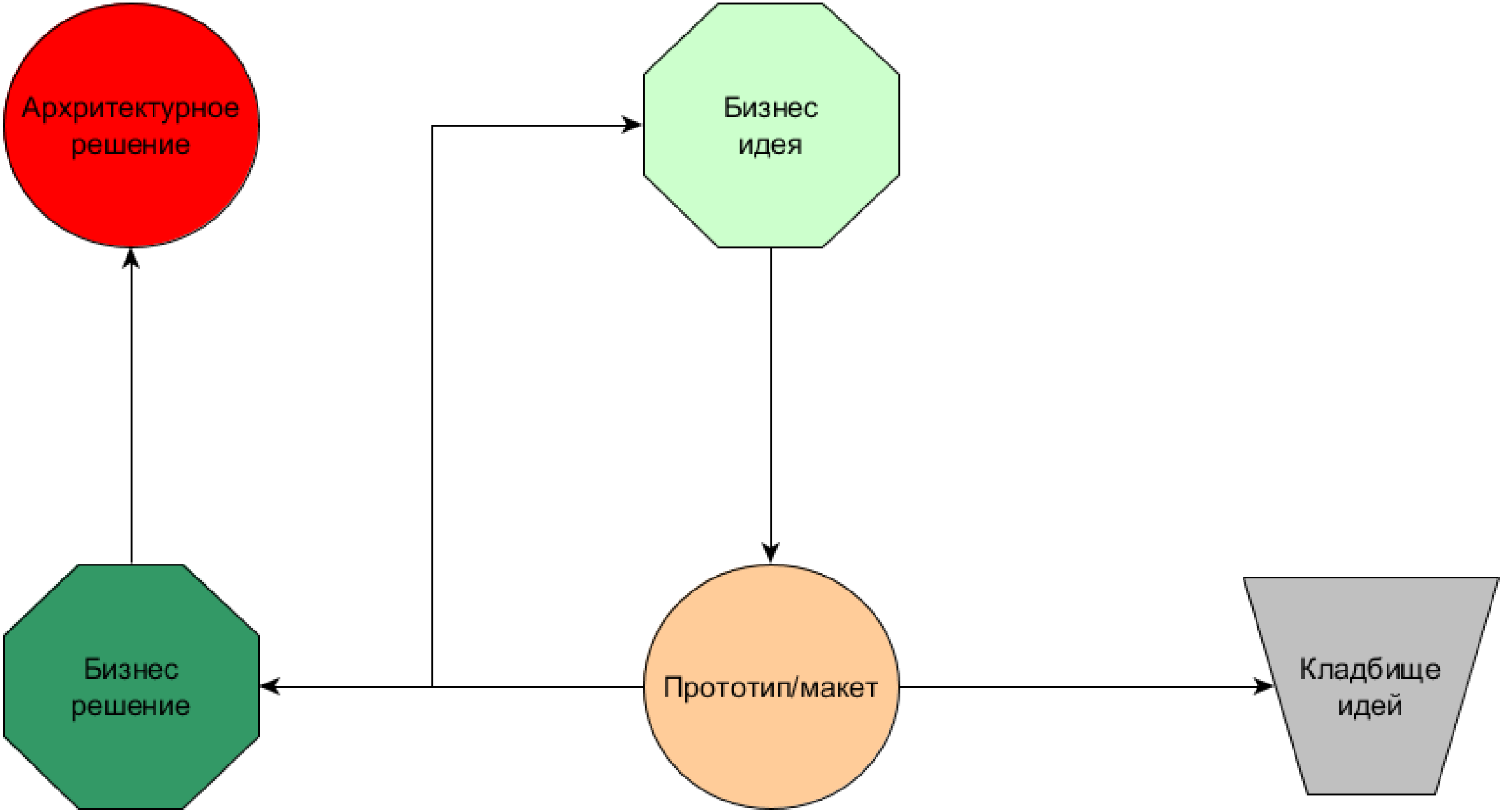
Архитектура desktop-приложений

Часть I

Архитектурные решения - это выбор принципов, которые позволят удовлетворить требования.


Архритектурное решение
Бизнес идея
Бизнес решение
Прототип/макет
Кладбище идей

О каких UI идет речь?

Допущения:

- Desktop-приложения, ограничения по ресурсам нет, 1GB RAM - это нормально.
- Сложные приложения. Приложение предназначенные для работы с более чем 3-мя бизнес сущностями.
- Приложение не критично к скорости работы (это не про кассовые терминалы).
- Приложение активно развивается, требования меняются, функционал растет.

Что не так с UI?

- UI - самое уязвимое место для изменения требований.
- Большинство изменений в бизнес процессе отражаются на UI приложения.
- По UI судят о программе в целом.
- в UI очень много взаимодействующих между собой частей, связей много и они разные (одно направленные/двунаправленные, одноуровневые/разноуровневые)

Характеристики кодовой базы с точки зрения бизнеса

- Безопасность.
- Производительность.
- Сопровождаемость. <- *Вы здесь*

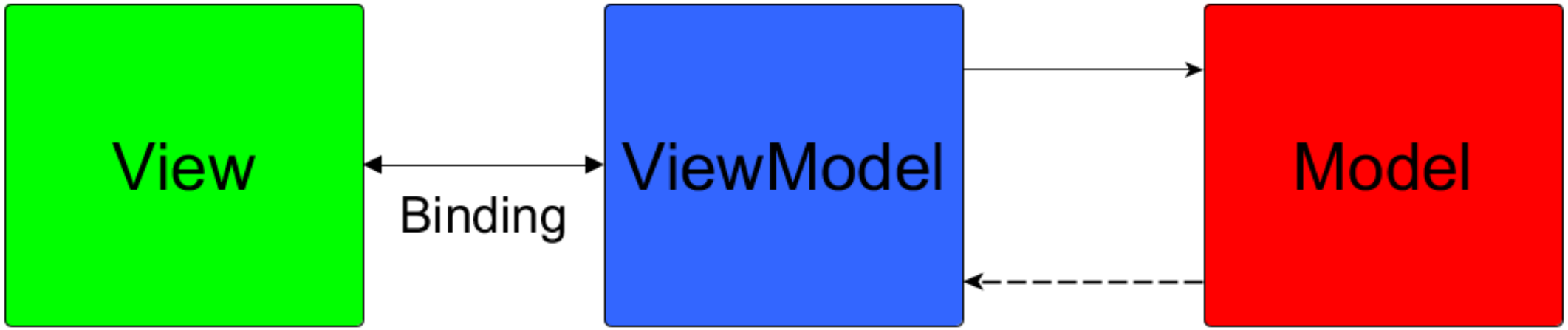
Сопровождаемость

- Скорость внесения изменений
- Скорость включения нового человека в разработку

Model-View-Controller/Presenter/ViewModel – всё просто

- смешивать данные, логику, представление, анимации, стили, темы в одном месте - это плохо
- технически разделяя разные аспекты приложения мы упрощаем разработку
- структуры данных меняются реже, чем пользовательский интерфейс
- одни и те же данные могут быть представлены разными способами
- пользователь может взаимодействовать с приложением разными способами (мышь, клавиатура, манипуляторы, голос)

MVVM



.NET: WPF, UWP, Xamarin

- DataContext
- Binding
- DependencyObject/DependencyProperty
- INotifyPropertyChanged
- INotifyCollectionChanged
- ICommand

```
public interface INotifyPropertyChanged
{
    event PropertyChangedEventHandler PropertyChanged;
}

public interface INotifyCollectionChanged
{
    event NotifyCollectionChangedEventHandler CollectionChanged;
}

public interface ICommand
{
    event EventHandler CanExecuteChanged;
    bool CanExecute(object parameter);
    void Execute(object parameter);
}
```

```
<UserControl
    x:Class="SlidesExample.AccountView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="clr-namespace:SlidesExample">
    <UserControl.DataContext>
        <local:AccountViewModel />
    </UserControl.DataContext>
    <StackPanel>
        <TextBox Text="{Binding Path=AccountName}" />
        <ListBox ItemsSource="{Binding Path=UserPresentations}" />
        <Button
            Command="{Binding Path=Load}"
            Content="Load" />
    </StackPanel>
</UserControl>
```

И тут что-то пошло не так...

- Model-мутант
- ViewModel-монстр
- View-чудовище

Model-мутант

- ObservableCollection<T>
- INotifyPropertyChanged
- Selected...

Скажите свойствам "нет"

- нет свойств - нет желания привязаться к ним
- DTO, Entity будут инкапсулированы настоящими объектами

Обертки-представления

- объекты для создания привязок
- избавляет модель от лишнего состояния
- изолирует View от Model
- валидация ввода пользователя
- агрегация по нескольким моделям

```
public interface IUser
{
    string FirstName();
    string LastName();
    string Title();
}
public sealed class User : IUser {...}

public sealed class UserPresentation
{
    private readonly IUser _user;
    public UserPresentation(IUser user)
    {
        _user = user;
    }
    public string Text => $"{_user.Title()}. {_user.FirstName()} {_user.LastName()}";
}

public sealed class UserShortPresentation
{
    private readonly IUser _user;
    public UserShortPresentation(IUser user)
    {
        _user = user;
    }
    public string Text => _user.LastName();
}
```

ViewModel-монстр

- 1-2-3 KLOC
- множество приватных методов, _load.., _canExecute..., _execute..., _onSelect..., _onChanged...

```
1    using [redacted];
2    using [redacted];
3    using [redacted];
4    using [redacted];
5
6    namespace [redacted]
7    {
8        public class [redacted] : [redacted]
9        {
10           private ObservableCollection<[redacted]> _result[redacted] = new ObservableCollection<[redacted]>();
11
12           public [redacted]()
13               : base([redacted])
14           {
48
49           #region Properties
387
388
389          #region Commands
827
828
829          #region Methods
1529     }
1530 }
```

RelayCommand/DelegateCommand

- Команды отдельным классом
- Команды делегируют выполнение работы модели или сервису
- Необходимые данные передаются команде через конструктор или параметр вызова
- Типизированные команды Command<T>: ICommand
- Расширенные команды Undo/Redo
- Команды-обертки для логирования, для перехвата ошибок

ObservableCollection<T>

- Отдельный класс-коллекция
- Загрузка
- Добавление/Удаление
- Фильтрация
- Сортировка
- Очистка
- Индексы
- Инкапсуляция специфичной реализации AsyncObservableCollection, BatchObservableCollection

```
public sealed class UserPresentations : ObservableCollection<UserPresentation>
{
    public void Add(...) {...}
    public void Remove(...) {...}
    public void Load() {...}
    public void Clear() {...}
    public void Filter(...) {...}
    public void Sort(...) {...}
}
public sealed class AccountViewModel
{
    public UserPresentations UserPresentations { get; }
    public ICommand Load { get; }
    public ICommand Add { get; }
    public ICommand Delete { get; }
}
```

```
public sealed class LoadCommand : ICommand
{
    private readonly UserPresentations _userPresentations;
    private bool _loaded = false;
    public LoadCommand(UserPresentations userPresentations)
    {
        _userPresentations = userPresentations;
    }
    public bool CanExecute(object parameter)
    {
        return !_loaded;
    }
    public void Execute(object parameter)
    {
        _userPresentations.Load();
        _loaded = true;
    }
    public event EventHandler CanExecuteChanged;
}
```

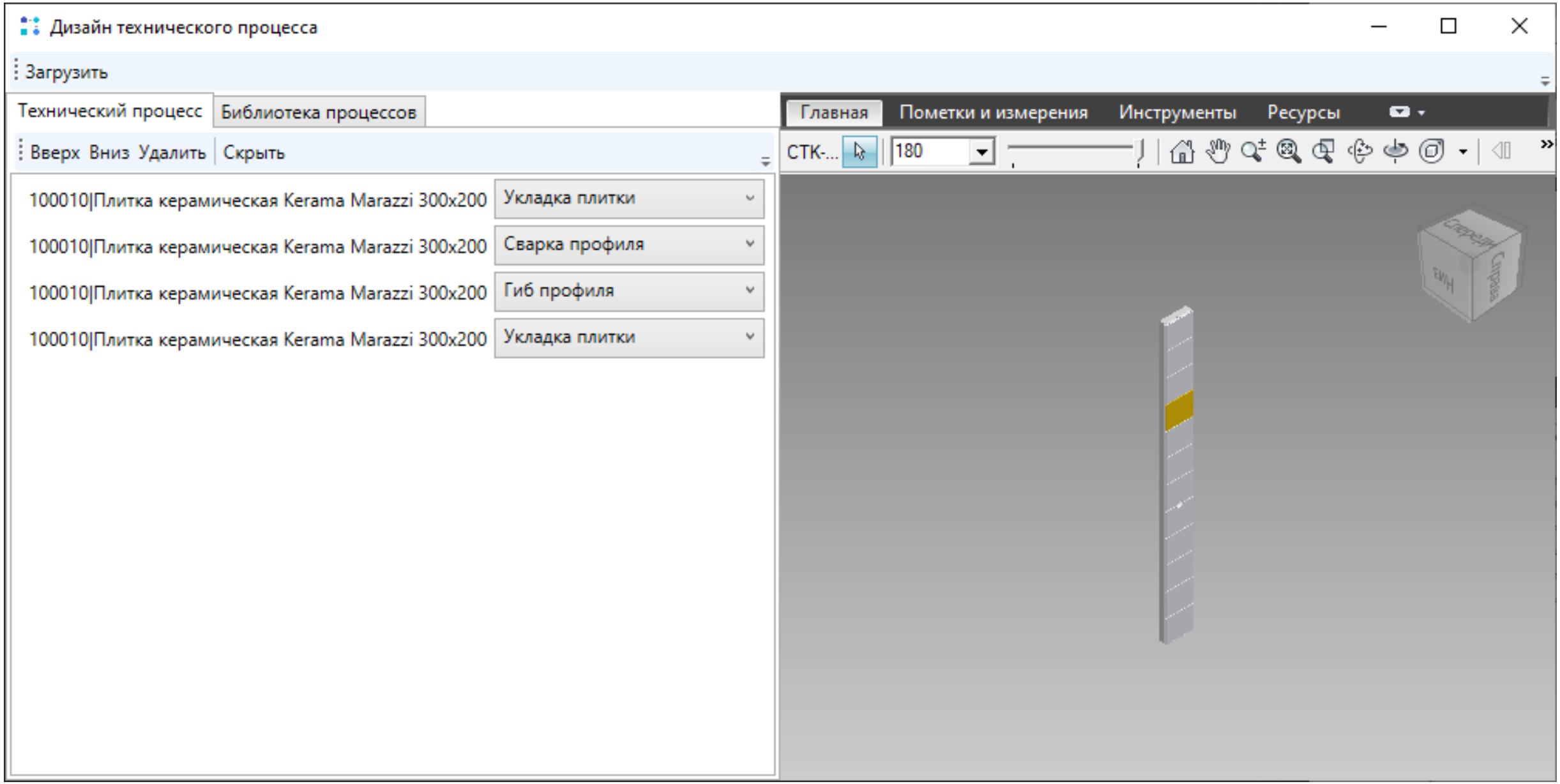
```
public interface IAccount
{
    IEnumerable<IUser> Users();
}
public sealed class UserPresentations : ObservableCollection<UserPresentation>
{
    private readonly IAccount _account;
    public UserPresentations(IAccount account)
    {
        _account = account;
    }
    public void Load()
    {
        foreach (var user in _account.Users())
        {
            Add(new UserPresentation(user));
        }
    }
}
public sealed class AccountViewModel
{
    public AccountViewModel(IAccount account)
    {
        UserPresentations = new UserPresentations(account);
        Load = new LoadCommand(UserPresentations);
        ...
    }
...
}
```

View-чудовище

- 1-2-3 KLOC
- комментарии поясняющие какая это часть UI
- n-вложенные Grid-ы, StackPanel-и

UserControls

- для связанных одной функцией элементов, например:
  - указать путь к файлу
  - параметры соединения
- DataTemplate
- таблицы, деревья


Дизайн технического процесса
Загрузить
Технический процесс
Библиотека процессов
Вверх Вниз Удалить Скрыть
100010|Плитка керамическая Kerama Marazzi 300x200
Укладка плитки
100010|Плитка керамическая Kerama Marazzi 300x200
Сварка профиля
100010|Плитка керамическая Kerama Marazzi 300x200
Гиб профиля
100010|Плитка керамическая Kerama Marazzi 300x200
Укладка плитки
Главная
Пометки и измерения
Инструменты
Ресурсы
СТК-...
180

```
<Window ...>
    <DockPanel>
        <ToolBar DockPanel.Dock="Top">
            <Button Click="ButtonBase_OnClick">Загрузить</Button>
        </ToolBar>
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="*" />
            </Grid.ColumnDefinitions>
            <TabControl Grid.Column="0">
                <TabItem Header="Технический процесс">
                    <DockPanel>
                        <ToolBar DockPanel.Dock="Top">
                            <Button>Вверх</Button>
                            <Button>Вниз</Button>
                            <Button>Удалить</Button>
                            <Separator />
                            <Button Click="Hide_OnClick">Скрыть</Button>
                        </ToolBar>
                        <ListView
                            HorizontalContentAlignment="Stretch"
                            ItemsSource="{Binding Path=ImpProcessPresenters}"
                            SelectedItem="{Binding Path=SelectedImpProcessPresenter}">
                            <ListView.ItemTemplate>
                                <DataTemplate DataType="local:ImpProcessPresenter">
                                    <Grid>
                                        <Grid.ColumnDefinitions>
                                            <ColumnDefinition Width="*" />
                                            <ColumnDefinition Width="*" />
                                        </Grid.ColumnDefinitions>
                                        <TextBlock
                                            Grid.Column="0"
                                            Margin="5"
                                            VerticalAlignment="Center"
                                            Text="{Binding Path=Part}"
                                            TextAlignment="Justify" />
                                        <ComboBox
                                            Grid.Column="1"
                                            ItemsSource="{Binding Path=LibProcesses}"
                                            SelectedItem="{Binding Path=Process}" />
                                    </Grid>
                                </DataTemplate>
                            </ListView.ItemTemplate>
                        </ListView>
                    </DockPanel>
                </TabItem>
                <TabItem Header="Библиотека процессов">
                    <DockPanel>
                        <ToolBar DockPanel.Dock="Top">
                            <Button>Добавить</Button>
                            <Button>Удалить</Button>
                        </ToolBar>
                        <ListView
                            HorizontalContentAlignment="Stretch"
                            ItemsSource="{Binding Path=ProcessLibPresenters}"
                            SelectedItem="{Binding Path=SelectedLibProcessPresenter}">
                            <ListView.ItemTemplate>
                                <DataTemplate DataType="local:LibProcessPresenter">
                                    <Grid>
                                        <Grid.ColumnDefinitions>
                                            <ColumnDefinition Width="*" />
                                            <ColumnDefinition Width="*" />
                                        </Grid.ColumnDefinitions>
                                        <TextBox
                                            Grid.Column="0"
                                            Margin="5"
                                            Text="{Binding Path=Process}" />
                                        <TextBox
                                            Grid.Column="1"
                                            Margin="5"
                                            Text="{Binding Path=Parts}" />
                                    </Grid>
                                </DataTemplate>
                            </ListView.ItemTemplate>
                        </ListView>
                    </DockPanel>
                </TabItem>
            </TabControl>
            <WindowsFormsHost
                x:Name="ReviewWindowsFormsHost"
                Grid.Column="1" />
        </Grid>
    </DockPanel>
</Window>
```

Refactoring

```
<Window ...>
    <DockPanel>
        <ToolBar DockPanel.Dock="Top">
            <Button Click="ButtonBase_OnClick">Загрузить</Button>
        </ToolBar>
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="*" />
            </Grid.ColumnDefinitions>

            <TabControl Grid.Column="0">
                <TabItem Header="Технический процесс">
                    <local:TechProcessControl x:Name="TechProcessControl" />
                </TabItem>
                <TabItem Header="Библиотека процессов">
                    <local:LibraryControl />
                </TabItem>
            </TabControl>

            <WindowsFormsHost
                x:Name="ReviewWindowsFormsHost"
                Grid.Column="1" />
        </Grid>
    </DockPanel>
</Window>
```

```
<UserControl
    x:Class="TechProcess.TechProcessControl"
    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:local="clr-namespace:TechProcess"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    d:DataContext="{d:DesignInstance Type=local:ViewModel}"
    d:DesignHeight="450"
    d:DesignWidth="800"
    mc:Ignorable="d">
    <DockPanel>
        <ToolBar DockPanel.Dock="Top">
            <Button>Вверх</Button>
            <Button>Вниз</Button>
            <Button>Удалить</Button>
            <Separator />
            <Button Click="Hide_OnClick">Скрыть</Button>
        </ToolBar>
        <ListView
            HorizontalContentAlignment="Stretch"
            ItemsSource="{Binding Path=ImpProcessPresenters}"
            SelectedItem="{Binding Path=SelectedImpProcessPresenter}">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <local:ProcessStepControl />
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </DockPanel>
</UserControl>
```

```
<UserControl
    x:Class="TechProcess.ProcessStepControl"
    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:local="clr-namespace:TechProcess"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    d:DataContext="{d:DesignInstance Type=local:ImpProcessPresenter}"
    d:DesignHeight="450"
    d:DesignWidth="800"
    mc:Ignorable="d">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <TextBlock
            Grid.Column="0"
            Margin="5"
            VerticalAlignment="Center"
            Text="{Binding Path=Part}"
            TextAlignment="Justify" />
        <ComboBox
            Grid.Column="1"
            ItemsSource="{Binding Path=LibProcesses}"
            SelectedItem="{Binding Path=Process}" />
    </Grid>
</UserControl>
```