МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования

Тема «Программное средство «Diabeta.net» »

Исполнитель

студент 2 курса 6 группы Трошко Валерия Николаевна

(Ф.И.О.)

Руководитель работы Радиванович Д. А.

(подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

(подпись)

Минск 2021

**Содержание**

[Введение 4](#_Toc73099099)

[1 Обзор литературы и аналогов 5](#_Toc73099100)

[1.1 Glucose Buddy Diabetes Tracker 5](#_Toc73099101)

[1.2 Диабет 6](#_Toc73099102)

[1.3 Diabetes Tracker 7](#_Toc73099103)

[1.4 Аналогичность и различие программных средств 8](#_Toc73099104)

[2 Анализ требований к программному средству и разработка функциональных требований 9](#_Toc73099105)

[2.1 Определение требований к программному средству 9](#_Toc73099106)

[2.2 Описание средств разработки 9](#_Toc73099107)

[2.3 Описание функциональности программного средства 10](#_Toc73099108)

[2.4 Спецификация функциональных требований 10](#_Toc73099109)

[3 Проектирование программного средства 11](#_Toc73099110)

[3.1 Архитектура проектируемого программного средства 11](#_Toc73099111)

[3.1.1 Реализация MVVM 11](#_Toc73099112)

[3.1.2 Схема и описание авторизации 12](#_Toc73099113)

[3.1.3 Построение графиков в WPF 12](#_Toc73099114)

[3.1.4 Подключение к БД 13](#_Toc73099115)

[3.2 Разработка базы данных 13](#_Toc73099116)

[4 Описание интерфейсов программного средства, его тестирование и анализ полученных данных 16](#_Toc73099117)

[4.1 Описание интерфейсов программного средства 16](#_Toc73099118)

[4.1.1 Вход в качестве пользователя 17](#_Toc73099119)

[4.1.2 Вход в качестве администратора 20](#_Toc73099120)

[4.2 Тестирование программного средства и анализ полученных данных 21](#_Toc73099121)

[5 Руководство по установке и использованию 24](#_Toc73099122)

[5.1 Установка программного средства 24](#_Toc73099123)

[5.2 Руководство пользователя по работе с программным средством 25](#_Toc73099124)

[Заключение 28](#_Toc73099125)

[Список использованных источников 29](#_Toc73099126)

[Приложение А 30](#_Toc73099127)

[Приложение Б 31](#_Toc73099128)

# Введение

В современном мире информационные технологии развиваются с невероятной скоростью. Так, за относительно небольшой промежуток времени истории человечества они стали играть немаловажную роль во всех сферах жизни общества. С развитием технологий человеку становятся доступны все новые и новые возможности, о которых ранее приходилось только мечтать.

Стоит отметить, среди трендов последних лет очень четко выделяется направление самоорганизации. Приложения для повышения продуктивности занимают все более высокие позиции в рейтингах по количеству скачиваний в PlayMarket и AppStore. Тренд очевиден: 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. А особенно те люди, кому поставлен такой неприятный диагноз, как «Сахарный диабет».

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. А когда это ещё усугубляется постоянным приёмом лекарств, то следить за этим становится ещё труднее.

Раньше для этого пользовались специальными таблицами, где искали продукты, смотрели количество калорий в 100г 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.

При написании курсового проекта одной из задач являлся простой и доступный дизайн как для пользователей, владеющих техническим образованием, так и нет.

Целью данной работы является создание приложения, которое работает с личными данными пользователя. Расчет дневной нормы калорий пользователя зависит от его параметров и целей. А также функционал приложения позволяет:

* добавлять/изменять/удалять данные из базы данных;
* получить статистику веса пользователя и уровня сахара в крови;
* контролировать дневную норму калорий пользователя;
* контролировать водный баланс пользователя;
* контролировать выпитые медикаменты и сделанные уколы

Структура приложения: приложение, написанное на С#. Оно устанавливается на ПК и представляет собой интерфейс.

ПК приложение для оптимальной работы требует:

* установленный Microsoft SQL Server 2019;
* установленный .Net framework 5.0.202 или более поздней версии.

# Обзор литературы и аналогов

При изучении программ-прототипов было выявлено несколько схожих по функциональности программ. Все программные средства представлены в качестве приложений для поддержания здоровья. Результат анализа представлен ниже.

## Glucose Buddy Diabetes Tracker

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

Интерфейс приложения представлен на рисунке 1.1.

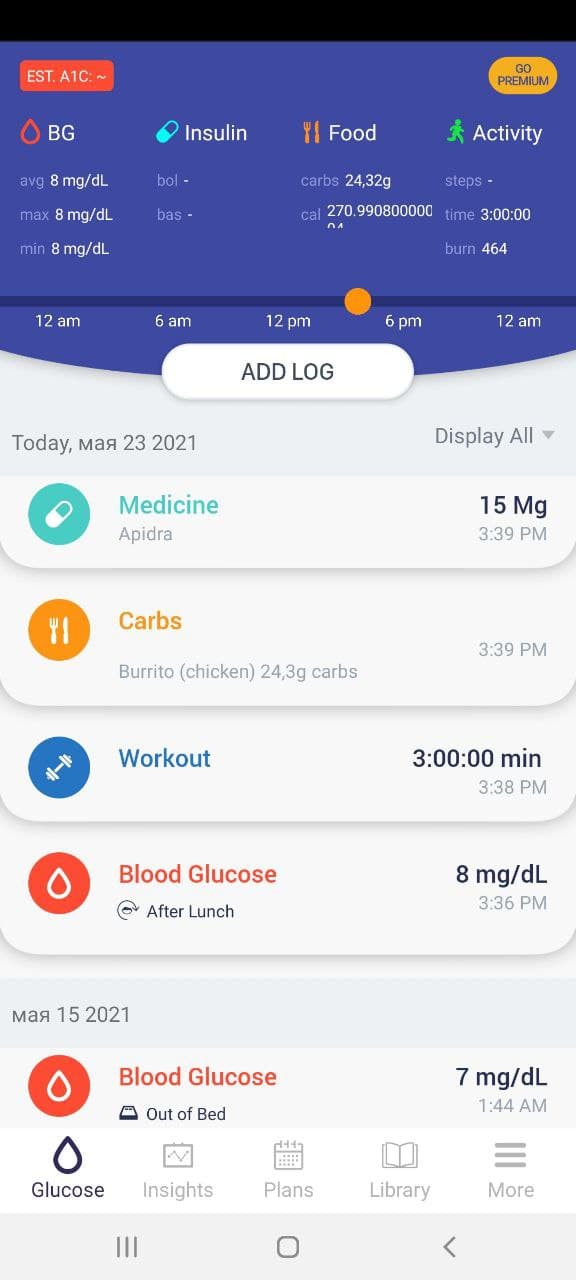


Рисунок 1.1 − Интерфейс приложения Glucose Buddy Diabetes Tracker

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.

## Диабет

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

Внешний вид программы представлен на рисунке 1.2.

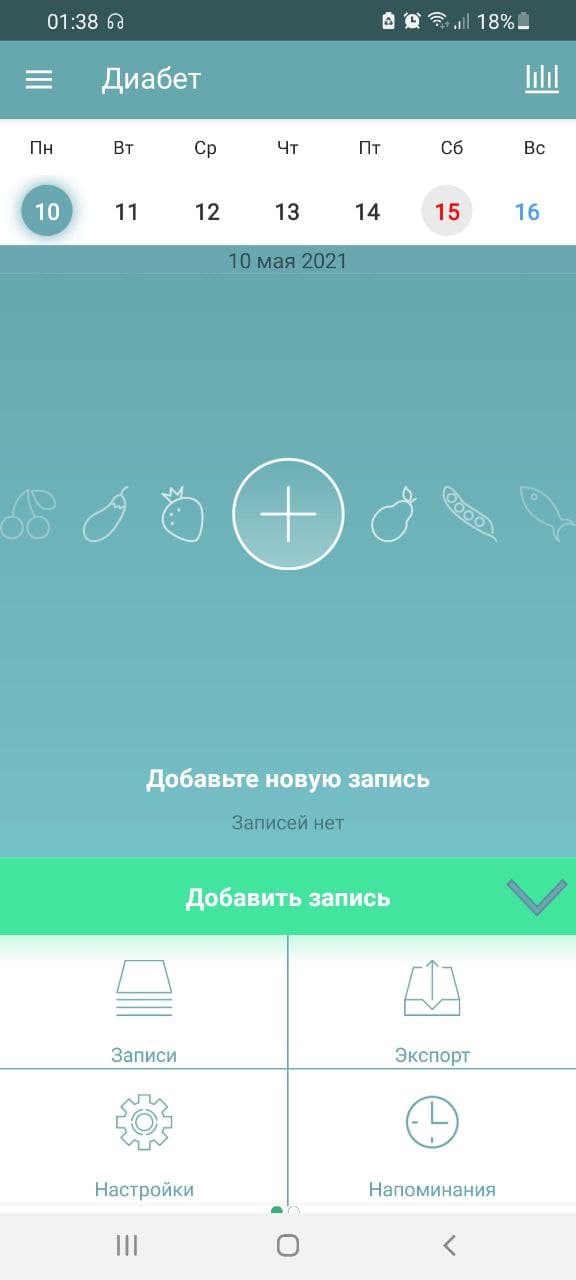


Рисунок 1.2 — Интерфейс приложения Диабет

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.

## 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. Функционал:

* 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

Главное окно программы представлено на рисунке 1.3.



Рисунок 1.3 — Интерфейс приложения Diabetes Tracker

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.

Анализируя ранее приведённые примеры, можно составить основные функциональные особенности для подобного типа приложений:

* Высчитывание нормы КБЖУ исходя из данных человека;
* Авторизация пользователя;
* База продуктов;
* Контроль воды, лекарств и инсулина;
* Статистика пользователя;

Как сказано выше, программа должна быть предназначена для различной аудитории пользователей. Это значит, что приложение должно быть простое и иметь доступный дизайн.

Все эти пункты и были учтены при выполнении данного курсового проектирования.

# Анализ требований к программному средству и разработка функциональных требований

Для того чтобы приступить к этапу разработки проекта, необходимо четко сформулировать функциональные требования к программному средству.

## 2.1 Определение требований к программному средству

Функционально ПС должно выполнять следующие задачи:

* регистрация и авторизация пользователей;
* сохранение информации в базе данных;
* для администратора одобрение или отклонение предложенных продуктов;
* для пользователя изменение личных данных;
* для пользователя учёт выпитых медикаментов,
* для пользователя водного и пищевого баланса;
* уведомление пользователя об высоком или низком уровне сахара;
* для администратора добавление новых продуктов и рецептов.

## 2.2 Описание средств разработки

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2019;
* программная платформа .NET Framework 5.0.202;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология ADO.NET;
* 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. В основе графической технологии WPF лежит мощная инфраструктура, основанная на DirectX. Это является одним из основных отличий WPF от более ранней технологии создания пользовательских интерфейсов – Windows Forms.

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

Если различные механизмы работы с БД в С#. Т.к. моя база данных небольшая, мой выбор пал на ADO.Net. ADO.Net предоставляет самый прямой способ доступа к данным в .Net Framework.

## 2.3 Описание функциональности программного средства

Описание функциональности программного средства представлено с помощью UML-диаграммы вариантов использования (Приложение А).

Спецификация функциональности программного средства представлена в следующем пункте

## 2.4 Спецификация функциональных требований

Для функциональности ПС необходимо создание базы данных для хранения информации приложения. Подробно о базе данных описано в главе 3.

В программном средстве «Diabeta.net» при запуске необходимо реализовать регистрацию и авторизацию пользователей для дальнейшего использования приложения. Для авторизации входными параметрами являются логин и пароль пользователя, которые содержатся в базе данных. Для того чтобы зарегистрироваться необходимо ввести логин, пароль, фамилию, имя, возраст, рост, вес, пол, желаемый уровень сахара, цель и уровень подвижности. Введенные данные, успешно прошедшие валидацию, заносятся в базу данных, а пользователь получает доступ к приложению.

При авторизации администратора он может на странице «*Одобрить*» просматривать продукты, добавлять и удалять, при нажатии на кнопку «*Продукты*» может просматривать все продукты и добавлять новые. На странице «*Рецепты»* просматривать имеющиеся рецепты, при нажатии на «*Добавить рецепт*» администратор может создать и добавить новый рецепт.

При авторизации пользователя он может на главной странице просматривать и отмечать все съеденные продукты и блюда, весь инсулин и все таблетки, которые употреблял, уровень сахара и выпитые стаканы воды. На странице «*Инфо»* он может просмотреть личную информацию и редактировать её. На странице «*Рецепты*» он может увидеть существующие рецепты и просмотреть их. На страницах статистик пользователь может просмотреть статистику по весу и статистику по уровню сахара в крови.

При авторизации администратора он может на странице «*Товары*» просматривать все товары от всех продавцов и удалять их. На странице «*Заказы»* просматривать заказы и удалять их. На странице «*Продавцы*» просматривать всех продавцов и удалять их. На странице «*Клиенты*» просматривать всех клиентов и удалять их.

Кнопка для выхода из аккаунта возвращает пользователя к окну авторизации, а кнопка для выхода из приложения закрывает приложение.

Все данные о продуктах, рецептах, таблетках и сахаре в крови должны заноситься в базу данных в соответствующие таблицы.

# Проектирование программного средства

## Архитектура проектируемого программного средства

При создании программного средства использовался архитектурный паттерн Model-View-ViewModel (MVVM). Он был представлен архитектором программного обеспечения компании Microsoft Джоном Госсманом в 2005 году. Одной из главных целей шаблона является отделение и абстрагирование видимого пользовательского интерфейса от логики представления. MVVM состоит из трех компонентов: модель (Model), представление (View) и модель представления (ViewModel). Наглядное представление паттерна представлено на рисунке 3.1.

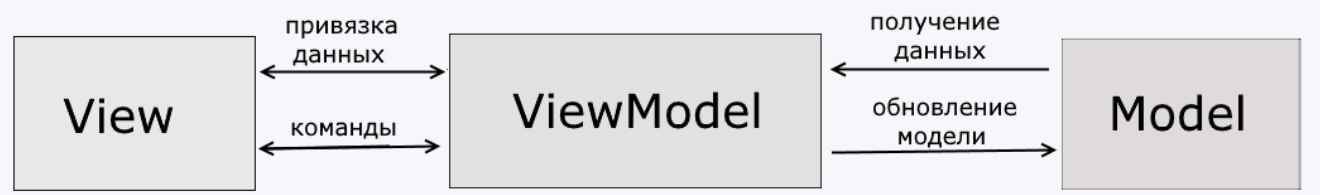


Рисунок 3.1 − Наглядное представление MVVM

Модель (Model) — это субъект, который описывает используемые в приложении данные. Это может быть что угодно, будь то обычный продукт или же сложный рецепт. Представление (View) — это графический элемент управления или набор элементов управления, отвечающих за отображение данных на экране. Модель представления (ViewModel) — это и есть та магия, стоящая за всем. Она содержит логику пользовательского интерфейса, команды, события и ссылку на модель. В MVVM модель преставления не обязана обновлять непосредственно данные представления. Благодаря мощному механизму привязки, представленному в WPF, пользовательский интерфейс обновляется, как только обновляется модель преставления. Для этого модель преставления должна реализовывать интерфейс INotifyPropertyChanged и запустить событие PropertyChanged [4].

### **3.1.1** **Реализация MVVM**

Для работы с паттерном использовалась библиотека DevExpressMVVM. Для реализации паттерна файлы программы были распределены по соответствующим пространствам имен и реализованы соответствующие функции. На рисунке 3.1 видим, что класс MainViewModel наследует класс ViewModelBase, который лежит в пространстве имен DevExpress.Mvvm. Метод

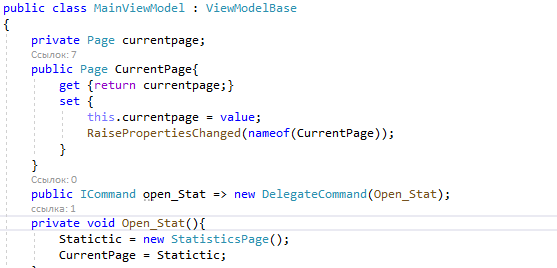


Рис. 3.1 – Пример использования DevExpressMVVM

### **3.1.2 Схема и описание авторизации**

Схема авторизации представлена в приложении B. Сразу после запуска программу пользователь попадает в окно авторизации. Если поля логина и пароля будут пустыми, в окне будет выведено сообщение об ошибке. После программа обращается к базе данных и проверяет существует ли пользователь с соответствующим логином и паролем. Если пользователь не найден, будет выведено сообщение об ошибке, в противном случае вход в приложение.

### **3.1.3 Построение графиков в WPF**

NuGet предоставляет достаточное количество библиотек для построения графиков. Мой выбор упал на библиотеку LiveCharts. Она очень проста в использовании (пример заполнения рисунок 3.2).

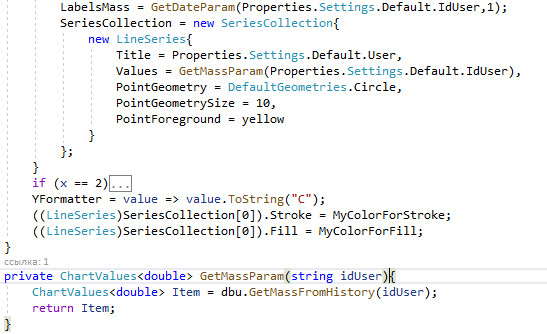


Рис. 3.2 – Пример использования LiveCharts

### **3.1.4** **Подключение к БД**

Если различные механизмы работы с БД в С#. Т.к. моя база данных небольшая, мой выбор пал на ADO.Net. ADO.Net предоставляет самый прямой способ доступа к данным в .Net Framework. В Приложении Г продемонстрирован листинг одного из запросов к БД.

## Разработка базы данных

Для обеспечения работоспособности программного средства была разработана база данных, что являлось обязательным условием курсового проекта. База данных была тщательно сконструирована до начала написания самого приложения. Для ее создания использовалась реляционная система управления базами данных (РСУБД) MS SQL.

Таким образом была разработана база данных KP\_DataBase, которая включает в себя таблицы Users, Daily\_Food, Recipe, Prod\_Rec, Products, Type\_of\_Food, Daily\_Insulin, Type\_of\_Insulin, Daily\_Pill, Daily\_Cal, Daily\_Water, History, History\_Blood\_Sugar, Products\_Awaiting\_Approval. Схема базы данных представлена на рисунке 3.2

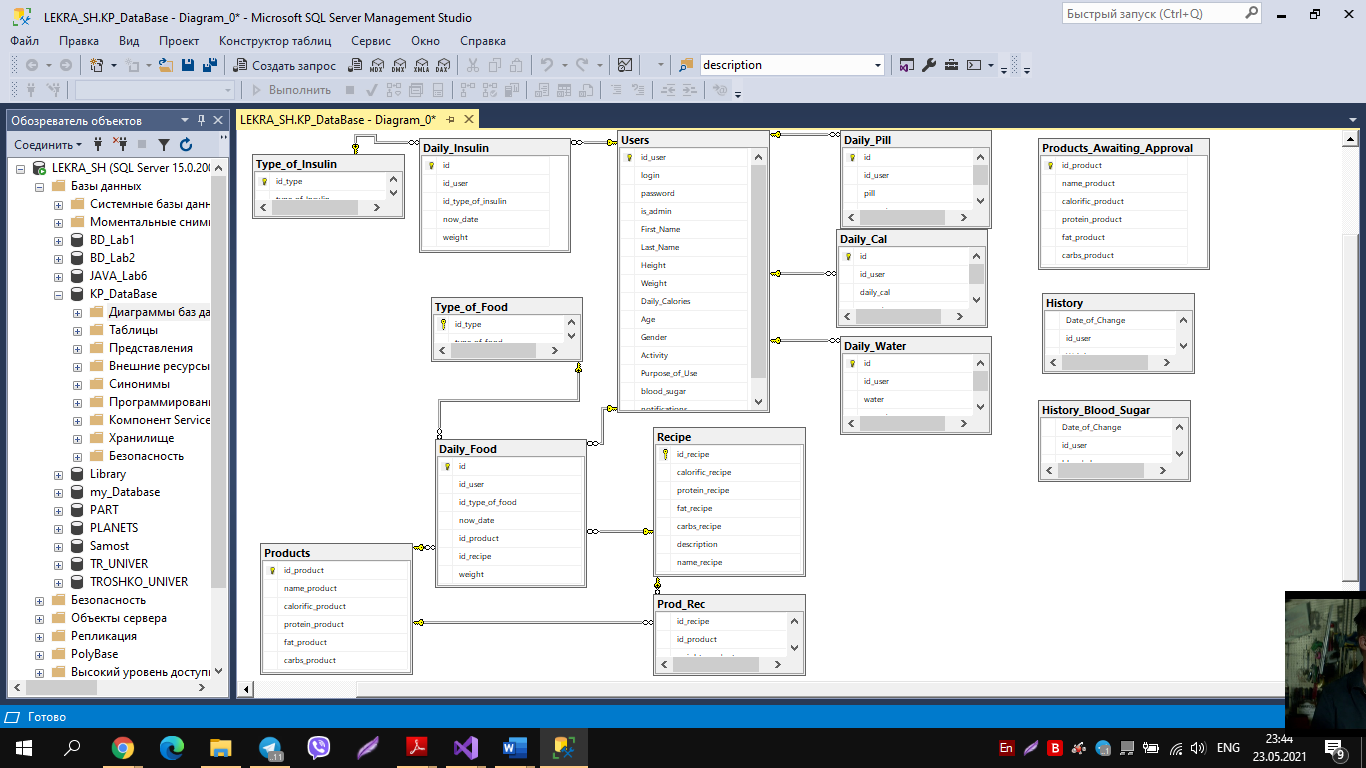


Рисунок 3.2 — Схема базы данных KP\_DataBase

Для хранения информации о пользователях была создана таблица Users. Включает 15 столбцов: id\_user – id пользователя, login – логин пользователя, password – пароль пользователя, при записи в базу данных хэшируется, isadmin – 0 – пользователь, 1 - администратор, First\_Name – имя пользователя, Last\_Name – фамилия пользователя, Height – рост, Weight – вес, Daily\_Calories – дневная норма калорий, Age – возраст, Gender – пол, Activity – коэффициент активности, Purpose\_of\_Use – цель использования, blood\_sugar – коэффициент уровня сахара в крови, notifications – поле доступа для уведомлений.

Таблица Daily\_Food используется для хранения съеденной пищи пользователем в течении дня.

Таблица Recipe используется для хранения информации о рецептах. Включает 7 столбцов: id\_recipe – id рецепта, name\_ recipe – название рецепта, calorific\_ recipe – калорийность рецепта на 100г, protein\_ recipe - белки рецепта на 100г, fat\_ recipe – жиры рецепта на 100г, carbs\_ recipe - углеводы рецепта на 100г, description – способ приготовления.

Таблица Prod\_Rec используется для связи между таблицами Recipe и Products. Используется для определения того, какие ингредиенты входят в состав блюда. Включает 3 столбца: id\_recipe – id рецепта, id\_product – id продукта, weight\_product – вес продукта, необходимого для рецепта.

Таблица Products используется для хранения продуктов. Включает 6 столбцов: id\_product – id продукта, name\_product – название продукта, calorific\_product – калорийность продукта на 100г, protein\_ product – белки продукта на 100г, fat\_ product – жиры продукта на 100г, carbs\_ product - углеводы продукта на 100г.

Таблица Type\_of\_Food используется для хранения данных о типах приема пищи. Включает 2 столбца: id\_type – id приема пищи, type\_of\_food – тип приема пищи (завтрак, обед, ужин, перекус).

Таблица Daily\_Cal хранит оставшееся число калорий пользователя на день.

Таблица Daily\_Water используется для хранения выпитой воды пользователем в течении дня.

Таблица Daily\_Pill используется для хранения выпитых медикаментов пользователем в течении дня.

Таблица Daily\_Insulin используется для хранения сделанных уколов пользователем в течении дня.

Таблица Type\_of\_Insulin используется для хранения данных о типах уколов. Включает 2 столбца: id\_type – id приема инсулина, type\_of\_insulin – тип укола инсулина (ночной, дневной).

Таблица Product\_Awaiting\_Approval используется для хранения добавленных пользователем новых продуктов. Элементы жду подтверждения администратора.

Таблица History используется для хранения истории веса пользователя. Включает 3 столбца: Date\_of\_Change – дата изменения веса,id\_user- id пользователя, Weight – обновленный вес. Таблица History автозаполняется, когда срабатывает триггер таблицы Users.

Таблица History\_Blood\_Sugar используется для хранения истории уровня сахара пользователя. Включает 3 столбца: Date\_of\_Change – дата изменения веса,id\_user- id пользователя, blood\_shugar – обновленный Уровень сахара. Таблица History автозаполняется, когда срабатывает триггер таблицы Users.

В базе данных между таблицами Users и Daily\_Insulin реализована связь один-ко-многим, т. к. одному пользователю может соответствовать несколько записанных уколов, а каждый укол в таблице Daily\_Insulin соответствует только одному пользователю. Аналогичная связь реализована в таблицах Users и Daily\_Pill и ещё много где. Обобщенная структура проекта

В общей сложности проект включает в себя 4 папки. Его структура представлена на рисунке 3.4.

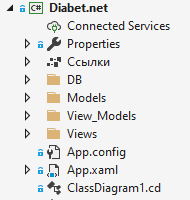


Рисунок 3.4 − Структура проекта

В папке Models хранятся классы, описывающие логическую обработку данных проекта.

Папка View\_Models содержит в себе классы со свойствами и командами, которые связаны механизмом привязки с объектами папки Views.

В папке Views содержатся элементы пользовательского интерфейса, которые пользователь сможет увидеть на экране.

Содержимое папки DB представляет из себя все взаимодействия с базами данных.

Взаимосвязь между классами хорошо видна в диаграмме классов (Приложение Б).

Приложение включает в себя 13 окон и 12 страниц. Связи между ними представлены на рисунке 3.5.

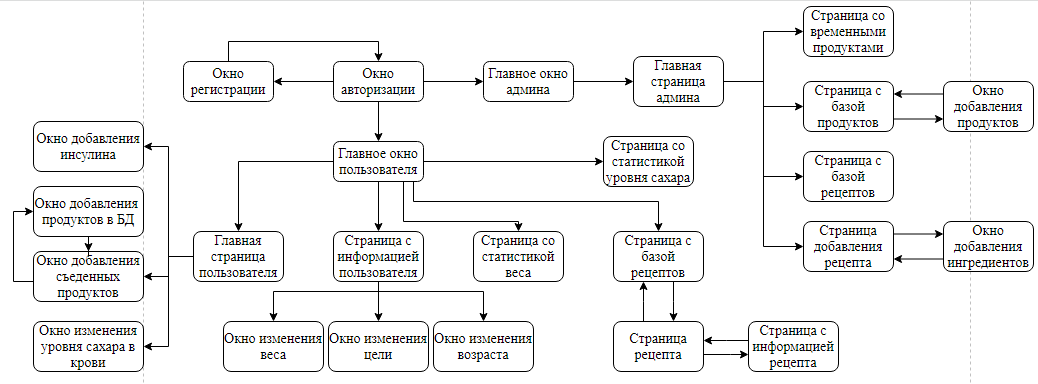


Рис. 3.5 – Схема связи между окнами

# Описание интерфейсов программного средства, его тестирование и анализ полученных данных

## Описание интерфейсов программного средства

Сразу после запуска приложения перед пользователем открывается окно авторизации, в которое требуется ввести логин и пароль. Внешний вид окна представлен на рисунке 4.1.

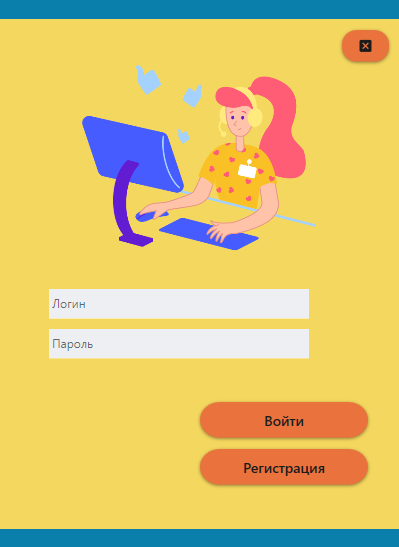


Рисунок 4.1 − Внешний вид окна авторизации

Из окна авторизации пользователи, которых нет в базе данных приложения, могут зарегистрироваться, нажав на кнопку «Регистрация». После того, как это произойдет, откроется окно регистрации новых пользователей, интерфейс которого представлен на рисунке 4.2.

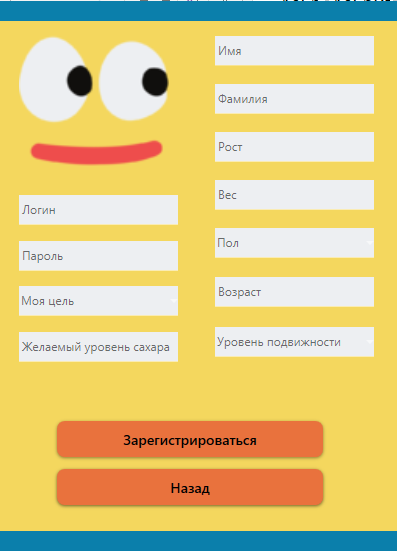


Рисунок 4.2 − Интерфейс окна регистрации

Также в окне регистрации есть функция возврата к окну авторизации, которую реализует элемент управления с надписью «Назад».

### **4.1.1 Вход в качестве пользователя**

После успешной авторизации открывается главное окно приложения, которое встречает пользователя. С главной страницы которого мы можем добавить съеденную еду, контролировать воду, медикаменты и инсулин, вводить данные об уровне сахара в крови, включать/выключать уведомления, а так же перейти на другие страницы приложения. Внешний вид главного окна для пользователя представлен на рисунке 4.3.

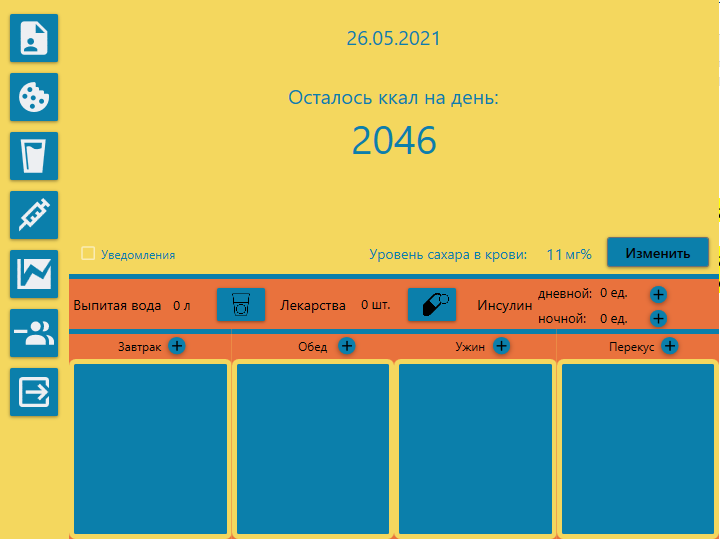


Рисунок 4.3 − Внешний вид главного окна пользователя

При нажатии на кнопки в левой боковой панели открываются различные страницы «Личный кабинет», «Рецепты», «Статистика уровня сахара в крови», «Статистика веса», а так же можно выйти из аккаунта или со всего приложения. На рисунках 4.4 – 4.7 представлен внешний вид этих страницах.

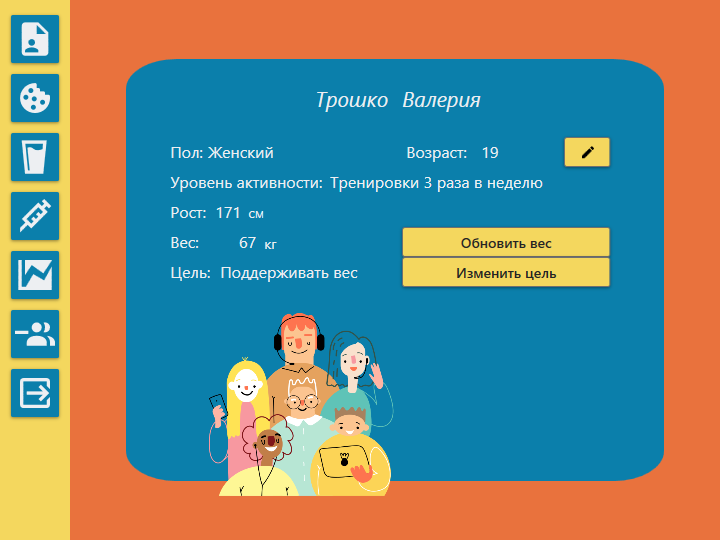


Рисунок 4.4 − Страница «Личный кабинет»

В «Личном кабинете» пользователь может просмотреть информацию о себе, а так же внести изменения веса, возраста, цели.

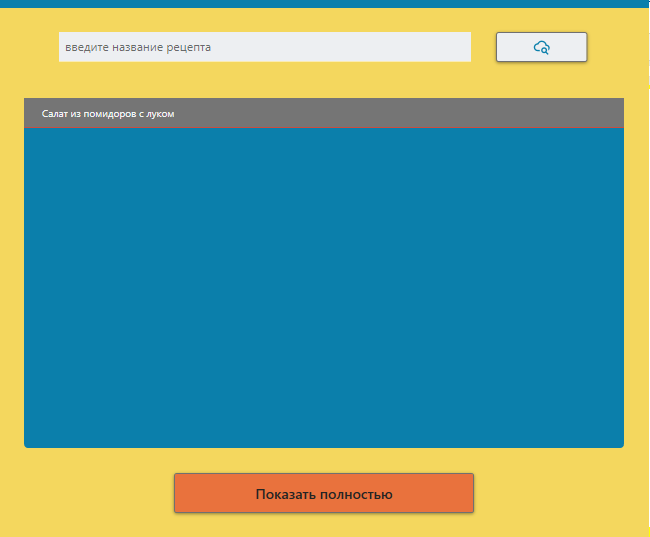


Рисунок 4.5 − Страница «Рецепты»

На странице «Рецепты» пользователь может выбрать и просмотреть различные рецепты, при нажатии кнопки «Показать полностью» мы может ознакомится с рецептом, а точнее с его КБЖУ, ингредиентами и способом приготовления рис. 4.6.

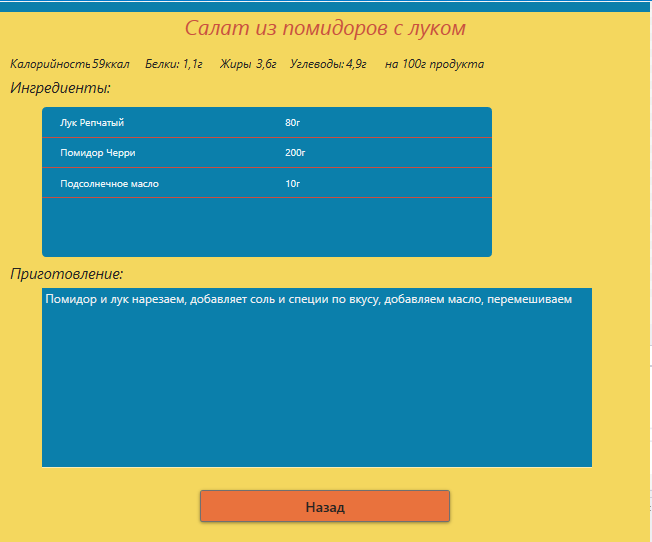


Рисунок 4.6 − Страница «Информация рецепта»

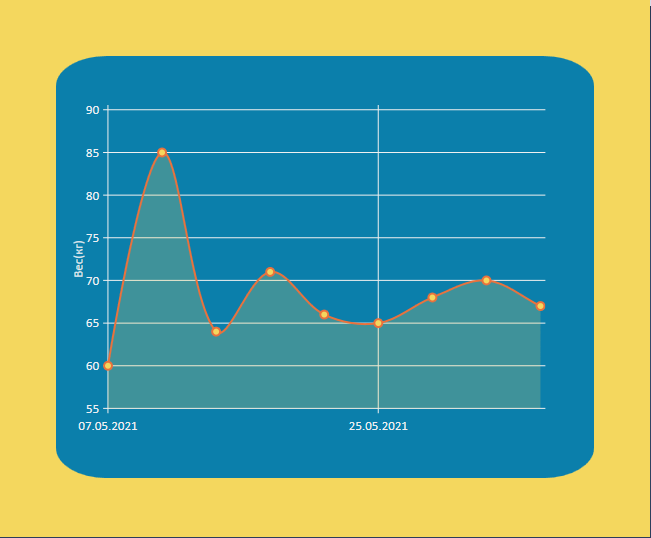


Рисунок 4.7 − Страница «Статистика веса»

Страницы «Статистика веса» и «Статистика уровня сахара в крови» построены по одному принципу.

### **4.1.2 Вход в качестве администратора**

После успешной авторизации открывается главное окно приложения (рис. 4.8), которое встречает администратора.



Рисунок 4.8 − Внешний вид главного окна администратора

На рисунке 4.9 продемонстрирована таблица админа, где находятся продукты, которые ожидают подтверждения.

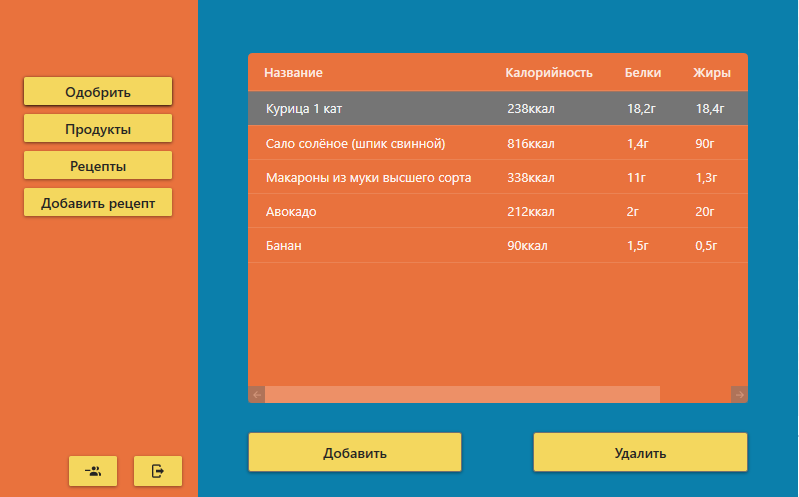


Рисунок 4.9 − Страница продуктов для одобрения

Админ контролирует базу продуктов и базу рецептов (рис. 4.10). Соответственно он может пополнять их.

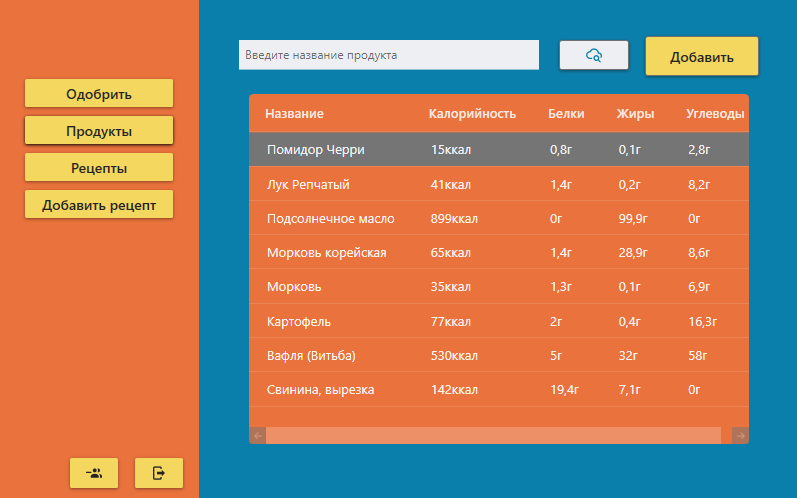


Рисунок 4.10 − Страница базы продуктов

## Тестирование программного средства и анализ полученных данных

Для проверки работоспособности программного средства было проведено его тщательное тестирование, в ходе которого осуществлялась проверка заполняемости базы данных, корректность извлекаемых из базы данных значений, а также реакция программы на вводимые, заведомо неверные, данные. Было проработано множество ситуаций, которые могут возникнуть в ходе работы с программным средством.

Так, при входе осуществляется проверка на действительные логин и пароль. Например, пользователь ввел неверный пароль или логин (результат на рис 4.11):

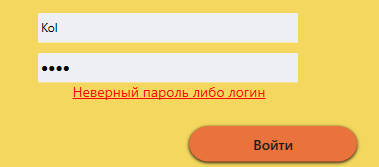


Рисунок 4.11 − Результат входа с неверным логином или паролем

Или вы упустили пустое поле и нажали войти (результат на рис. 4.12):

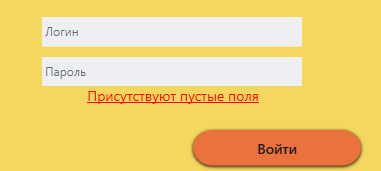


Рисунок 4.12 − Пустые поля при попытке авторизации

При рассмотрении исключительных ситуаций в окне регистрации также была организована проверка вводимых пользователем данных. Так, например, если при регистрации пароль будет меньше 8 символов, то (результат на рис. 4.13):

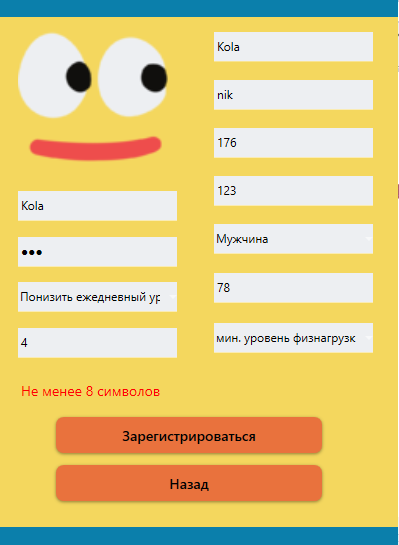


Рисунок 4.13 − Короткий пароль

При попытке изменить уровень сахара, вес, цель, возраст, но поле изменения оставить пустым (результат на рис 4.14):

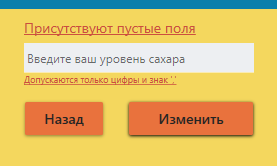


Рисунок 4.14 − Пустые поля при попытке изменить данные

Или при попытке изменения ввести значение идентичное прошлому (результат на рис 4.15):

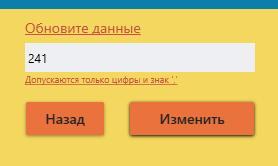


Рисунок 4.15 − Новые данные равные старым

И вот еще пример, где мы хотели добавить продукт, при этом мы не вводили его массу (результат рис. 4.16):

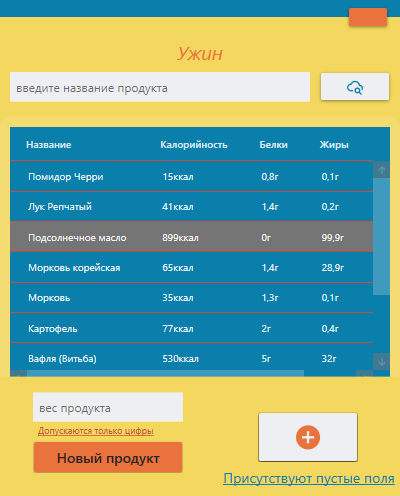


Рисунок 4.16 − Пустые поле веся при добавлении блюда в рацион

# Руководство по установке и использованию

## 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 5.1.

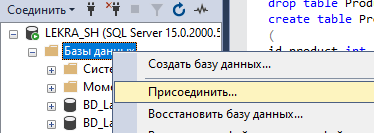


Рисунок 5.1 − Выбор пункта «Присоединить»

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

После всех проделанных шагов в раскрывающемся списке «Базы данных» обозревателя объектов должна появиться база данных Diabet.net, как на рисунке 5.2.

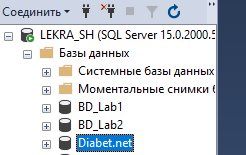


Рисунок 5.2 − Появление базы данных Diabet.net

Далее необходимо запустить установщик приложения и установить на свой компьютер. На рисунке 5.3 представлен установщик. После чего на рабочем столе будет создан ярлык для быстрого запуска приложения.



Рисунок 5.3 − 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 5.4.

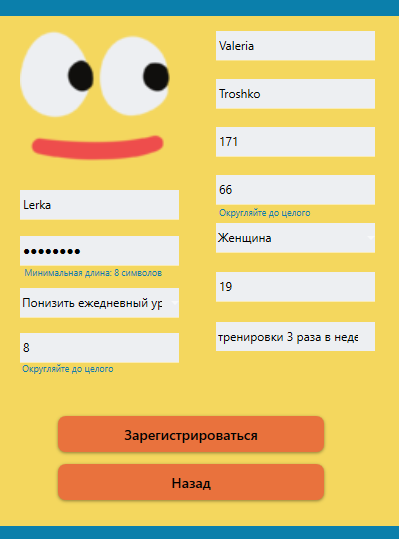


Рисунок 5.4 − Пример правильного заполнения полей окна регистрации

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. 5.5).



Рисунок 5.4 − Пример правильного заполнения полей окна регистрации

В случае, если в базе нет блюда, которое пользователь съел, он может сам создать его заполнив специальную форму (рис. 5.5).

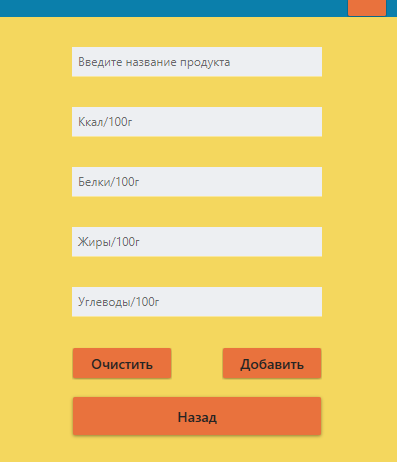


Рисунок 5.5 − Форма для создания нового блюда

На странице рецептов, пользователь может выбрать интересующее его блюди и просмотреть информацию о нём (рис. 5.6).

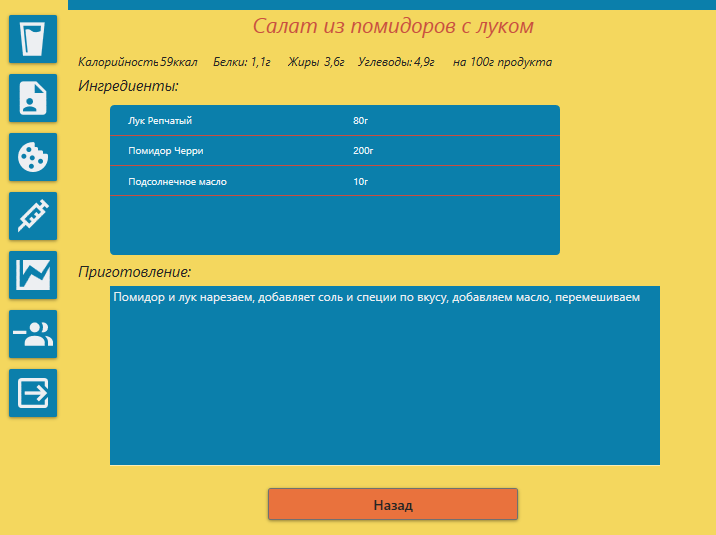


Рисунок 5.6 − Информация о выбранном рецепте

Для просмотра своей истории веса или уровня сахара в крови, пользователь может зайти на страницу статистики (рис 5.7).

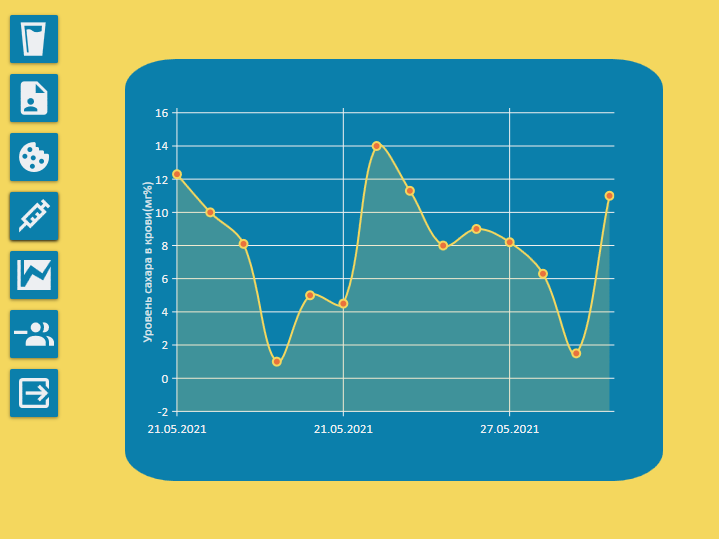


Рисунок 5.7 − Пример правильного заполнения полей окна регистрации

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.

В ходе выполнения курсовой работы было выполнено следующее:

- разработка базы данных для хранения информации;

- разработка архитектуры приложения;

- разработка функциональных части приложения;

- разработка пользовательского интерфейса;

- написание исходного кода приложения;

- тестирование приложения.

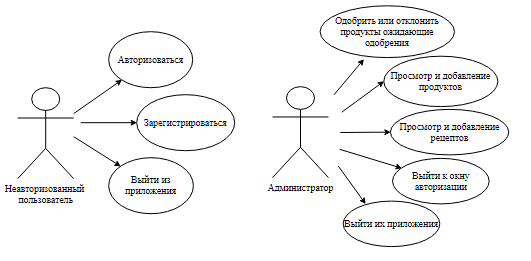
В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает, верно, а требования технического задания выполнены в полном объеме.

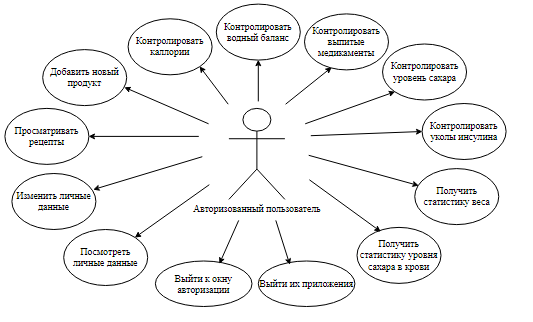
Само приложение – первый опыт в данной области. Естественно, существует ряд приложений, имеющих функционал и дизайн в разы лучше. Подводя итог всей курсовой работе, можно сделать выводы, что в разработке приложений мне предстоит еще долгий путь.

# Список использованных источников

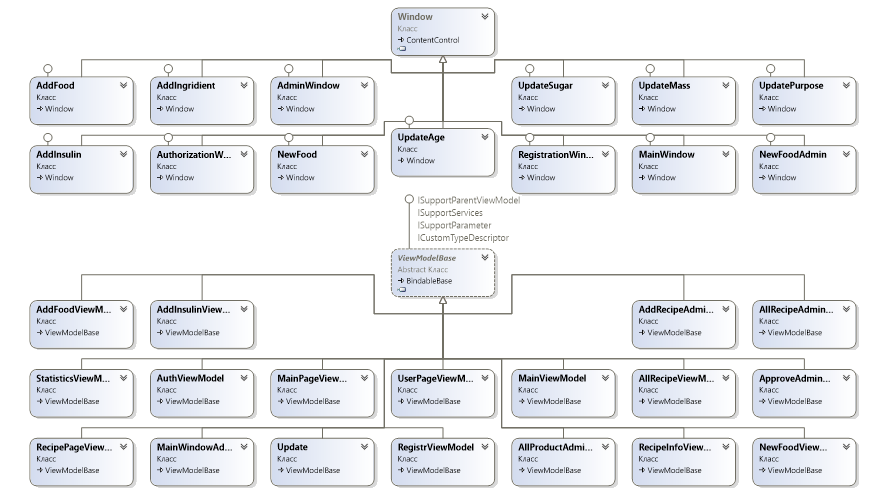
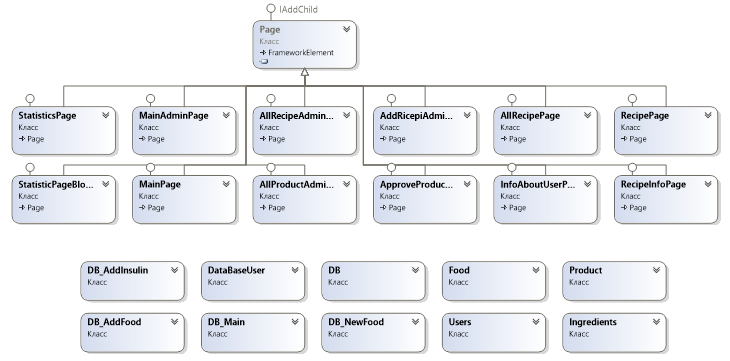
1. Информационный сайт: Диабет [Электронный ресурс]. Режим доступа: https://play.google.com/store/apps/details?id=ru.hintsolutions.diabets&hl=ru&gl=US. Дата доступа: 24.05.2021.
2. Информационный сайт: GlucoseBuddy [Электронный ресурс]. Режим доступа: https://play.google.com/store/apps/details?id=com.skyhealth.glucosebuddyfree&hl=ru&gl=US. Дата доступа: 24.05.2021.
3. 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.
4. Garofallo, R. Building Enterprise Applications with Windows Presentation Foundation and the Model View ViewModel Pattern / R. Garofallo − California: O’Reilly Media, − 2011 − 220 p.
5. Информационный сайт: базы данных [Электронный ресурс]. Режим доступа:[inform.sch117nn.edusite.ru/DswMedia/klyaks\_netuchitelyutestyibazyidannyix.htm](http://inform.sch117nn.edusite.ru/DswMedia/klyaks-_netuchitelyutestyibazyidannyix.htm) – Дата доступа: 24.05.2021.
6. 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

# Тестирование

Для проверки работоспособности программного средства было проведено его тщательное тестирование, в ходе которого осуществлялась проверка заполняемости базы данных, корректность извлекаемых из базы данных значений, а также реакция программы на вводимые, заведомо неверные, данные. Было проработано множество ситуаций, которые могут возникнуть в ходе работы с программным средством.

Так, при входе осуществляется проверка на действительные логин и пароль. Например, пользователь ввел неверный пароль или логин (результат на рис 4.1):

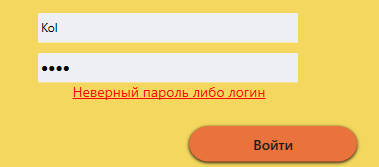


Рисунок 4.1 − Результат входа с неверным логином или паролем

Или вы упустили пустое поле и нажали войти (результат на рис. 4.2):

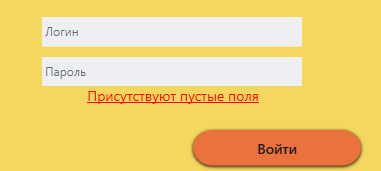


Рисунок 4.2 − Пустые поля при попытке авторизации

При регистрации пользователь обязан заполнить все поля, иначе форма не пройдёт проверку на пустые поля (результат на рис. 4.3):

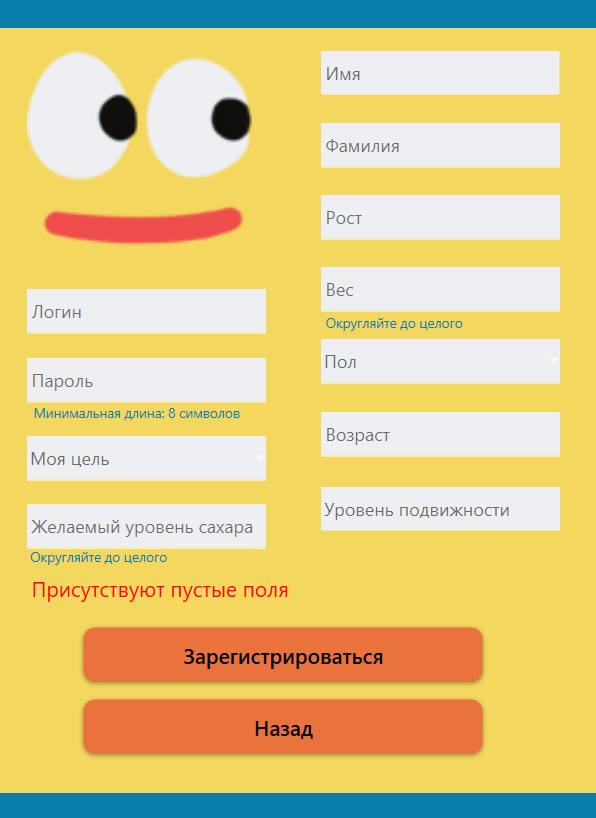


Рисунок 4.3 – Проверка на пустоту полей

При рассмотрении исключительных ситуаций в окне регистрации также была организована проверка вводимых пользователем данных. Так, например, если при регистрации пароль будет меньше 8 символов, то (результат на рис. 4.3):

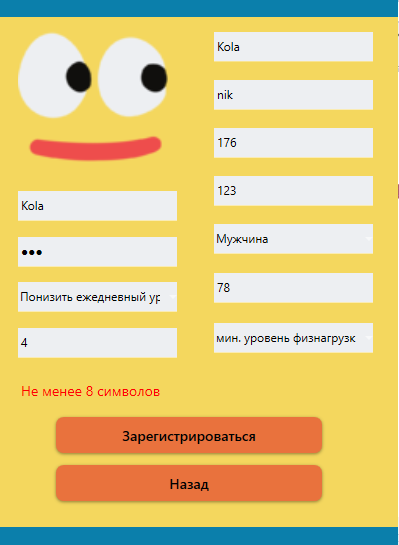


Рисунок 4.3 − Короткий пароль

Если при регистрации пользователь выберет логин который уже существует, у него не получится зарегистрироваться, пока он его не изменит (результат на рис. 4.4):

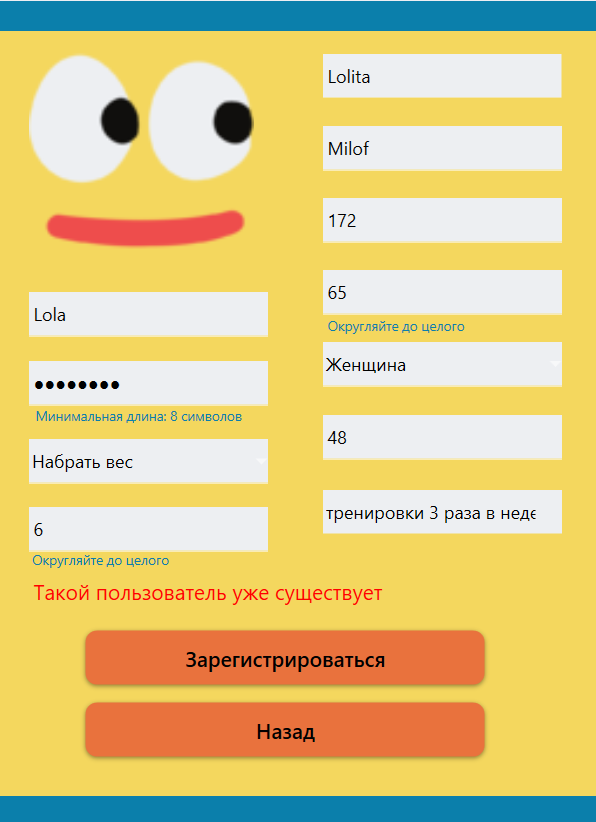


Рисунок 4.4 – Логин уже существует

При попытке изменить уровень сахара, вес, цель, возраст, но поле изменения оставить пустым (результат на рис. 4.4):

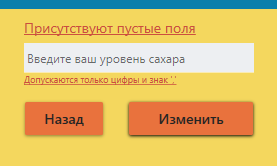


Рисунок 4.4 − Пустые поля при попытке изменить данные

Или при попытке изменения ввести значение идентичное прошлому (результат на рис 4.5):

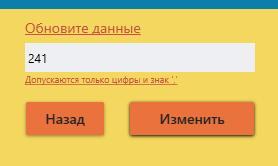


Рисунок 4.5 − Новые данные равные старым

И вот еще пример, где мы хотели добавить продукт, при этом мы не вводили его массу (результат рис. 4.6):

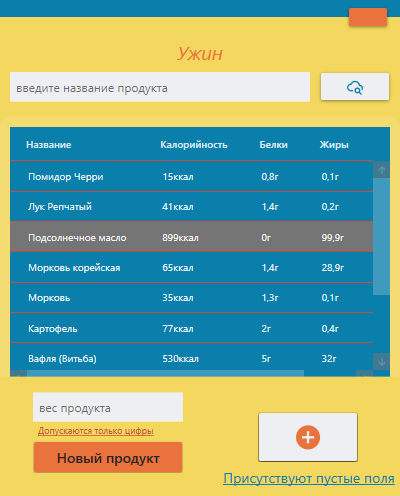


Рисунок 4.6 − Пустые поле веса при добавлении блюда в рацион

Если администратор решить одобрить продукт, который был ранее предложен пользователем, но в базе продуктов такой уже есть, то у него это не получится (результат на рис 4.8):



Рисунок 4.8 – Таблица одобрения после добавления одинакового продукта

При попытке администратора не выбрав элемент и попробовать его добавить или удалить (результат на рис. 4.9):



Рисунок 4.9 – Таблица одобрения при попытке пустого добавления

Админ может удалить продукт из базы, если он не пользует спросом у пользователя, но если же пользователи им пользуются, то даже администратор его не во власти удалять (результат на рис. 4.10):

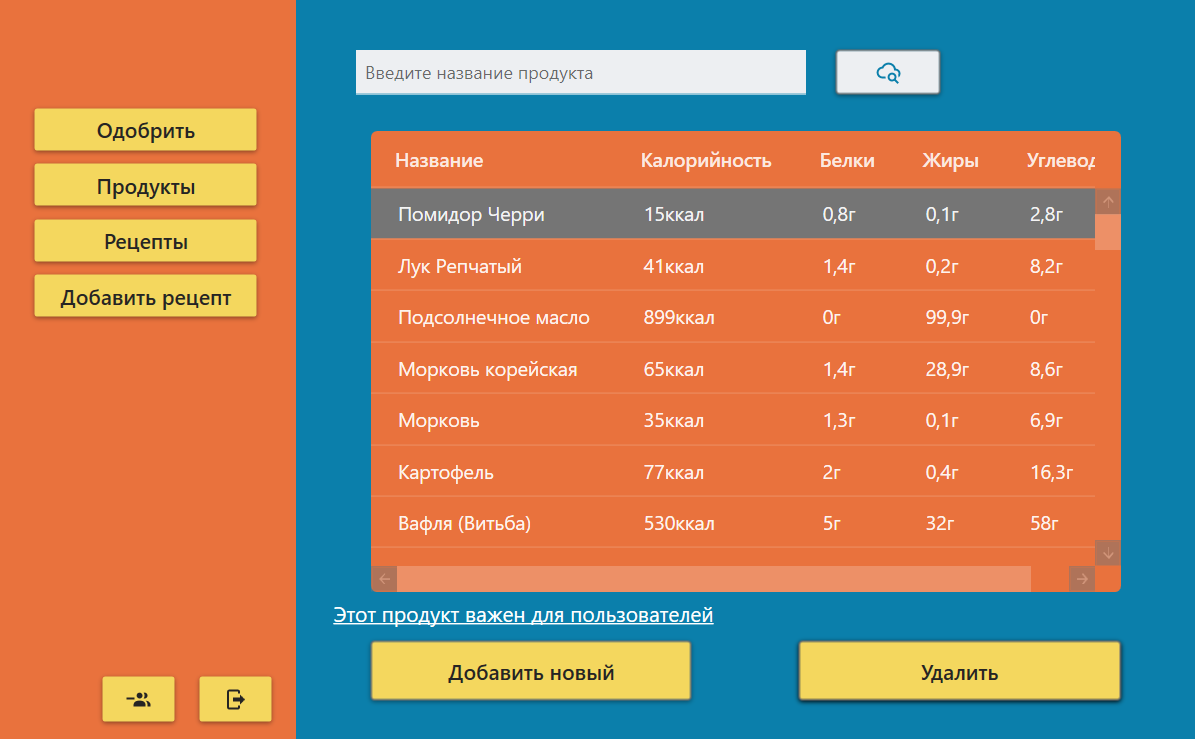


Рисунок 4.9 – Таблица одобрения при попытке пустого добавления

Такие же правила и с таблицей рецептов, т.е. если кто-нибудь этим рецептом пользуется, то удалить его не выйдет (результат на рис. 4.10):

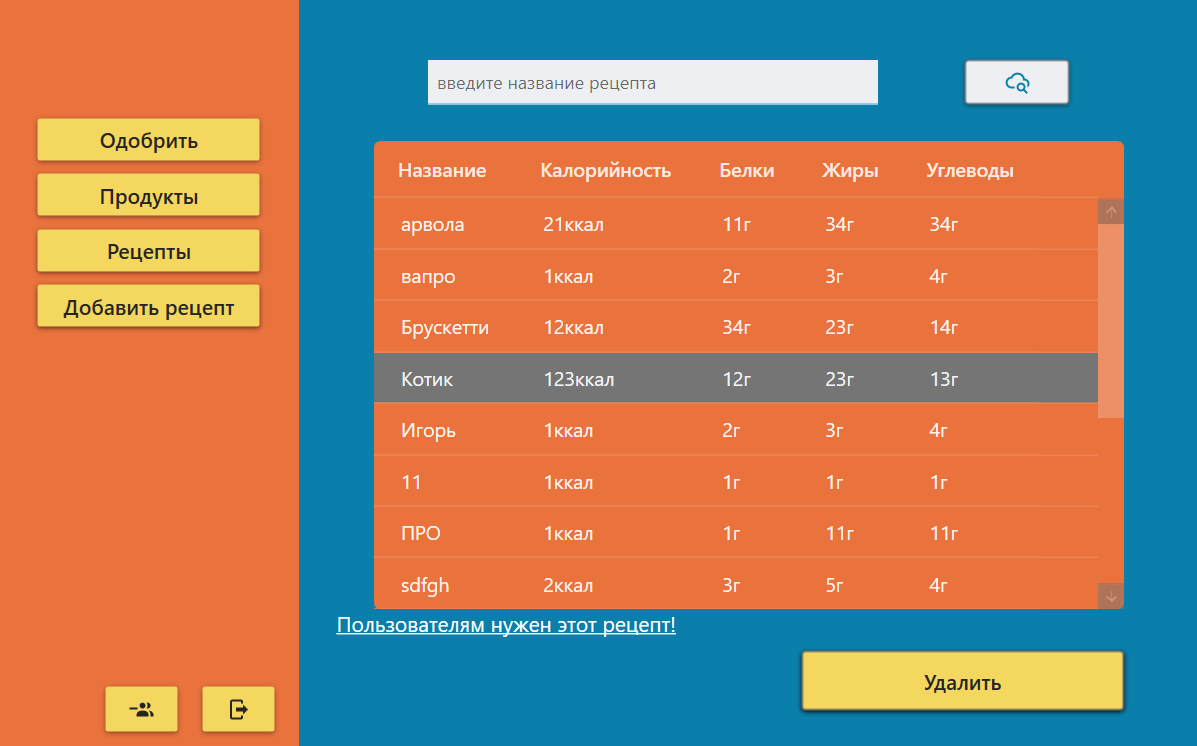


Рисунок 4.10 – Удаление рецепта, которым пользуются

Если при создании рецепта, какие-то поля не будут заполнены, то мы увидим предупреждение, а пока его не исправить, создания не произойдёт (результат на рис. 4.10):

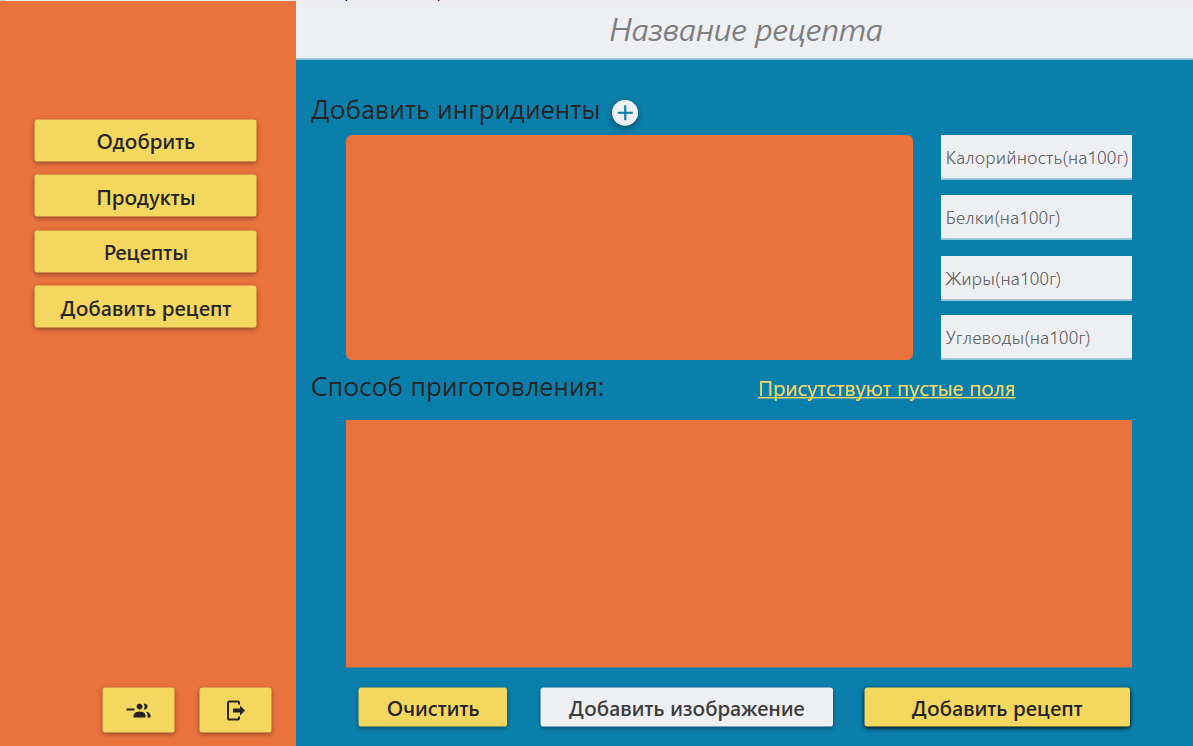


Рисунок 4.10 – Новый рецепт с пустыми полями

При попытке создать новый рецепт с очень длинным названием (более 50 символов), система его оповестит об ошибке (результат на рис 4.11):



Рисунок 4.11 – Новый рецепт с очень длинным названием

Таким образом в проекте были обработаны все возможные ошибки, что позволяет минимизировать шансы некорректной работы приложения и повысить комфорт его использования.