внутренней структуре БД. Это самый лучший и надежный способ тестирования БД особенно для приложений с низким и средним уровнем сложности. Данный метод не только дает уверенность, что тестирование выполнено качественно, но также повышает мастерство написания SQL-запросов.

В MS SQL Server оптимизация запросом в основном заключается в построение индексов над таблицами, и изменением плана запроса.

Для тестирования производительности в таблицу Comments были добавлены 100 000 записей. Код добавления 100 000 записей представлен в Приложении Г.

Ниже на рисунке 5.1 представлена карта запроса при выполнении процедуры поиска продукта по полю Name.

Clustered Index Scan (Clustered)	
Просмотр всего кластеризованного индекса или его части.	
Физическая операция	Clustered Index Scan
Логическая операция	Clustered Index Scan
Предполагаемый режим выполнения	Row
Хранилище	RowStore
Предполагаемая стоимость оператора	1,27032 (100%)
Предполагаемая стоимость операций ввода-вывода	1,16016
Предполагаемая стоимость поддерева	1,27032
Предполагаемая стоимость ЦП	0,110159
Предполагаемое количество выполнений	1
Приблизительное число считываемых строк	100002
Расчетное число строк для всех выполнений	1,00018
Расчетное число строк на выполнение	1,00018
Предполагаемый размер строки	122 Б
Отсортировано	False
Идентификатор узла	0
Предикат	
[food].[dbo].[Products].[Name]=CONVERT_IMPLICIT(nvarchar(4000),[@1],0)	
Объект	
[food].[dbo].[Products].[PK_Products]	
Список выходных столбцов	
[food].[dbo].[Products].ProductId; [food].[dbo].[Products].Name; [food].[dbo].[Products].Calories; [food].[dbo].[Products].UserId	

Рисунок 5.1 – Карта запроса при выполнении процедуры поиска продукта, перед созданием индекса покрытия

Как можно увидеть, стоимость запроса достаточно велика. Для оптимизации поиска был добавлен индекс покрытия, код которого представлен на рисунке 5.2.

```
create index #Products_index on Products(Name)
```

Рисунок 5.2 – Индекс покрытия для таблицы «Products»,

Ниже на рисунке 5.3 представлена карта запроса при выполнении процедура поиска продукта по имени, после создания индекса покрытия.

Вложенные циклы	
Для каждой строки верхнего (внешнего) входа производится просмотр нижнего (внутреннего) входа и вывод совпадающих строк.	
Физическая операция	Вложенные циклы
Логическая операция	внутреннее соединение
Предполагаемый режим выполнения	Row
Предполагаемая стоимость операций ввода-вывода	0
Предполагаемая стоимость оператора	0,00000042 (0%)
Предполагаемая стоимость ЦП	0,00000042
Предполагаемая стоимость поддерева	0,0065704
Предполагаемое количество выполнений	1
Расчетное число строк на выполнение	1
Расчетное число строк для всех выполнений	1
Предполагаемый размер строки	122 Б
Идентификатор узла	0
Список выходных столбцов	
[food].[dbo].[Products].ProductId; [food].[dbo].[Products].Name; [food].[dbo].[Products].Calories; [food].[dbo].[Products].UserId	
Внешние ссылки	
[food].[dbo].[Products].ProductId	

Рисунок 5.3 – Карта запроса при выполнении процедуры поиска продукта, после создания индекса покрытия

По результатам стоимости запроса можно сказать, что построение индексов в данной таблице необходимо, так как во втором случае общая стоимость запроса уменьшилась.

6. Описание технологий

6.1 Резервное копирование и восстановление базы данных

Для копирования базы данных была разработана процедура ExDB. Для этого используется конструкция backup database, которая экспортирует базу данных в файл с расширением bak. Эта процедура представлена ниже на рисунке 6.1.

```

--Export DB
go
Create Procedure ExportDB
AS
Begin
    BACKUP DATABASE food
    TO DISK = N'C:\univer\DB\Course Project\ADO.NET\scripts\food.bak'
    with
        stats = 10
End;

drop procedure ExportDB

exec ExportDB

```

Рисунок 6.1 – Пример создания процедуры ExportDB

Для восстановления базы данных была разработана процедура ImportDB. Для этого используется конструкция restore database, которая экспортирует базу данных в файл с расширением bak. Эта процедура представлена ниже на рисунке 6.2.

```

Create or ALTER Procedure ExProductstoXml
AS
Begin
    select * from Products
        for xml PATH('PRODUCT'), root('PRODUCTS');

    --to use xp_cmdshell
    EXEC master.dbo.sp_configure 'show advanced options', 1
        RECONFIGURE WITH OVERRIDE
    EXEC master.dbo.sp_configure 'xp_cmdshell', 1
        RECONFIGURE WITH OVERRIDE;
    -- Save XML records to a file
    DECLARE @fileName nVARCHAR(100)
    DECLARE @sqlStr VARCHAR(1000)
    DECLARE @sqlCmd VARCHAR(1000)

    SET @fileName = 'C:\univer\DB\products.xml'
    SET @sqlStr = 'USE food; select * from Products for xml path(''PRODUCT''), root(''PRODUCTS'')';
    SET @sqlCmd = 'bcp "' + @sqlStr + '" queryout ' + @fileName + ' -w -T'
    EXEC xp_cmdshell @sqlCmd;
End;

exec ExProductstoXml

```

Рисунок 6.2 – Пример создания процедуры ImportDB

Для запуска и исполнения данной процедуры необходимо ее создавать системной базе данных master. Это необходимо для избегания коллизий при восстановлении. Так же восстановление невозможно, при наличии

подключенных к базе пользователей, поэтому переводим базу данных в монопольный режим с помощью `set single_user` и отменяем все изменения через параметр `rollback immediate`. После восстановления возвращаем базу данных в прежнее состояние через `set multi_user`.

7. Руководство пользователя

При запуске программы первым откроется окно авторизации. Оно приведено на рисунке 7.1.

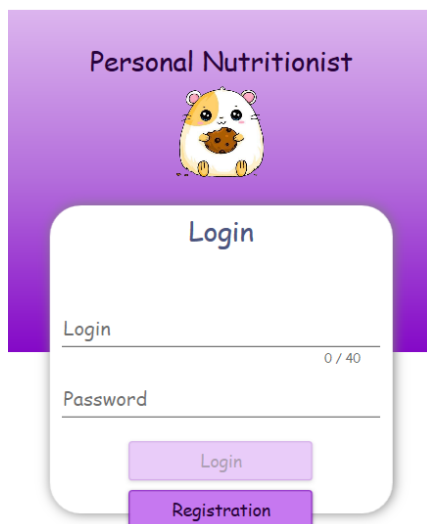


Рисунок 7.1 – Стартовое окно

После установки и регистрации в зависимости от роли пользователя предоставляются различные функции. Далее будут перечислены возможности обычного пользователя на рисунках 6.1 – 6.10.

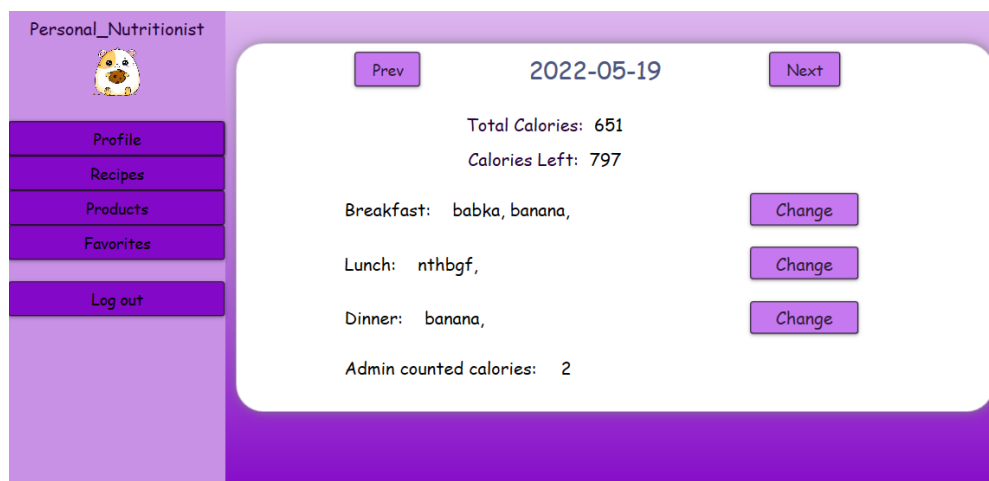


Рисунок 7.2 – Просмотр приемов пищи

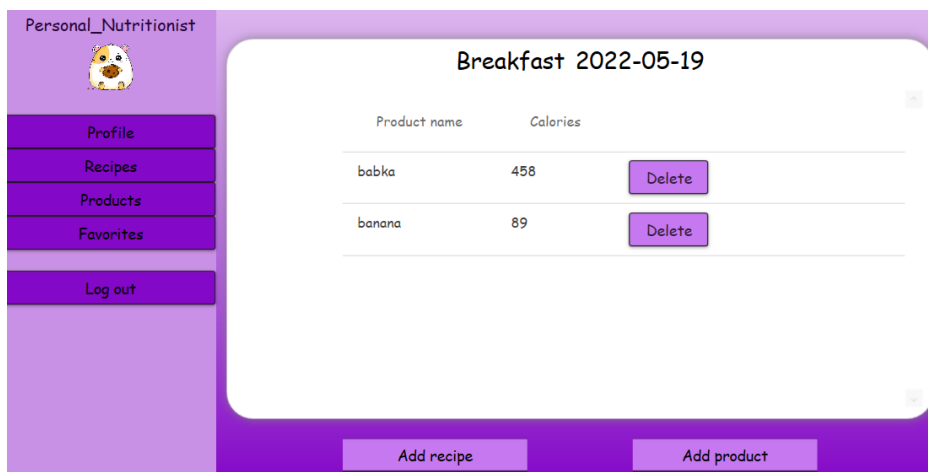


Рисунок 7.3– Добавление и удаление еды

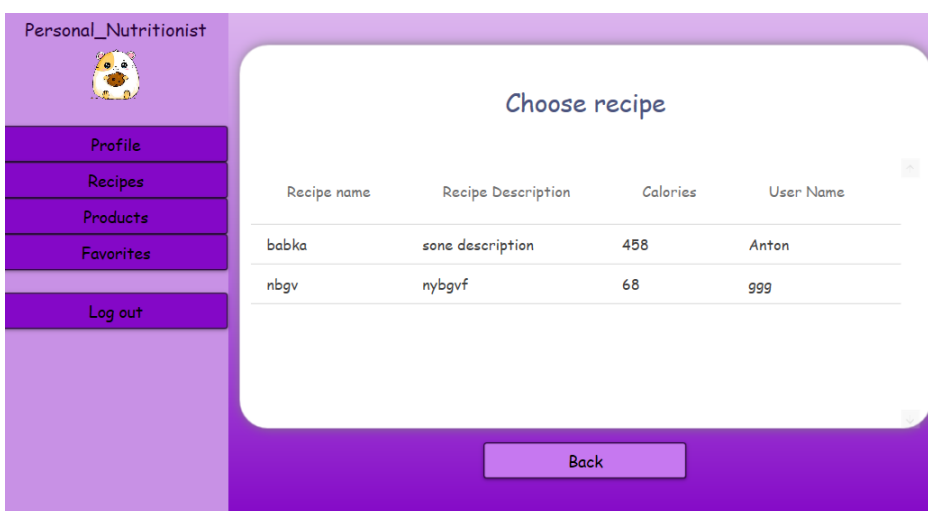


Рисунок 7.4– Выбор рецепта при добавлении в рацион

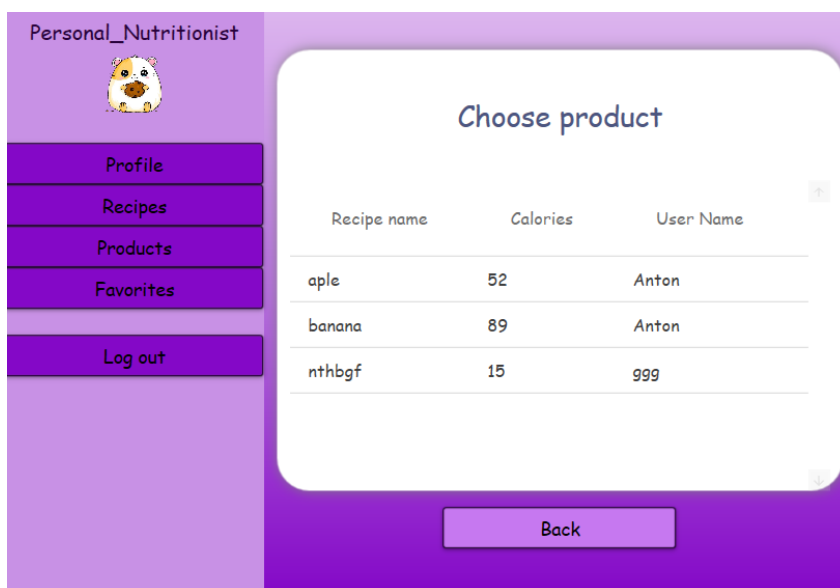


Рисунок 7.5– Выбор продукта при добавлении в рацион

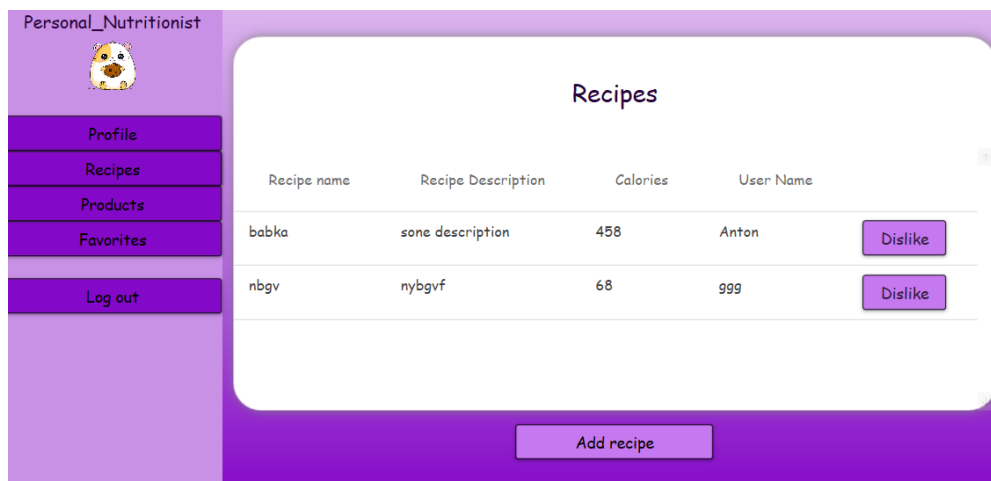


Рисунок 7.6 – Просмотр рецептов

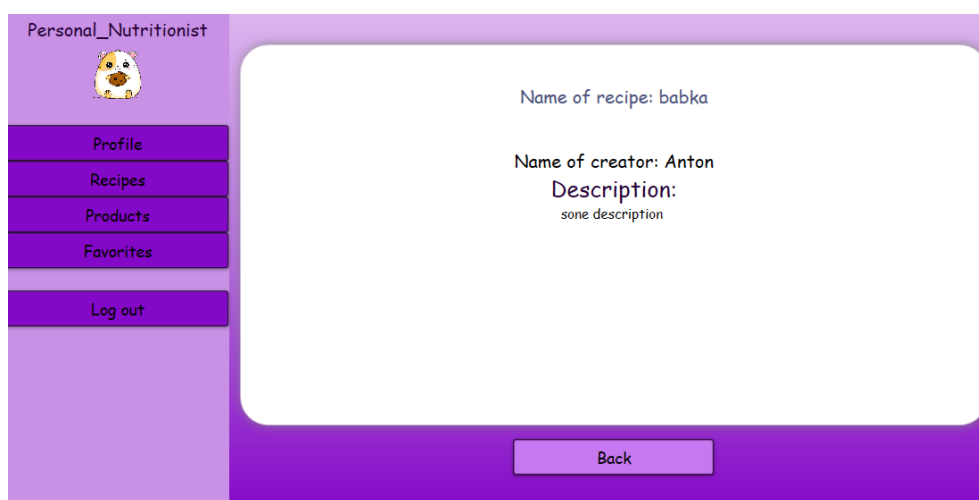


Рисунок 7.7 – Просмотр описания рецепта

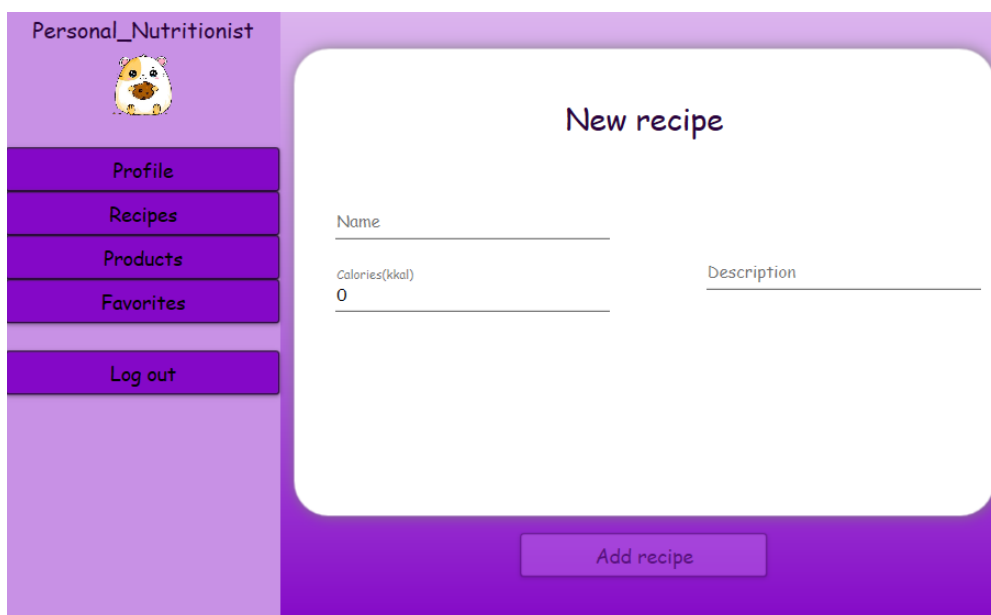


Рисунок 7.8 – Добавление рецепта

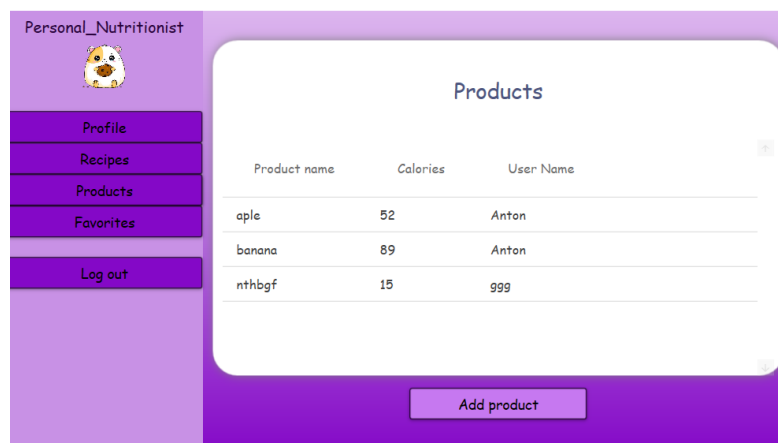


Рисунок 7.9 – Просмотр продуктов питания

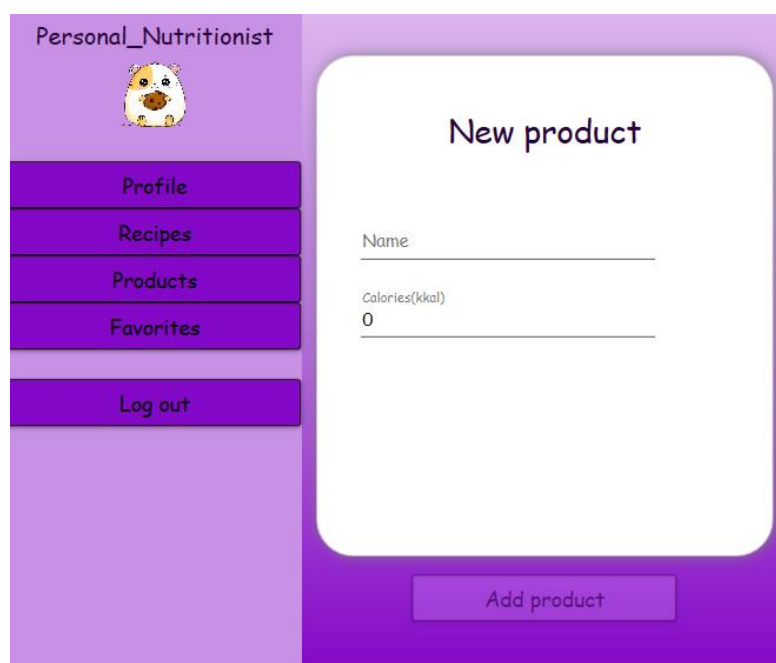


Рисунок 7.10 – Добавление продуктов питания

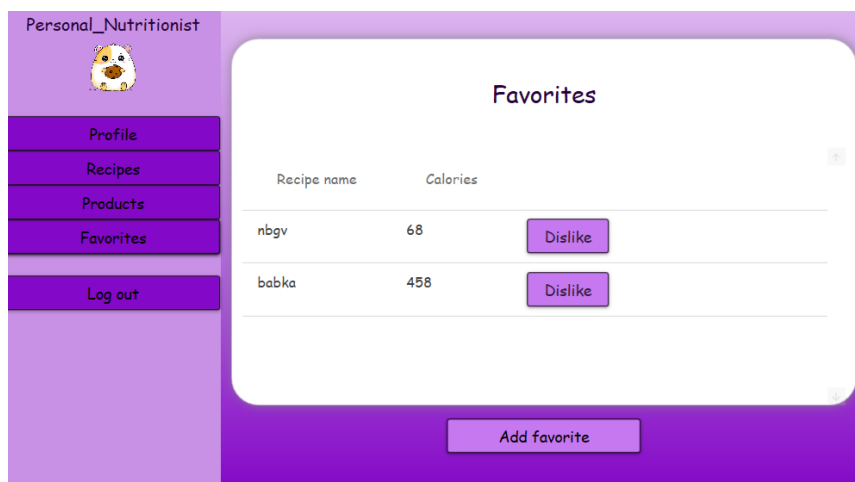


Рисунок 7.11 – Просмотр избранного

На рисунках 7.12-7.18 показаны возможности администратора.

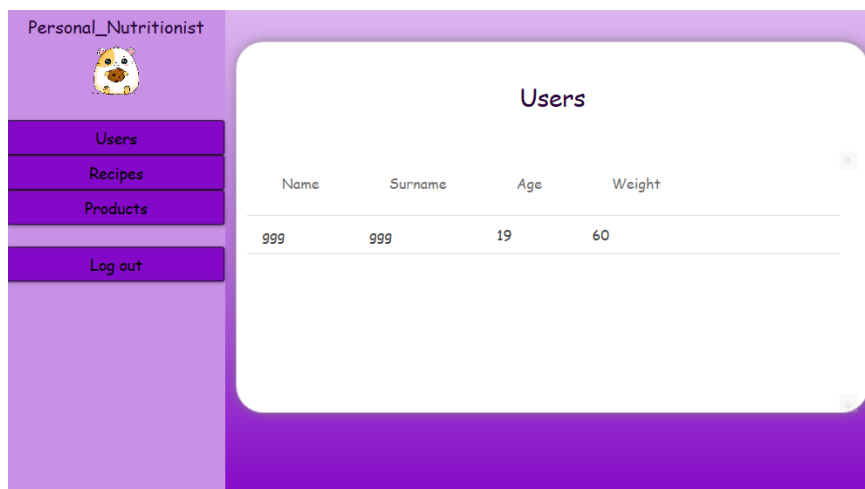


Рисунок 7.12 – Просмотр списка зарегистрированных пользователей

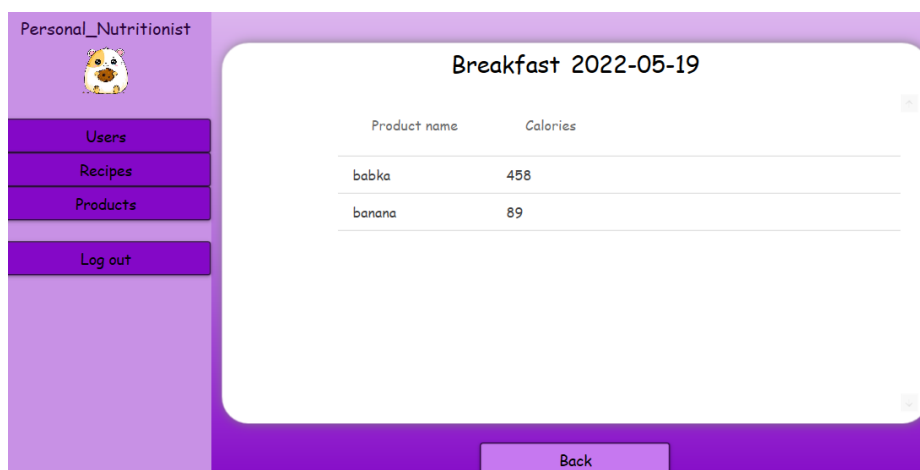


Рисунок 7.13 – Просмотр рациона

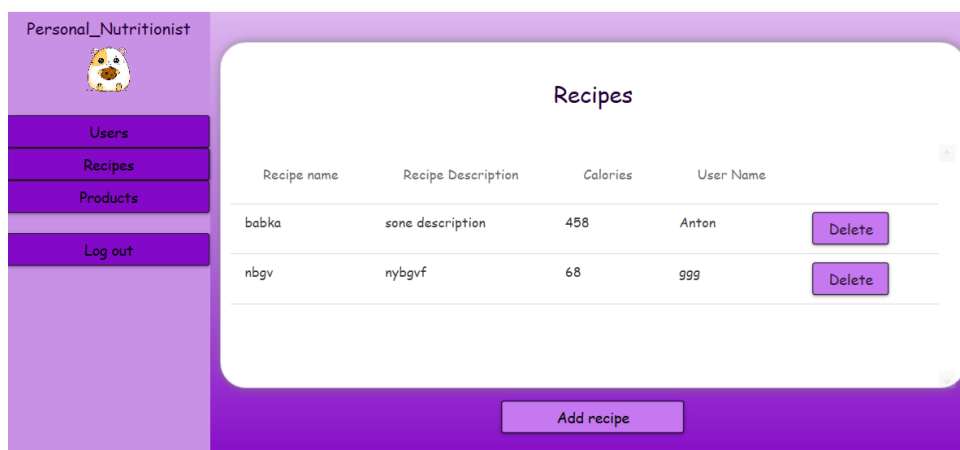


Рисунок 7.14 – Просмотр/ удаление рецептов

The screenshot shows the 'New recipe' form in the Personal_Nutritionist application. The interface has a purple sidebar on the left with the title 'Personal_Nutritionist' and a hamster icon. Below the icon are four buttons: 'Users', 'Recipes', 'Products', and 'Log out'. The main area is white with a rounded rectangle containing the title 'New recipe'. There are three input fields: 'Name' (a single line), 'Calories(kkal)' (a single line with the value '0' entered), and 'Description' (a single line). At the bottom right of the white area is a purple button labeled 'Add recipe'.

Рисунок 7.15 – Добавление рецепта

The screenshot shows the recipe details view in the Personal_Nutritionist application. The interface has a purple sidebar on the left with the title 'Personal_Nutritionist' and a hamster icon. Below the icon are four buttons: 'Users', 'Recipes', 'Products', and 'Log out'. The main area is white with a rounded rectangle containing the following text: 'Name of recipe: babka', 'Name of creator: Anton', and 'Description: sone description'.

Рисунок 7.16 – Просмотр описания рецепта

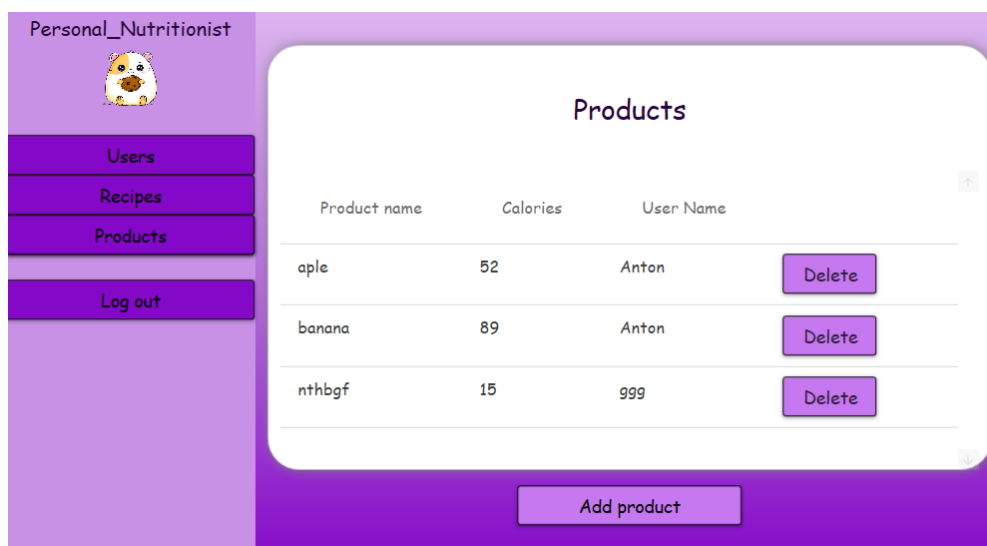


Рисунок 7.17 – Просмотр/удаление продуктов питания

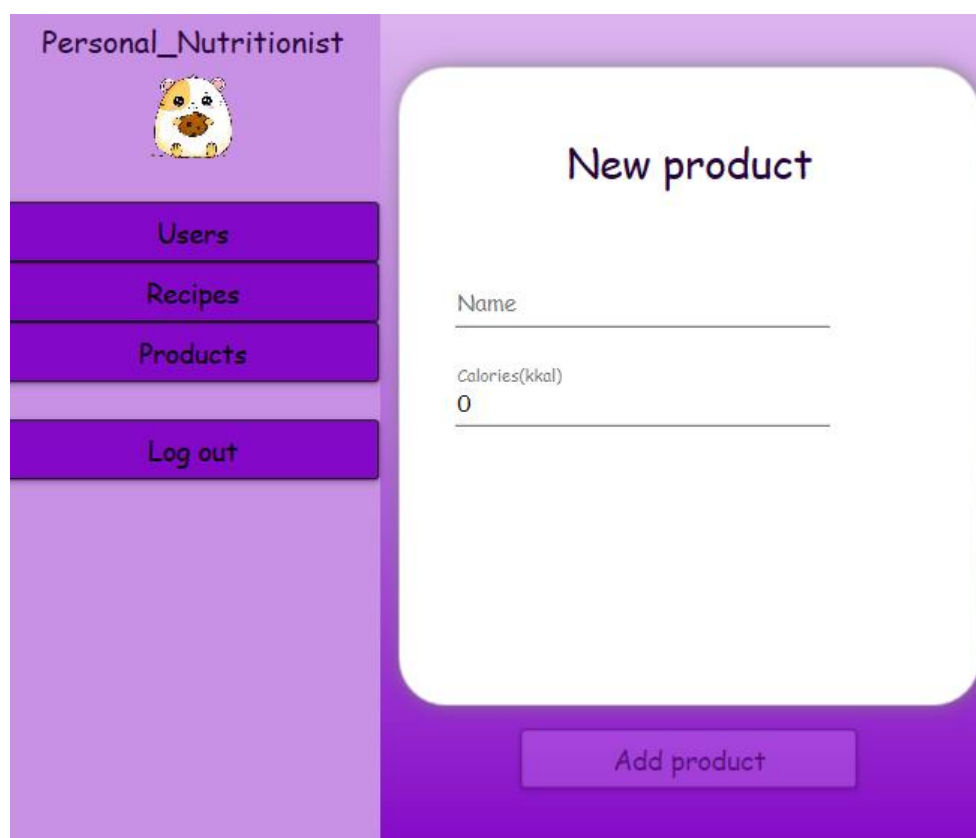


Рисунок 7.18 – Добавление продукта питания

Исходя из приведённых рисунков видно, как именно нужно использовать программное средство всех ролей.

ЗАКЛЮЧЕНИЕ

В ходе данного курсового проекта: была разработана модель и объекты базы данных food для приложения Personal Nutritionist; были разработаны, описаны и применены на практике процедуры экспорта и импорта данных; было проведено тестирование производительности базы данных, входе которого были разработаны индексы, ускорившие получение данных; разработаны процедуры резервного копирования и восстановления базы данных ; было описано руководство пользователя для сайта.

Функционально были выполнены следующие задачи:

- Добавление/удаление продуктов и рецептов;
- Регистрация и авторизация;
- Просмотр рецептов;
- Добавление/удаление продуктов и блюд в историю питания;
- Добавление/удаление в избранное;
- Рекомендовать пользователю блюда.

Список использованных литературных источников

- 1 Блинова Е.А. Курс лекций по базам данных/ Е.А. Блинова
- 2 MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа –
URL: <http://msdn.microsoft.com/library/rus/> – Дата доступа: 10.09.2022
- 3 Примеры массового импорта и экспорта XML-документов (SQL Server) [Электронный ресурс] / Режим доступа – URL:
<https://stackoverflow.com/questions/9672780/importing-xml-data-from-xml-file-to-sql-database> – Дата доступа: 08.11.2022
- 4 METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа – URL: <https://metanit.com> – Дата доступа: 03.10.2022
- 5 Запрос с полнотекстовым поиском [Электронный ресурс] / Режим доступа – URL: <https://docs.microsoft.com/ru-ru/sql/relational-databases/search/query-with-full-text-search?view=sql-server-2017> – Дата доступа 26.10.2022
- 6 Запрос с полнотекстовым поиском [Электронный ресурс] / Режим доступа – URL: <https://info-comp.ru/sisadminst/486-full-text-search-ms-sql-server.html> – Дата доступа: 19.10.2022
- 7 Об индексах в MS SQL Server [Электронный ресурс] / Режим доступа – URL: <https://habr.com/ru/post/247373/> – Дата доступа: 10.11.2022

ПРИЛОЖЕНИЕ А

```

drop database food
Create database food

use food
GO
create table Roles
(
    RoleId int IDENTITY(1,1) NOT NULL,
    Name nvarchar(max) NOT NULL,

    CONSTRAINT [PK_Roles] PRIMARY KEY CLUSTERED ([RoleId] ASC )
)
GO
create table Users
(
    UserId int IDENTITY(1,1) NOT NULL,
    Login nvarchar(100) NOT NULL,
    Password nvarchar(max) NOT NULL,
    Name nvarchar(max) NOT NULL,
    Surname nvarchar(max) NOT NULL,
    Weight real NOT NULL,
    Age int NOT NULL,
    Height int NOT NULL,
    Sex int NOT NULL,
    RoleId int NOT NULL,

    CONSTRAINT PK_Users PRIMARY KEY CLUSTERED ( UserId ASC ),
    CONSTRAINT FK_Users_Roles_RoleId FOREIGN KEY(RoleId) References Roles(RoleId) ON
DELETE CASCADE
)
GO
create table Products
(
    ProductId int IDENTITY(1,1) NOT NULL,
    Name nvarchar(300) NOT NULL,
    Calories int NOT NULL,
    UserId int NOT NULL,

    CONSTRAINT PK_Products PRIMARY KEY CLUSTERED ( ProductId ASC ),
    CONSTRAINT FK_Products_Users_UserId FOREIGN KEY(UserId) References Users(UserId) ON
DELETE CASCADE
)
create table Recipes
(
    RecipeId int IDENTITY(1,1) NOT NULL,
    Name nvarchar(300) NOT NULL,
    Calories int NOT NULL,
    Description nvarchar(max) NOT NULL,
    UserId int NOT NULL,
    CONSTRAINT PK_Recipes PRIMARY KEY CLUSTERED ( RecipeId ASC ),
    CONSTRAINT FK_Recipes_Users_UserId FOREIGN KEY(UserId) References Users(UserId) ON
DELETE CASCADE
)
create table Favorites
(
    FavoritesId int IDENTITY(1,1) NOT NULL,
    UserId int NOT NULL,
    RecipeId int NOT NULL,

```

```

        CONSTRAINT PK_Favorites PRIMARY KEY CLUSTERED (FavoritesId ASC),
        CONSTRAINT FK_Favorites_Users_UserId FOREIGN KEY(UserId) References Users(UserId),
        CONSTRAINT FK_Favorites_Recipes_RecipeId FOREIGN KEY(RecipeId) References
Recipes(RecipeId) ON DELETE CASCADE
    )

create table MealHistory
(
    MealHistoryId int IDENTITY(1,1) NOT NULL,
    Date date NOT NULL,
    MealType int NOT NULL,
    UserId int NOT NULL,
    CONSTRAINT PK_MealHistory PRIMARY KEY CLUSTERED ( MealHistoryId ASC),
    CONSTRAINT FK_MealHistory_Users_UserId FOREIGN KEY(UserId) References Users(UserId)
)

create table MealFood
(
    MealFoodId int IDENTITY(1,1) NOT NULL,
    ProductId int NULL,
    RecipeId int NULL,
    MealHistoryId int NOT NULL,
    CONSTRAINT PK_MealFood PRIMARY KEY CLUSTERED (MealFoodId ASC),
    CONSTRAINT FK_MealFood_MealHistory_MealHistoryId FOREIGN KEY(MealHistoryId)
References MealHistory(MealHistoryId),
    CONSTRAINT FK_MealFood_Products_ProductId FOREIGN KEY(ProductId) References
Products(ProductId) ON DELETE CASCADE,
    CONSTRAINT FK_MealFood_Recipes_RecipeId FOREIGN KEY(RecipeId) References
Recipes(RecipeId),
)

create table AdminRecommendations
(
    AdminRecommendationId int IDENTITY(1,1) NOT NULL,
    UserId int NOT NULL,
    RecipeId int NOT NULL,
    CONSTRAINT PK_AdminRecommendations PRIMARY KEY CLUSTERED (AdminRecommendationId
ASC),
    CONSTRAINT FK_AdminRecommendations_Recipes_RecipeId FOREIGN KEY(RecipeId)
References Recipes(RecipeId) ON DELETE CASCADE,
    CONSTRAINT FK_AdminRecommendations_Users_UserId FOREIGN KEY(UserId) References
Users(UserId),
)

create table AdminCountingCalories
(
    AdminCountingCaloriesId int IDENTITY(1,1) NOT NULL,
    UserId int NOT NULL,
    Date date NOT NULL,
    Calories int NOT NULL,
    CONSTRAINT PK_AdminCountingCalories PRIMARY KEY CLUSTERED (
AdminCountingCaloriesId ASC),
    CONSTRAINT FK_AdminCountingCalories_Users_UserId FOREIGN KEY(UserId) References
Users(UserId),
)

```

Листинг1 – Скрипт создания базы данных и таблиц

ПРИЛОЖЕНИЕ Б

```
--Insert inside Roles
go
Create Procedure AddNewRole
    @Name nvarchar(max)
AS
Begin
    Insert into Roles(Name)
    values(@Name);
End;
go

--Insert inside Users
go
Create Procedure AddNewUser
    @Login nvarchar(100),
    @Password nvarchar(max),
    @Name nvarchar(max),
    @Surname nvarchar(max),
    @Weight real,
    @Age int,
    @Height int,
    @Sex int,
    @RoleId int
AS
Begin
    Insert into Users(Login, Password, Name, Surname, Weight, Age, Height, Sex, RoleId)
    values(@Login, @Password, @Name, @Surname, @Weight, @Age, @Height, @Sex, @RoleId);
End;
go

--Insert inside Products
go
Create Procedure AddNewProduct
    @Name nvarchar(300),
    @Calories int,
    @UserId int
AS
Begin
    Insert into Products(Name, Calories, UserId)
    values(@Name, @Calories, @UserId);
End;
go

--Insert inside Recipes
go
Create Procedure AddNewRecipe
    @Name nvarchar(300),
    @Calories int,
    @Description nvarchar(max),
    @UserId int
AS
Begin
    Insert into Recipes(Name, Calories, Description, UserId)
    values(@Name, @Calories, @Description, @UserId);
End;
go

--Insert inside Favorites
go
Create Procedure AddNewFavorite
```

```

        @UserId int,
        @RecipeId int
AS
Begin
    Insert into Favorites(UserId, RecipeId)
    values(@UserId, @RecipeId);
End;
go

--Insert inside MealHistory
go
Create Procedure AddNewMealHistory
    @Date date,
    @MealType int,
    @UserId int
AS
Begin
    Insert into MealHistory(Date, MealType, UserId)
    values(@Date, @MealType, @UserId);
End;
go

--Insert inside MealFood product
go
Create Procedure AddNewMealFoodProduct
    @ProductId int ,
    @MealHistoryId int
AS
Begin
    Insert into MealFood(ProductId, RecipeId, MealHistoryId)
    values(@ProductId, NULL, @MealHistoryId);
End;
go

--Insert inside MealFood recipe
go
Create Procedure AddNewMealFoodRecipe
    @RecipeId int ,
    @MealHistoryId int
AS
Begin
    Insert into MealFood(ProductId, RecipeId, MealHistoryId)
    values(NULL, @RecipeId, @MealHistoryId);
End;
go

--Insert inside AdminRecommendations
go
Create Procedure AddNewAdminRecommendation
    @UserId int,
    @RecipeId int
AS
Begin
    Insert into AdminRecommendations(UserId, RecipeId)
    values(@UserId, @RecipeId);
End;
go

--Insert inside AdminCountingCalories
go
Create Procedure AddAdminCountingCalorie
    @UserId int,

```

```
        @Date date,  
        @Calories int  
AS  
Begin  
    Insert into AdminCountingCalories(UserId, Date, Calories)  
    values(@UserId, @Date, @Calories);  
End;  
go
```

Листинг 2 – Скрипт создания процедур

ПРИЛОЖЕНИЕ В

```
create index #User_index on Users(Login)
create index #Products_index on Products(Name)
create index #Recipes_index on Recipes(Name)
```