**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ОТЧЕТ**

**по учебной практике УП.02.01 (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/3 группы

Писарец Андрей Андреевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «20» февраля 2023 г. по «18» марта 2023 г.

Руководитель практики Кузнецов К.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2023

**ЗАДАНИЕ**

**на учебную практику (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/3 группы

Писарец Андрей Андреевич

(Фамилия, имя, отчество)

Место прохождения практики:УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «20» февраля 2023 г. по «18» марта 2023 г.

**Виды работ, обязательные для выполнения***(переносится из программы, соответствующего ПМ):*

* Участие в выработке требований к программному обеспечению;
* Стадии проектирования программного обеспечения;
* Разработка модулей программного обеспечения;
* Тестирование программных модулей и их интеграции;
* Разработка программной документации и стандарты кодирования.

**Индивидуальное задание: ВАРИАНТ 14**

Задание выдал «20» февраля 2023 г. \_\_\_\_\_\_\_\_\_\_\_\_\_ Кузнецов К.С.

(подпись) (Ф.И.О.)

Задание получил «20» февраля 2023 г. \_\_\_\_\_\_\_\_\_\_\_\_ Писарец А.А. (подпись) (Ф.И.О.)

**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ДНЕВНИК**

**прохождения учебной практики УП.02.01**

**(по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/3 группы

Писарец Андрей Андреевич

(Фамилия, имя, отчество)

Место прохождения практики:УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «20» февраля 2023 г. по «18» марта 2023 г.

Руководитель практики Кузнецов К.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике

Санкт-Петербург

2023

**Содержание дневника**

|  |  |  |
| --- | --- | --- |
| **Дата** | **Виды выполненных работ и заданий по программе практики** | **Подпись руководителя практики** |
| **1** | **2** | **3** |
| 20.02.23 | Анализ предметной области |  |
| 21.02.23 | Выявление требований к программе |  |
| 22.02.23 | Разработка технического задания |  |
| 25.02.23 | UML. Проектирование диаграммы вариантов использования |  |
| 27.02.23 | UML. Проектирование диаграммы последовательности |  |
| 28.02.23 | UML. Проектирование диаграммы активности |  |
| 01.03.23 | Моделирование структуры ПО |  |
| 02.03.23 | Проектирование инфологической и даталогической модели данных |  |
| 03.03.23 | Проектирование интерфейса пользователя. Создание Wireframe эскизов |  |
| 04.03.23 | Разработка дизайна программы в соответствии с руководством по стилю |  |
| 06.03.23 | Разработка базы данных |  |
| 07.03.23 | Разработка словаря данных |  |
| 09.03.23 | Создание приложения. Форма авторизации |  |
| 10.03.23 | Создание приложения. Форма заказов |  |
| 11.03.23 | Создание приложения. Основные формы приложения |  |
| 13.03.23 | Разработка библиотеки классов |  |
| 14.03.23 | Подготовка отчетов и выгрузка документов для печати |  |
| 15.03.23 | Отладка программных модулей |  |
| 16.03.23 | Модульное тестирование |  |
| 16.03.23 | Создание тестовых случаев |  |
| 16.03.23 | Интеграционное тестирование |  |
| 16.03.23 | Разработка самодокументирующегося кода |  |
| 16.03.23 | Инспекция кода на соответствие стандартам кодирования |  |
| 18.03.23 | Подготовка отчета и размещение результатов в репозитории контроля версий |  |

**АТТЕСТАЦИОННЫЙ ЛИСТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ УП.02.01 (ПО ПРОФИЛЮ СПЕЦИАЛЬНОСТИ)**

по профессиональному модулю ПМ.02«Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/3 группы

Писарец Андрей Андреевич

(Фамилия, имя, отчество)

Место прохождения практики:УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «20» февраля 2023г. по «18» марта 2023 г.

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| **Виды выполненных работ обучающимся**  **во время практики** | **Объем работ, час.** | **Качество выполнения работ (оценка по пятибалльной системе)** |
| Участие в выработке требований к программному обеспечению | 18 |  |
| Стадии проектирования программного обеспечения | 42 |  |
| Разработка модулей программного обеспечения | 48 |  |
| Тестирование программных модулей и их интеграции | 18 |  |
| Разработка программной документации и стандарты кодирования | 18 |  |

**Характеристика учебной/профессиональной деятельности обучающегося во время учебной практики (по профилю специальности):**

Общие и профессиональные компетенции, предусмотренные программой практики, освоены**/**не освоены.

(нужное подчеркнуть)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики Кузнецов К.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись)

Дата «18» марта 2023 г.

**СОДЕРЖАНИЕ**

[1 Проектирование системы 3](#_Toc129774639)

[1.1 Диаграмма вариантов использования 3](#_Toc129774640)

[1.2 Диаграмма активности 4](#_Toc129774641)

[1.3 Диаграмма последовательности 5](#_Toc129774642)

[1.4 ER-диаграмма 6](#_Toc129774643)

[1.5 Словарь данных 6](#_Toc129774644)

[2 Создание базы данных 8](#_Toc129774645)

[3 Разработка клиентского приложения 9](#_Toc129774646)

[4 Тестирование приложения 12](#_Toc129774647)

[4.1 Разработка тест-кейсов 12](#_Toc129774648)

[4.2 Модульное тестирование 17](#_Toc129774649)

[Приложение А 18](#_Toc129774650)

[Приложение Б 21](#_Toc129774651)

[Приложение В 112](#_Toc129774652)

## Проектирование системы

### Диаграмма вариантов использования

На диаграмме вариантов использования (рисунок 1) показаны акторы (повар, шеф-повар, администратор) и действия, которые они могут делать при помощи информационной системы.

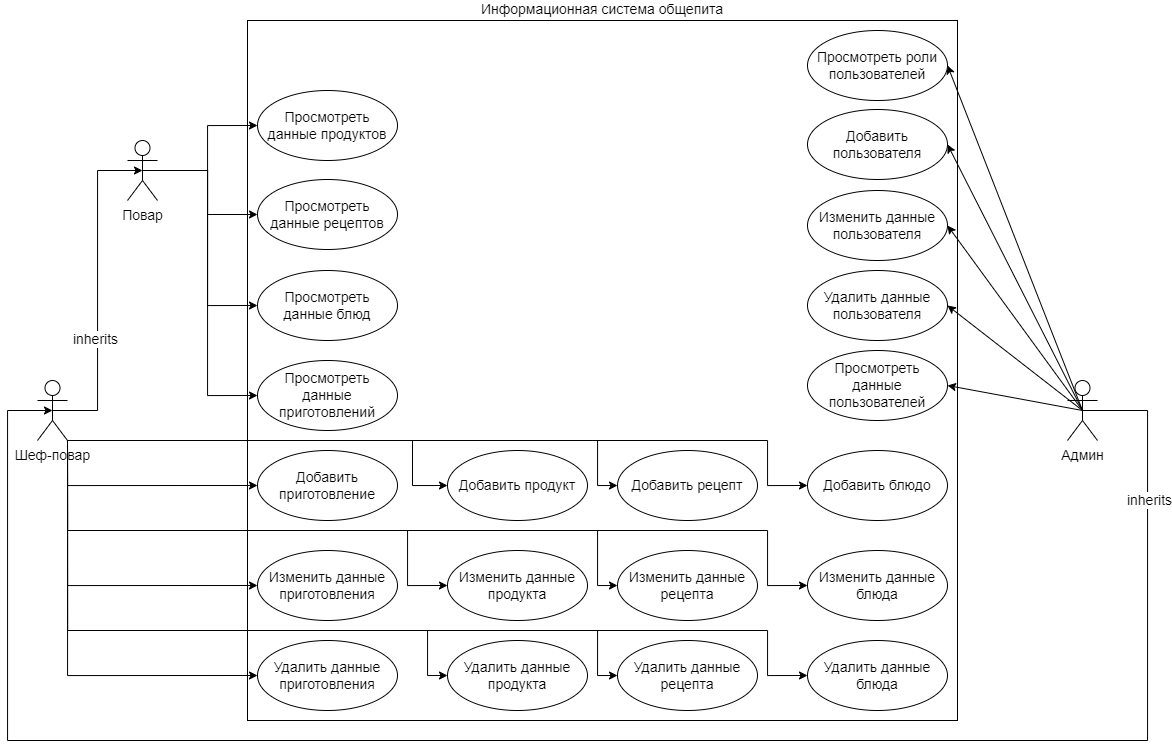


Рисунок 1 – Диаграмма вариантов использования

### Диаграмма активности

Диаграмма активности (рисунок 2) демонстрирует взаимодействие шеф-повара, приложения и базы данных во время создания приготовления блюда.

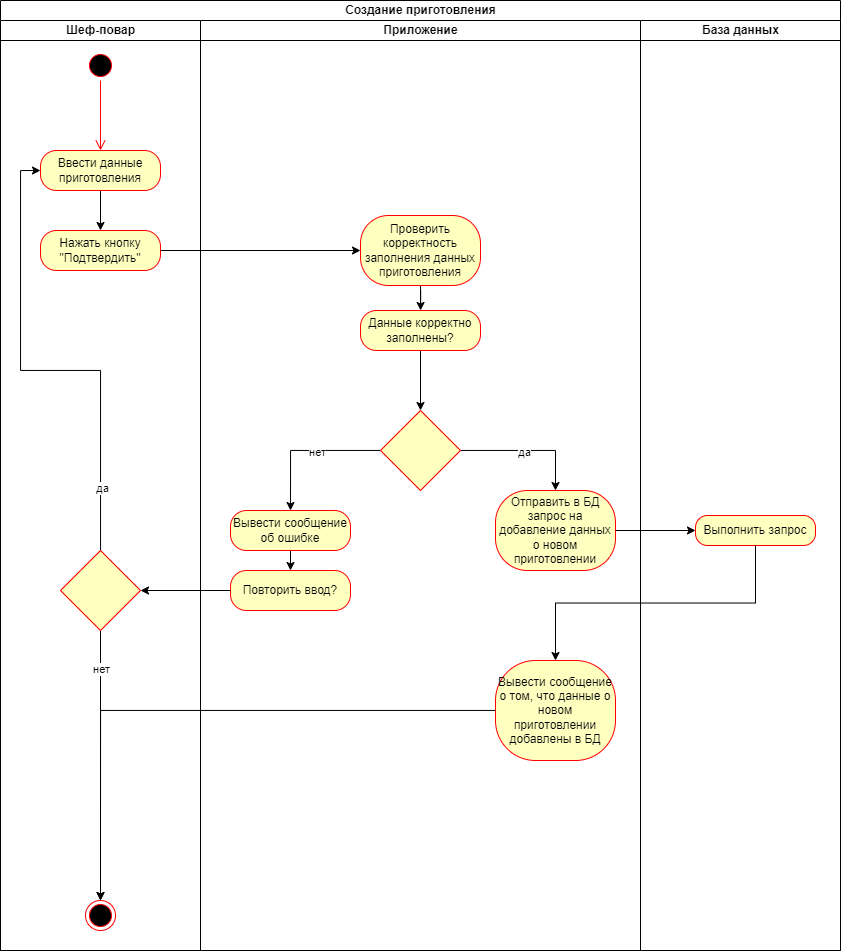


Рисунок 2 – Диаграмма активности

### Диаграмма последовательности

На рисунке 3 представлена диаграмма последовательности, показывающая процесс авторизации пользователя в системе.

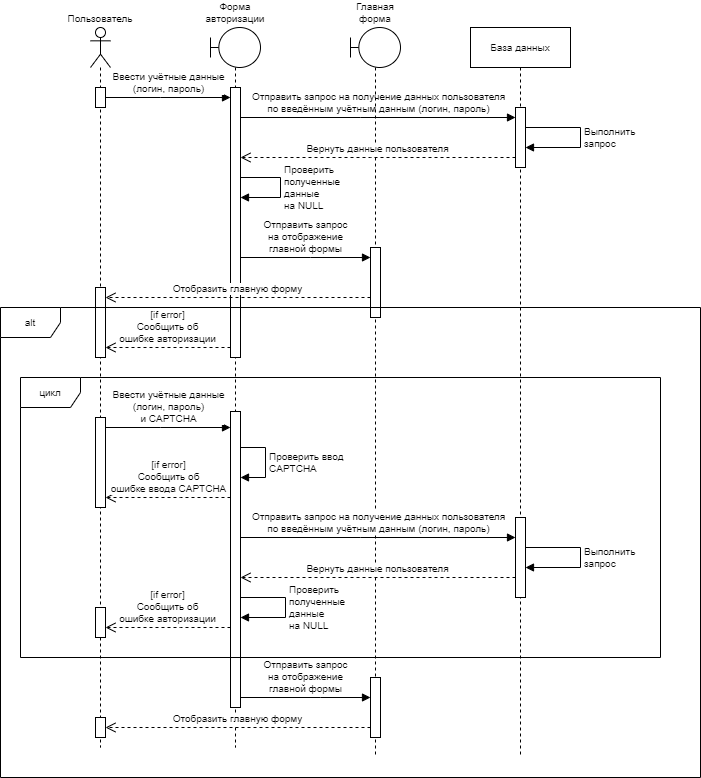


Рисунок 3 – Диаграмма последовательности

### ER-диаграмма

Схема базы данных представлена на рисунке 4 в виде ER-диаграммы.

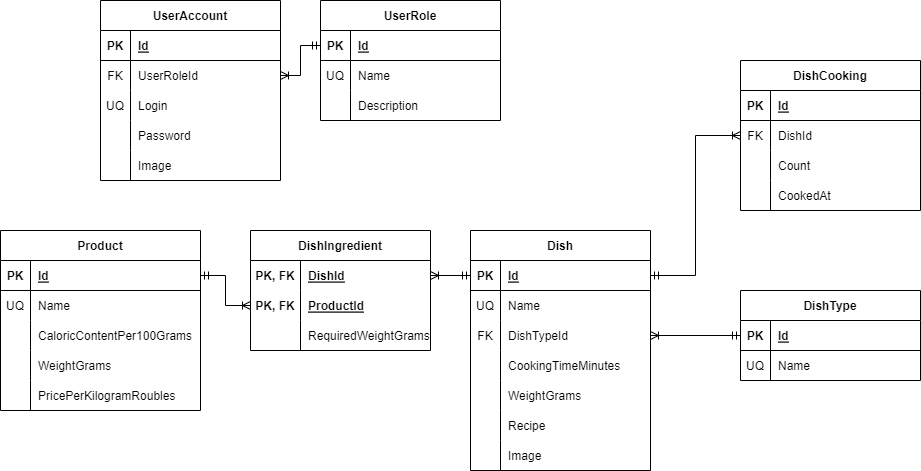


Рисунок 4 – ER-диаграмма

### Словарь данных

В таблице 1 приведён словарь данных для каждой таблицы из схемы базы данных.

Таблица 1 – Словарь данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Product** |  |  |  |  |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | Id | INT | Y | IDENTITY |
| UQ | Name | NVARCHAR(100) | Y | NOT NULL UNIQUE |
|  | CaloricContentPer100Grams | FLOAT | Y | NOT NULL CHECK (CaloricContentPer100Grams > 0) |
|  | WeightGrams | FLOAT | Y | NOT NULL CHECK (WeightGrams > 0) |
|  | PricePerKilogramRoubles | DECIMAL(18, 2) | Y | NOT NULL CHECK (PricePerKilogramRoubles > 0) |
|  |  |  |  |  |
| **DishType** |  |  |  |  |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | Id | INT | Y | IDENTITY |
| UQ | Name | NVARCHAR(100) | Y | NOT NULL UNIQUE |
|  |  |  |  |  |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dish** |  |  |  |  |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | Id | INT | Y | IDENTITY |
| UQ | Name | NVARCHAR(100) | Y | NOT NULL UNIQUE |
| FK | DishTypeId | INT | Y | NOT NULL |
|  | CookingTimeMinutes | FLOAT | Y | NOT NULL CHECK (CookingTimeMinutes > 0) |
|  | WeightGrams | FLOAT | Y | NOT NULL CHECK (CookedDishWeightGrams > 0) |
|  | Recipe | NVARCHAR(MAX) | Y | NOT NULL |
|  | Image | VARBINARY(MAX) | N | NULL |
|  |  |  |  |  |
| **DishIngredient** |  |  |  |  |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK, FK | DishId | INT | Y | NOT NULL |
| PK, FK | ProductId | INT | Y | NOT NULL |
|  | RequiredWeightGrams | FLOAT | Y | NOT NULL CHECK (RequiredWeightGrams > 0) |
|  |  |  |  |  |
| **DishCooking** |  |  |  |  |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | Id | INT | Y | IDENTITY |
| FK | DishId | INT | Y | NOT NULL |
|  | Count | INT | Y | NOT NULL CHECK (Count > 0) |
|  | CookedAt | DATETIME2(7) | Y | NOT NULL |
|  |  |  |  |  |
| **UserRole** |  |  |  |  |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | Id | INT | Y | NOT NULL |
| UQ | Name | NVARCHAR(100) | Y | NOT NULL UNIQUE |
|  |  |  |  |  |
| **UserAccount** |  |  |  |  |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | Id | INT | Y | IDENTITY |
| FK | UserRoleId | INT | Y | NOT NULL |
| UQ | Login | NVARCHAR(100) | Y | NOT NULL UNIQUE |
|  | Password | VARBINARY(MAX) | Y | NOT NULL |
|  | Image | VARBINARY(MAX) | N | NULL |

## Создание базы данных

База данных была создана при помощи Microsoft SQL Server 2022 и Microsoft SQL Server Management Studio 2018. Код создания таблиц, представлений, заполнения таблиц данными приведён в приложении А. Диаграмма базы данных, созданная и хранящаяся на сервере, приведена на рисунке 5.



Рисунок 5 – Диаграмма базы данных

## Разработка клиентского приложения

Исходный код клиентского приложения находится в приложении Б, а также в репозитории GitHub (ссылка: https://github.com/ProgramSlayer/FoodServiceDbCrud.git).

Снимки основных экранных форм приведены на рисунках 6-10.



Рисунок 6 – Форма авторизации

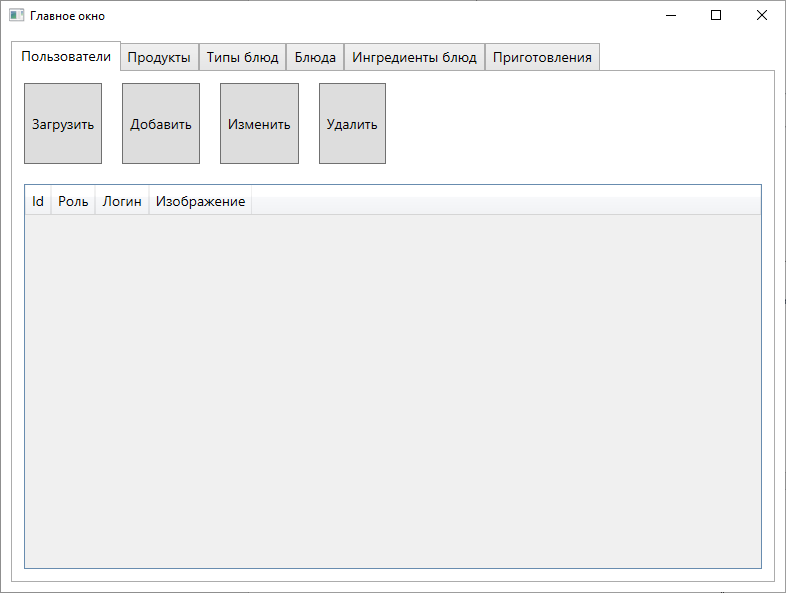


Рисунок 7 – Экранная форма администратора системы

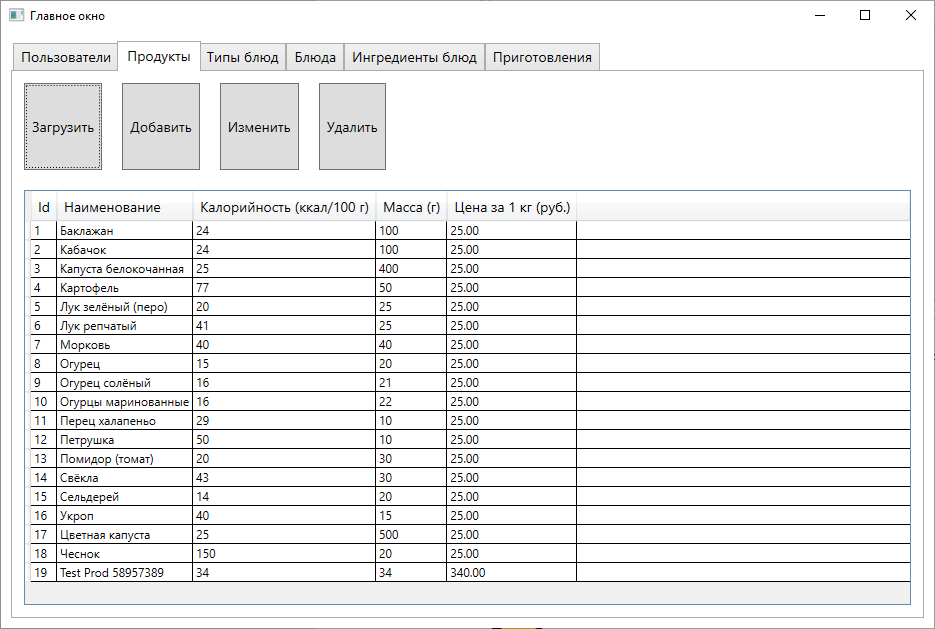


Рисунок 8 – Экранная форма таблицы продуктов

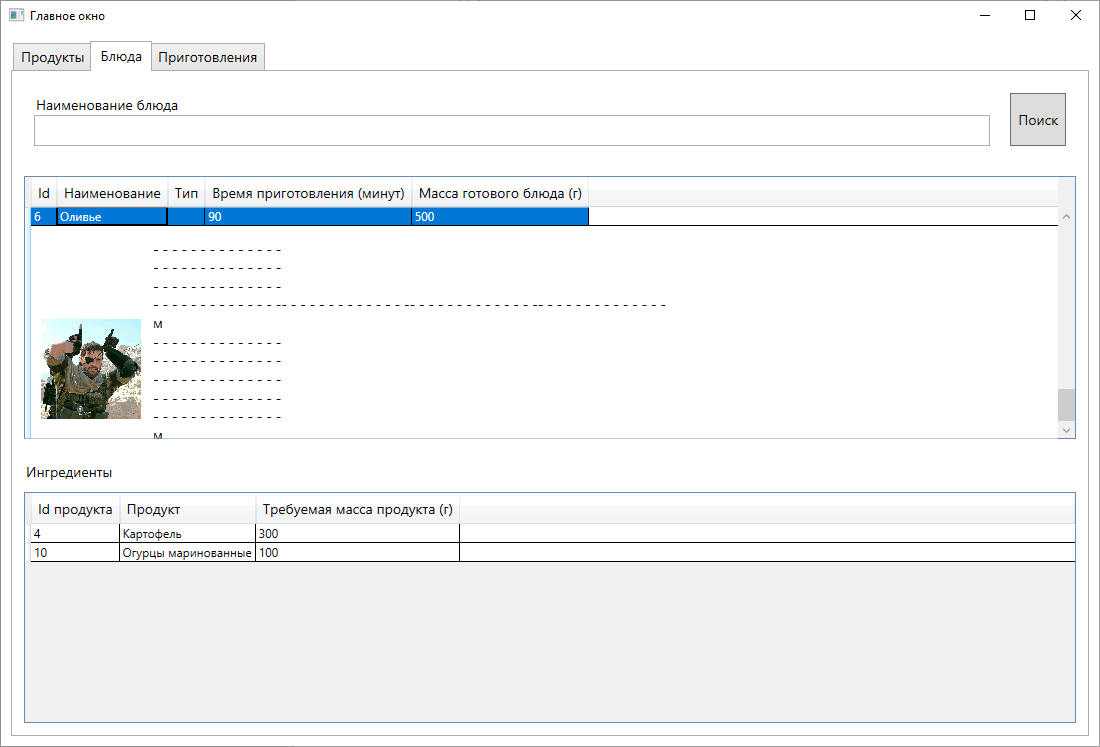


Рисунок 9 – Экранная форма повара

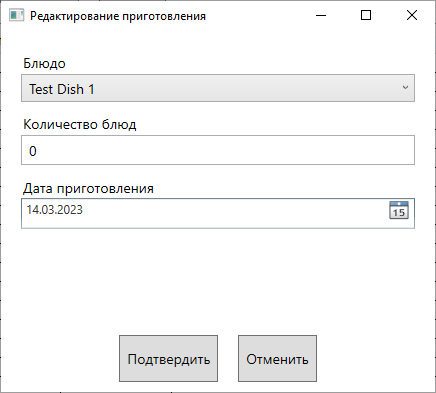


Рисунок 10 – Экранная форма редактирования приготовления

## Тестирование приложения

### Разработка тест-кейсов

Тест-кейсы описывают совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемых функций или их частей. Разработанные тест-кейсы приведены в таблицах 2-6.

Таблица 2 - Тест-кейс TC\_AUTH\_POSITIVE\_1

|  |  |
| --- | --- |
| **Название проекта** | FoodServiceDbCrud |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Andrey |
| **Дата(ы) теста** | 13.03.2022 |
| **Тестовый пример #** | TC\_AUTH\_POSITIVE\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Авторизация администратора системы с действительным логином и паролем |
| **Краткое изложение теста** | Приложение должно провести авторизацию пользователя и отобразить главное окно, соответствующее роли пользователя, после успешной авторизации. |
| **Этапы теста** | 1. ввести логин; 2. ввести пароль; 3. ввести CAPTCHA; 4. нажать кнопку «Вход». |
| **Тестовые данные** | Логин: GigaAdmin  Пароль: giga\_admin  CAPCHA: 1337 |
| **Ожидаемый результат** | 1. появится сообщение о прохождении авторизации; 2. загрузится экранная форма администратора. |
| **Фактический результат** | Соответствует ожиданиям. |
| **Статус** | Зачёт |
| **Предварительное условие** | Запустить приложение. |
| **Постусловие** | Коррекное состояние системы |
| **Примечания/комментарии** |  |

Таблица 3 - Тест-кейс TC\_AUTH\_NEGATIVE\_1

|  |  |
| --- | --- |
| **Название проекта** | FoodServiceDbCrud |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Andrey |
| **Дата(ы) теста** | 13.03.2022 |
| **Тестовый пример #** | TC\_AUTH\_NEGATIVE\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Авторизация администратора системы с недействительным логином и паролем |
| **Краткое изложение теста** | Приложение должно сообщить пользователю об ошибочности введённых учётных данных. |
| **Этапы теста** | 1) ввести логин;  2) ввести пароль;  3) ввести CAPCHA;  4) нажать кнопку «Вход». |
| **Тестовые данные** | Логин: SusClassified  Пароль: nocomprende  CAPTCHA: 1337. |
| **Ожидаемый результат** | Появится сообщение об ошибке авторизации. |
| **Фактический результат** | Соответствует ожиданиям. |
| **Статус** | Зачёт |
| **Предварительное условие** | Запустить приложение |
| **Постусловие** |  |
| **Примечания/комментарии** |  |

Таблица 4 - Тест-кейс TC\_AUTH\_POSITIVE\_2

|  |  |
| --- | --- |
| **Название проекта** | FoodServiceDbCrud |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Andrey |
| **Дата(ы) теста** | 13.03.2022 |
| **Тестовый пример #** | TC\_AUTH\_POSITIVE\_2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Авторизация шеф-повара с действительными логином и паролем |
| **Краткое изложение теста** | Приложение должно провести успешную авторизацию шеф-повара и отобразить экранную форму шеф-повара для работы с БД. |
| **Этапы теста** | 1. ввести логин; 2. ввести пароль; 3. ввести CAPTCHA; 4. нажать кнопку «Вход» |
| **Тестовые данные** | Логин: MasterChef  Пароль: master\_chef  CAPTCHA: 1337 |
| **Ожидаемый результат** | 1. появится сообщение о прохождении авторизации; 2. загрузится экранная форма шеф-повара. |
| **Фактический результат** | Соответствует ожиданиям. |
| **Статус** | Зачёт |
| **Предварительное условие** | Запустить приложение. |
| **Постусловие** |  |
| **Примечания/комментарии** |  |

Таблица 5 - Тест-кейс TC\_AUTH\_POSITIVE\_3

|  |  |
| --- | --- |
| **Название проекта** | FoodServiceDbCrud |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Andrey |
| **Дата(ы) теста** | 13.03.2022 |
| **Тестовый пример #** | TC\_AUTH\_POSITIVE\_3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Авторизация повара с действительными логином и паролем |
| **Краткое изложение теста** | Приложение должно провести успешную авторизацию повара и отобразить экранную форму повара для работы с БД. |
| **Этапы теста** | 1. ввести логин; 2. ввести пароль; 3. ввести CAPTCHA; 4. нажать кнопку «Вход» |
| **Тестовые данные** | Логин: Cook1  Пароль: cook\_1  CAPTCHA: 1337 |
| **Ожидаемый результат** | 1. появится сообщение о прохождении авторизации; 2. загрузится экранная форма повара. |
| **Фактический результат** | Соответствует ожиданиям. |
| **Статус** | Зачёт |
| **Предварительное условие** | Запустить приложение. |
| **Постусловие** |  |
| **Примечания/комментарии** |  |

Таблица 6 - Тест-кейс TC\_CreateDishCooking\_POSITIVE\_1

|  |  |
| --- | --- |
| **Название проекта** | FoodServiceDbCrud |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Andrey |
| **Дата(ы) теста** | 13.03.2022 |
| **Тестовый пример #** | TC\_CreateDishCooking\_POSITIVE\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Создание записи о приготовлении блюда |
| **Краткое изложение теста** | Данные нового приготовления должны быть добавлены в БД, а затем отображены в программе. |
| **Этапы теста** | 1. перейти на вкладку «Приготовления»; 2. нажать кнопку «Добавить»; 3. ввести данные приготовления; 4. нажать кнопку «Подтвердить». |
| **Тестовые данные** | Блюдо: Test Dish 1  Количество блюд: 101  Дата приготовления: 13.03.2022 |
| **Ожидаемый результат** | 1. приготовление будет добавлено в БД; 2. список приготовлений перезагрузится; 3. новое приготовление окажется внизу списка. |
| **Фактический результат** | Соответствует ожиданиям. |
| **Статус** | Зачёт |
| **Предварительное условие** | Пройти авторизацию. |
| **Постусловие** |  |
| **Примечания/комментарии** |  |

### Модульное тестирование

Исходные коды модульных тестов приведены в приложении В. Результаты прохождения модульных тестов показаны на рисунке 11.

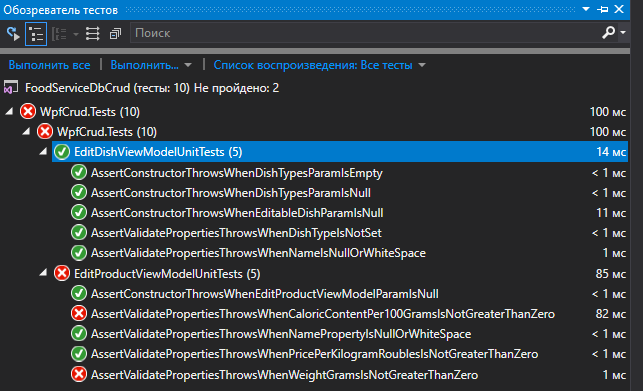


Рисунок 11 – Результаты прохождения модульного тестирования

## Приложение А

Код SQL

А.1 Запросы на создание таблиц и заполнение их данными

BEGIN TRAN

CREATE TABLE UserRole(

Id INT PRIMARY KEY,

[Name] NVARCHAR(100) NOT NULL UNIQUE

);

INSERT INTO UserRole (Id, [Name])

VALUES (1, 'Admin'),

(2, 'Chef'),

(3, 'Cook');

CREATE TABLE [UserAccount](

Id INT PRIMARY KEY IDENTITY,

UserRoleId INT NOT NULL FOREIGN KEY REFERENCES UserRole(Id) ON UPDATE NO ACTION ON DELETE NO ACTION,

[Login] NVARCHAR(100) NOT NULL UNIQUE,

[Password] VARBINARY(MAX) NOT NULL,

[Image] VARBINARY(MAX) NULL

);

INSERT INTO UserAccount (UserRoleId, [Login], [Password], [Image])

VALUES

(1, 'GigaAdmin', HASHBYTES('sha2\_512', 'giga\_admin'), NULL),

(2, 'MasterChef', HASHBYTES('sha2\_512', 'master\_chef'), NULL),

(3, 'Cook1', HASHBYTES('sha2\_512', 'cook\_1'), NULL);

CREATE TABLE Product(

Id INT PRIMARY KEY IDENTITY,

[Name] NVARCHAR(100) NOT NULL UNIQUE,

CaloricContentPer100Grams FLOAT NOT NULL CHECK (CaloricContentPer100Grams > 0),

WeightGrams FLOAT NOT NULL CHECK (WeightGrams > 0),

PricePerKilogramRoubles DECIMAL(18, 2) NOT NULL CHECK (PricePerKilogramRoubles > 0),

);

INSERT INTO Product ([Name], CaloricContentPer100Grams, WeightGrams, PricePerKilogramRoubles)

VALUES

('Баклажан', 24, 100, 25),

('Кабачок', 24, 100, 25),

('Капуста белокочанная', 25, 400, 25),

('Картофель', 77, 50, 25),

('Лук зелёный (перо)', 20, 25, 25),

('Лук репчатый', 41, 25, 25),

('Морковь', 40, 40, 25),

('Огурец', 15, 20, 25),

('Огурец солёный', 16, 21, 25),

('Огурцы маринованные', 16, 22, 25),

('Перец халапеньо', 29, 10, 25),

('Петрушка', 50, 10, 25),

('Помидор (томат)', 20, 30, 25),

('Свёкла', 43, 30, 25),

('Сельдерей', 14, 20, 25),

('Укроп', 40, 15, 25),

('Цветная капуста', 25, 500, 25),

('Чеснок', 150, 20, 25);

CREATE TABLE DishType(

Id INT PRIMARY KEY IDENTITY,

[Name] NVARCHAR(100) NOT NULL UNIQUE,

);

INSERT INTO DishType ([Name])

VALUES

('Выпечка'),

('Гарнир'),

('Горячее блюдо'),

('Горячая закуска'),

('Десерт'),

('Завтрак'),

('Консервация'),

('На углях'),

('Напиток'),

('Салат'),

('Соус'),

('Паста'),

('Заправка'),

('Суп'),

('Бульон'),

('Хлеб'),

('Холодная закуска'),

('Шашлык');

CREATE TABLE Dish(

Id INT PRIMARY KEY IDENTITY,

[Name] NVARCHAR(100) NOT NULL UNIQUE,

DishTypeId INT NOT NULL FOREIGN KEY REFERENCES DishType(Id) ON UPDATE NO ACTION ON DELETE NO ACTION,

CookingTimeMinutes FLOAT NOT NULL CHECK (CookingTimeMinutes > 0),

WeightGrams FLOAT NOT NULL CHECK (WeightGrams > 0),

Recipe NVARCHAR(MAX) NOT NULL,

[Image] VARBINARY(MAX) NULL

);

INSERT INTO Dish ([Name], DishTypeId, CookingTimeMinutes, WeightGrams, Recipe, [Image])

VALUES

('Test Dish 1', 1, 30, 100, 'Test Recipe 1', NULL),

('Test Dish 2', 2, 40, 200, 'Test Recipe 2', NULL),

('Test Dish 3', 3, 50, 300, 'Test Recipe 3', NULL),

('Test Dish 4', 4, 60, 400, 'Test Recipe 4', NULL),

('Test Dish 5', 5, 70, 500, 'Test Recipe 5', NULL);

CREATE TABLE DishIngredient(

DishId INT NOT NULL FOREIGN KEY REFERENCES Dish(Id) ON UPDATE NO ACTION ON DELETE NO ACTION,

ProductId INT NOT NULL FOREIGN KEY REFERENCES Product(Id) ON UPDATE NO ACTION ON DELETE NO ACTION,

RequiredWeightGrams FLOAT NOT NULL CHECK(RequiredWeightGrams > 0),

PRIMARY KEY(DishId, ProductId)

);

INSERT INTO DishIngredient (DishId, ProductId, RequiredWeightGrams)

SELECT d.Id, p.Id, 100 FROM Dish d, Product p;

CREATE TABLE DishCooking(

Id INT PRIMARY KEY IDENTITY,

DishId INT NOT NULL FOREIGN KEY REFERENCES Dish(Id) ON UPDATE NO ACTION ON DELETE NO ACTION,

[Count] INT NOT NULL CHECK ([Count] > 0),

CookedAt DATETIME2(7) NOT NULL

);

INSERT INTO dbo.DishCooking (DishId, [Count], CookedAt)

VALUES

(1, 100, '01-01-2023 7:15'),

(2, 99, '01-01-2023 7:20'),

(3, 98, '01-01-2023 7:25'),

(4, 97, '01-01-2023 7:30'),

(5, 96, '01-01-2023 7:35'),

(1, 200, '03-01-2023 7:40'),

(2, 199, '04-01-2023 7:45'),

(3, 50, '05-01-2023 7:50'),

(4, 45, '06-01-2023 8:15'),

(5, 69, '07-01-2023 9:15'),

(1, 330, '08-01-2023 10:15'),

(2, 45, '09-01-2023 11:15'),

(3, 78, '09-01-2023 12:15'),

(4, 44, '09-01-2023 7:56'),

(5, 69, '10-01-2023 8:58'),

(1, 222, '11-01-2023 9:59'),

(2, 228, '13-01-2023 10:11'),

(3, 14, '14-01-2023 8:01'),

(4, 78, '15-01-2023 9:00'),

(5, 45, '16-01-2023 11:15');

IF @@ERROR <> 0

ROLLBACK

COMMIT

А.2 Создание представления ViewDish

CREATE VIEW dbo.ViewDish

AS

SELECT

d.Id AS 'DishId',

d.[Name] AS 'DishName',

dt.Id AS 'DishTypeId',

dt.[Name] AS 'DishTypeName',

d.CookingTimeMinutes AS 'DishCookingTimeMinutes',

d.WeightGrams AS 'DishWeightGrams',

(

SELECT SUM(p.CaloricContentPer100Grams \* di.RequiredWeightGrams / 100) / d.WeightGrams \* 100

FROM dbo.Product p

INNER JOIN dbo.DishIngredient di

ON p.Id = di.ProductId

AND di.DishId = d.Id

) AS 'DishCaloricContentPer100Grams',

(

SELECT SUM(p.PricePerKilogramRoubles / 1000 \* di.RequiredWeightGrams)

FROM dbo.Product p

INNER JOIN dbo.DishIngredient di

ON p.Id = di.ProductId

AND di.DishId = d.Id

) AS 'DishPriceRoubles',

d.Recipe AS 'DishRecipe',

d.[Image] AS 'DishImage'

FROM dbo.Dish d

INNER JOIN dbo.DishType dt

ON d.DishTypeId = dt.Id

## Приложение Б

Исходный код приложения

Б.1 App.config

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<configSections>

<!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->

<section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />

</configSections>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.1" />

</startup>

<connectionStrings>

<add name="FoodServiceDbContext" connectionString="metadata=res://\*/DbModels.FoodServiceDbModel.csdl|res://\*/DbModels.FoodServiceDbModel.ssdl|res://\*/DbModels.FoodServiceDbModel.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=ADCLG1;initial catalog=db\_pisarets\_419/3\_foodservice;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />

<!--<add name="FoodServiceDbContext" connectionString="metadata=res://\*/DbModels.FoodServiceDbModel.csdl|res://\*/DbModels.FoodServiceDbModel.ssdl|res://\*/DbModels.FoodServiceDbModel.msl;provider=System.Data.SqlClient;provider connection string=&quot;Data Source=DESKTOP-NLLOE6S\SQLEXPRESS;Initial Catalog=FoodServiceDb;Integrated Security=True;Connect Timeout=30;Encrypt=False&quot;" providerName="System.Data.EntityClient" />-->

</connectionStrings>

<entityFramework>

<defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityFramework" />

<providers>

<provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />

</providers>

</entityFramework>

</configuration>

Б.2 CommandBase.cs

using System;

using System.Windows.Input;

namespace WpfCrud.Commands

{

/// <summary>

/// Абстрактная реализация интерфейса <see cref="ICommand"/>.

/// </summary>

public abstract class CommandBase : ICommand

{

public event EventHandler CanExecuteChanged;

public virtual bool CanExecute(object parameter)

{

return true;

}

public abstract void Execute(object parameter);

/// <summary>

/// Вызывает событие <see cref="CanExecuteChanged"/>.

/// </summary>

protected void OnCanExecuteChanged() => CanExecuteChanged?.Invoke(this, EventArgs.Empty);

}

}

Б.3 DelegateCommand.cs

using System;

namespace WpfCrud.Commands

{

/// <summary>

/// Команда, принимающая делегат.

/// </summary>

public class DelegateCommand : CommandBase

{

/// <summary>

/// Действие, выполняемое командой.

/// </summary>

private readonly Action<object> \_execute;

/// <summary>

/// Метод, определяющий доступность команды для выполнения.

/// </summary>

private readonly Predicate<object> \_canExecute;

/// <summary>

///

/// </summary>

/// <param name="execute">Метод, выполняемый командой.</param>

/// <param name="canExecute">Метод, определяющий доступность команды для выполнения.</param>

/// <exception cref="ArgumentNullException">Параметр <paramref name="execute"/> имеет значение <see cref="null"/></exception>

public DelegateCommand(Action<object> execute, Predicate<object> canExecute = null)

{

\_execute = execute ?? throw new ArgumentNullException(nameof(execute));

\_canExecute = canExecute;

}

public override bool CanExecute(object parameter)

{

return \_canExecute is null || \_canExecute(parameter);

}

public override void Execute(object parameter)

{

\_execute(parameter);

OnCanExecuteChanged();

}

}

}

Б.4 AsyncCommandBase.cs

using System;

using System.Threading.Tasks;

namespace WpfCrud.Commands

{

/// <summary>

/// Абстрактная реализация команды, способной выполняться асинхронно.

/// </summary>

public abstract class AsyncCommandBase : CommandBase

{

/// <summary>

/// Определяет, выполняется ли команда в данный момент.

/// </summary>

private bool \_isLoading;

/// <summary>

/// Метод, определяющий обработку исключений во время выполнения команды.

/// </summary>

private readonly Action<Exception> \_onException;

/// <summary>

///

/// </summary>

/// <param name="onException">Метод, определяющий обработку исключений во время выполнения команды.</param>

protected AsyncCommandBase(Action<Exception> onException = null)

{

\_onException = onException;

}

/// <summary>

/// Определяет, выполняется ли команда в данный момент.

/// </summary>

public bool IsLoading

{

get => \_isLoading;

set

{

\_isLoading = value;

OnCanExecuteChanged();

}

}

public override bool CanExecute(object parameter)

{

return !IsLoading;

}

public override async void Execute(object parameter)

{

IsLoading = true;

try

{

await ExecuteAsync(parameter);

}

catch (Exception e)

{

\_onException?.Invoke(e);

}

IsLoading = false;

}

protected abstract Task ExecuteAsync(object parameter);

}

}

Б.5 AsyncDelegateCommand.cs

using System;

using System.Threading.Tasks;

namespace WpfCrud.Commands

{

/// <summary>

/// Асинхронная команда, принимающая делегат.

/// </summary>

public class AsyncDelegateCommand : AsyncCommandBase

{

private readonly Func<Task> \_callback;

/// <summary>

///

/// </summary>

/// <param name="callback">Метод, выполняемый командой.</param>

/// <param name="onException">Метод, вызываемый при исключении во время выполнения команды.</param>

/// <exception cref="ArgumentNullException"></exception>

public AsyncDelegateCommand(Func<Task> callback, Action<Exception> onException = null) : base(onException)

{

\_callback = callback ?? throw new ArgumentNullException(nameof(callback));

}

protected override async Task ExecuteAsync(object parameter)

{

await \_callback();

}

}

}

Б.6 NullImageConverter.cs

using System;

using System.Globalization;

using System.Windows;

using System.Windows.Data;

namespace WpfCrud.Converters

{

/// <summary>

/// Конвертер для преобразования <see cref="null"/> в источник изображения (???).

/// </summary>

public class NullImageConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

return value ?? DependencyProperty.UnsetValue;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

return Binding.DoNothing;

}

}

}

Б.7 FoodServiceDbModel.Context.cs

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан по шаблону.

//

// Изменения, вносимые в этот файл вручную, могут привести к непредвиденной работе приложения.

// Изменения, вносимые в этот файл вручную, будут перезаписаны при повторном создании кода.

// </auto-generated>

//------------------------------------------------------------------------------

namespace WpfCrud.DbModels

{

using System;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

public partial class FoodServiceDbContext : DbContext

{

public FoodServiceDbContext()

: base("name=FoodServiceDbContext")

{

}

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

throw new UnintentionalCodeFirstException();

}

public virtual DbSet<Dish> Dishes { get; set; }

public virtual DbSet<DishCooking> DishCookings { get; set; }

public virtual DbSet<DishIngredient> DishIngredients { get; set; }

public virtual DbSet<DishType> DishTypes { get; set; }

public virtual DbSet<Product> Products { get; set; }

public virtual DbSet<UserAccount> UserAccounts { get; set; }

public virtual DbSet<UserRole> UserRoles { get; set; }

public virtual DbSet<ViewDish> ViewDishes { get; set; }

}

}

Б.8 WpfCrud.DbModels.Dish.cs

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан по шаблону.

//

// Изменения, вносимые в этот файл вручную, могут привести к непредвиденной работе приложения.

// Изменения, вносимые в этот файл вручную, будут перезаписаны при повторном создании кода.

// </auto-generated>

//------------------------------------------------------------------------------

namespace WpfCrud.DbModels

{

using System;

using System.Collections.Generic;

public partial class Dish

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Dish()

{

this.DishCookings = new HashSet<DishCooking>();

this.DishIngredients = new HashSet<DishIngredient>();

}

public int Id { get; set; }

public string Name { get; set; }

public int DishTypeId { get; set; }

public double CookingTimeMinutes { get; set; }

public double WeightGrams { get; set; }

public string Recipe { get; set; }

public byte[] Image { get; set; }

public virtual DishType DishType { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<DishCooking> DishCookings { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<DishIngredient> DishIngredients { get; set; }

}

}

Б.9 WpfCrud.DbModels.DishCooking.cs

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан по шаблону.

//

// Изменения, вносимые в этот файл вручную, могут привести к непредвиденной работе приложения.

// Изменения, вносимые в этот файл вручную, будут перезаписаны при повторном создании кода.

// </auto-generated>

//------------------------------------------------------------------------------

namespace WpfCrud.DbModels

{

using System;

using System.Collections.Generic;

public partial class DishCooking

{

public int Id { get; set; }

public int DishId { get; set; }

public int Count { get; set; }

public System.DateTime CookedAt { get; set; }

public virtual Dish Dish { get; set; }

}

}

Б.10 WpfCrud.DbModels.DishIngredient.cs

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан по шаблону.

//

// Изменения, вносимые в этот файл вручную, могут привести к непредвиденной работе приложения.

// Изменения, вносимые в этот файл вручную, будут перезаписаны при повторном создании кода.

// </auto-generated>

//------------------------------------------------------------------------------

namespace WpfCrud.DbModels

{

using System;

using System.Collections.Generic;

public partial class DishIngredient

{

public int DishId { get; set; }

public int ProductId { get; set; }

public double RequiredWeightGrams { get; set; }

public virtual Dish Dish { get; set; }

public virtual Product Product { get; set; }

}

}

Б.11 WpfCrud.DbModels.DishType.cs

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан по шаблону.

//

// Изменения, вносимые в этот файл вручную, могут привести к непредвиденной работе приложения.

// Изменения, вносимые в этот файл вручную, будут перезаписаны при повторном создании кода.

// </auto-generated>

//------------------------------------------------------------------------------

namespace WpfCrud.DbModels

{

using System;

using System.Collections.Generic;

public partial class DishType

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public DishType()

{

this.Dishes = new HashSet<Dish>();

}

public int Id { get; set; }

public string Name { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Dish> Dishes { get; set; }

}

}

Б.12 WpfCrud.DbModels.Product.cs

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан по шаблону.

//

// Изменения, вносимые в этот файл вручную, могут привести к непредвиденной работе приложения.

// Изменения, вносимые в этот файл вручную, будут перезаписаны при повторном создании кода.

// </auto-generated>

//------------------------------------------------------------------------------

namespace WpfCrud.DbModels

{

using System;

using System.Collections.Generic;

public partial class Product

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Product()

{

this.DishIngredients = new HashSet<DishIngredient>();

}

public int Id { get; set; }

public string Name { get; set; }

public double CaloricContentPer100Grams { get; set; }

public double WeightGrams { get; set; }

public decimal PricePerKilogramRoubles { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<DishIngredient> DishIngredients { get; set; }

}

}

Б.13 WpfCrud.DbModels.UserAccount.cs

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан по шаблону.

//

// Изменения, вносимые в этот файл вручную, могут привести к непредвиденной работе приложения.

// Изменения, вносимые в этот файл вручную, будут перезаписаны при повторном создании кода.

// </auto-generated>

//------------------------------------------------------------------------------

namespace WpfCrud.DbModels

{

using System;

using System.Collections.Generic;

public partial class UserAccount

{

public int Id { get; set; }

public int UserRoleId { get; set; }

public string Login { get; set; }

public byte[] Password { get; set; }

public byte[] Image { get; set; }

public virtual UserRole UserRole { get; set; }

}

}

Б.14 WpfCrud.DbModels.UserRole.cs

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан по шаблону.

//

// Изменения, вносимые в этот файл вручную, могут привести к непредвиденной работе приложения.

// Изменения, вносимые в этот файл вручную, будут перезаписаны при повторном создании кода.

// </auto-generated>

//------------------------------------------------------------------------------

namespace WpfCrud.DbModels

{

using System;

using System.Collections.Generic;

public partial class UserRole

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public UserRole()

{

this.UserAccounts = new HashSet<UserAccount>();

}

public int Id { get; set; }

public string Name { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<UserAccount> UserAccounts { get; set; }

}

}

Б.15 WpfCrud.DbModels.ViewDish.cs

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан по шаблону.

//

// Изменения, вносимые в этот файл вручную, могут привести к непредвиденной работе приложения.

// Изменения, вносимые в этот файл вручную, будут перезаписаны при повторном создании кода.

// </auto-generated>

//------------------------------------------------------------------------------

namespace WpfCrud.DbModels

{

using System;

using System.Collections.Generic;

public partial class ViewDish

{

public int DishId { get; set; }

public string DishName { get; set; }

public int DishTypeId { get; set; }

public string DishTypeName { get; set; }

public double DishCookingTimeMinutes { get; set; }

public double DishWeightGrams { get; set; }

public Nullable<double> DishCaloricContentPer100Grams { get; set; }

public Nullable<double> DishPriceRoubles { get; set; }

public string DishRecipe { get; set; }

public byte[] DishImage { get; set; }

}

}

Б.16 UserRoleEnum.cs

namespace WpfCrud.Models.Enums

{

/// <summary>

/// Роли пользователей в системе.

/// </summary>

public enum UserRoleEnum

{

/// <summary>

/// Неавторизованный пользователь.

/// </summary>

Unauthorized = 0,

/// <summary>

/// Администратор.

/// </summary>

Admin = 1,

/// <summary>

/// Шеф-повар.

/// </summary>

Chef = 2,

/// <summary>

/// Повар.

/// </summary>

Cook = 3,

}

}

Б.17 CurrentUser.cs

using WpfCrud.Models.Enums;

namespace WpfCrud.Models

{

/// <summary>

/// Информация о текущем пользователе системы.

/// </summary>

public class CurrentUser

{

/// <summary>

/// Идентификатор учётной записи.

/// </summary>

public int Id { get; set; }

/// <summary>

/// Имя (логин) учётной записи.

/// </summary>

public string Login { get; set; }

/// <summary>

/// Роль пользователя в системе.

/// </summary>

public UserRoleEnum Role { get; set; }

}

}

Б.18 DishType.cs

namespace WpfCrud.Models

{

/// <summary>

/// Тип блюда.

/// </summary>

public class DishType

{

public DishType(int id, string name)

{

Id = id;

Name = name;

}

public DishType()

{

}

/// <summary>

/// Идентификатор.

/// </summary>

public int Id { get; set; }

/// <summary>

/// Название.

/// </summary>

public string Name { get; set; }

}

}

Б.19 DishWithIngredients.cs

using System.Collections.Generic;

using System.Linq;

namespace WpfCrud.Models

{

/// <summary>

/// Блюдо с его ингредиентами.

/// </summary>

public class DishWithIngredients

{

public DishWithIngredients(int id, string name, DishType dishType, double cookingTimeMinutes, double weightGrams, string recipe, byte[] image, IEnumerable<ViewDishIngredient> ingredients)

{

Id = id;

Name = name;

DishType = dishType;

CookingTimeMinutes = cookingTimeMinutes;

WeightGrams = weightGrams;

Recipe = recipe;

Image = image;

Ingredients = ingredients ?? Enumerable.Empty<ViewDishIngredient>();

}

/// <summary>

/// Идентификатор.

/// </summary>

public int Id { get; }

/// <summary>

/// Название.

/// </summary>

public string Name { get; }

/// <summary>

/// Тип.

/// </summary>

public DishType DishType { get; }

/// <summary>

/// Время готовки (минут).

/// </summary>

public double CookingTimeMinutes { get; }

/// <summary>

/// Масса готового блюда (г).

/// </summary>

public double WeightGrams { get; }

/// <summary>

/// Рецепт.

/// </summary>

public string Recipe { get; }

/// <summary>

/// Изображение.

/// </summary>

public byte[] Image { get; }

/// <summary>

/// Ингредиенты.

/// </summary>

public IEnumerable<ViewDishIngredient> Ingredients { get; }

}

}

Б.20 EditableDish.cs

namespace WpfCrud.Models

{

public class EditableDish

{

public EditableDish()

{

}

public EditableDish(int id, string name, int dishTypeId, double cookingTimeMinutes, double weightGrams, string recipe, byte[] image)

{

Id = id;

Name = name;

DishTypeId = dishTypeId;

CookingTimeMinutes = cookingTimeMinutes;

WeightGrams = weightGrams;

Recipe = recipe;

Image = image;

}

public int Id { get; set; }

public string Name { get; set; }

public int DishTypeId { get; set; }

public double CookingTimeMinutes { get; set; }

public double WeightGrams { get; set; }

public string Recipe { get; set; }

public byte[] Image { get; set; }

}

}

Б.21 EditableDishCooking.cs

using System;

namespace WpfCrud.Models

{

public class EditableDishCooking

{

public EditableDishCooking()

{

}

public EditableDishCooking(int id, int dishId, int count, DateTime cookedAt)

{

Id = id;

DishId = dishId;

Count = count;

CookedAt = cookedAt;

}

public int Id { get; set; }

public int DishId { get; set; }

public int Count { get; set; }

public DateTime CookedAt { get; set; } = DateTime.Now;

}

}

Б.22 EditableDishIngredient.cs

namespace WpfCrud.Models

{

public class EditableDishIngredient

{

public EditableDishIngredient(int dishId, int productId, double requiredWeightGrams)

{

DishId = dishId;

ProductId = productId;

RequiredWeightGrams = requiredWeightGrams;

}

public EditableDishIngredient()

{

}

public int DishId { get; set; }

public int ProductId { get; set; }

public double RequiredWeightGrams { get; set; }

}

}

Б.23 EditableProduct.cs

namespace WpfCrud.Models

{

public class EditableProduct

{

public EditableProduct(int id, string name, double caloricContentPer100Grams, double weightGrams, decimal pricePerKilogramRoubles)

{

Id = id;

Name = name;

CaloricContentPer100Grams = caloricContentPer100Grams;

WeightGrams = weightGrams;

PricePerKilogramRoubles = pricePerKilogramRoubles;

}

public EditableProduct()

{

}

public int Id { get; set; }

public string Name { get; set; }

public double CaloricContentPer100Grams { get; set; }

public double WeightGrams { get; set; }

public decimal PricePerKilogramRoubles { get; set; }

}

}

Б.24 EditableUserAccount.cs

using WpfCrud.Models.Enums;

namespace WpfCrud.Models

{

public class EditableUserAccount

{

public EditableUserAccount()

{

}

public EditableUserAccount(int id, string login, string password, UserRoleEnum role, byte[] image)

{

Id = id;

Login = login;

Password = password;

Role = role;

Image = image;

}

public int Id { get; set; }

public string Login { get; set; }

public string Password { get; set; }

public UserRoleEnum Role { get; set; } = UserRoleEnum.Cook;

public byte[] Image { get; set; }

}

}

Б.25 ViewDish.cs

namespace WpfCrud.Models

{

/// <summary>

/// Информация о блюде (без ингредиентов).

/// </summary>

public class ViewDish

{

public ViewDish(

int id,

string name,

int dishTypeId,

string dishTypeName,

double cookingTimeMinutes,

double weightGrams,

string recipe,

byte[] image,

double? caloricContentPer100Grams,

double? dishPriceRoubles)

{

Id = id;

Name = name;

DishTypeId = dishTypeId;

DishTypeName = dishTypeName;

CookingTimeMinutes = cookingTimeMinutes;

WeightGrams = weightGrams;

Recipe = recipe;

Image = image;

CaloricContentPer100Grams = caloricContentPer100Grams;

DishPriceRoubles = dishPriceRoubles;

}

/// <summary>

/// Идентификатор.

/// </summary>

public int Id { get; }

/// <summary>

/// Название.

/// </summary>

public string Name { get; }

/// <summary>

/// Идентификатор типа блюда.

/// </summary>

public int DishTypeId { get; }

/// <summary>

/// Название типа блюда.

/// </summary>

public string DishTypeName { get; }

/// <summary>

/// Время готовки (минут).

/// </summary>

public double CookingTimeMinutes { get; }

/// <summary>

/// Масса готового блюда (г).

/// </summary>

public double WeightGrams { get; }

/// <summary>

/// Рецепт.

/// </summary>

public string Recipe { get; }

/// <summary>

/// Изображение.

/// </summary>

public byte[] Image { get; }

/// <summary>

/// Калорийность (ккал / 100 г) (на основе калорийности ингредиентов).

/// </summary>

public double? CaloricContentPer100Grams { get; }

/// <summary>

/// Стоимость 1 ед. блюда (руб.).

/// </summary>

public double? DishPriceRoubles { get; }

}

}

Б.26 ViewDishCooking.cs

using System;

namespace WpfCrud.Models

{

public class ViewDishCooking

{

public ViewDishCooking(int id, int dishId, string dishName, string dishTypeName, int count, DateTime cookedAt)

{

Id = id;

DishId = dishId;

DishName = dishName;

DishTypeName = dishTypeName;

Count = count;

CookedAt = cookedAt;

}

public int Id { get; }

public int DishId { get; }

public string DishName { get; }

public string DishTypeName { get; }

public int Count { get; }

public DateTime CookedAt { get; }

}

}

Б.27 ViewDishIngredient.cs

namespace WpfCrud.Models

{

public class ViewDishIngredient

{

public ViewDishIngredient(int dishId, string dishName, string dishTypeName, int productId, string productName, double requiredWeightGrams)

{

DishId = dishId;

DishName = dishName;

DishTypeName = dishTypeName;

ProductId = productId;

ProductName = productName;

RequiredWeightGrams = requiredWeightGrams;

}

public int DishId { get; }

public string DishName { get; }

public string DishTypeName { get; }

public int ProductId { get; }

public string ProductName { get; }

public double RequiredWeightGrams { get; }

}

}

Б.28 ViewProduct.cs

namespace WpfCrud.Models

{

public class ViewProduct

{

public ViewProduct(int id, string name, double caloricContentPer100Grams, double weightGrams, decimal pricePerKilogramRoubles)

{

Id = id;

Name = name;

CaloricContentPer100Grams = caloricContentPer100Grams;

WeightGrams = weightGrams;

PricePerKilogramRoubles = pricePerKilogramRoubles;

}

public int Id { get; }

public string Name { get; }

public double CaloricContentPer100Grams { get; }

public double WeightGrams { get; }

public decimal PricePerKilogramRoubles { get; }

}

}

Б.29 ViewUserAccount.cs

using WpfCrud.Models.Enums;

namespace WpfCrud.Models

{

public class ViewUserAccount

{

public ViewUserAccount(int id, int roleId, string roleName, string login, byte[] image)

{

Id = id;

RoleId = roleId;

RoleName = roleName;

Login = login;

Image = image;

}

public int Id { get; }

public int RoleId { get; }

public UserRoleEnum Role => (UserRoleEnum)RoleId;

public string RoleName { get; }

public string Login { get; }

public byte[] Image { get; }

}

}

Б.30 DishCookingService.cs

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Threading.Tasks;

using WpfCrud.DbModels;

using WpfCrud.Models;

namespace WpfCrud.Services.DishCookings

{

/// <summary>

/// Сервис для работы с приготовлениями блюд.

/// </summary>

public class DishCookingService

{

/// <summary>

/// Фабрика по производству контекста.

/// </summary>

private readonly Func<FoodServiceDbContext> \_factory = () => new FoodServiceDbContext();

public DishCookingService()

{

}

/// <summary>

/// Извлекает все приготовления блюд из БД и возвращает их.

/// </summary>

/// <returns>Список приготовлений блюд.</returns>

public async Task<IEnumerable<ViewDishCooking>> GetViewDishCookingsAsync()

{

using (var context = \_factory())

{

var dbResults = await (

from dc in context.DishCookings

join d in context.Dishes on dc.DishId equals d.Id

join dt in context.DishTypes on d.DishTypeId equals dt.Id

select new

{

DishCookingId = dc.Id,

DishId = d.Id,

DishName = d.Name,

DishTypeName = dt.Name,

dc.Count,

dc.CookedAt,

}).ToListAsync();

var results = from i in dbResults

select new ViewDishCooking(

i.DishCookingId,

i.DishId,

i.DishName,

i.DishTypeName,

i.Count,

i.CookedAt);

return results;

}

}

/// <summary>

/// Возвращает все приготовления указанного блюда.

/// </summary>

/// <param name="dishName">Наименование блюда.</param>

/// <returns>Список приготовлений указанного блюда.</returns>

public async Task<IEnumerable<ViewDishCooking>> GetViewDishCookingsByDishNameAsync(string dishName)

{

if (string.IsNullOrWhiteSpace(dishName))

{

var results = await GetViewDishCookingsAsync();

return results;

}

using (var context = \_factory())

{

var dbResults = await (

from dc in context.DishCookings

join d in context.Dishes on dc.DishId equals d.Id

join dt in context.DishTypes on d.DishTypeId equals dt.Id

where d.Name.Contains(dishName)

select new

{

DishCookingId = dc.Id,

DishId = d.Id,

DishName = d.Name,

DishTypeName = dt.Name,

dc.Count,

dc.CookedAt,

}).ToListAsync();

var results = from i in dbResults

select new ViewDishCooking(

i.DishCookingId,

i.DishId,

i.DishName,

i.DishTypeName,

i.Count,

i.CookedAt);

return results;

}

}

/// <summary>

/// Добавляет приготовление в БД.

/// </summary>

/// <param name="dc">Приготовление.</param>

/// <returns></returns>

/// <exception cref="ArgumentNullException"></exception>

public async Task<EditableDishCooking> AddDishCookingAsync(EditableDishCooking dc)

{

if (dc is null)

{

throw new ArgumentNullException(nameof(dc));

}

using (var context = \_factory())

{

var dbdc = new DishCooking

{

DishId = dc.DishId,

CookedAt = dc.CookedAt,

Count = dc.Count,

};

context.DishCookings.Add(dbdc);

await context.SaveChangesAsync();

dc.Id = dbdc.Id;

return dc;

}

}

/// <summary>

/// Обновляет указанное приготовление в БД.

/// </summary>

/// <param name="dc">Приготовление.</param>

/// <returns></returns>

/// <exception cref="ArgumentNullException"></exception>

/// <exception cref="Exception"></exception>

public async Task UpdateDishCookingAsync(EditableDishCooking dc)

{

if (dc is null)

{

throw new ArgumentNullException(nameof(dc));

}

using (var context = \_factory())

{

var dbdc = await context

.DishCookings

.SingleOrDefaultAsync(c => c.Id == dc.Id)

?? throw new Exception($"Приготовление (Id = {dc.Id}) не найдено в базе данных!");

dbdc.DishId = dc.DishId;

dbdc.Count = dc.Count;

dbdc.CookedAt = dc.CookedAt;

await context.SaveChangesAsync();

}

}

/// <summary>

/// Удаляет указанное приготовление из БД.

/// </summary>

/// <param name="id">Идентификатор приготовления.</param>

/// <returns></returns>

/// <exception cref="Exception"></exception>

public async Task DeleteDishCookingAsync(int id)

{

using (var context = \_factory())

{

var dbdc = await context

.DishCookings

.SingleOrDefaultAsync(c => c.Id == id)

?? throw new Exception($"Приготовление (Id = {id}) не найдено в базе данных!");

context.DishCookings.Remove(dbdc);

await context.SaveChangesAsync();

}

}

}

}

Б.31 DishService.cs

using System;

using ViewDish = WpfCrud.Models.ViewDish;

using DbDish = WpfCrud.DbModels.Dish;

using WpfCrud.DbModels;

using System.Threading.Tasks;

using System.Collections.Generic;

using System.Linq;

using System.Data.Entity;

using WpfCrud.Models;

namespace WpfCrud.Services.Dishes

{

/// <summary>

/// Сервис для работы с данными блюд в БД.

/// </summary>

public class DishService

{

private readonly Func<FoodServiceDbContext> \_factory = () => new FoodServiceDbContext();

public DishService()

{

}

/// <summary>

/// Возвращает список данных блюд из БД.

/// </summary>

/// <returns>Список данных блюд.</returns>

public async Task<IEnumerable<ViewDish>> GetDishInfosAsync()

{

using (var context = \_factory())

{

var dbViewDishes = await context.ViewDishes.ToListAsync();

var viewDishes = from i in dbViewDishes

select new ViewDish(

i.DishId,

i.DishName,

i.DishTypeId,

i.DishTypeName,

i.DishCookingTimeMinutes,

i.DishWeightGrams,

i.DishRecipe,

i.DishImage,

i.DishCaloricContentPer100Grams,

i.DishPriceRoubles);

return viewDishes;

}

}

/// <summary>

/// Возвращает список данных блюд вместе с их ингредиентами.

/// </summary>

/// <returns></returns>

public async Task<IEnumerable<DishWithIngredients>> GetDishesWithIngredientsAsync()

{

using (var context = \_factory())

{

var dbDishesWithIngredients = await context

.Dishes

.Include(d => d.DishType)

.Include(d => d.DishIngredients.Select(di => di.Product))

.ToListAsync();

List<DishWithIngredients> results = ToDishesWithIngredients(dbDishesWithIngredients);

return results;

}

}

/// <summary>

/// Возвращает список блюд и их ингредиентов по названию блюда.

/// </summary>

/// <param name="dishName">Название блюда.</param>

/// <returns></returns>

public async Task<IEnumerable<DishWithIngredients>> GetDishesWithIngredientsByDishNameAsync(string dishName)

{

if (string.IsNullOrWhiteSpace(dishName))

{

var result = await GetDishesWithIngredientsAsync();

return result;

}

using (var context = \_factory())

{

var dbDishesWithIngredients = await context

.Dishes

.Where(d => d.Name.Contains(dishName))

.Include(d => d.DishType)

.Include(d => d.DishIngredients.Select(di => di.Product))

.ToListAsync();

List<DishWithIngredients> results = ToDishesWithIngredients(dbDishesWithIngredients);

return results;

}

}

/// <summary>

/// Преобразует коллекцию <see cref="DbDish"/> в коллекцию <see cref="DishWithIngredients"/>.

/// </summary>

/// <param name="dbDishesWithIngredients"></param>

/// <returns></returns>

private static List<DishWithIngredients> ToDishesWithIngredients(IEnumerable<DbDish> dbDishesWithIngredients)

{

var results = new List<DishWithIngredients>();

foreach (var dbDish in dbDishesWithIngredients)

{

var ingredients = from di in dbDish.DishIngredients

select new ViewDishIngredient(

di.DishId,

dbDish.Name,

dbDish.DishType.Name,

di.Product.Id,

di.Product.Name,

di.RequiredWeightGrams);

var dishWithIngredients = new DishWithIngredients(

dbDish.Id,

dbDish.Name,

new Models.DishType(dbDish.DishTypeId, dbDish.DishType.Name),

dbDish.CookingTimeMinutes,

dbDish.WeightGrams,

dbDish.Recipe,

dbDish.Image,

ingredients);

results.Add(dishWithIngredients);

}

return results;

}

/// <summary>

/// Добавляет данные блюда в БД.

/// </summary>

/// <param name="dish">Блюдо.</param>

/// <returns></returns>

/// <exception cref="ArgumentNullException"></exception>

/// <exception cref="Exception"></exception>

public async Task<EditableDish> AddDishAsync(EditableDish dish)

{

if (dish is null)

{

throw new ArgumentNullException(nameof(dish));

}

using (var context = \_factory())

{

var alreadyInBase = await context.Dishes.AnyAsync(d => d.Id == dish.Id || d.Name == dish.Name);

if (alreadyInBase)

{

throw new Exception($"Блюдо (Name = {dish.Name}) уже есть в базе данных!");

}

var dbDish = new DbDish

{

Name = dish.Name,

CookingTimeMinutes = dish.CookingTimeMinutes,

DishTypeId = dish.DishTypeId,

WeightGrams = dish.WeightGrams,

Image = dish.Image,

Recipe = dish.Recipe,

};

context.Dishes.Add(dbDish);

await context.SaveChangesAsync();

dish.Id = dbDish.Id;

return dish;

}

}

/// <summary>

/// Обновляет данные блюда в БД.

/// </summary>

/// <param name="dish"></param>

/// <returns></returns>

/// <exception cref="ArgumentNullException"></exception>

/// <exception cref="Exception"></exception>

public async Task UpdateDishAsync(EditableDish dish)

{

if (dish == null)

{

throw new ArgumentNullException(nameof(dish));

}

using (var context = \_factory())

{

var dbDish = await context

.Dishes

.SingleOrDefaultAsync(d => d.Id == dish.Id)

?? throw new Exception($"Блюдо (Id = {dish.Id}) не найдено в базе данных!");

dbDish.Name = dish.Name;

dbDish.DishTypeId = dish.DishTypeId;

dbDish.CookingTimeMinutes = dish.CookingTimeMinutes;

dbDish.WeightGrams = dish.WeightGrams;

dbDish.Recipe = dish.Recipe;

dbDish.Image = dish.Image;

await context.SaveChangesAsync();

}

}

/// <summary>

/// Удаляет данные блюда из БД.

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

/// <exception cref="Exception"></exception>

public async Task DeleteDishAsync(int id)

{

using (var context = \_factory())

{

var dbDish = await context

.Dishes

.SingleOrDefaultAsync(d => d.Id == id)

?? throw new Exception($"Блюдо (Id = {id}) не найдено в базе данных!");

var hasDishIngredients = await context.Entry(dbDish).Collection(d => d.DishIngredients).Query().AnyAsync();

if (hasDishIngredients)

{

throw new Exception($"Нельзя удалить блюдо (Id = {id}) из-за наличия связанных записей в таблице ингредиентов!");

}

var hasDishCookings = await context.Entry(dbDish).Collection(d => d.DishCookings).Query().AnyAsync();

if (hasDishCookings)

{

throw new Exception($"Нельзя удалить блюдо (Id = {id}) из-за наличия записей в таблице приготовлений!");

}

context.Dishes.Remove(dbDish);

await context.SaveChangesAsync();

}

}

}

}

Б.32 DishIngredientService.cs

using System;

using System.Collections.Generic;

using System.Threading.Tasks;

using WpfCrud.DbModels;

using WpfCrud.Models;

using System.Linq;

using System.Data.Entity;

namespace WpfCrud.Services.DishIngredients

{

public class DishIngredientService

{

private readonly Func<FoodServiceDbContext> \_factory = () => new FoodServiceDbContext();

public DishIngredientService()

{

}

public DishIngredientService(Func<FoodServiceDbContext> factory)

{

\_factory = factory;

}

public async Task<IEnumerable<ViewDishIngredient>> GetViewDishIngredientsAsync()

{

using (var context = \_factory())

{

var dbResults = await

(from di in context.DishIngredients

join d in context.Dishes on di.DishId equals d.Id

join dt in context.DishTypes on d.DishTypeId equals dt.Id

join p in context.Products on di.ProductId equals p.Id

select new

{

DishId = d.Id,

DishName = d.Name,

DishTypeName = dt.Name,

ProductId = p.Id,

ProductName = p.Name,

ProductRequiredWeightGrams = di.RequiredWeightGrams,

}

).ToListAsync();

var results = from i in dbResults

select new ViewDishIngredient(

i.DishId,

i.DishName,

i.DishTypeName,

i.ProductId,

i.ProductName,

i.ProductRequiredWeightGrams);

return results;

}

}

public async Task<EditableDishIngredient> AddDishIngredientAsync(EditableDishIngredient dishIngredient)

{

if (dishIngredient is null)

{

throw new ArgumentNullException(nameof(dishIngredient));

}

using (var context = \_factory())

{

var alreadyInDatabase = await context

.DishIngredients

.AnyAsync(di => di.DishId == dishIngredient.DishId && di.ProductId == dishIngredient.ProductId);

if (alreadyInDatabase)

{

throw new Exception($"В таблице ингредиентов уже имеется такая запись!");

}

var dbDi = new DishIngredient

{

DishId = dishIngredient.DishId,

ProductId = dishIngredient.ProductId,

RequiredWeightGrams = dishIngredient.RequiredWeightGrams,

};

context.DishIngredients.Add(dbDi);

await context.SaveChangesAsync();

return dishIngredient;

}

}

public async Task UpdateDishIngredientAsync(EditableDishIngredient dishIngredient)

{

if (dishIngredient is null)

{

throw new ArgumentNullException(nameof(dishIngredient));

}

using (var context = \_factory())

{

var dbDi = await context

.DishIngredients

.SingleOrDefaultAsync(

di => di.DishId == dishIngredient.DishId

&& di.ProductId == dishIngredient.ProductId)

?? throw new Exception($"Ингредиент (DishId = {dishIngredient.DishId}, ProductId = {dishIngredient.ProductId}) не найден в базе данных!");

dbDi.RequiredWeightGrams = dishIngredient.RequiredWeightGrams;

await context.SaveChangesAsync();

}

}

public async Task DeleteDishIngredientAsync(int dishId, int productId)

{

using (var context = \_factory())

{

var dbDi = await context

.DishIngredients

.SingleOrDefaultAsync(

di => di.DishId == dishId

&& di.ProductId == productId)

?? throw new Exception($"Ингредиент (DishId = {dishId}, ProductId = {productId}) не найден в базе данных!");

context.DishIngredients.Remove(dbDi);

await context.SaveChangesAsync();

}

}

}

}

Б.33 DishTypeService.cs

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Threading.Tasks;

using WpfCrud.DbModels;

using DbDishType = WpfCrud.DbModels.DishType;

using DishType = WpfCrud.Models.DishType;

namespace WpfCrud.Services.DishTypes

{

public class DishTypeService

{

private readonly Func<FoodServiceDbContext> \_factory = () => new FoodServiceDbContext();

public DishTypeService()

{

}

public DishTypeService(Func<FoodServiceDbContext> factory)

{

\_factory = factory ?? throw new ArgumentNullException(nameof(factory));

}

public async Task<IEnumerable<DishType>> GetDishTypesAsync()

{

using (var context = \_factory())

{

var dbDishTypes = await context.DishTypes.ToListAsync();

var dishTypes =

from dbDt in dbDishTypes

orderby dbDt.Id

select new DishType

{

Id = dbDt.Id,

Name = dbDt.Name

};

return dishTypes;

}

}

public async Task<DishType> AddDishTypeAsync(DishType dishType)

{

if (dishType == null)

{

throw new ArgumentNullException(nameof(dishType));

}

using (var context = \_factory())

{

var dbDishType = await context.DishTypes.SingleOrDefaultAsync(dt => dt.Name == dishType.Name);

if (!(dbDishType is null))

{

throw new Exception("Указанный тип блюда уже существует в базе данных!");

}

dbDishType = new DbDishType { Name = dishType.Name };

context.DishTypes.Add(dbDishType);

await context.SaveChangesAsync();

dishType.Id = dbDishType.Id;

return dishType;

}

}

public async Task<int> GetDishCountByDishType(int dishTypeId)

{

using (var context = \_factory())

{

var count = await context.Dishes.CountAsync(d => d.DishTypeId == dishTypeId);

return count;

}

}

public async Task DeleteDishTypeAsync(int id)

{

using (var context = \_factory())

{

var dbDishType = await context.DishTypes.SingleOrDefaultAsync(dt => dt.Id == id);

if (dbDishType is null)

{

throw new Exception($"Тип блюда (Id = {id}) не найден в базе данных!");

}

context.DishTypes.Remove(dbDishType);

await context.SaveChangesAsync();

}

}

public async Task UpdateDishTypeAsync(DishType dishType)

{

if (dishType == null)

{

throw new ArgumentNullException(nameof(dishType));

}

using (var context = \_factory())

{

var dbDishType = await context.DishTypes.SingleOrDefaultAsync(dt => dt.Id == dishType.Id);

if (dbDishType is null)

{

throw new Exception($"Тип блюда (Id = {dishType.Id}) не найден в базе данных!");

}

dbDishType.Name = dishType.Name;

await context.SaveChangesAsync();

}

}

}

}

Б.34 ProductService.cs

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Threading.Tasks;

using WpfCrud.DbModels;

using WpfCrud.Models;

namespace WpfCrud.Services.Products

{

public class ProductService

{

private readonly Func<FoodServiceDbContext> \_factory = () => new FoodServiceDbContext();

public ProductService()

{

}

public ProductService(Func<FoodServiceDbContext> factory)

{

\_factory = factory;

}

public async Task<IEnumerable<ViewProduct>> GetViewProductsAsync()

{

using (var context = \_factory())

{

var dbProducts = await context.Products.ToListAsync();

var products = from dbp in dbProducts

select new ViewProduct(

dbp.Id,

dbp.Name,

dbp.CaloricContentPer100Grams,

dbp.WeightGrams,

dbp.PricePerKilogramRoubles);

return products;

}

}

public async Task<IEnumerable<ViewProduct>> GetViewProductsByNameAsync(string prodName)

{

if (string.IsNullOrWhiteSpace(prodName))

{

var allProducts = await GetViewProductsAsync();

return allProducts;

}

using (var context = \_factory())

{

var dbProducts = await context.Products.Where(p => p.Name.Contains(prodName)).ToListAsync();

var products = from dbp in dbProducts

select new ViewProduct(

dbp.Id,

dbp.Name,

dbp.CaloricContentPer100Grams,

dbp.WeightGrams,

dbp.PricePerKilogramRoubles);

return products;

}

}

public async Task<EditableProduct> AddProductAsync(EditableProduct product)

{

if (product is null)

{

throw new ArgumentNullException(nameof(product));

}

using (var context = \_factory())

{

var dbProd = await context.Products.SingleOrDefaultAsync(p => p.Name == product.Name);

if (!(dbProd is null))

{

throw new Exception("Продукт с таким наименованием уже есть в базе данных!");

}

dbProd = new Product

{

Name = product.Name,

CaloricContentPer100Grams = product.CaloricContentPer100Grams,

PricePerKilogramRoubles = product.PricePerKilogramRoubles,

WeightGrams = product.WeightGrams,

};

context.Products.Add(dbProd);

await context.SaveChangesAsync();

product.Id = dbProd.Id;

return product;

}

}

public async Task UpdateProductAsync(EditableProduct product)

{

if (product == null)

{

throw new ArgumentNullException(nameof(product));

}

using (var context = \_factory())

{

var dbProd = await context.Products.SingleOrDefaultAsync(p => p.Id == product.Id)

?? throw new Exception($"Продукт (Id = {product.Id}) не найден в базе данных!");

dbProd.Name = product.Name;

dbProd.CaloricContentPer100Grams = product.CaloricContentPer100Grams;

dbProd.PricePerKilogramRoubles = product.PricePerKilogramRoubles;

dbProd.WeightGrams = product.WeightGrams;

await context.SaveChangesAsync();

}

}

public async Task DeleteProductAsync(int id)

{

using (var context = \_factory())

{

var dbProd = await context.Products.SingleOrDefaultAsync(p => p.Id == id)

?? throw new Exception($"Продукт (Id = {id}) не найден в базе данных!");

var hasRelatedIngredients = await context

.Entry(dbProd)

.Collection(p => p.DishIngredients)

.Query()

.AnyAsync();

if (hasRelatedIngredients)

{

throw new Exception($"Продукт (Id = {id}) используется в блюдах!");

}

context.Products.Remove(dbProd);

await context.SaveChangesAsync();

}

}

}

}

Б.35 UserAccountService.cs

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using WpfCrud.DbModels;

using WpfCrud.Models;

namespace WpfCrud.Services.UserAccounts

{

public class UserAccountService

{

private readonly Func<FoodServiceDbContext> \_factory = () => new FoodServiceDbContext();

public UserAccountService()

{

}

public UserAccountService(Func<FoodServiceDbContext> factory)

{

\_factory = factory ?? throw new ArgumentNullException(nameof(factory));

}

public async Task<IEnumerable<ViewUserAccount>> GetViewUserAccountsAsync()

{

using (var context = \_factory())

{

var dbResults = await (

from ua in context.UserAccounts

join ur in context.UserRoles on ua.UserRoleId equals ur.Id

select new

{

ua.Id,

RoleId = ur.Id,

RoleName = ur.Name,

ua.Login,

ua.Image,

}).ToListAsync();

var results = from dbr in dbResults

select new ViewUserAccount(

dbr.Id,

dbr.RoleId,

dbr.RoleName,

dbr.Login,

dbr.Image);

return results;

}

}

public async Task<EditableUserAccount> AddUserAccountAsync(EditableUserAccount userAccount)

{

if (userAccount is null)

{

throw new ArgumentNullException(nameof(userAccount));

}

using (var context = \_factory())

{

var loginAlreadyInBase = await context.UserAccounts.AnyAsync(ua => ua.Login == userAccount.Login);

if (loginAlreadyInBase)

{

throw new Exception($"Пользователь с логином <{userAccount.Login}> уже есть в базе данных!");

}

var dbUserAccount = new UserAccount()

{

Login = userAccount.Login,

Password = GetPasswordUtf8Sha512Hash(userAccount.Password),

UserRoleId = (int)userAccount.Role,

Image = userAccount.Image,

};

context.UserAccounts.Add(dbUserAccount);

await context.SaveChangesAsync();

userAccount.Id = dbUserAccount.Id;

return userAccount;

}

}

public async Task UpdateUserAccountAsync(EditableUserAccount userAccount)

{

if (userAccount is null)

{

throw new ArgumentNullException(nameof(userAccount));

}

using (var context = \_factory())

{

var dbUserAccount = await context.UserAccounts.SingleOrDefaultAsync(a => a.Id == userAccount.Id)

?? throw new Exception($"Пользователь (Id = {userAccount.Id}) не найден в базе данных!");

dbUserAccount.Login = userAccount.Login;

dbUserAccount.Password = GetPasswordUtf8Sha512Hash(userAccount.Password);

dbUserAccount.UserRoleId = (int)userAccount.Role;

dbUserAccount.Image = userAccount.Image;

await context.SaveChangesAsync();

}

}

public async Task DeleteUserAccountAsync(int id)

{

using (var context = \_factory())

{

var dbUserAccount = await context.UserAccounts.SingleOrDefaultAsync(a => a.Id == id)

?? throw new Exception($"Пользователь (Id = {id}) не найден в базе данных!");

context.UserAccounts.Remove(dbUserAccount);

await context.SaveChangesAsync();

}

}

public static byte[] GetPasswordUtf8Sha512Hash(string password)

{

if (password is null)

{

throw new ArgumentNullException(nameof(password));

}

var utf8Bytes = Encoding.UTF8.GetBytes(password);

using (var sha512 = new SHA512Managed())

{

var hash = sha512.ComputeHash(utf8Bytes);

return hash;

}

}

}

}

Б.36 AdminViewModel.cs

using WpfCrud.Models;

namespace WpfCrud.ViewModels

{

public class AdminViewModel : ViewModelBase

{

private readonly CurrentUser \_currentUser;

private readonly UserAccountListingCrudViewModel \_userAccountListingCrudViewModel;

private readonly ProductListingCrudViewModel \_productListingCrudViewModel;

private readonly DishTypeListingCrudViewModel \_dishTypeListingCrudViewModel;

private readonly DishCookingListingCrudViewModel \_dishCookingListingCrudViewModel;

private readonly DishIngredientListingCrudViewModel \_dishIngredientListingCrudViewModel;

private readonly DishListingCrudViewModel \_dishListingCrudViewModel;

public AdminViewModel(CurrentUser currentUser)

{

\_currentUser = currentUser ?? throw new System.ArgumentNullException(nameof(currentUser));

\_userAccountListingCrudViewModel = new UserAccountListingCrudViewModel(currentUser);

\_productListingCrudViewModel = new ProductListingCrudViewModel();

\_dishTypeListingCrudViewModel = new DishTypeListingCrudViewModel();

\_dishCookingListingCrudViewModel = new DishCookingListingCrudViewModel();

\_dishIngredientListingCrudViewModel = new DishIngredientListingCrudViewModel();

\_dishListingCrudViewModel = new DishListingCrudViewModel();

}

public AdminViewModel(

ProductListingCrudViewModel productListingCrudViewModel,

DishTypeListingCrudViewModel dishTypeListingCrudViewModel,

DishCookingListingCrudViewModel dishCookingListingCrudViewModel,

DishIngredientListingCrudViewModel dishIngredientListingCrudViewModel,

DishListingCrudViewModel dishListingCrudViewModel,

UserAccountListingCrudViewModel userAccountListingCrudViewModel)

{

\_productListingCrudViewModel = productListingCrudViewModel ?? throw new System.ArgumentNullException(nameof(productListingCrudViewModel));

\_dishTypeListingCrudViewModel = dishTypeListingCrudViewModel ?? throw new System.ArgumentNullException(nameof(dishTypeListingCrudViewModel));

\_dishCookingListingCrudViewModel = dishCookingListingCrudViewModel ?? throw new System.ArgumentNullException(nameof(dishCookingListingCrudViewModel));

\_dishIngredientListingCrudViewModel = dishIngredientListingCrudViewModel ?? throw new System.ArgumentNullException(nameof(dishIngredientListingCrudViewModel));

\_dishListingCrudViewModel = dishListingCrudViewModel ?? throw new System.ArgumentNullException(nameof(dishListingCrudViewModel));

\_userAccountListingCrudViewModel = userAccountListingCrudViewModel ?? throw new System.ArgumentNullException(nameof(userAccountListingCrudViewModel));

}

public ProductListingCrudViewModel ProductListingCrudViewModel => \_productListingCrudViewModel;

public DishTypeListingCrudViewModel DishTypeListingCrudViewModel => \_dishTypeListingCrudViewModel;

public DishCookingListingCrudViewModel DishCookingListingCrudViewModel => \_dishCookingListingCrudViewModel;

public DishIngredientListingCrudViewModel DishIngredientListingCrudViewModel => \_dishIngredientListingCrudViewModel;

public DishListingCrudViewModel DishListingCrudViewModel => \_dishListingCrudViewModel;

public UserAccountListingCrudViewModel UserAccountListingCrudViewModel => \_userAccountListingCrudViewModel;

}

}

Б.37 ChefViewModel.cs

namespace WpfCrud.ViewModels

{

public class ChefViewModel : ViewModelBase

{

private readonly ProductListingCrudViewModel \_productListingCrudViewModel;

private readonly DishTypeListingCrudViewModel \_dishTypeListingCrudViewModel;

private readonly DishCookingListingCrudViewModel \_dishCookingListingCrudViewModel;

private readonly DishIngredientListingCrudViewModel \_dishIngredientListingCrudViewModel;

private readonly DishListingCrudViewModel \_dishListingCrudViewModel;

public ChefViewModel()

{

\_productListingCrudViewModel = new ProductListingCrudViewModel();

\_dishTypeListingCrudViewModel = new DishTypeListingCrudViewModel();

\_dishCookingListingCrudViewModel = new DishCookingListingCrudViewModel();

\_dishIngredientListingCrudViewModel = new DishIngredientListingCrudViewModel();

\_dishListingCrudViewModel = new DishListingCrudViewModel();

}

public ChefViewModel(

ProductListingCrudViewModel productListingCrudViewModel,

DishTypeListingCrudViewModel dishTypeListingCrudViewModel,

DishCookingListingCrudViewModel dishCookingListingCrudViewModel,

DishIngredientListingCrudViewModel dishIngredientListingCrudViewModel,

DishListingCrudViewModel dishListingCrudViewModel)

{

\_productListingCrudViewModel = productListingCrudViewModel ?? throw new System.ArgumentNullException(nameof(productListingCrudViewModel));

\_dishTypeListingCrudViewModel = dishTypeListingCrudViewModel ?? throw new System.ArgumentNullException(nameof(dishTypeListingCrudViewModel));

\_dishCookingListingCrudViewModel = dishCookingListingCrudViewModel ?? throw new System.ArgumentNullException(nameof(dishCookingListingCrudViewModel));

\_dishIngredientListingCrudViewModel = dishIngredientListingCrudViewModel ?? throw new System.ArgumentNullException(nameof(dishIngredientListingCrudViewModel));

\_dishListingCrudViewModel = dishListingCrudViewModel ?? throw new System.ArgumentNullException(nameof(dishListingCrudViewModel));

}

public ProductListingCrudViewModel ProductListingCrudViewModel => \_productListingCrudViewModel;

public DishTypeListingCrudViewModel DishTypeListingCrudViewModel => \_dishTypeListingCrudViewModel;

public DishCookingListingCrudViewModel DishCookingListingCrudViewModel => \_dishCookingListingCrudViewModel;

public DishIngredientListingCrudViewModel DishIngredientListingCrudViewModel => \_dishIngredientListingCrudViewModel;

public DishListingCrudViewModel DishListingCrudViewModel => \_dishListingCrudViewModel;

}

}

Б.38 CookViewModel.cs

namespace WpfCrud.ViewModels

{

public class CookViewModel : ViewModelBase

{

public CookViewModel()

{

ProductListingViewModel = new ProductListingViewModel();

DishListingViewModel = new DishListingViewModel();

DishCookingListingViewModel = new DishCookingListingViewModel();

}

public CookViewModel(ProductListingViewModel productListingViewModel, DishListingViewModel dishListingViewModel, DishCookingListingViewModel dishCookingListingViewModel)

{

ProductListingViewModel = productListingViewModel ?? throw new System.ArgumentNullException(nameof(productListingViewModel));

DishListingViewModel = dishListingViewModel ?? throw new System.ArgumentNullException(nameof(dishListingViewModel));

DishCookingListingViewModel = dishCookingListingViewModel ?? throw new System.ArgumentNullException(nameof(dishCookingListingViewModel));

}

public ProductListingViewModel ProductListingViewModel { get; }

public DishListingViewModel DishListingViewModel { get; }

public DishCookingListingViewModel DishCookingListingViewModel { get; }

}

}

Б.39 DishCookingListingCrudViewModel.cs

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.Models;

using WpfCrud.Services.DishCookings;

using WpfCrud.Services.Dishes;

namespace WpfCrud.ViewModels

{

public class DishCookingListingCrudViewModel : ViewModelBase

{

private readonly DishCookingService \_dishCookingService = new DishCookingService();

private readonly DishService \_dishService = new DishService();

private readonly ObservableCollection<ViewDishCooking> \_dishCookings = new ObservableCollection<ViewDishCooking>();

private ViewDishCooking \_selectedDishCooking;

public ReadOnlyObservableCollection<ViewDishCooking> DishCookings { get; }

public ViewDishCooking SelectedDishCooking

{

get => \_selectedDishCooking;

set

{

\_selectedDishCooking = value;

OnPropertyChanged(nameof(SelectedDishCooking));

}

}

public ICommand LoadDishCookingsCommand { get; }

public ICommand AddDishCookingCommand { get; }

public ICommand EditDishCookingCommand { get; }

public ICommand DeleteDishCookingCommand { get; }

public DishCookingListingCrudViewModel()

{

DishCookings = new ReadOnlyObservableCollection<ViewDishCooking>(\_dishCookings);

LoadDishCookingsCommand = new AsyncDelegateCommand(LoadDishCookingsAsync, HandleException);

AddDishCookingCommand = new AsyncDelegateCommand(AddDishCookingAsync, HandleException);

EditDishCookingCommand = new AsyncDelegateCommand(EditDishCookingAsync, HandleException);

DeleteDishCookingCommand = new AsyncDelegateCommand(DeleteDishCookingAsync, HandleException);

}

private async Task DeleteDishCookingAsync()

{

if (SelectedDishCooking is null)

{

return;

}

await \_dishCookingService.DeleteDishCookingAsync(SelectedDishCooking.Id);

await LoadDishCookingsAsync();

}

private bool? EditDishCookingWindowShowDialog(EditableDishCooking dishCooking, IEnumerable<ViewDish> dishes)

{

var editDishCookingViewModel = new EditDishCookingViewModel(dishCooking, dishes);

var editDishCookingWindow = new EditDishCookingWindow(editDishCookingViewModel);

var result = editDishCookingWindow.ShowDialog();

return result;

}

private async Task EditDishCookingAsync()

{

if (SelectedDishCooking is null)

{

return;

}

var dc = new EditableDishCooking(

SelectedDishCooking.Id,

SelectedDishCooking.DishId,

SelectedDishCooking.Count,

SelectedDishCooking.CookedAt);

var dishes = await \_dishService.GetDishInfosAsync();

var dialogResult = EditDishCookingWindowShowDialog(dc, dishes);

if (dialogResult != true)

{

return;

}

await \_dishCookingService.UpdateDishCookingAsync(dc);

await LoadDishCookingsAsync();

}

private async Task AddDishCookingAsync()

{

var dc = new EditableDishCooking();

var dishes = await \_dishService.GetDishInfosAsync();

var dialogResult = EditDishCookingWindowShowDialog(dc, dishes);

if (dialogResult != true)

{

return;

}

\_ = await \_dishCookingService.AddDishCookingAsync(dc);

await LoadDishCookingsAsync();

}

private void HandleException(Exception e)

{

MessageBox.Show($"{e.GetType().Name} : {e.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

private async Task LoadDishCookingsAsync()

{

var dishCookings = await \_dishCookingService.GetViewDishCookingsAsync();

\_dishCookings.Clear();

foreach (var dc in dishCookings)

{

\_dishCookings.Add(dc);

}

}

public DishCookingListingCrudViewModel(DishCookingService dishCookingService, DishService dishService) : this()

{

\_dishCookingService = dishCookingService ?? throw new ArgumentNullException(nameof(dishCookingService));

\_dishService = dishService ?? throw new ArgumentNullException(nameof(dishService));

}

}

}

Б.40 DishCookingListingViewModel.cs

using System;

using System.Collections.ObjectModel;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.Models;

using WpfCrud.Services.DishCookings;

namespace WpfCrud.ViewModels

{

public class DishCookingListingViewModel : ViewModelBase

{

private readonly DishCookingService \_dishCookingService = new DishCookingService();

private readonly ObservableCollection<ViewDishCooking> \_filteredDishCookings = new ObservableCollection<ViewDishCooking>();

private string \_dishNameFilter;

public string DishNameFilter

{

get => \_dishNameFilter;

set

{

\_dishNameFilter = value;

OnPropertyChanged(nameof(DishNameFilter));

}

}

public ReadOnlyObservableCollection<ViewDishCooking> FilteredDishCookings { get; }

public ICommand FilterDishCookingsCommand { get; }

public DishCookingListingViewModel()

{

FilteredDishCookings = new ReadOnlyObservableCollection<ViewDishCooking>(\_filteredDishCookings);

FilterDishCookingsCommand = new AsyncDelegateCommand(FilterDishCookingsAsync, HandleException);

}

private void HandleException(Exception e)

{

MessageBox.Show(e.Message, e.GetType().Name, MessageBoxButton.OK, MessageBoxImage.Error);

}

private async Task FilterDishCookingsAsync()

{

var filteredDishCookings = await \_dishCookingService.GetViewDishCookingsByDishNameAsync(DishNameFilter);

\_filteredDishCookings.Clear();

foreach (var fdc in filteredDishCookings)

{

\_filteredDishCookings.Add(fdc);

}

}

public DishCookingListingViewModel(DishCookingService dishCookingService)

{

\_dishCookingService = dishCookingService ?? throw new System.ArgumentNullException(nameof(dishCookingService));

}

}

}

Б.41 DishIngredientListingCrudViewModel.cs

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.Models;

using WpfCrud.Services.Dishes;

using WpfCrud.Services.DishIngredients;

using WpfCrud.Services.Products;

namespace WpfCrud.ViewModels

{

public class DishIngredientListingCrudViewModel : ViewModelBase

{

private readonly DishIngredientService \_dishIngredientService = new DishIngredientService();

private readonly DishService \_dishService = new DishService();

private readonly ProductService \_productService = new ProductService();

private readonly ObservableCollection<ViewDishIngredient> \_dishIngredients = new ObservableCollection<ViewDishIngredient>();

private ViewDishIngredient selectedDishIngredient;

public DishIngredientListingCrudViewModel()

{

DishIngredients = new ReadOnlyObservableCollection<ViewDishIngredient>(\_dishIngredients);

LoadDishIngredientsCommand = new AsyncDelegateCommand(LoadDishIngredientsAsync, HandleException);

AddDishIngredientCommand = new AsyncDelegateCommand(AddDishIngredientAsync, HandleException);

EditDishIngredientCommand = new AsyncDelegateCommand(EditDishIngredientAsync, HandleException);

DeleteDishIngredientCommand = new AsyncDelegateCommand(DeleteDishIngredientAsync, HandleException);

}

private async Task DeleteDishIngredientAsync()

{

if (SelectedDishIngredient is null)

{

return;

}

await \_dishIngredientService.DeleteDishIngredientAsync(SelectedDishIngredient.DishId, SelectedDishIngredient.ProductId);

}

private bool? AddDishIngredientWindowShowDialog(

EditableDishIngredient di,

IEnumerable<ViewDish> dishes,

IEnumerable<ViewProduct> products)

{

var editDishIngredientViewModel = new AddDishIngredientViewModel(di, dishes, products);

var editDishIngredientWindow = new AddDishIngredientWindow(editDishIngredientViewModel);

var result = editDishIngredientWindow.ShowDialog();

return result;

}

private bool? EditExistingDishIngredientWindowShowDialog(

EditableDishIngredient di)

{

var editExistingDishIngredientViewModel = new EditExistingDishIngredientViewModel(di);

var editExistingDishIngredientWindow = new EditExistingDishIngredientWindow(editExistingDishIngredientViewModel);

var result = editExistingDishIngredientWindow.ShowDialog();

return result;

}

private async Task EditDishIngredientAsync()

{

if (SelectedDishIngredient is null)

{

return;

}

var di = new EditableDishIngredient(

SelectedDishIngredient.DishId,

SelectedDishIngredient.ProductId,

SelectedDishIngredient.RequiredWeightGrams);

var dialogResult = EditExistingDishIngredientWindowShowDialog(di);

if (dialogResult != true)

{

return;

}

await \_dishIngredientService.UpdateDishIngredientAsync(di);

await LoadDishIngredientsAsync();

}

private async Task AddDishIngredientAsync()

{

var di = new EditableDishIngredient();

var dishes = await \_dishService.GetDishInfosAsync();

var products = await \_productService.GetViewProductsAsync();

var dialogResult = AddDishIngredientWindowShowDialog(di, dishes, products);

if (dialogResult != true)

{

return;

}

\_ = await \_dishIngredientService.AddDishIngredientAsync(di);

await LoadDishIngredientsAsync();

}

public DishIngredientListingCrudViewModel(DishIngredientService dishIngredientService, DishService dishService, ProductService productService) : this()

{

\_dishIngredientService = dishIngredientService ?? throw new System.ArgumentNullException(nameof(dishIngredientService));

\_dishService = dishService ?? throw new System.ArgumentNullException(nameof(dishService));

\_productService = productService ?? throw new System.ArgumentNullException(nameof(productService));

}

private async Task LoadDishIngredientsAsync()

{

var dishIngredients = await \_dishIngredientService.GetViewDishIngredientsAsync();

\_dishIngredients.Clear();

foreach (var di in dishIngredients)

{

\_dishIngredients.Add(di);

}

}

private void HandleException(Exception e)

{

MessageBox.Show($"{e.GetType().Name} : {e.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

public ReadOnlyObservableCollection<ViewDishIngredient> DishIngredients { get; }

public ViewDishIngredient SelectedDishIngredient

{

get => selectedDishIngredient;

set

{

selectedDishIngredient = value;

OnPropertyChanged(nameof(SelectedDishIngredient));

}

}

public ICommand LoadDishIngredientsCommand { get; }

public ICommand AddDishIngredientCommand { get; }

public ICommand EditDishIngredientCommand { get; }

public ICommand DeleteDishIngredientCommand { get; }

}

}

Б.42 DishListingCrudViewModel.cs

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.Models;

using WpfCrud.Services.Dishes;

using WpfCrud.Services.DishTypes;

namespace WpfCrud.ViewModels

{

public class DishListingCrudViewModel : ViewModelBase

{

private readonly DishService \_dishService = new DishService();

private readonly DishTypeService \_dishTypeService = new DishTypeService();

private readonly ObservableCollection<ViewDish> \_dishes = new ObservableCollection<ViewDish>();

private ViewDish \_selectedDish;

public ViewDish SelectedDish

{

get => \_selectedDish;

set

{

\_selectedDish = value;

OnPropertyChanged(nameof(SelectedDish));

}

}

public ReadOnlyObservableCollection<ViewDish> Dishes { get; }

public ICommand LoadDishesCommand { get; }

public ICommand AddDishCommand { get; }

public ICommand EditDishCommand { get; }

public ICommand DeleteDishCommand { get; }

public DishListingCrudViewModel()

{

Dishes = new ReadOnlyObservableCollection<ViewDish>(\_dishes);

LoadDishesCommand = new AsyncDelegateCommand(LoadDishesAsync, HandleException);

AddDishCommand = new AsyncDelegateCommand(AddDishAsync, HandleException);

EditDishCommand = new AsyncDelegateCommand(EditDishAsync, HandleException);

DeleteDishCommand = new AsyncDelegateCommand(DeleteDishAsync, HandleException);

}

private async Task DeleteDishAsync()

{

if (SelectedDish is null)

{

return;

}

await \_dishService.DeleteDishAsync(SelectedDish.Id);

await LoadDishesAsync();

}

private bool? EditDishWindowShowDialog(EditableDish dish, IEnumerable<DishType> dishTypes)

{

var editDishViewModel = new EditDishViewModel(dish, dishTypes);

var editDishWindow = new EditDishWindow(editDishViewModel);

var result = editDishWindow.ShowDialog();

return result;

}

private async Task EditDishAsync()

{

if (SelectedDish is null)

{

return;

}

var dishTypes = await \_dishTypeService.GetDishTypesAsync();

var dish = new EditableDish(

SelectedDish.Id,

SelectedDish.Name,

SelectedDish.DishTypeId,

SelectedDish.CookingTimeMinutes,

SelectedDish.WeightGrams,

SelectedDish.Recipe,

SelectedDish.Image);

var dialogResult = EditDishWindowShowDialog(dish, dishTypes);

if (dialogResult != true)

{

return;

}

await \_dishService.UpdateDishAsync(dish);

await LoadDishesAsync();

}

private async Task AddDishAsync()

{

var dishTypes = await \_dishTypeService.GetDishTypesAsync();

var dish = new EditableDish();

var dialogResult = EditDishWindowShowDialog(dish, dishTypes);

if (dialogResult != true)

{

return;

}

\_ = await \_dishService.AddDishAsync(dish);

await LoadDishesAsync();

}

private async Task LoadDishesAsync()

{

var dishes = await \_dishService.GetDishInfosAsync();

\_dishes.Clear();

foreach (var d in dishes)

{

\_dishes.Add(d);

}

}

private void HandleException(Exception e)

{

MessageBox.Show($"{e.GetType().Name} : {e.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

Б.43 DishListingViewModel.cs

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.Models;

using WpfCrud.Services.Dishes;

namespace WpfCrud.ViewModels

{

public class DishListingViewModel : ViewModelBase

{

private readonly DishService \_dishService = new DishService();

private readonly ObservableCollection<DishWithIngredients> \_filteredDishes = new ObservableCollection<DishWithIngredients>();

public ReadOnlyObservableCollection<DishWithIngredients> FilteredDishes { get; }

public ICommand FilterDishesCommand { get; }

private string \_dishNameFilter = string.Empty;

private DishWithIngredients \_selectedDish;

public string DishNameFilter

{

get => \_dishNameFilter;

set

{

\_dishNameFilter = value;

OnPropertyChanged(nameof(DishNameFilter));

}

}

public DishWithIngredients SelectedDish

{

get => \_selectedDish;

set

{

\_selectedDish = value;

OnPropertyChanged(nameof(SelectedDish));

OnPropertyChanged(nameof(SelectedDishIngredients));

}

}

public IEnumerable<ViewDishIngredient> SelectedDishIngredients => \_selectedDish?.Ingredients;

public DishListingViewModel()

{

FilteredDishes = new ReadOnlyObservableCollection<DishWithIngredients>(\_filteredDishes);

FilterDishesCommand = new AsyncDelegateCommand(FilterDishesAsync, HandleException);

}

private void HandleException(Exception obj)

{

MessageBox.Show($"{obj.GetType().Name} : {obj.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

private async Task FilterDishesAsync()

{

var filteredDishes = await \_dishService.GetDishesWithIngredientsByDishNameAsync(DishNameFilter);

\_filteredDishes.Clear();

foreach (var fd in filteredDishes)

{

\_filteredDishes.Add(fd);

}

}

public DishListingViewModel(DishService dishService)

{

\_dishService = dishService ?? throw new ArgumentNullException(nameof(dishService));

}

}

}

Б.44 DishTypeListingCrudViewModel.cs

using System;

using System.Collections.ObjectModel;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.Models;

using WpfCrud.Services.DishTypes;

namespace WpfCrud.ViewModels

{

// TODO: при обновлении названия типа блюда нужно обновить представление блюд.

public class DishTypeListingCrudViewModel : ViewModelBase

{

private readonly DishTypeService \_dishTypeService = new DishTypeService();

private readonly ObservableCollection<DishType> \_dishTypes = new ObservableCollection<DishType>();

private DishType \_selectedDishType;

public ReadOnlyObservableCollection<DishType> DishTypes { get; }

public ICommand LoadDishTypesCommand { get; }

public ICommand AddDishTypeCommand { get; }

public ICommand EditDishTypeCommand { get; }

public ICommand DeleteDishTypeCommand { get; }

public DishType SelectedDishType

{

get => \_selectedDishType;

set

{

\_selectedDishType = value;

OnPropertyChanged(nameof(SelectedDishType));

}

}

public DishTypeListingCrudViewModel()

{

DishTypes = new ReadOnlyObservableCollection<DishType>(\_dishTypes);

LoadDishTypesCommand = new AsyncDelegateCommand(LoadDishTypesAsync, HandleException);

AddDishTypeCommand = new AsyncDelegateCommand(AddDishTypeAsync, HandleException);

EditDishTypeCommand = new AsyncDelegateCommand(EditDishTypeAsync, HandleException);

DeleteDishTypeCommand = new AsyncDelegateCommand(DeleteDishTypeAsync, HandleException);

}

private async Task DeleteDishTypeAsync()

{

if (SelectedDishType is null)

{

return;

}

var linkedDishCount = await \_dishTypeService.GetDishCountByDishType(SelectedDishType.Id);

if (linkedDishCount != 0)

{

throw new Exception($"Нельзя удалить тип блюда (Id = {SelectedDishType.Id}), так как есть блюда этого типа!");

}

await \_dishTypeService.DeleteDishTypeAsync(SelectedDishType.Id);

await LoadDishTypesAsync();

}

private bool? EditDishTypeWindowShowDialog(DishType dishType)

{

var editDishTypeViewModel = new EditDishTypeViewModel(dishType);

var editDishTypeWindow = new EditDishTypeWindow(editDishTypeViewModel);

var result = editDishTypeWindow.ShowDialog();

return result;

}

private async Task EditDishTypeAsync()

{

if (SelectedDishType is null)

{

return;

}

var dishType = new DishType

{

Id = SelectedDishType.Id,

Name = SelectedDishType.Name,

};

var dialogResult = EditDishTypeWindowShowDialog(dishType);

if (dialogResult != true)

{

return;

}

await \_dishTypeService.UpdateDishTypeAsync(dishType);

await LoadDishTypesAsync();

}

private async Task AddDishTypeAsync()

{

var dishType = new DishType();

var dialogResult = EditDishTypeWindowShowDialog(dishType);

if (dialogResult != true)

{

return;

}

\_ = await \_dishTypeService.AddDishTypeAsync(dishType);

await LoadDishTypesAsync();

}

private void HandleException(Exception e)

{

MessageBox.Show($"{e.GetType().Name} : {e.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

private async Task LoadDishTypesAsync()

{

var dishTypes = await \_dishTypeService.GetDishTypesAsync();

\_dishTypes.Clear();

foreach (var dt in dishTypes)

{

\_dishTypes.Add(dt);

}

}

}

}

Б.45 EditDishCookingViewModel.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.Models;

namespace WpfCrud.ViewModels

{

public class EditDishCookingViewModel : ViewModelBase

{

private readonly EditableDishCooking \_dishCooking;

public IEnumerable<ViewDish> Dishes { get; }

public EditDishCookingViewModel(EditableDishCooking dishCooking, IEnumerable<ViewDish> dishes)

{

if (dishCooking is null)

{

throw new ArgumentNullException(nameof(dishCooking));

}

if (dishes is null)

{

throw new ArgumentNullException(nameof(dishes));

}

if (!dishes.Any())

{

throw new InvalidOperationException("Список блюд для выбора пуст!");

}

\_dishCooking = dishCooking;

Dishes = dishes;

if (DishId == 0)

{

DishId = Dishes.First().Id;

}

SubmitCommand = new DelegateCommand(Submit);

CancelCommand = new DelegateCommand(Cancel);

}

private void Cancel(object obj)

{

OnCancelSubmit();

}

private void Submit(object obj)

{

try

{

if (DishId == 0)

{

throw new Exception("Блюдо должно быть выбрано!");

}

if (Count <= 0)

{

throw new Exception("Количество блюд должно быть положительным числом!");

}

}

catch (Exception e)

{

OnSubmitError(e);

return;

}

OnSubmitSuccess();

}

public int DishId

{

get => \_dishCooking.DishId;

set

{

\_dishCooking.DishId = value;

OnPropertyChanged(nameof(DishId));

}

}

public int Count

{

get => \_dishCooking.Count;

set

{

\_dishCooking.Count = value;

OnPropertyChanged(nameof(Count));

}

}

public DateTime CookedAt

{

get => \_dishCooking.CookedAt;

set

{

\_dishCooking.CookedAt = value;

OnPropertyChanged(nameof(CookedAt));

}

}

public ICommand SubmitCommand { get; }

public ICommand CancelCommand { get; }

public event Action<Exception> SubmitError;

public event Action SubmitSuccess;

public event Action CancelSubmit;

protected void OnSubmitError(Exception e) => SubmitError?.Invoke(e);

protected void OnSubmitSuccess() => SubmitSuccess?.Invoke();

protected void OnCancelSubmit() => CancelSubmit?.Invoke();

}

}

Б.46 EditDishIngredientViewModel.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.Models;

namespace WpfCrud.ViewModels

{

public class AddDishIngredientViewModel : ViewModelBase

{

private readonly EditableDishIngredient \_di;

public IEnumerable<ViewDish> Dishes { get; }

public IEnumerable<ViewProduct> Products { get; }

public int DishId

{

get => \_di.DishId;

set

{

\_di.DishId = value;

OnPropertyChanged(nameof(DishId));

}

}

public int ProductId

{

get => \_di.ProductId;

set

{

\_di.ProductId = value;

OnPropertyChanged(nameof(ProductId));

}

}

public double RequiredWeightGrams

{

get => \_di.RequiredWeightGrams;

set

{

\_di.RequiredWeightGrams = value;

OnPropertyChanged(nameof(RequiredWeightGrams));

}

}

public ICommand SubmitCommand { get; }

public ICommand CancelCommand { get; }

public event Action<Exception> SubmitError;

public event Action SubmitSuccess;

public event Action CancelSubmit;

protected void OnSubmitError(Exception e) => SubmitError?.Invoke(e);

protected void OnSubmitSuccess() => SubmitSuccess?.Invoke();

protected void OnCancelSubmit() => CancelSubmit?.Invoke();

public AddDishIngredientViewModel(EditableDishIngredient di, IEnumerable<ViewDish> dishes, IEnumerable<ViewProduct> products)

{

if (di is null)

{

throw new ArgumentNullException(nameof(di));

}

if (dishes is null)

{

throw new ArgumentNullException(nameof(dishes));

}

if (products is null)

{

throw new ArgumentNullException(nameof(products));

}

if (!dishes.Any())

{

throw new InvalidOperationException("Список блюд для выбора пуст!");

}

if (!products.Any())

{

throw new InvalidOperationException("Список продуктов для выбора пуст!");

}

\_di = di;

Dishes = dishes;

Products = products;

if (\_di.DishId == 0)

{

DishId = Dishes.First().Id;

}

if (\_di.ProductId == 0)

{

ProductId = Products.First().Id;

}

SubmitCommand = new DelegateCommand(Submit);

CancelCommand = new DelegateCommand(Cancel);

}

private void Cancel(object obj)

{

OnCancelSubmit();

}

private void Submit(object obj)

{

try

{

ValidateProperties();

}

catch (Exception e)

{

OnSubmitError(e);

return;

}

OnSubmitSuccess();

}

private void ValidateProperties()

{

if (DishId == 0)

{

throw new Exception("Блюдо должно быть выбрано!");

}

if (ProductId == 0)

{

throw new Exception("Продукт должен быть выбран!");

}

if (RequiredWeightGrams <= 0d)

{

throw new Exception("Требуемая масса продукта (г) должна быть положительным числом!");

}

}

}

}

Б.47 EditDishTypeViewModel.cs

using System;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.Models;

namespace WpfCrud.ViewModels

{

public class EditDishTypeViewModel : ViewModelBase

{

private readonly DishType \_dishType;

public EditDishTypeViewModel(DishType dishType)

{

\_dishType = dishType ?? throw new System.ArgumentNullException(nameof(dishType));

SubmitCommand = new DelegateCommand(Submit);

CancelCommand = new DelegateCommand(Cancel);

}

private void Cancel(object obj)

{

OnCancelSubmit();

}

private void Submit(object obj)

{

try

{

if (string.IsNullOrWhiteSpace(Name))

{

throw new Exception("Название типа блюда должно быть заполнено!");

}

}

catch (Exception e)

{

OnSubmitError(e);

return;

}

OnSubmitSuccess();

}

public int Id => \_dishType.Id;

public string Name

{

get => \_dishType.Name;

set

{

\_dishType.Name = value;

OnPropertyChanged(nameof(Name));

}

}

public ICommand SubmitCommand { get; }

public ICommand CancelCommand { get; }

public event Action<Exception> SubmitError;

public event Action SubmitSuccess;

public event Action CancelSubmit;

protected void OnSubmitError(Exception e) => SubmitError?.Invoke(e);

protected void OnSubmitSuccess() => SubmitSuccess?.Invoke();

protected void OnCancelSubmit() => CancelSubmit?.Invoke();

}

}

Б.48 EditDishViewModel.cs

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.Models;

namespace WpfCrud.ViewModels

{

public class EditDishViewModel : ViewModelBase

{

private readonly EditableDish \_dish;

public IEnumerable<DishType> DishTypes { get; }

public EditDishViewModel(EditableDish dish, IEnumerable<DishType> dishTypes)

{

if (dish is null)

{

throw new ArgumentNullException(nameof(dish));

}

if (dishTypes is null)

{

throw new ArgumentNullException(nameof(dishTypes));

}

if (!dishTypes.Any())

{

throw new InvalidOperationException($"Список типов блюд <{nameof(dishTypes)}> пуст!");

}

\_dish = dish;

DishTypes = dishTypes;

if (DishTypeId == 0)

{

DishTypeId = DishTypes.First().Id;

}

SubmitCommand = new DelegateCommand(Submit);

CancelCommand = new DelegateCommand(Cancel);

SelectDishImageCommand = new DelegateCommand(SelectDishImage);

DeselectDishImageCommand = new DelegateCommand(\_ => Image = null);

}

private void SelectDishImage(object obj)

{

var dialog = new OpenFileDialog

{

Filter = "Изображения (\*.png, \*.jpeg, \*.jpg)|\*.png;\*.jpeg;\*.jpg",

Multiselect = false,

CheckFileExists = true,

CheckPathExists = true,

ValidateNames = true

};

if (dialog.ShowDialog() != true)

{

return;

}

var imagePath = dialog.FileName;

var imageData = File.ReadAllBytes(imagePath);

Image = imageData;

}

public void Cancel(object obj)

{

OnCancelSubmit();

}

public void Submit(object obj)

{

try

{

ValidateProperties();

}

catch (Exception e)

{

OnSubmitError(e);

return;

}

OnSubmitSuccess();

}

public void ValidateProperties()

{

if (string.IsNullOrWhiteSpace(Name))

{

throw new Exception("Название блюда должно быть заполнено!");

}

if (DishTypeId == 0)

{

throw new Exception("Тип блюда должен быть выбран!");

}

if (CookingTimeMinutes <= 0d)

{

throw new Exception("Время готовки блюда (минут) должно быть положительным числом!");

}

if (WeightGrams <= 0d)

{

throw new Exception("Масса готового блюда (г) должна быть положительным числом!");

}

if (string.IsNullOrEmpty(Recipe))

{

throw new Exception("Рецепт блюда должен быть заполнен!");

}

}

protected void OnSubmitError(Exception e) => SubmitError?.Invoke(e);

protected void OnSubmitSuccess() => SubmitSuccess?.Invoke();

protected void OnCancelSubmit() => CancelSubmit?.Invoke();

public int Id => \_dish.Id;

public string Name

{

get => \_dish.Name;

set

{

\_dish.Name = value;

OnPropertyChanged(nameof(Name));

}

}

public int DishTypeId

{

get => \_dish.DishTypeId;

set

{

\_dish.DishTypeId = value;

OnPropertyChanged(nameof(DishTypeId));

}

}

//public DishType DishType

//{

// get => \_dish.DishType;

// set

// {

// \_dish.DishType = value;

// OnPropertyChanged(nameof(DishType));

// }

//}

public double CookingTimeMinutes

{

get => \_dish.CookingTimeMinutes;

set

{

\_dish.CookingTimeMinutes = value;

OnPropertyChanged(nameof(CookingTimeMinutes));

}

}

public double WeightGrams

{

get => \_dish.WeightGrams;

set

{

\_dish.WeightGrams = value;

OnPropertyChanged(nameof(WeightGrams));

}

}

public string Recipe

{

get => \_dish.Recipe;

set

{

\_dish.Recipe = value;

OnPropertyChanged(nameof(Recipe));

}

}

public byte[] Image

{

get => \_dish.Image;

set

{

\_dish.Image = value;

OnPropertyChanged(nameof(Image));

}

}

public ICommand SubmitCommand { get; }

public ICommand CancelCommand { get; }

public ICommand SelectDishImageCommand { get; }

public ICommand DeselectDishImageCommand { get; }

public event Action<Exception> SubmitError;

public event Action SubmitSuccess;

public event Action CancelSubmit;

}

}

Б.49 EditExistingDishIngredientViewModel.cs

using System;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.Models;

using WpfCrud.ViewModels;

namespace WpfCrud.ViewModels

{

public class EditExistingDishIngredientViewModel : ViewModelBase

{

private readonly EditableDishIngredient \_di;

public EditExistingDishIngredientViewModel(EditableDishIngredient di)

{

if (di is null)

{

throw new ArgumentNullException(nameof(di));

}

if (di.DishId == 0)

{

throw new InvalidOperationException("di.DishId is 0");

}

if (di.ProductId == 0)

{

throw new InvalidOperationException("di.ProductId is 0");

}

\_di = di;

SubmitCommand = new DelegateCommand(Submit);

CancelCommand = new DelegateCommand(Cancel);

}

private void Cancel(object obj)

{

OnCancelSubmit();

}

private void Submit(object obj)

{

try

{

if (RequiredWeightGrams <= 0)

{

throw new Exception("Требуемая масса продукта (г) должна быть положительным числом!");

}

}

catch (Exception e)

{

OnSubmitError(e);

return;

}

OnSubmitSuccess();

}

public double RequiredWeightGrams

{

get => \_di.RequiredWeightGrams;

set

{

\_di.RequiredWeightGrams = value;

OnPropertyChanged(nameof(RequiredWeightGrams));

}

}

public ICommand SubmitCommand { get; }

public ICommand CancelCommand { get; }

public event Action<Exception> SubmitError;

public event Action SubmitSuccess;

public event Action CancelSubmit;

protected void OnSubmitError(Exception e) => SubmitError?.Invoke(e);

protected void OnSubmitSuccess() => SubmitSuccess?.Invoke();

protected void OnCancelSubmit() => CancelSubmit?.Invoke();

}

}

Б.50 EditProductViewModel.cs

using System;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.Models;

namespace WpfCrud.ViewModels

{

public class EditProductViewModel : ViewModelBase

{

private readonly EditableProduct \_editableProduct;

public EditProductViewModel(EditableProduct editableProduct)

{

\_editableProduct = editableProduct ?? throw new System.ArgumentNullException(nameof(editableProduct));

SubmitCommand = new DelegateCommand(Submit);

CancelCommand = new DelegateCommand(Cancel);

}

public void Cancel(object obj)

{

OnCancelSubmit();

}

public void Submit(object obj)

{

try

{

ValidateProperties();

}

catch (Exception e)

{

OnSubmitError(e);

return;

}

OnSubmitSucces();

}

public void ValidateProperties()

{

if (string.IsNullOrWhiteSpace(Name))

{

throw new Exception("Наименование должно быть заполнено!");

}

if (CaloricContentPer100Grams <= 0d)

{

throw new Exception("Калорийность (ккал / 100 г) должна быть положительным числом!");

}

if (WeightGrams <= 0d)

{

throw new Exception("Масса (г) должна быть положительным числом!");

}

if (PricePerKilogramRoubles <= 0m)

{

throw new Exception("Цена за 1 кг должна быть положительным числом!");

}

}

public int Id => \_editableProduct.Id;

public string Name

{

get => \_editableProduct.Name;

set

{

\_editableProduct.Name = value;

OnPropertyChanged(nameof(Name));

}

}

public double CaloricContentPer100Grams

{

get => \_editableProduct.CaloricContentPer100Grams;

set

{

\_editableProduct.CaloricContentPer100Grams = value;

OnPropertyChanged(nameof(CaloricContentPer100Grams));

}

}

public double WeightGrams

{

get => \_editableProduct.WeightGrams;

set

{

\_editableProduct.WeightGrams = value;

OnPropertyChanged(nameof(WeightGrams));

}

}

public decimal PricePerKilogramRoubles

{

get => \_editableProduct.PricePerKilogramRoubles;

set

{

\_editableProduct.PricePerKilogramRoubles = value;

OnPropertyChanged(nameof(PricePerKilogramRoubles));

}

}

public ICommand SubmitCommand { get; }

public ICommand CancelCommand { get; }

public event Action<Exception> SubmitError;

public event Action SubmitSuccess;

public event Action CancelSubmit;

protected void OnSubmitError(Exception e) => SubmitError?.Invoke(e);

protected void OnSubmitSucces() => SubmitSuccess?.Invoke();

protected void OnCancelSubmit() => CancelSubmit?.Invoke();

}

}

Б.51 EditUserAccountViewModel.cs

using Microsoft.Win32;

using System;

using System.IO;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.Models;

using WpfCrud.Models.Enums;

namespace WpfCrud.ViewModels

{

public class EditUserAccountViewModel : ViewModelBase

{

private static readonly UserRoleEnum[] \_userRoles =

{

UserRoleEnum.Cook,

UserRoleEnum.Chef,

UserRoleEnum.Admin,

};

private readonly EditableUserAccount \_userAccount;

public static UserRoleEnum[] UserRoles => \_userRoles;

public EditUserAccountViewModel(EditableUserAccount userAccount)

{

\_userAccount = userAccount ?? throw new ArgumentNullException(nameof(userAccount));

SubmitCommand = new DelegateCommand(Submit);

CancelCommand = new DelegateCommand(Cancel);

SelectImageCommand = new DelegateCommand(SelectImage);

DeselectImageCommand = new DelegateCommand(\_ => Image = null);

}

private void SelectImage(object obj)

{

var dialog = new OpenFileDialog

{

Filter = "Изображения (\*.png, \*.jpeg, \*.jpg)|\*.png;\*.jpeg;\*.jpg",

Multiselect = false,

CheckFileExists = true,

CheckPathExists = true,

ValidateNames = true

};

if (dialog.ShowDialog() != true)

{

return;

}

var imagePath = dialog.FileName;

var imageData = File.ReadAllBytes(imagePath);

Image = imageData;

}

private void Cancel(object obj)

{

OnCancelSubmit();

}

private void Submit(object obj)

{

try

{

if (string.IsNullOrWhiteSpace(Login))

{

throw new Exception("Логин должен быть заполнен!");

}

if (string.IsNullOrWhiteSpace(Password))

{

throw new Exception("Пароль должен быть заполнен!");

}

if (UserRole == UserRoleEnum.Unauthorized)

{

throw new Exception("Роль пользователя в системе должна быть выбрана!");

}

}

catch (Exception e)

{

OnSubmitError(e);

return;

}

OnSubmitSuccess();

}

public int Id => \_userAccount.Id;

public string Login

{

get => \_userAccount.Login;

set

{

\_userAccount.Login = value;

OnPropertyChanged(nameof(Login));

}

}

public string Password

{

get => \_userAccount.Password;

set

{

\_userAccount.Password = value;

OnPropertyChanged(nameof(Password));

}

}

public UserRoleEnum UserRole

{

get => \_userAccount.Role;

set

{

\_userAccount.Role = value;

OnPropertyChanged(nameof(UserRole));

}

}

public byte[] Image

{

get => \_userAccount.Image;

set

{

\_userAccount.Image = value;

OnPropertyChanged(nameof(Image));

}

}

public ICommand SubmitCommand { get; }

public ICommand CancelCommand { get; }

public ICommand SelectImageCommand { get; }

public ICommand DeselectImageCommand { get; }

public event Action<Exception> SubmitError;

public event Action SubmitSuccess;

public event Action CancelSubmit;

protected void OnSubmitError(Exception e) => SubmitError?.Invoke(e);

protected void OnSubmitSuccess() => SubmitSuccess?.Invoke();

protected void OnCancelSubmit() => CancelSubmit?.Invoke();

}

}

Б.52 LoginViewModel.cs

using System;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.DbModels;

using WpfCrud.Models;

using WpfCrud.Models.Enums;

namespace WpfCrud.ViewModels

{

public class LoginViewModel : ViewModelBase

{

public const string CorrectCaptcha = "1337";

private string \_password;

private string \_captcha;

private readonly CurrentUser \_currentUser = new CurrentUser();

public string Login

{

get => \_currentUser.Login;

set

{

\_currentUser.Login = value;

OnPropertyChanged(nameof(Login));

OnPropertyChanged(nameof(CanSubmit));

}

}

public string Password

{

get => \_password;

set

{

\_password = value;

OnPropertyChanged(nameof(Password));

OnPropertyChanged(nameof(CanSubmit));

}

}

public int CurrentUserId

{

get => \_currentUser.Id;

set

{

\_currentUser.Id = value;

OnPropertyChanged(nameof(CurrentUserId));

}

}

public UserRoleEnum CurrentUserRole

{

get => \_currentUser.Role;

set

{

\_currentUser.Role = value;

OnPropertyChanged(nameof(CurrentUserRole));

}

}

public ICommand LoginCommand { get; }

public event Action LoginSuccess;

public event Action<string> LoginFailure;

private void OnLoginSuccess() => LoginSuccess?.Invoke();

private void OnLoginFailure(string reason) => LoginFailure?.Invoke(reason);

public LoginViewModel()

{

LoginCommand = new DelegateCommand(Submit, null);

}

public bool CanSubmit =>

!string.IsNullOrWhiteSpace(Login)

&& !string.IsNullOrWhiteSpace(Password)

&& Captcha == CorrectCaptcha;

public string Captcha

{

get => \_captcha;

set

{

\_captcha = value;

OnPropertyChanged(nameof(CanSubmit));

}

}

private void Submit(object obj)

{

UserAccount foundUser = null;

try

{

var loginParam = new SqlParameter("@login", SqlDbType.NVarChar, 100) { Value = Login };

var passwordParam = new SqlParameter(@"password", SqlDbType.VarChar) { Value = Password };

using (var context = new FoodServiceDbContext())

{

foundUser = context

.UserAccounts

.SqlQuery(

"SELECT \* FROM UserAccount WHERE Login = @Login AND Password = HASHBYTES('sha2\_512', @Password)",

loginParam,

passwordParam)

.SingleOrDefault();

}

}

catch (Exception e)

{

OnLoginFailure($"{e.GetType().Name} : {e.Message}");

return;

}

if (foundUser is null)

{

OnLoginFailure("Логин и/или пароль введены неверно.");

return;

}

CurrentUserId = foundUser.Id;

CurrentUserRole = (UserRoleEnum)foundUser.UserRoleId;

OnLoginSuccess();

}

}

}

Б.53 MainViewModel.cs

namespace WpfCrud.ViewModels

{

public class MainViewModel : ViewModelBase

{

private readonly ViewModelBase \_currentViewModel;

public MainViewModel(ViewModelBase currentViewModel)

{

\_currentViewModel = currentViewModel ?? throw new System.ArgumentNullException(nameof(currentViewModel));

}

public ViewModelBase CurrentViewModel => \_currentViewModel;

}

}

Б.54 ProductListingCrudViewModel.cs

using System;

using System.Collections.ObjectModel;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.Models;

using WpfCrud.Services.Products;

namespace WpfCrud.ViewModels

{

public class ProductListingCrudViewModel : ViewModelBase

{

private readonly ProductService \_productService = new ProductService();

private readonly ObservableCollection<ViewProduct> \_products = new ObservableCollection<ViewProduct>();

private ViewProduct \_selectedProduct;

public ReadOnlyObservableCollection<ViewProduct> Products { get; }

public ICommand LoadProductsCommand { get; }

public ICommand AddProductCommand { get; }

public ICommand EditProductCommand { get; }

public ICommand DeleteProductCommand { get; }

public ViewProduct SelectedProduct

{

get => \_selectedProduct;

set

{

\_selectedProduct = value;

OnPropertyChanged(nameof(SelectedProduct));

}

}

public ProductListingCrudViewModel()

{

Products = new ReadOnlyObservableCollection<ViewProduct>(\_products);

LoadProductsCommand = new AsyncDelegateCommand(LoadProductsAsync, HandleException);

AddProductCommand = new AsyncDelegateCommand(AddProductAsync, HandleException);

EditProductCommand = new AsyncDelegateCommand(EditProductAsync, HandleException);

DeleteProductCommand = new AsyncDelegateCommand(DeleteProductAsync, HandleException);

}

private async Task DeleteProductAsync()

{

if (SelectedProduct is null)

{

return;

}

await \_productService.DeleteProductAsync(SelectedProduct.Id);

await LoadProductsAsync();

}

private bool? EditProductWindowShowDialog(EditableProduct product)

{

if (product is null)

{

throw new ArgumentNullException(nameof(product));

}

var editProductViewModel = new EditProductViewModel(product);

var editProductWindow = new EditProductWindow(editProductViewModel);

var result = editProductWindow.ShowDialog();

return result;

}

private async Task EditProductAsync()

{

if (SelectedProduct is null)

{

return;

}

var product = new EditableProduct(

SelectedProduct.Id,

SelectedProduct.Name,

SelectedProduct.CaloricContentPer100Grams,

SelectedProduct.WeightGrams,

SelectedProduct.PricePerKilogramRoubles);

var dialogResult = EditProductWindowShowDialog(product);

if (dialogResult != true)

{

return;

}

await \_productService.UpdateProductAsync(product);

await LoadProductsAsync();

}

private async Task AddProductAsync()

{

var product = new EditableProduct();

var dialogResult = EditProductWindowShowDialog(product);

if (dialogResult != true)

{

return;

}

\_ = await \_productService.AddProductAsync(product);

await LoadProductsAsync();

}

private void HandleException(Exception obj)

{

MessageBox.Show(

$"{obj.GetType().Name} : {obj.Message}",

"Ошибка",

MessageBoxButton.OK,

MessageBoxImage.Error);

}

private async Task LoadProductsAsync()

{

var products = await \_productService.GetViewProductsAsync();

\_products.Clear();

foreach (var p in products)

{

\_products.Add(p);

}

}

}

}

Б.55 ProductListingViewModel.cs

using System;

using System.Collections.ObjectModel;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.Models;

using WpfCrud.Services.Products;

namespace WpfCrud.ViewModels

{

public class ProductListingViewModel : ViewModelBase

{

private readonly ProductService \_productService = new ProductService();

private readonly ObservableCollection<ViewProduct> \_filteredProducts = new ObservableCollection<ViewProduct>();

public ReadOnlyObservableCollection<ViewProduct> FilteredProducts { get; }

private string \_productNameFilter;

public string ProductNameFilter

{

get => \_productNameFilter;

set

{

\_productNameFilter = value;

OnPropertyChanged(nameof(ProductNameFilter));

}

}

public ICommand FilterProductsCommand { get; }

private async Task FilterProductsAsync()

{

var filteredProducts = await \_productService.GetViewProductsByNameAsync(ProductNameFilter);

\_filteredProducts.Clear();

foreach (var fp in filteredProducts)

{

\_filteredProducts.Add(fp);

}

}

private void HandleException(Exception obj)

{

MessageBox.Show($"{obj.GetType().Name} : {obj.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

public ProductListingViewModel()

{

FilteredProducts = new ReadOnlyObservableCollection<ViewProduct>(\_filteredProducts);

FilterProductsCommand = new AsyncDelegateCommand(FilterProductsAsync, HandleException);

}

public ProductListingViewModel(ProductService productService) : this()

{

\_productService = productService ?? throw new System.ArgumentNullException(nameof(productService));

}

}

}

Б.56 UserAccountListingCrudViewModel.cs

using System;

using System.Collections.ObjectModel;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using WpfCrud.Commands;

using WpfCrud.Models;

using WpfCrud.Services.UserAccounts;

namespace WpfCrud.ViewModels

{

public class UserAccountListingCrudViewModel : ViewModelBase

{

private readonly CurrentUser \_currentUser;

private readonly UserAccountService \_userAccountService = new UserAccountService();

private readonly ObservableCollection<ViewUserAccount> \_userAccounts = new ObservableCollection<ViewUserAccount>();

private ViewUserAccount \_selectedUserAccount;

public UserAccountListingCrudViewModel(CurrentUser currentUser)

{

\_currentUser = currentUser ?? throw new ArgumentNullException(nameof(currentUser));

UserAccounts = new ReadOnlyObservableCollection<ViewUserAccount>(\_userAccounts);

LoadUserAccountsCommand = new AsyncDelegateCommand(LoadUserAccountsAsync, HandleException);

AddUserAccountCommand = new AsyncDelegateCommand(AddUserAccountAsync, HandleException);

EditUserAccountCommand = new AsyncDelegateCommand(EditUserAccountAsync, HandleException);

DeleteUserAccountCommand = new AsyncDelegateCommand(DeleteUserAccountAsync, HandleException);

}

private async Task DeleteUserAccountAsync()

{

if (SelectedUserAccount is null)

{

return;

}

if (SelectedUserAccount.Id == \_currentUser.Id)

{

throw new Exception("Удаление текущего пользователя системы запрещено!");

}

await \_userAccountService.DeleteUserAccountAsync(SelectedUserAccount.Id);

await LoadUserAccountsAsync();

}

private bool? EditUserAccountWindowShowDialog(EditableUserAccount editableUserAccount)

{

var editUserAccountViewModel = new EditUserAccountViewModel(editableUserAccount);

var editUserAccountWindow = new EditUserAccountWindow(editUserAccountViewModel);

var result = editUserAccountWindow.ShowDialog();

return result;

}

private async Task EditUserAccountAsync()

{

if (SelectedUserAccount is null)

{

return;

}

if (SelectedUserAccount.Id == \_currentUser.Id)

{

throw new Exception("Редактирование текущего пользователя системы не реализовано!");

}

var userAccount = new EditableUserAccount(

SelectedUserAccount.Id,

SelectedUserAccount.Login,

null,

SelectedUserAccount.Role,

SelectedUserAccount.Image);

var dialogResult = EditUserAccountWindowShowDialog(userAccount);

if (dialogResult != true)

{

return;

}

await \_userAccountService.UpdateUserAccountAsync(userAccount);

await LoadUserAccountsAsync();

}

private async Task AddUserAccountAsync()

{

var userAccount = new EditableUserAccount();

var dialogResult = EditUserAccountWindowShowDialog(userAccount);

if (dialogResult != true)

{

return;

}

\_ = await \_userAccountService.AddUserAccountAsync(userAccount);

await LoadUserAccountsAsync();

}

private void HandleException(Exception obj)

{

MessageBox.Show($"{obj.GetType().Name} : {obj.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

private async Task LoadUserAccountsAsync()

{

var userAccounts = await \_userAccountService.GetViewUserAccountsAsync();

\_userAccounts.Clear();

foreach (var ua in userAccounts)

{

\_userAccounts.Add(ua);

}

}

public UserAccountListingCrudViewModel(CurrentUser currentUser, UserAccountService userAccountService) : this(currentUser)

{

\_userAccountService = userAccountService ?? throw new ArgumentNullException(nameof(userAccountService));

}

public ReadOnlyObservableCollection<ViewUserAccount> UserAccounts { get; }

public ViewUserAccount SelectedUserAccount

{

get => \_selectedUserAccount;

set

{

\_selectedUserAccount = value;

OnPropertyChanged(nameof(SelectedUserAccount));

}

}

public ICommand LoadUserAccountsCommand { get; }

public ICommand AddUserAccountCommand { get; }

public ICommand EditUserAccountCommand { get; }

public ICommand DeleteUserAccountCommand { get; }

}

}

Б.57 ViewModelBase.cs

using System.ComponentModel;

namespace WpfCrud.ViewModels

{

public abstract class ViewModelBase : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged(string propertyName = null) => PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

Б.58 AdminView.xaml

<UserControl x:Class="WpfCrud.Views.AdminView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

xmlns:views="clr-namespace:WpfCrud.Views"

d:DataContext="{d:DesignInstance Type=viewmodels:AdminViewModel}"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

d:Background="White">

<TabControl>

<TabItem Header="Пользователи">

<views:UserAccountListingCrudView

DataContext="{Binding UserAccountListingCrudViewModel}"

Margin="10"/>

</TabItem>

<TabItem Header="Продукты">

<views:ProductListingCrudView

DataContext="{Binding ProductListingCrudViewModel}"

Margin="10"/>

</TabItem>

<TabItem Header="Типы блюд">

<views:DishTypeListingCrudView

DataContext="{Binding DishTypeListingCrudViewModel}"

Margin="10"/>

</TabItem>

<TabItem Header="Блюда">

<views:DishListingCrudView

DataContext="{Binding DishListingCrudViewModel}"

Margin="10"/>

</TabItem>

<TabItem Header="Ингредиенты блюд">

<views:DishIngredientListingCrudView

DataContext="{Binding DishIngredientListingCrudViewModel}"

Margin="10"/>

</TabItem>

<TabItem Header="Приготовления">

<views:DishCookingListingCrudView

DataContext="{Binding DishCookingListingCrudViewModel}"

Margin="10"/>

</TabItem>

</TabControl>

</UserControl>

Б.59 ChefView.xaml

<UserControl x:Class="WpfCrud.Views.ChefView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

xmlns:views="clr-namespace:WpfCrud.Views"

d:DataContext="{d:DesignInstance Type=viewmodels:ChefViewModel}"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800">

<TabControl>

<TabItem Header="Продукты">

<views:ProductListingCrudView

Margin="10"

DataContext="{Binding ProductListingCrudViewModel}"/>

</TabItem>

<TabItem Header="Типы блюд">

<views:DishTypeListingCrudView

Margin="10"

DataContext="{Binding DishTypeListingCrudViewModel}"/>

</TabItem>

<TabItem Header="Блюда">

<views:DishListingCrudView

Margin="10"

DataContext="{Binding DishListingCrudViewModel}"/>

</TabItem>

<TabItem Header="Ингредиенты блюд">

<views:DishIngredientListingCrudView

Margin="10"

DataContext="{Binding DishIngredientListingCrudViewModel}"/>

</TabItem>

<TabItem Header="Приготовления">

<views:DishCookingListingCrudView

Margin="10"

DataContext="{Binding DishCookingListingCrudViewModel}"/>

</TabItem>

</TabControl>

</UserControl>

Б.60 CookView.xaml

<UserControl x:Class="WpfCrud.Views.CookView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

d:DataContext="{d:DesignInstance Type=viewmodels:CookViewModel}"

mc:Ignorable="d"

d:DesignHeight="800" d:DesignWidth="800">

<TabControl>

<TabItem Header="Продукты"

DataContext="{Binding ProductListingViewModel}">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="5\*"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Horizontal"

Margin="10"

Grid.Row="0">

<TextBlock Text="Продукт"/>

<TextBox Text="{Binding ProductNameFilter, UpdateSourceTrigger=PropertyChanged}"

Margin="20 0 0 0"

Width="200"/>

<Button Content="Поиск"

Command="{Binding FilterProductsCommand}"

Margin="20 0 0 0"/>

</StackPanel>

<DataGrid Grid.Row="2"

IsReadOnly="True"

ItemsSource="{Binding FilteredProducts}"

Margin="10"

AutoGenerateColumns="False">

<DataGrid.Columns>

<DataGridTextColumn

Header="Id"

Binding="{Binding Id}"/>

<DataGridTextColumn

Header="Продукт"

Binding="{Binding Name}"/>

<DataGridTextColumn

Header="Калорийность (ккал / 100 г)"

Binding="{Binding CaloricContentPer100Grams}"/>

<DataGridTextColumn

Header="Масса продукта (г)"

Binding="{Binding WeightGrams}"/>

<DataGridTextColumn

Header="Цена за 1 кг (руб.)"

Binding="{Binding PricePerKilogramRoubles}"/>

</DataGrid.Columns>

</DataGrid>

</Grid>

</TabItem>

<TabItem Header="Блюда"

DataContext="{Binding DishListingViewModel}">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="5\*"/>

<RowDefinition Height="5\*"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0"

Margin="10">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<StackPanel Orientation="Vertical"

Margin="10"

Grid.Column="0">

<TextBlock Text="Наименование блюда"/>

<TextBox Text="{Binding DishNameFilter}"/>

</StackPanel>

<Button Grid.Column="1"

Content="Поиск"

Command="{Binding FilterDishesCommand}"

Margin="10"/>

</Grid>

<DataGrid Grid.Row="1"

IsReadOnly="True"

ItemsSource="{Binding FilteredDishes}"

Margin="10"

AutoGenerateColumns="False"

SelectedItem="{Binding SelectedDish}">

<DataGrid.Columns>

<DataGridTextColumn Header="Id"

Binding="{Binding Id}"/>

<DataGridTextColumn Header="Наименование"

Binding="{Binding Name}"/>

<DataGridTextColumn Header="Тип"

Binding="{Binding DishTypeName}"/>

<DataGridTextColumn Header="Время приготовления (минут)"

Binding="{Binding CookingTimeMinutes}"/>

<DataGridTextColumn Header="Масса готового блюда (г)"

Binding="{Binding WeightGrams}"/>

</DataGrid.Columns>

<DataGrid.RowDetailsTemplate>

<DataTemplate>

<DockPanel>

<Image DockPanel.Dock="Left"

Source="{Binding Image}"

Height="100"

Width="100"

Margin="10"/>

<Grid Margin="0 10">

<TextBlock Text="{Binding Recipe}"

TextWrapping="Wrap"/>

</Grid>

</DockPanel>

</DataTemplate>

</DataGrid.RowDetailsTemplate>

</DataGrid>

<Grid Margin="10" Grid.Row="2">

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Text="Ингредиенты" Grid.Row="0"/>

<DataGrid

Grid.Row="1"

ItemsSource="{Binding SelectedDishIngredients}"

Margin="0 10 0 0"

IsReadOnly="True"

AutoGenerateColumns="False">

<DataGrid.Columns>

<DataGridTextColumn Header="Id продукта"

Binding="{Binding ProductId}"/>

<DataGridTextColumn Header="Продукт"

Binding="{Binding ProductName}"/>

<DataGridTextColumn Header="Требуемая масса продукта (г)"

Binding="{Binding RequiredWeightGrams}"/>

</DataGrid.Columns>

</DataGrid>

</Grid>

</Grid>

</TabItem>

<TabItem Header="Приготовления"

DataContext="{Binding DishCookingListingViewModel}">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="5\*"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Horizontal"

Margin="10"

Grid.Row="0">

<TextBlock Text="Блюдо"/>

<TextBox Text="{Binding DishNameFilter}"

Margin="20 0 0 0"

Width="200"/>

<Button Content="Поиск"

Command="{Binding FilterDishCookingsCommand}"

Margin="20 0 0 0"/>

</StackPanel>

<DataGrid ItemsSource="{Binding FilteredDishCookings}"

Margin="10"

Grid.Row="1"

IsReadOnly="True"

AutoGenerateColumns="False">

<DataGrid.Columns>

<DataGridTextColumn Header="Id"

Binding="{Binding Id}"/>

<DataGridTextColumn Header="Id блюда"

Binding="{Binding DishId}"/>

<DataGridTextColumn Header="Блюдо"

Binding="{Binding DishName}"/>

<DataGridTextColumn Header="Количество"

Binding="{Binding Count}"/>

<DataGridTextColumn

Header="Время"

Binding="{Binding CookedAt, StringFormat={}{0:dd.MM.yyyy hh:mm:ss}}"/>

</DataGrid.Columns>

</DataGrid>

</Grid>

</TabItem>

</TabControl>

</UserControl>

Б.61 DishCookingListingCrudView.xaml

<UserControl x:Class="WpfCrud.Views.DishCookingListingCrudView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

d:DataContext="{d:DesignInstance Type=viewmodels:DishCookingListingCrudViewModel}"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="5\*"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Horizontal"

Grid.Row="0">

<Button Content="Загрузить"

Command="{Binding LoadDishCookingsCommand}"/>

<Button Content="Добавить"

Command="{Binding AddDishCookingCommand}"

Margin="20 0 0 0"/>

<Button Content="Изменить"

Command="{Binding EditDishCookingCommand}"

Margin="20 0 0 0"/>

<Button Content="Удалить"

Command="{Binding DeleteDishCookingCommand}"

Margin="20 0 0 0"/>

</StackPanel>

<DataGrid ItemsSource="{Binding DishCookings}"

Margin="0 20 0 0"

Grid.Row="1"

IsReadOnly="True"

AutoGenerateColumns="False"

SelectedItem="{Binding SelectedDishCooking}">

<DataGrid.Columns>

<DataGridTextColumn Header="Id"

Binding="{Binding Id}"/>

<DataGridTextColumn Header="Id блюда"

Binding="{Binding DishId}"/>

<DataGridTextColumn Header="Блюдо"

Binding="{Binding DishName}"/>

<DataGridTextColumn Header="Количество"

Binding="{Binding Count}"/>

<DataGridTextColumn

Header="Время"

Binding="{Binding CookedAt, StringFormat={}{0:dd.MM.yyyy hh:mm:ss}}"/>

</DataGrid.Columns>

</DataGrid>

</Grid>

</UserControl>

Б.62 DishIngredientListingCrudView.xaml

<UserControl x:Class="WpfCrud.Views.DishIngredientListingCrudView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

d:DataContext="{d:DesignInstance Type=viewmodels:DishIngredientListingCrudViewModel}"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="5\*"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Horizontal"

Grid.Row="0">

<Button Content="Загрузить"

Command="{Binding LoadDishIngredientsCommand}"/>

<Button Content="Добавить"

Command="{Binding AddDishIngredientCommand}"

Margin="20 0 0 0"/>

<Button Content="Изменить"

Command="{Binding EditDishIngredientCommand}"

Margin="20 0 0 0"/>

<Button Content="Удалить"

Command="{Binding DeleteDishIngredientCommand}"

Margin="20 0 0 0"/>

</StackPanel>

<DataGrid ItemsSource="{Binding DishIngredients}"

Margin="0 20 0 0"

Grid.Row="1"

IsReadOnly="True"

AutoGenerateColumns="False"

SelectedItem="{Binding SelectedDishIngredient}">

<DataGrid.Columns>

<DataGridTextColumn Header="Id блюда"

Binding="{Binding DishId}"/>

<DataGridTextColumn Header="Блюдо"

Binding="{Binding DishName}"/>

<DataGridTextColumn Header="Тип блюда"

Binding="{Binding DishTypeName}"/>

<DataGridTextColumn Header="Id продукта"

Binding="{Binding ProductId}"/>

<DataGridTextColumn Header="Продукт"

Binding="{Binding ProductName}"/>

<DataGridTextColumn Header="Требуемая масса продукта (г)"

Binding="{Binding RequiredWeightGrams}"/>

</DataGrid.Columns>

</DataGrid>

</Grid>

</UserControl>

Б.63 DishListingCrudView.xaml

<UserControl x:Class="WpfCrud.Views.DishListingCrudView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

d:DataContext="{d:DesignInstance Type=viewmodels:DishListingCrudViewModel}"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="5\*"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Horizontal"

Grid.Row="0">

<Button Content="Загрузить"

Command="{Binding LoadDishesCommand}"/>

<Button Content="Добавить"

Command="{Binding AddDishCommand}"

Margin="20 0 0 0"/>

<Button Content="Изменить"

Command="{Binding EditDishCommand}"

Margin="20 0 0 0"/>

<Button Content="Удалить"

Command="{Binding DeleteDishCommand}"

Margin="20 0 0 0"/>

</StackPanel>

<DataGrid ItemsSource="{Binding Dishes}"

Margin="0 20 0 0"

Grid.Row="1"

IsReadOnly="True"

AutoGenerateColumns="False"

SelectedItem="{Binding SelectedDish}">

<DataGrid.Columns>

<DataGridTextColumn Header="Id"

Binding="{Binding Id}"/>

<DataGridTextColumn Header="Наименование"

Binding="{Binding Name}"/>

<DataGridTextColumn Header="Тип"

Binding="{Binding DishTypeName}"/>

<DataGridTextColumn Header="Время приготовления (минут)"

Binding="{Binding CookingTimeMinutes}"/>

<DataGridTextColumn Header="Масса готового блюда (г)"

Binding="{Binding WeightGrams}"/>

<DataGridTextColumn Header="Калорийность (ккал / 100 г)"

Binding="{Binding CaloricContentPer100Grams}"/>

<DataGridTextColumn Header="Цена блюда (руб.)"

Binding="{Binding DishPriceRoubles}"/>

<DataGridTextColumn Header="Рецепт"

Binding="{Binding Recipe}"/>

<DataGridTemplateColumn Header="Изображение">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Image Source="{Binding Image, Converter={StaticResource NullImageConverter}}"

MaxWidth="150" MaxHeight="150"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

</Grid>

</UserControl>

Б.64 DishTypeListingCrudView.xaml

<UserControl x:Class="WpfCrud.Views.DishTypeListingCrudView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

d:DataContext="{d:DesignInstance Type=viewmodels:DishTypeListingCrudViewModel}"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="5\*"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Horizontal"

Grid.Row="0">

<Button Content="Загрузить"

Command="{Binding LoadDishTypesCommand}"/>

<Button Content="Добавить"

Command="{Binding AddDishTypeCommand}"

Margin="20 0 0 0"/>

<Button Content="Изменить"

Command="{Binding EditDishTypeCommand}"

Margin="20 0 0 0"/>

<Button Content="Удалить"

Command="{Binding DeleteDishTypeCommand}"

Margin="20 0 0 0"/>

</StackPanel>

<DataGrid ItemsSource="{Binding DishTypes}"

Margin="0 20 0 0"

Grid.Row="1"

IsReadOnly="True"

AutoGenerateColumns="False"

SelectedItem="{Binding SelectedDishType}">

<DataGrid.Columns>

<DataGridTextColumn Header="Id"

Binding="{Binding Id}"/>

<DataGridTextColumn Header="Наименование"

Binding="{Binding Name}"/>

</DataGrid.Columns>

</DataGrid>

</Grid>

</UserControl>

Б.65 ProductListingCrudView.xaml

<UserControl x:Class="WpfCrud.Views.ProductListingCrudView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

d:DataContext="{d:DesignInstance Type=viewmodels:ProductListingCrudViewModel}"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="5\*"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Horizontal"

Grid.Row="0">

<Button Content="Загрузить"

Command="{Binding LoadProductsCommand}"/>

<Button Content="Добавить"

Command="{Binding AddProductCommand}"

Margin="20 0 0 0"/>

<Button Content="Изменить"

Command="{Binding EditProductCommand}"

Margin="20 0 0 0"/>

<Button Content="Удалить"

Command="{Binding DeleteProductCommand}"

Margin="20 0 0 0"/>

</StackPanel>

<DataGrid ItemsSource="{Binding Products}"

Margin="0 20 0 0"

Grid.Row="1"

IsReadOnly="True"

AutoGenerateColumns="False"

SelectedItem="{Binding SelectedProduct}">

<DataGrid.Columns>

<DataGridTextColumn Header="Id"

Binding="{Binding Id}"/>

<DataGridTextColumn Header="Наименование"

Binding="{Binding Name}"/>

<DataGridTextColumn Header="Калорийность (ккал/100 г)"

Binding="{Binding CaloricContentPer100Grams}"/>

<DataGridTextColumn Header="Масса (г)"

Binding="{Binding WeightGrams}"/>

<DataGridTextColumn Header="Цена за 1 кг (руб.)"

Binding="{Binding PricePerKilogramRoubles}"/>

</DataGrid.Columns>

</DataGrid>

</Grid>

</UserControl>

Б.66 UserAccountListingCrudView.xaml

<UserControl x:Class="WpfCrud.Views.UserAccountListingCrudView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

d:DataContext="{d:DesignInstance Type=viewmodels:UserAccountListingCrudViewModel}"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="5\*"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Horizontal"

Grid.Row="0">

<Button Content="Загрузить"

Command="{Binding LoadUserAccountsCommand}"/>

<Button Content="Добавить"

Command="{Binding AddUserAccountCommand}"

Margin="20 0 0 0"/>

<Button Content="Изменить"

Command="{Binding EditUserAccountCommand}"

Margin="20 0 0 0"/>

<Button Content="Удалить"

Command="{Binding DeleteUserAccountCommand}"

Margin="20 0 0 0"/>

</StackPanel>

<DataGrid ItemsSource="{Binding UserAccounts}"

Margin="0 20 0 0"

Grid.Row="1"

IsReadOnly="True"

AutoGenerateColumns="False"

SelectedItem="{Binding SelectedUserAccount}">

<DataGrid.Columns>

<DataGridTextColumn Header="Id"

Binding="{Binding Id}"/>

<DataGridTextColumn Header="Роль"

Binding="{Binding Role}"/>

<DataGridTextColumn Header="Логин"

Binding="{Binding Login}"/>

<DataGridTemplateColumn Header="Изображение">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Image Source="{Binding Image, Converter={StaticResource NullImageConverter}}"

MaxWidth="150" MaxHeight="150"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

</Grid>

</UserControl>

Б.67 AddDishIngredientWindow.xaml

<Window x:Class="WpfCrud.AddDishIngredientWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

d:DataContext="{d:DesignInstance Type=viewmodels:AddDishIngredientViewModel}"

mc:Ignorable="d"

Title="Добавление ингредиента" Height="450" Width="800"

Closed="Window\_Closed"

WindowStartupLocation="CenterScreen">

<Grid Margin="10">

<Grid.RowDefinitions>

<RowDefinition Height="5\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Vertical" Margin="10"

Grid.Row="0">

<TextBlock Text="Блюдо"/>

<ComboBox ItemsSource="{Binding Dishes}"

SelectedValue="{Binding DishId}"

SelectedValuePath="Id">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Name}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<TextBlock Text="Продукт" Margin="0 10 0 0"/>

<ComboBox ItemsSource="{Binding Products}"

SelectedValue="{Binding ProductId}"

SelectedValuePath="Id">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Name}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<TextBlock Text="Требуемая масса продукта (г)"

Margin="0 10 0 0"/>

<TextBox Text="{Binding RequiredWeightGrams}"/>

</StackPanel>

<StackPanel Orientation="Horizontal"

Grid.Row="1" Margin="0 10 0 0"

HorizontalAlignment="Center">

<Button Content="Подтвердить"

Command="{Binding SubmitCommand}">

</Button>

<Button Content="Отменить"

Name="BtnCancel"

Margin="20 0 0 0"

Command="{Binding CancelCommand}"/>

</StackPanel>

</Grid>

</Window>

Б.68 AddDishIngredientWindow.xaml.cs

using System;

using System.Windows;

using WpfCrud.ViewModels;

namespace WpfCrud

{

public partial class AddDishIngredientWindow : Window

{

private readonly AddDishIngredientViewModel \_editDishIngredientViewModel;

public AddDishIngredientWindow(AddDishIngredientViewModel editDishIngredientViewModel)

{

\_editDishIngredientViewModel = editDishIngredientViewModel ?? throw new ArgumentNullException(nameof(editDishIngredientViewModel));

DataContext = editDishIngredientViewModel;

\_editDishIngredientViewModel.SubmitError += OnSubmitError;

\_editDishIngredientViewModel.SubmitSuccess += OnSubmitSuccess;

\_editDishIngredientViewModel.CancelSubmit += OnCancelSubmit;

InitializeComponent();

}

private void OnCancelSubmit()

{

DialogResult = false;

Close();

}

private void OnSubmitSuccess()

{

DialogResult = true;

Close();

}

private void OnSubmitError(Exception obj)

{

MessageBox.Show(obj.Message, obj.GetType().Name, MessageBoxButton.OK, MessageBoxImage.Error);

}

private void Window\_Closed(object sender, EventArgs e)

{

\_editDishIngredientViewModel.SubmitError -= OnSubmitError;

\_editDishIngredientViewModel.SubmitSuccess -= OnSubmitSuccess;

\_editDishIngredientViewModel.CancelSubmit -= OnCancelSubmit;

}

}

}

Б.69 App.xaml

<Application x:Class="WpfCrud.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:converters="clr-namespace:WpfCrud.Converters">

<Application.Resources>

<Style TargetType="Button">

<Setter Property="FontSize" Value="14"/>

<Setter Property="Padding" Value="5"/>

</Style>

<Style TargetType="TextBox">

<Setter Property="FontSize" Value="14"/>

<Setter Property="Padding" Value="5"/>

</Style>

<Style TargetType="PasswordBox">

<Setter Property="FontSize" Value="14"/>

<Setter Property="Padding" Value="5"/>

</Style>

<Style TargetType="TextBlock">

<Setter Property="FontSize" Value="14"/>

<Setter Property="Padding" Value="2"/>

</Style>

<converters:NullImageConverter x:Key="NullImageConverter"/>

</Application.Resources>

</Application>

Б.70 App.xaml.cs

using System.Windows;

using WpfCrud.ViewModels;

namespace WpfCrud

{

public partial class App : Application

{

public App()

{

}

protected override void OnStartup(StartupEventArgs e)

{

base.OnStartup(e);

var loginWindow = new LoginWindow(new LoginViewModel());

loginWindow.Show();

}

}

}

Б.71 EditDishCookingWindow.xaml

<Window x:Class="WpfCrud.EditDishCookingWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

d:DataContext="{d:DesignInstance Type=viewmodels:EditDishCookingViewModel}"

mc:Ignorable="d"

Title="Редактирование приготовления"

Height="400"

MaxHeight="500"

Width="450"

MaxWidth="500"

WindowStartupLocation="CenterScreen"

Closed="Window\_Closed">

<Grid Margin="10">

<Grid.RowDefinitions>

<RowDefinition Height="5\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Vertical"

Margin="10">

<TextBlock Text="Блюдо"/>

<ComboBox ItemsSource="{Binding Dishes}"

SelectedValue="{Binding DishId}"

SelectedValuePath="Id">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Name}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<TextBlock Text="Количество блюд" Margin="0 10 0 0"/>

<TextBox Text="{Binding Count}"/>

<TextBlock Text="Дата приготовления" Margin="0 10 0 0"/>

<DatePicker SelectedDate="{Binding CookedAt}"/>

</StackPanel>

<StackPanel Orientation="Horizontal"

Margin="0 10 0 0"

HorizontalAlignment="Center"

Grid.Row="1">

<Button Name="BtnSubmit" Content="Подтвердить"

Command="{Binding SubmitCommand}"/>

<Button Name="BtnCancel" Content="Отменить"

Command="{Binding CancelCommand}"

Margin="20 0 0 0"/>

</StackPanel>

</Grid>

</Window>

Б.72 EditDishCookingWindow.xaml.cs

using System;

using System.Windows;

using WpfCrud.ViewModels;

namespace WpfCrud

{

public partial class EditDishCookingWindow : Window

{

private readonly EditDishCookingViewModel \_editDishCookingViewModel;

public EditDishCookingWindow(EditDishCookingViewModel editDishCookingViewModel)

{

\_editDishCookingViewModel = editDishCookingViewModel ?? throw new ArgumentNullException(nameof(editDishCookingViewModel));

DataContext = \_editDishCookingViewModel;

\_editDishCookingViewModel.SubmitError += OnSubmitError;

\_editDishCookingViewModel.SubmitSuccess += OnSubmitSuccess;

\_editDishCookingViewModel.CancelSubmit += OnCancelSubmit;

InitializeComponent();

}

private void OnCancelSubmit()

{

DialogResult = false;

Close();

}

private void OnSubmitSuccess()

{

DialogResult = true;

Close();

}

private void OnSubmitError(Exception e)

{

MessageBox.Show(e.Message, e.GetType().Name, MessageBoxButton.OK, MessageBoxImage.Error);

}

private void Window\_Closed(object sender, EventArgs e)

{

\_editDishCookingViewModel.SubmitError -= OnSubmitError;

\_editDishCookingViewModel.SubmitSuccess -= OnSubmitSuccess;

\_editDishCookingViewModel.CancelSubmit -= OnCancelSubmit;

}

}

}

Б.73 EditDishTypeWindow.xaml

<Window x:Class="WpfCrud.EditDishTypeWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

d:DataContext="{d:DesignInstance Type=viewmodels:EditDishTypeViewModel}"

mc:Ignorable="d"

Title="Редактирование типа блюда"

Height="400"

MaxHeight="500"

Width="400"

MaxWidth="500"

Closed="Window\_Closed"

WindowStartupLocation="CenterScreen">

<Grid Margin="10">

<Grid.RowDefinitions>

<RowDefinition Height="5\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<StackPanel Grid.Row="0"

Orientation="Vertical"

Margin="10">

<TextBlock Text="Название"/>

<TextBox Text="{Binding Name}"/>

</StackPanel>

<StackPanel Orientation="Horizontal"

Margin="0 10 0 0"

HorizontalAlignment="Center"

Grid.Row="1">

<Button Name="BtnSubmit"

Content="Подтвердить"

Command="{Binding SubmitCommand}"/>

<Button Name="BtnCancel"

Content="Отменить"

Command="{Binding CancelCommand}"

Margin="20 0 0 0"/>

</StackPanel>

</Grid>

</Window>

Б.74 EditDishTypeWindow.xaml.cs

using System;

using System.Windows;

using WpfCrud.ViewModels;

namespace WpfCrud

{

public partial class EditDishTypeWindow : Window

{

private readonly EditDishTypeViewModel \_editDishTypeViewModel;

public EditDishTypeWindow(EditDishTypeViewModel editDishTypeViewModel)

{

\_editDishTypeViewModel = editDishTypeViewModel ?? throw new ArgumentNullException(nameof(editDishTypeViewModel));

DataContext = \_editDishTypeViewModel;

\_editDishTypeViewModel.SubmitError += OnSubmitError;

\_editDishTypeViewModel.SubmitSuccess += OnSubmitSuccess;

\_editDishTypeViewModel.CancelSubmit += OnCancelSubmit;

InitializeComponent();

}

private void OnCancelSubmit()

{

DialogResult = false;

Close();

}

private void OnSubmitSuccess()

{

DialogResult = true;

Close();

}

private void OnSubmitError(Exception e)

{

MessageBox.Show(e.Message, e.GetType().Name, MessageBoxButton.OK, MessageBoxImage.Error);

}

private void Window\_Closed(object sender, EventArgs e)

{

\_editDishTypeViewModel.SubmitError -= OnSubmitError;

\_editDishTypeViewModel.SubmitSuccess -= OnSubmitSuccess;

\_editDishTypeViewModel.CancelSubmit -= OnCancelSubmit;

}

}

}

Б.75 EditDishWindow.xaml

<Window x:Class="WpfCrud.EditDishWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

d:DataContext="{d:DesignInstance Type=viewmodels:EditDishViewModel}"

mc:Ignorable="d"

Title="Окно редактирования блюда" Height="600" Width="600"

Closed="Window\_Closed"

WindowStartupLocation="CenterScreen">

<Grid Margin="10">

<Grid.RowDefinitions>

<RowDefinition Height="5\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<ScrollViewer Grid.Row="0">

<StackPanel Orientation="Vertical" Margin="10">

<TextBlock Text="Название"/>

<TextBox Text="{Binding Name}"/>

<TextBlock Text="Тип" Margin="0 10 0 0"/>

<ComboBox ItemsSource="{Binding DishTypes}"

SelectedValue="{Binding DishTypeId}"

SelectedValuePath="Id">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Name}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<TextBlock Text="Время готовки (минут)" Margin="0 10 0 0"/>

<TextBox Text="{Binding CookingTimeMinutes}"/>

<TextBlock Text="Масса готового блюда (г)"

Margin="0 10 0 0"/>

<TextBox Text="{Binding WeightGrams}"/>

<TextBlock Text="Рецепт" Margin="0 10 0 0"/>

<TextBox Text="{Binding Recipe}" AcceptsReturn="True"

AcceptsTab="True" Height="300"/>

<StackPanel Orientation="Horizontal"

Margin="0 10 0 0">

<Button Content="Выбрать изображение"

Command="{Binding SelectDishImageCommand}"/>

<Button Content="Убрать изображение"

Command="{Binding DeselectDishImageCommand}"

Margin="20 0 0 0"/>

</StackPanel>

<Image Source="{Binding Image, Converter={StaticResource NullImageConverter}}"

Margin="0 5 0 0"

MaxWidth="500" MaxHeight="500"/>

</StackPanel>

</ScrollViewer>

<StackPanel Orientation="Horizontal"

Grid.Row="1" Margin="0 10 0 0"

HorizontalAlignment="Center">

<Button Content="Подтвердить"

Command="{Binding SubmitCommand}">

</Button>

<Button Content="Отменить"

Name="BtnCancel"

Margin="20 0 0 0"

Command="{Binding CancelCommand}"/>

</StackPanel>

</Grid>

</Window>

Б.76 EditDishWindow.xaml.cs

using System;

using System.Windows;

using WpfCrud.ViewModels;

namespace WpfCrud

{

public partial class EditDishWindow : Window

{

private readonly EditDishViewModel \_editDishViewModel;

public EditDishWindow(EditDishViewModel editDishViewModel)

{

\_editDishViewModel = editDishViewModel ?? throw new ArgumentNullException(nameof(editDishViewModel));

DataContext = \_editDishViewModel;

\_editDishViewModel.SubmitError += OnSubmitError;

\_editDishViewModel.SubmitSuccess += OnSubmitSuccess;

\_editDishViewModel.CancelSubmit += OnCancelSubmit;

InitializeComponent();

}

private void OnCancelSubmit()

{

DialogResult = false;

Close();

}

private void OnSubmitSuccess()

{

DialogResult = true;

Close();

}

private void OnSubmitError(Exception obj)

{

MessageBox.Show(obj.Message, obj.GetType().Name, MessageBoxButton.OK, MessageBoxImage.Error);

}

private void Window\_Closed(object sender, EventArgs e)

{

\_editDishViewModel.SubmitError -= OnSubmitError;

\_editDishViewModel.SubmitSuccess -= OnSubmitSuccess;

\_editDishViewModel.CancelSubmit -= OnCancelSubmit;

}

}

}

Б.77 EditExistingDishIngredientWindow.xaml

<Window x:Class="WpfCrud.EditExistingDishIngredientWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

d:DataContext="{d:DesignInstance Type=viewmodels:EditExistingDishIngredientViewModel}"

mc:Ignorable="d"

Title="Редактирование ингредиента"

Height="400" Width="400"

Closed="Window\_Closed"

WindowStartupLocation="CenterScreen">

<Grid Margin="10">

<Grid.RowDefinitions>

<RowDefinition Height="5\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Vertical"

Margin="10">

<TextBlock Text="Требуемая масса продукта (г)"/>

<TextBox Text="{Binding RequiredWeightGrams}"/>

</StackPanel>

<StackPanel Orientation="Horizontal"

Grid.Row="1" Margin="0 10 0 0"

HorizontalAlignment="Center">

<Button Content="Подтвердить"

Command="{Binding SubmitCommand}"/>

<Button Content="Отменить"

Command="{Binding CancelCommand}"

Margin="20 0 0 0"/>

</StackPanel>

</Grid>

</Window>

Б.78 EditExistingDishIngredientWindow.xaml.cs

using System;

using System.Windows;

using WpfCrud.ViewModels;

namespace WpfCrud

{

public partial class EditExistingDishIngredientWindow : Window

{

private readonly EditExistingDishIngredientViewModel \_editExistingDishIngredientViewModel;

public EditExistingDishIngredientWindow(EditExistingDishIngredientViewModel editExistingDishIngredientViewModel)

{

\_editExistingDishIngredientViewModel = editExistingDishIngredientViewModel ?? throw new System.ArgumentNullException(nameof(editExistingDishIngredientViewModel));

DataContext = \_editExistingDishIngredientViewModel;

\_editExistingDishIngredientViewModel.SubmitError += OnSubmitError;

\_editExistingDishIngredientViewModel.SubmitSuccess += OnSubmitSuccess;

\_editExistingDishIngredientViewModel.CancelSubmit += OnCancelSubmit;

InitializeComponent();

}

private void OnCancelSubmit()

{

DialogResult = false;

Close();

}

private void OnSubmitSuccess()

{

DialogResult = true;

Close();

}

private void OnSubmitError(Exception e)

{

MessageBox.Show(e.Message, e.GetType().Name, MessageBoxButton.OK, MessageBoxImage.Error);

}

private void Window\_Closed(object sender, EventArgs e)

{

\_editExistingDishIngredientViewModel.SubmitError -= OnSubmitError;

\_editExistingDishIngredientViewModel.SubmitSuccess -= OnSubmitSuccess;

\_editExistingDishIngredientViewModel.CancelSubmit -= OnCancelSubmit;

}

}

}

Б.79 EditProductWindow.xaml

<Window x:Class="WpfCrud.EditProductWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

d:DataContext="{d:DesignInstance Type=viewmodels:EditProductViewModel}"

mc:Ignorable="d"

Title="Редактирование продукта" Height="450" Width="800"

Closed="Window\_Closed"

WindowStartupLocation="CenterScreen">

<Grid Margin="10">

<Grid.RowDefinitions>

<RowDefinition Height="5\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Vertical" Margin="10"

Grid.Row="0">

<TextBlock Text="Наименование"/>

<TextBox Text="{Binding Name}"/>

<TextBlock Text="Калорийность (ккал / 100 г)"

Margin="0 10 0 0"/>

<TextBox Text="{Binding CaloricContentPer100Grams}"/>

<TextBlock Text="Масса продукта (г)"

Margin="0 10 0 0"/>

<TextBox Text="{Binding WeightGrams}"/>

<TextBlock Text="Цена за 1 кг (руб.)"

Margin="0 10 0 0"/>

<TextBox Text="{Binding PricePerKilogramRoubles}"/>

</StackPanel>

<StackPanel Orientation="Horizontal"

Grid.Row="1" Margin="0 10 0 0"

HorizontalAlignment="Center">

<Button Content="Подтвердить"

Command="{Binding SubmitCommand}">

</Button>

<Button Content="Отменить"

Name="BtnCancel"

Margin="20 0 0 0"

Command="{Binding CancelCommand}"/>

</StackPanel>

</Grid>

</Window>

Б.80 EditProductWindow.xaml.cs

using System;

using System.Windows;

using WpfCrud.ViewModels;

namespace WpfCrud

{

public partial class EditProductWindow : Window

{

private readonly EditProductViewModel \_editProductViewModel;

public EditProductWindow(EditProductViewModel editProductViewModel)

{

\_editProductViewModel = editProductViewModel ?? throw new ArgumentNullException(nameof(editProductViewModel));

DataContext = \_editProductViewModel;

\_editProductViewModel.SubmitError += OnSubmitError;

\_editProductViewModel.SubmitSuccess += OnSubmitSuccess;

\_editProductViewModel.CancelSubmit += OnCancelSubmit;

InitializeComponent();

}

private void OnCancelSubmit()

{

DialogResult = false;

Close();

}

private void OnSubmitSuccess()

{

DialogResult = true;

Close();

}

private void OnSubmitError(Exception obj)

{

MessageBox.Show(obj.Message, obj.GetType().Name, MessageBoxButton.OK, MessageBoxImage.Error);

}

private void Window\_Closed(object sender, EventArgs e)

{

\_editProductViewModel.SubmitError -= OnSubmitError;

\_editProductViewModel.SubmitSuccess -= OnSubmitSuccess;

\_editProductViewModel.CancelSubmit -= OnCancelSubmit;

}

}

}

Б.81 EditUserAccountWindow.xaml

<Window x:Class="WpfCrud.EditUserAccountWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

d:DataContext="{d:DesignInstance Type=viewmodels:EditUserAccountViewModel}"

mc:Ignorable="d"

Title="EditUserAccountWindow" Height="450" Width="800"

WindowStartupLocation="CenterScreen"

Closed="Window\_Closed">

<Grid Margin="10">

<Grid.RowDefinitions>

<RowDefinition Height="5\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<ScrollViewer Grid.Row="0"

Margin="10">

<StackPanel Orientation="Vertical">

<TextBlock Text="Логин"/>

<TextBox Text="{Binding Login, UpdateSourceTrigger=PropertyChanged}"

MaxLength="100"/>

<TextBlock Text="Пароль" Margin="0 10 0 0"/>

<TextBox Text="{Binding Password, UpdateSourceTrigger=PropertyChanged}"

MaxLength="100"/>

<TextBlock Text="Роль в системе" Margin="0 10 0 0"/>

<ComboBox ItemsSource="{Binding UserRoles}"

SelectedItem="{Binding UserRole, UpdateSourceTrigger=PropertyChanged}"/>

<StackPanel Orientation="Horizontal"

Margin=" 0 10 0 0"

HorizontalAlignment="Center">

<Button Content="Выбрать изображение"

Command="{Binding SelectImageCommand}"/>

<Button Content="Удалить изображение"

Command="{Binding DeselectImageCommand}"

Margin="20 0 0 0"/>

</StackPanel>

<Image Source="{Binding Image, Converter={StaticResource NullImageConverter}}" Margin="0 10 0 0"

MaxHeight="500" MaxWidth="500"/>

</StackPanel>

</ScrollViewer>

<StackPanel Orientation="Horizontal" Margin="0 10 0 0"

Grid.Row="1" HorizontalAlignment="Center">

<Button Content="Подтвердить"

Command="{Binding SubmitCommand}"/>

<Button Content="Отменить"

Command="{Binding CancelCommand}"

Margin="20 0 0 0"/>

</StackPanel>

</Grid>

</Window>

Б.82 EditUserAccountWindow.xaml.cs

using System;

using System.Windows;

using WpfCrud.ViewModels;

namespace WpfCrud

{

public partial class EditUserAccountWindow : Window

{

private readonly EditUserAccountViewModel \_editUserAccountViewModel;

public EditUserAccountWindow(EditUserAccountViewModel editUserAccountViewModel)

{

\_editUserAccountViewModel = editUserAccountViewModel ?? throw new ArgumentNullException(nameof(editUserAccountViewModel));

DataContext = \_editUserAccountViewModel;

\_editUserAccountViewModel.SubmitError += OnSubmitError;

\_editUserAccountViewModel.SubmitSuccess += OnSubmitSuccess;

\_editUserAccountViewModel.CancelSubmit += OnCancelSubmit;

InitializeComponent();

}

private void OnCancelSubmit()

{

DialogResult = false;

Close();

}

private void OnSubmitSuccess()

{

DialogResult = true;

Close();

}

private void OnSubmitError(Exception e)

{

MessageBox.Show(e.Message, e.GetType().Name, MessageBoxButton.OK, MessageBoxImage.Error);

}

private void Window\_Closed(object sender, EventArgs e)

{

\_editUserAccountViewModel.SubmitError -= OnSubmitError;

\_editUserAccountViewModel.SubmitSuccess -= OnSubmitSuccess;

\_editUserAccountViewModel.CancelSubmit -= OnCancelSubmit;

}

}

}

Б.83 LoginWindow.xaml

<Window x:Class="WpfCrud.LoginWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

d:DataContext="{d:DesignInstance Type=viewmodels:LoginViewModel}"

mc:Ignorable="d"

WindowStartupLocation="CenterScreen"

Title="Авторизация" Height="700" Width="450"

Closing="Window\_Closing">

<StackPanel Margin="10">

<TextBlock Text="Логин" Margin="10 10 10 0"/>

<TextBox Margin="10 0" Text="{Binding Login, UpdateSourceTrigger=PropertyChanged}"/>

<TextBlock Text="Пароль" Margin="10 20 10 0"/>

<PasswordBox Margin="10 0"

PasswordChanged="PwdBox\_PasswordChanged"/>

<Image Margin="0 20 0 0"

HorizontalAlignment="Center"

Width="300" Height="300"

Source="/Images/Captcha1337.jpeg"/>

<TextBlock Text="Текст с картинки"

Margin="10 20 10 0"/>

<TextBox Text="{Binding Captcha, UpdateSourceTrigger=PropertyChanged}"

Margin="10 0"/>

<Button Content="Вход"

HorizontalAlignment="Center"

Command="{Binding LoginCommand}"

Margin="0 20 0 0"

Width="100" Height="50"

IsEnabled="{Binding CanSubmit, Mode=OneWay}"/>

</StackPanel>

</Window>

Б.84 LoginWindow.xaml.cs

using System;

using System.Windows;

using System.Windows.Controls;

using WpfCrud.Models;

using WpfCrud.ViewModels;

namespace WpfCrud

{

public partial class LoginWindow : Window

{

private readonly LoginViewModel \_loginViewModel;

private MainWindow \_mainWindow;

public LoginWindow(LoginViewModel loginViewModel)

{

\_loginViewModel = loginViewModel ?? throw new ArgumentNullException(nameof(loginViewModel));

\_loginViewModel.LoginSuccess += OnLoginSuccess;

\_loginViewModel.LoginFailure += OnLoginFailure;

DataContext = loginViewModel;

InitializeComponent();

}

private void OnLoginFailure(string obj)

{

MessageBox.Show(obj, "Ошибка авторизации", MessageBoxButton.OK, MessageBoxImage.Error);

}

private void OnLoginSuccess()

{

MessageBox.Show("Добро пожаловать", "Авторизация пройдена", MessageBoxButton.OK, MessageBoxImage.Information);

switch (\_loginViewModel.CurrentUserRole)

{

case Models.Enums.UserRoleEnum.Admin:

var currentUser = new CurrentUser

{

Id = \_loginViewModel.CurrentUserId,

Login = \_loginViewModel.Login,

Role = \_loginViewModel.CurrentUserRole,

};

ShowMainWindow(new AdminViewModel(currentUser));

break;

case Models.Enums.UserRoleEnum.Chef:

ShowMainWindow(new ChefViewModel());

break;

case Models.Enums.UserRoleEnum.Cook:

ShowMainWindow(new CookViewModel());

break;

default:

Close();

return;

}

Hide();

}

private void ShowMainWindow(ViewModelBase currentViewModel)

{

var mainViewModel = new MainViewModel(currentViewModel);

\_mainWindow = new MainWindow(mainViewModel)

{

Owner = this,

WindowStartupLocation = WindowStartupLocation.CenterOwner

};

\_mainWindow.Closed += OnMainWindowClosed;

\_mainWindow.Show();

}

private void OnMainWindowClosed(object sender, EventArgs e)

{

\_mainWindow.Closed -= OnMainWindowClosed;

Show();

}

private void PwdBox\_PasswordChanged(object sender, RoutedEventArgs e)

{

\_loginViewModel.Password = (sender as PasswordBox).Password;

}

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

\_loginViewModel.LoginSuccess -= OnLoginSuccess;

\_loginViewModel.LoginFailure -= OnLoginFailure;

}

}

}

Б.85 MainWindow.xaml

<Window x:Class="WpfCrud.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:viewmodels="clr-namespace:WpfCrud.ViewModels"

xmlns:views="clr-namespace:WpfCrud.Views"

d:DataContext="{d:DesignInstance Type=viewmodels:MainViewModel}"

mc:Ignorable="d"

Title="Главное окно" Height="600" Width="800">

<Window.Resources>

<DataTemplate DataType="{x:Type viewmodels:AdminViewModel}">

<views:AdminView/>

</DataTemplate>

<DataTemplate DataType="{x:Type viewmodels:ChefViewModel}">

<views:ChefView/>

</DataTemplate>

<DataTemplate DataType="{x:Type viewmodels:CookViewModel}">

<views:CookView/>

</DataTemplate>

</Window.Resources>

<Grid Margin="10">

<ContentControl Content="{Binding CurrentViewModel}"/>

</Grid>

</Window>

Б.86 MainWindow.xaml.cs

using System;

using System.Windows;

namespace WpfCrud

{

public partial class MainWindow : Window

{

public MainWindow(ViewModels.MainViewModel mainViewModel)

{

if (mainViewModel is null)

{

throw new ArgumentNullException(nameof(mainViewModel));

}

InitializeComponent();

DataContext = mainViewModel;

}

}

}

## Приложение В

Модульное тестирование

В.1 EditDishViewModelUnitTests.cs

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Collections.Generic;

using System.Linq;

using WpfCrud.Models;

using WpfCrud.ViewModels;

namespace WpfCrud.Tests

{

[TestClass]

public class EditDishViewModelUnitTests

{

private static readonly List<DishType> s\_dishTypes = new List<DishType>

{

new DishType(1, "A"),

new DishType(2, "B"),

new DishType(3, "C"),

new DishType(4, "D"),

new DishType(5, "E"),

};

[TestMethod]

public void AssertConstructorThrowsWhenEditableDishParamIsNull()

{

EditableDish editableDish = null;

Assert

.ThrowsException<ArgumentNullException>(

() => new EditDishViewModel(editableDish, s\_dishTypes));

}

[TestMethod]

public void AssertConstructorThrowsWhenDishTypesParamIsNull()

{

var editableDish = new EditableDish();

List<DishType> dishTypes = null;

Assert.ThrowsException<ArgumentNullException>(

() => new EditDishViewModel(editableDish, dishTypes));

}

[TestMethod]

public void AssertConstructorThrowsWhenDishTypesParamIsEmpty()

{

const string expectedErrorMessage = "Список типов блюд <dishTypes> пуст!";

var et = new EditableDish();

var dishTypes = Enumerable.Empty<DishType>();

var e = Assert.ThrowsException<InvalidOperationException>(

() => new EditDishViewModel(et, dishTypes));

Assert.AreEqual(expectedErrorMessage, e.Message);

}

[TestMethod]

public void AssertValidatePropertiesThrowsWhenNameIsNullOrWhiteSpace()

{

const string expectedErrorMessage = "Название блюда должно быть заполнено!";

var et = new EditableDish();

var etvm = new EditDishViewModel(et, s\_dishTypes);

Action validateCall = () => etvm.ValidateProperties();

Exception e = null;

etvm.Name = null;

e = Assert.ThrowsException<Exception>(validateCall);

Assert.AreEqual(expectedErrorMessage, e.Message);;

etvm.Name = string.Empty;

e = Assert.ThrowsException<Exception>(validateCall);

Assert.AreEqual(expectedErrorMessage, e.Message);;

etvm.Name = " ";

e = Assert.ThrowsException<Exception>(validateCall);

Assert.AreEqual(expectedErrorMessage, e.Message);

}

[TestMethod]

public void AssertValidatePropertiesThrowsWhenDishTypeIsNotSet()

{

const string expectedErrorMessage = "Тип блюда должен быть выбран!";

var et = new EditableDish();

var etvm = new EditDishViewModel(et, s\_dishTypes);

etvm.Name = "TEST";

Action validateCall = () => etvm.ValidateProperties();

Exception e = null;

etvm.DishTypeId = 0;

e = Assert.ThrowsException<Exception>(validateCall);

Assert.AreEqual(expectedErrorMessage, e.Message);

}

}

}

В.2 EditProductViewModelUnitTests.cs

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using WpfCrud.Models;

using WpfCrud.ViewModels;

namespace WpfCrud.Tests

{

[TestClass]

public class EditProductViewModelUnitTests

{

[TestMethod]

public void AssertConstructorThrowsWhenEditProductViewModelParamIsNull()

{

Assert.ThrowsException<ArgumentNullException>(() => new EditProductViewModel(null));

}

[TestMethod]

public void AssertValidatePropertiesThrowsWhenNamePropertyIsNullOrWhiteSpace()

{

const string expectedErrorMessage = "Наименование должно быть заполнено!";

var editableProduct = new EditableProduct();

var editProductVm = new EditProductViewModel(editableProduct);

Action validatePropertiesMethodCall = () => editProductVm.ValidateProperties();

editProductVm.Name = null;

var e = Assert.ThrowsException<Exception>(validatePropertiesMethodCall);

Assert.AreEqual(expectedErrorMessage, e.Message);

editProductVm.Name = string.Empty;

e = Assert.ThrowsException<Exception>(validatePropertiesMethodCall);

Assert.AreEqual(expectedErrorMessage, e.Message);

editProductVm.Name = " ";

e = Assert.ThrowsException<Exception>(validatePropertiesMethodCall);

Assert.AreEqual(expectedErrorMessage, e.Message);

}

[TestMethod]

public void AssertValidatePropertiesThrowsWhenCaloricContentPer100GramsIsNotGreaterThanZero()

{

const string expectedErrorMessage = "Калорийность (ккал / 100 г) должна быть положительным числом!";

var editableProduct = new EditableProduct();

var editProductVm = new EditProductViewModel(editableProduct);

editProductVm.Name = "TEST";

Action validatePropertiesMethodCall = () => editProductVm.ValidateProperties();

editProductVm.CaloricContentPer100Grams = 0;

var e = Assert.ThrowsException<Exception>(validatePropertiesMethodCall);

Assert.AreEqual(expectedErrorMessage, e.Message);

editProductVm.CaloricContentPer100Grams = -1;

e = Assert.ThrowsException<Exception>(validatePropertiesMethodCall);

Assert.AreEqual(expectedErrorMessage, e.Message);

editProductVm.CaloricContentPer100Grams = double.NegativeInfinity;

e = Assert.ThrowsException<Exception>(validatePropertiesMethodCall);

Assert.AreEqual(expectedErrorMessage, e.Message);

editProductVm.CaloricContentPer100Grams = double.NaN;

e = Assert.ThrowsException<Exception>(validatePropertiesMethodCall);

Assert.AreEqual(expectedErrorMessage, e.Message);

}

[TestMethod]

public void AssertValidatePropertiesThrowsWhenWeightGramsIsNotGreaterThanZero()

{

const string expectedErrorMessage = "Масса (г) должна быть положительным числом!";

var editableProduct = new EditableProduct();

var editProductVm = new EditProductViewModel(editableProduct);

editProductVm.Name = "TEST";

editProductVm.CaloricContentPer100Grams = 1;

Action validatePropertiesMethodCall = () => editProductVm.ValidateProperties();

editProductVm.WeightGrams = 0;

var e = Assert.ThrowsException<Exception>(validatePropertiesMethodCall);

Assert.AreEqual(expectedErrorMessage, e.Message);

editProductVm.WeightGrams = -1;

e = Assert.ThrowsException<Exception>(validatePropertiesMethodCall);

Assert.AreEqual(expectedErrorMessage, e.Message);

editProductVm.WeightGrams = double.NegativeInfinity;

e = Assert.ThrowsException<Exception>(validatePropertiesMethodCall);

Assert.AreEqual(expectedErrorMessage, e.Message);

editProductVm.WeightGrams = double.NaN;

e = Assert.ThrowsException<Exception>(validatePropertiesMethodCall);

Assert.AreEqual(expectedErrorMessage, e.Message);

}

[TestMethod]

public void AssertValidatePropertiesThrowsWhenPricePerKilogramRoublesIsNotGreaterThanZero()

{

const string expectedErrorMessage = "Цена за 1 кг должна быть положительным числом!";

var editableProduct = new EditableProduct();

var editProductVm = new EditProductViewModel(editableProduct);

editProductVm.Name = "TEST";

editProductVm.CaloricContentPer100Grams = 1;

editProductVm.WeightGrams = 1;

Action validatePropertiesMethodCall = () => editProductVm.ValidateProperties();

editProductVm.PricePerKilogramRoubles = 0;

var e = Assert.ThrowsException<Exception>(validatePropertiesMethodCall);

Assert.AreEqual(expectedErrorMessage, e.Message);

editProductVm.PricePerKilogramRoubles = -1;

e = Assert.ThrowsException<Exception>(validatePropertiesMethodCall);

Assert.AreEqual(expectedErrorMessage, e.Message);

editProductVm.PricePerKilogramRoubles = decimal.MinValue;

e = Assert.ThrowsException<Exception>(validatePropertiesMethodCall);

Assert.AreEqual(expectedErrorMessage, e.Message);

}

}

}