# Вебинар 2
# Основы создания CRUD-приложений, продолжение

- MV**C** / MV**P** / MV**VM** / MV**U**
- MVVM в .NET (XAML,WPF / XAML,UWP, Xamarin / Uno,Avalonia)
- MVVM-фреймворки (Prism, Caliburn) + MaterialDesign
- MVC (MVP?) пример в web-backend (ASP.NET Core)

+ бонусы

+ бонусы

- делегаты и события .NET
- генерация фейковых данных (Bogus)
- паттерн Command
- маппинг (AutoMapper)
- CCC, AOP (Fody, PostSharp)
- логирование (Serilog)
- валидация (FluentValidation, IDataErrorInfo, DataAnnotations)

View

## Presentation tier

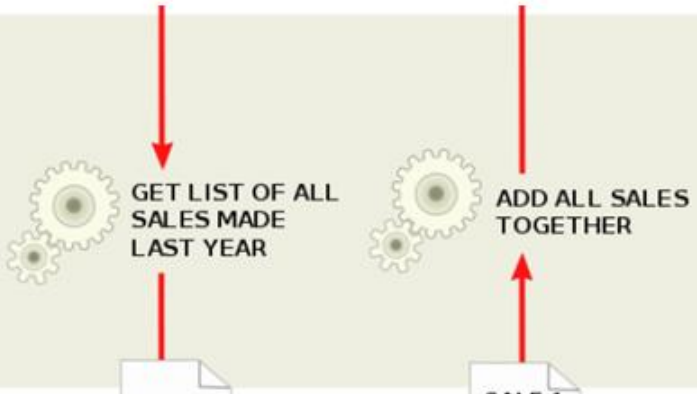
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.


Controller / Presenter / etc.

Model

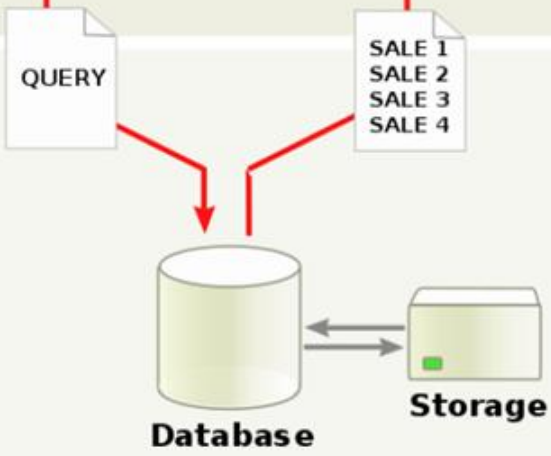
## Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.




## Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.



# События в .NET

Publisher (ex.: data loader)

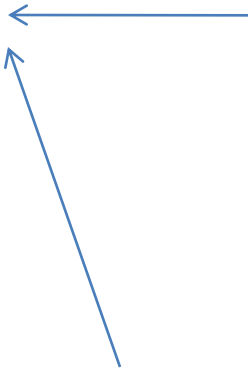
```
event EventHandler DataLoaded;
//..

DataLoaded();
```

Subscriber (ex: data processor)

```
pub.DataLoaded += ProcessData;
//..

метод ProcessData() { … }
```

Subscriber (ex.: logger)

```
pub.DataLoaded += LogData;
//..

метод LogData() { … }
```



# Сначала посмотрим на делегаты

```
class DataLoader
{
    public delegate void DataLoadedHandler(string data);

    private readonly DataLoadedHandler _dataLoadedHandler;

    public DataLoader()
    {
        _dataLoadedHandler = ProcessData;
        _dataLoadedHandler += Log;
        _dataLoadedHandler += (data) => MessageBox.Show(data);
    }

    public void Run()
    {
        //...
        _dataLoadedHandler("data!");
        //_dataLoadedHandler.Invoke("data!");
    }

    public void ProcessData(string data) { /* ... */ }

    public void Log(string data) { /* ... */ }
}
```

multicast

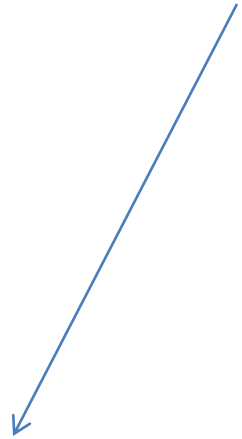
Вызов

# Action и Func

```
Action          _action1;
Action<int>     _action2;

Func<int>           _func1;
Func<string, int>   _func2;
```

```
delegate void Action1();
delegate void Action2(int param);

delegate int Function1();
delegate int Function2(string param);
```

# Делегаты -> События



```
private DataLoader _loader;

public void Subscribe1()
{
    _loader.DataLoaded += ProcessData;
    _loader.DataLoaded += (data) => MessageBox.Show(data);
}

public void ProcessData(string data)
{
    // ...
}
```

```
class DataLoader // event publisher
{
    public delegate void DataLoadedHandler(string data);
    public event DataLoadedHandler DataLoaded;

    public void Run()
    {
        //...
        DataLoaded("data!");
    }
}
```



```
private DataLoader _loader;

public void Subscribe2()
{
    _loader.DataLoaded += Log;
}

public void Log(string data)
{
    // log data
}
```

# EventHandler

```
event EventHandler<TEventArgs> OnSmth;
```

```
OnSmth += ProcessEventSmth;
```

```
void ProcessEventSmth(object sender, TEventArgs args)
{
    // ...
}
```

# Контракты событий

```
class DataLoadedEventArgs : EventArgs
{
    public string Data { get; set; }
}
```

```
interface IDataLoader
{
    event EventHandler<DataLoadedEventArgs> DataLoaded;

    void Load(string url);

    public string Mode { get; set; }
}
```

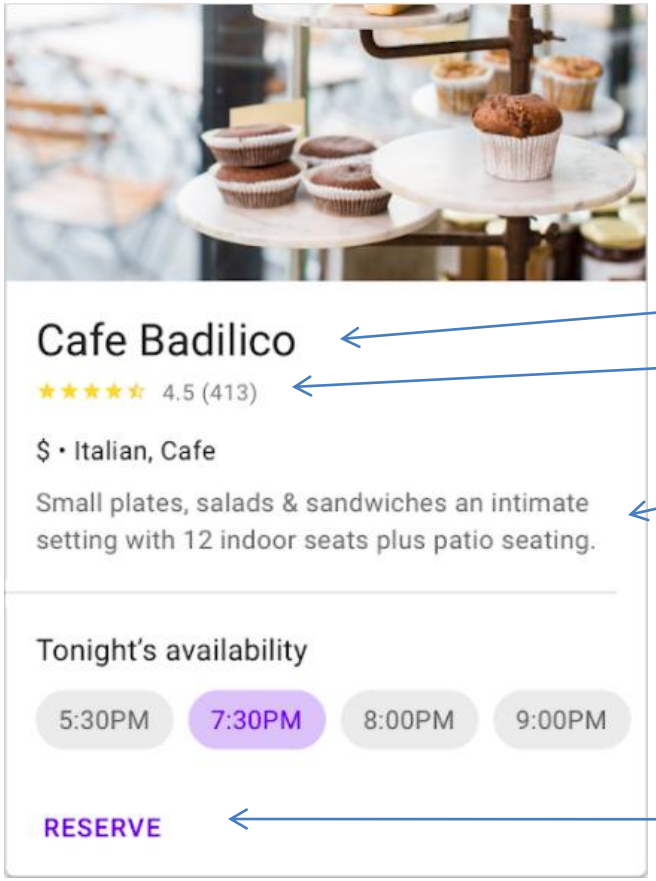
1

2

3

```
private EventHandler _MyEvent;
public event EventHandler MyEvent
{
    add { lock (this) { _MyEvent += value; } }
    remove { lock (this) { _MyEvent -= value; } }
}
```

# ViewModel example

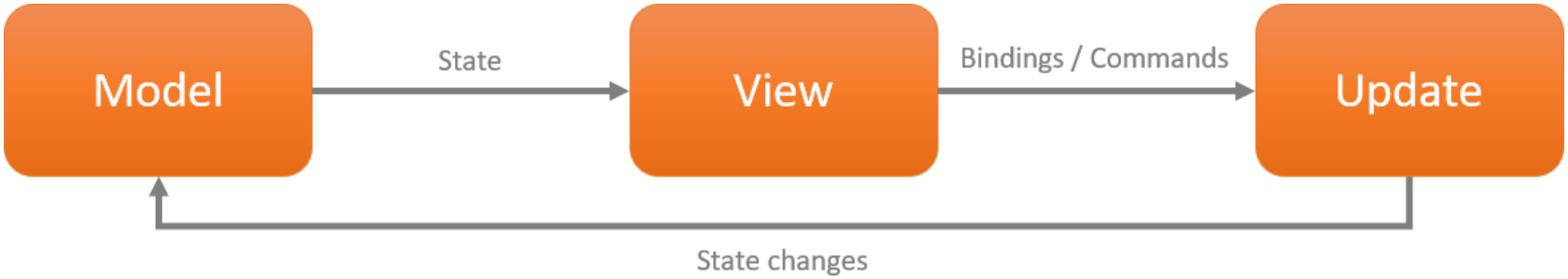



```
ReservationViewModel
{
    string ImagePath;
    string Name;
    double Rate;
    int VoteCount;
    string Description;

    List<DateTime> Availability;

    ICommand ReserveCommand;

    // ...
}
```

© https://material.io/components/cards

1) данные модели, адаптированные к представлению
2) команды представления для влияния на модель

# MVU: Model-View-Update



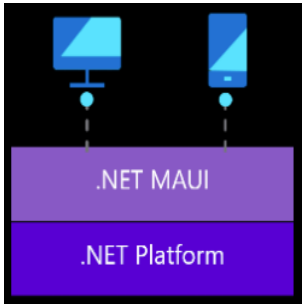

```
readonly State<int> count = 0;

[Body]
View body() => new StackLayout
{
    new Label("Welcome!"),
    new Button(
        () => $"You clicked {count} times.",
        () => count.Value ++)
    )
};
```

# MVVM в .NET

Степень мульти-платформенности,
Охват подходов





# MVVM Boilerplate

- Базовый класс для ViewModel (интерфейс INPC)
- Автоматическая привязка view к viewmodel
- Паттерн «Команда» (RelayCommand)
- Координация views (DialogService)
- Коммуникация через события (EventAggregator)
- DI (Dependency Injection)

# PRISM vs. Caliburn.Micro

1. Базовый класс для ViewModel

PRISM

```
public class MainWindowViewModel : BindableBase
{
    private string _title = "TaxiApp";
    public string Title
    {
        get { return _title; }
        set { SetProperty(ref _title, value); }
    }
```

Caliburn.Micro

```
// Можно наследоваться еще от Screen и Conductor<>...
class MainWindowViewModel : PropertyChangedBase
{
    private BindableCollection<DriverViewModel> _drivers;
    public BindableCollection<DriverViewModel> Drivers
    {
        get => _drivers;
        set
        {
            _drivers = value;
            NotifyOfPropertyChange(() => Drivers);
        }
    }
```

# PRISM vs. Caliburn.Micro

2 Автоматическая привязка view к viewmodel

PRISM

```
prism:ViewModelLocator.AutoWireViewModel="True"
```

+ naming conventions:

Ex.
DriverWindow
|
DriverWindowViewModel

Caliburn.Micro

Only naming conventions:

Ex.
DriverWindow
|
DriverWindowViewModel

# PRISM vs. Caliburn.Micro

3 Привязка к командам

PRISM

```
private DelegateCommand _removeDriverCommand;
public DelegateCommand RemoveDriverCommand =>
            _removeDriverCommand ??= new DelegateCommand(
                                        RemoveDriver,
                                        CanRemoveDriver);
```

Caliburn.Micro

```
<Button x:Name="RemoveDriver"
        Content="REMOVE DRIVER"
```

+ naming convention:

```
public void RemoveDriver()
{
    Drivers.RemoveAt(0);
    NotifyOfPropertyChange(() => CanRemoveDriver);
}
```