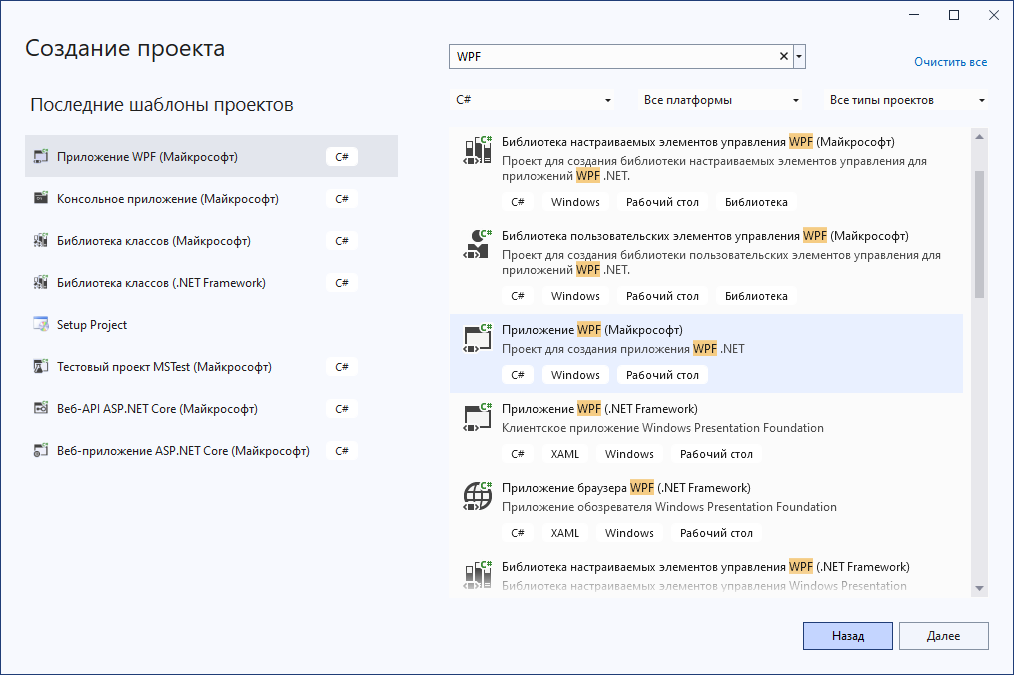
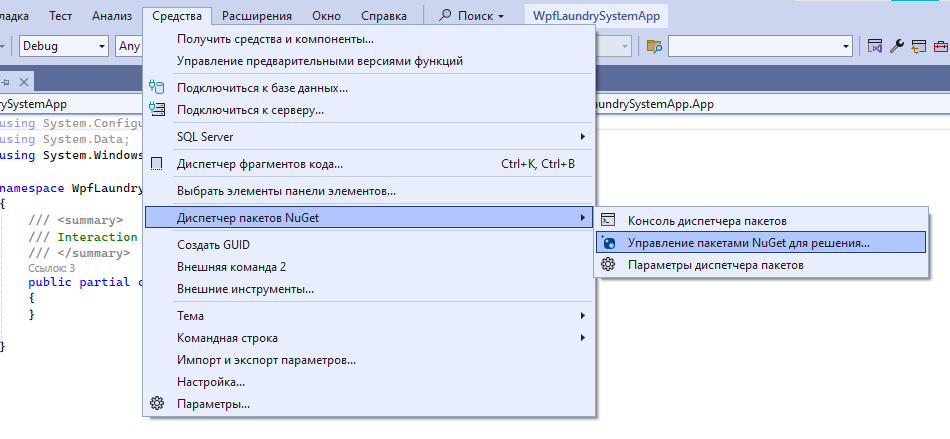
**Отчёт по 4 заданию**

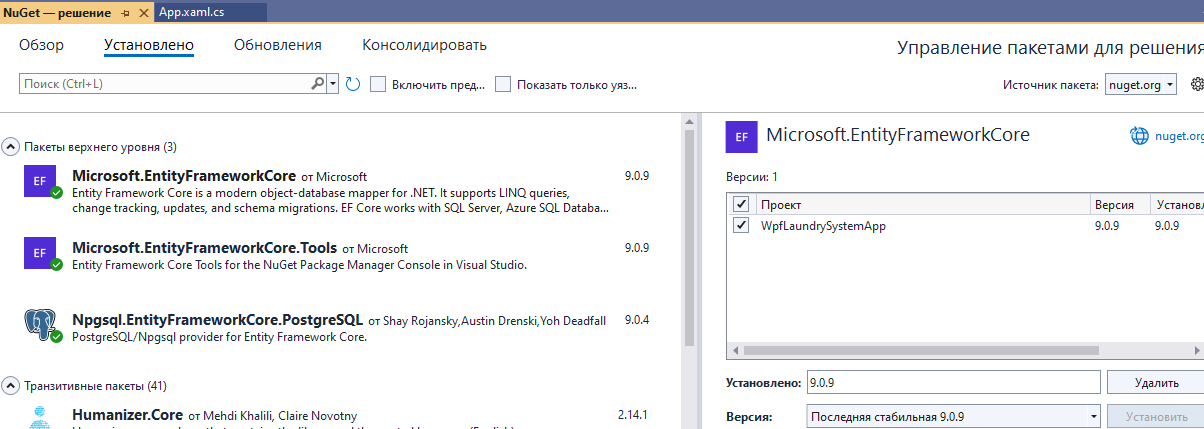
Первым делом создаётся новый проект в VisualStudio Приложение WPF (Майкрософт)



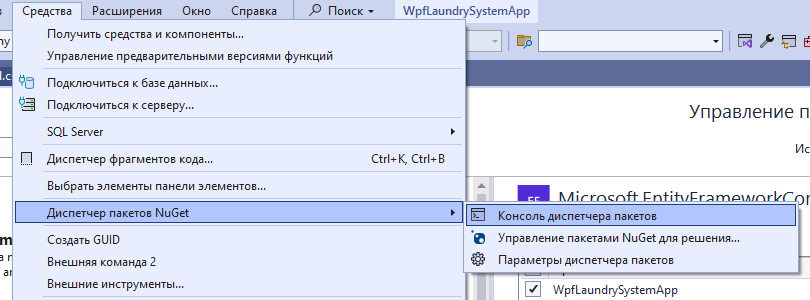
Затем в Средства > Диспетчер пакетов NuGet > управление пакетами NuGet для решения скачиваем все нужные пакеты



Это пакеты EntityFramework.Core EntityFramework.Tools и NpgSQL. EntityFramework.PostgreSQL



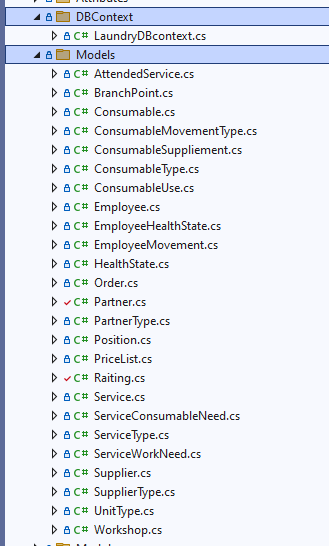
Затем в Средства > Диспетчер пакетов NuGet > консоль диспетчера пакетов



Вводим следующую команду для создания модели БД

dotnet ef dbcontext scaffold "Host=localhost;Port=5432;Database=laundryStudPractice4Course;Username=postgres;Password=POSTGREmoiseiev;SSL Mode=Prefer" Npgsql.EntityFrameworkCore.PostgreSQL -o Models -c ApplicationDbContext --data-annotations –force

Результатом команды стали файлы классов моделей сущностей

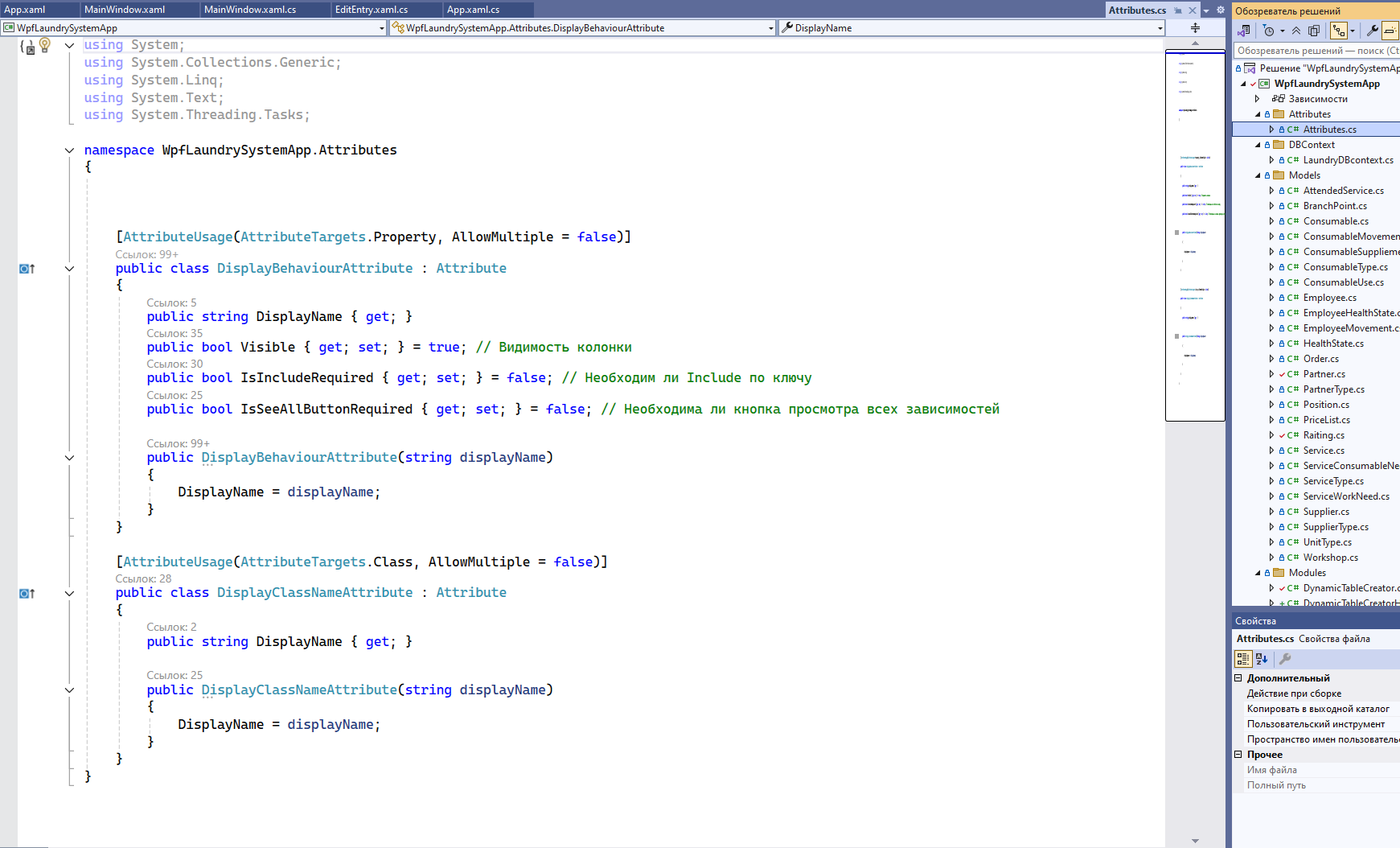


В данной работе было решено подойти к разработке универсального класса интерфейса, поэтому создаём модуль DynamicTableCreator, суть которого будет в том, что с его помощью и использованием рефлексии C#, а также метаданных EntityFramework автогенерировать интерфейс (код модуля DynamicTableCreator можно просмотреть в репозитории гитхаб)





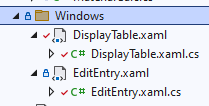
Для работы DynamicTableCreator классы моделей БД необходимо разметить специальносозданными атрибутами:



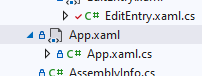
Пример разметки на Partners

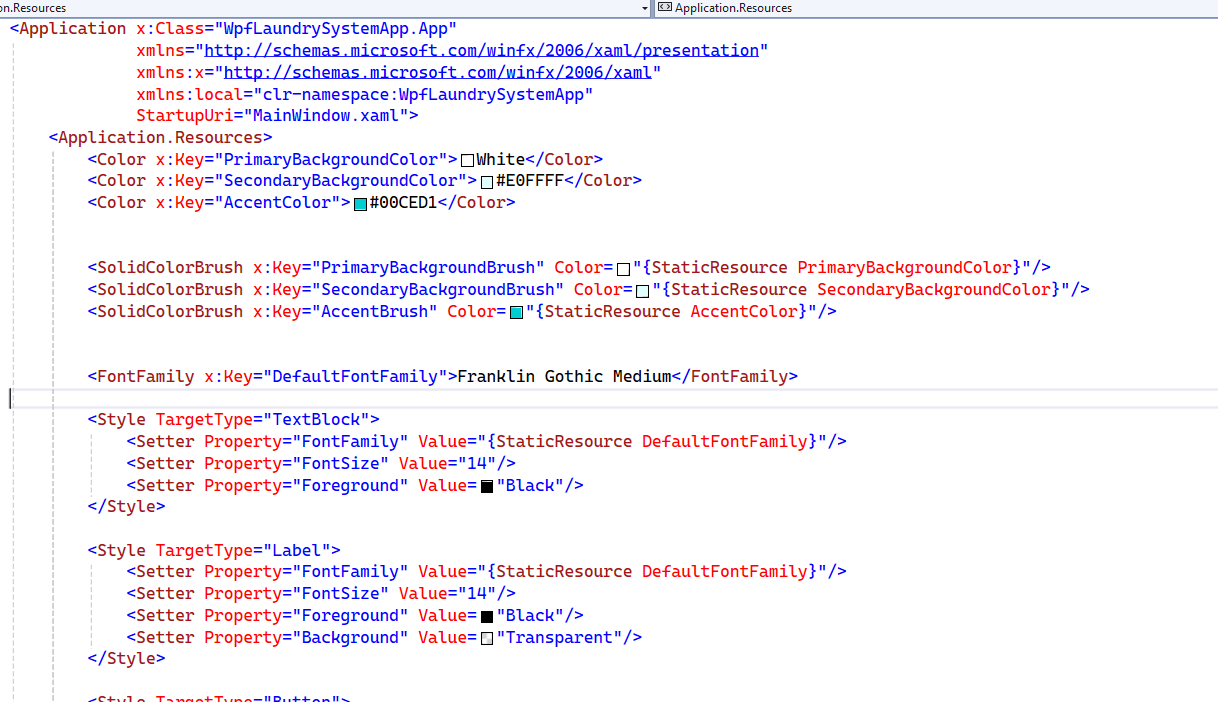


Далее создадим два окна, через которые будут происходить взаимодействия с БД: DisplayTable и EditEntry (их код также в репозитории)

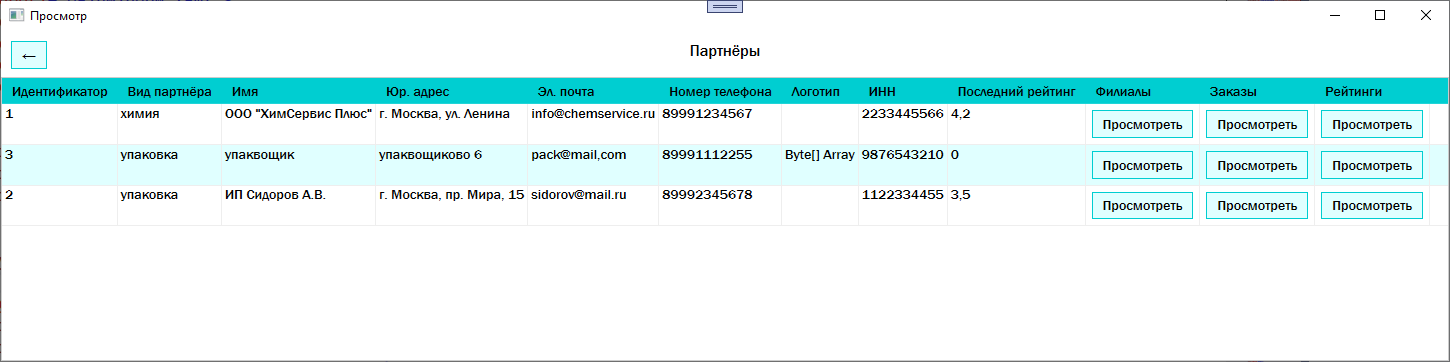


Далее следует настроить стиль в соответствии с ТЗ, для этого в App.xaml в Resources добавим необходимые настройки (весь код app.xaml также в репозитории)



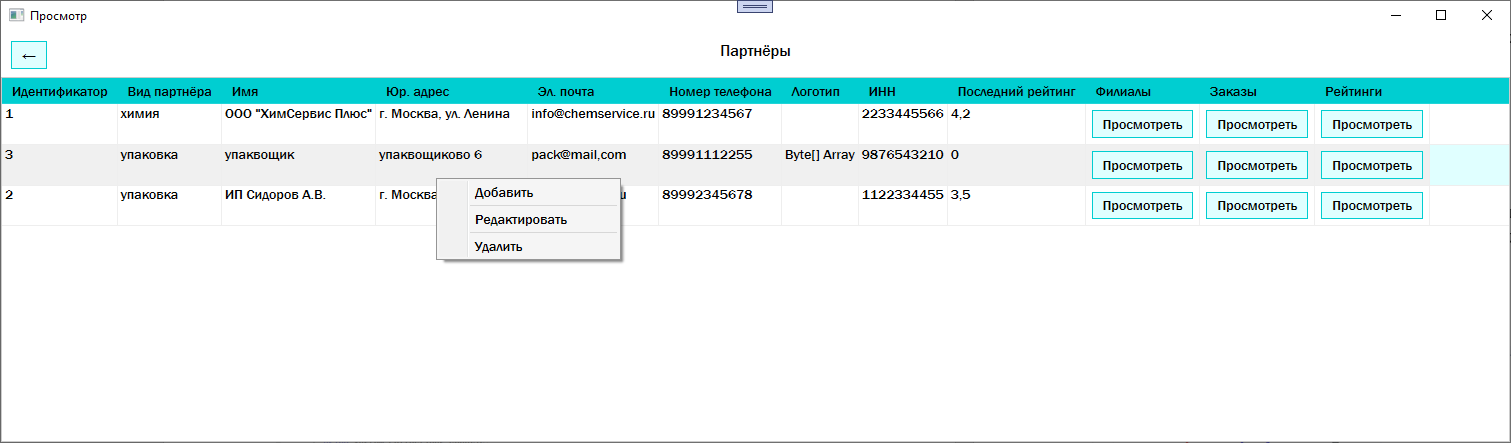


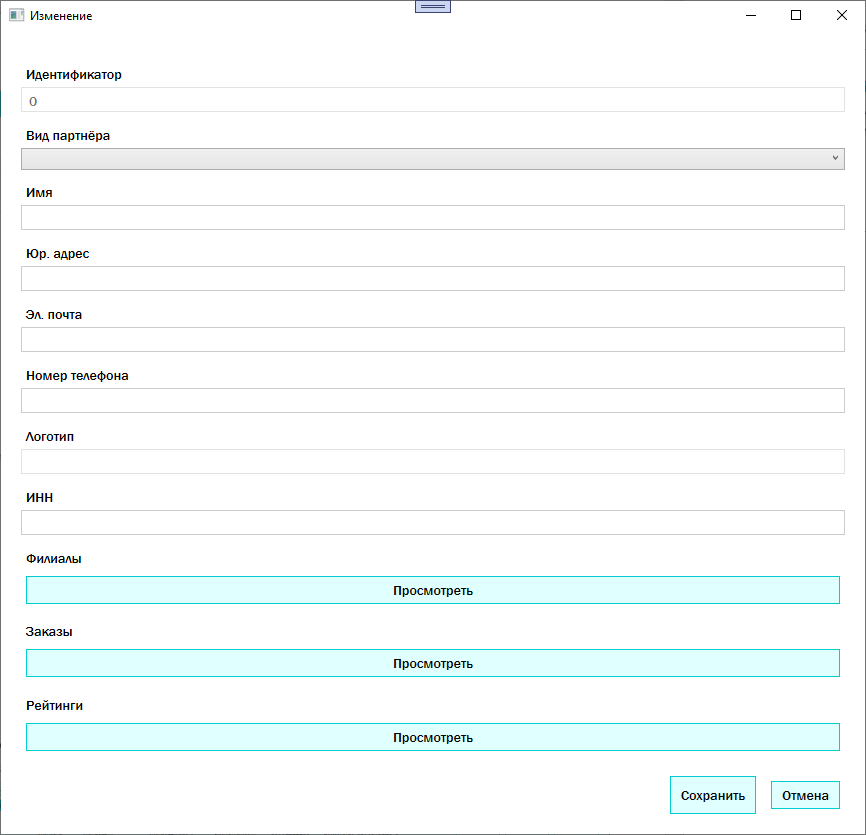
По итогу выходит следующее приложение:



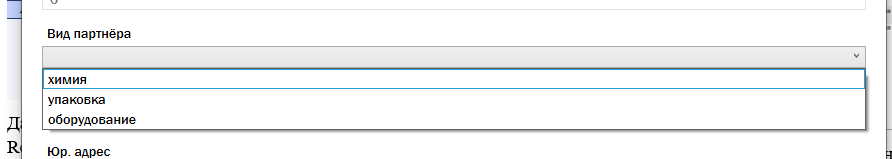
По умолчанию запускается таблица партнёры

При нажатии ПКМ появляется контекстное меню и в нём 3 опции.

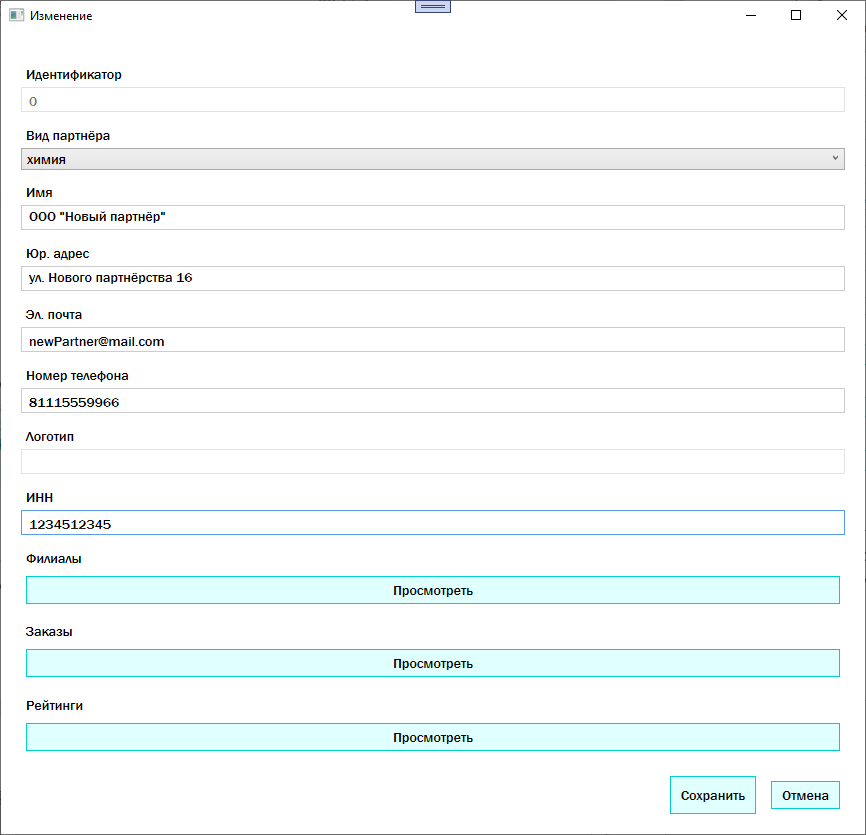


Добавление открывает новое окно с автоматически сгенерированными полями ввода

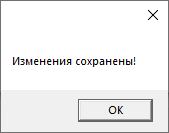
А вместо внешних ключей выпадающие списки (идентификатор и логотип вручную редактировать невозможно)



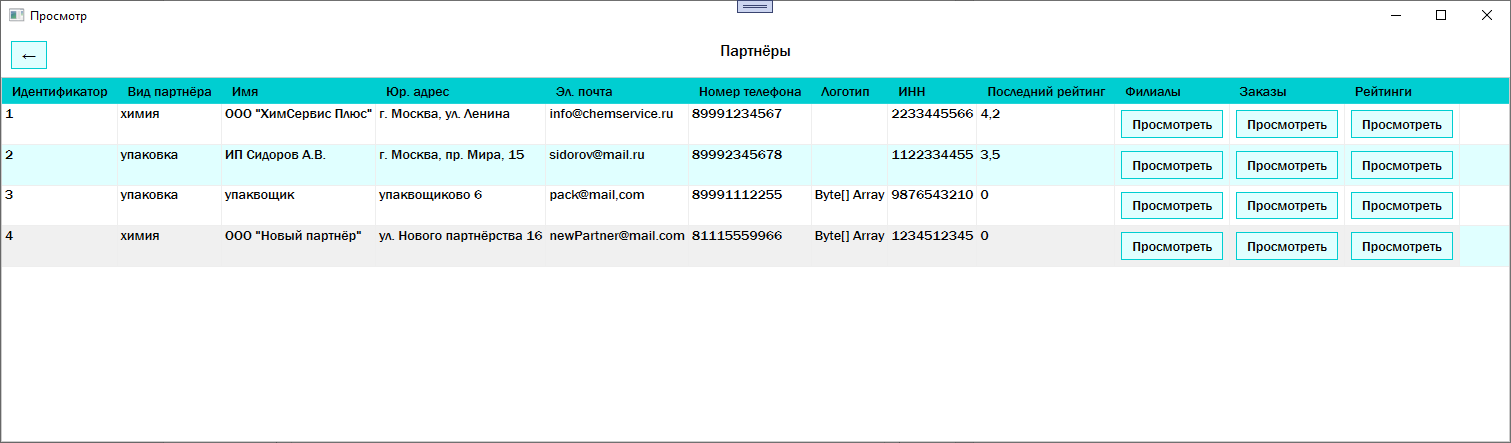
Вводим данные



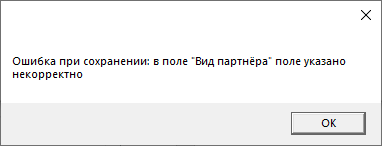
Сообщение о сохранении



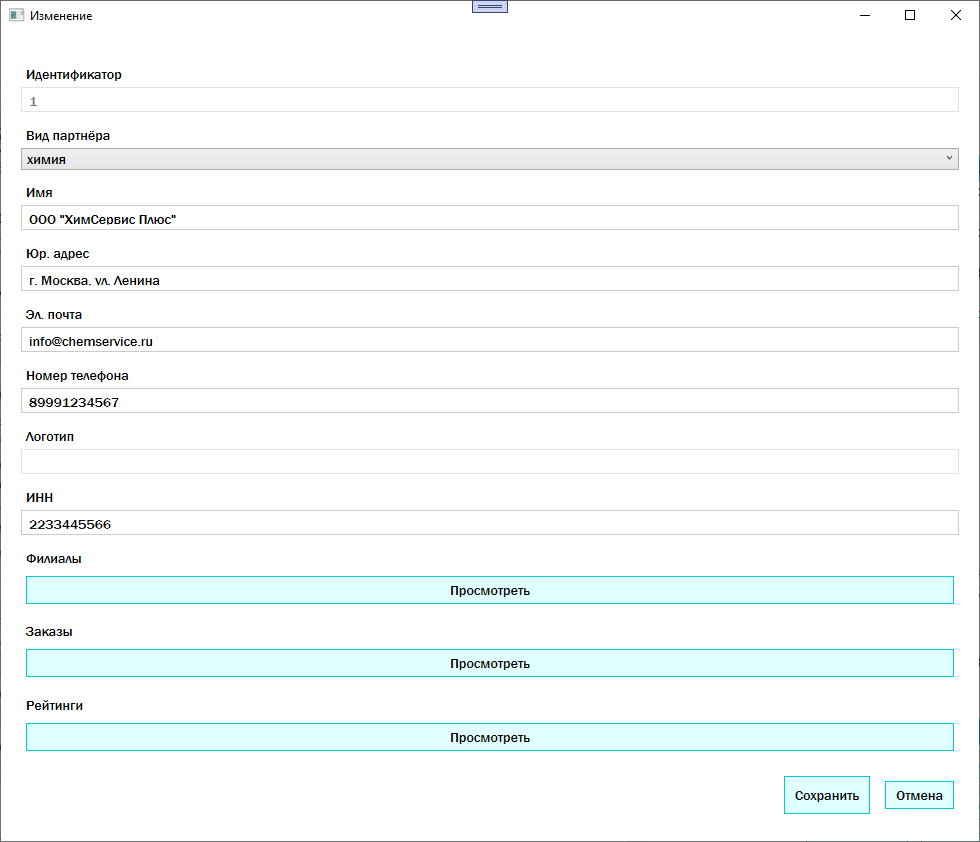
И новый партнёр добавлен



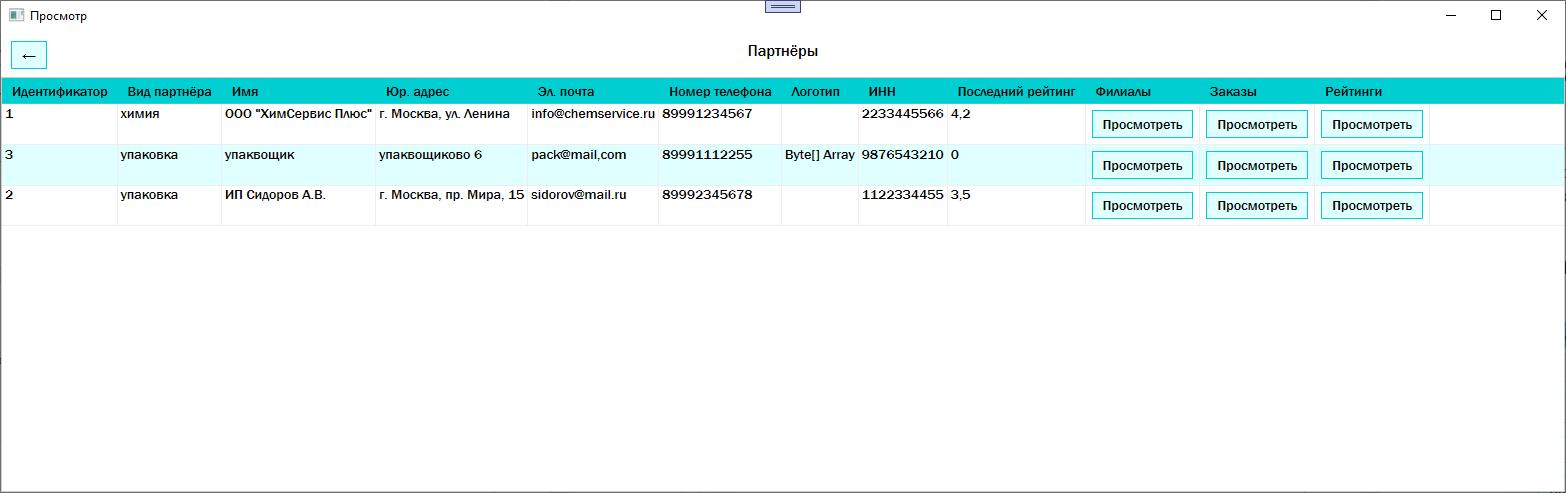
Если не выбрать вид партнёра при сохранении, то будет выведено следующее сообщение



Редактирование происходит аналогично, только открывается окно с уже заполненными полями



Удаление соответсвенно, удаляет запись



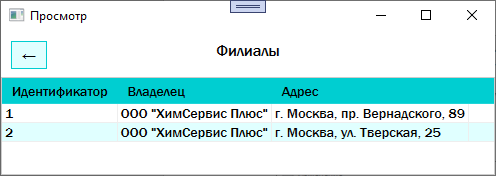
И новый партнёр не будет создан, а пользователь сможет продолжить редактировать и всё же выбрать вид партнёра.

У партнёра есть три столбца с и коллекциями

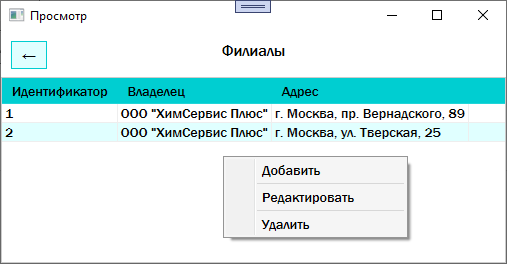


Это их филиалы, их заказы и их рейтинги (история рейтингов)

Нажатие на любой из них откроет советующую таблицу с записями, относящимися к выбранному партнёру.



В этих таблицах абсолютно аналогично можно вызвать контекстное меню

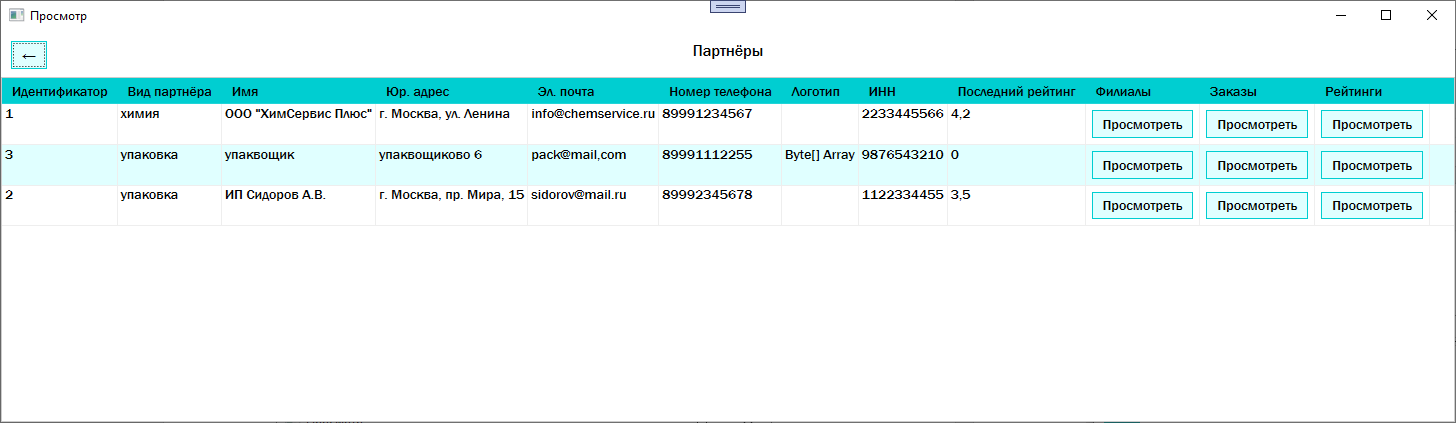


Которое вызывает соответсвующее автогенерируемое окно изменений



Чтоб вернуться на предыдущую таблицу можно нажать кнопку со стрелочкой влево

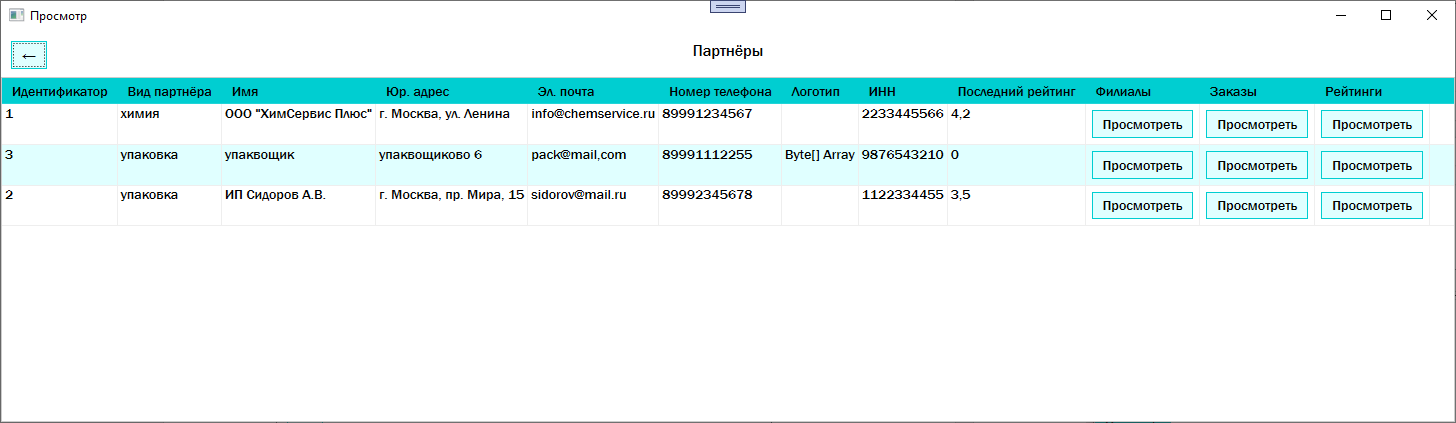


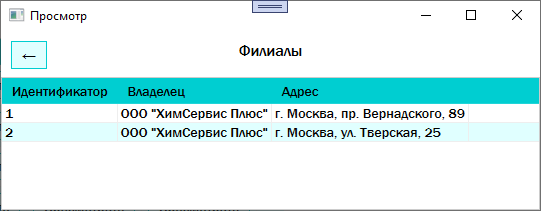


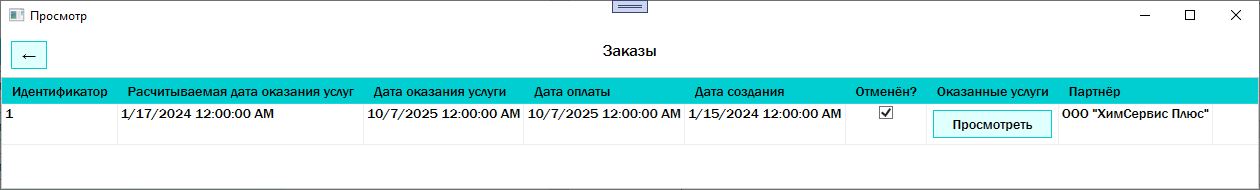
У партнёра также отображается колонка с актуальным (то есть самым новым рейтингом)

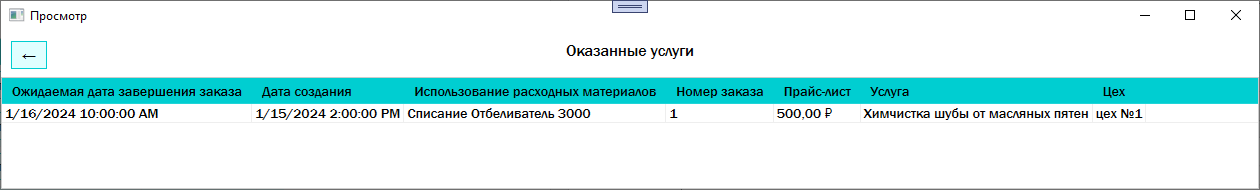


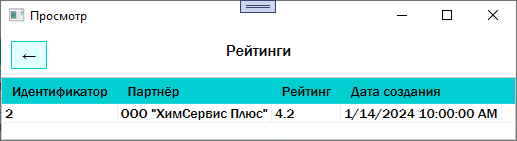
Просмотрим все таблицы, которые можно просмотреть:

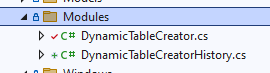


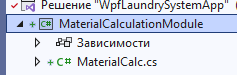








По итогу вышло 3 модуля  




Модуль MaterialCalc – отдельный модуль, согласно ТЗ практики

**Код модуля MaterialCalc**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WpfLaundrySystemApp.Modules

{

public static class MaterialCalc

{

public static int CalculateMaterialAmount(

int serviceTypeId,

int materialTypeId,

int serviceCount,

params double[] parameters)

{

if (serviceCount <= 0)

throw new ArgumentException("Количество услуг должно быть больше нуля.", nameof(serviceCount));

if (parameters is null || parameters.Length == 0)

throw new ArgumentException("Необходимо указать хотя бы один параметр услуги.", nameof(parameters));

double serviceCoefficient = ServiceCoefficient(serviceTypeId);

double wastePercent = MaterialWastePercent(materialTypeId);

double baseAmount = 1;

foreach (var p in parameters)

{

if (p <= 0)

throw new ArgumentException("Параметры услуги должны быть положительными числами.");

baseAmount \*= p;

}

double totalMaterial = baseAmount \* serviceCoefficient \* serviceCount \* ((100 + wastePercent) / 100.0);

return (int)Math.Ceiling(totalMaterial);

}

private static double ServiceCoefficient(int serviceTypeId) =>((double)serviceTypeId) / 2;

private static double MaterialWastePercent(int materialTypeId) => materialTypeId \* 1.5;

}

}