# **№ 4-5 Программное средство продажи товаров или услуг на WPF**

## Задание

Создать приложение для продажи товаров (тематику выберите самостоятельно) или услуг (спа, ремонт, фитнес и т.п.) на основе шаблона проекта WPF. Роли – администратор, клиент. Требования:

1. Каждый товар/услуга должен иметь название краткое и полное, описание, изображение (можно несколько), категория, рейтинг, цену, количество и другие параметры (цвет, размер, страна доставки, скидки, нет в наличии, связанные товары/услуги, количество купленных, производство). Товары/услуги сохраняйте в файле любого формата (json, xml).
2. Создайте окна: для вывода всех товаров/услуг (списком, таблицей); детализации каждого товар/услуги с возможностью удаления и редактирования, окно для добавления нового товара/услуги. Добавьте возможность фильтрации товаров/услуг по разным критериям, поиска, выборки по категориям, диапазону цен. Удобно создать одно окно разделенное на панели: для рабочей области (работа с товарами/услугами) и управления.
3. Все действия с товарами/услугами нужно реализовывать через Command. Используйте привязки данных к ЭУ.
4. Должен поддерживать респонсивный дизайн. Панели инструментов, меню, основное окно и др. элементы должны изменять размеры, ориентацию или расположение пропорционально изменениям размера окна. Используйте вложенные контейнеры компоновки (Grid, StrackPanel, DockPanel и т.д.) и свойства. Можно использовать  Ribbon control из библиотеки RibbonControlsLibrary.dll  При необходимости можно добавить поддержку drag-and-drop (например, добавление изображений).
5. Добавьте поддержку двух языков интерфейса (например, русский и английский). Должна поддерживаться динамическая локализация на основе Resource Dictionary.
6. Подгрузить в приложение свой курсор и нарисованную свою иконку.
7. Создайте и примените **стиль** для вашего приложения.

*Приложение должно быть уникальным у каждого студента.*

## Вопросы

### ****1. Перечислите преимущества и недостатки WPF****

**Преимущества:**

* ✔ **Разделение логики и интерфейса**: благодаря XAML можно отделить UI от логики.
* ✔ **Мощная система привязки данных (data binding)**.
* ✔ **Поддержка шаблонов (templates)** — внешний вид можно полностью изменить без изменения логики.
* ✔ **Аппаратное ускорение (hardware acceleration)** через DirectX.
* ✔ **Поддержка стилей и ресурсов** для централизованного оформления.
* ✔ **Масштабируемость интерфейса** (vector-based UI).
* ✔ **Поддержка MVVM**, что упрощает тестирование и поддержку кода.

**Недостатки:**

* ❌ Крутая кривая обучения.
* ❌ Производительность может падать при неправильной реализации привязок или визуальных эффектов.
* ❌ Проблемы с кроссплатформенностью (работает только на Windows).
* ❌ Не всегда интуитивно понятные ошибки при компиляции XAML.

### ****2. Зачем нужен язык XAML? Как он используется в WPF?****

**XAML (Extensible Application Markup Language)** — декларативный язык разметки, используемый для описания интерфейса пользователя.

**Используется в WPF для:**

* Определения интерфейса приложения (окна, кнопки, списки, стили, шаблоны и т.д.).
* Автоматического связывания UI с кодом на C# (через x:Name, DataContext, Binding и т.д.).

**Преимущества:**

* Отделяет логику от интерфейса.
* Упрощает чтение и поддержку UI.
* Поддерживается инструментами (например, Visual Studio и Blend).

### ****3. Какие бывают контейнеры компоновки (панели)?****

**Основные контейнеры (панели):**

* Grid — табличная разметка, мощная и гибкая.
* StackPanel — укладывает элементы в **столбик** (Vertical) или **в строку** (Horizontal).
* WrapPanel — оборачивает элементы на следующую строку, если не хватает места.
* DockPanel — элементы можно "прикреплять" к сторонам контейнера.
* Canvas — абсолютное позиционирование по координатам.
* UniformGrid — делит пространство на равные ячейки.

### ****4. Перечислите основные группы элементов управления. Объектная модель WPF****

**Основные группы элементов управления:**

* 📥 **Ввод**: TextBox, PasswordBox, ComboBox, CheckBox, RadioButton.
* 🧭 **Навигация и команда**: Button, Menu, Toolbar, Hyperlink.
* 🧱 **Контейнеры**: Grid, StackPanel, Canvas, DockPanel.
* 📊 **Вывод данных**: TextBlock, Label, Image, ProgressBar.
* 📜 **Списки и деревья**: ListBox, ListView, TreeView, DataGrid.

**Объектная модель WPF:**

* Основана на древовидной структуре — **визуальное дерево**.
* Наследование от UIElement, FrameworkElement, Control, ContentControl, Panel и т.д.
* Пример: Button : ContentControl : Control : FrameworkElement : UIElement : Visual : DependencyObject.

### ****5. Что такое DependencyProperty (свойство зависимости)?****

**DependencyProperty** — это особый тип свойства в WPF, который:

* Поддерживает **систему привязки данных**,
* Участвует в **наследовании значений**,
* Позволяет **эффективно реагировать на изменения** (через события),
* Позволяет задавать значения через **стили, триггеры, анимации** и т.д.

**Когда использовать:**

* Когда нужно, чтобы свойство участвовало в системе WPF (binding, animation, styles и т.д.).

### ****6. Обработка событий в WPF и маршрутизованные события****

**Обработка событий:**

* Можно подписываться в XAML или коде (Click="OnClick", myButton.Click += ...).
* Есть привязка через команды.

**Маршрутизованные события (Routed Events)** — это особенность WPF, при которой событие может **"путешествовать" по дереву элементов**:

* **Bubbling** — всплывающее (от дочернего вверх к родителю),
* **Tunneling** — туннельное (от корня вниз к дочернему),
* **Direct** — прямое, как в обычном .NET

**Основные группы событий:**

* Ввод (мышь, клавиатура): MouseDown, KeyDown, TextInput.
* Жизненный цикл: Loaded, Unloaded, Initialized.
* Фокус: GotFocus, LostFocus.
* Перетаскивание: DragEnter, Drop.

### ****7. Что такое Resource Dictionary?****

**Resource Dictionary** — это хранилище ресурсов WPF: стилей, шаблонов, кистей, строк и т.д.

### ****8. Что такое стиль и как его создать? Преимущества****

**Стиль (Style)** — набор свойств для оформления элементов управления.

### ****9. Что такое Command? Как используется в WPF?****

**Command** — паттерн, реализующий действие как объект. В WPF это ICommand.

**Зачем нужен в WPF:**

* Позволяет связать UI и действия **без обработчиков событий**.
* Поддерживает CanExecute — включение/отключение кнопок.
* Отлично работает с MVVM.

# Пример интерфейса:



 









