Министерство образования Московской области

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ МОСКОВСКОЙ ОБЛАСТИ

ЛЮБЕРЕЦКИЙ ТЕХНИКУМ ИМЕНИ ГЕРОЯ СОВЕТСКОГО СОЮЗА,

ЛЁТЧИКА-КОСМОНАВТА Ю. А. ГАГАРИНА

ОТЧЕТ

по производственной практике

ПМ 01 «Разработка модулей программного обеспечения

для компьютерных систем»

Костина Макара Руслановича

Специальность: 09.02.07 «Информационные системы и программирование»

Период практики с «08» декабря 2021 г. по «14» декабря 2021 г.

Группа № 185, курс 4

Руководители работы:

Жирнова Юлия Витальевна

Лобзин Роман Валентинович

Дзержинский 2021

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc92713204)

[Описание предметной области 4](#_Toc92713205)

[Изучение программных средств 5](#_Toc92713206)

[Создание базы данных 8](#_Toc92713207)

[Разработка desktop-приложения 12](#_Toc92713208)

[Выводы 26](#_Toc92713209)

[Список литературы 27](#_Toc92713210)

Введение

«Разработка модулей программного обеспечения для компьютерных систем» является важным профессиональным модулем для специальности 09.02.07 «Информационные системы и программирование», обеспечивающим получение знаний, умений и опыта, необходимых для профессиональной деятельности выпускника.

Производственная практика, проходящая в реальных условиях различных предприятий, является важной частью учебного процесса, необходимой для подготовки квалифицированных специалистов.

Во время прохождения производственной практики происходит закрепление и углубление полученных теоретических знаний, приобретение студентами профессиональных умений и навыков практической работы, что способствует повышению общего уровня профессиональной подготовки.

Целью производственной практики является получение навыков практической работы в условиях конкретного предприятия, а именно: требуется создать базу данных и разработать desktop-приложение, опираясь на деятельность организации.

Для достижения поставленной цели необходимо решить следующие задачи:

* Описание предметной области.
* Изучение программных средств.
* Создание каркаса приложения.
* Функциональное тестирование.
* Осуществление рефакторинга и оптимизации программного кода.

В ходе производственной практики планируется освоение следующего вида деятельности: «Разработка модулей программного обеспечения для компьютерных систем» (ПК):

ПК 1.1. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.

ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием.

ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК 1.4. Выполнять тестирование программных модулей.

ПК 1.5. Осуществлять рефакторинг и оптимизацию программного кода.

ПК 1.6. Разрабатывать модули программного обеспечения для мобильных платформ.

Описание предметной области

Для разработки desktop-приложения необходимо располагать описанием выбранной предметной области, которое должно охватывать реальные объекты и процессы, определять все необходимые источники информации (документы предметной области) для обеспечения предполагаемых запросов пользователя и решаемых в приложении задач.

Определение состава и структуры данных, которые должны быть загружены в базу данных, осуществляется на основе анализа предметной области.

Прежде всего следует изучить структуру и деятельность предприятия и получить представление о его функционировании. Анализируются запросы пользователей, выбираются информационные объекты и их характеристики, которые предопределяют содержание будущего приложения.

Описание предметной области организации:

Акционерное Общество «МЕТТЭМ-БЕТАР» является официальным представителем ООО ПКФ «БЕТАР» — одного из крупнейших производителей приборов учета воды и газа, электрической энергии, тепла, компонентов систем водоснабжения.

Осуществляет оптовую и розничную продажу юридическим и физическим лицам следующих категорий товаров: счетчики воды бытовые, счетчики воды промышленные, счетчики газа, регуляторы давления, преобразователи воды магнитные, счетчики электричества, теплосчетчики, индикаторы магнитного поля, фитинги для труб. Кроме того, организация занимается оптовыми продажами широкого ассортимента фасованной таблетированной соли.

Процесс деятельности организация начинается с поступления заказа от клиента.

На основе описания предметной области выявлены следующие документы-источники: прайс-листы и перечни товаров с описанием их характеристик. Информационный анализ этих документов определит функциональные зависимости и выделит информационные объекты для последующего проектирования и разработки.

Постановка задачи:

Задачей производственной практики являлась разработка desktop-приложения согласно его назначению и в соответствии с запросами организации. Данное приложение должно обеспечить занесение и редактирование информации при учёте продукции на складе, а также быстрый и наглядный доступ к данным о продукции при формировании заказов.

Изучение программных средств

Выбор программных средств разработки для реализации проекта осуществляется на основании требований, выдвигаемых от организации.

Интегрированная среда разработки (IDE) — многофункциональная программа, которая поддерживает многие аспекты разработки программного обеспечения.

Для выполнения практического задания использовались следующие программные средства:

* Microsoft Visual Studio

Интегрированная среда разработки Visual Studio — стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Помимо стандартного редактора и отладчика, которые есть в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки.

* SQL Server Management Studio (SSMS)

SQL Server Management Studio (SSMS) — интегрированная среда для управления любой инфраструктурой SQL, от SQL Server до баз данных SQL Azure. SSMS предоставляет средства для настройки, наблюдения и администрирования экземпляров SQL Server и баз данных. С помощью SSMS можно развертывать, отслеживать и обновлять компоненты уровня данных, используемые приложениями, а также создавать запросы и скрипты.

Установка Microsoft Visual Studio Community:

1. Скачать файл начального загрузчика Visual Studio с официального сайта — https://visualstudio.microsoft.com/ru/downloads/ (Рис. 1).

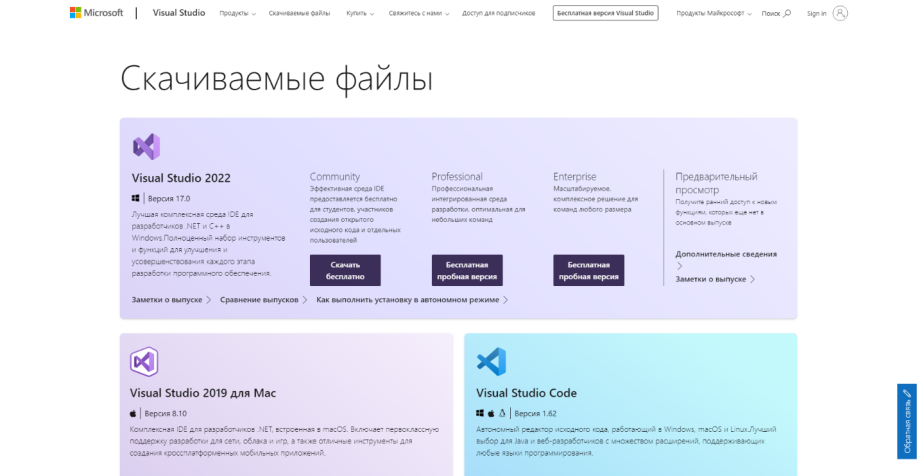


Рисунок 1. Официальная страница загрузки Microsoft Visual Studio

2. Запустить файл начального загрузчика.

Если появляется оповещение системы контроля учетных записей, нажать кнопку «Да».

3. Принять условия лицензии и заявление о конфиденциальности корпорации Майкрософт. Нажать «Продолжить» (Рис. 2).

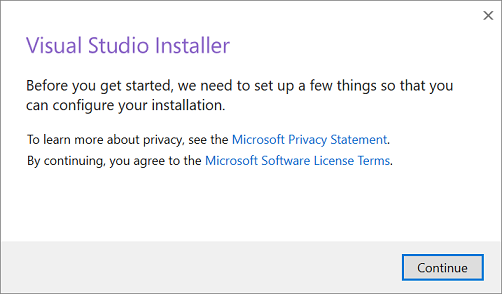


Рисунок 2. Условия конфиденциальности и лицензии

4. Выбор рабочих нагрузок после завершения установки.

Для выполнения задания понадобятся такие рабочие нагрузки как: «ASP.NET и разработка мобильных приложений» и «Разработка классических приложений .NET» (Рис. 3).

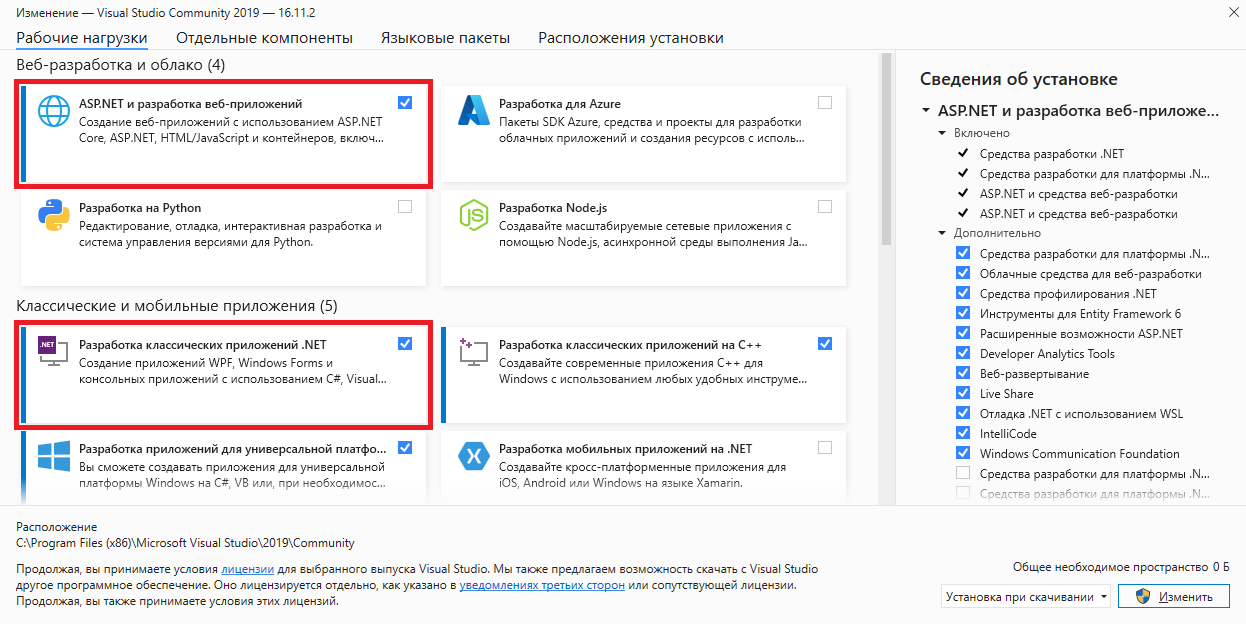


Рисунок 3. Выбор рабочих нагрузок

Установка SQL Server Management Studio (SSMS):

1. Скачать загрузчик SSMS с официального сайта https://docs.microsoft.com/ru-ru/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15 (Рис. 4).

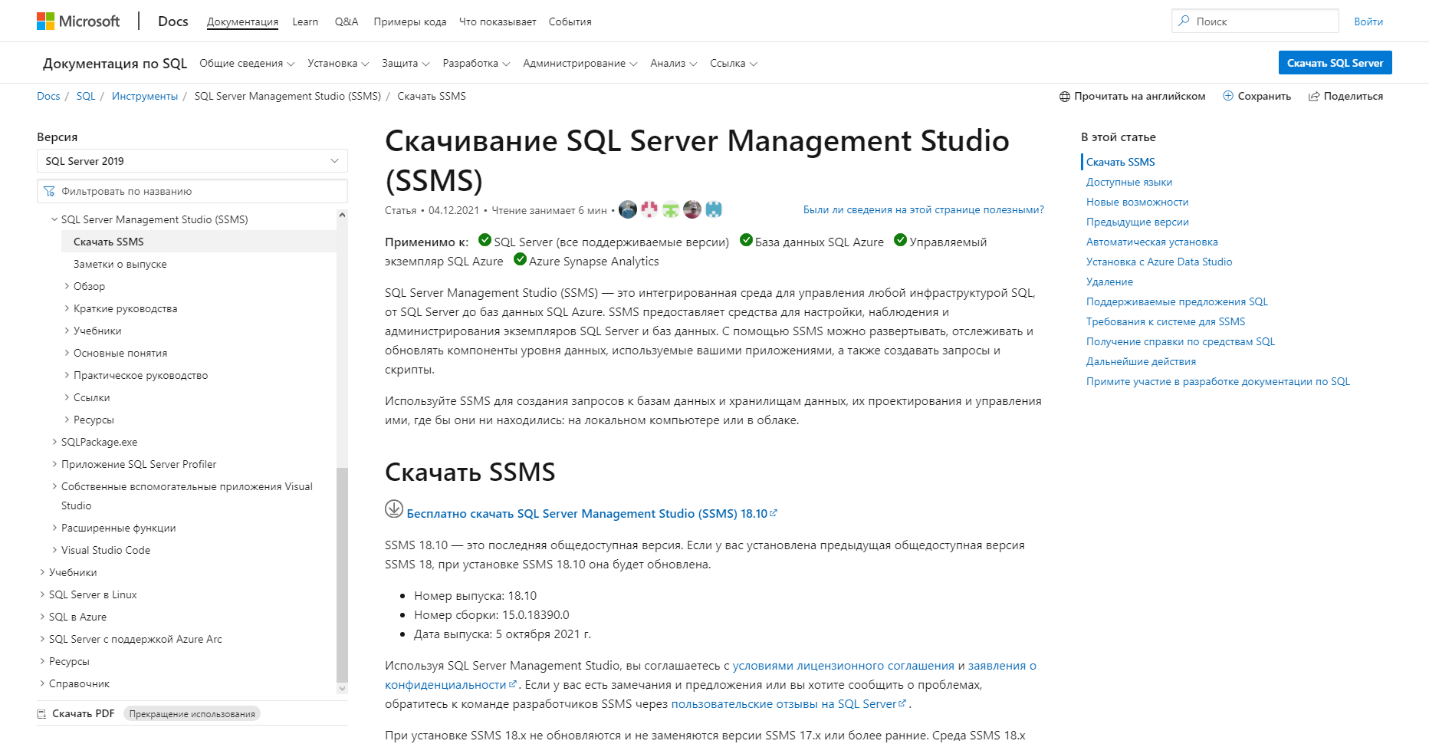


Рисунок 4. Официальная страница загрузки SSMS

2. Запустить загрузчик и выполнить установку программы.

На рисунке 5 изображено начальное окно установщика SSMS.

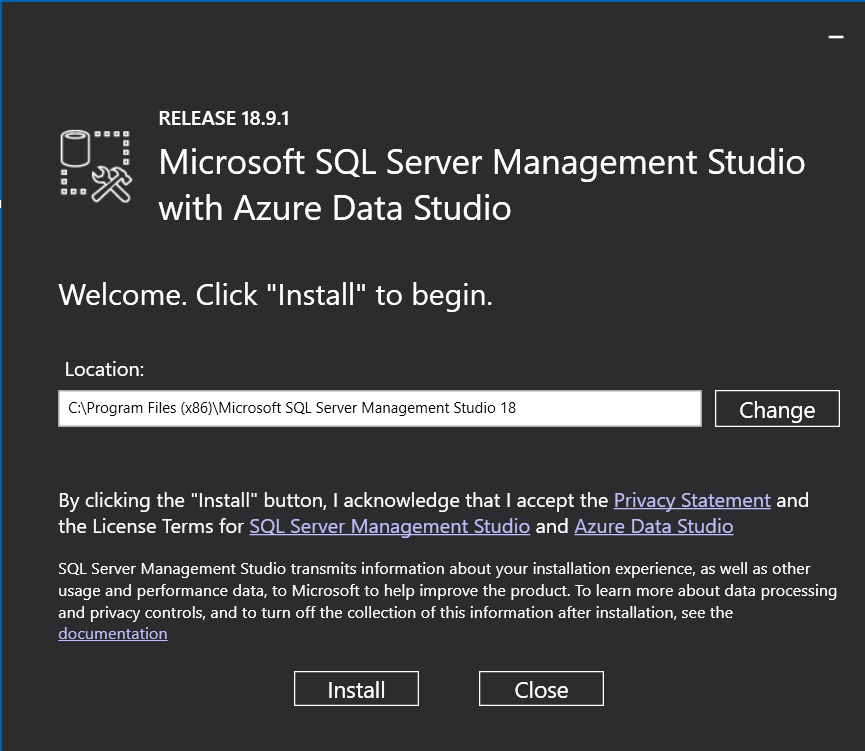


Рисунок 5. Начальное окно установки SSMS

Создание базы данных

База данных (БД) — организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации, преимущественно больших объемов.

Скрипт — отдельные последовательности действий, созданные для автоматического выполнения задачи.

Система управления базами данных (СУБД) — комплекс программ, позволяющих создать базу данных и манипулировать данными (вставлять, обновлять, удалять и выбирать).

Первичный ключ (PK, Primary Key) — минимальный набор атрибутов, совокупность значений которых однозначно определяет кортеж в отношении.

Внешний ключ (FK, Foreign Key) — столбец или комбинация столбцов, значения которых соответствуют Первичному ключу в другой таблице.

Для создания БД запустим MS SQL Management Studio, подключимся к локальному серверу и создадим новую базу данных.

Далее создадим таблицы с помощью конструктора диаграмм (Рис. 6).

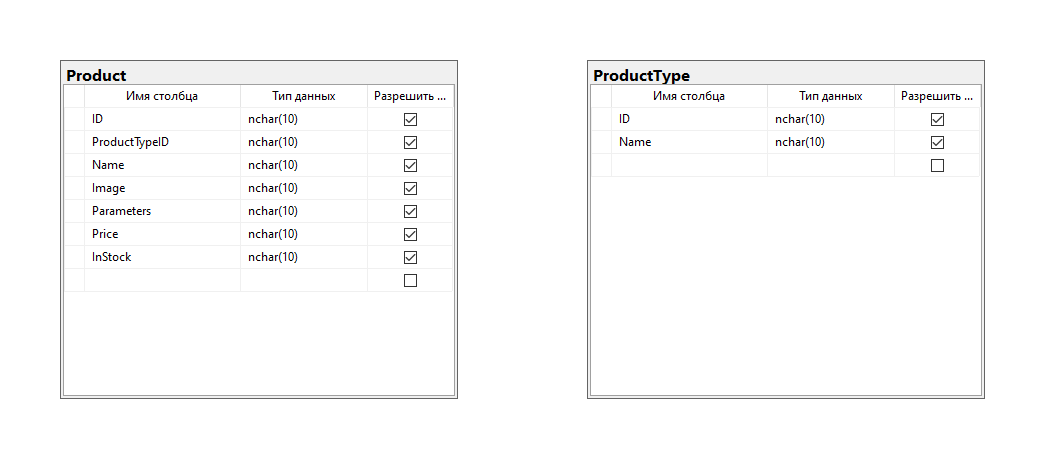


Рисунок 6. Таблицы в конструкторе диаграмм

Устанавливаем типы данных и обязательность или необязательность поля (Рис. 7).

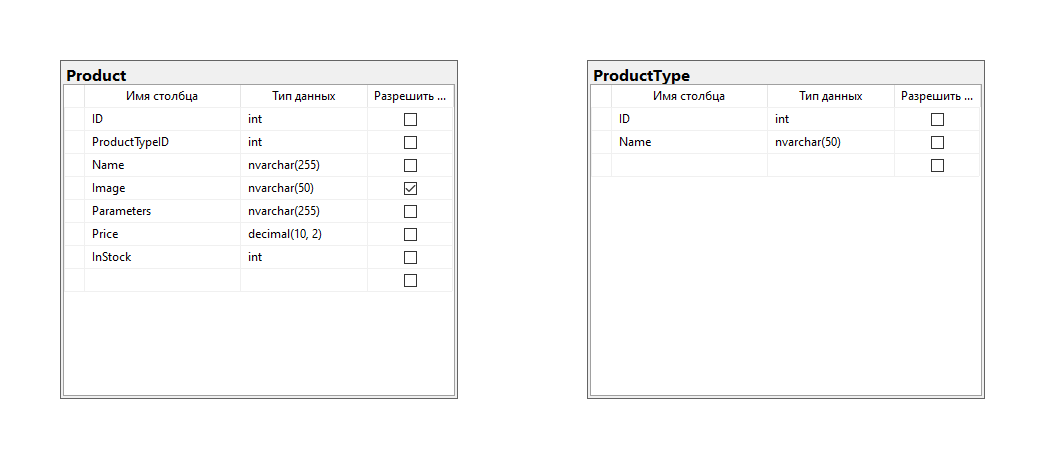


Рисунок 7. Установка типов данных и обязательности полей

В третьем столбце есть маркер, отвечающий за обязательность поля. В случае если мы отметим его галочкой, поле будет необязательным при заполнении в таблицу.

Расставим первичные ключи и связи между таблицами (Рис. 8).

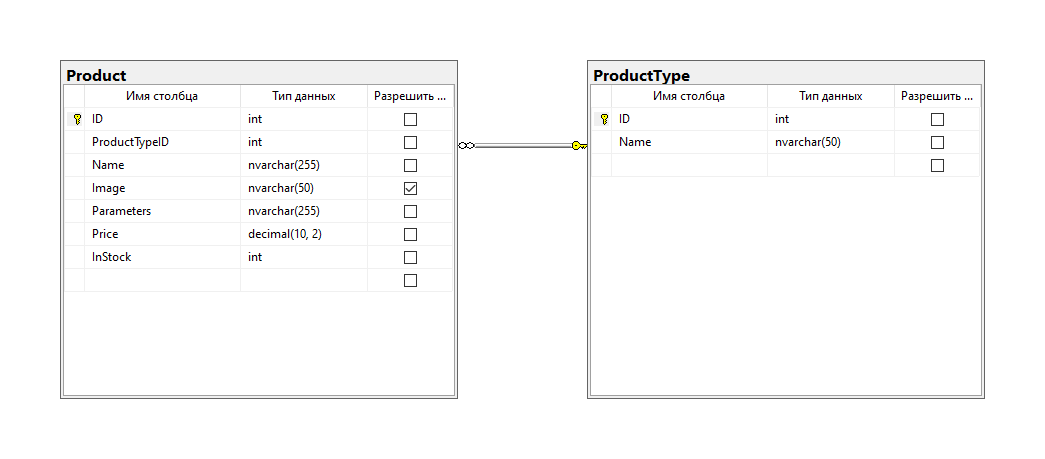


Рисунок 8. Расстановка первичных ключей и связей между таблицами

Первичный ключ — поле, которое уникально характеризует запись (строку) в таблице.

Связь между таблицами устанавливает отношения между значениями в ключевых полях — часто между полями, имеющими одинаковые имена в обеих таблицах. В большинстве случаев с первичным ключом одной таблицы, являющимся уникальным идентификатором каждой записи, связывается внешний ключ другой таблицы.

Заполним таблицы актуальными данными (Рис. 9–10).

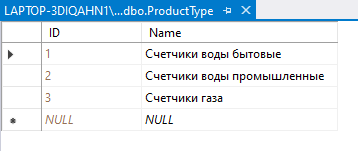


Рисунок 9. Таблица «ProductType» c данными

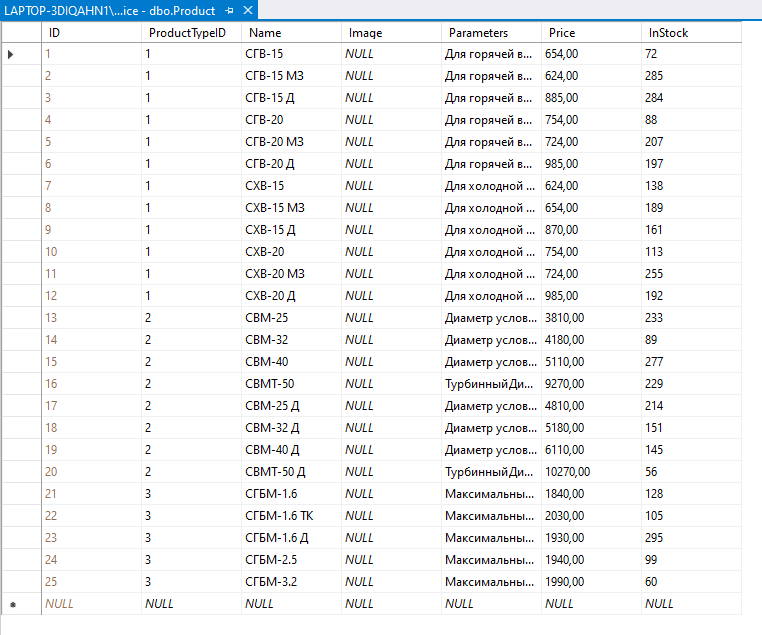


Рисунок 10. Таблица «Product» c данными

Для того, чтобы не вести учет количества порядковых номеров, в настройках поля можно установить автоматическое определение значения поля при добавлении записи. Например, если в таблице есть 3 записи с номерами от 1 до 3, то следующая запись автоматически будет иметь номер 4 (Рис. 11).

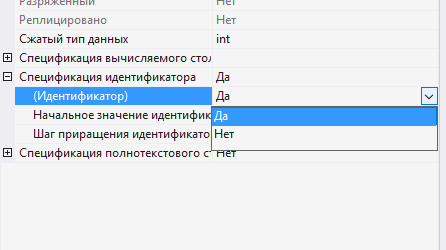


Рисунок 11. Установка автоматического определения значения поля

Разработка desktop-приложения

WPF (Windows Presentation Foundation) — аналог WinForms, система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

Настольное приложение (desktop application) — программное обеспечение, предназначенное для работы на персональных компьютерах.

Элемент управления (control) — элемент, который предназначен для взаимодействия с пользователем или для отображения ему какой-либо информации

Запустим Visual Studio и создадим проект WPF. В проекте создадим каркас главного окна (Рис. 12).

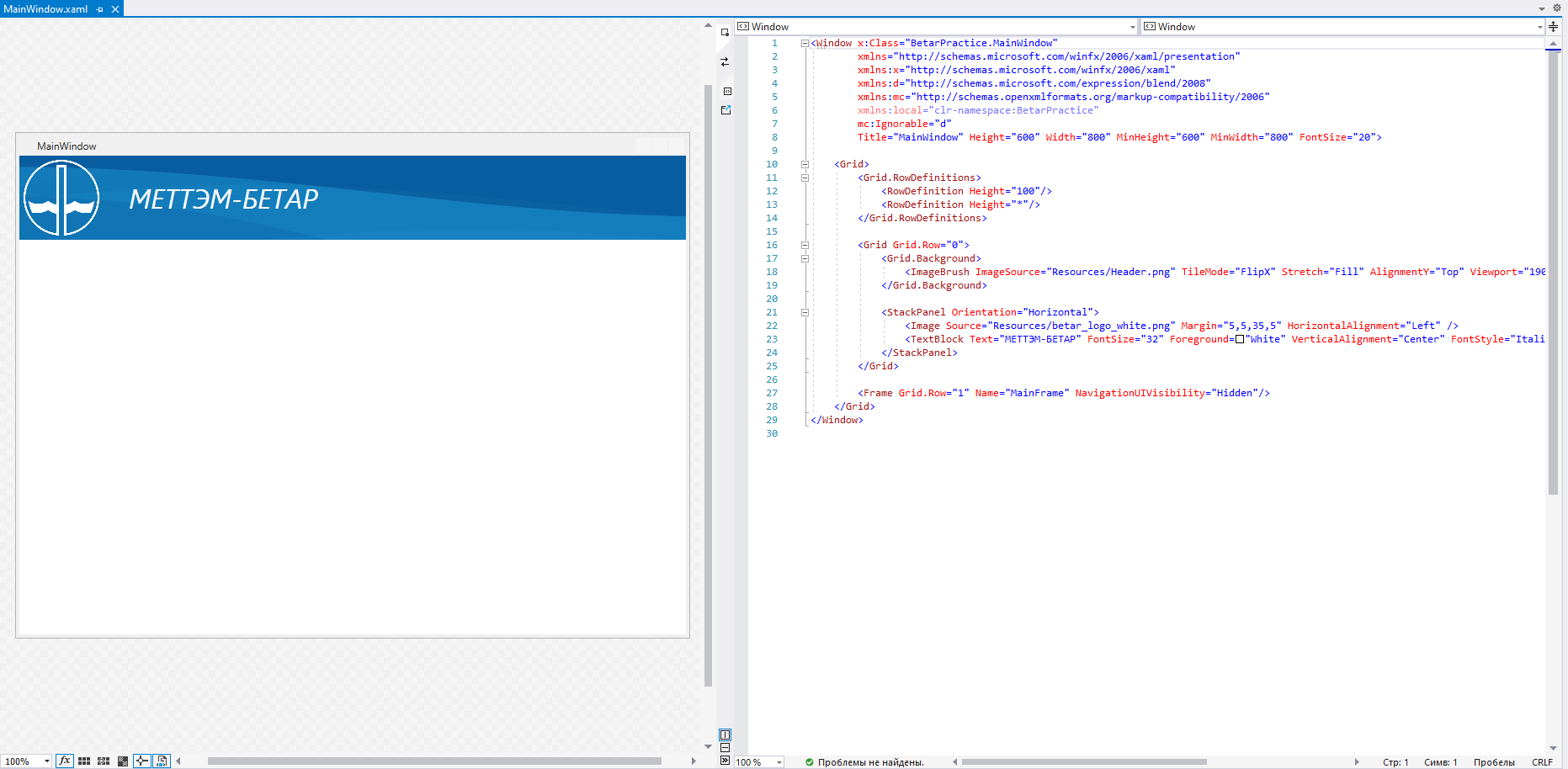


Рисунок 12. Каркас главного окна

Подключим к проекту базу данных. Для этого создадим в проекте модель ADO.NET: Проект > Добавить > Создать элемент > ADO.NET EDM.

На рисунке 13 изображено окно мастера моделей EDM.

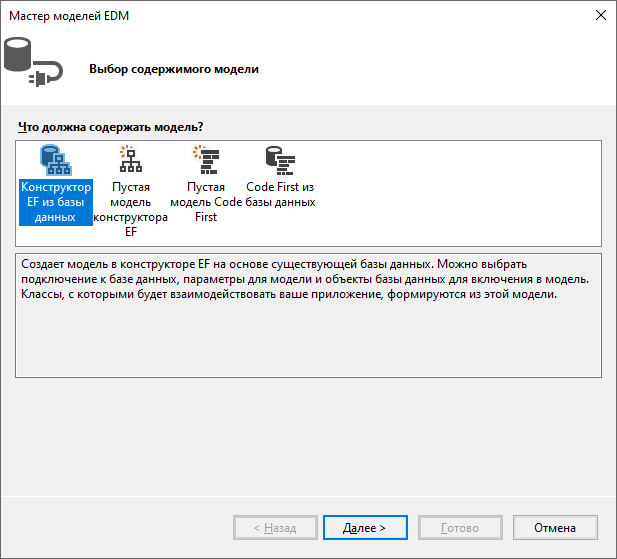


Рисунок 13. Окно мастера моделей EDM

Для обращения к модели данных откроем файл Model1.Context.cs и добавим приватное статичное поле, которое будет контекстом:

private static BetarPracticeEntities \_context;

Затем добавим метод получения экземпляра этого контекста:

public static BetarPracticeEntities GetContext()

{

if (\_context == null)

{

\_context = new BetarPracticeEntities();

}

return \_context;

}

Теперь создадим страницу и добавим на неё DataGrid для вывода списка продукции (Рис. 14).

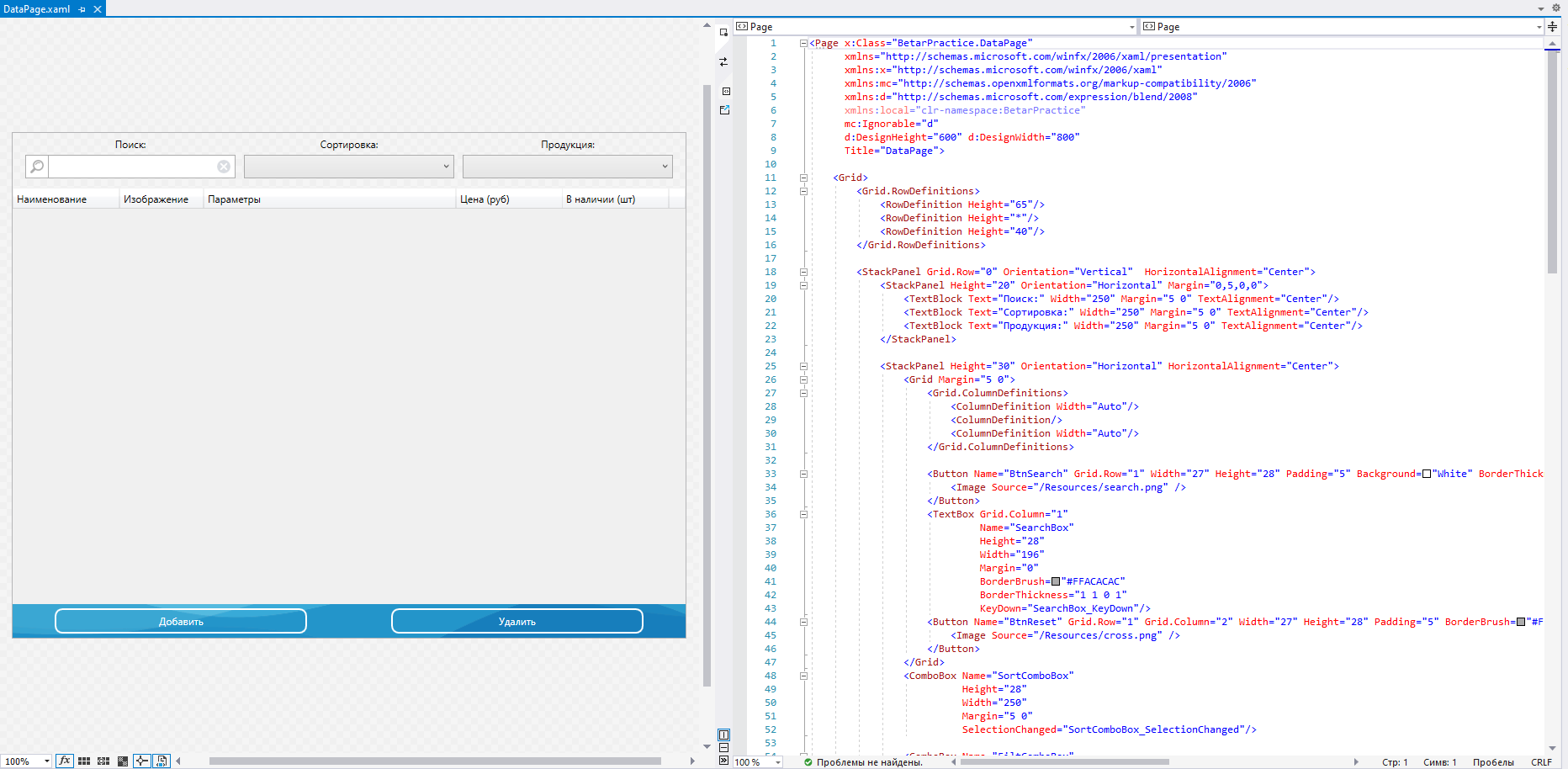


Рисунок 14. Страница для вывода списка продукции

На этом этапе были определены столбцы DataGrid: «Наименование», «Изображение», «Параметры», «Цена (руб)», «В наличии (шт)»; созданы кнопки «Добавить», «Удалить», «Редактировать»; а также добавлены элементы поиска и фильтрации данных («Поиск», «Сортировка», «Продукция») в соответствии с требованиями организации к приложению.

**Добавление изображений.**

Существует два способа загрузки изображений: в БД и на сервер.

Поскольку организация не планирует использование сервера, мы будем загружать изображения продукции непосредственно в БД. Для этого создадим SQL запрос.

Изображения будут храниться в организации в определенной папке, на которую мы будем ссылаться в запросе.

На рисунке 15 изображено окно выполнения запроса и результат.

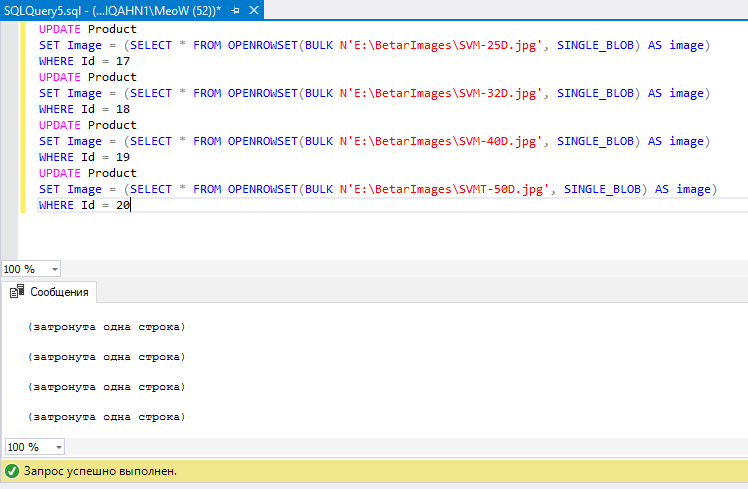


Рисунок 15. Окно выполнения запроса

Подобным образом загрузим в БД остальные изображения.

При необходимости добавить в БД новое изображение или заменить существующее, следует поместить его в папку с хранящимися изображениями и создать SQL запрос по примеру, описанному выше.

Затем перейдём к коду страницы и выведем список продукции в DataGrid.

public DataPage()

{

InitializeComponent();

DGrid.ItemsSource = BetarPracticeEntities.GetContext().Products.ToList();

}

На рисунке 16 показан результат вывода данных.

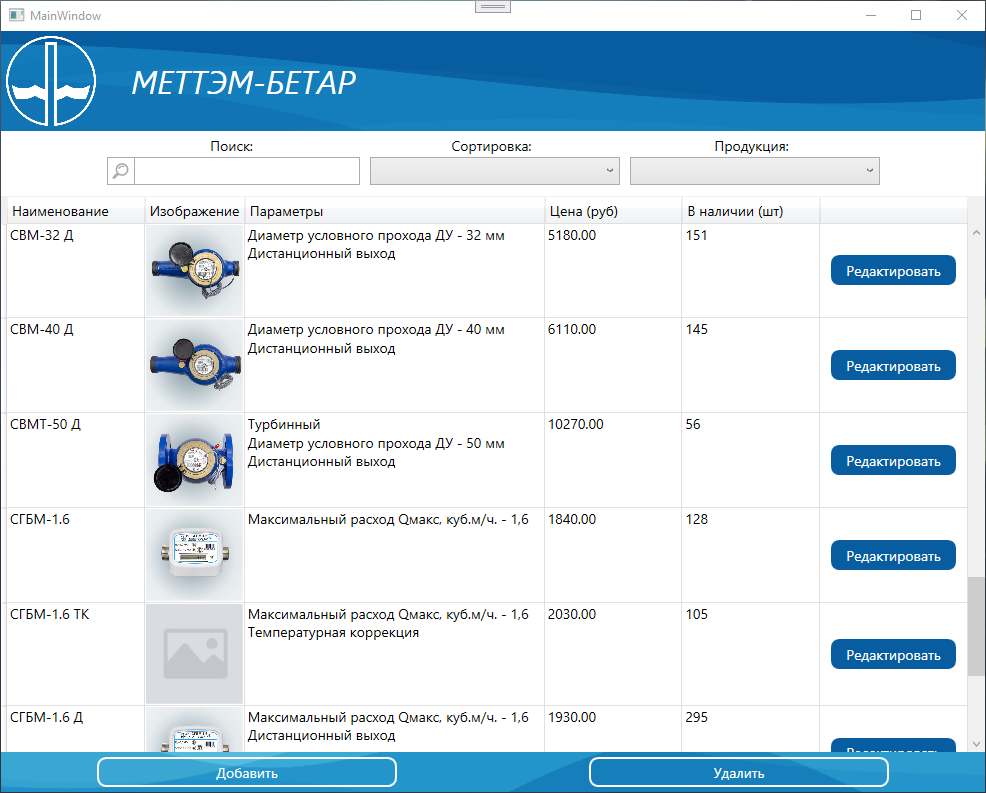


Рисунок 16. Вывод данных

**Реализация функции добавления данных через приложение.**

Для этого сначала создадим новую страницу (Рис. 17).

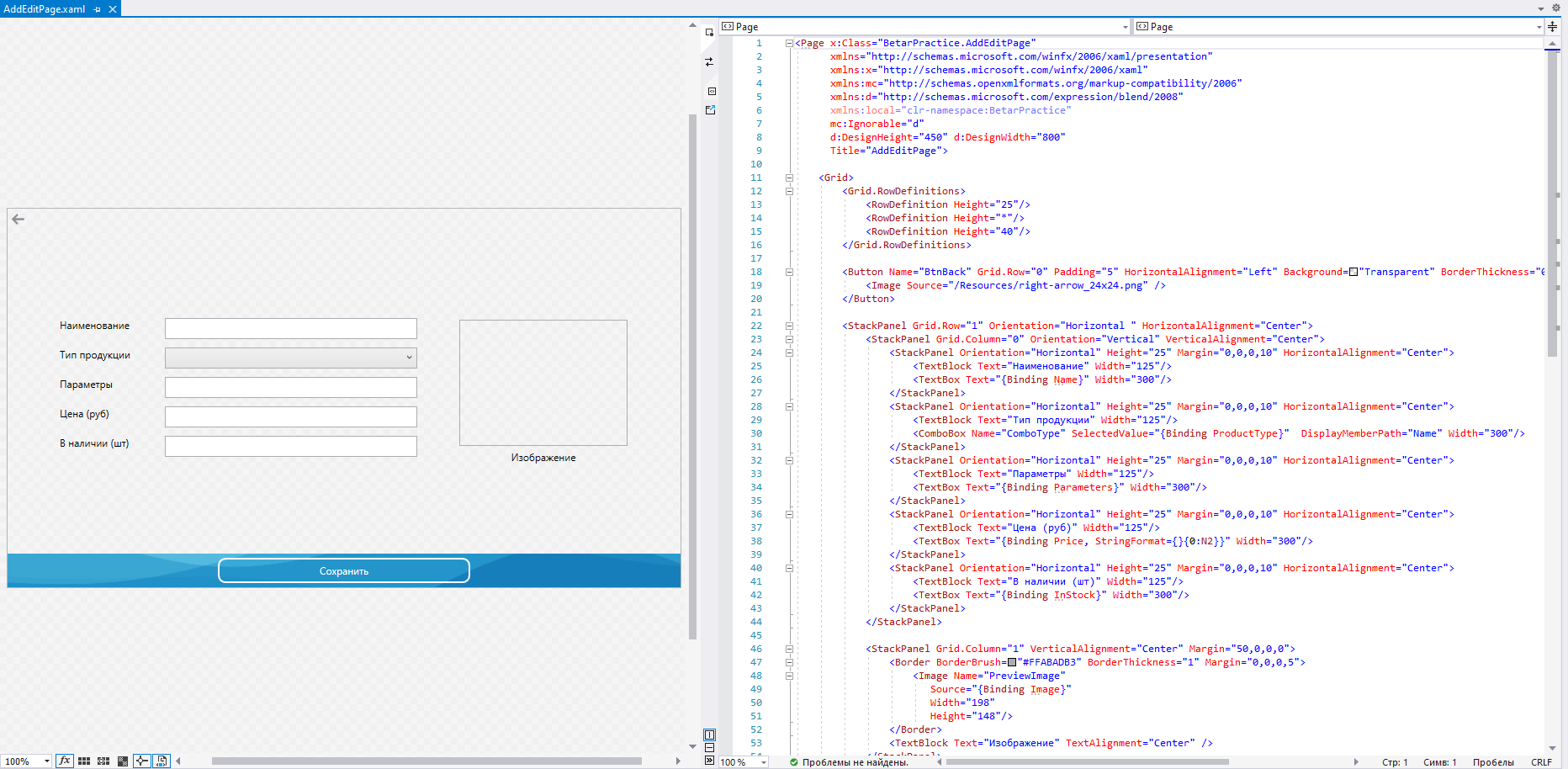


Рисунок 17. Каркас страницы для добавления и редактирования данных

Далее обработаем нажатие на кнопку «Сохранить» и в коде пропишем логику обращения к модели данных и добавления нового экземпляра продукта. Также сделаем проверки на заполняемость объектов, используя переменную errors.

На рисунке 18 показан код страницы добавления данных.



Рисунок 18. Код страницы добавления данных

Запускаем программу и проверяем функцию добавления данных (Рис. 19–20).

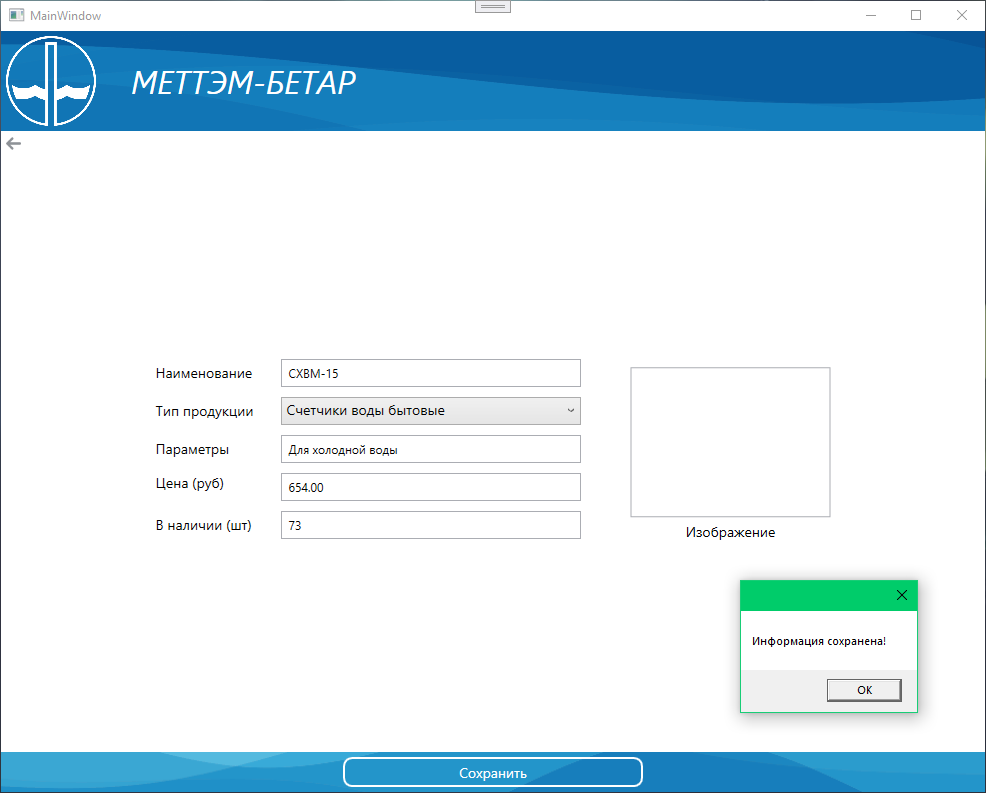


Рисунок 19. Сохранение добавленной информации

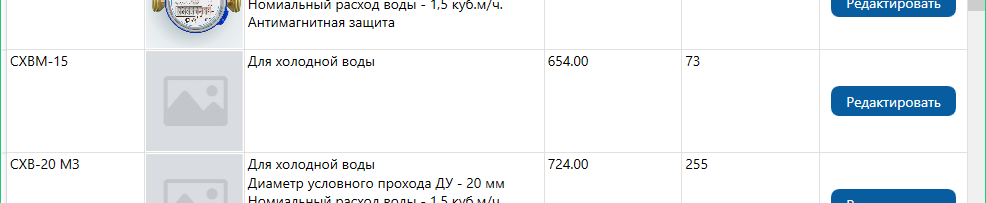


Рисунок 20. Добавленная информация

В нашем случае, при отсутствии изображения продукции, будет выводиться универсальная картинка из ресурсов проекта. Прописываем это в коде разметки страницы:

<Image MinWidth="100">

<Image.Source>

<Binding Path="Image">

<Binding.TargetNullValue>

<ImageSource>/Resources/EmptyImage.jpg</ImageSource>

</Binding.TargetNullValue>

</Binding>

</Image.Source>

</Image>

**Реализация функции редактирования.**

Для функции редактирования данных целесообразно использовать ту же страницу, что и для их добавления. Измененная информация будет фиксироваться в базе данных и отображаться в списке.

В первую очередь добавим параметр нашей странице AddEditPage. В нее мы будем передавать экземпляр выбранного продукта и, в случае если он не пустой, присваивать нашему полю CurrentProduct.

public AddEditPage(Product selectedProduct)

{

InitializeComponent();

if (selectedProduct != null)

{

\_currentProduct = selectedProduct;

}

DataContext = \_currentProduct;

ComboType.ItemsSource = BetarPracticeEntities.GetContext().ProductTypes.ToList();

}

Вызов страницы AddEditPage теперь требует какого-то аргумента. В случае если мы будем делать добавление, мы просто пропишем null (отправим пустой экземпляр). При этом для редактирования BtnEdit мы уже будем передавать экземпляр, прописав для этого код. Вместо null будем обращаться к кнопке, на которую нажали, получать ее контекст и знать, что это — продукт.

private void BtnAdd\_Click(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new AddEditPage(null));

}

private void BtnEdit\_Click(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new AddEditPage((sender as Button).DataContext as Product));

}

Теперь те данные, которые мы хотим отредактировать, автоматически привязались к этим элементам управления и отображаются корректно. В случае если мы отредактируем какое-то поле (например, поле «В наличии (шт)») и нажмём кнопку «Сохранить» — информация будет обновлена (Рис. 21–22).

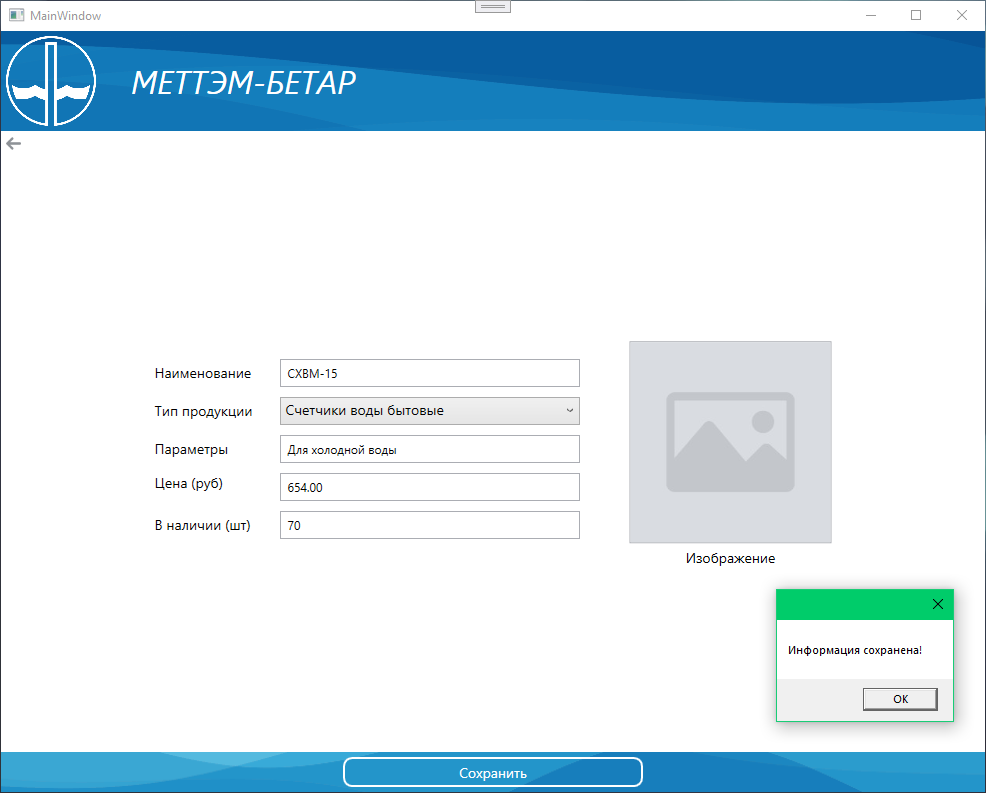


Рисунок 21. Сохранение измененных данных

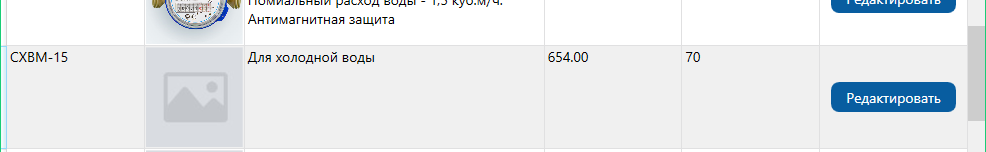


Рисунок 22. Результат изменения данных

**Реализация функции удаления.**

Любые действия, безвозвратно изменяющие данные в БД, должны запрашивать подтверждение пользователя. Поэтому, в первую очередь, для нажатия на кнопку удаления мы реализуем сообщение с вопросом: действительно ли пользователь хочет это сделать.

Получим список продукции для удаления, обратившись к таблице с продукцией. Выберем все элементы, которые мы выделили, преобразуем их в список продукции. Затем, в сообщении будем спрашивать пользователя: «Вы точно хотите удалить следующие productsForRemoving.Count() элементов?». Укажем здесь заголовок сообщения — «Внимание», затем укажем, какие кнопки доступны при диалоге с пользователем: «Да» или «Нет», и выберем изображение — «Question».

Если результатом диалога от пользователя было нажатие на кнопку «yes», то мы будем выполнять удаление. Для этого обратимся к модели данных, используя блок TryCatch. Получив контекст, попробуем с помощью метода RemoveRange удалить все полученные выделенные строки. В случае если все будет хорошо, отобразим сообщение. Иначе выводим сообщение об ошибке. Также, в случае если удаление произойдет успешно, отдельно вызовем обновление актуальных данных (Рис. 23).



Рисунок 23. Обработчик кнопки удаления

Проверим работу приложения на предмет удаления данных (Рис. 24–25).

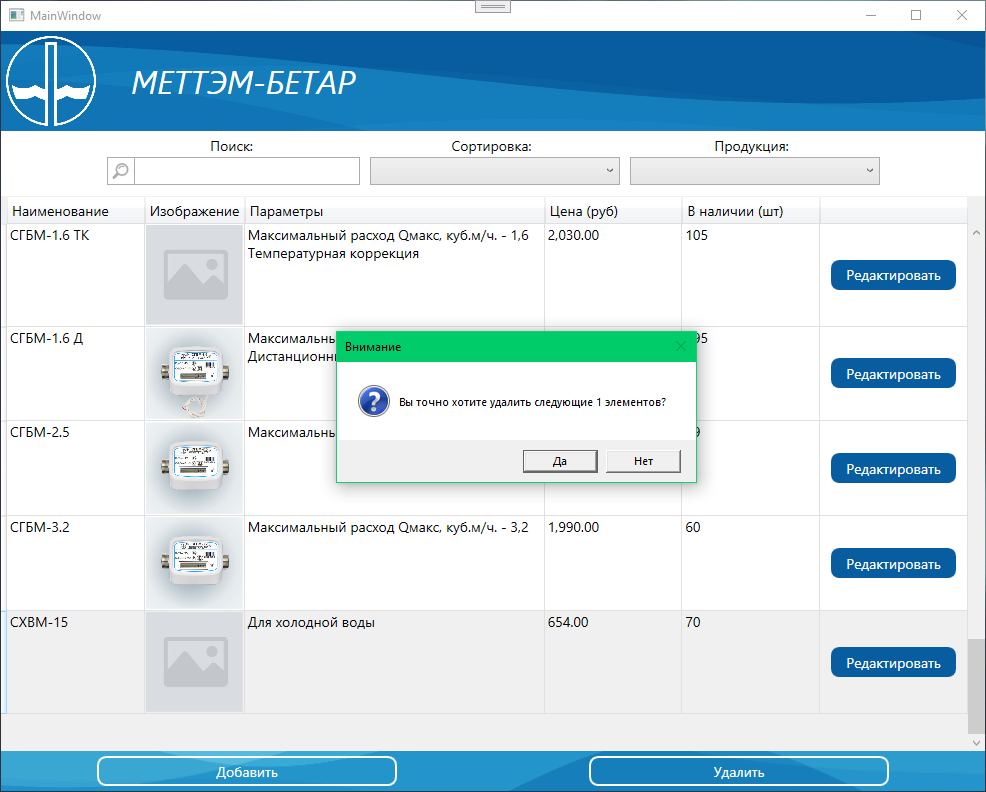


Рисунок 24. Запрос на подтверждение удаления

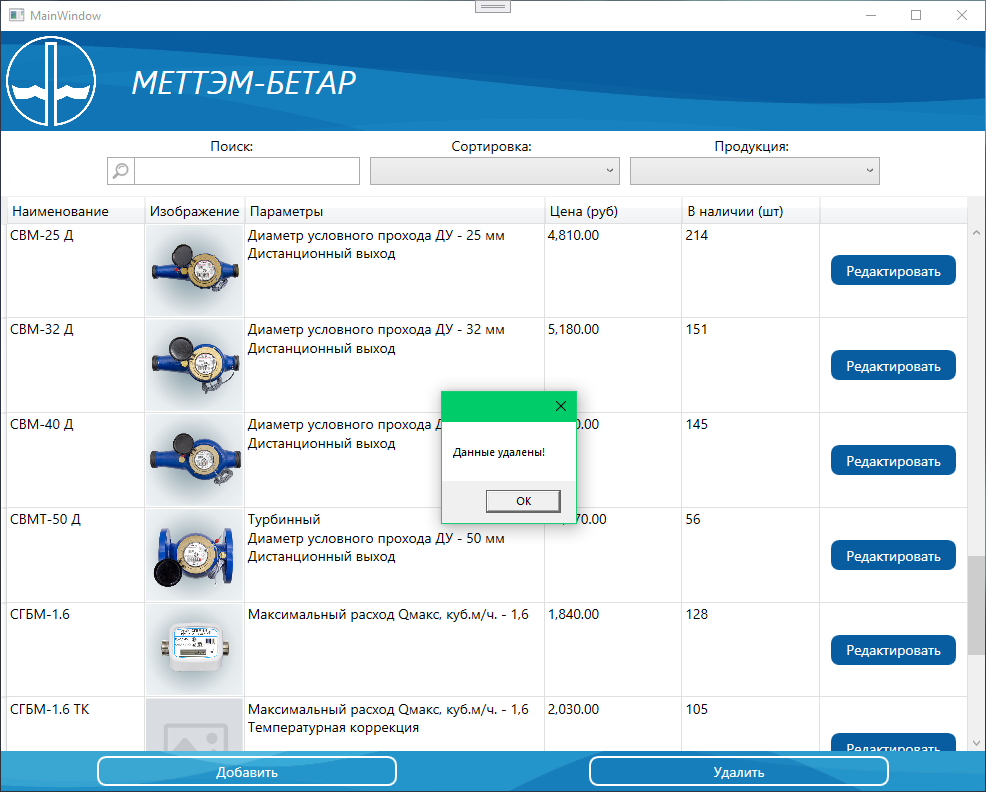


Рисунок 25. Удаление данных

**Реализация поиска, сортировки и фильтрации данных.**

Для работы с большими объемами информации полезно реализовать поиск, сортировку и фильтрацию.

Загрузим данные в FiltComboBox и SortComboBox и установим стартовые значения для элементов управления:

var allTypes = BetarPracticeEntities.GetContext().ProductTypes.ToList();

allTypes.Insert(0, new ProductType { Name = "Все виды" });

var Filt = new List<string>() { "Все типы" };

Filt.AddRange(BetarPracticeEntities.GetContext().ProductTypes.Select(c => c.Name).ToList());

FiltComboBox.ItemsSource = Filt;

FiltComboBox.SelectedIndex = 0;

SortComboBox.Items.Add("Цена (по возрастанию)");

SortComboBox.Items.Add("Цена (по убыванию)");

SortComboBox.Items.Add("В наличии (по возрастанию)");

SortComboBox.Items.Add("В наличии (по убыванию)");

SortComboBox.SelectedIndex = 0;

Обработаем методы поиска и фильтрации так, чтобы они работали вместе. Для этого создадим метод Update и выполним в нем фильтрацию коллекции, поиск по ней и сортировку, а затем вызовем этот метод из обработчиков событий всех наших элементов управления (Рис. 26).

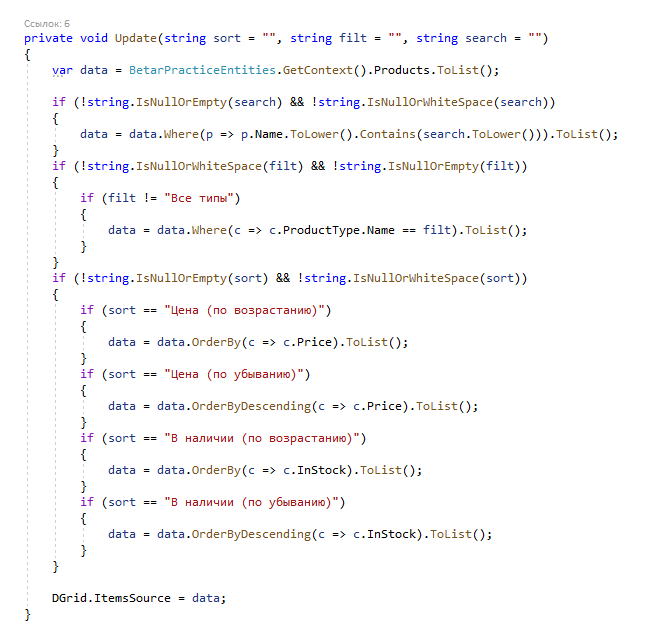


Рисунок 26. Метод Update: поиск, фильтрация и сортировка данных

Проверим работоспособность функций поиска, сортировки и фильтрации данных в приложении (Рис. 27–29).

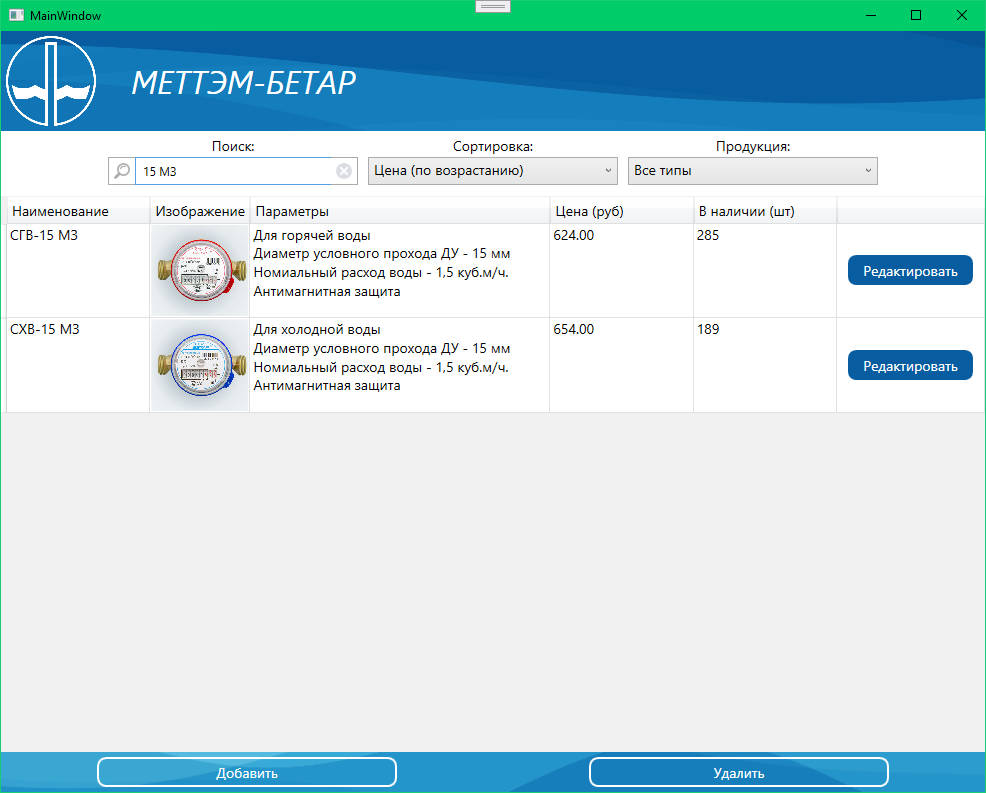


Рисунок 27. Поиск по введенному запросу

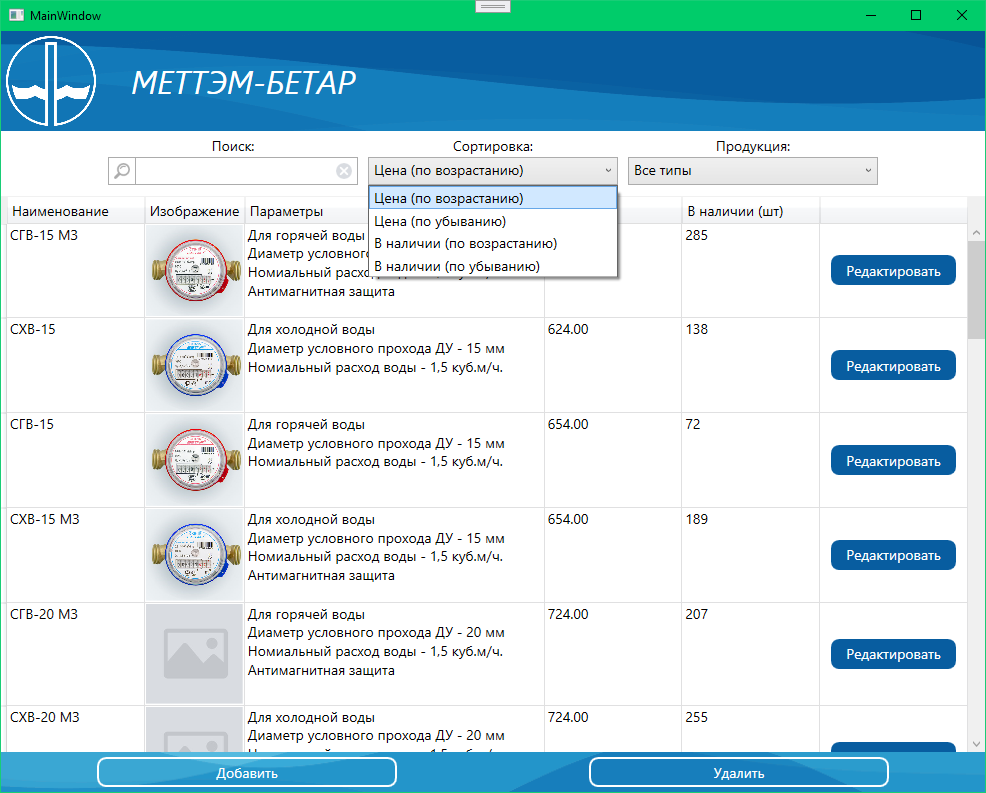


Рисунок 28. Сортировка по возрастанию цены

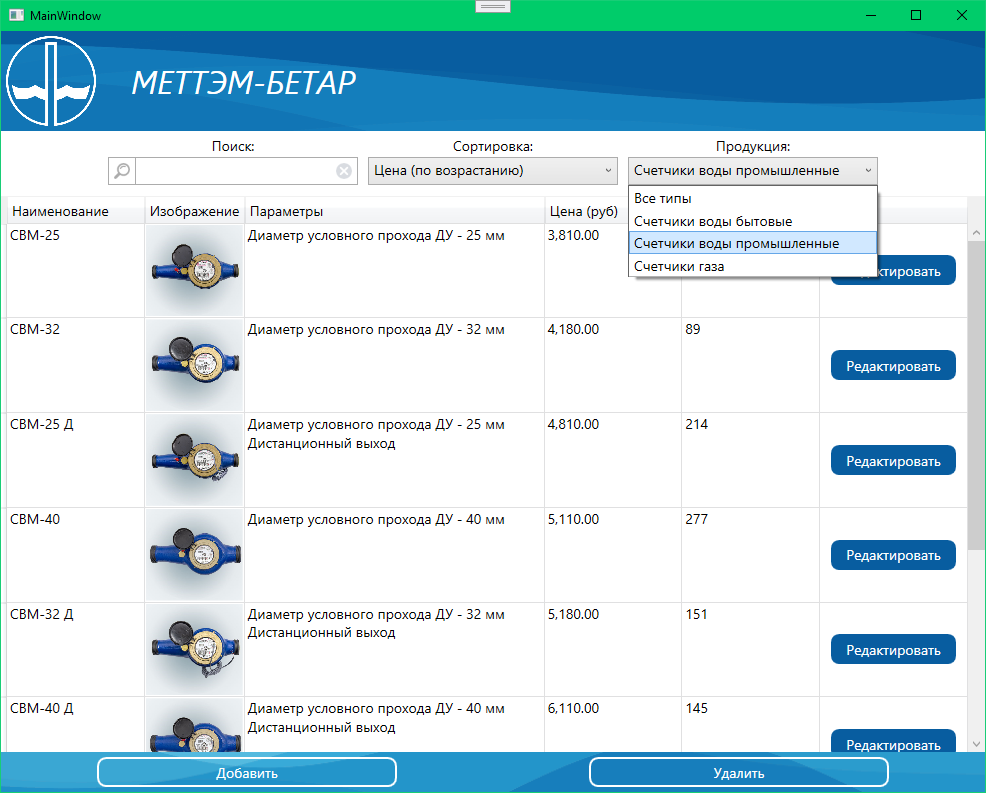


Рисунок 29. Фильтрация по типу продукции

Выводы

По окончании производственной практики были решены следующие задачи:

* Описана предметная область.
* Изучены программные средства.
* Создан каркас приложения.
* Проведено функциональное тестирование.
* Осуществлен рефакторинг и оптимизация программного кода.

Вид профессиональной деятельности был освоен, а также были освоены профессиональные компетенции:

ПК 1.1. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.

ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием.

ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК 1.4. Выполнять тестирование программных модулей.

ПК 1.5. Осуществлять рефакторинг и оптимизацию программного кода.

ПК 1.6. Разрабатывать модули программного обеспечения для мобильных платформ.

Список литературы

1. GitHub репозиторий — URL https://github.com/KostinMakar
2. WPF и C# | Полное руководство — URL https://metanit.com/sharp/wpf/
3. Документация по C#. Начало работы, руководства, справочные материалы. — URL https://docs.microsoft.com/ru-ru/dotnet/csharp/
4. НОУ ИНТУИТ | Лекция | Основы языка XAML — URL https://intuit.ru/studies/courses/690/546/lecture/12351
5. Программные решения для бизнеса — Национальная сборная Worldskills Россия — URL https://nationalteam.worldskills.ru/skills/programmnye-resheniya-dlya-biznesa/
6. Продукция — URL http://betar.ru/catalog/
7. Счетчики воды бытовые, промышленные, Счетчики газа, тепла и электричества, Меттэм-Бетар, 8(985)764-9834, www.betar-moskva.ru — URL http://www.betar-moskva.ru/production.htm