ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ»)

Строительно-политехнический колледж

**Отчет**

о прохождении производственной практики (по профилю специальности) профессионального модуля

ПМ.01. Разработка модулей программного обеспечения для компьютерных систем Название практики: Производственная практика (по профилю специальности) Разработка модулей программного обеспечения для компьютерных систем

Обучающегося \_Жуковой Анастасии Владимировны\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*Ф.И.О.)*

Специальность 09.02.07 Информационные системы и программирование.

4 курса ИСП-202 группы.

Сроки прохождения практики:

с «19» марта 2024г. по «15» апреля 2024 г.

в объеме час.

Место прохождения практики

*(наименование организации)*

Итоговая оценка \_\_\_\_\_\_\_\_\_\_\_\_ Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики

от организации

*(подпись, дата) (И.О.Фамилия, должность)*

Руководитель практики

от ВГТУ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Балабаева М.А

*(подпись, дата) (И.О.Фамилия, должность)*

Воронеж 2024

Оглавление

[Индивидуальное задание 3](#_Toc163746024)

[Введение 3](#_Toc163746025)

[Часть 1. Формирование алгоритмов разработки программных модулей в соответствии с техническим заданием. 5](#_Toc163746026)

[Часть 2. Разработка программных модулей в соответствии с техническим заданием. 7](#_Toc163746027)

[Демонстрация функций интерфейса 11](#_Toc163746028)

[Часть 3. Выполнение отладки программных модулей с использованием специализированных программных средств 16](#_Toc163746029)

[Часть 4. Осуществление рефакторинга и оптимизации программного кода. 20](#_Toc163746030)

[Заключение 23](#_Toc163746031)

[Список используемых источников 24](#_Toc163746032)

# **Индивидуальное задание**

В ходе выполнения производственной практики будет разработан интерфейс для предприятия «FT Solution».

# **Введение**

В современном мире работа любой организации связана с хранением и обработкой огромного количества информации.

Для того, чтобы хранить информацию о товарах данного предприятия нужно сначала создать базу данных, которая будет предоставлять всю необходимую информацию.

Но чтобы сотрудники могли комфортно работать с базой данных нужно создать интерфейс, в котором будет другие функции для упрощения работы.

Компания FT Solution ("ФТ Солюшен") разрабатывает и адаптирует программное обеспечение для [учётных систем](https://www.tadviser.ru/index.php/%D0%A3%D1%87%D0%B5%D1%82%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B) предприятий, учреждений [образования](https://www.tadviser.ru/index.php/%D0%9E%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B8_%D0%BD%D0%B0%D1%83%D0%BA%D0%B0) и [здравоохранения](https://www.tadviser.ru/index.php/%D0%97%D0%B4%D1%80%D0%B0%D0%B2%D0%BE%D0%BE%D1%85%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F) (социальной сферы). Является частью группы компаний FT Solution.

Информационная система «FT Solution» должна обеспечивать организацию по продаже тахографов, а именно:

* Хранение и обработку информации о тахографов
* Просмотр продукции
* Удаление товара из базы данных
* Добавление нового товара
* Редактирование информации товара

Автоматизация деятельности «FT Solution» позволяет:

* повысить точность, сохранность и защищённость данных  
  осуществить контроль вводимых данных
* упростить процедуру ввода данных
* Сократить время, затрачиваемое сотрудниками организации на выполнение таких процедур как: редактирование информации, удаление и добавление

Цель данной работы – создать интерфейс и подключить заранее созданную базу данных, добавить функционал в интерфейс.

# **Часть 1. Формирование алгоритмов разработки программных модулей в соответствии с техническим заданием.**

Первый шаг разработки программного модуля в значительной степени представляет собой смежный контроль структуры программы снизу: изучая спецификацию модуля, разработчик должен убедиться, что она ему понятна и достаточна для разработки этого модуля. В завершении этого шага выбирается язык программирования: хотя язык программирования может быть уже предопределен для всего ПС, все же в ряде случаев (если система программирования это допускает) может быть выбран другой язык, более подходящий для реализации данного модуля.

На втором шаге разработки программного модуля необходимо выяснить, не известны ли уже какие-либо алгоритмы для решения поставленной и или близкой к ней задачи. И если найдется подходящий алгоритм, то целесообразно им воспользоваться. Выбор подходящих структур данных, которые будут использоваться при выполнении модулем своих функций, в значительной степени предопределяет логику и качественные показатели разрабатываемого модуля, поэтому его следует рассматривать как весьма ответственное решение.

При программировании модуля следует иметь в виду, что программа должна быть понятной не только компьютеру, но и человеку: и разработчик модуля, и лица, проверяющие модуль, и тестовики, готовящие тесты для отладки модуля, и сопроводители ПС, осуществляющие требуемые изменения модуля, вынуждены будут многократно разбирать логику работы модуля. В современных языках программирования достаточно средств, чтобы запутать эту логику сколь угодно сильно, тем самым, сделать модуль трудно понимаемым для человека и, как следствие этого, сделать его ненадежным 3 или трудно сопровождаемым. Поэтому необходимо принимать меры для выбора подходящих языковых средств и следовать определенной дисциплине программирования. В связи с этим предложили строить программу как композицию из нескольких типов управляющих конструкций (структур), которые позволяют сильно повысить понимание логики работы программы. Программирование с использованием только таких конструкций назвали структурным.

На Рисунке 1 представлен алгоритм программы.

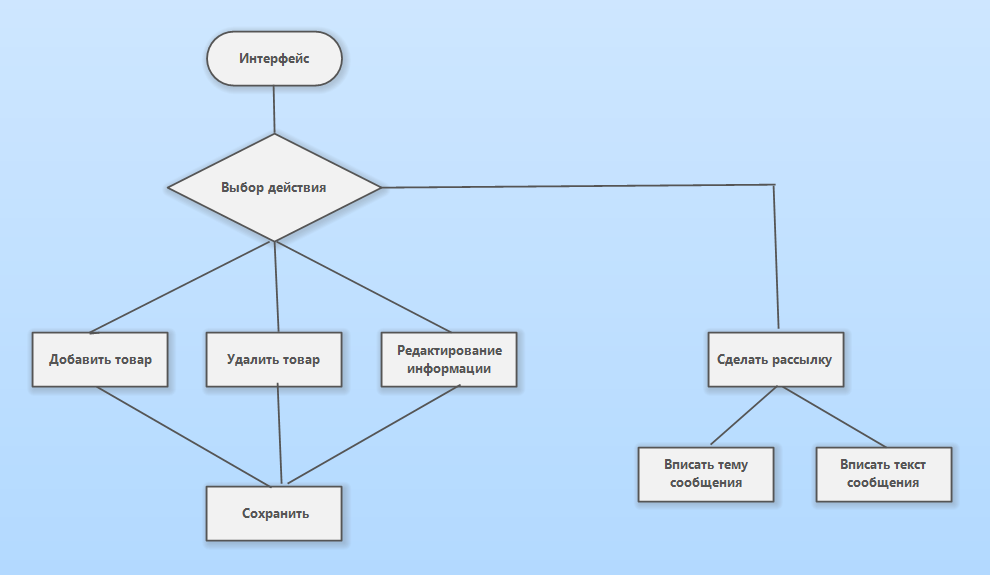


Рисунок 1 - Блок схема алгоритма программы

Исходя из Рисунка 1 видно, какие функции должны быть разработаны и как эти функции будут работать.

# **Часть 2. Разработка программных модулей в соответствии с техническим заданием.**

Создание любого программного средства выполняется по некоторой схеме. Данная схема представляет собой последовательность стандартных этапов производственного процесса. Этот процесс нужно спланировать, оценить его ресурсы. В ходе реализации этого процесса нужно: спроектировать ПС в виде системы, состоящей из компонент; описать функции этих компонент и их связи между собой; запрограммировать эти компоненты, автономно их отладить, собрать вместе и провести комплексную отладку; подготовить документацию на ПС; обучить пользователей; провести опытную эксплуатацию ПС; организовать сопровождение системы.

Под жизненным циклом программного средства понимают весь период его разработки и эксплуатации, начиная от момента возникновения замысла ПС и кончая прекращением его использования. В настоящее время можно выделить пять основных подходов к организации процесса создания и использования ПС.

Этапы разработки ПО.

Разработка любой программы состоит из нескольких этапов, грамотная реализация которых является обязательным условием для получения хорошего результата.

Анализ требований.

В рамках этой стадии происходит максимально эффективное взаимодействие нуждающегося в программном решении клиента и сотрудников компании- разработчика, в ходе обсуждения деталей проекта помогающих более четко сформулировать предъявляемые к ПО требования. Результатом проведенного анализа становится формирование основного регламента, на который будет опираться исполнитель в своей работе — технического задания на разработку программного обеспечения. ТЗ должно полностью описывать поставленные перед разработчиком задачи и охарактеризовать конечную цель проекта в понимании заказчика.

Как было определенно в блок схеме из функционала нужно реализовать удаление, редактирование, добавление, сохранение и рассылку почты.

Для начала необходимо сделать отдельный метод для удаления данных в таблице:

После создания метода, нужно сделать сохранение действия в базе данных:

private void button2\_Click(object sender, EventArgs e)

{

int index = dataGridView1.CurrentCell.RowIndex;

dataGridView1.Rows[index].Visible = false;

if (dataGridView1.Rows[index].Cells[0].Value.ToString() == string.Empty)

{

dataGridView1.Rows[index].Cells[5].Value = RowState.Deleted;

return;

}

dataGridView1.Rows[index].Cells[4].Value = RowState.Deleted;

dataBase.OpenConnection();

var id = textBoxId.Text;

string query = $"DELETE FROM products\_tbl WHERE id\_products = " + id;

NpgsqlCommand command = new NpgsqlCommand(query,dataBase.GetConnection());

command.ExecuteNonQuery();

dataBase.CloseConnection();

}

И сохранение изменений информации в базе данных:

private void button3\_Click(object sender, EventArgs e)

{

var selectedRowIndex = dataGridView1.CurrentCell.RowIndex;

var id = textBoxId.Text;

var name = textBoxName.Text;

var description = textBoxDescription.Text;

var price = textBoxPrice.Text;

var image = textBoxImage.Text;

if (dataGridView1.Rows[selectedRowIndex].Cells[0].Value.ToString() != string.Empty)

{

dataGridView1.Rows[selectedRowIndex].SetValues(id, name, description, price, image);

dataGridView1.Rows[selectedRowIndex].Cells[5].Value = RowState.Modfied;

}

dataBase.OpenConnection();

var id = textBoxId.Text;

var name = textBoxName.Text;

var description = textBoxDescription.Text;

var price = textBoxPrice.Text;

var image = textBoxImage.Text;

string query = $"UPDATE products\_tbl SET name\_products = '{name}',description = '{description}',price\_products = '{price}' WHERE id\_products = " + id;

NpgsqlCommand command = new NpgsqlCommand(query, dataBase.GetConnection());

command.ExecuteNonQuery();

dataBase.CloseConnection();

Теперь необходимо реализовать функцию добавления:

Для этого была создана отдельная форма, где необходимо вписывать такие данные как: название товара, описание, цена, путь изображения

dataBase.OpenConnection();

var name = textBoxName.Text;

var description = textBoxDescription.Text;

var price = textBoxPrice.Text;

var image = textBoxImage.Text;

string query = $"insert into products\_tbl (name\_products,description,price\_products,image\_products) values ('{name}','{description}','{price}','{image}')";

NpgsqlCommand command = new NpgsqlCommand(query,dataBase.GetConnection());

command.ExecuteNonQuery();

MessageBox.Show("Товар успешно добавлен!", "Успех!", MessageBoxButtons.OK, MessageBoxIcon.Information);

dataBase.CloseConnection();

Form1 form1 = new Form1();

this.Close();

form1.Show();

Теперь осталось реализовать функцию рассылки почты:

string smtpServer = "smtp.mail.ru";

int smtpPort = 587;

string smtpUsername = "waguteru@mail.ru";

string smtpPassword = "kG6K9KvM5PtpENRLi1Vp";

using (SmtpClient smtpClient = new SmtpClient(smtpServer, smtpPort))

{

smtpClient.Credentials = new NetworkCredential(smtpUsername, smtpPassword);

smtpClient.EnableSsl = true;

using (MailMessage mailMessage = new MailMessage())

{

mailMessage.From = new MailAddress(smtpUsername);

mailMessage.To.Add("zhukanas04@mail.ru");

mailMessage.Subject = txt3.Text;

mailMessage.Body = txt2.Text;

try

{

smtpClient.Send(mailMessage);

MessageBox.Show("Сообщение отправлено");

Form1 form1 = new Form1();

form1.Show();

this.Close();

}

catch (Exception ex)

{

MessageBox.Show($"Сообщение не отправлено {ex.Message}");

}

}

}

## **Демонстрация функций интерфейса**

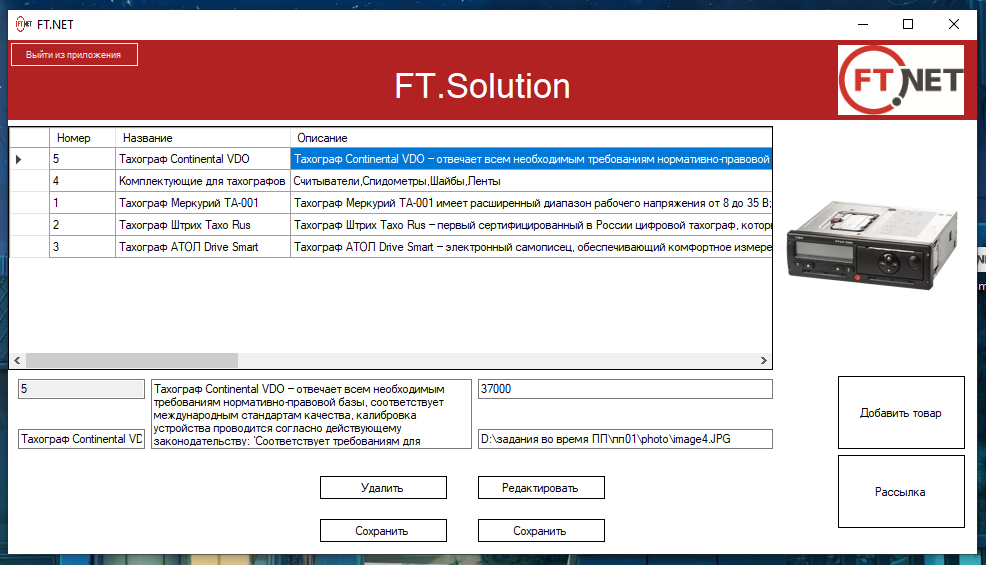


Рисунок 2 - Общий вид интерфейса

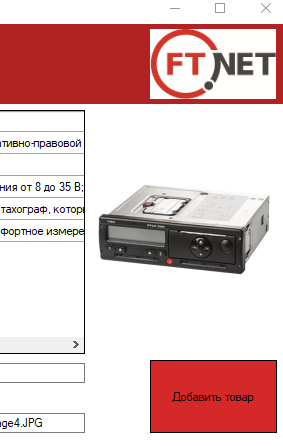


Рисунок 3 - демонстрация изменения цвета кнопок

На рисунке 3 показано, что при наведении курсора на кнопку, она изменяет цвет, но если убрать курсор, то цвет возвращает основной цвет.



Рисунок 4 - Демонстрация окна "Добавление"

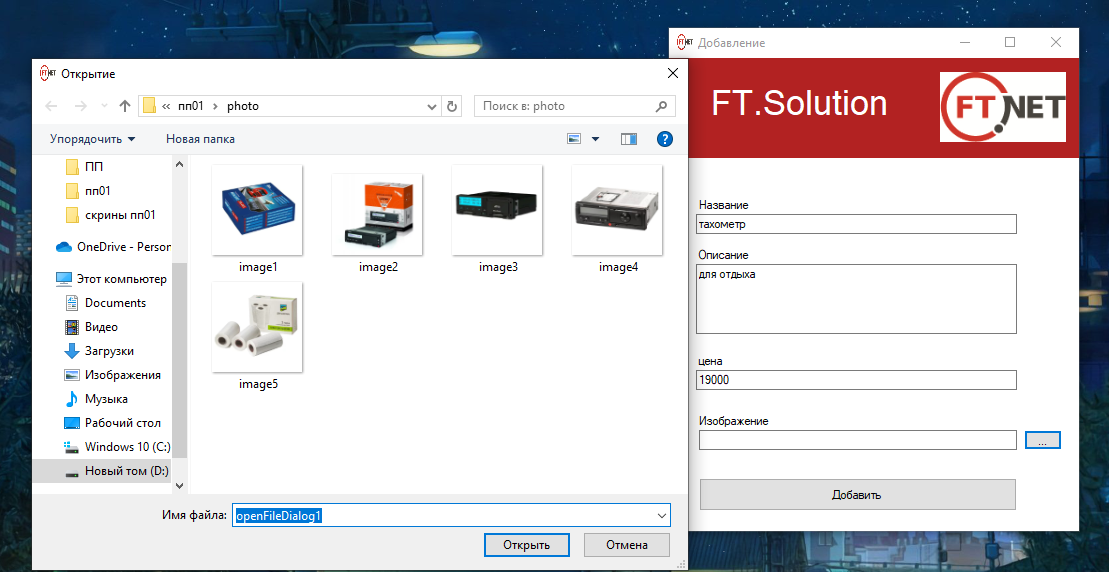


Рисунок 5 - Демонстрация выбора изображения

Чтобы сотруднику было проще указать путь изображения, была реализована функция по автоматической вставке пути.

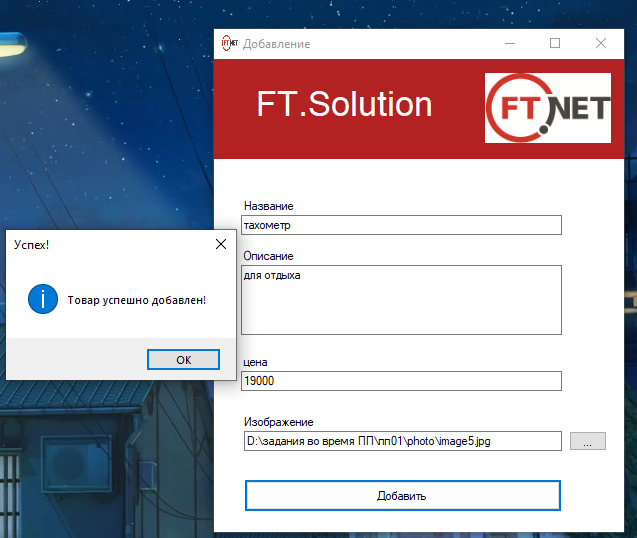


Рисунок 6 - Результат успешного добавления



Рисунок 7 - Показ окна для рассылки

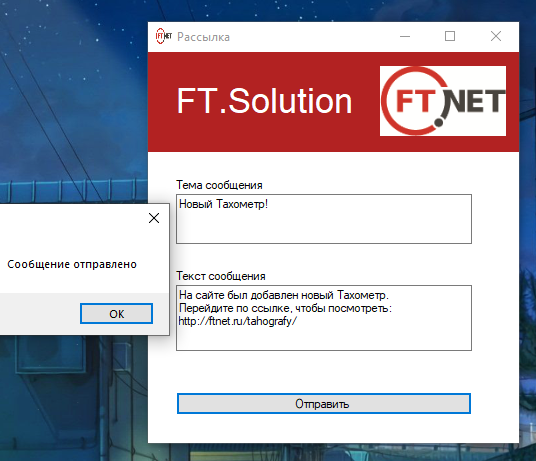


Рисунок 8 - Результат того, что письмо было отправлено

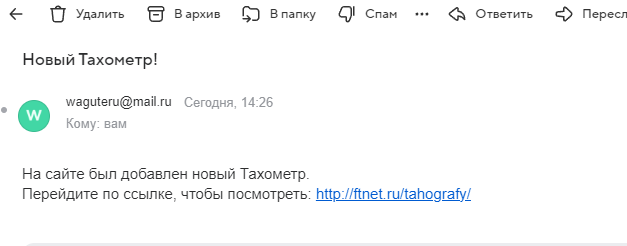


Рисунок 9 - Результат того, что письмо пришло на почту



Рисунок 10 - Кнопка "Выйти из приложения"

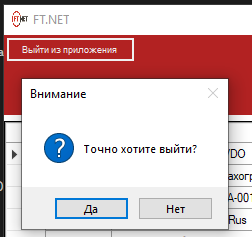


Рисунок 11 - Подтверждение выхода

# **Часть 3. Выполнение отладки программных модулей с использованием специализированных программных средств**

Отладка – это процесс поиска и исправления ошибок или неполадок в исходном коде какого-либо программного обеспечения. Когда программное обеспечение не работает, как ожидалось, компьютерные программисты изучают код, чтобы выяснить причину появления ошибок. Они используют инструменты отладки для запуска программного обеспечения в контролируемой среде, пошаговой проверки кода, а также анализа и поиска проблем.

Недочеты и ошибки случаются в компьютерном программировании, поскольку это абстрактная и концептуальная работа. Компьютеры обрабатывают данные в виде электронных сигналов. Языки программирования абстрагируют эту информацию, чтобы люди могли более эффективно взаимодействовать с компьютерами. Любой тип программного обеспечения имеет несколько уровней абстракции, на которых различные компоненты взаимодействуют для правильной работы приложения. Когда возникают ошибки, найти и решить проблему может быть непросто. Инструменты и стратегии отладки помогают быстрее устранять проблемы и повышать производительность разработчиков. В результате улучшается как качество программного обеспечения, так и опыт конечного пользователя.

Дефекты программного обеспечения возникают из-за сложности, присущей разработке программного обеспечения. Кроме того, незначительные производственные ошибки наблюдаются после запуска программного обеспечения, потому что клиенты используют его неожиданным образом. Ниже приведены некоторые распространенные типы ошибок, которые зачастую требуют отладки.

**Синтаксические ошибки**

Синтаксическая ошибка возникает, когда в компьютерной программе неправильно указано значение. Это эквивалент опечатки или орфографической ошибки в текстовом редакторе. Если есть синтаксические ошибки, программа не будет компилироваться или запускаться. Как правило, программное обеспечение для редактирования кода выделяет эту ошибку.

**Семантические ошибки**

Семантические ошибки возникают из-за неправильного использования операторов программирования.

**Логические ошибки**

Логические ошибки возникают, когда программисты искажают поэтапный процесс или алгоритм компьютерной программы.

**Ошибки времени выполнения**

Ошибки времени выполнения возникают из-за среды вычисления, в которой выполняется программный код.

Для того, чтобы проверить созданный интерфейс будут разработаны тесты, которые будут проверять базовый функционал приложения.

Для начала необходимо создать отдельный класс, в котором будет вынесена логика функционала, чтобы её можно было проверить.

public class TestData

{

DataBase dataBase = new DataBase();

public TestData() { }

public void AddProducts(string name\_products, string description, string image\_products)

{

dataBase.OpenConnection();

var querystring = $"insert into products\_tbl (name\_products,description,image\_products) values ('{name\_products}','{description}','{image\_products}')";

var comm = new NpgsqlCommand(querystring, dataBase.GetConnection());

comm.ExecuteNonQuery();

MessageBox.Show("данные успешно добавлены!", "Успех!", MessageBoxButtons.OK, MessageBoxIcon.Information);

dataBase.CloseConnection();

}

public void UPDATEitem(string name\_products)

{

dataBase.OpenConnection();

var id = Convert.ToInt32(1);

string query = $"UPDATE products\_tbl SET name\_products = '{name\_products}' WHERE id\_products = " + id;

NpgsqlCommand command = new NpgsqlCommand(@query, dataBase.GetConnection());

command.ExecuteNonQuery();

dataBase.CloseConnection();

}

public void Deleteitem(string name\_products)

{

dataBase.OpenConnection();

string query = $"DELETE FROM products\_tbl WHERE id\_products = '{name\_products}'";

NpgsqlCommand command = new NpgsqlCommand(@query, dataBase.GetConnection());

command.ExecuteNonQuery();

dataBase.CloseConnection();

}

}

После этого нужно написать сами тесты, используя методы из класса.

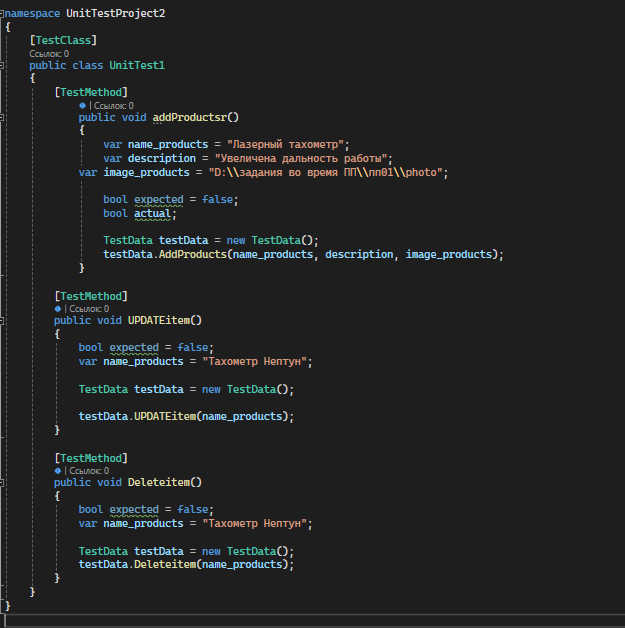


Рисунок 12 - Код тестов

В данных тестах проверяется добавление товара, удаление и редактирование

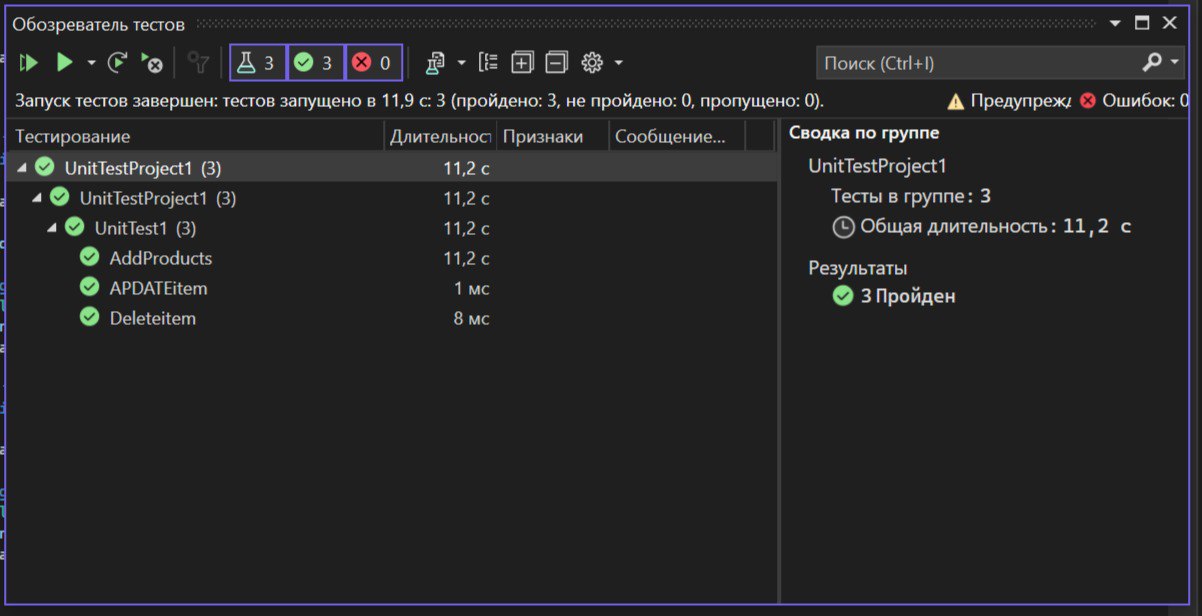


Рисунок 13 - Результат того, что тесты проходят успешно

# **Часть 4. Осуществление рефакторинга и оптимизации программного кода.**

Рефакторинг — это переработка исходного кода программы, чтобы он стал более простым и понятным.

Рефакторинг не меняет поведение программы, не исправляет ошибки и не добавляет новую функциональность. Он делает код более понятным и удобочитаемым.

Стройный, хорошо структурированный код легко читается и быстро дорабатывается. Но редко удаётся сразу сделать его таким. Разработчики спешат, в процессе могут меняться требования к задаче, тестировщики находят баги, которые нужно быстро исправить, или возникают срочные доработки, и их приходится делать второпях.

В результате даже изначально хорошо структурированный исходник становится беспорядочным и непонятным. Программисты знают, как легко завязнуть в этом хаосе. Причём неважно, чужой это код или собственный.

Чтобы решить все эти проблемы, делается рефакторинг программы. В новом проекте он нужен, чтобы:

* сохранить архитектуру проекта, не допустить потери структурированности;
* упростить будущую жизнь разработчиков, сделать код понятным и прозрачным для всех членов команды;
* ускорить разработку и поиск ошибок.

Но любое приложение со временем устаревает: язык программирования совершенствуется, появляются новые функции, библиотеки, операторы, делающие код проще и понятнее. То, что год назад требовало пятидесяти строк, сегодня может решаться всего одной.

Поэтому даже идеальная когда-то программа со временем требует нового рефакторинга, обновляющего устаревшие участки кода.

Рефакторинг — не оптимизация, хотя и, может быть, с нею связан. Часто его проводят одновременно с оптимизацией, поэтому понятия кажутся синонимами. Но у этих процессов разные цели.

Цель оптимизации — улучшение производительности программы, а рефакторинга — улучшение понятности кода. После оптимизации исходный код может стать сложнее для понимания.

Чтобы упросить и улучшить читаемость кода были изменены данные блоки кода:



Рисунок 14 – Удаление

Чтобы не нагружать событие по нажатию кнопки на удаление большим кодом, код был выделен в отдельный метод

Такое же было проведено и с редактированием.

Также чтобы не нагружать код, используемый для подключения к базе данных, был перенесён в отдельный класс.

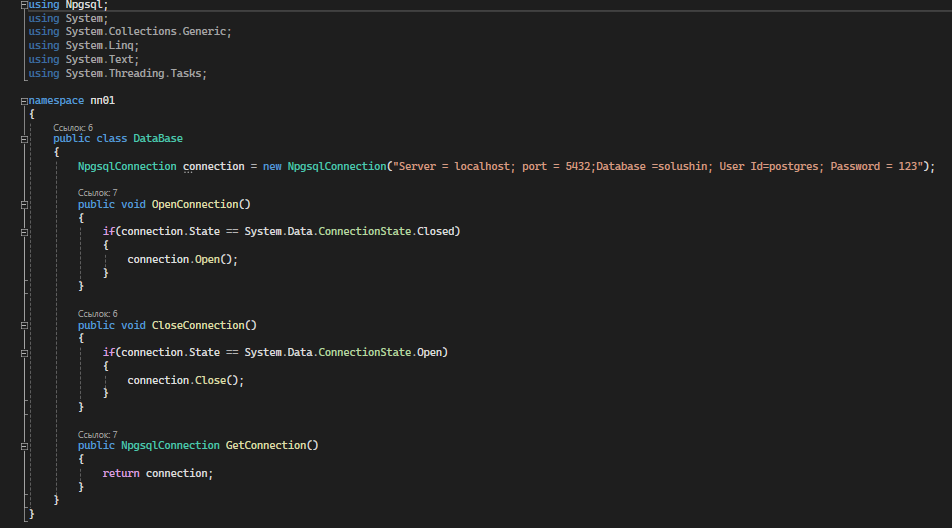


Рисунок 15 - Класс для подключения

# **Заключение**

В ходе производственной практики, были приобретены следующие умения: выбор и применение способов решения профессиональных задач, оценка эффективности и качества выполнения задач, нахождение, использование, анализ и интерпретация информации, используя различные источники, включая электронные, для эффективного выполнения профессиональных задач, профессионального и личностного развития; демонстрация навыков отслеживания изменений в нормативной и законодательной базах, оценка эффективности и качества выполнения задач, взаимодействие с сотрудниками организации в ходе практики, а также с руководством и сотрудниками экономического субъекта во время прохождения практики, а также было получено экспертное наблюдение и оценка результатов формирования поведенческих навыков в ходе обучения.

В данной производственной практики был реализован интерфейс для базы данных.

Интерфейс был разработан для упрощения работы сотрудников, увеличению их эффективности работы.

Сначала был сформулирован алгоритм, используя техническое задание.

Затем разработаны программные модули. Для проверки правильной работоспособности приложения были разработаны тесты. И заключением был оптимизирован код.

# **Список используемых источников**

1. Зубкова Т.М. Технология разработки программного обеспечения [Электронный ресурс]: учебное пособие/ Зубкова Т.М.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017.— 469 c.— Режим доступа: http://www.iprbookshop.ru/78846.html.— ЭБС «IPRbooks»
2. Иноземцева С.А. Информатика и программирование [Электронный ресурс]: лабораторный практикум/ Иноземцева С.А.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2018.— 68 c.— Режим доступа: http://www.iprbookshop.ru/75691.html.— ЭБС «IPRbooks»
3. Лобачев С.Л. Основы разработки электронных образовательных ресурсов [Электронный ресурс]/ Лобачев С.Л.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Эр Медиа, 2019.— 188 c.— Режим доступа: http://www.iprbookshop.ru/79711.html.— ЭБС «IPRbooks»
4. Мальшина Н.А. Моделирование и оптимизация процессов и систем сервиса [Электронный ресурс]: учебное пособие/ Мальшина Н.А.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2019.— 127 c.— Режим доступа: http://www.iprbookshop.ru/79773.html.— ЭБС «IPRbooks»
5. Учебники по программированию http://programm.ws/index.php
6. Рассел, Джесси Интерфейс пользователя / Джесси Рассел. - М.: VSD, **2017. - 587** c.
7. Рассел, Джесси Интерфейс программирования приложений / Джесси Рассел. - М.: VSD, **2015. - 707** c.