Содержание

[Введение 6](#_Toc199934731)

[1 Аналитическая часть 8](#_Toc199934732)

[1.1 Анализ предметной области 8](#_Toc199934733)

[1.2 Анализ и организация входных и выходных данных 9](#_Toc199934734)

[2 Проектирование базы данных 12](#_Toc199934735)

[3 Разработка программного обеспечения 21](#_Toc199934736)

[3.1 Описание программных модулей 21](#_Toc199934737)

[3.2 Описание интерфейса программного обеспечения 22](#_Toc199934738)

[3.3 Тестирование программного обеспечения 26](#_Toc199934739)

[Заключение 28](#_Toc199934740)

[Список используемых источников 29](#_Toc199934741)

[Приложение А «Руководство пользователя» 32](#_Toc199934742)

[Приложение Б «Листинг программы» 40](#_Toc199934743)

Введение

Учебная практика (УП) по профессиональному модулю (ПМ) 01 «Разработка модулей программного обеспечения компьютерных систем» пройдена на базе ГБПОУ Нефтекамский машиностроительный колледж (НМК) Сроки прохождения УП: 26.05.2025 – 07.06.2025.

Рабочее место, предоставленное организацией для выполнения целей и задач УП представлено на рисунке 1.



Рисунок 1 – Рабочее место

Целью УП является разработка программного модуля, предназначенного для управления ингредиентами и поставщиками для кофейни «Амброзия Богов». В ходе УП необходимо выполнить индивидуальное задание по теме «Разработка программного приложения для управления ингредиентами и поставщиками «Амброзия Богов».

Для достижения целей необходимо выполнить следующие задачи:

* анализировать предметную область;
* разработать систему на базе WPF и базы данных SQL Server, которая автоматизирует учёт ингредиентов и взаимодействие с поставщиками;
* разработать пользовательский интерфейс, реализующий возможность добавления, редактирования и удаления данных в таблицах Ingredients и Suppliers;
* реализовать WPF с использованием MVVM;
* реализовать расчёт стоимости минимального заказа на основе дефицита ингредиент;
* обеспечить корректное обновления данных;
* разработан полный SQL–скрипт (SQLScript.sql) для создания базы данных CoffeeDBMurtazin.

По итогам УП необходимо обобщить и систематизировать полученную информацию, сформировать отчётную документацию, записать файлы с выполненными заданиями на носитель, оформить дневник практики.

1 Аналитическая часть

1.1 Анализ предметной области

Кофейня «Амброзия Богов» специализируется на приготовлении кофейных напитков, десертов и чая. Компания закупает сырье у поставщиков и реализует готовую продукцию через собственный пункт обслуживания клиентов. Для оптимизации деятельности разрабатывается система, включающая следующие подсистемы:

* меню и ингредиенты;
* склад и поставки;
* производство напитков и десертов;
* сотрудники и доступ;
* заказы клиентов.

1. Поставщики: Поставщики обеспечивают ингредиенты (кофейные зерна, молочные продукты, сиропы, специи и другие). Каждый поставщик характеризуется типом (производитель, дистрибьютор), наименованием компании, юридическим адресом, ИНН, ФИО представителя, контактными данными (телефон, email), логотипом, рейтингом надёжности и историей поставок. Рейтинг влияет на выбор поставщиков: аналитики оценивают качество поставок и рекомендуют прекращение сотрудничества с ненадёжными компаниями.
2. Менеджеры: Менеджеры занимаются поиском и регистрацией поставщиков, решением об изменении рейтинга поставщиков, ведением истории изменений рейтинга, приёмом заявок на поставку ингредиентов, формированием предложений на основе статистики использования сырья (полученной от аналитиков).
3. Заказы на поставку ингредиентов: если поставщик ранее работал с кофейней, менеджер формирует заявку на основе истории использования ингредиентов и текущих остатков на складе. Для нового поставщика менеджер предоставляет каталог ингредиентов и принимает заявку. После согласования количества, стоимости и сроков поставки заявка считается созданной. Заказ может быть отменён поставщиком до оплаты.
4. Сотрудники: Сотрудники имеют ФИО, дату рождения, паспортные данные, банковские реквизиты, семейное положение, состояние здоровья (для допуска к работе с оборудованием). Доступ к помещениям осуществляется через карту сотрудника, данные о перемещении используются для аналитики.
5. Производство: Производство включает приготовление напитков и десертов по рецептам меню, контроль качества готовой продукции, заказ недостающих ингредиентов у поставщиков, учёт времени изготовления и себестоимости блюд.
6. Склад: на складе хранятся ингредиенты, характеризующиеся типом (кофейные зерна, молочные продукты, сиропы и другие), наименованием, единицей измерения (кг, л, шт), стоимостью за единицу, количеством на складе, минимальным допустимым количеством, сроком годности, историей изменений остатков. Склад обеспечивает регистрацию поступлений от поставщиков, резервирование ингредиентов для производства, списание материалов при приготовлении блюд, контроль остатков и движений товаров.
7. Меню и заказы клиентов: Меню включает категории: кофе, чай, десерты, коктейли, завтраки. Каждое блюдо имеет название, описание, изображение, цену, калорийность, необходимые ингредиенты (связь с таблицей Ingredients). Заказы клиентов имеют статусы («Оплачен», «На доставке», «Возврат» и другие), детализацию (выбранные блюда, количество, общая стоимость), оплату через кассу и учёт времени выполнения заказа.

1.2 Анализ и организация входных и выходных данных

Входные данные приложения представляют собой информацию, поступающую из внешних источников или формируемую внутри системы для дальнейшей обработки. Основные категории входных данных:[7]

1. Данные о поставщиках

Включают наименование компании, тип (производитель/дистрибьютор), юридический адрес, ИНН, ФИО представителя, контактные данные (телефон, email), логотип, рейтинг надежности и историю поставок. Эти данные используются для управления взаимодействием с поставщиками, анализа их надежности и принятия решений о сотрудничестве.

1. Информация об ингредиентах

Характеризуется типом, названием, единицей измерения, стоимостью за единицу, количеством на складе, минимальным допустимым остатком, сроком годности и историей изменений остатков. Эти данные поступают при регистрации поставок, списании материалов в производство и корректировке складских запасов.

1. Заявки на поставку ингредиентов

Формируются менеджерами на основе статистики использования сырья и текущих остатков. Содержат параметры: количество, стоимость, сроки поставки. Данные используются для автоматического расчета минимальных партий, контроля оплаты и отслеживания выполнения заказов.

1. Данные о сотрудниках

Включают ФИО, дату рождения, паспортные и банковские данные, семейное положение, состояние здоровья и допуск к оборудованию. Эта информация необходима для управления доступом и планирования графиков работы.

1. Транзакции склада

Регистрируются при поступлении ингредиентов, их резервировании для производства, списании или корректировке остатков. Каждая транзакция содержит дату, количество, тип операции, ответственного сотрудника и связь с поставщиком или рецептом.

Выходные документы

Выходные документы формируются системой на основе обработки входных данных и предназначены для анализа, отчетности и управления. Основные категории:[2]

1. Отчеты по складским остаткам

Показывают текущее количество ингредиентов, минимальные допустимые уровни и прогнозы дефицита. Интегрируются с аналитикой для автоматической генерации заявок поставщикам при достижении пороговых значений.

1. Детализация транзакций

Включают журнал операций с указанием типа (поступление, списание, корректировка), даты, ответственного сотрудника и связанных объектов (ингредиент, поставщик, рецепт). Используются для аудита и контроля движения материалов.

1. Аналитика по поставщикам

Формируются рейтинги надежности, отчеты о качестве поставок и рекомендации по прекращению сотрудничества с ненадежными поставщиками. Данные основываются на истории поставок и отзывах сотрудников.

Все входные данные валидируются системой перед записью в базу (например, проверка ИНН, формата email, допустимых значений рейтинга). Интеграция с аналитическими модулями позволяет автоматически формировать рекомендации (например, обновление рецептов, корректировку цен на меню) на основе исторических данных.

2 Проектирование базы данных

База данных (БД) представляет собой структурированное хранилище информации, предназначенное для эффективного хранения, управления и обработки данных. Во–первых, она служит централизованным хранилищем для систематизированного накопления данных различного характера. Во–вторых, обеспечивает оперативный доступ к информации через специализированные запросы, что значительно ускоряет процесс обработки данных. В контексте разработанной системы для кофейни «Амброзия Богов» база данных используется для учета поставщиков, ингредиентов, сотрудников, заказов клиентов и других сущностей, связанных с операционной деятельностью предприятия.[4]

В качестве системы управления базами данных (СУБД) выбран Microsoft SQL Server, что подтверждается синтаксисом SQL–скриптов и настройками проекта. SQL Server – это мощная реляционная СУБД, разработанная Microsoft, которая обеспечивает высокую производительность, надежность и безопасность данных. Она поддерживает сложные транзакции, аналитическую обработку, интеграцию с другими инструментами Microsoft (например, .NET Framework) и предоставляет широкие возможности для администрирования и масштабирования. В данной системе SQL Server используется для реализации логики хранения и обработки данных, включая управление складскими остатками, регистрацию транзакций, работу с заказами и взаимодействие с поставщиками. [3]

Таблица 2.1 – Suppliers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null | Свойства |
| SupplierID | int | PK | Not Null |  |
| Name | nvarchar(50) |  | Not Null |  |

Продолжение таблицы 2.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | nvarchar(25) |  | Null | Ссылка на TypeSupplier |
| YuridAdres | nvarchar(100) |  | Null |  |
| INN | float |  | Null |  |
| FIO | nvarchar(100) |  | Null |  |
| Phone | nvarchar(25) |  | Null |  |
| Email | nvarchar(100) |  | Null |  |
| Logo | nvarchar(100) |  | Null |  |
| RatingReliability | decimal(18, 2) |  | Null |  |

Таблица 2.2 – TypeSupplier

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null | Свойства |
| Type | nvarchar(25) | PK | Not Null |  |

Таблица 2.3 – Ingredients

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null | Свойства |
| IngredientID | int | PK | Not Null |  |
| IngredientsName | nvarchar(50) |  | Not Null |  |
| TypeIngredients | nvarchar(50) | FK | Null | Ссылка на TypeIngredients |
| UnitOfMeasurement | nvarchar(5) |  | Null | Ссылка на UnitOfMeasurement |
| KolichSklad | decimal(18, 2) |  | Null |  |

Продолжение таблицы 2.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MinimKolich | int |  | Null |  |
| KolichUpakovka | int |  | Null |  |
| CostForOne | int |  | Null |  |
| Description | nvarchar(100) |  | Null |  |
| ExpirationDate | datetime |  | Null |  |

Таблица 2.4 – UnitOfMeasurement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null | Свойства |
| UnitOfMeasurement | nvarchar(5) | PK | Not Null |  |

Таблица 2.5 – IngredientSuppliers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null | Свойства |
| IngredientID | int | FK | Not Null | Ссылка на Ingredients |
| SupplierID | int | FK | Not Null | Ссылка на Suppliers |
| IDIngredientSupplier | int | PK | Not Null |  |

Таблица 2.7 – InventoryTransactions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null | Свойства |
| TransactionID | int | PK | Not Null |  |
| TransactionDate | datetime |  | Not Null |  |
| Quantity | decimal(10, 2) |  | Not Null |  |

Продолжение таблицы 2.7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TransactionType | nvarchar(50) |  | Not Null | Ссылка на TransactionType |
| EmployeeID | int | FK | Null | Ссылка на Employees |
| ExpirationDate | datetime |  | Null |  |
| IngredientID | int | FK | Null | Ссылка на Ingredients |
| SupplierID | int | FK | Null | Ссылка на Suppliers |

Таблица 2.7 – TypeIngredients

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null | Свойства |
| TypeIngredients | nvarchar(50) | PK | Not Null |  |

Таблица 2.8 – TransactionType

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null | Свойства |
| TransactionType | nvarchar(20) | PK | Not Null |  |

Таблица 2.9 – Employees

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null | Свойства |
| EmployeeID | int | PK | Not Null |  |
| FIO | nvarchar(100) |  | Null |  |

Продолжение таблицы 2.9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PositionID | Int | FK | Null | Ссылка на EmployeesPositions |
| Phone | nvarchar(25) |  | Null |  |
| Email | nvarchar(100) |  | Null |  |
| HireDate | date |  | Null |  |

Таблица 2.10 – PositionsEmployee

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null | Свойства |
| PositionID | int | PK | Not Null |  |
| PositionName | nvarchar(50) |  | Null |  |
| Description | nvarchar(200) |  | Null |  |

Таблица 2.11 – WorkSchedules

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null | Свойства |
| ScheduleID | int | PK | Not Null |  |
| EmployeeID | int | FK | Not Null | Ссылка на Employees |
| WorkDate | date |  | Null |  |
| StartTime | time(7) |  | Null |  |
| EndTime | time(7) |  | Null |  |

Таблица 2.12 –MenuCategories

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null | Свойства |
| CategoryID | int | PK | Not Null |  |
| CategoryName | nvarchar(100) |  | Not Null |  |
| Description | nvarchar(200) |  | Null |  |

Таблица 2.13 – MenuItems

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null | Свойства |
| MenuItemID | int | PK | Not Null |  |
| CategoryID | int | FK | Not Null | Ссылка на MenuCategories |
| ItemName | nvarchar(100) |  | Not Null |  |
| Description | nvarchar(200) |  | Null |  |
| ImagePath | nvarchar(255) |  | Null | Путь к изображению |

Таблица 2.14 – Recipes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null | Свойства |
| RecipeID | int | PK | Not Null |  |
| MenuItemID | int | FK | Not Null | Ссылка на Ingredients |
| Quantity | decimal(10, 2) |  | Not Null |  |
| Instructions | nvarchar(500) |  | Null |  |
| IngredientID | int | FK | Null | Ссылка на Ingredients |

Таблица 2.15 – Orders

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null/Null | Свойства |
| OrderID | int | PK | Not Null |  |
| StatusID | int | FK | Not Null | Ссылка на OrderStatuses |
| TotalPrice | decimal(18, 2) |  | Not Null |  |
| OrderDate | datetime |  | Not Null | Дата создания |

Таблица 2.16 – OrderStatuses

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null/Null | Свойства |
| StatusID | int | PK | Not Null |  |
| StatusName | nvarchar(50) |  | Not Null |  |
| Description | nvarchar(200) |  | Null |  |

Таблица 2.17 – OrderDetails

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null/Null | Свойства |
| OrderDetailID | int | PK | Not Null |  |
| OrderID | int | FK | Not Null | Ссылка на Orders |
| MenuItemID | int | FK | Not Null | Ссылка на MenuItems |
| Quantity | int |  | Not Null |  |
| PriceOfOrder | decimal(10, 2) |  | Not Null |  |

Таблица 2.18 – InventoryAlerts

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип данных | PK/FK | Not Null/Null | Свойства |
| AlertID | int | PK | Not Null |  |
| AlertDate | datetime |  | Not Null |  |
| IsResolved | bit |  | Not Null |  |
| ResolvedBy | int |  | Null |  |
| ResolvedDate | datetime |  | Null |  |
| IngredientID | int | FK | Null | Ссылка на Ingredients |

Следует определить связи, степени связи между сущностями и построить ER–диаграмму БД (смотреть рисунок 2.1).

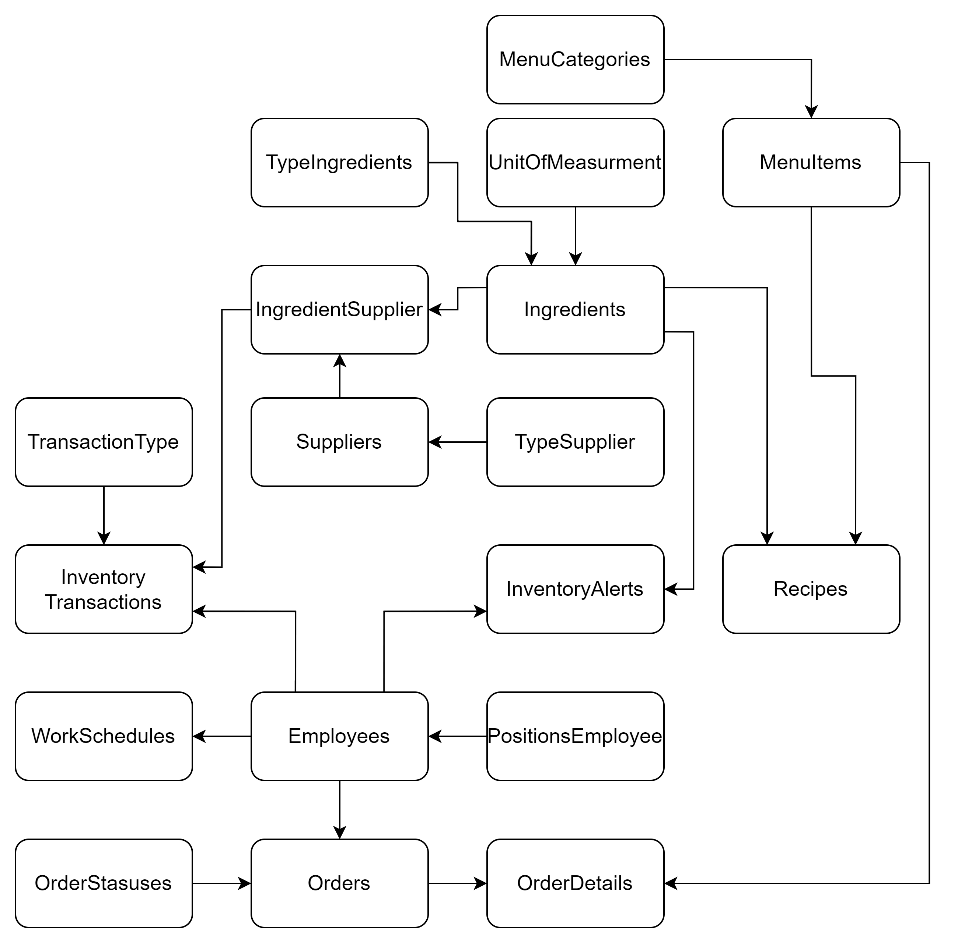


Рисунок 2.1 – Связи между таблицами

Проанализировав работу «Амброзия Богов», определив связи и степени связи между сущностями, можно построить ER–диаграмму БД с указанием атрибутов (смотреть рисунок 2.2).

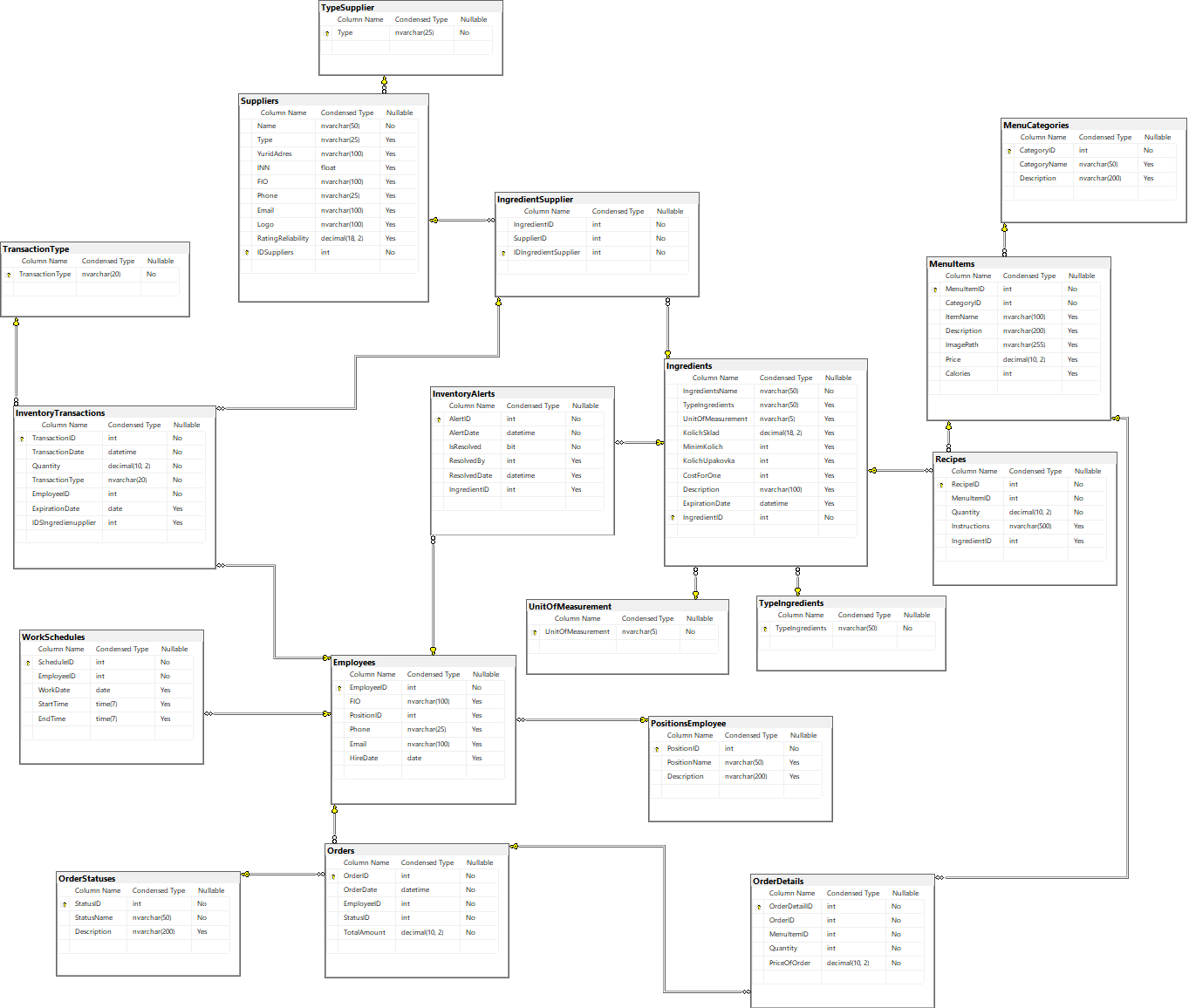


Рисунок 2.2 – ER–диаграмма

3 Разработка программного обеспечения

3.1 Описание программных модулей

Программный модуль – это автономная часть программного обеспечения, выполняющая определенные функции и взаимодействующая с другими модулями. Он объединяет логически связанные компоненты, такие как интерфейс, бизнес–логика и взаимодействие с данными, обеспечивая структурированность и модульность приложения. Его ключевые характеристики включают высокую связность, низкую связанность, что упрощает разработку, тестирование и поддержку кода.[1]

Проектирование модулей включает в себя:

Определение функциональных задач каждого модуля, его взаимодействие с другими компонентами системы, архитектурные принципы (разделение ответственности, инкапсуляция), реализацию пользовательского интерфейса и логики обработки данных, а также документирование назначения модуля, его API и ограничений для удобства дальнейшей разработки. В разрабатываемой системе для кофейни «Амброзия Богов» модули охватывают управление поставщиками, ингредиентами, заказами и аналитикой.

Разрабатываемый программный продукт состоит из 12 основных модулей.

Модули системы можно разделить на три основные группы:

1. Сущности данных (Models). «Entity», «SupplierModel», «IngredientsViewModel», «SuppliersViewModel», «IngredientModel», «TypeSupplier», «SupplierModel», «Suppliers», «IngredientSupplier», «TypeIngredients» и «UnitOfMeasurement». Эти модули отвечают за хранение и обработку данных, а также уведомления об изменениях (через «Notify»).

2. Представления (Views). Содержат «MainWindow» и «SuppliersWindow», которые являются графическими интерфейсами (наследуют от «Window»). Эти модули обеспечивают взаимодействие с пользователем.

3. Ядро уведомлений. почти во всех модулях для реализации механизма оповещений («Notify»), что указывает на систему событий или паттерн «Observer» для отслеживания изменений данных. Схема взаимодействия модулей представлена на рисунке 3.1.1.



Рисунок 3.1.1 – Диаграмма модулей

3.2 Описание интерфейса программного обеспечения

Интерфейс служит каналом взаимодействия между компонентами системы, пользователями или внешними приложениями. Он определяет правила и механизмы обмена информацией, обеспечивая согласованность и стандартизацию процессов. В программной инженерии интерфейсы могут быть программными (например, API), аппаратными (устройства ввода/вывода) или визуальными. Их главная цель – упростить коммуникацию, разделить функциональность на логические блоки и минимизировать сложность системы за счет четкого определения ответственности. [5]

Пользовательский интерфейс (UI) – это визуальная и функциональная реализация взаимодействия между человеком и системой. Он включает элементы управления (кнопки, поля ввода, меню), информационные блоки (тексты, графики) и навигационные инструменты (вкладки, ссылки). Эффективный UI должен быть интуитивно понятным, обеспечивать быстрое выполнение задач и минимизировать когнитивную нагрузку на пользователя. Например, в разрабатываемом приложении для кофейни «Амброзия Богов» окно управления поставщиками содержит форму редактирования данных с автозаполнением типов и валидацией полей, что упрощает работу менеджера и снижает риск ошибок. [5]

Принципы разработки пользовательского интерфейса сосредоточены на создании удобного, предсказуемого и адаптивного взаимодействия. Ключевые аспекты включают:

* Простоту навигации: элементы интерфейса организованы логично, например, в модуле учета ингредиентов фильтрация по типам расположена в верхней части окна, чтобы пользователь сразу видел доступные опции.
* Последовательность дизайна: унифицированные стили кнопок, шрифтов и цветовых схем в разных модулях.
* Обратную связь: система мгновенно реагирует на действия пользователя, например, при сохранении изменений в данных поставщика отображается уведомление об успешном обновлении.
* Адаптивность: интерфейс оптимизирован под разные устройства и разрешения экрана.
* Минимализм: исключение лишних элементов, например, в окне редактирования ингредиента отсутствуют второстепенные функции, чтобы пользователь фокусировался на ключевых параметрах (название, тип, остатки на складе).[6]

При запуске модуля учета ингредиентов пользователь видит список ингредиентов, сгруппированных по наименованию. В верхней части интерфейса размещены фильтр сортировки с опцией «Все типы» и две кнопки: «А–Я» и «Я–А» позволяющие сортировать список ингредиентов по наименованию. Также в правой части экрана снизу у нас есть три кнопки, «СОХРАНИТЬ ВСЕ», «+ ДОБАВИТЬ НОВЫЙ», «Удалить» и «Эксель». Как можно понять из названий первая кнопка сохраняет все внесенные изменения в БД, вторая кнопка добавляет новые ингредиенты, а третья удаляет выбранный ингредиент. Каждый ингредиент отображается в карточке, где указаны его наименование, категория, минимальное количество для заказа, объем упаковки, срок годности и стоимость минимальной партии. Интерфейс обеспечивает быстрое взаимодействие с ключевыми параметрами и эффективное управление складскими остатками. Данное окно представлено на рисунке 3.2.1.

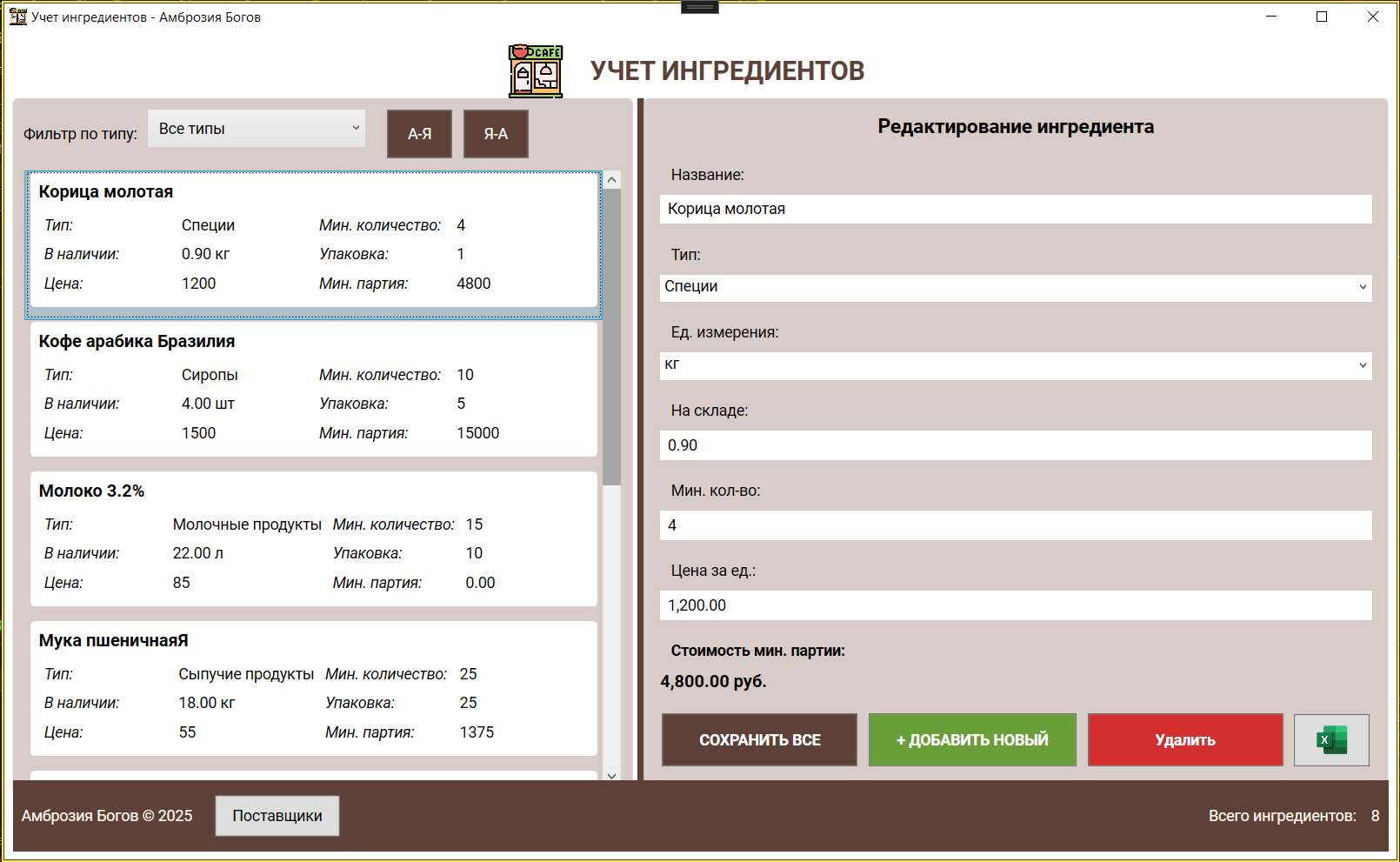


Рисунок 3.2.1 – Главное окно

При запуске модуля управления поставщиками пользователь видит список поставщиков, сгруппированных по наименованию. В верхней части интерфейса размещены фильтр сортировки с опцией «Все типы» и две кнопки: «А–Я» и «Я–А» позволяющие сортировать список поставщиков по наименованию. Также в правой части экрана снизу у нас есть три кнопки, «СОХРАНИТЬ ВСЕ», «+ ДОБАВИТЬ НОВЫЙ» и «Удалить». Как можно понять из названий первая кнопка сохраняет все внесенные изменения в БД, вторая кнопка добавляет новых поставщиков, а третья удаляет выбранного поставщика. Каждый поставщик отображается в карточке, где указаны его название компании, тип деятельности (производитель или дистрибьютер), ИНН, контактное лицо, телефон, электронную почту и рейтинг. При нажатии же у нас выведутся полные данные о поставщике, появятся данные о юридическом адресе. Интерфейс обеспечивает быстрое взаимодействие с ключевыми параметрами и эффективное управление данными о поставщиках. Данное окно представлено на рисунке 3.2.2.

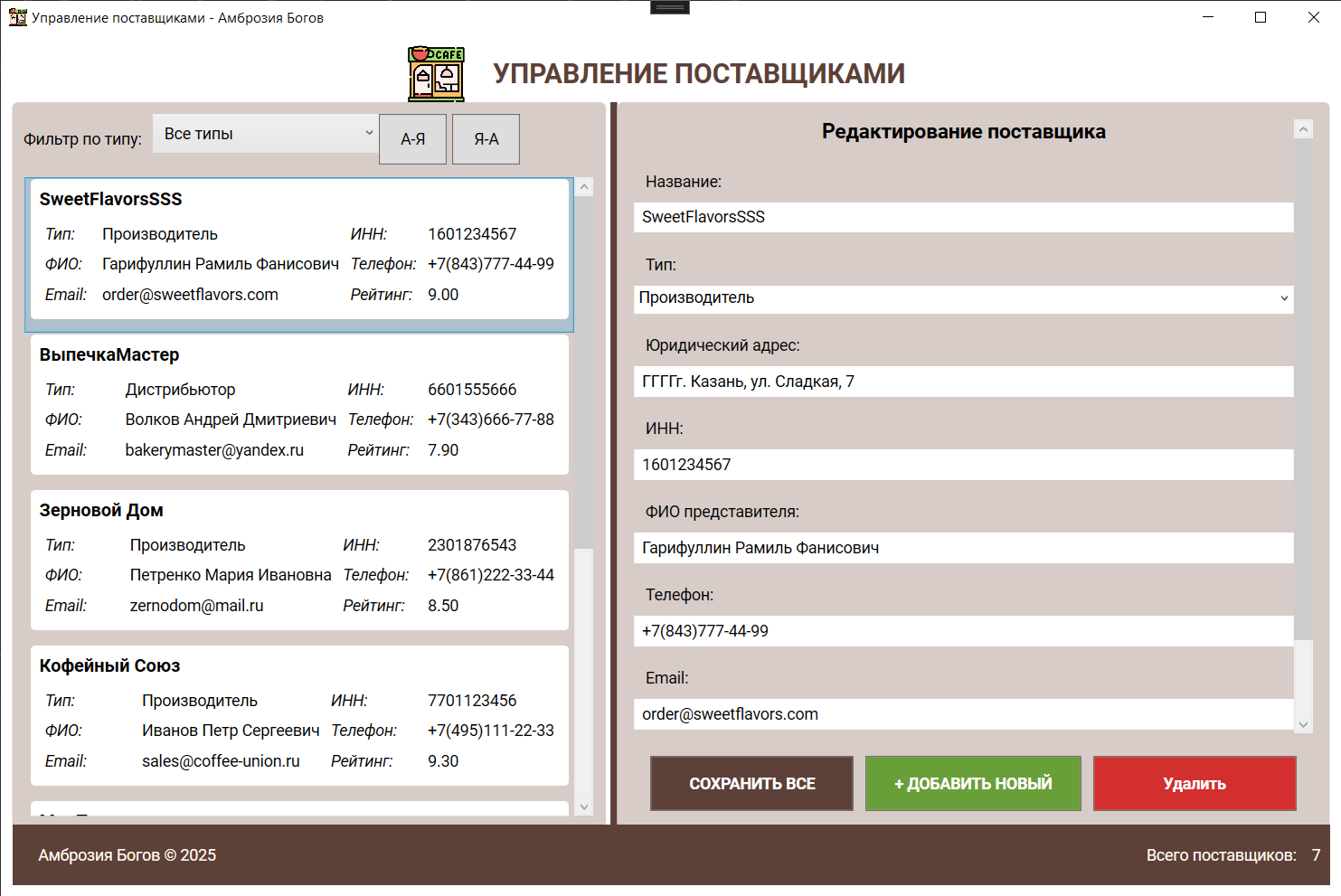


Рисунок 3.2.2 – Окно управления поставщиками

3.3 Тестирование программного обеспечения

Тестирование программного обеспечения представляет собой процесс проверки соответствия разработанного приложения заданным требованиям, выявления ошибок и обеспечения его надежной работы в различных условиях. Основная цель тестирования – подтвердить, что система выполняет свои функции корректно, обрабатывает данные без сбоев и взаимодействует с пользователями и внешними системами в соответствии с ожиданиями. Тестирование охватывает различные уровни, включая проверку отдельных модулей, интеграции компонентов, пользовательского интерфейса и производительности системы в целом.[5]

Unit–тест – это метод тестирования, направленный на проверку работоспособности отдельных функций, методов или классов программного кода в изоляции от остальной системы. Такие тесты разрабатываются разработчиками на этапе написания кода и автоматически выполняются для подтверждения корректности логики каждого элемента. Например, в контексте приложения для кофейни «Амброзия Богов» unit–тесты могут проверять корректность расчета стоимости минимальной партии ингредиентов, валидацию данных при добавлении нового поставщика или корректность обновления остатков на складе после списания материалов. Для реализации unit–тестов используются специализированные фреймворки, такие как NUnit, XUnit и MSTest, которые позволяют автоматизировать проверку сценариев и интегрировать тестирование в процесс разработки. [6]

Был проведет тест на модуле «Ингредиенты», протестированы следующие функции «Вывод Ингредиентов», «Добавление Ингредиента», «Удаление Ингредиента». Результаты показаны на рисунках 3.3.1–3.3.2

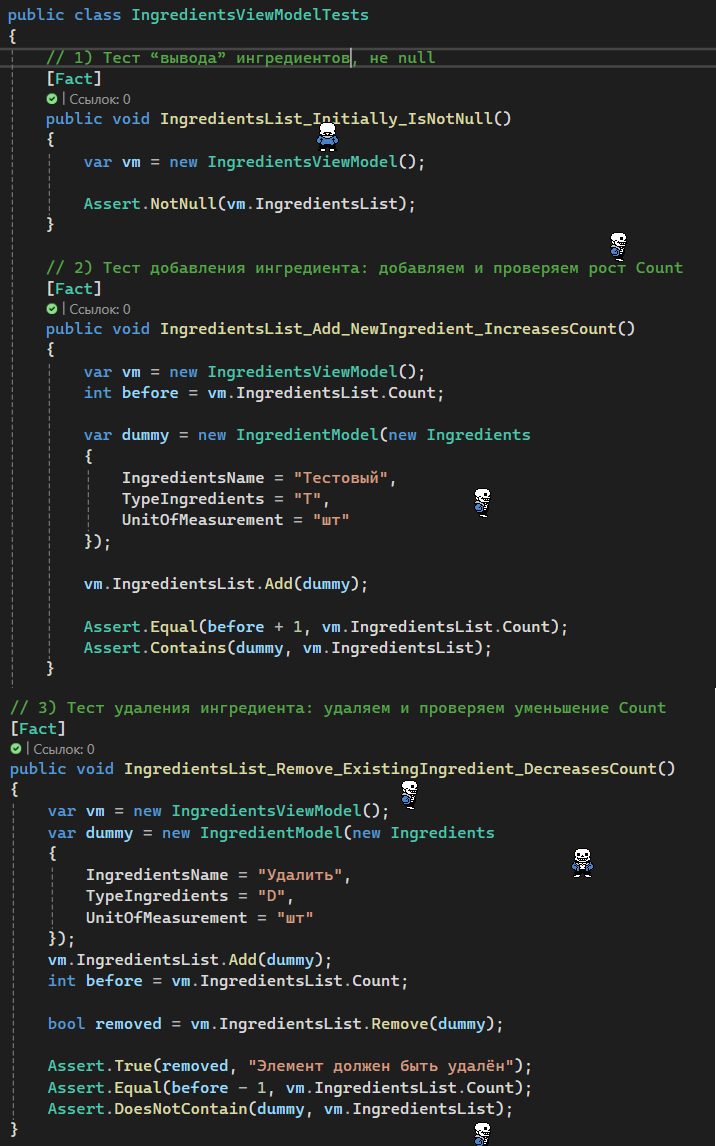


Рисунок 3.3.1 – Код тестирования

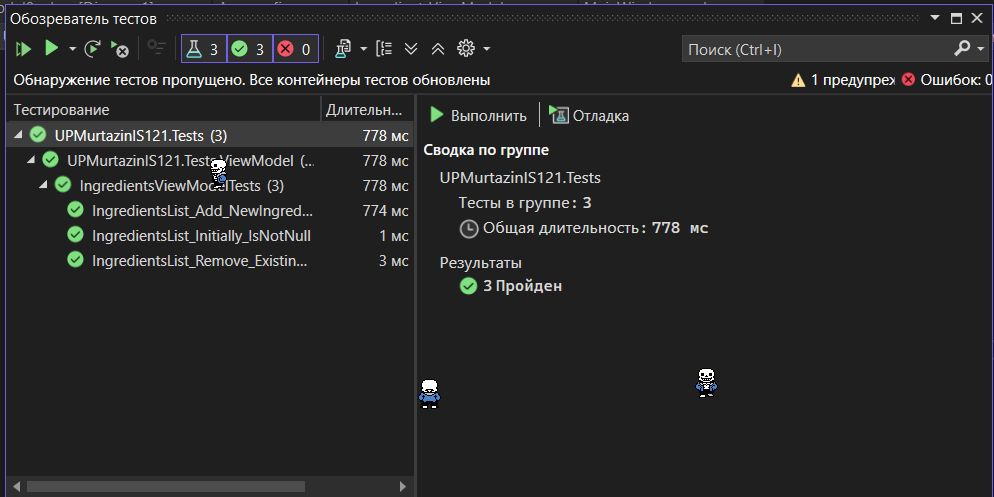


Рисунок 3.3.2 – Результат тестирования

Заключение

В ходе выполнения задач по учебной практике был разработан программный модуль «Амброзия Богов» в интегрированной среде разработки Visual Studio.

«Амброзия Богов» разработан на основе SQL–языка для формирования БД, знаний C# и применения паттерна MVVM. «Амброзия Богов» предусматривает функции вывода данных о ингредиентах и поставщиках, дальнейшего добавления новых данных, а также их редактирования с подключением к локальной БД. «Амброзия Богов» содержит два модуля, интерфейс упрощен и ориентирован на обычных неопытных пользователей. Для подключения БД применили подключение ADO.NET по СУБД Microsoft SQL Management Server. Для проведения тестирования были применены библиотеки Xunit. Для формирования всей системы и связи интерфейса с моделью данных были применены и реализованы интерфейсы INotifyPropertyChanged и ICommand. Для обмена данными в каталоге предусмотрен экспорт в Excel файл.[6]

В качестве источников информации использовалась и глобальная сеть Интернет, книги, которые перечислены в разделе использованных источников.

Для отображения результатов практики сформирована отчетная документация, файлы с выполненным индивидуальным заданием записаны на флеш–накопитель.

Таким образом, цель учебной практики по ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем» достигнута, задачи выполнены.

Список используемых источников

Основные источники:

1. Arnaud Weil Learn WPF MVVM – XAML, C# and the MVVM pattern: учебное пособие / Arnaud Weil ; под ред. Arnaud Weil.: ФОРУМ : ИНФРА–М, 2016. – 174 с. – ISBN 978–1326847999. – Текст : электронный. – URL: <https://leanpub.com/learnwpf/read_sample> (дата обращения: 02.06.2025).
2. К. Дж. Дейт. Введение в системы баз данных = An Introduction to Database System, 7th Edition. – 7–е изд. – М.: Вильямс, 2001. – С. 1072. – ISBN 5–8459–0138–3. URL: <https://ilshatpro.wordpress.com/wp-content/uploads/2017/08/d0ba-d0b4d0b6-d0b4d0b5d0b9d182-d0b2d0b2d0b5d0b4d0b5d0bdd0b8d0b5-d0b2-d181d0b8d181d182d0b5d0bcd18b-d0b1d0b0d0b7-d0b4d0b0d0bdd0bdd18b.pdf> (дата обращения: 01.06.2025).
3. Хомопепко А. Д., Цыганков В. М., Мальцев М. Г.Базы данных: Учебник для высших учебных заведений / Под ред. np(xj). А. Д. Хомопепко. –6–е изд., доп. – СПб.: КОРОНА–Век, 2009. – 736 с.ISBN 978–5–7931–0527–9 URL: <https://studizba.com/show/1084484-1-homonenko-ad-cygankov-vm-malcev-mg-bazy.html> (дата обращения: 01.06.2025).

Интернет–ресурсы:

1. Электронный ресурс "Документация Microsoft SQL Server". Форма доступа: <https://docs.microsoft.com/en-us/sql/>
2. Электронный ресурс «Язык программирования C# и платформа .NET». Форма доступа: <https://metanit.com/sharp/>
3. Электронный ресурс «Определение паттерна MVVM» Форма доступа: <https://metanit.com/sharp/wpf/22.1.php>

Дополнительные источники:

1. Кеннет, Х. "Основы реляционных баз данных: проектирование и работа с SQL" / Х. Кеннет, Т. Карт. – 7–е изд., Пер. с англ. – СПб.: Вильямс, 2009. – 1100 с.

|  |
| --- |
| Приложение А |
|  |
| РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ |
| Разработка программного приложения для кофейни «Амброзия Богов» |
| УП ИС121.М1840.34 ОС |

Содержание

1 Общие сведения о программе 32

2 Описание установка 33

3 Описание запуска 34

4 Описание пользовательского интерфейса 35

5 Сообщения пользователю 37

# Общие сведения о программе

Программное обеспечение «Амброзия Богов» разработано для кофейни «Амброзия Богов» и предназначено для решения задач автоматизации учета ингредиентов и поставщиков. Программное обеспечение позволяет автоматизировать работу менеджера, вести учет ингредиентов, автоматизировать запись поставщиков, редактировать данные склада кофейни и обновлять их. База данных позволяет сократить время на поиск нужной информации.

Программа обеспечивает хранение сведений об ингредиентах и поставщиках кофейни. Сведения об ингредиентах включают их наименование, тип, минимальное количество, количество за упаковку, цену за единицу, описание, срок годности. В программе ведется журнал записей поставщиков ингредиентов.

В обязанности администратора кофейни входит ведение базы данных, поиск необходимой информации, заполнение данных.

В базе данных предусмотрена возможность фильтрации ингредиентов и поставщиков, а также их сортировка. Предусмотрена возможность вывода информации в текстовый редактор Excel.

При появлении новых задач можно в кратчайшие сроки реализовать их в программе, путем добавления новых функциональных возможностей.

Программное обеспечение «Амброзия Богов» может быть полезно при работе в области предоставления услуги общепита и отдыха.

# Описание установка

Программное обеспечение «Амброзия Богов» может быть свободно установлено на компьютер в любой, удобный для пользователя каталог путем копирования папки «DownloadUP».[5]

Для установки требуется 4,7 Мб свободного места на диске.

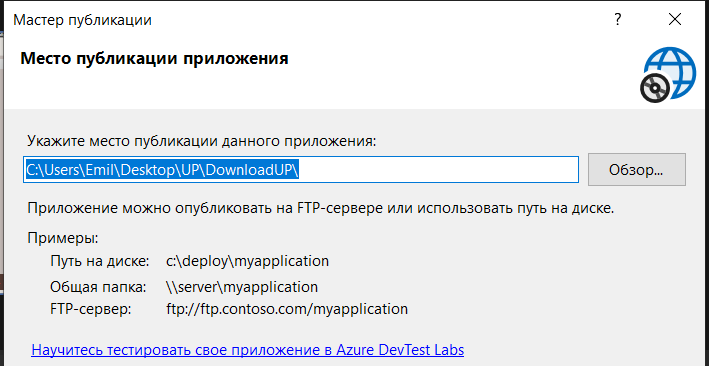


Рисунок А.2.1 – Публикация проекта

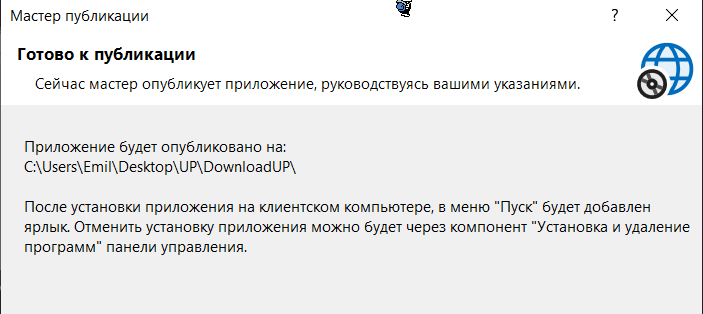


Рисунок А.2.2 – Готовность к публикации

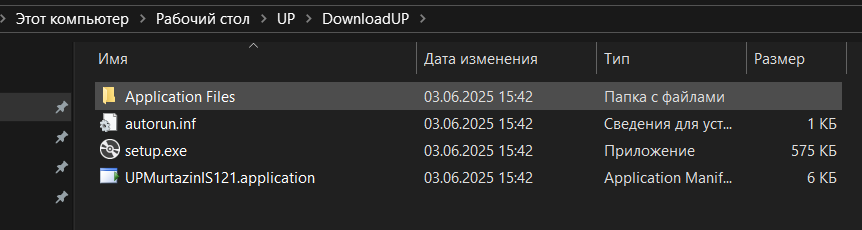


Рисунок А.2.3 – папка с установленным обеспечением

# Описание запуска

Рекомендуется создать ярлык на рабочем столе для запуска программы. Для этого скопировать ярлык UPMurtazinIS121.application из папки «DownloadUP» на рабочий стол.

Запуск программы осуществляется двойным щелчком по ярлыку UPMurtazinIS121.application или путем запуска файла приложений VisualStudio.



Рисунок А.3.1 – ярлык ПО

# Описание пользовательского интерфейса

При запуске программы появляется основное окно–заставка с логотипом кофейни и наименования текущего окна и выведенными данными ингредиентов. (рисунок А.4.1).

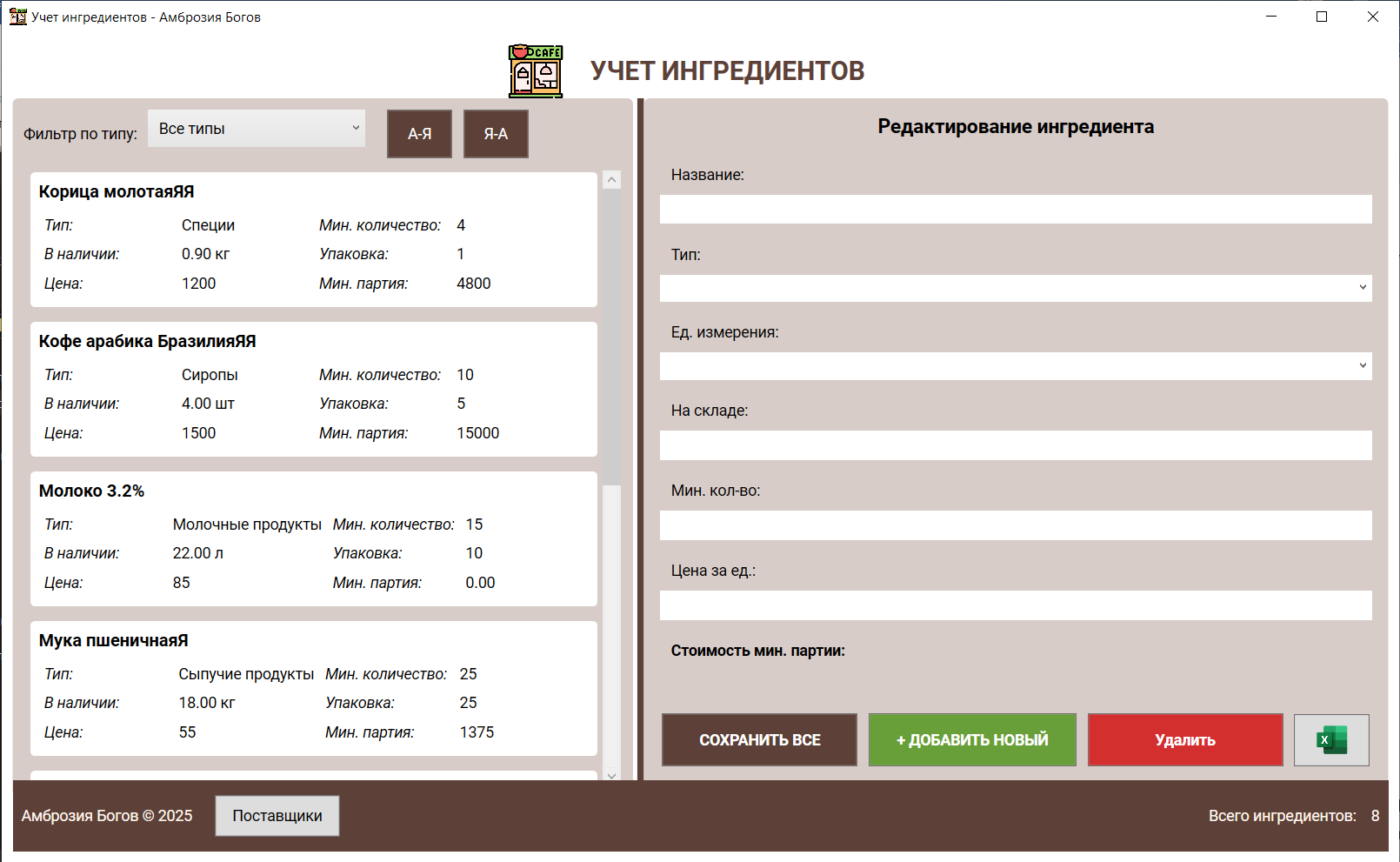


Рисунок А.4.1 – Главное окно программы

Здесь же имеются кнопки для перехода к соответствующим разделам базы данных: Ингредиенты и Поставщики.

На закладке Ингредиенты расположена информация об ингредиентах, а также данные, которые можно изменить (рисунок А.4.1).

Предусмотрено добавление, редактирование и удаление записей.

При выборе меню «+ Добавить Новый» на форме появляется дополнительная строка для ввода новых данных и кнопка «Добавить» (рисунок А.3).

Для продолжения работы с другими разделами программы достаточно закрыть форму.

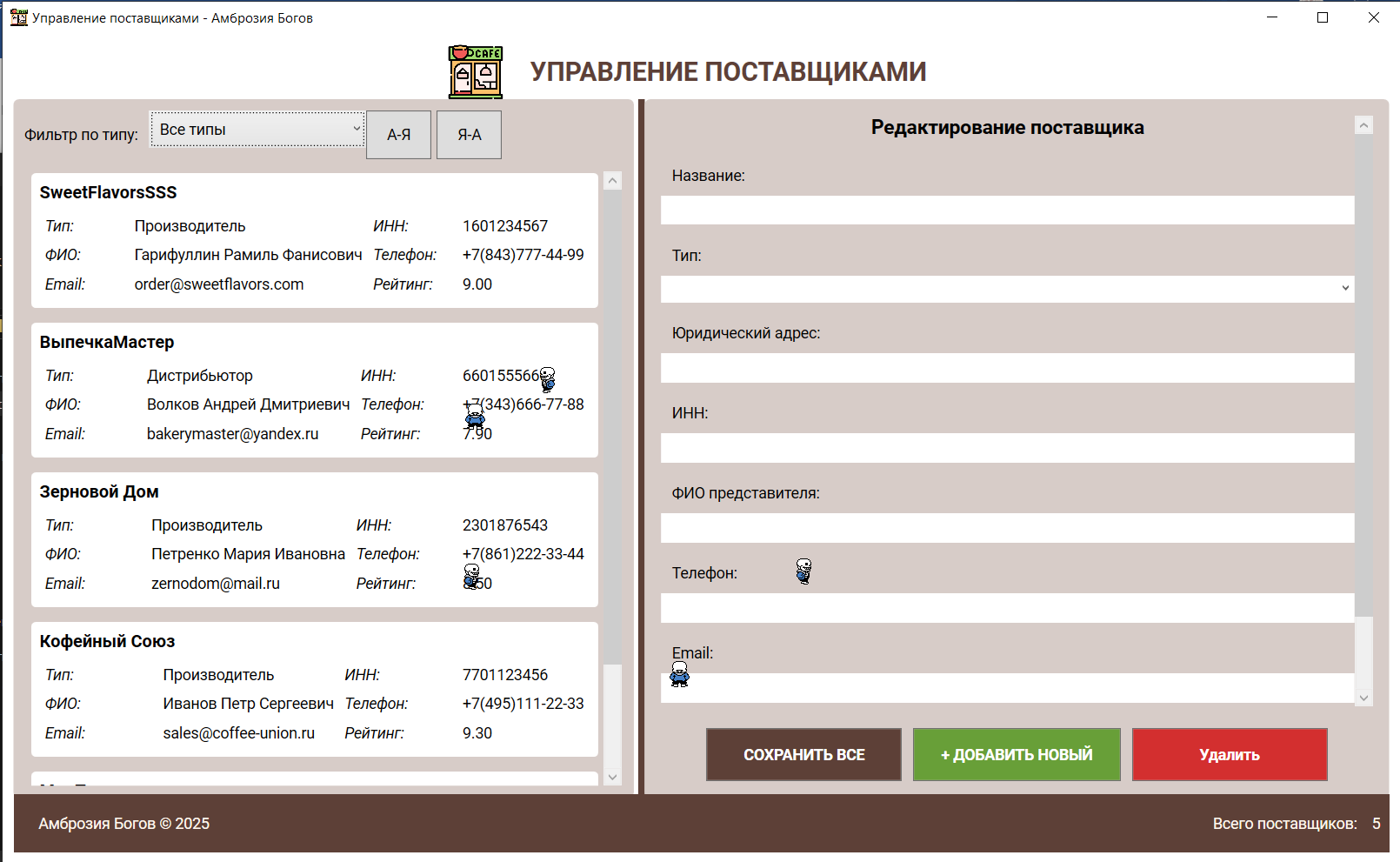


Рисунок А.4.2 – Окно «Поставщики»

В разделе «Поставщики» ведется учет данных о поставщиках (рисунок А.4.2).

На обоих формах имеются боксы для сортировки по названию данных. На обеих формах есть конкретизирующие поля, в первом окне они позволяют влиять на данные в БД. Поля в обеих окнах имеют маски ввода

# Сообщения пользователю

При попытке добавления данных в базу без заполнения поля «Тип» программа выдает сообщение, показанное на рисунке А.5.1

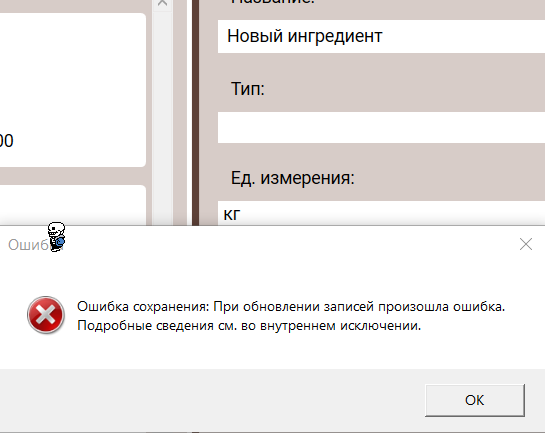


Рисунок А.5.1 – Сообщение об ошибке

При удалении записей появляется сообщение (рисунок А.5.2).

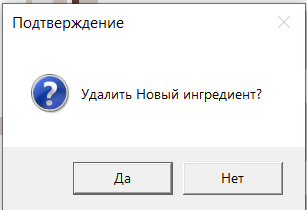


Рисунок А.5.2 – Сообщение о подтверждении удаления

|  |
| --- |
| Приложение Б |
|  |
| ЛИСТИНГ ПРОГРАММЫ |
| Разработка программного приложения для кофейни «Амброзия Богов» |
| УП ИС121.М1840.12 ОС |
|  |

using System;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Windows;

using System.Windows.Input;

using Microsoft.Win32;

using Excel = Microsoft.Office.Interop.Excel;

using System.Diagnostics;

using UPMurtazinIS121.Model;

using UPMurtazinIS121.Validations;

namespace UPMurtazinIS121.ViewModel

{

public class IngredientsViewModel : INotifyPropertyChanged

{

private readonly CoffeeDBMurtazinEntities2 \_context;

private IngredientModel \_selectedIngredient;

public ObservableCollection<IngredientModel> IngredientsList { get; } = [];

private string \_selectedFilterType;

public ObservableCollection<string> IngredientTypes { get; } = [];

public ObservableCollection<IngredientModel> FilteredIngredientsList { get; } = [];

public ObservableCollection<string> MeasurementUnits { get; } =

[

"шт", "кг", "г", "л"

];

public string SelectedFilterType

{

get => \_selectedFilterType;

set

{

\_selectedFilterType = value;

OnPropertyChanged();

ApplyFilter();

}

}

public IngredientModel SelectedIngredient

{

get => \_selectedIngredient;

set

{

\_selectedIngredient = value;

OnPropertyChanged();

}

}

public ICommand SaveCommand { get; }

public ICommand AddNewCommand { get; }

public ICommand DeleteCommand { get; }

public ICommand ExportToExcelCommand { get; }

public IngredientsViewModel()

{

\_context = new CoffeeDBMurtazinEntities2();

\_context.Configuration.ProxyCreationEnabled = false;

LoadData();

SaveCommand = new RelayCommand(SaveChanges);

AddNewCommand = new RelayCommand(AddNewIngredient);

DeleteCommand = new RelayCommand(DeleteIngredient);

ExportToExcelCommand = new RelayCommand(\_ => ExportToExcel());

SortAscCommand = new RelayCommand(\_ => SortIngredients(true));

SortDescCommand = new RelayCommand(\_ => SortIngredients(false));

}

private void LoadData()

{

try

{

\_context.Ingredients.Load();

IngredientsList.Clear();

IngredientTypes.Clear();

IngredientTypes.Add("Все типы");

foreach (var ingredient in \_context.Ingredients.Local)

{

var vm = new IngredientModel(ingredient);

IngredientsList.Add(vm);

if (!string.IsNullOrEmpty(vm.TypeIngredients))

{

if (!IngredientTypes.Contains(vm.TypeIngredients))

IngredientTypes.Add(vm.TypeIngredients);

}

}

ApplyFilter();

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка загрузки: {ex.Message}", " Ошибка");

}

}

private void ApplyFilter()

{

FilteredIngredientsList.Clear();

if (string.IsNullOrEmpty(SelectedFilterType))

{

foreach (var item in IngredientsList)

FilteredIngredientsList.Add(item);

}

else if (SelectedFilterType == "Все типы")

{

foreach (var item in IngredientsList)

FilteredIngredientsList.Add(item);

}

else

{

foreach (var item in IngredientsList.Where(i => i.TypeIngredients == SelectedFilterType))

FilteredIngredientsList.Add(item);

}

}

public bool HasChanges()

{

return \_context.ChangeTracker.Entries()

.Any(e => e.State != EntityState.Unchanged);

}

private void SaveChanges(object parameter)

{

try

{

if (HasChanges())

{

\_context.SaveChanges();

MessageBox.Show("Изменения сохранены!", "Успех",

MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка сохранения: {ex.Message}", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void AddNewIngredient(object parameter)

{

var newIngredient = new Ingredients

{

IngredientsName = "Новый ингредиент",

TypeIngredients = "Тип",

UnitOfMeasurement = "кг",

KolichSklad = 0,

MinimKolich = 0,

KolichUpakovka = 1,

CostForOne = 0,

ExpirationDate = DateTime.Now.AddMonths(6)

};

var wrapper = new IngredientModel(newIngredient);

IngredientsList.Add(wrapper);

ApplyFilter(); // <---- Добавить

SelectedIngredient = wrapper;

\_context.Ingredients.Add(newIngredient);

}

private void DeleteIngredient(object parameter)

{

if (SelectedIngredient == null) return;

if (MessageBox.Show($"Удалить {SelectedIngredient.IngredientsName}?",

"Подтверждение",

MessageBoxButton.YesNo,

MessageBoxImage.Question) != MessageBoxResult.Yes) return;

try

{

var model = SelectedIngredient.GetModel();

\_context.Ingredients.Remove(model);

IngredientsList.Remove(SelectedIngredient); // Удаляем из основной коллекции

ApplyFilter(); // <---- Добавить

\_context.SaveChanges();

SelectedIngredient = IngredientsList.FirstOrDefault();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка удаления: {ex.Message}", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

public ICommand SortAscCommand { get; }

public ICommand SortDescCommand { get; }

private void SortIngredients(bool ascending)

{

var sorted = ascending

? [.. FilteredIngredientsList.OrderBy(s => s.IngredientsName)]

: FilteredIngredientsList.OrderByDescending(s => s.IngredientsName).ToList();

FilteredIngredientsList.Clear();

foreach (var s in sorted)

FilteredIngredientsList.Add(s);

}

private void ExportToExcel()

{

try

{

var dlg = new SaveFileDialog

{

Filter = "Excel Workbook|\*.xlsx",

FileName = "Ingredients.xlsx"

};

if (dlg.ShowDialog() != true)

return;

string path = dlg.FileName;

// Запускаем Excel

var excelApp = new Excel.Application

{

Visible = true

};

var workbook = excelApp.Workbooks.Add();

var sheet = (Excel.\_Worksheet)workbook.Sheets[1];

// Заголовки столбцов

sheet.Cells[1, 1] = "Название";

sheet.Cells[1, 2] = "Тип";

sheet.Cells[1, 3] = "Ед. измерения";

sheet.Cells[1, 4] = "В наличии";

sheet.Cells[1, 5] = "Мин. кол-во";

sheet.Cells[1, 6] = "Цена за ед.";

sheet.Cells[1, 7] = "Мин. партия";

for (int i = 0; i < FilteredIngredientsList.Count; i++)

{

var ing = FilteredIngredientsList[i];

int row = i + 2;

sheet.Cells[row, 1] = ing.IngredientsName;

sheet.Cells[row, 2] = ing.TypeIngredients;

sheet.Cells[row, 3] = ing.UnitOfMeasurement;

sheet.Cells[row, 4] = ing.KolichSklad;

sheet.Cells[row, 5] = ing.MinimKolich;

sheet.Cells[row, 6] = ing.CostForOne;

sheet.Cells[row, 7] = ing.MinOrderCost;

}

workbook.SaveAs(path);

}

catch (Exception ex)

{

MessageBox.Show($"Не удалось экспортировать в Excel: {ex.Message}",

"Ошибка экспорта",

MessageBoxButton.OK,

MessageBoxImage.Error);

}

}

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

}

using System;

using System.ComponentModel;

using System.Runtime.CompilerServices;

namespace UPMurtazinIS121.Model

{

public class IngredientModel(Ingredients ingredient) : INotifyPropertyChanged

{

private readonly Ingredients \_ingredient = ingredient;

public int IngredientID

{

get => \_ingredient.IngredientID;

set

{

if (\_ingredient.IngredientID != value)

{

\_ingredient.IngredientID = value;

OnPropertyChanged();

}

}

}

public string IngredientsName

{

get => \_ingredient.IngredientsName;

set

{

if (\_ingredient.IngredientsName != value)

{

\_ingredient.IngredientsName = value;

OnPropertyChanged();

}

}

}

public string TypeIngredients

{

get => \_ingredient.TypeIngredients;

set

{

if (\_ingredient.TypeIngredients != value)

{

\_ingredient.TypeIngredients = value;

OnPropertyChanged();

}

}

}

public string UnitOfMeasurement

{

get => \_ingredient.UnitOfMeasurement;

set

{

if (\_ingredient.UnitOfMeasurement != value)

{

\_ingredient.UnitOfMeasurement = value;

OnPropertyChanged();

}

}

}

public decimal? KolichSklad

{

get => \_ingredient.KolichSklad;

set

{

if (\_ingredient.KolichSklad != value)

{

\_ingredient.KolichSklad = value;

OnPropertyChanged();

OnPropertyChanged(nameof(MinOrderCost));

}

}

}

public int? MinimKolich

{

get => \_ingredient.MinimKolich;

set

{

if (\_ingredient.MinimKolich != value)

{

\_ingredient.MinimKolich = value;

OnPropertyChanged();

OnPropertyChanged(nameof(MinOrderCost));

}

}

}

public int? KolichUpakovka

{

get => \_ingredient.KolichUpakovka;

set

{

if (\_ingredient.KolichUpakovka != value)

{

\_ingredient.KolichUpakovka = value;

OnPropertyChanged();

OnPropertyChanged(nameof(MinOrderCost));

}

}

}

public int? CostForOne

{

get => \_ingredient.CostForOne;

set

{

if (\_ingredient.CostForOne != value)

{

\_ingredient.CostForOne = value;

OnPropertyChanged();

OnPropertyChanged(nameof(MinOrderCost));

}

}

}

public string Description

{

get => \_ingredient.Description;

set

{

if (\_ingredient.Description != value)

{

\_ingredient.Description = value;

OnPropertyChanged();

}

}

}

public DateTime? ExpirationDate

{

get => \_ingredient.ExpirationDate;

set

{

if (\_ingredient.ExpirationDate != value)

{

\_ingredient.ExpirationDate = value;

OnPropertyChanged();

}

}

}

public decimal MinOrderCost

{

get

{

if (KolichSklad >= MinimKolich ||

!KolichSklad.HasValue ||

!MinimKolich.HasValue ||

!KolichUpakovka.HasValue ||

!CostForOne.HasValue)

{

return 0.00m;

}

decimal deficit = (decimal)(MinimKolich - KolichSklad);

int packages = (int)System.Math.Ceiling(deficit / (decimal)KolichUpakovka);

return packages \* (decimal)KolichUpakovka \* (decimal)CostForOne;

}

}

public Ingredients GetModel() => \_ingredient;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

public string this[string columnName]

{

get

{

switch (columnName)

{

case nameof(IngredientsName):

if (string.IsNullOrWhiteSpace(IngredientsName))

return "Название обязательно";

break;

case nameof(KolichSklad):

if (KolichSklad < 0) return "Не отрицательное";

break;

case nameof(MinimKolich):

if (MinimKolich < 0) return "Не отрицательное";

break;

case nameof(CostForOne):

if (CostForOne < 0) return "Не отрицательное";

break;

}

return null;

}

}

}

}

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Data.Entity;

using System.Runtime.CompilerServices;

using UPMurtazinIS121.Model;

namespace UPMurtazinIS121.ViewModel

{

public class MainViewModel : INotifyPropertyChanged

{

private readonly CoffeeDBMurtazinEntities2 \_context = new();

public ObservableCollection<IngredientModel> IngredientsList { get; } = [];

public MainViewModel()

{

\_context.Ingredients.Load();

foreach (var ingredient in \_context.Ingredients.Local)

{

IngredientsList.Add(new IngredientModel(ingredient));

}

}

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

}

using System;

using System.Windows.Input;

namespace UPMurtazinIS121.Validations

{

public class RelayCommand(Action<object> execute) : ICommand

{

private readonly Action<object> \_execute = execute ?? throw new ArgumentNullException(nameof(execute));

public event EventHandler CanExecuteChanged

{

add => CommandManager.RequerySuggested += value;

remove => CommandManager.RequerySuggested -= value;

}

public bool CanExecute(object parameter) => true;

public void Execute(object parameter) => \_execute(parameter);

}

}

using System.ComponentModel;

using System.Runtime.CompilerServices;

using System.Text.RegularExpressions;

namespace UPMurtazinIS121.Model

{

public class SupplierModel(Suppliers supplier) : INotifyPropertyChanged, IDataErrorInfo

{

private readonly Suppliers \_supplier = supplier;

public string Name

{

get => \_supplier.Name;

set { if (\_supplier.Name != value) { \_supplier.Name = value; OnPropertyChanged(); } }

}

public string Type

{

get => \_supplier.Type;

set { if (\_supplier.Type != value) { \_supplier.Type = value; OnPropertyChanged(); } }

}

public string YuridAdres

{

get => \_supplier.YuridAdres;

set { if (\_supplier.YuridAdres != value) { \_supplier.YuridAdres = value; OnPropertyChanged(); } }

}

public double? INN

{

get => \_supplier.INN;

set { if (\_supplier.INN != value) { \_supplier.INN = value; OnPropertyChanged(); } }

}

public string FIO

{

get => \_supplier.FIO;

set { if (\_supplier.FIO != value) { \_supplier.FIO = value; OnPropertyChanged(); } }

}

public string Phone

{

get => \_supplier.Phone;

set { if (\_supplier.Phone != value) { \_supplier.Phone = value; OnPropertyChanged(); } }

}

public string Email

{

get => \_supplier.Email;

set { if (\_supplier.Email != value) { \_supplier.Email = value; OnPropertyChanged(); } }

}

public string Logo

{

get => \_supplier.Logo;

set { if (\_supplier.Logo != value) { \_supplier.Logo = value; OnPropertyChanged(); } }

}

public decimal? RatingReliability

{

get => \_supplier.RatingReliability;

set { if (\_supplier.RatingReliability != value) { \_supplier.RatingReliability = value; OnPropertyChanged(); } }

}

public Suppliers GetModel() => \_supplier;

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged([CallerMemberName] string propertyName = null)

=> PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

public string this[string columnName]

{

get

{

switch (columnName)

{

case nameof(INN):

var s = INN?.ToString() ?? "";

if (s.Length != 12) return "ИНН — ровно 12 цифр";

break;

case nameof(Phone):

// разрешаем +, цифры, скобки, дефис

if (!Regex.IsMatch(Phone ?? "",

@"^**\+**7**\(**\d{3}**\)**\d{3}-\d{2}-\d{2}$"))

return "Телефон: +7(000)000-00-00";

break;

case nameof(Email):

if (!Regex.IsMatch(Email ?? "",

@"^[^@\s]+@[^@\s]+**\.**[^@\s]+$"))

return "Неверный e-mail";

break;

case nameof(RatingReliability):

if (RatingReliability < 0m || RatingReliability > 10m)

return "Рейтинг 0.00–10.00";

break;

}

return null;

}

}

public string Error => null;

}

}

using System;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Windows;

using System.Windows.Input;

using UPMurtazinIS121.Model;

using UPMurtazinIS121.Validations;

namespace UPMurtazinIS121.ViewModel

{

public class SuppliersViewModel : INotifyPropertyChanged

{

private readonly CoffeeDBMurtazinEntities2 \_context = new();

private SupplierModel \_selectedSupplier;

private string \_selectedFilterType;

public ObservableCollection<SupplierModel> SuppliersList { get; } = [];

public ObservableCollection<string> SupplierTypes { get; } = [];

public ObservableCollection<SupplierModel> FilteredSuppliersList { get; } = [];

public ICommand SortAscCommand { get; }

public ICommand SortDescCommand { get; }

public string SelectedFilterType

{

get => \_selectedFilterType;

set { \_selectedFilterType = value; OnPropertyChanged(); ApplyFilter(); }

}

public SupplierModel SelectedSupplier

{

get => \_selectedSupplier;

set

{

\_selectedSupplier = value;

OnPropertyChanged();

}

}

public ICommand SaveCommand { get; }

public ICommand AddNewCommand { get; }

public ICommand DeleteCommand { get; }

public SuppliersViewModel()

{

\_context.Configuration.ProxyCreationEnabled = false;

LoadData();

SaveCommand = new RelayCommand(SaveChanges);

AddNewCommand = new RelayCommand(AddNewSupplier);

DeleteCommand = new RelayCommand(DeleteSupplier);

SortAscCommand = new RelayCommand(\_ => SortSuppliers(true));

SortDescCommand = new RelayCommand(\_ => SortSuppliers(false));

}

private void SortSuppliers(bool ascending)

{

var sorted = ascending

? [.. FilteredSuppliersList.OrderBy(s => s.Name)]

: FilteredSuppliersList.OrderByDescending(s => s.Name).ToList();

FilteredSuppliersList.Clear();

foreach (var s in sorted)

FilteredSuppliersList.Add(s);

}

private void LoadData()

{

try

{

\_context.Suppliers.Load();

SuppliersList.Clear();

SupplierTypes.Clear();

SupplierTypes.Add("Все типы");

foreach (var supplier in \_context.Suppliers.Local)

{

var vm = new SupplierModel(supplier);

SuppliersList.Add(vm);

if (!string.IsNullOrEmpty(vm.Type) && !SupplierTypes.Contains(vm.Type))

SupplierTypes.Add(vm.Type);

}

ApplyFilter();

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка загрузки: {ex.Message}", " Ошибка");

}

}

private void ApplyFilter()

{

FilteredSuppliersList.Clear();

var items = SelectedFilterType switch

{

null or "Все типы" => SuppliersList,

\_ => SuppliersList.Where(i => i.Type == SelectedFilterType)

};

foreach (var item in items)

FilteredSuppliersList.Add(item);

}

public bool HasChanges()

{

return \_context.ChangeTracker.Entries()

.Any(e => e.State != EntityState.Unchanged);

}

private void SaveChanges(object parameter)

{

try

{

if (HasChanges())

{

\_context.SaveChanges();

MessageBox.Show("Изменения сохранены!", "Успех",

MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка сохранения: {ex.Message}", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void AddNewSupplier(object parameter)

{

var newSupplier = new Suppliers

{

Name = "Новый поставщик",

Type = "Производитель",

YuridAdres = "Адрес",

INN = 0,

FIO = "ФИО",

Phone = "Телефон",

Email = "email@example.com"

};

var wrapper = new SupplierModel(newSupplier);

SuppliersList.Add(wrapper);

SelectedSupplier = wrapper;

\_context.Suppliers.Add(newSupplier);

if (!SupplierTypes.Contains(wrapper.Type))

SupplierTypes.Add(wrapper.Type);

}

private void DeleteSupplier(object parameter)

{

if (SelectedSupplier == null) return;

if (MessageBox.Show($"Удалить {SelectedSupplier.Name}?", "Подтверждение",

MessageBoxButton.YesNo, MessageBoxImage.Question) != MessageBoxResult.Yes) return;

try

{

\_context.Suppliers.Remove(SelectedSupplier.GetModel());

SuppliersList.Remove(SelectedSupplier);

\_context.SaveChanges();

SelectedSupplier = SuppliersList.FirstOrDefault();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка удаления: {ex.Message}", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged([CallerMemberName] string propertyName = null)

=> PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

<Application x:Class="UPMurtazinIS121.App"

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

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

xmlns:local="clr-namespace:UPMurtazinIS121"

StartupUri="MainWindow.xaml">

<Application.Resources>

<Style TargetType="ComboBox" BasedOn="{StaticResource {x:Type ComboBox}}">

<Setter Property="Background" Value="White"/>

<Setter Property="BorderThickness" Value="0"/>

<Setter Property="BorderBrush" Value="#D7CCC8"/>

<Setter Property="Margin" Value="0,0,0,10"/>

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

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

<Setter Property="Height" Value="25"/>

<Setter Property="FontFamily" Value="Roboto"/>

<Style.Triggers>

<Trigger Property="IsFocused" Value="True">

<Setter Property="BorderBrush" Value="#5D4037"/>

</Trigger>

</Style.Triggers>

</Style>

<Style TargetType="TextBox" BasedOn="{StaticResource {x:Type TextBox}}">

<Setter Property="BorderThickness" Value="0"/>

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

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

<Setter Property="Margin" Value="0,0,0,10"/>

<Setter Property="FontFamily" Value="Roboto"/>

<Setter Property="Validation.ErrorTemplate">

<Setter.Value>

<ControlTemplate>

<DockPanel>

<Border BorderBrush="Red" BorderThickness="1">

<AdornedElementPlaceholder/>

</Border>

</DockPanel>

</ControlTemplate>

</Setter.Value>

</Setter>

<Style.Triggers>

<Trigger Property="Validation.HasError" Value="true">

<Setter Property="ToolTip"

Value="{Binding RelativeSource={RelativeSource Self}, Path=(Validation.Errors)[0].ErrorContent}"/>

</Trigger>

</Style.Triggers>

</Style>

<Color x:Key="PrimaryBackground">#F9F5F0</Color>

<Color x:Key="SecondaryBackground">#D7CCC8</Color>

<Color x:Key="AccentColor">#5D4037</Color>

<Color x:Key="LightTextColor">#FFF8F0</Color>

<SolidColorBrush x:Key="PrimaryBackgroundBrush" Color="{StaticResource PrimaryBackground}"/>

<SolidColorBrush x:Key="SecondaryBackgroundBrush" Color="{StaticResource SecondaryBackground}"/>

<SolidColorBrush x:Key="AccentBrush" Color="{StaticResource AccentColor}"/>

<SolidColorBrush x:Key="LightTextBrush" Color="{StaticResource LightTextColor}"/>

<SolidColorBrush x:Key="DarkTextBrush" Color="{StaticResource AccentColor}"/>

<Style TargetType="Window">

<Setter Property="Background" Value="{StaticResource PrimaryBackgroundBrush}"/>

<Setter Property="FontFamily" Value="Roboto"/>

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

<Setter Property="Foreground" Value="{StaticResource DarkTextBrush}"/>

</Style>

<Style TargetType="TextBlock">

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

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

<Setter Property="FontFamily" Value="Roboto"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

<Style x:Key="ColumnHeaderStyle" TargetType="GridViewColumnHeader">

<Setter Property="Background" Value="{StaticResource AccentBrush}"/>

<Setter Property="Foreground" Value="{StaticResource LightTextBrush}"/>

<Setter Property="FontWeight" Value="Bold"/>

<Setter Property="FontFamily" Value="Roboto"/>

<Setter Property="Padding" Value="10,5"/>

<Setter Property="HorizontalContentAlignment" Value="Center"/>

<Setter Property="BorderThickness" Value="0"/>

</Style>

<Style x:Key="StyledListView" TargetType="ListView">

<Setter Property="Background" Value="Transparent"/>

<Setter Property="FontFamily" Value="Roboto"/>

<Setter Property="BorderThickness" Value="0"/>

<Setter Property="Margin" Value="10"/>

</Style>

<Style x:Key="ListViewItemStyle" TargetType="ListViewItem">

<Setter Property="Background" Value="Transparent"/>

<Setter Property="FontFamily" Value="Roboto"/>

<Setter Property="BorderThickness" Value="0"/>

<Setter Property="FocusVisualStyle" Value="{x:Null}"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="ListViewItem">

<Border

Background="{TemplateBinding Background}"

BorderThickness="0"

Padding="0">

<ContentPresenter />

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

<Style.Triggers>

<Trigger Property="ItemsControl.AlternationIndex" Value="0">

<Setter Property="Background" Value="#F9F5F0"/>

</Trigger>

<Trigger Property="ItemsControl.AlternationIndex" Value="1">

<Setter Property="Background" Value="#E8E0D9"/>

</Trigger>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="Background" Value="#D7CCC8"/>

</Trigger>

<Trigger Property="IsSelected" Value="True">

<Setter Property="Background" Value="#A1887F"/>

</Trigger>

</Style.Triggers>

</Style>

</Application.Resources>

</Application>

using System;

using System.Globalization;

using System.Windows.Data;

using System.Windows.Media;

namespace UPMurtazinIS121

{

public class BoolToBrushConverter : IValueConverter

{

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

{

return (bool)value ? Brushes.LightGreen : Brushes.Transparent;

}

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

{

throw new NotImplementedException();

}

}

public class InverseBoolToBrushConverter : IValueConverter

{

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

{

return (bool)value ? Brushes.Transparent : Brushes.LightGreen;

}

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

{

throw new NotImplementedException();

}

}

}

<Window x:Class="UPMurtazinIS121.MainWindow"

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

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

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

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

xmlns:local="clr-namespace:UPMurtazinIS121"

xmlns:locall="clr-namespace:UPMurtazinIS121.ViewModel"

xmlns:localll="clr-namespace:UPMurtazinIS121.Validations"

mc:Ignorable="d"

Title="Учет ингредиентов - Амброзия Богов"

MinHeight="800" MinWidth="1300"

WindowStartupLocation="CenterScreen">

<Window.DataContext>

<locall:IngredientsViewModel/>

</Window.DataContext>

<Grid Margin="10">

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center" Height="50" VerticalAlignment="Top">

<Image Source="/logo\_coffee\_paradise.png"

Width="50"

Height="50"

VerticalAlignment="Top"/>

<TextBlock Text="УЧЕТ ИНГРЕДИЕНТОВ"

FontSize="24" FontWeight="Bold"

Foreground="{StaticResource AccentBrush}"

HorizontalAlignment="Center"

VerticalAlignment="Top"

Margin="25,10"/>

</StackPanel>

<Grid Margin="0,50,0,50">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="10"/>

<ColumnDefinition Width="1.2\*"/>

</Grid.ColumnDefinitions>

<!-- Левая часть - Список ингредиентов -->

<Border Grid.Column="0"

Background="{StaticResource SecondaryBackgroundBrush}"

CornerRadius="5"

Padding="10">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!-- Фильтр по типу -->

<StackPanel Grid.Row="0" Orientation="Horizontal" Margin="0,0,0,10">

<TextBlock Text="Фильтр по типу:" VerticalAlignment="Center" Margin="0,0,10,0"/>

<ComboBox ItemsSource="{Binding IngredientTypes}"

SelectedItem="{Binding SelectedFilterType}"

Width="200"

Height="35"

IsEditable="False">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<Button Content="А-Я" Command="{Binding SortAscCommand}" Width="60" Margin="20,0,5,0"

Background="#5D4037" Foreground="White"/>

<Button Content="Я-А" Command="{Binding SortDescCommand}" Width="60" Margin="5,0,5,0"

Background="#5D4037" Foreground="White"/>

</StackPanel>

<ListView Grid.Row="1"

ItemsSource="{Binding FilteredIngredientsList}"

SelectedItem="{Binding SelectedIngredient}"

Background="{StaticResource SecondaryBackgroundBrush}"

BorderThickness="0"

HorizontalContentAlignment="Stretch"

ScrollViewer.CanContentScroll="False"

ScrollViewer.VerticalScrollBarVisibility="Auto">

<ListView.ItemTemplate>

<DataTemplate>

<Border Background="White" CornerRadius="4" Padding="8" Margin="0,0,0,10">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<!-- Название -->

<RowDefinition Height="Auto"/>

<!-- Первая строка данных -->

<RowDefinition Height="Auto"/>

<!-- Вторая строка данных -->

<RowDefinition Height="Auto"/>

<!-- Третья строка данных -->

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*" />

<ColumnDefinition Width="2\*" />

<ColumnDefinition Width="2\*" />

<ColumnDefinition Width="2\*" />

</Grid.ColumnDefinitions>

<!-- Название ингредиента -->

<TextBlock Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="4"

Text="{Binding IngredientsName}"

FontWeight="Bold" FontSize="16"

Margin="0,0,0,8"/>

<!-- Первая строка: Тип и Мин. кол-во -->

<TextBlock Grid.Row="1" Grid.Column="0" Text="Тип:" FontStyle="Italic"/>

<TextBlock Grid.Row="1" Grid.Column="1" Text="{Binding TypeIngredients}" />

<TextBlock Grid.Row="1" Grid.Column="2" Text="Мин. количество:" FontStyle="Italic"/>

<TextBlock Grid.Row="1" Grid.Column="3" Text="{Binding MinimKolich}" />

<!-- Вторая строка: Кол-во на складе и в упаковке -->

<TextBlock Grid.Row="2" Grid.Column="0" Text="В наличии:" FontStyle="Italic"/>

<TextBlock Grid.Row="2" Grid.Column="1">

<TextBlock.Text>

<MultiBinding StringFormat="{}{0:N2} {1}">

<Binding Path="KolichSklad"/>

<Binding Path="UnitOfMeasurement"/>

</MultiBinding>

</TextBlock.Text>

</TextBlock>

<TextBlock Grid.Row="2" Grid.Column="2" Text="Упаковка:" FontStyle="Italic"/>

<TextBlock Grid.Row="2" Grid.Column="3" Text="{Binding KolichUpakovka}" />

<!-- Третья строка: Цена и мин. партия -->

<TextBlock Grid.Row="3" Grid.Column="0" Text="Цена:" FontStyle="Italic"/>

<TextBlock Grid.Row="3" Grid.Column="1" Text="{Binding CostForOne}" />

<TextBlock Grid.Row="3" Grid.Column="2" Text="Мин. партия:" FontStyle="Italic"/>

<TextBlock Grid.Row="3" Grid.Column="3" Text="{Binding MinOrderCost}" />

</Grid>

</Border>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</Grid>

</Border>

<GridSplitter Grid.Column="1" Width="6" Background="{StaticResource AccentBrush}" IsEnabled="False"/>

<!-- Правая часть - Редактирование -->

<Border Grid.Column="2"

Background="{StaticResource SecondaryBackgroundBrush}"

CornerRadius="5"

Padding="15">

<StackPanel>

<TextBlock Text="Редактирование ингредиента"

FontSize="18" FontWeight="Bold"

HorizontalAlignment="Center"

Margin="0,0,0,15"

FontFamily="Roboto"/>

<Label Content="Название:"/>

<TextBox Text="{Binding SelectedIngredient.IngredientsName, UpdateSourceTrigger=PropertyChanged}"/>

<Label Content="Тип:"/>

<ComboBox ItemsSource="{Binding IngredientTypes}"

SelectedItem="{Binding SelectedIngredient.TypeIngredients, UpdateSourceTrigger=PropertyChanged}"

IsEditable="True" IsReadOnly="True">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<Label Content="Ед. измерения:"/>

<!--<TextBox Text="{Binding SelectedIngredient.UnitOfMeasurement, UpdateSourceTrigger=PropertyChanged}"/>-->

<ComboBox ItemsSource="{Binding MeasurementUnits}"

SelectedItem="{Binding SelectedIngredient.UnitOfMeasurement, UpdateSourceTrigger=PropertyChanged}"

IsEditable="True" IsReadOnly="True">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<Label Content="На складе:"/>

<TextBox Text="{Binding SelectedIngredient.KolichSklad, StringFormat={}{0:N2}, UpdateSourceTrigger=PropertyChanged}"/>

<Label Content="Мин. кол-во:"/>

<TextBox Text="{Binding SelectedIngredient.MinimKolich, UpdateSourceTrigger=PropertyChanged}"/>

<Label Content="Цена за ед.:"/>

<TextBox Text="{Binding SelectedIngredient.CostForOne, StringFormat={}{0:N2}, UpdateSourceTrigger=PropertyChanged}"/>

<Label Content="Стоимость мин. партии:" FontWeight="Bold"/>

<TextBlock Text="{Binding SelectedIngredient.MinOrderCost, StringFormat={}{0:N2} руб.}"

FontSize="16" FontWeight="Bold" Margin="0,0,0,20"/>

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center">

<Button Content="СОХРАНИТЬ ВСЕ"

Command="{Binding SaveCommand}"

Background="#5D4037"

Foreground="White"

FontWeight="Bold"

Margin="0,0,10,0"

Padding="20,10"

MinWidth="180"/>

<Button Content="+ ДОБАВИТЬ НОВЫЙ"

Command="{Binding AddNewCommand}"

Background="#689F38"

Foreground="White"

FontWeight="Bold"

Padding="20,10"

MinWidth="180"/>

<Button Content="Удалить" Command="{Binding DeleteCommand}"

Background="#D32F2F"

Foreground="White"

FontWeight="Bold"

Margin="10,0,0,0"

Padding="20,10"

MinWidth="180"/>

<Button Command="{Binding ExportToExcelCommand}"

ToolTip="Экспорт в Excel"

Margin="10,0,0,0"

Padding="20,10"

Height="48">

<Image Source="/Microsoft\_Office\_Excel\_.png"/>

</Button>

</StackPanel>

</StackPanel>

</Border>

</Grid>

<StatusBar VerticalAlignment="Bottom" Background="{StaticResource AccentBrush}">

<StatusBarItem>

<TextBlock Text="Амброзия Богов © 2025" Foreground="{StaticResource LightTextBrush}"/>

</StatusBarItem>

<StatusBarItem HorizontalAlignment="Center">

<Button Content="Поставщики"

Margin="10"

Padding="10,5"

Click="SuppliersButton\_Click"/>

</StatusBarItem>

<StatusBarItem HorizontalAlignment="Right">

<StackPanel Orientation="Horizontal">

<TextBlock Text="Всего ингредиентов: " Foreground="{StaticResource LightTextBrush}"/>

<TextBlock Text="{Binding IngredientsList.Count}" Foreground="{StaticResource LightTextBrush}"/>

</StackPanel>

</StatusBarItem>

</StatusBar>

</Grid>

</Window>

<Window x:Class="UPMurtazinIS121.SuppliersWindow"

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

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

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

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

xmlns:local="clr-namespace:UPMurtazinIS121"

xmlns:locall="clr-namespace:UPMurtazinIS121.Validations"

xmlns:vm="clr-namespace:UPMurtazinIS121.ViewModel"

mc:Ignorable="d"

Title="Управление поставщиками - Амброзия Богов"

MinHeight="800" MinWidth="1300"

WindowStartupLocation="CenterScreen">

<Window.DataContext>

<vm:SuppliersViewModel/>

</Window.DataContext>

<Grid Margin="10">

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center" Height="50" VerticalAlignment="Top">

<Image Source="/logo\_coffee\_paradise.png"

Width="50"

Height="50"

VerticalAlignment="Top"/>

<TextBlock Text="УПРАВЛЕНИЕ ПОСТАВЩИКАМИ"

FontSize="24" FontWeight="Bold"

Foreground="{StaticResource AccentBrush}"

HorizontalAlignment="Center"

VerticalAlignment="Top"

Margin="25,10"/>

</StackPanel>

<Grid Margin="0,50,0,50">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="10"/>

<ColumnDefinition Width="1.2\*"/>

</Grid.ColumnDefinitions>

<!-- Левая часть - Список поставщиков -->

<Border Grid.Column="0"

Background="{StaticResource SecondaryBackgroundBrush}"

CornerRadius="5"

Padding="10">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!-- Фильтр по типу -->

<StackPanel Grid.Row="0" Orientation="Horizontal" Margin="0,0,0,10">

<TextBlock Text="Фильтр по типу:" VerticalAlignment="Center" Margin="0,0,10,0"/>

<ComboBox ItemsSource="{Binding SupplierTypes}"

SelectedItem="{Binding SelectedFilterType}"

Width="200"

Height="35"

IsEditable="False">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<Button Content="А-Я" Command="{Binding SortAscCommand}" Width="60" Margin="0,0,5,0"/>

<Button Content="Я-А" Command="{Binding SortDescCommand}" Width="60"/>

</StackPanel>

<ListView Grid.Row="1"

ItemsSource="{Binding FilteredSuppliersList}"

SelectedItem="{Binding SelectedSupplier}"

Background="{StaticResource SecondaryBackgroundBrush}"

BorderThickness="0"

HorizontalContentAlignment="Stretch"

ScrollViewer.CanContentScroll="False"

ScrollViewer.VerticalScrollBarVisibility="Auto">

<ListView.ItemTemplate>

<DataTemplate>

<Border Background="White" CornerRadius="4" Padding="8" Margin="0,0,0,10">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*" />

<ColumnDefinition Width="2\*" />

<ColumnDefinition Width="2\*" />

<ColumnDefinition Width="2\*" />

</Grid.ColumnDefinitions>

<!-- Название поставщика -->

<TextBlock Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="4"

Text="{Binding Name}"

FontWeight="Bold" FontSize="16"

Margin="0,0,0,8"/>

<!-- Первая строка: Тип и ИНН -->

<TextBlock Grid.Row="1" Grid.Column="0" Text="Тип:" FontStyle="Italic"/>

<TextBlock Grid.Row="1" Grid.Column="1" Text="{Binding Type}" />

<TextBlock Grid.Row="1" Grid.Column="2" Text="ИНН:" FontStyle="Italic"/>

<TextBlock Grid.Row="1" Grid.Column="3" Text="{Binding INN}" />

<!-- Вторая строка: ФИО и Телефон -->

<TextBlock Grid.Row="2" Grid.Column="0" Text="ФИО:" FontStyle="Italic"/>

<TextBlock Grid.Row="2" Grid.Column="1" Text="{Binding FIO}" />

<TextBlock Grid.Row="2" Grid.Column="2" Text="Телефон:" FontStyle="Italic"/>

<TextBlock Grid.Row="2" Grid.Column="3" Text="{Binding Phone}" />

<!-- Третья строка: Email и Рейтинг -->

<TextBlock Grid.Row="3" Grid.Column="0" Text="Email:" FontStyle="Italic"/>

<TextBlock Grid.Row="3" Grid.Column="1" Text="{Binding Email}" />

<TextBlock Grid.Row="3" Grid.Column="2" Text="Рейтинг:" FontStyle="Italic"/>

<TextBlock Grid.Row="3" Grid.Column="3" Text="{Binding RatingReliability}" />

</Grid>

</Border>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</Grid>

</Border>

<GridSplitter Grid.Column="1" Width="6" Background="{StaticResource AccentBrush}" IsEnabled="False"/>

<!-- Правая часть - Редактирование -->

<Border Grid.Column="2"

Background="{StaticResource SecondaryBackgroundBrush}"

CornerRadius="5"

Padding="15">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<!-- Область редактирования с прокруткой -->

<ScrollViewer Grid.Row="0" VerticalScrollBarVisibility="Auto">

<StackPanel>

<TextBlock Text="Редактирование поставщика"

FontSize="18" FontWeight="Bold"

HorizontalAlignment="Center"

Margin="0,0,0,15"/>

<Label Content="Название:"/>

<TextBox Text="{Binding SelectedSupplier.Name, UpdateSourceTrigger=PropertyChanged}"/>

<Label Content="Тип:"/>

<ComboBox ItemsSource="{Binding SupplierTypes}"

SelectedItem="{Binding SelectedSupplier.Type, UpdateSourceTrigger=PropertyChanged}"

IsEditable="True" IsReadOnly="True">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<Label Content="Юридический адрес:"/>

<TextBox Text="{Binding SelectedSupplier.YuridAdres, UpdateSourceTrigger=PropertyChanged}"/>

<Label Content="ИНН:"/>

<TextBox Text="{Binding SelectedSupplier.INN, UpdateSourceTrigger=PropertyChanged}"/>

<Label Content="ФИО представителя:"/>

<TextBox Text="{Binding SelectedSupplier.FIO, UpdateSourceTrigger=PropertyChanged}"/>

<Label Content="Телефон:"/>

<TextBox Text="{Binding SelectedSupplier.Phone, UpdateSourceTrigger=PropertyChanged}"/>

<Label Content="Email:"/>

<TextBox Text="{Binding SelectedSupplier.Email, UpdateSourceTrigger=PropertyChanged}"/>

<Label Content="Рейтинг надежности:"/>

<TextBox Text="{Binding SelectedSupplier.RatingReliability, UpdateSourceTrigger=PropertyChanged}"/>

</StackPanel>

</ScrollViewer>

<!-- Кнопки управления (фиксированные внизу) -->

<StackPanel Grid.Row="1" Orientation="Horizontal" HorizontalAlignment="Center" Margin="0,20,0,0">

<Button Content="СОХРАНИТЬ ВСЕ"

Command="{Binding SaveCommand}"

Background="#5D4037"

Foreground="White"

FontWeight="Bold"

Margin="0,0,10,0"

Padding="20,10"

MinWidth="180"/>

<Button Content="+ ДОБАВИТЬ НОВЫЙ"

Command="{Binding AddNewCommand}"

Background="#689F38"

Foreground="White"

FontWeight="Bold"

Padding="20,10"

MinWidth="180"/>

<Button Content="Удалить"

Command="{Binding DeleteCommand}"

Background="#D32F2F"

Foreground="White"

FontWeight="Bold"

Margin="10,0,0,0"

Padding="20,10"

MinWidth="180"/>

</StackPanel>

</Grid>

</Border>

</Grid>

<StatusBar VerticalAlignment="Bottom" Background="{StaticResource AccentBrush}">

<StatusBarItem>

<TextBlock Text="Амброзия Богов © 2025" Foreground="{StaticResource LightTextBrush}"

Margin="10"

Padding="10,5"/>

</StatusBarItem>

<StatusBarItem HorizontalAlignment="Right">

<StackPanel Orientation="Horizontal">

<TextBlock Text="Всего поставщиков: " Foreground="{StaticResource LightTextBrush}"/>

<TextBlock Text="{Binding SuppliersList.Count}" Foreground="{StaticResource LightTextBrush}"/>

</StackPanel>

</StatusBarItem>

</StatusBar>

</Grid>

</Window>