Министерство образования Московской области

ГБПОУ МО «Колледж «Коломна»

09.02.07

КУРСОВОЙ ПРОЕКТ

ПМ. 02 Осуществление интеграции программных модулей

Тема: Разработка информационной системы «Пиццерия Итальянский двор»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

ККОО.КП1201.000ПЗ

|  |  |
| --- | --- |
| Студент: | Безуглов В.А. |
| Руководитель: | Карташова Е.В. |
| Нормоконтролер: | Грушникова Т.Н. |
| Дата защиты: | Оценка |

2024

Министерство образования московской области

ГБПОУ МО «Колледж «Коломна»

ЗАДАНИЕ

на курсовой проект

ПМ.02 Осуществление интеграции программных модулей

студента очной формы обучения 4 курса группы 42112

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема: Разработка информационной системы «Ремонт электронной техники»

Разработать информационную систему в соответствии с темой и приложение для работы с ней. Разработанная информационная система должна отражать основные бизнес-процессы в соответствии с техническим заданием на разработку. Для защиты информации необходимо разграничить права пользователей системы. Разработанные формы должны обеспечить возможность пользователям осуществлять работу с данными в соответствии с их правами.

При разработке информационной системы необходимо решить следующие задачи:

* сформулировать цель разработки информационной системы;
* описать предметную область, для которой разрабатывается информационная система
* провести анализ предметной области, выявить основные бизнес-процессы и пользователей, которые будут с ней работать.
* определить круг запросов и задач, которые предполагается решать с использованием созданной информационной системы;
* написать техническое задание на разработку информационной системы;
* провести функциональное моделирование и объектно-ориентированное проектирование информационной системы;
* провести проектирование интерфейса информационной системы:
* разработать модель данных и реализовать ее в выбранной СУБД
* разработать программное решение для информационной системы;
* оценить качество разработанной системы путем тестирования основного функционала.

Объем курсовой работы:

* + 1. Пояснительная записка (25-50 листов печатного текста формата А4):

Введение

1. Разработка системного проекта

1.1 Описание предметной области

1.2 Анализ предметной области

1.3 Техническое задание

2 Проектирование информационной системы

2.1 Функциональное моделирование

2.2 Проектирование интерфейсов приложения

3 Разработка информационной системы

3.1 Модель данных

3.2 Структура проекта

4 Оценка качества программного обеспечения

Выводы и заключение

Список литературы

Приложение А – Код программных модулей

* + 1. Презентация для защиты курсового проекта (10-15 слайдов)

Дата выдачи задания \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Срок окончания \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Зав структурным подразделением \_\_\_\_\_\_\_\_\_\_ Емельянова В. А.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_ Карташова Е.В.

СОДЕржАние

|  |  |
| --- | --- |
| Введение |  |
| 1. Разработка системного проекта | 6 |
| 1.1 Описание предметной области |  |
| 1.2 Анализ предметной области | 7 |
| 1.3 Техническое задание |  |
| 2 Проектирование информационной системы | 13 |
| 2.1 Функциональное моделирование |  |
| 2.2 Проектирование интерфейсов приложения | 16 |
| 3 Разработка информационной системы | 23 |
| 3.1 Модель данных |  |
| 3.2 Структура проекта | 27 |
| 4 Оценка качества программного обеспечения | 32 |
| Выводы и заключение | 36 |
| Список литературы | 37 |
| Приложение А – Код программных модулей | 38 |

Целью курсового проекта является.

При разработке информационной системы необходимо решить следующие задачи:

* сформулировать цель разработки информационной системы;
* описать предметную область, для которой разрабатывается информационная система
* провести анализ предметной области, выявить основные бизнес-процессы и пользователей, которые будут с ней работать.
* определить круг запросов и задач, которые предполагается решать с использованием созданной информационной системы;
* написать техническое задание на разработку информационной системы;
* провести функциональное моделирование и объектно-ориентированное проектирование информационной системы;
* провести проектирование интерфейса информационной системы:
* разработать модель данных и реализовать ее в выбранной СУБД
* разработать программное решение для информационной системы;
* оценить качество разработанной системы путем тестирования основного функционала.

1. РАЗРАБОТКА СИСТЕМНОГО ПРОЕКТА
   1. Описание предметной области

Пиццерия "Итальянский двор" — это место, где каждый может насладиться вкусом настоящей итальянской пиццы, приготовленной по традиционным рецептам. В современном мире, где быстрый ритм жизни часто оставляет мало времени на приготовление пищи дома, посещение пиццерии становится не просто удовольствием, а важным элементом повседневной жизни. Заказ пиццы навынос или доставка на дом позволяют сэкономить время и силы, не жертвуя качеством и вкусом блюд.

Кроме того, пиццерия "Итальянский двор" играет важную роль в поддержании культурных традиций и кулинарного наследия Италии. Использование свежих и качественных ингредиентов, а также соблюдение классических рецептов, позволяет предложить гостям аутентичный вкус итальянской кухни. Это не только радует гурманов, но и способствует сохранению культурного разнообразия в гастрономическом мире.

Пиццерия также уделяет большое внимание экологическим аспектам. Использование местных продуктов и поддержка фермеров помогает снизить углеродный след и способствует устойчивому развитию. Кроме того, упаковка для доставки выбирается с учетом экологических стандартов, что позволяет минимизировать количество отходов и защитить окружающую среду.

Таким образом, пиццерия "Итальянский двор" не только предлагает вкусные и качественные блюда, но и вносит свой вклад в сохранение культурных традиций и защиту природы. Посещение этой пиццерии — это возможность насладиться настоящей итальянской кухней, поддержать местные продукты и сделать шаг к более устойчивому образу жизни.

* 1. Анализ предметной области

Цель информационной системы пиццери Итальянский двор – автоматизация деятельности пиццерии.

Задачи данной информационной системы:

1. Автоматизация регистрации данных о клиенте. Необходимо, чтобы данные о клиенте сохранялись в нашей ИС. Эти данные используются для идентификации клиента.

2. Автоматизация регистрации заказов клиентов.

Необходимо хранить все сведения по заказам клиентов. Эти сведения используются как для оформления общего отчета, так и для дальнейшего анализа складского учета.

3. Автоматизация складского учета. Необходимо в ИС заносить факты принятых и выданных продуктов, составлять общий отчеты по продажам и доставкам.

4. Автоматизация бухгалтерского учета. Необходимо в ИС вести бухгалтерский учет согласно действующему законодательству.

Список пользователей ИС ремонта электронной техники:

– администратор;

* 1. Техническое задание

Введение

Пиццерия Итальянский двор

Настоящее техническое задание распространяется на разработку автоматизированной информационной системы «Пиццерия Итальянский двор», предназначенной для автоматизации учета движения товаров на складе.

Целью создания автоматизированной информационной системы является сокращение времени на получение сведений о наличии товаров на складе, на формирование отчетности по поступлению и реализации товаров, учета остатков товара на складе и т.д. Система обеспечивает повышение эффективности процесса обработки и движения документов, снижает затраты на ведение документооборота, повышает точность и достоверность данных.

Работа выполняется в рамках проекта автоматизации управления торговым предприятием.

Основание для разработки

Основанием для разработки является договор № \_\_\_\_\_\_\_ от \_\_\_\_\_\_\_

Организация, утвердившая договор: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Наименование работы: Информационная система «Сервисный центр по ремонту электронной техники»

Назначение разработки.

Информационная система «Пиццерия Итальянский двор» предназначена для автоматизации учета движения и наличия товаров, а также для обработки заявок на обслуживание от юридических и физических лиц. Пользователями системы являются менеджеры склада, специалисты отдела учета, сотрудники отдела приема и оформления документов. Система должна обеспечить учет товаров, поступивших на склад для продажи или обслуживания, а также фиксацию всех операций, связанных с их перемещением, продажей клиентам пицерии и учетом совершенной доставки.  
Пользователи системы:

1. Менеджеры склада — отвечают за приемку и выдачу товаров для ремонта, их хранение и учет.

2. Специалисты отдела учета — ведут учетные записи всех операций по поступлению, ремонту и выдаче товаров, осуществляют проверку данных.

3. Сотрудники отдела приема и оформления документов — занимаются оформлением первичных документов на прием товаров в ремонт, а также контролем за их исправностью и возвратом клиентам. Основные функции системы:

1. Учет поступления товаров на склад по пиццерии:

* Оформление и учет поступления товаров от производителей (физических и юридических лиц) на основании документов (заявок на закупку, накладная, акт приемки).
* Регистрация условий приемки товаров (дата, документ, количество).
* Ведение журнала поступления товаров, содержащего следующие данные: название документа, его дату и номер, описание товара, количество.

2. Оформление заказа и обслуживания:

* Регистрация заказов через сайт или приложение.
* Оформление этапов заказа (принятие, приготовление, продажа).

3. Учет выдачи товаров после выполнениния заказа:

* Оформление документов после продажи товара клиенту.
* Ведение журнала учета сбыта товаров, в котором фиксируются: номер по порядку, дата выдачи, наименование товара, количество, описание произведенных работ на кухне.

Требования к программе

3.1. Требования к функциональным характеристикам

Автоматизированная информационная система «Пиццерия Итальянский дворик» должна обеспечивать выполнение следующих функций:

* ввод, хранение, поиск и обработку информации о поступлении товаров для продажи, выполненных заказах и доставки заказа;
* ведение журнала доставки;
* своевременное получение информации о наличии товаров на складе;

Нормативно-справочная информация автоматизированной информационной системы «Пиццери Итальянский двор» представлена справочниками:

а) клиентов (юридические и физические лица);

б) товаров, подлежащих продаже и доставке;

Первичными документами для учета товаров в системе являются:

Приходные данные:

* данные о товаре;
* данные о клинента;
* данные доставки.

Выходными данными являются:

* список с клиентами;
* список с данными доставок;
* список с ассортиментом.

Все отчеты должны формироваться на даты, заданные пользователем, и содержать итоговые значения. В приложении к техническому заданию требуется включить образцы всех первичных документов и образцы всех отчетов.

В программе необходимо предусмотреть:

* Возможность клиенту управлять интерфейсом;
* Формирования списка заказов;
* Пополнить склад;
* Формирование списка ассортимента.

3.2. Требования к надежности

Программное обеспечение должно соответствовать требованиям надежности и безопасности, включая:

* возможность восстановления после сбоев (отключение питания, сбои в операционной системе и т.д.);
* парольную защиту при запуске программы; ограничение несанкционированного доступа к данным;
* возможность резервного копирования информационной базы;
* разграничение пользовательских прав;
* предотвращение несанкционированного копирования программы.

Необходимо предусмотреть контроль вводимой информации, обработку или блокировку некорректных действий пользователя. Все поля первичных документов должны быть заполнены.

3.3. Требования к составу и параметрам технических средств

Минимальные системные требования для работы программного продукта:

* тактовая частота процессора – 2.1 GHz;
* объем оперативной памяти – 4 ГБ;
* объем свободного дискового пространства 15 ГБ;
* разрешение монитора – 1920 x 1080 px

3.4. Требования к информационной и программной совместимости

Программа должна работать в операционных системах Windows... Все формируемые отчеты должны иметь возможность экспортирования в редактор электронных таблиц MS Office Excel.

3.5. Требования к транспортировке и хранению

Установка программы осуществляется через установочные архивные файлы, доступные для скачивания в интернете на сайте разработчика.

Программная документация также доступна на сайте разработчика в электронном виде.

3.6. Специальные требования

Программное обеспечение должно иметь дружественный интерфейс, рассчитанный на пользователя средней квалификации в области компьютерной грамотности.

Проект будет выполняться поэтапно, с совместимостью модулей, разработанных в разное время.

4. Требования к программной документации

В ходе разработки программы должна быть подготовлена следующая документация:

* текст программы;
* описание программы;
* программа и методика испытаний;
* руководство пользователя.

5. Технико-экономическое обоснование

Использование автоматизированной системы «Пиццерия Итальянский дворик» позволит сократить время на выполнение операций по регистрации поступления товаров для ремонта, их ремонту и выдаче клиентам. Ожидается значительное уменьшение времени на подготовку отчетов, проверку данных и уменьшение ошибок, связанных с ручным вводом информации. Экономический эффект будет достигнут за счет повышения скорости и точности обработки данных, сокращения ручного труда, а также улучшения контроля за складскими операциями и ремонтными процессами.

1. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ
   1. Функциональное моделирование

Функциональное моделирование диаграммы — это графическое представление функций предприятия в определённой области. На рисунке 1 представлена диаграмма IDEF0.

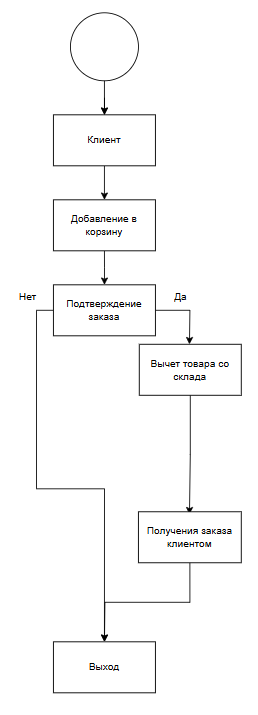


Рисунок 1 – Диаграмма IDEF0

На рисунке 2 представлена диаграмма DFD.



Рисунок 2 - Диаграмма DFD

На рисунке 3 представлена диаграмма деятельности.

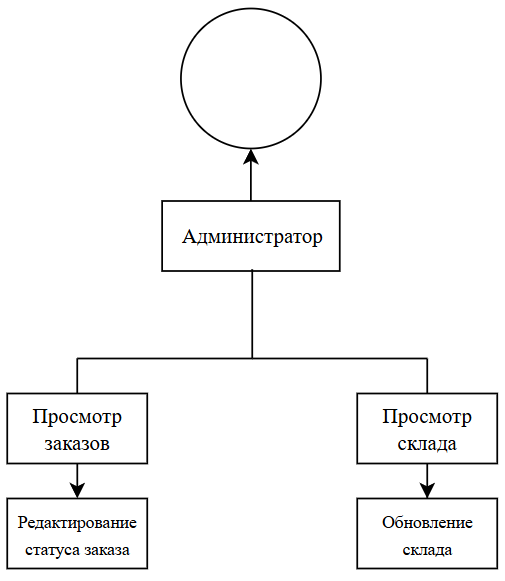


Рисунок 3 - диаграмма деятельности

На рисунке 4 представлена диаграмма последовательности.

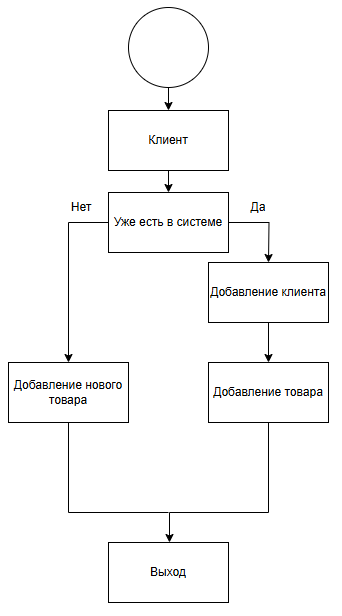


Рисунок 4 - Диаграмма последовательности

* 1. Проектирование интерфейсов приложения

На окне MainWindow (Рисунок 5) находятся следующие элементы:

* ClientButton – открывает страницу клиента;
* Admin Button – открывает страницу администрации;

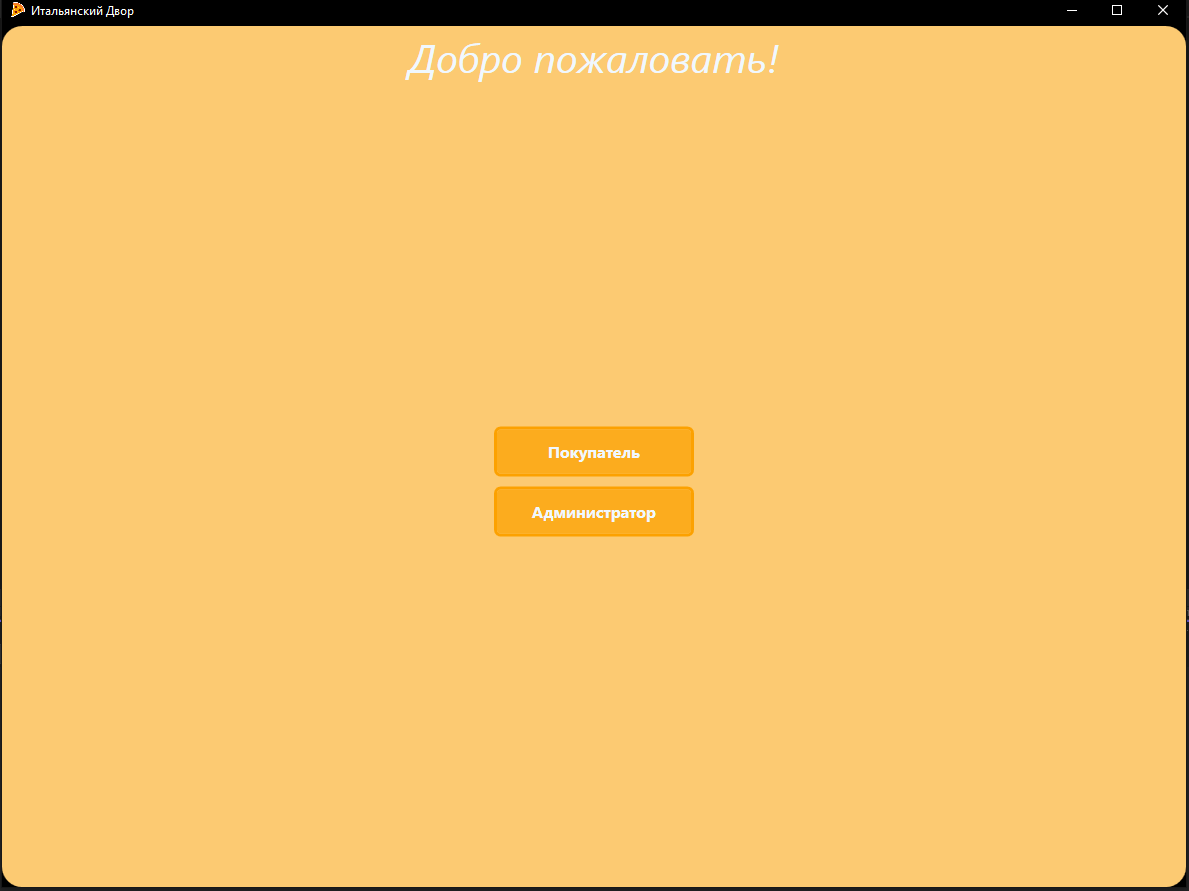


Рисунок 5 – Окно MainWindow

На странице заказы (Рисунок 6) находятся следующие элементы:

* OrdersDataGrid – поле для вывода заказов
* DoneButton – кнопка для завершения заказа

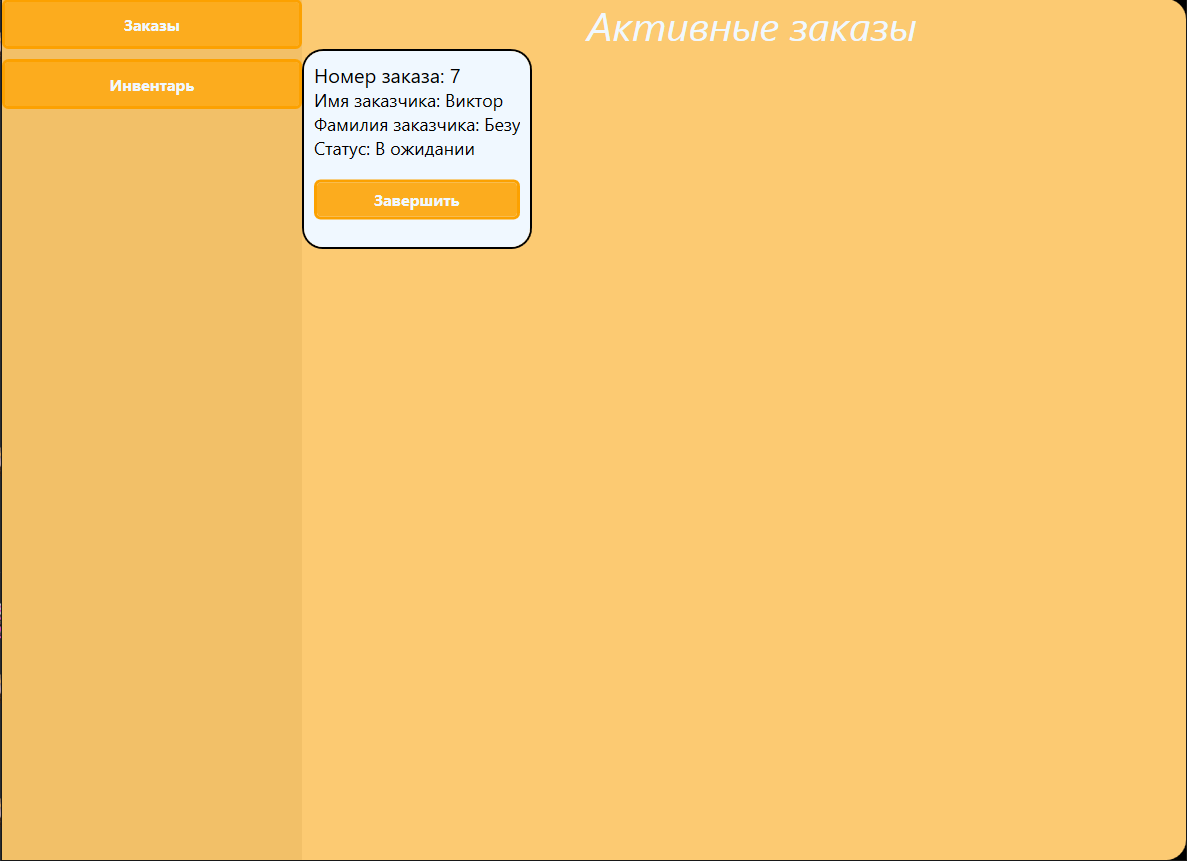


Рисунок 6 – OrdersPage

На окне InventoryPage (Рисунок 7) находятся следующие элементы:

* RefilButton – Кнопка пополнения товара

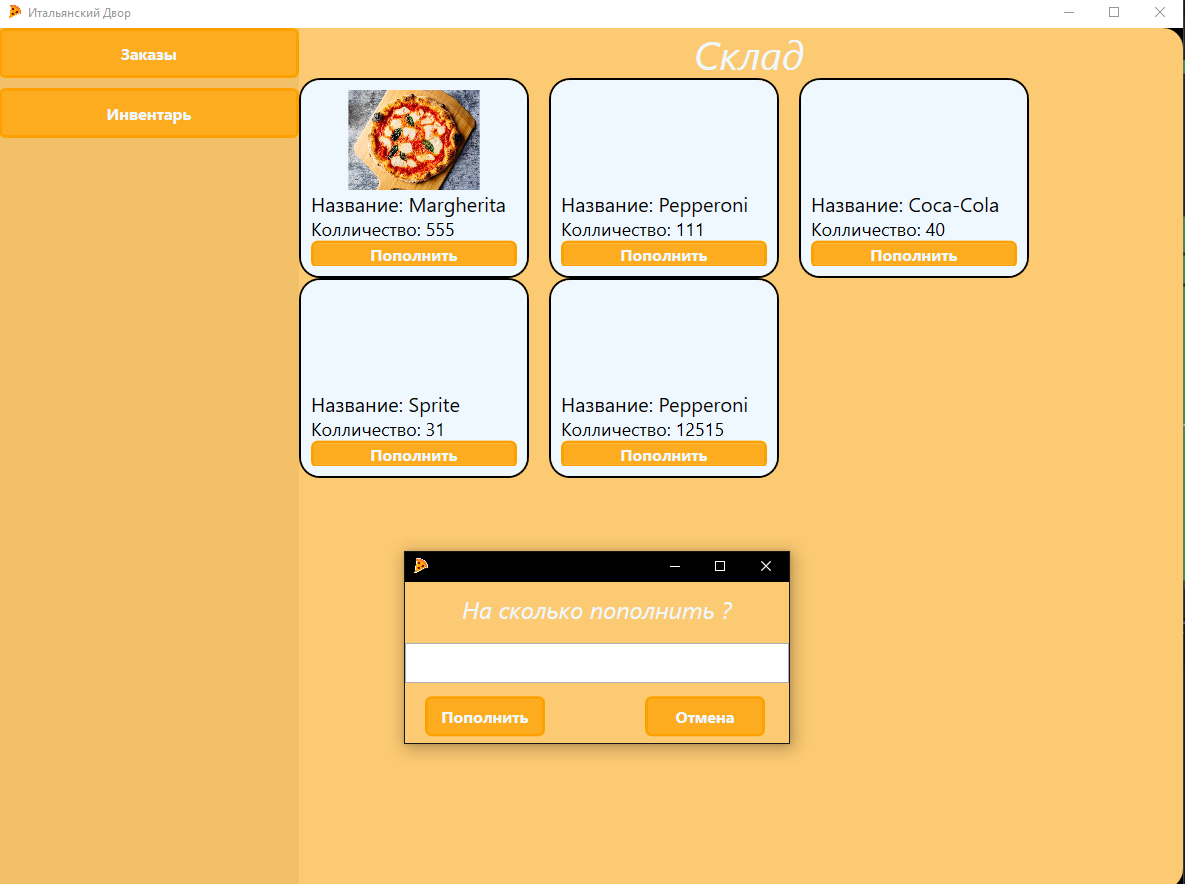


Рисунок 7 – InventoryPage

На окне MenuPage (Рисунок 8) находятся следующие элементы:

* AddToCart – кнопка добавления элеменита в заказ
* MenuGrid – Вывод доступных для заказа продуктов

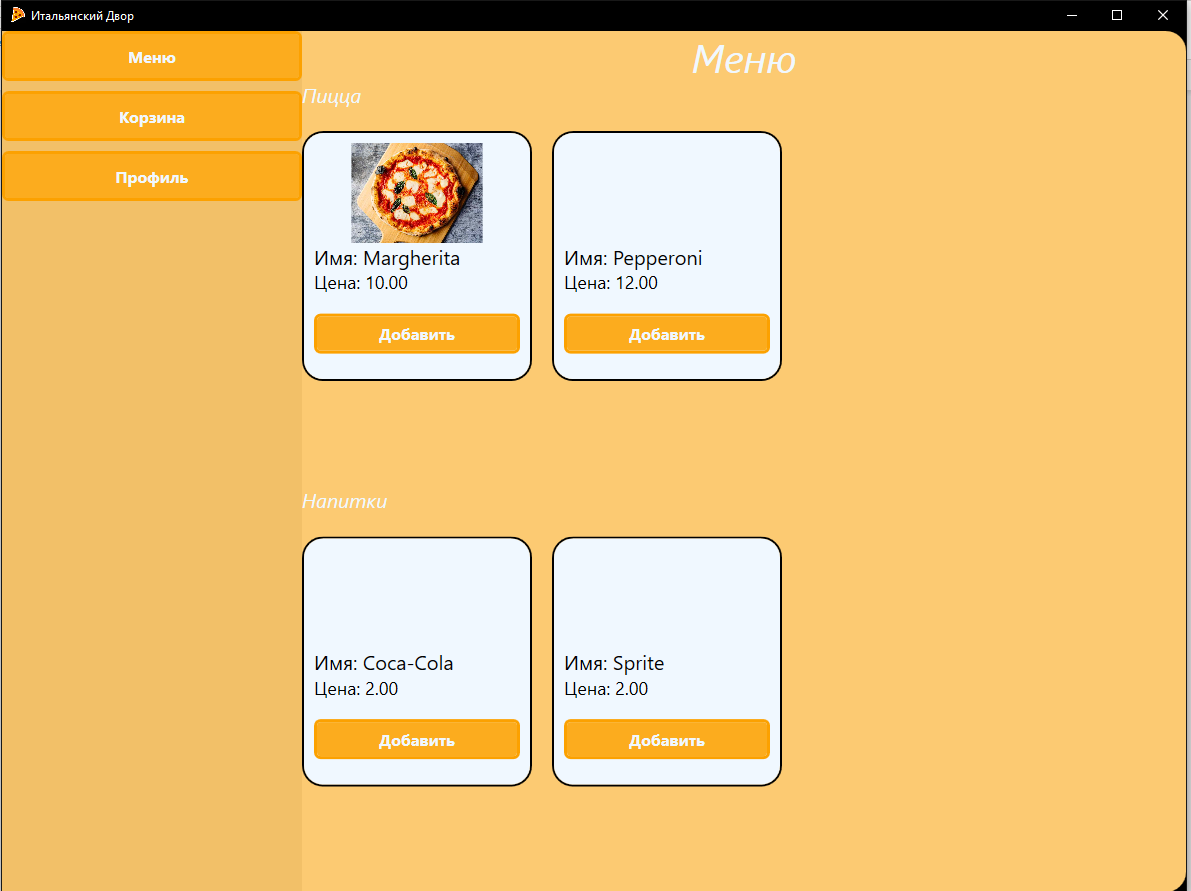


Рисунок 8 – MenuPage

На окне CartPage (Рисунок 9) находятся следующие элементы:

* DeleteFromCart – кнопка убирает продукт из корзины
* AddOrder – кнопка оформляет заказ

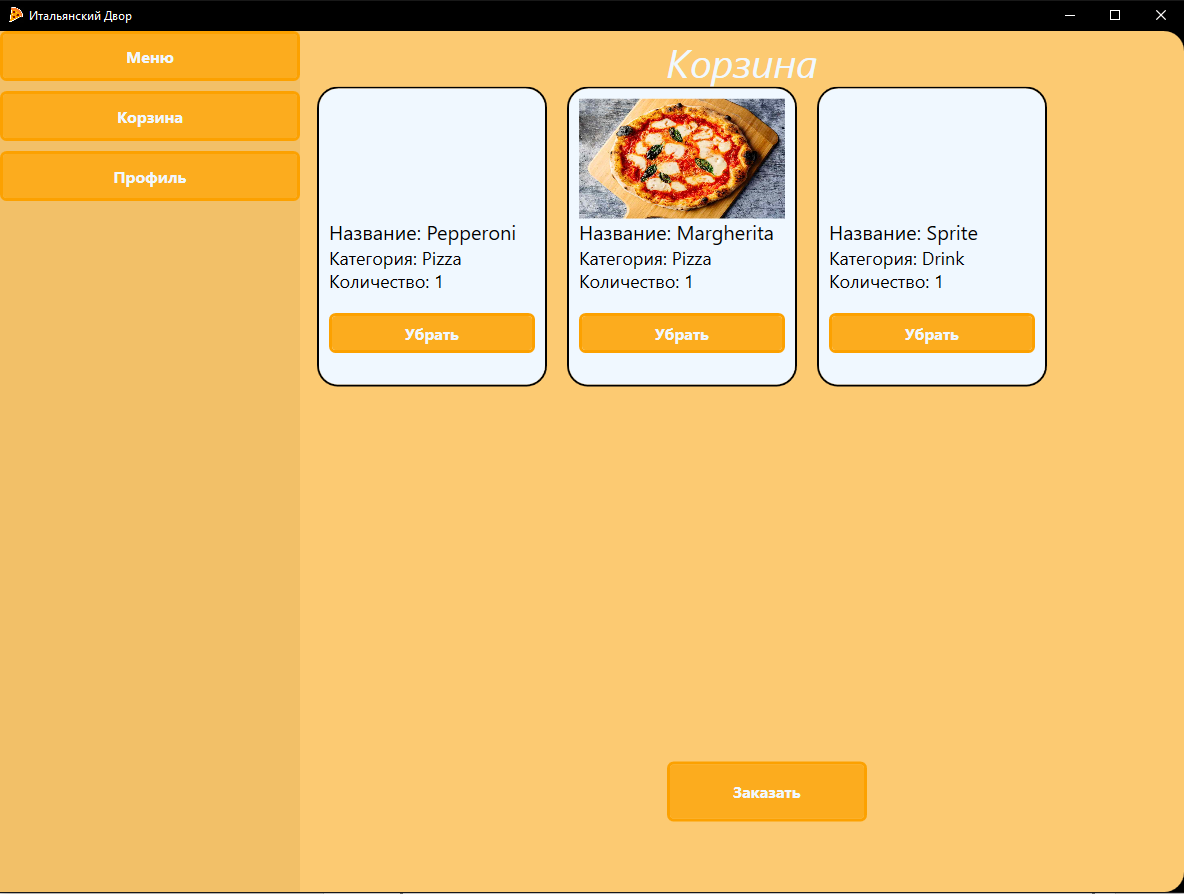


Рисунок 9 - CartPage

На окне ProfilePage (Рисунок 10) находятся следующие элементы:

* PhotoImage – фотография пользователя
* LastOrder – список заказов
* Info – информация о пользователе

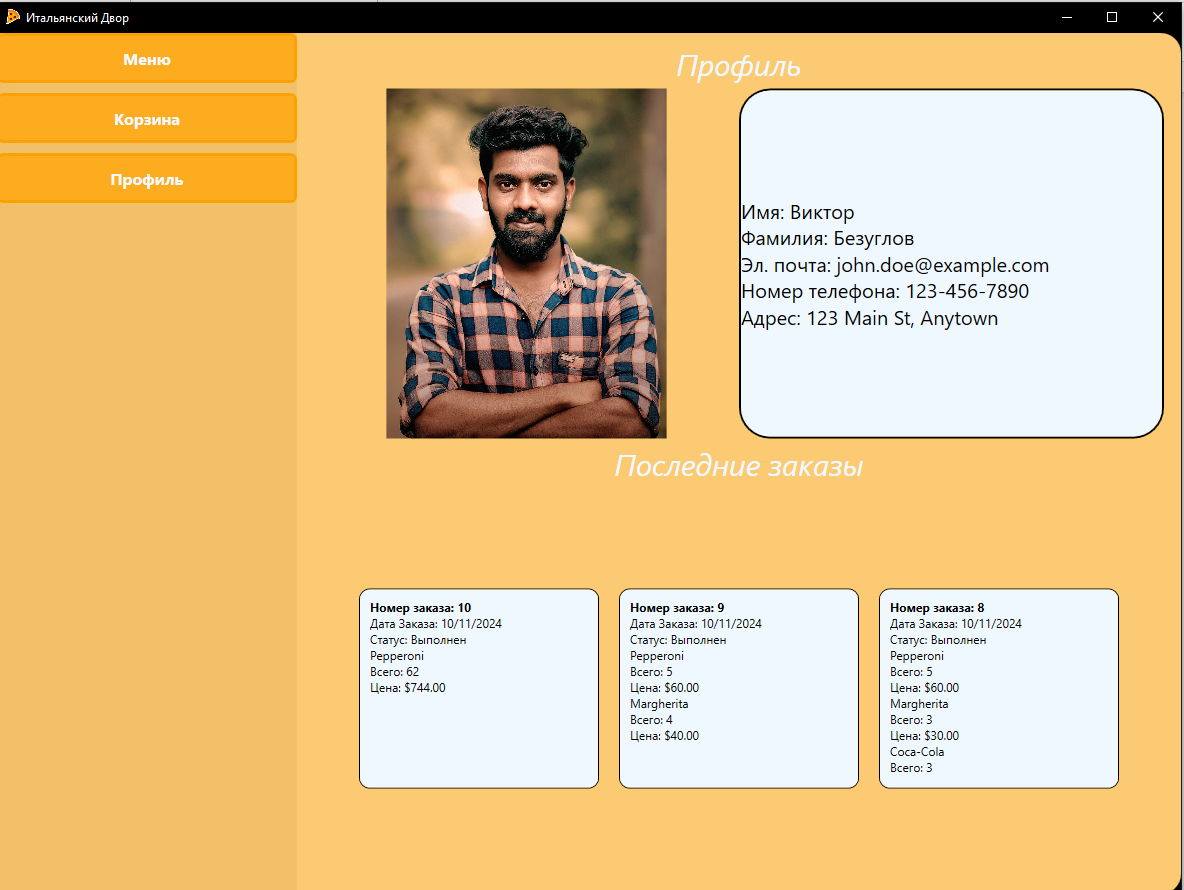


Рисунок 10 – ProfilePage

1. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ
   1. Модель данных

Модель данных реализована в СУБД Microsoft SQL Management studio.

В результате разработки базы данных были спроектированы следующие таблицы:

1. Customers (CustomerID, FirstName, LastName, PhoneNumber, Email, Adress);
2. Orders (OrderID, CustomerID, OrderDate, Status);
3. Cart (CartID, ProductID, CustomerID, Quantity).
4. Admins (AdminId, FirstName, LastName, Email, PhoneNumber, Address)
5. ProductCategories (CategoryID, Name)
6. Products (ProductID, MenuID, Name, Price, CategoryId, PhotoPath)
7. Inventory (InventoryId, ProductID, Quantity)
8. Menu (MenuID, Name, Description)
9. OrderDetailts (OrderDetailsId, OrderId, ProductId, Quantity, Price)

На Рисунке 11 представлена база данных, на Рисунках 12 - 20 показаны таблицы.

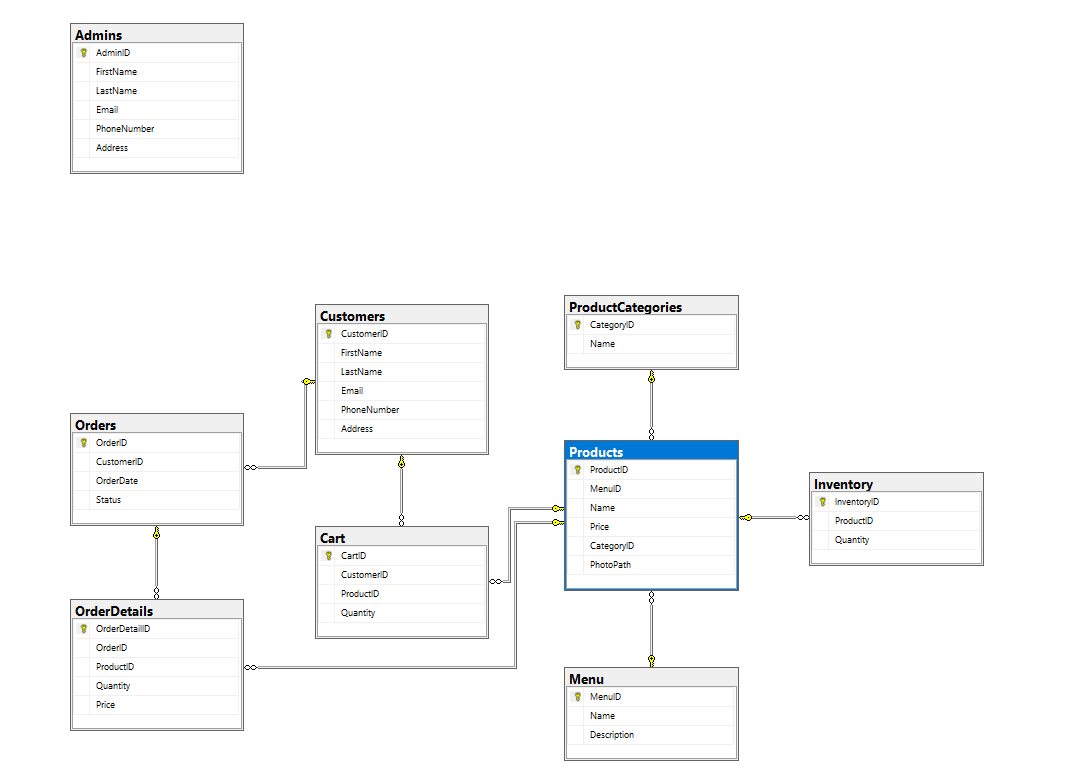


Рисунок 11 – База данных ItalyPizza

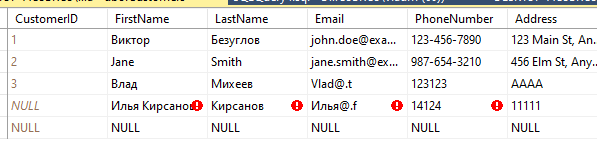


Рисунок 12 – Таблица Customers

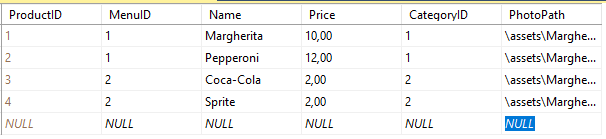


Рисунок 13 - Таблица Products

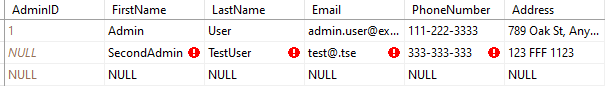


Рисунок 14 – Таблица Admins

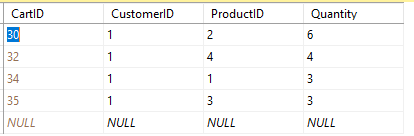


Рисунок 15 – Таблица Cart

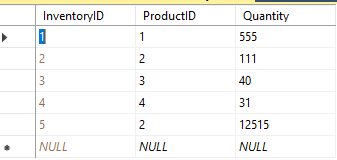


Рисунок 16 – Таблица Inventory

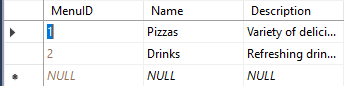


Рисунок 17 – Таблица Menu

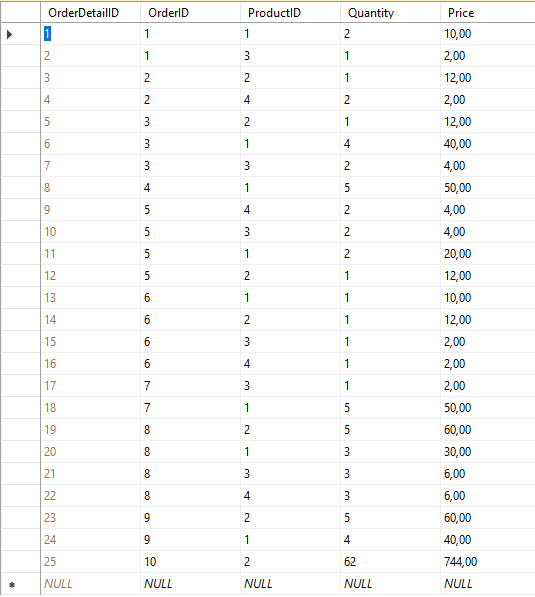


Рисунок 18 – Таблица OrderDetails

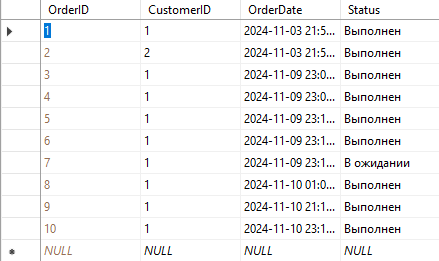


Рисунок 19 – Таблица Orders

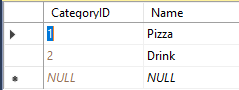


Рисунок 20 – Таблица ProductCategories

* 1. Структура проекта

В процессе разработки программного решения были спроектированы следующие вещи:

1. Окна:

* InventoryPage;
* AdminOrdersPage;
* PizzaAdminWindow;
* ProductsPage;
* PizzaMenuWindow;
* CartPage;
* HaveCart;
* NoCart;
* HaveOrdrs;
* NoOrders;
* DrinksPage;
* Menues;
* MenuPage;
* PizzaClientWindow;
* PizzasPage;
* ProfilePage;
* StartPage;
* MainWindow;
* RefilIventory;
* DeleteFromCart;
* CompleteOrder;

1. Элементы базы данных

* Customers;
* Admins;
* Cart;
* Customers;
* Inventory;
* Menu;
* OrderDetails;
* Orders;
* ProductCategories;
* Products;

На Рисунке 21 представлена вся структура программного решения.

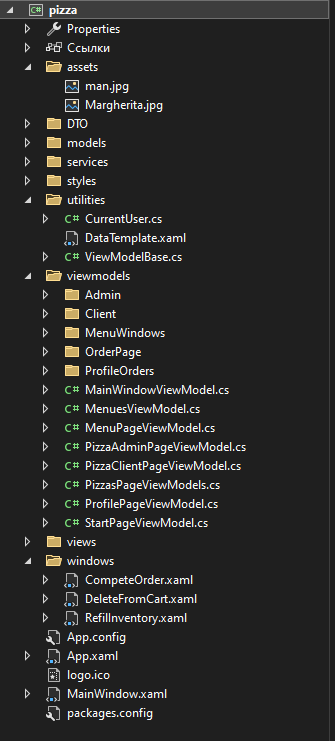


Рисунок 21 – Структура проекта

Класс ProfilePageViewModel.cs имеет следующие методы (Рисунок 22):

* private readonly \_ordersSerivce – доступ к сервису заказов;
* Address – свойство хранщие в себе адрес;
* CurrentOrdersPage – текуща страница заказов;
* CustomerID – ID пользователя
* Email – почта пользователя
* FirstName – имя пользователя
* LastName – фамилия пользователя
* PhoneNumber – номер телефона пользователя
* ProfilePageViewModel() – конструктор класса

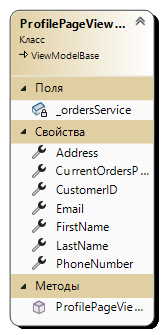


Рисунок 22 – Структура класса ProfilePageViewModel.cs

Класс PizzaClientPageViewModel.cs имеет следующие методы (Рисунок 23):

* CurrentPage – текущая страница;
* ToMenuCommand – команда направляющая на страницу Меню;
* ToOrderCommand – команда направляющая на страницу корзины;
* ToProfilePage – команда направляющая на страницу профиля;
* PizzaClientPageViewModel() – конструктор класса

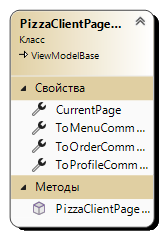


Рисунок 23 - Структура класса PizzaClientPageViewModel

Класс PizzaAdminPageViewModel имеет следующие методы (Рисунок 24):

* CurrentPage – текущая страница;
* ToInventory – команда направлющая на склад;
* ToOrders – команда направляющая на страницу заказов;
* PizzaAdminPageViewModel() – конструктор класса;

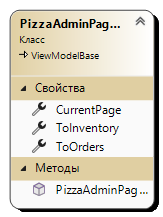


Рисунок 24 - Структура класса IssuePage

4 ОЦЕНКА КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Сценарий тестирования представлен в таблице 1.

Таблица 1. Сценарий тестирования

|  |  |  |
| --- | --- | --- |
| Test case # | Название теста | Резюме испытания  (краткое описание) |
| TC\_UI\_1 | Проверка пополнения товара на отрицательное значение | 1. Проверка пополнения товара на отрицательное значение с отображением предупреждения.   2.Проверка невозможности сохранения отрицательного значения. |
| TC\_UI\_2 | Проверка изменения статуса заказа | 1. Проверка изменения статуса заказа с правильным вводом. 2. Проверка невозможности изменения статуса заказа с некорректным вводом. |
| TC\_UI\_3 | Проверка добавления продукта в корзину с правильным вводом | 1. Проверка добавления продукта в корзину с правильным вводом.  2. Проверка отображения продукта в корзине. |

Тест – кейсы представлены в таблицах 2 – 4.

Таблица 2. Тест-кейс #1

|  |  |
| --- | --- |
| Элементы тест-кейса | Значения |
| Название проекта | Информационная система «Пиццерия Итальянский двор» |
| Номер версии | V 1.0.0.1 |
| Имя тестера | Безуглов Виктор Алексеевич |
| Даты тестирования | 10.10 -11.11 |
| Test Case # | TC\_UI\_1 |
| Приоритет теста | Высокий |
| Название теста | Проверка пополнения товара на отрицательное значение |
| Резюме испытания | 1. Проверка пополнения товара на отрицательное значение с отображением предупреждения. 2. Проверка невозможности сохранения отрицательного значения. |
| Шаги тестирования | 1. Ввод отрицательного значения для пополнения товара.  2. Попытка сохранения отрицательного значения. |
| Данные тестирования | 1. Количество: -5.  2. Количество: -5. |
| Ожидаемый результат | 1. Появляется предупреждение о некорректном вводе.  2. Отрицательное значение не сохраняется в базе данных. |
| Фактический результат | 1. Появилось предупреждение о некорректном вводе.  2. Отрицательное значение не сохранилось в базе данных. |
| Предпосылки | Запуск приложения и появление формы пополнения товара. |
| Постусловия | Подключение к базе данных ItalyPizza |
| Статус | - |
| Комментарии | - |

Таблица 3. Тест-кейс #2

|  |  |
| --- | --- |
| Элементы тест-кейса | Значения |
| Название проекта | Информационная система «Ремонт электронной техники» |
| Номер версии | V 1.0.0.1 |
| Имя тестера | Безуглов Виктор Алексеевич |
| Даты тестирования | 10.10 -11.11 |
| Test Case # | TC\_UI\_2 |
| Приоритет теста | Высокий |
| Название теста | Проверка изменения статуса заказа |
| Резюме испытания | 1. Проверка изменения статуса заказа с правильным вводом; 2. Проверка невозможности изменения статуса заказа с некорректным вводом; |
| Шаги тестирования | 1. Ввод корректного статуса заказа с правильным вводом; 2. Ввод некорректного статуса и попытка изменения; |
| Данные тестирования | 1. Статус: В процессе; 2. Статус: Несуществующий; |
| Ожидаемый результат | 1. Статус заказа изменяется в базе данных; 2. Изменения не сохранятся и появится окно ошибки; |
| Фактический результат | 1. Изменения сохранились в базе данных; 2. Изменения не сохранились и появилось окно ошибки; |
| Предпосылки | Запуск приложения и появление формы управления заказами; |
| Постусловия | Подключение к базе данных ItalyPizza |
| Статус | - |
| Комментарии | - |

Таблица 4. Тест-кейс #3

|  |  |
| --- | --- |
| Элементы тест-кейса | Значения |
| Название проекта | Информационная система «Пиццерия Итальянский двор» |
| Номер версии | V 1.0.0.1 |
| Имя тестера | Безуглов Виктор Алексеевич |
| Даты тестирования | 10.10 -11.11 |
| Test Case # | TC\_UI\_3 |
| Приоритет теста | Высокий |
| Название теста | Проверка добавления продукта в корзину с правильным вводом |
| Резюме испытания | 1. Проверка добавления продукта в корзину с правильным вводом. 2. Проверка отображения продукта в корзине. |
| Шаги тестирования | 1. Выбор продукта и добавление его в корзину. 2. Проверка отображения продукта в корзине. |
| Данные тестирования | 1. Продукт: Пицца Маргарита, Количество: 1. 2. Продукт: Пицца Маргарита, Количество: 1. |
| Ожидаемый результат | 1. Продукт добавляется в корзину. 2. Продукт отображается в корзине |
| Фактический результат | 1. Продукт добавился в корзину. 2. Продукт отобразился в корзине. |
| Предпосылки | Подключение к базе данных ItalyPizza |
| Постусловия | - |
| Статус | - |
| Комментарии | - |

ВЫВОДЫ И ЗАКЛЮЧЕНИЕ

В процессе создания курсового проекта, были достигнуты все выше поставленные цели:

* Создание программного обеспечения, которое позволяет эффективно управлять процессом ремонта электронной техники.
* Оптимизация рабочих процессов для сокращения времени ремонта и уменьшения количества ошибок.
* Реализация механизмов обратной связи для клиентов.

Для этого были выполнены следующие задачи:

* сформулирована цель разработки информационной системы;
* описана предметная область, для которой разрабатывается информационная система
* проведён анализ предметной области, выявлены основные бизнес-процессы и пользователи, которые будут с ней работать.
* определён круг запросов и задач, которые предполагается решать с использованием созданной информационной системы;
* написано техническое задание на разработку информационной системы;
* проведено функциональное моделирование и объектно-ориентированное проектирование информационной системы;
* проведено проектирование интерфейса информационной системы:
* разработана модель данных и реализована в выбранной СУБД
* разработано программное решение для информационной системы;
* оценено качество разработанной системы путем тестирования основного функционала.

СПИСОК ЛИТЕРАТУРЫ

1) Абрамян А. В. Разработка пользовательского интерфейса на основе технологии Windows Presentation Foundation : учебник для студ. Учреждений сред. Проф. Образования. - М: Издательство Южного федерального университета, 2019.

2) Ганенко А. П. Оформление текстовых и графических материалов при подготовке дипломных проектов, курсовых и письменных экзаменационных работ (требования ЕСКД) : учебно-метод. Пособие для студ. Учреждения сред. Проф. Образования – М: Издательский центр «Академия», 2020.

Электронные ресурсы:

3) Metanit – язык программирования С# и платформа .NET [Электронный ресурс] Режим доступа: свободный <https://metanit.com/sharp/> (Дата обращения 03.11.2024)

4) Национальный открытый университет [Электронный ресурс] Режим доступа: свободный https://www.intuit.ru/ (Дата обращения 03.11.2024)

5) Professor Web [Электронный ресурс] Режим доступа: свободный https://professorweb.ru/ (Дата обращения 03.11.2024)

Приложение A – Код программных модулей

CartItems.cs:

using pizza.models;

using System.Runtime.InteropServices;

namespace pizza.DTO

{

public class CartItems

{

public Products Product { get; set; }

public int Quantity { get; set; }

public CartItems(Products product, int quantity)

{

Product = product;

Quantity = quantity;

}

}

}

CartService.cs:

using pizza.models;

using pizza.utilities;

using System.IO.Compression;

using System.Linq;

namespace pizza.services

{

public class CartService : ViewModelBase

{

private readonly ItalyPizzaEntities \_context;

public CartService()

{

\_context = new ItalyPizzaEntities();

}

public Cart[] GetCurrentCustomerCarts()

{

Cart[] currentCart = \_context.Cart

.Where(cart => cart.CustomerID == CurrentUser.CustomerID)

.ToArray();

return currentCart;

}

public void ClearCarts()

{

Cart[] carts = \_context.Cart

.Where(cart => cart.CustomerID == CurrentUser.CustomerID)

.ToArray();

\_context.Cart.RemoveRange(carts);

\_context.SaveChanges();

}

public void DeleteProductFromCart(Products product)

{

Cart[] carts = \_context.Cart

.Where(cart => cart.ProductID == product.ProductID)

.ToArray();

\_context.Cart.RemoveRange(carts);

\_context.SaveChanges();

}

public Cart GetCartByProductIds(int productIds)

{

return \_context.Cart

.Where(cart =>cart.ProductID == productIds)

.FirstOrDefault();

}

public void AddToCart(Products product)

{

Cart existCart = GetCartByProductIds(product.ProductID);

if (existCart != null)

{

existCart.Quantity += 1;

\_context.SaveChanges();

return;

}

Cart cart = new Cart()

{

CustomerID = CurrentUser.CustomerID,

ProductID = product.ProductID,

Quantity = 1

};

\_context.Cart.Add(cart);

\_context.SaveChanges();

}

}

}

InventoryService.cs:

using pizza.models;

using System.Linq;

namespace pizza.services

{

public class InventoryService

{

private readonly ItalyPizzaEntities \_context;

public InventoryService()

{

\_context = new ItalyPizzaEntities();

}

public Inventory[] GetCurrentInventory()

{

return \_context.Inventory.ToArray();

}

}

}

MenuService.cs:

using pizza.models;

using System.Linq;

namespace pizza.services

{

public class MenuService

{

private readonly ItalyPizzaEntities \_context;

public MenuService()

{

\_context = new ItalyPizzaEntities();

}

public Menu[] GetMenuItems()

{

return \_context.Menu.ToArray();

}

}

}

OrdersService.cs:

using pizza.models;

using pizza.utilities;

using System;

using System.Linq;

using System.Security.Claims;

using System.Windows;

namespace pizza.services

{

public class OrdersService

{

private readonly ItalyPizzaEntities \_context;

private readonly CartService \_cartService;

public OrdersService()

{

\_context = new ItalyPizzaEntities();

\_cartService = new CartService();

}

public Orders[] GetOrdersForProfile()

{

return \_context.Orders

.Where(ord => ord.CustomerID == CurrentUser.CustomerID)

.OrderByDescending(ord => ord.OrderID)

.Take(3)

.ToArray();

}

public void CreateOrder(Cart[] carts)

{

Orders order = new Orders()

{

CustomerID = CurrentUser.CustomerID,

Status = "В ожидании",

OrderDate = DateTime.Now,

};

\_context.Orders.Add(order);

\_context.SaveChanges();

Orders currentOrder = GetCurrentOrder();

foreach (Cart cart in carts)

{

OrderDetails details = new OrderDetails()

{

OrderID = currentOrder.OrderID,

ProductID = cart.ProductID,

Quantity = cart.Quantity,

Price = cart.Products.Price \* cart.Quantity

};

\_context.OrderDetails.Add(details);

\_context.SaveChanges();

}

\_cartService.ClearCarts();

MessageBox.Show("Успешно");

}

public Orders GetCurrentOrder()

{

return \_context.Orders

.Where(ord => ord.CustomerID == CurrentUser.CustomerID)

.OrderByDescending(ord => ord.OrderID)

.FirstOrDefault();

}

public bool IsCustomerHaveOrders()

{

return \_context.Orders.Any();

}

public Orders[] GetActiveOrders()

{

return \_context.Orders

.Where(ord => ord.Status != "Выполнен")

.ToArray();

}

public void CompleteOrder(Orders order)

{

Orders currentOrder = \_context.Orders.Where(ord => ord.OrderID == order.OrderID).FirstOrDefault();

currentOrder.Status = "Выполнен";

foreach(var orderDetail in currentOrder.OrderDetails)

{

var currentInventory = \_context.Inventory

.Where(inv => inv.ProductID == orderDetail.ProductID)

.FirstOrDefault();

currentInventory.Quantity -= orderDetail.Quantity;

if (currentInventory.Quantity <= 0)

{

currentInventory.Quantity = 0;

}

}

\_context.SaveChanges();

}

}

}

ProductService.cs:

using pizza.DTO;

using pizza.models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Management.Instrumentation;

namespace pizza.services

{

public class ProductService

{

private readonly ItalyPizzaEntities \_context;

public ProductService()

{

\_context = new ItalyPizzaEntities();

}

public Products[] GetPizzas()

{

return \_context.Products

.Where(prod => prod.MenuID == 1)

.ToArray(); ;

}

public Products[] GetDrinks()

{

return \_context.Products.Where(prod => prod.MenuID == 2).ToArray();

}

public CartItems[] GetProductsFromCart(Cart[] carts)

{

List<CartItems> cartItems = new List<CartItems>();

foreach (Cart cart in carts)

{

Products product = \_context.Products.FirstOrDefault(p => p.ProductID == cart.ProductID);

cartItems.Add(new CartItems(product, cart.Quantity ?? 0));

}

return cartItems.ToArray();

}

}

}

BeautifulButtonStyle.xaml:

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

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

<Style x:Key="BeautifulButtonStyle" TargetType="Button">

<Setter Property="Background" Value="#fcac1e"/>

<Setter Property="Foreground" Value="AliceBlue"/>

<Setter Property="FontSize" Value="16"/>

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

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

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

<Setter Property="BorderThickness" Value="3"/>

<Setter Property="BorderBrush" Value="#fca100"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}"

BorderBrush="{TemplateBinding BorderBrush}"

BorderThickness="{TemplateBinding BorderThickness}"

CornerRadius="5">

<ContentPresenter HorizontalAlignment="Center"

VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

<Style.Triggers>

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

<Setter Property="Background" Value="#ffa80d"/>

<Setter Property="Cursor" Value="Hand"/>

<Setter Property="BorderBrush" Value="#cf8400"/>

</Trigger>

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

<Setter Property="Background" Value="#fae4be"/>

<Setter Property="Cursor" Value="Wait"/>

<Setter Property="BorderBrush" Value="#d1c0a3"/>

</Trigger>

</Style.Triggers>

</Style>

</ResourceDictionary>

CurrentUser.cs:

namespace pizza.utilities

{

public static class CurrentUser

{

public static int CustomerID { get; } = 1;

public static string FirstName { get; } = "Виктор";

public static string LastName { get; } = "Безуглов";

public static string Email { get; } = "john.doe@example.com";

public static string PhoneNumber { get; } = "123-456-7890";

public static string Address { get; } = "123 Main St, Anytown";

}

}

ViewModelBase.cs:

using ReactiveUI;

using System.Windows;

namespace pizza.utilities

{

public class ViewModelBase : ReactiveObject

{

public double Width => SystemParameters.PrimaryScreenWidth \* 0.8;

public double Height => SystemParameters.PrimaryScreenHeight \* 0.8;

}

}

InventoryPageViewModel.cs:  
using pizza.models;

using pizza.services;

using pizza.utilities;

using ReactiveUI.Fody.Helpers;

namespace pizza.viewmodels.Admin.InventoryPage

{

public class InventoryPageViewModel : ViewModelBase

{

private readonly InventoryService \_inventoryService;

[Reactive] public Inventory[] Inventories { get; set; }

public InventoryPageViewModel()

{

\_inventoryService = new InventoryService();

Inventories = \_inventoryService.GetCurrentInventory();

}

}

}

OrdersAdminViewModels.cs:

using pizza.models;

using pizza.services;

using ReactiveUI.Fody.Helpers;

namespace pizza.viewmodels.Admin

{

public class OrdersAdminViewModels

{

private readonly OrdersService \_ordersService;

[Reactive] public Orders[] Orders { get; set; }

public OrdersAdminViewModels()

{

\_ordersService = new OrdersService();

Orders = \_ordersService.GetActiveOrders();

}

}

}

ProductsPageViewModel.cs:

using pizza.models;

using pizza.services;

using pizza.utilities;

using ReactiveUI.Fody.Helpers;

namespace pizza.viewmodels.Client

{

public class ProductsPageViewModel : ViewModelBase

{

private readonly ProductService \_productService;

[Reactive] public Products[] Pizzas { get; set; }

[Reactive] public Products[] Drinks { get; set; }

public ProductsPageViewModel()

{

\_productService = new ProductService();

Pizzas = \_productService.GetPizzas();

Drinks = \_productService.GetDrinks();

}

}

}

PizzaMenuWinViewModel.cs:

using pizza.models;

using pizza.services;

using pizza.utilities;

using ReactiveUI.Fody.Helpers;

namespace pizza.viewmodels.MenuWindows

{

public class PizzaMenuWinViewModel : ViewModelBase

{

private readonly ProductService \_productService;

[Reactive] public Products[] Pizzas { get; set; }

public PizzaMenuWinViewModel()

{

\_productService = new ProductService();

Pizzas = \_productService.GetPizzas();

}

}

}

CartPageViewModel.cs:

using pizza.models;

using pizza.services;

using pizza.utilities;

using pizza.views.OrderPage;

using ReactiveUI.Fody.Helpers;

using System.Linq;

using System.Windows.Controls;

namespace pizza.viewmodels.OrderPage

{

public class CartPageViewModel : ViewModelBase

{

private readonly CartService \_cartService;

[Reactive] public UserControl CurrentView { get; set; }

public CartPageViewModel()

{

\_cartService = new CartService();

Cart[] carts = \_cartService.GetCurrentCustomerCarts();

if (carts.Any()) CurrentView = new HaveCart();

else CurrentView = new NoCart();

}

}

}

using pizza.DTO;

using pizza.models;

using pizza.services;

using pizza.utilities;

using ReactiveUI;

using ReactiveUI.Fody.Helpers;

using System;

using System.Linq;

using System.Reactive;

using System.Windows;

namespace pizza.viewmodels.OrderPage

{

public class HaveCartViewModel : ViewModelBase

{

private readonly ProductService \_productService;

private readonly CartService \_cartService;

private readonly OrdersService \_ordersService;

[Reactive] public CartItems[] CartItems { get; set; }

[Reactive] public Visibility isVisible { get; set; }

public ReactiveCommand<Unit, Unit> DeleteCommand { get; set; }

public ReactiveCommand<Unit, Unit> CreateOrder { get; set; }

public HaveCartViewModel()

{

\_cartService = new CartService();

\_productService = new ProductService();

\_ordersService = new OrdersService();

Cart[] carts = \_cartService.GetCurrentCustomerCarts();

CartItems = \_productService.GetProductsFromCart(carts);

isVisible = Visibility.Visible;

DeleteCommand = ReactiveCommand.Create(() =>

{

bool doWhile = true;

while (doWhile)

{

Cart[] newCarts = \_cartService.GetCurrentCustomerCarts();

if (newCarts.Length < CartItems.Length)

{

CartItems = \_productService.GetProductsFromCart(carts);

doWhile = false;

}

}

});

CreateOrder = ReactiveCommand.Create(() =>

{

\_ordersService.CreateOrder(carts);

var items = CartItems.ToList();

items.Clear();

CartItems = items.ToArray();

isVisible = Visibility.Collapsed;

});

}

}

}

HaveOrdersViewModel.cs:

using pizza.models;

using pizza.services;

using pizza.utilities;

using ReactiveUI;

using ReactiveUI.Fody.Helpers;

namespace pizza.viewmodels.ProfileOrders

{

public class HaveOrdersViewModel : ViewModelBase

{

[Reactive] public Orders[] Orders { get; set; }

private readonly OrdersService \_ordersService;

public HaveOrdersViewModel()

{

\_ordersService = new OrdersService();

Orders = \_ordersService.GetOrdersForProfile();

}

}

}

MenuesViewModel.cs:

using pizza.models;

using pizza.services;

using pizza.utilities;

using pizza.views;

using ReactiveUI;

using ReactiveUI.Fody.Helpers;

using System.Reactive;

using System.Windows;

namespace pizza.viewmodels

{

public class MenuesViewModel : ViewModelBase

{

private readonly MenuService \_menuService;

[Reactive] public Menu[] MenuItems { get; set; }

ReactiveCommand<Unit, Unit> ToPizzasPageCommand { get; set; }

public MenuesViewModel()

{

\_menuService = new MenuService();

MenuItems = \_menuService.GetMenuItems();

ToPizzasPageCommand = ReactiveCommand.Create(() =>

{

});

}

}

}

using pizza.models;

using pizza.services;

using pizza.utilities;

using pizza.views;

using ReactiveUI.Fody.Helpers;

using System.ComponentModel;

using System.Windows;

using System.Windows.Controls;

namespace pizza.viewmodels

{

public class MenuPageViewModel : ViewModelBase

{

[Reactive] public Visibility IsBackButtonVisible { get; set; }

[Reactive] public UserControl CurrentPage { get; set; }

public MenuPageViewModel()

{

CurrentPage = new Menues();

IsBackButtonVisible = Visibility.Hidden;

}

}

}

PizzaAdminPageViewModel.cs:

using pizza.models;

using pizza.utilities;

using pizza.views.Admin;

using pizza.views.Admin.Inventory;

using ReactiveUI;

using ReactiveUI.Fody.Helpers;

using System.Reactive;

using System.Windows.Controls;

namespace pizza.viewmodels

{

public class PizzaAdminPageViewModel : ViewModelBase

{

[Reactive] public UserControl CurrentPage { get; set; }

public ReactiveCommand<Unit, Unit> ToOrders { get; set; }

public ReactiveCommand<Unit, Unit> ToInventory { get; set; }

public PizzaAdminPageViewModel()

{

ToOrders = ReactiveCommand.Create(() =>

{

if (CurrentPage is AdminOrdersPage) return;

CurrentPage = new AdminOrdersPage();

});

ToInventory = ReactiveCommand.Create(() =>

{

if (CurrentPage is InventoryPage) return;

CurrentPage = new InventoryPage();

});

CurrentPage = new AdminOrdersPage();

}

}

}

PizzaClientPageViewModel.cs:

using pizza.utilities;

using pizza.views;

using pizza.views.Client;

using pizza.views.OrderPage;

using ReactiveUI;

using ReactiveUI.Fody.Helpers;

using System.Reactive;

using System.Windows.Controls;

namespace pizza.viewmodels

{

public class PizzaClientPageViewModel : ViewModelBase

{

[Reactive] public UserControl CurrentPage { get; set; }

public ReactiveCommand<Unit, Unit> ToMenuCommand{ get; set; }

public ReactiveCommand<Unit, Unit> ToOrderCommand { get; set; }

public ReactiveCommand<Unit, Unit> ToProfileCommand { get; set; }

public PizzaClientPageViewModel()

{

CurrentPage = new ProductsPage();

ToMenuCommand = ReactiveCommand.Create(() =>

{

if(CurrentPage is ProductsPage) return;

CurrentPage = new ProductsPage();

});

ToProfileCommand = ReactiveCommand.Create(() =>

{

if(CurrentPage is ProfilePage) return;

CurrentPage = new ProfilePage();

});

ToOrderCommand = ReactiveCommand.Create(() =>

{

if(CurrentPage is CartPage) return;

CurrentPage = new CartPage();

});

}

}

}

PizzasPageViewModels.cs:

using pizza.models;

using pizza.services;

using pizza.utilities;

using ReactiveUI.Fody.Helpers;

namespace pizza.viewmodels

{

public class PizzasPageViewModels : ViewModelBase

{

private readonly ProductService \_productService;

[Reactive] public Products[] Pizzas { get; set; }

public PizzasPageViewModels()

{

\_productService = new ProductService();

Pizzas = \_productService.GetPizzas();

}

}

}

using pizza.services;

using pizza.utilities;

using pizza.views.ProfileOrders;

using ReactiveUI.Fody.Helpers;

using System.Security.Permissions;

using System.Windows.Controls;

namespace pizza.viewmodels

{

public class ProfilePageViewModel : ViewModelBase

{

private readonly OrdersService \_ordersService;

public int CustomerID { get; } = 1;

public string FirstName { get; } = "Виктор";

public string LastName { get; } = "Безуглов";

public string Email { get; } = "john.doe@example.com";

public string PhoneNumber { get; } = "123-456-7890";

public string Address { get; } = "123 Main St, Anytown";

[Reactive] public UserControl CurrentOrdersPage { get; set; }

public ProfilePageViewModel()

{

\_ordersService = new OrdersService();

bool isUserHaveOrders = \_ordersService.IsCustomerHaveOrders();

if (isUserHaveOrders) CurrentOrdersPage = new HaveOrders();

else CurrentOrdersPage = new NoOrders();

}

}

}

InventoryPage.xaml:

<UserControl x:Class="pizza.views.Admin.Inventory.InventoryPage"

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

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

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

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

xmlns:local="clr-namespace:pizza.views.Admin.Inventory"

mc:Ignorable="d"

Background="Transparent"

Height="900"

Width="900">

<UserControl.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/styles/BeautifulButtonStyle.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</UserControl.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<TextBlock Text="Склад"

FontSize="40"

FontStyle="Italic"

Foreground="AliceBlue"

VerticalAlignment="Center"

HorizontalAlignment="Center"/>

</Grid>

<Grid Grid.Row="1">

<Grid>

<Grid>

<ScrollViewer VerticalScrollBarVisibility="Auto">

<ItemsControl ItemsSource="{Binding Inventories}">

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<WrapPanel Orientation="Horizontal"

ItemHeight="200"

ItemWidth="250"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

<ItemsControl.ItemTemplate>

<DataTemplate>

<Border Background="AliceBlue"

CornerRadius="20"

BorderThickness="2"

BorderBrush="Black"

Margin="0,0,20,0"

Padding="10">

<Grid>

<Grid Grid.Row="1">

<Grid.RowDefinitions>

<RowDefinition Height="100"/>

<RowDefinition Height="100"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Image Source="{Binding Products.PhotoPath}"/>

</Grid>

<Grid Grid.Row="1">

<StackPanel Orientation="Vertical">

<TextBlock Text="{Binding Products.Name, StringFormat='Название: {0}'}"

FontSize="20"/>

<TextBlock Text="{Binding Quantity, StringFormat='Колличество: {0}'}"

FontSize="18"/>

<Button Click="Button\_Click" Style="{StaticResource BeautifulButtonStyle}"

Content="Пополнить"/>

</StackPanel>

</Grid>

</Grid>

</Grid>

</Border>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

</ScrollViewer>

</Grid>

</Grid>

</Grid>

</Grid>

</UserControl>

AdminOrdersPage.cs:

<UserControl x:Class="pizza.views.Admin.AdminOrdersPage"

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

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

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

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

xmlns:local="clr-namespace:pizza.views.Admin"

mc:Ignorable="d"

Background="Transparent"

Height="900"

Width="900">

<UserControl.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/styles/BeautifulButtonStyle.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</UserControl.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<TextBlock Text="Активные заказы"

FontSize="40"

FontStyle="Italic"

Foreground="AliceBlue"

VerticalAlignment="Center"

HorizontalAlignment="Center"/>

</Grid>

<Grid Grid.Row="1">

<Grid>

<ScrollViewer VerticalScrollBarVisibility="Auto">

<ItemsControl ItemsSource="{Binding Orders}">

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<WrapPanel Orientation="Horizontal"

ItemHeight="200"

ItemWidth="250"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

<ItemsControl.ItemTemplate>

<DataTemplate>

<Border Background="AliceBlue"

CornerRadius="20"

BorderThickness="2"

BorderBrush="Black"

Margin="0,0,20,0"

Padding="10">

<Grid>

<Grid Grid.Row="1">

<StackPanel Orientation="Vertical">

<TextBlock Text="{Binding OrderID, StringFormat='Номер заказа: {0}'}"

FontSize="20"/>

<TextBlock Text="{Binding Customers.FirstName, StringFormat='Имя заказчика: {0}'}"

FontSize="18"/>

<TextBlock Text="{Binding Customers.LastName, StringFormat='Фамилия заказчика: {0}'}"

FontSize="18"/>

<TextBlock Text="{Binding Status, StringFormat='Статус: {0}'}"

FontSize="18"/>

<Button Content="Завершить"

Margin="0,20,0,0"

Click="Button\_Click"

Command="{Binding CompleteOrder}"

Style="{StaticResource BeautifulButtonStyle}"

Height="40"/>

</StackPanel>

</Grid>

</Grid>

</Border>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

</ScrollViewer>

</Grid>

</Grid>

</Grid>

</UserControl>

PizzaAdminWindow.xaml:

<Window x:Class="pizza.views.PizzaAdminWindow"

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:pizza.views"

mc:Ignorable="d"

Background="Transparent"

Title="Итальянский Двор"

WindowStartupLocation="CenterScreen"

Height="900"

Width="1200">

<Window.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/styles/BeautifulButtonStyle.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Window.Resources>

<Border Background="#fcca72" CornerRadius="20">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="300"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid Background="#f2c068" Grid.Column="0">

<StackPanel Orientation="Vertical">

<Button Content="Заказы"

Style="{StaticResource BeautifulButtonStyle}"

Height="50"

Command="{Binding ToOrders}"/>

<Button Content="Инвентарь"

Style="{StaticResource BeautifulButtonStyle}"

Height="50"

Command="{Binding ToInventory}"/>

</StackPanel>

</Grid>

<Grid Grid.Column="1">

<UserControl Content="{Binding CurrentPage}"/>

</Grid>

</Grid>

</Border>

</Window>

ProductsPage.xaml:

<UserControl x:Class="pizza.views.Client.ProductsPage"

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

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

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

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

xmlns:local="clr-namespace:pizza.views.Client"

mc:Ignorable="d"

Background="Transparent"

d:DesignHeight="900"

d:DesignWidth="900">

<UserControl.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/styles/BeautifulButtonStyle.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</UserControl.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="50"/>

<RowDefinition/>

<RowDefinition Height="50"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<TextBlock Text="Меню"

FontSize="40"

FontStyle="Italic"

Foreground="AliceBlue"

VerticalAlignment="Center"

HorizontalAlignment="Center"/>

</Grid>

<Grid Grid.Row="1">

<TextBlock Text="Пицца"

FontSize="20"

FontStyle="Italic"

Foreground="AliceBlue"/>

</Grid>

<Grid Grid.Row="2">

<Grid>

<ScrollViewer VerticalScrollBarVisibility="Auto">

<ItemsControl ItemsSource="{Binding Pizzas}">

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<WrapPanel Orientation="Horizontal"

ItemHeight="250"

ItemWidth="250"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

<ItemsControl.ItemTemplate>

<DataTemplate>

<Border Background="AliceBlue"

CornerRadius="20"

BorderThickness="2"

BorderBrush="Black"

Margin="0,0,20,0"

Padding="10">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="100"/>

<RowDefinition Height="150"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Image Source="{Binding PhotoPath}"/>

</Grid>

<Grid Grid.Row="1">

<StackPanel Orientation="Vertical">

<TextBlock Text="{Binding Name, StringFormat='Имя: {0}'}"

FontSize="20"/>

<TextBlock Text="{Binding Price, StringFormat='Цена: {0}'}"

FontSize="18"/>

<Button Content="Добавить"

Margin="0,20,0,0"

Click="AddToCart\_Click"

Command="{Binding CompleteOrder}"

Style="{StaticResource BeautifulButtonStyle}"

Height="40"/>

</StackPanel>

</Grid>

</Grid>

</Border>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

</ScrollViewer>

</Grid>

</Grid>

<Grid Grid.Row="3">

<TextBlock Text="Напитки"

FontSize="20"

FontStyle="Italic"

Foreground="AliceBlue"/>

</Grid>

<Grid Grid.Row="4">

<Grid>

<ScrollViewer VerticalScrollBarVisibility="Auto">

<ItemsControl ItemsSource="{Binding Drinks}">

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<WrapPanel Orientation="Horizontal"

ItemHeight="250"

ItemWidth="250"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

<ItemsControl.ItemTemplate>

<DataTemplate>

<Border Background="AliceBlue"

CornerRadius="20"

BorderThickness="2"

BorderBrush="Black"

Margin="0,0,20,0"

Padding="10">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="100"/>

<RowDefinition Height="150"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Image Source="{Binding PhotoPath}"/>

</Grid>

<Grid Grid.Row="1">

<StackPanel Orientation="Vertical">

<TextBlock Text="{Binding Name, StringFormat='Имя: {0}'}"

FontSize="20"/>

<TextBlock Text="{Binding Price, StringFormat='Цена: {0}'}"

FontSize="18"/>

<Button Content="Добавить"

Margin="0,20,0,0"

Click="AddToCart\_Click"

Command="{Binding CompleteOrder}"

Style="{StaticResource BeautifulButtonStyle}"

Height="40"/>

</StackPanel>

</Grid>

</Grid>

</Border>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

</ScrollViewer>

</Grid>

</Grid>

</Grid>

</UserControl>

PizzaMenuWin.xaml:

<Window x:Class="pizza.views.MenuWindows.PizzaMenuWin"

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:pizza.views.MenuWindows"

mc:Ignorable="d"

Title="Итальянский Двор"

WindowStartupLocation="CenterScreen"

Height="900"

WindowStyle="None"

Width="1200">

<Window.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/styles/BeautifulButtonStyle.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Window.Resources>

<Border Background="#fcca72" CornerRadius="20">

<Grid>

<ScrollViewer VerticalScrollBarVisibility="Auto">

<ItemsControl ItemsSource="{Binding Pizzas}">

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<WrapPanel Orientation="Horizontal"

ItemHeight="300"

ItemWidth="250"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

<ItemsControl.ItemTemplate>

<DataTemplate>

<Border Background="AliceBlue"

CornerRadius="20"

BorderThickness="2"

BorderBrush="Black"

Margin="0,0,20,0"

Padding="10">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="120"/>

<RowDefinition Height="180"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Image Stretch="Fill" Source="{Binding Product.PhotoPath}"/>

</Grid>

<Grid Grid.Row="1">

<StackPanel Orientation="Vertical">

<TextBlock Text="{Binding Product.Name, StringFormat='Название: {0}'}"

FontSize="20"/>

<TextBlock Text="{Binding Product.ProductCategories.Name, StringFormat='Категория: {0}'}"

FontSize="18"/>

<TextBlock Text="{Binding Quantity, StringFormat='Количество: {0}'}"

FontSize="18"/>

</StackPanel>

</Grid>

</Grid>

</Border>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

</ScrollViewer>

</Grid>

</Border>

</Window>

CartPage.xaml:

<UserControl x:Class="pizza.views.OrderPage.CartPage"

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

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

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

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

xmlns:local="clr-namespace:pizza.views.OrderPage"

mc:Ignorable="d"

Background="Transparent"

Height="850"

Width="850">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<TextBlock Text="Корзина"

FontSize="40"

VerticalAlignment="Center"

HorizontalAlignment="Center"

Foreground="AliceBlue"

FontStyle="Italic"/>

</Grid>

<Grid Grid.Row="1">

<UserControl Content="{Binding CurrentView}"/>

</Grid>

</Grid>

</UserControl>

HaveCart.xaml:

<UserControl x:Class="pizza.views.OrderPage.HaveCart"

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

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

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

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

xmlns:local="clr-namespace:pizza.views.OrderPage"

mc:Ignorable="d"

Background="Transparent"

Height="900"

Width="900">

<UserControl.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/styles/BeautifulButtonStyle.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</UserControl.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="620"/>

<RowDefinition Height="180"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<ScrollViewer VerticalScrollBarVisibility="Auto">

<ItemsControl ItemsSource="{Binding CartItems}">

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<WrapPanel Orientation="Horizontal"

ItemHeight="300"

ItemWidth="250"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

<ItemsControl.ItemTemplate>

<DataTemplate>

<Border Background="AliceBlue"

CornerRadius="20"

BorderThickness="2"

BorderBrush="Black"

Margin="0,0,20,0"

Padding="10">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="120"/>

<RowDefinition Height="180"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Image Stretch="Fill" Source="{Binding Product.PhotoPath}"/>

</Grid>

<Grid Grid.Row="1">

<StackPanel Orientation="Vertical">

<TextBlock Text="{Binding Product.Name, StringFormat='Название: {0}'}"

FontSize="20"/>

<TextBlock Text="{Binding Product.ProductCategories.Name, StringFormat='Категория: {0}'}"

FontSize="18"/>

<TextBlock Text="{Binding Quantity, StringFormat='Количество: {0}'}"

FontSize="18"/>

<Button Content="Убрать"

Margin="0,20,0,0"

Click="Delete\_Click"

Command="{Binding DeleteCommand}"

Style="{StaticResource BeautifulButtonStyle}"

Height="40"/>

</StackPanel>

</Grid>

</Grid>

</Border>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

</ScrollViewer>

</Grid>

<Grid Grid.Row="1">

<Button Content="Заказать"

Style="{StaticResource BeautifulButtonStyle}"

Visibility="{Binding isVisible}"

Command="{Binding CreateOrder}"

VerticalAlignment="Center"

HorizontalAlignment="Center"

Width="200"

Height="60"/>

</Grid>

</Grid>

</UserControl>

NoCart.xaml:

<UserControl x:Class="pizza.views.OrderPage.NoCart"

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

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

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

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

xmlns:local="clr-namespace:pizza.views.OrderPage"

mc:Ignorable="d"

Background="Transparent"

Height="900"

Width="900">

<Grid>

<TextBlock Text="Корзина пуста"

VerticalAlignment="Center"

HorizontalAlignment="Center"

FontSize="40"

Foreground="AliceBlue"

FontStyle="Italic"/>

</Grid>

</UserControl>

HaveOrders.xaml:

<UserControl x:Class="pizza.views.ProfileOrders.HaveOrders"

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

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

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

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

xmlns:local="clr-namespace:pizza.views.ProfileOrders"

mc:Ignorable="d"

Background="Transparent"

d:DesignHeight="400"

d:DesignWidth="850">

<Grid>

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

<ItemsControl ItemsSource="{Binding Orders}">

<ItemsControl.ItemTemplate>

<DataTemplate>

<Border BorderBrush="Black"

BorderThickness="1"

Margin="10"

Background="AliceBlue"

CornerRadius="10"

Padding="10"

Width="240"

Height="200">

<StackPanel>

<TextBlock Text="{Binding OrderID, StringFormat='Номер заказа: {0}'}" FontWeight="Bold"/>

<TextBlock Text="{Binding OrderDate, StringFormat='Дата Заказа: {0:dd/MM/yyyy}'}"/>

<TextBlock Text="{Binding Status, StringFormat='Статус: {0}'}"/>

<ItemsControl ItemsSource="{Binding OrderDetails}">

<ItemsControl.ItemTemplate>

<DataTemplate>

<StackPanel Orientation="Vertical">

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

<TextBlock Text="{Binding Quantity, StringFormat='Всего: {0}'}"/>

<TextBlock Text="{Binding Price, StringFormat='Цена: {0:C}'}"/>

</StackPanel>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

</StackPanel>

</Border>

</DataTemplate>

</ItemsControl.ItemTemplate>

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<UniformGrid Columns="3"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

</ItemsControl>

</StackPanel>

</Grid>

</UserControl>

NoOrders.xaml:

<UserControl x:Class="pizza.views.ProfileOrders.NoOrders"

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

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

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

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

xmlns:local="clr-namespace:pizza.views.ProfileOrders"

mc:Ignorable="d"

Background="Transparent"

d:DesignHeight="400"

d:DesignWidth="850">

<Grid>

<TextBlock Text="Заказов еще не было"

FontSize="56"

VerticalAlignment="Center"

HorizontalAlignment="Center"

Foreground="AliceBlue"

FontStyle="Italic"/>

</Grid>

</UserControl>

Drinks.xaml:  
<UserControl x:Class="pizza.views.DrinksPage"

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

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

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

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

xmlns:local="clr-namespace:pizza.views"

mc:Ignorable="d"

Background="Transparent"

Height="850"

Width="850">

<Grid>

</Grid>

</UserControl>

Menues.xaml:

<UserControl x:Class="pizza.views.Menues"

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

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

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

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

xmlns:local="clr-namespace:pizza.views"

mc:Ignorable="d"

Background="Transparent"

Height="850"

Width="850">

<Grid>

<ItemsControl ItemsSource="{Binding MenuItems}">

<ItemsControl.ItemTemplate>

<DataTemplate>

<Border BorderBrush="Gray" BorderThickness="1" Margin="10" Padding="10" CornerRadius="5" Background="White">

<StackPanel>

<TextBlock Text="{Binding Name}" FontSize="20" FontWeight="Bold" />

<TextBlock Text="{Binding Description}" FontSize="16" />

</StackPanel>

</Border>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

</Grid>

</UserControl>

MenuPage.xaml:

<UserControl x:Class="pizza.views.MenuPage"

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

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

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

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

xmlns:local="clr-namespace:pizza.views"

mc:Ignorable="d"

Background="Transparent"

Height="900"

Width="900">

<UserControl.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/styles/BeautifulButtonStyle.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</UserControl.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Button Content="Вернуться"

VerticalAlignment="Center"

HorizontalAlignment="Left"

Margin="10,0,0,0"

Command="{Binding BackToMenu}"

Style="{StaticResource BeautifulButtonStyle}"

Visibility="{Binding IsBackButtonVisible}"/>

<TextBlock Text="Меню"

FontSize="40"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Foreground="AliceBlue"

FontStyle="Italic"/>

</Grid>

<Grid Grid.Row="1">

<UserControl Content="{Binding CurrentPage}"/>

</Grid>

</Grid>

</UserControl>

<Window x:Class="pizza.views.PizzaClientWindow"

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:pizza.views"

mc:Ignorable="d"

Background="Transparent"

Title="Итальянский Двор"

WindowStartupLocation="CenterScreen"

Height="900"

Width="1200">

<Window.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/styles/BeautifulButtonStyle.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Window.Resources>

<Border Background="#fcca72" CornerRadius="20">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="300"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid Background="#f2c068" Grid.Column="0">

<StackPanel Orientation="Vertical">

<Button Content="Меню"

Style="{StaticResource BeautifulButtonStyle}"

Height="50"

Command="{Binding ToMenuCommand}"/>

<Button Content="Корзина"

Style="{StaticResource BeautifulButtonStyle}"

Height="50"

Command="{Binding ToOrderCommand}"/>

<Button Content="Профиль"

Style="{StaticResource BeautifulButtonStyle}"

Height="50"

Command="{Binding ToProfileCommand}"/>

</StackPanel>

</Grid>

<Grid Grid.Column="1">

<UserControl Content="{Binding CurrentPage}"/>

</Grid>

</Grid>

</Border>

</Window>

<UserControl x:Class="pizza.views.PizzasPage"

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

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

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

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

xmlns:local="clr-namespace:pizza.views"

mc:Ignorable="d"

Background="Transparent"

Height="850"

Width="850">

<Grid>

<ItemsControl ItemsSource="{Binding Pizzas}">

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<WrapPanel Orientation="Horizontal" ItemWidth="200" ItemHeight="300"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

<ItemsControl.ItemTemplate>

<DataTemplate >

<Border BorderBrush="Black"

BorderThickness="2"

Background="White"

CornerRadius="20">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="100"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Image Source="{Binding PhotoPath}"/>

</Grid>

<Grid Grid.Row="1">

<StackPanel Orientation="Vertical">

<TextBlock Text="{Binding Name}"

FontSize="20"

/>

<TextBlock Text="{Binding Price, StringFormat='Цена: {0:d} $'}"/>

</StackPanel>

</Grid>

</Grid>

</Border>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

</Grid>

</UserControl>

<UserControl x:Class="pizza.views.ProfilePage"

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

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

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

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

xmlns:local="clr-namespace:pizza.views"

mc:Ignorable="d"

Background="Transparent"

Height="850"

Width="850">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="350"/>

<RowDefinition Height="50"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<TextBlock Text="Профиль"

FontSize="30"

Foreground="AliceBlue"

FontStyle="Italic"

HorizontalAlignment="Center"

VerticalAlignment="Center"/>

</Grid>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid Grid.Column="0">

<Image VerticalAlignment="Center"

HorizontalAlignment="Center"

Source="/assets/man.jpg"/>

</Grid>

<Grid Grid.Column="1">

<Border Background="AliceBlue"

CornerRadius="30"

BorderThickness="2"

BorderBrush="Black">

<StackPanel Orientation="Vertical"

VerticalAlignment="Center">

<TextBlock Text="{Binding FirstName, StringFormat='Имя: {0:d}'}"

FontSize="20"/>

<TextBlock Text="{Binding LastName, StringFormat='Фамилия: {0:d}'}"

FontSize="20"/>

<TextBlock Text="{Binding Email, StringFormat='Эл. почта: {0:d}'}"

FontSize="20"/>

<TextBlock Text="{Binding PhoneNumber, StringFormat='Номер телефона: {0:d}'}"

FontSize="20"/>

<TextBlock Text="{Binding Address, StringFormat='Адрес: {0:d}'}"

FontSize="20"/>

</StackPanel>

</Border>

</Grid>

</Grid>

<Grid Grid.Row="2">

<TextBlock Text="Последние заказы"

FontSize="30"

Foreground="AliceBlue"

FontStyle="Italic"

VerticalAlignment="Center"

HorizontalAlignment="Center"/>

</Grid>

<Grid Grid.Row="3">

<UserControl Content="{Binding CurrentOrdersPage}"/>

</Grid>

</Grid>

</UserControl>

<UserControl x:Class="pizza.views.StartPage"

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

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

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

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

xmlns:local="clr-namespace:pizza.views"

mc:Ignorable="d"

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

<UserControl.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/styles/BeautifulButtonStyle.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</UserControl.Resources>

<Border Background="#fcca72" CornerRadius="20">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="60"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid Grid.Row="0"

VerticalAlignment="Center"

HorizontalAlignment="Center">

<TextBlock Text="Добро пожаловать!"

FontStyle="Italic"

Foreground="AliceBlue"

FontSize="40"/>

</Grid>

<Grid Grid.Row="1"

VerticalAlignment="Center"

HorizontalAlignment="Center">

<StackPanel Orientation="Vertical"

Width="200">

<Button Style="{StaticResource BeautifulButtonStyle}"

Content="Покупатель"

Height="50"

Command="{Binding NavigateToClientPage}"/>

<Button Style="{StaticResource BeautifulButtonStyle}"

Content="Администратор"

Height="50"

Command="{Binding NavigateToAdminPage}"/>

</StackPanel>

</Grid>

</Grid>

</Border>

</UserControl>

<Window x:Class="pizza.windows.CompeteOrder"

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:pizza.windows"

mc:Ignorable="d"

Title=""

Height="200"

Width="400"

Background="#fcca72">

<Window.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/styles/BeautifulButtonStyle.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<TextBlock Text="Вы действительно хотите завершить этот заказ?"

TextWrapping="WrapWithOverflow"

FontSize="24"

VerticalAlignment="Center"

HorizontalAlignment="Center"

FontStyle="Italic"

Foreground="AliceBlue"/>

</Grid>

<Grid Grid.Row="1">

<StackPanel Orientation="Horizontal">

<Button Content="Завершить"

Height="40"

Width="120"

Click="Сomplete\_Click"

Style="{StaticResource BeautifulButtonStyle}"

Margin="20,0,0,0"/>

<Button Content="Отмена"

Height="40"

Width="120"

Click="Close\_Click"

Style="{StaticResource BeautifulButtonStyle}"

Margin="100,0,0,0"/>

</StackPanel>

</Grid>

</Grid>

</Window>

using pizza.models;

using pizza.services;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace pizza.windows

{

/// <summary>

/// Логика взаимодействия для CompeteOrder.xaml

/// </summary>

public partial class CompeteOrder : Window

{

private readonly OrdersService \_ordersService;

public Orders Order { get; private set; }

public CompeteOrder(Orders order)

{

InitializeComponent();

\_ordersService = new OrdersService();

Order = order;

}

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

{

\_ordersService.CompleteOrder(Order);

this.Close();

}

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

{

this.Close();

}

}

}

<Window x:Class="pizza.windows.DeleteFromCart"

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:pizza.windows"

mc:Ignorable="d"

Title=""

Height="200"

Width="400"

Background="#fcca72">

<Window.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/styles/BeautifulButtonStyle.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<TextBlock Text="Вы действительно хотите удалить этот продукт из корзины?"

TextWrapping="WrapWithOverflow"

FontSize="24"

VerticalAlignment="Center"

HorizontalAlignment="Center"

FontStyle="Italic"

Foreground="AliceBlue"/>

</Grid>

<Grid Grid.Row="1">

<StackPanel Orientation="Horizontal">

<Button Content="Удалить"

Height="40"

Width="120"

Click="Delete\_Click"

Style="{StaticResource BeautifulButtonStyle}"

Margin="20,0,0,0"/>

<Button Content="Отмена"

Height="40"

Width="120"

Click="Close\_Click"

Style="{StaticResource BeautifulButtonStyle}"

Margin="100,0,0,0"/>

</StackPanel>

</Grid>

</Grid>

</Window>

using pizza.models;

using pizza.services;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace pizza.windows

{

/// <summary>

/// Логика взаимодействия для DeleteFromCart.xaml

/// </summary>

public partial class DeleteFromCart : Window

{

private readonly CartService \_cartService;

public Products Product { get; private set; }

public DeleteFromCart(Products product)

{

InitializeComponent();

\_cartService = new CartService();

Product = product;

}

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

{

this.Close();

}

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

{

\_cartService.DeleteProductFromCart(Product);

this.Close();

}

}

}

<Window x:Class="pizza.windows.RefilInventory"

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:pizza.windows"

mc:Ignorable="d"

Height="200"

Width="400"

Background="#fcca72">

<Window.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/styles/BeautifulButtonStyle.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<TextBlock Text="На сколько пополнить ?"

TextWrapping="WrapWithOverflow"

FontSize="24"

VerticalAlignment="Center"

HorizontalAlignment="Center"

FontStyle="Italic"

Foreground="AliceBlue"/>

</Grid>

<Grid Grid.Row="1">

<TextBox Height="40" x:Name="RefilTextBox"/>

</Grid>

<Grid Grid.Row="2">

<StackPanel Orientation="Horizontal">

<Button Content="Пополнить"

Height="40"

Width="120"

Click="Button\_Click"

Style="{StaticResource BeautifulButtonStyle}"

Margin="20,0,0,0"/>

<Button Content="Отмена"

Height="40"

Width="120"

Click="Button\_Click\_1"

Style="{StaticResource BeautifulButtonStyle}"

Margin="100,0,0,0"/>

</StackPanel>

</Grid>

</Grid>

</Window>

using pizza.models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace pizza.windows

{

/// <summary>

/// Логика взаимодействия для RefilInventory.xaml

/// </summary>

public partial class RefilInventory : Window

{

private readonly ItalyPizzaEntities entities;

public Inventory Inventory { get; set; }

public RefilInventory(Inventory inventory)

{

InitializeComponent();

Inventory = inventory;

entities = new ItalyPizzaEntities();

}

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

{

if(String.IsNullOrEmpty(RefilTextBox.Text) || String.IsNullOrWhiteSpace(RefilTextBox.Text))

{

MessageBox.Show("Поле ввода пустое");

return;

}

if (Int32.TryParse(RefilTextBox.Text, out int result))

{

var a = entities.Inventory.Where(inv => inv.InventoryID == Inventory.InventoryID).FirstOrDefault();

if(a != null)

{

a.Quantity = result;

entities.SaveChanges();

}

else

{

MessageBox.Show("Продукт не найден");

return;

}

this.Close();

}

else

{

MessageBox.Show("Не число");

return;

}

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

this.Close();

}

}

}

<Window x:Class="pizza.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:pizza"

mc:Ignorable="d"

Background="Transparent"

Title="Итальянский Двор"

WindowStartupLocation="CenterScreen"

Height="900"

Width="1200">

<Window.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/styles/BeautifulButtonStyle.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Window.Resources>

<Border Background="#fcca72" CornerRadius="20">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="60"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid Grid.Row="0"

VerticalAlignment="Center"

HorizontalAlignment="Center">

<TextBlock Text="Добро пожаловать!"

FontStyle="Italic"

Foreground="AliceBlue"

FontSize="40"/>

</Grid>

<Grid Grid.Row="1"

VerticalAlignment="Center"

HorizontalAlignment="Center">

<StackPanel Orientation="Vertical"

Width="200">

<Button Style="{StaticResource BeautifulButtonStyle}"

Content="Покупатель"

Height="50"

Command="{Binding NavigateToClientWindow}"

Click="Button\_Click\_1"/>

<Button Style="{StaticResource BeautifulButtonStyle}"

Content="Администратор"

Height="50"

Command="{Binding NavigateToAdminWindow}"

Click="Button\_Click"/>

</StackPanel>

</Grid>

</Grid>

</Border>

</Window>