



AVALONIA

MOBILE
DESKTOP
WEB

СОДЕРЖАНИЕ

Введение

Краткая история

Основные возможности

Архитектура

Преимущества и недостатки

Заключение

ВВЕДЕНИЕ В AVALONIA

Avalonia UI — это кроссплатформенный фреймворк для создания современных графических приложений на C# и XAML. Он позволяет запускать одно и то же приложение на:

- Windows
- Linux
- macOS
- Android и iOS

КРАТКАЯ ИСТОРИЯ

Год	Событие
2013	Первый коммит — Perspex (эксперимент)
2015	Добавлен XAML → первая альфа
2016	Переименован в Avalonia
2019–2022	Версия 0.10 — стабильный десктоп
2023	11.0 — мобильные + веб + AOT
2024–2025	11.1+ — TV, производительность, инструменты

Что дальше?

Rich Text Editor

AI-инструменты для UI

Ещё лучше мобильный UX

Глубокая интеграция с .NET 9+

КРАТКАЯ ИСТОРИЯ

Почему выбирают Avalonia?

Один код → 7+ платформ

Полный .NET

Open Source (MIT)

Активное сообщество

Hot Reload, производительность

Avalonia = WPF 2.0 для всего мира

Почему не просто улучшить WPF?

Microsoft не может — WPF привязан к Windows API

Архитектура устарела — DirectX, Win32, COM

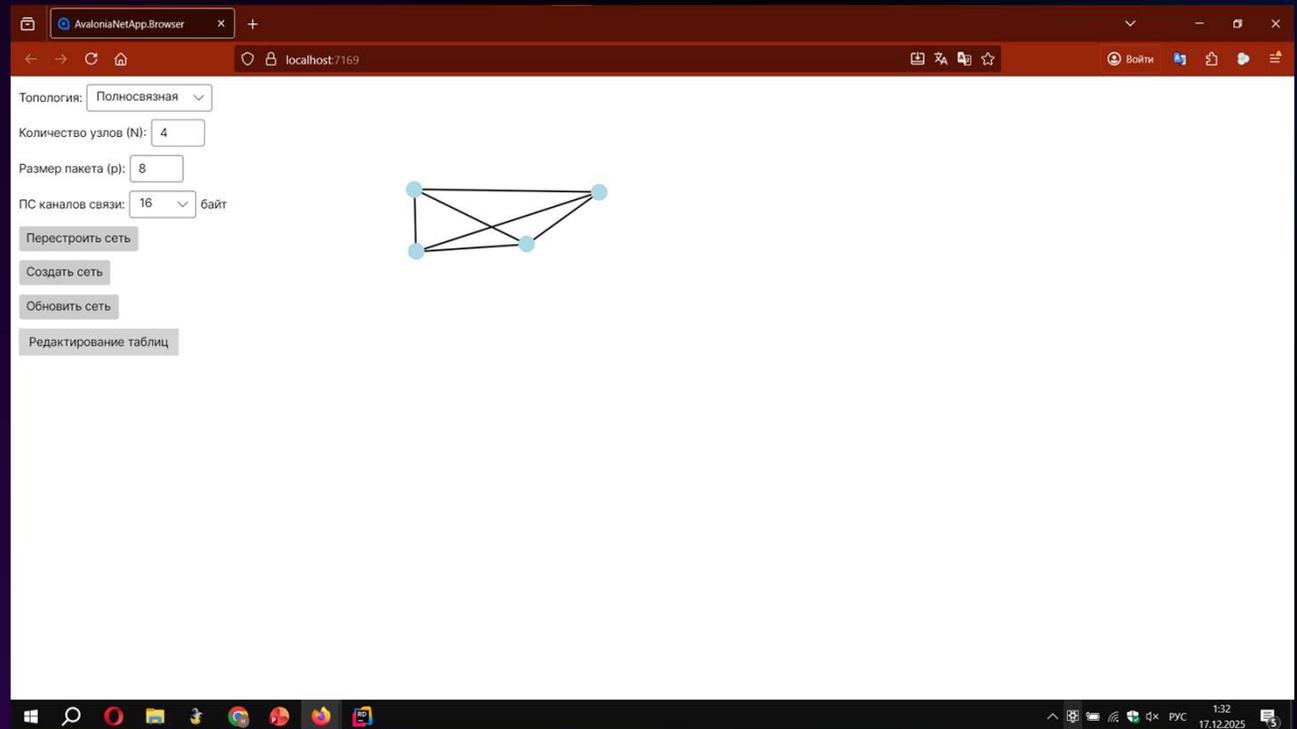
Нет планов на кросс-платформу — фокус на WinUI / MAUI

Сообщество хотело свободу → родился **Perspex** → **Avalonia**

ОСНОВНЫЕ ВОЗМОЖНОСТИ

Поддержка всех популярных платформ. Как пример:

Браузер



The screenshot shows a web browser window titled "AvaloniaNetApp.Browser" with the address bar displaying "localhost:7169". The interface includes a sidebar with the following settings:

- Топология: Полносвязная
- Количество узлов (N): 4
- Размер пакета (p): 8
- ПС каналов связи: 16 байт

Below the settings are several buttons: "Перестроить сеть", "Создать сеть", "Обновить сеть", and "Редактирование таблиц". The main content area displays a network diagram with four nodes and a fully connected mesh topology.

The Windows taskbar at the bottom shows the system tray with the time 1:32 and date 17.12.2025.

ОСНОВНЫЕ ВОЗМОЖНОСТИ

Поддержка всех популярных платформ. Как пример:

Десктоп

Скриншот интерфейса программы «Проектировщик сетей» (Network Designer) в режиме «Десктоп». Интерфейс содержит следующие элементы:

- Заголовок окна: Проектировщик сетей
- Панель параметров:
 - Топология:
 - Количество узлов (N):
 - Размер пакета (p):
 - ПС каналов связи: байт
- Кнопки управления:
 - Перестроить сеть
 - Создать сеть
 - Обновить сеть
 - Редактирование таблиц
- Диаграмма сети: Визуальное представление полносвязной топологии с 4 узлами (синие точки) и всеми возможными соединениями (черные линии).

ОСНОВНЫЕ ВОЗМОЖНОСТИ

Поддержка всех популярных платформ. Как пример:

Мобильный

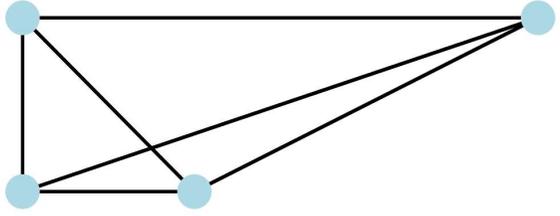
01:24 0,00 k/s Wi-Fi Vo LTE 4G LTE 23

Топология:

Количество узлов (N):

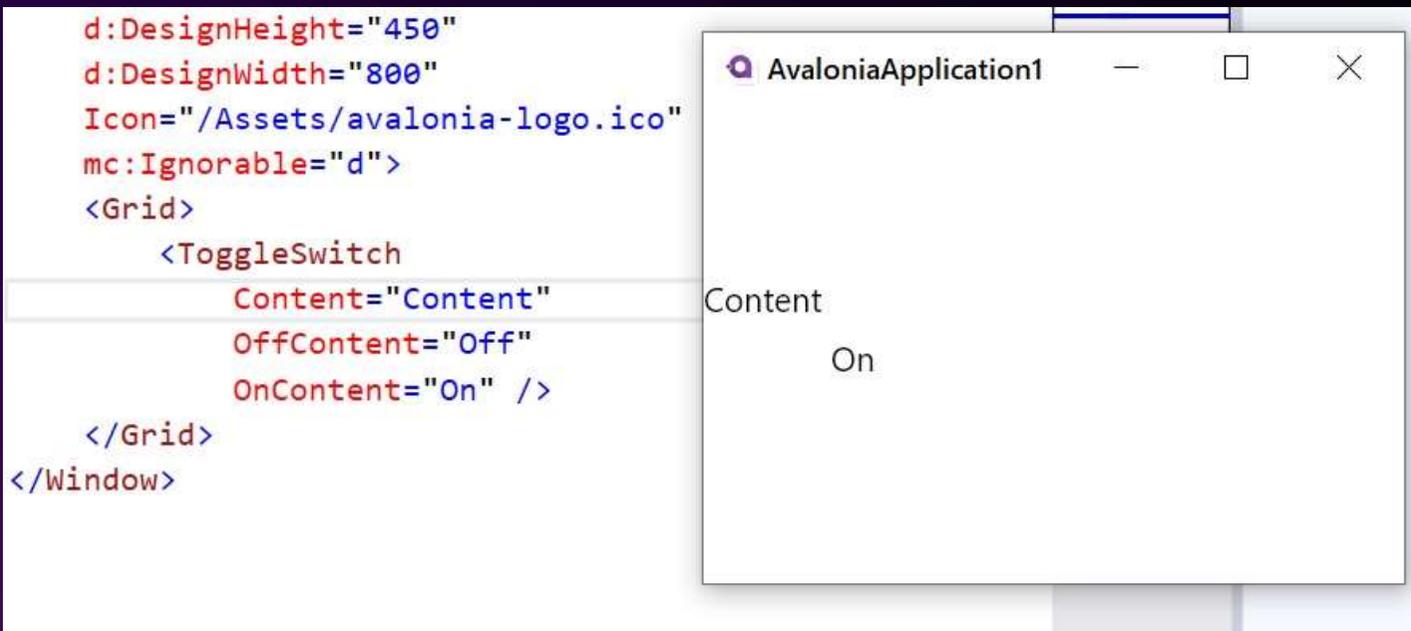
Размер пакета (p):

ПС каналов связи: байт



```
graph LR; A(( )) --- B(( )); A --- C(( )); A --- D(( )); B --- C; B --- D; C --- D;
```

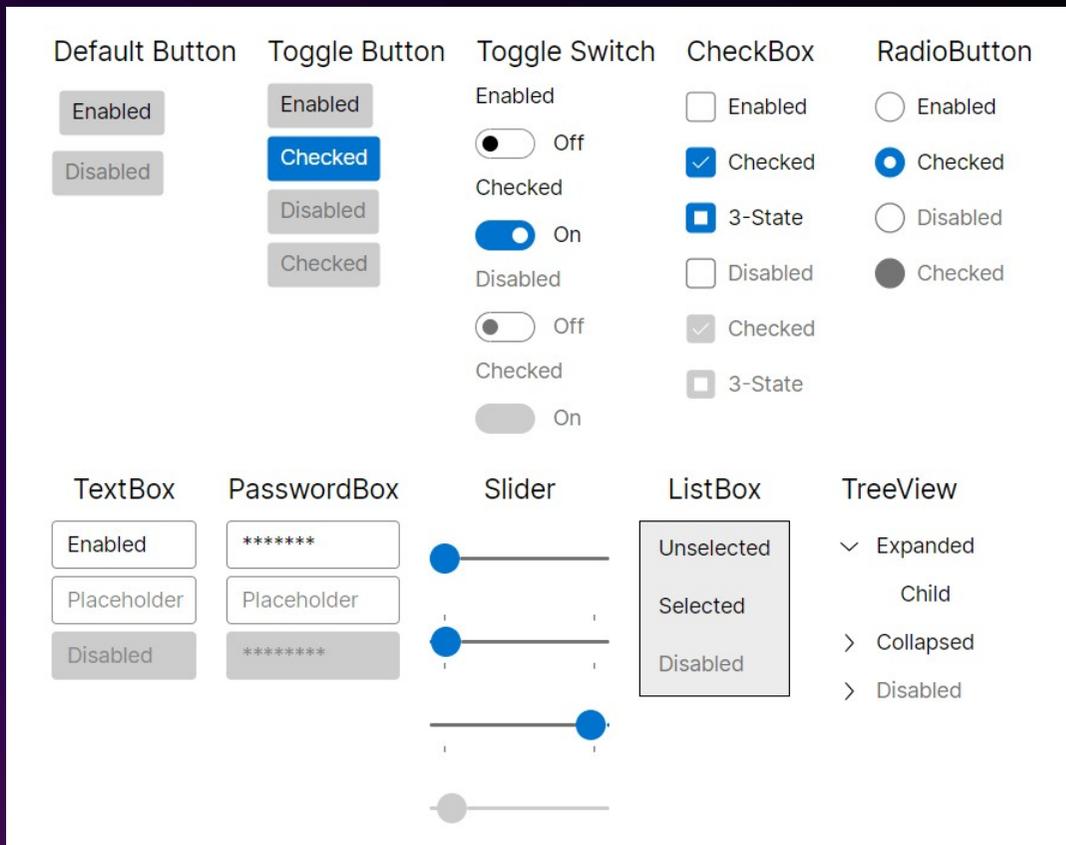
ОСНОВНЫЕ ВОЗМОЖНОСТИ



Использование знакомого XAML-синтаксиса (как в WPF и UWP).

ОСНОВНЫЕ ВОЗМОЖНОСТИ

Fluent

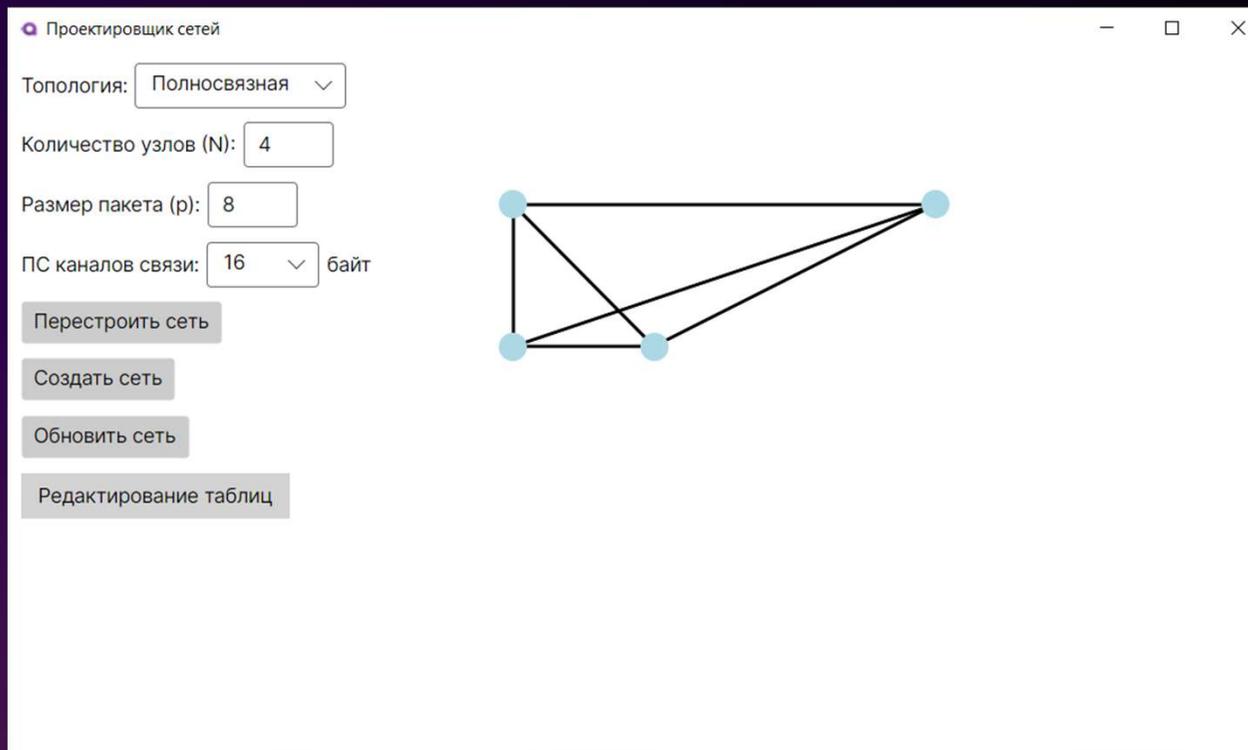


Мощная система стилей и тем (Fluent, Simple, Material).

Данная тема основана на рекомендациях от Microsoft по созданию интерактивного и привлекательного интерфейса.

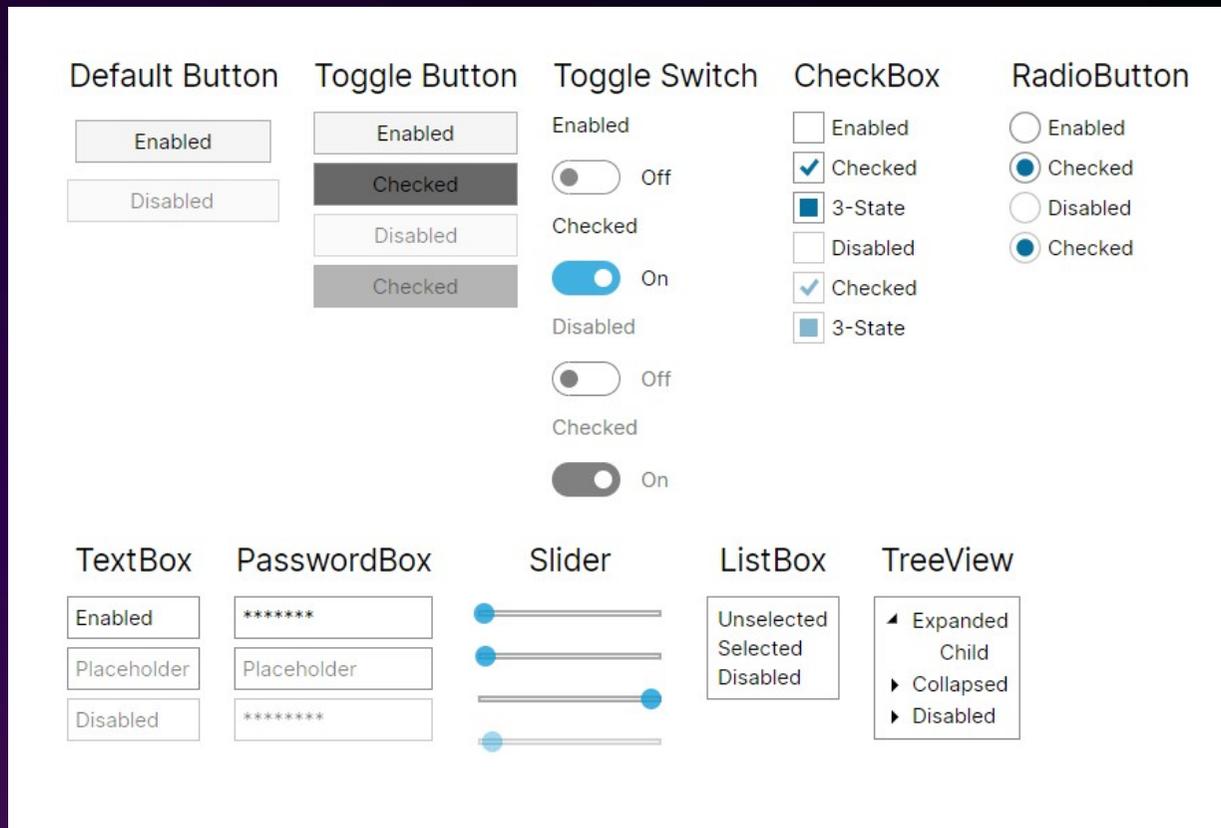
ОСНОВНЫЕ ВОЗМОЖНОСТИ

Fluent на примере приложения:



ОСНОВНЫЕ ВОЗМОЖНОСТИ

Simple

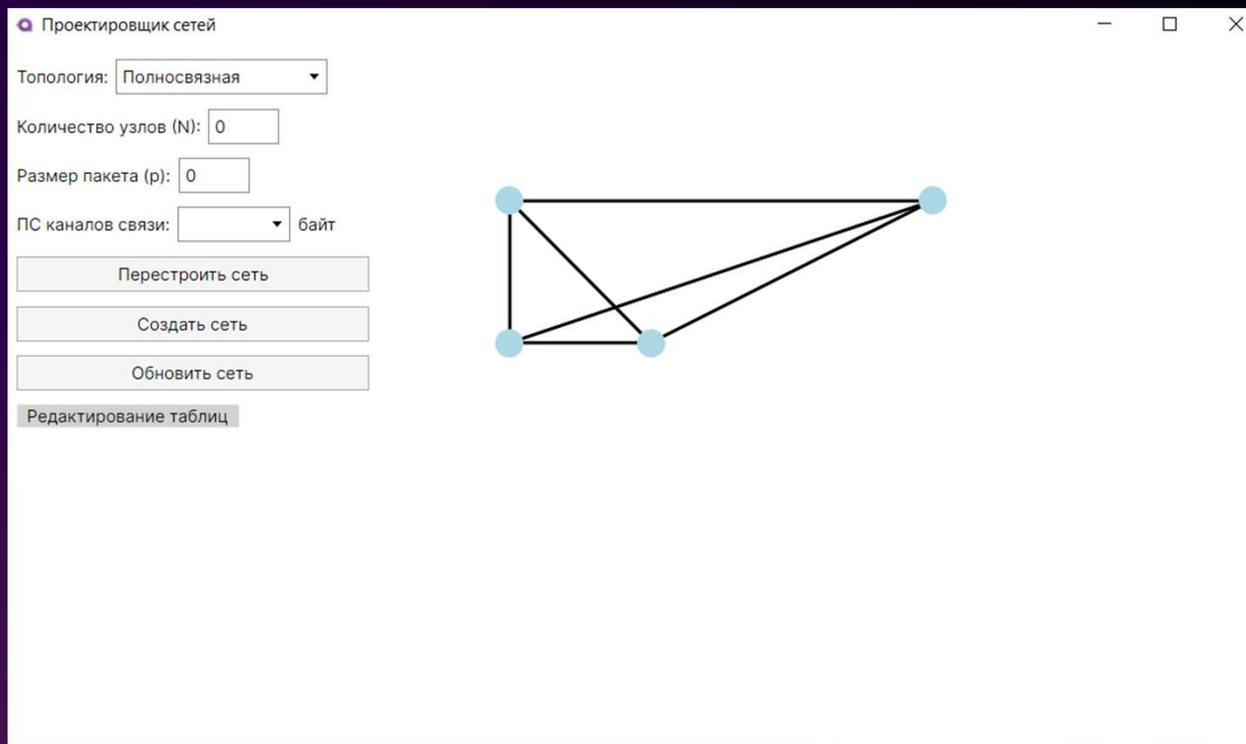


Мощная система стилей и тем (Fluent, Simple, Material).

Это самая простая и минималистичная тема.

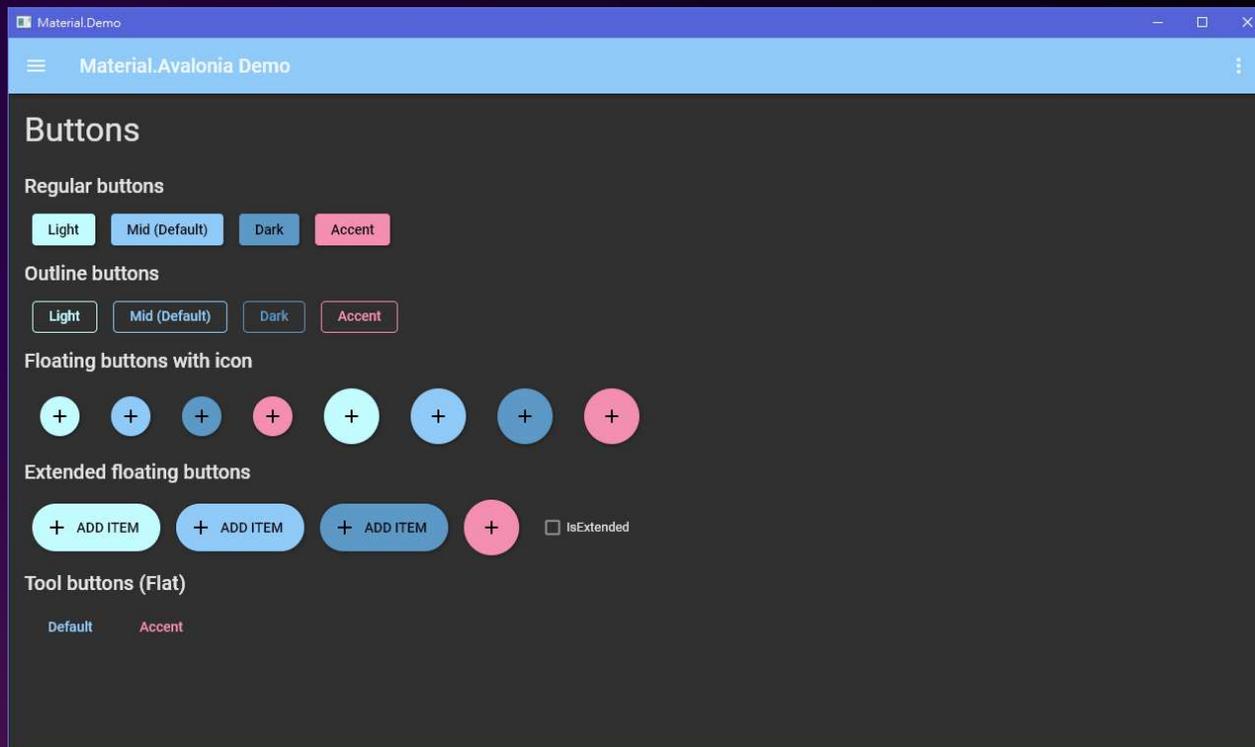
ОСНОВНЫЕ ВОЗМОЖНОСТИ

Simple на примере приложения:



ОСНОВНЫЕ ВОЗМОЖНОСТИ

Material



Мощная система стилей и тем (Fluent, Simple, Material).

Material основан на открытых стилях Гугл для создания удобных и красивых интерфейсов.

ОСНОВНЫЕ ВОЗМОЖНОСТИ

Material на примере приложения:

Проектировщик сетей

Топология: Полносвязная

Количество узлов (N): 4

Размер пакета (p): 8

ПС каналов связи: 16 байт

Перестроить сеть

Создать сеть

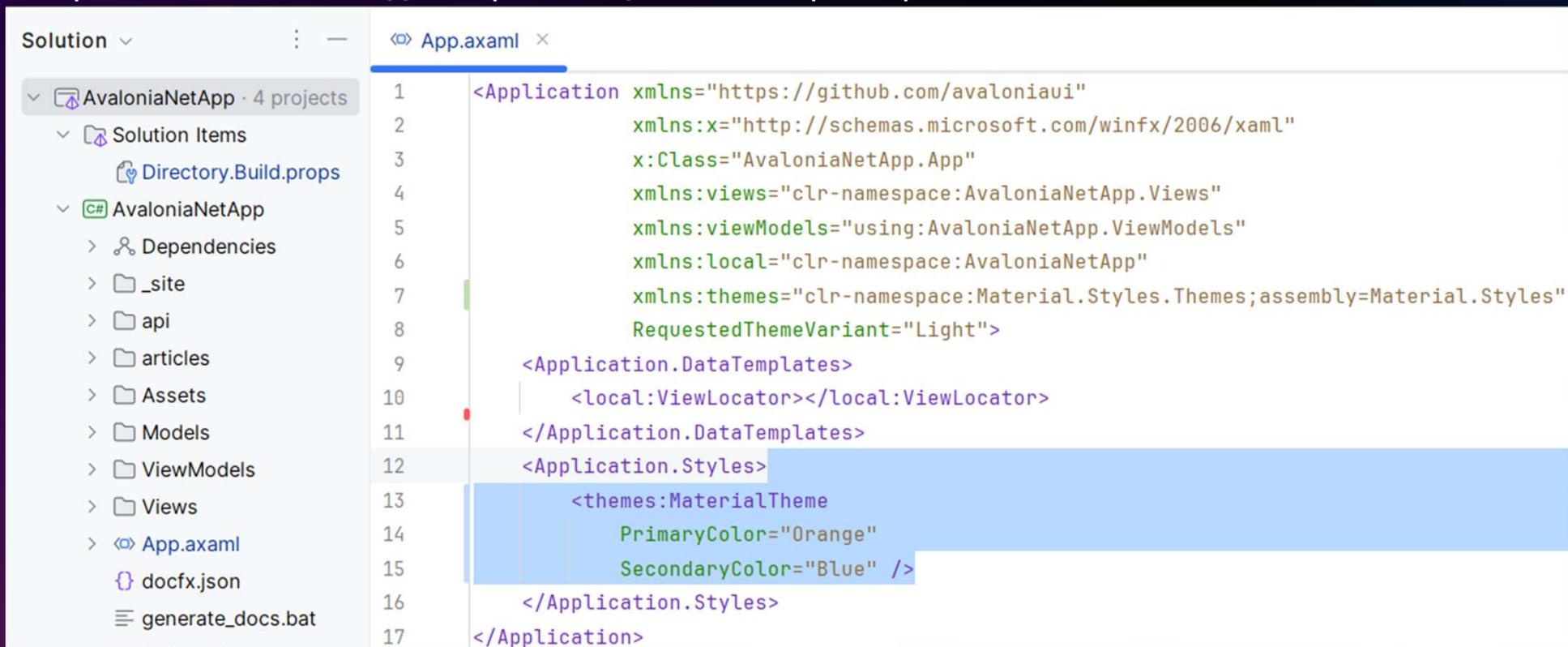
Обновить сеть

Редактирование таблиц

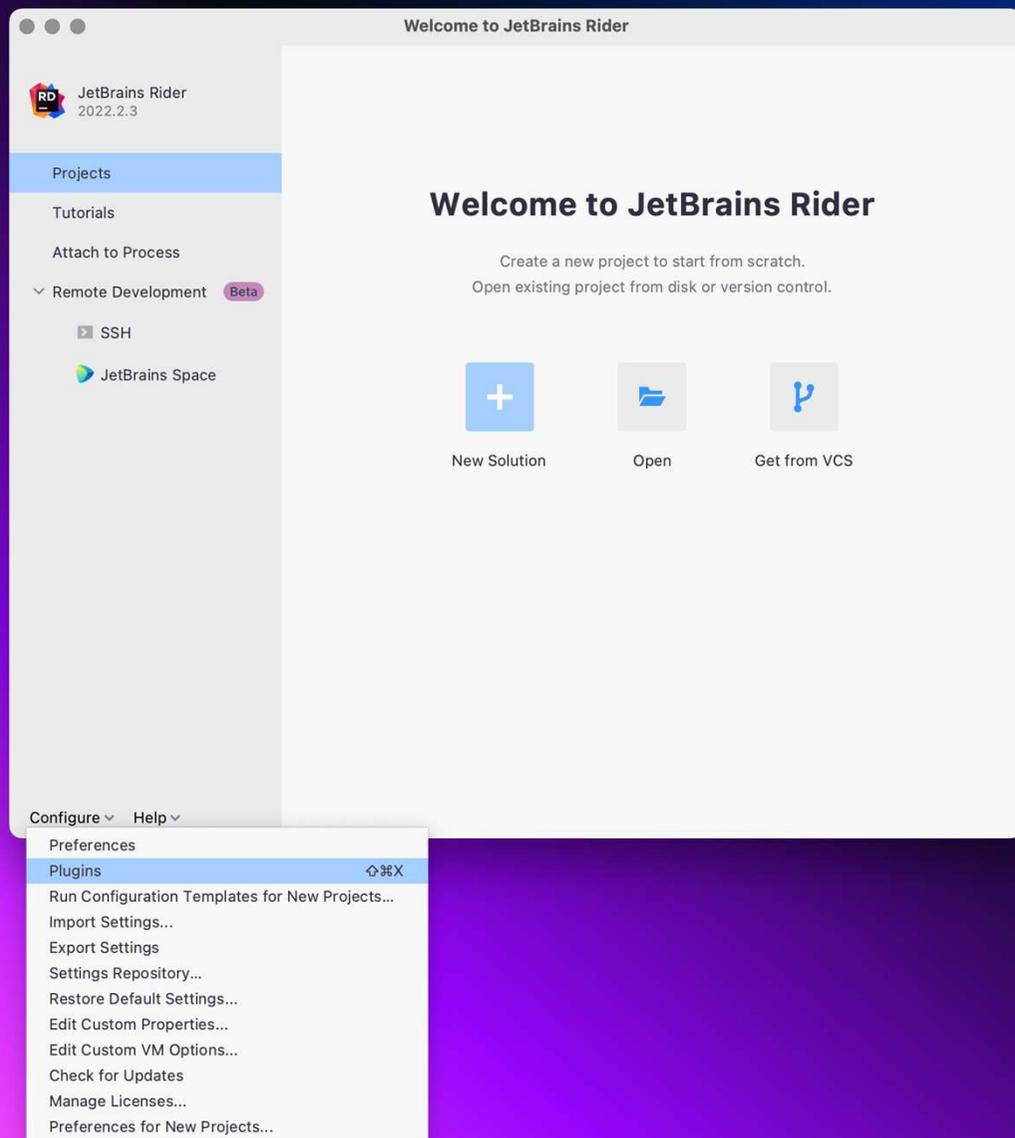
```
graph LR; N1(( )) --- N2(( )); N1 --- N3(( )); N1 --- N4(( )); N2 --- N3; N2 --- N4; N3 --- N4;
```

ОСНОВНЫЕ ВОЗМОЖНОСТИ

Цвет темы Material задан через специальные параметры



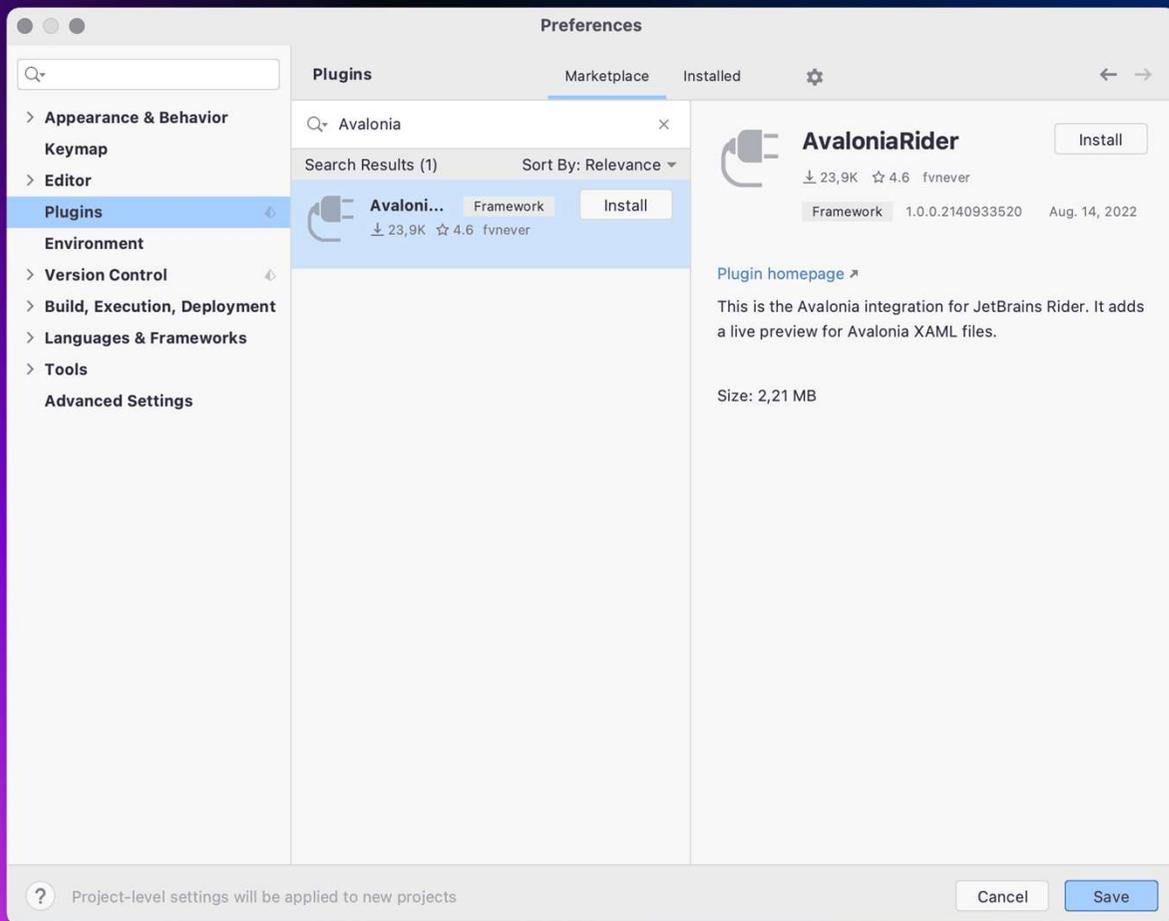
```
1 <Application xmlns="https://github.com/avaloniaui"
2           xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
3           x:Class="AvaloniaNetApp.App"
4           xmlns:views="clr-namespace:AvaloniaNetApp.Views"
5           xmlns:viewModels="using:AvaloniaNetApp.ViewModels"
6           xmlns:local="clr-namespace:AvaloniaNetApp"
7           xmlns:themes="clr-namespace:Material.Styles.Themes;assembly=Material.Styles"
8           RequestedThemeVariant="Light">
9     <Application.DataTemplates>
10      <local:ViewLocator/>
11    </Application.DataTemplates>
12    <Application.Styles>
13      <themes:MaterialTheme
14        PrimaryColor="Orange"
15        SecondaryColor="Blue" />
16    </Application.Styles>
17  </Application>
```



ОСНОВНЫЕ ВОЗМОЖНОСТИ

Интеграция с Rider.

Для создания проекта в данной IDE надо
открыть Конфигурация>Плагины

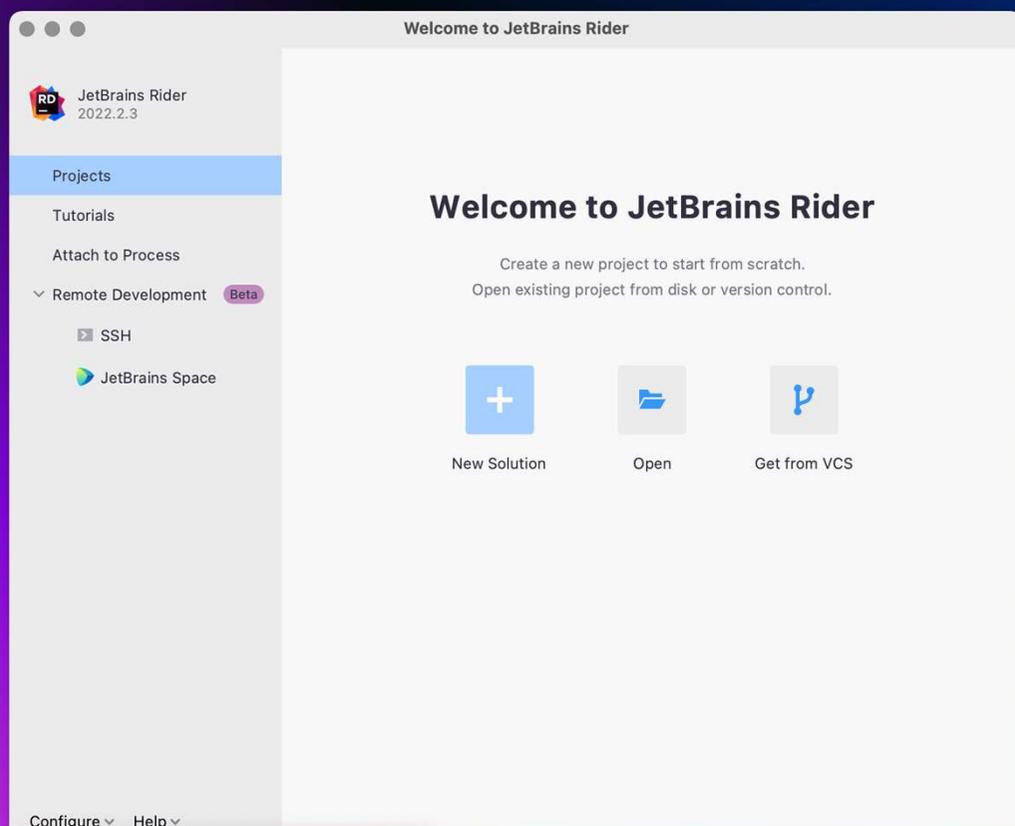


ОСНОВНЫЕ ВОЗМОЖНОСТИ

Интеграция с Rider.

Написать в поиск AvaloniaRider и скачать плагин.

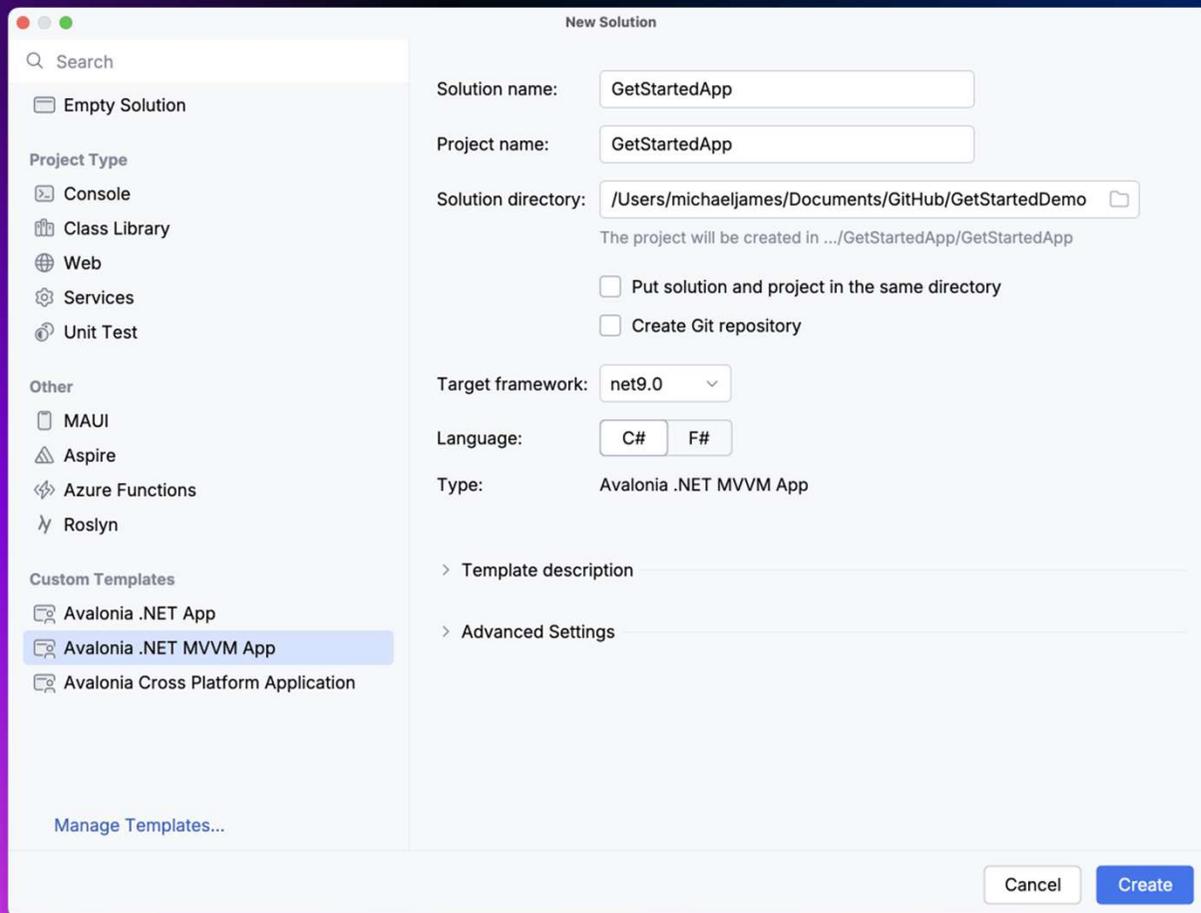
После завершения установки нажмите кнопку «Перезапустить IDE».



ОСНОВНЫЕ ВОЗМОЖНОСТИ

Интеграция с Rider.

После перезагрузки можно нажать на “+” чтобы создать новый проект.



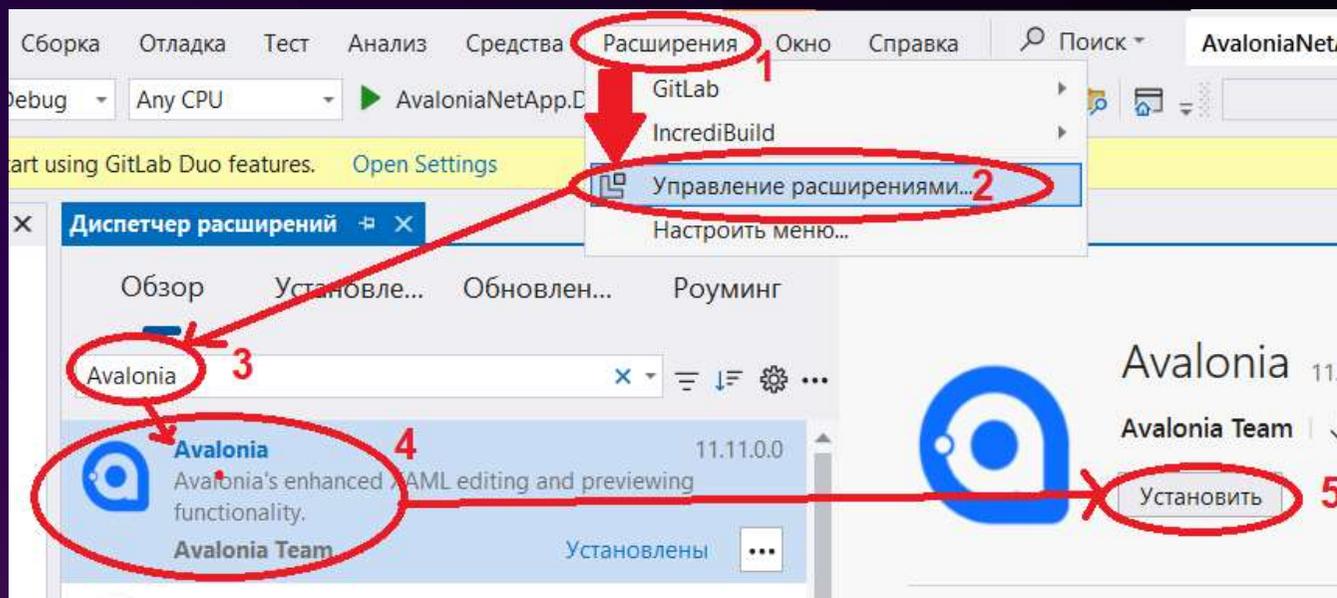
ОСНОВНЫЕ ВОЗМОЖНОСТИ

Интеграция с Rider.

В списке выбрать нужный вариант архитектуры Авалонии в списке слева и создать проект.

(Про Visual Studio аналогично)

ОСНОВНЫЕ ВОЗМОЖНОСТИ

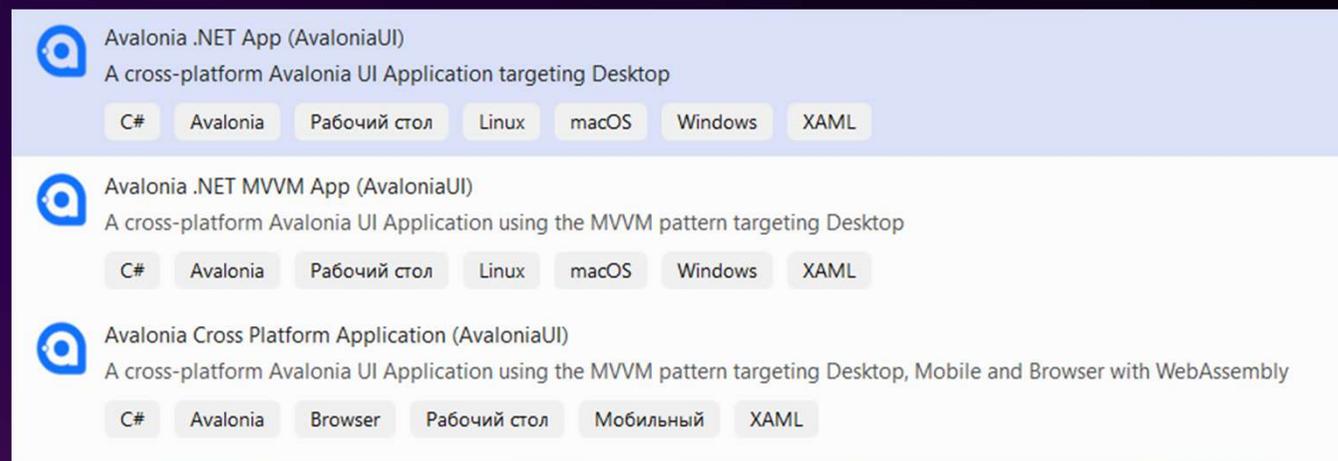


Интеграция с Visual Studio.

Для создания проекта в данной IDE надо:

- 1-2. открыть Расширения>Управление расширениями
3. В поиске ввести "Avalonia"
4. Выбрать первый вариант
5. Установить

ОСНОВНЫЕ ВОЗМОЖНОСТИ



Интеграция с Visual Studio.

Теперь можно просто создать проект Avalonia с одним из трёх шаблонов.

АРХИТЕКТУРА: ШАБЛОНЫ

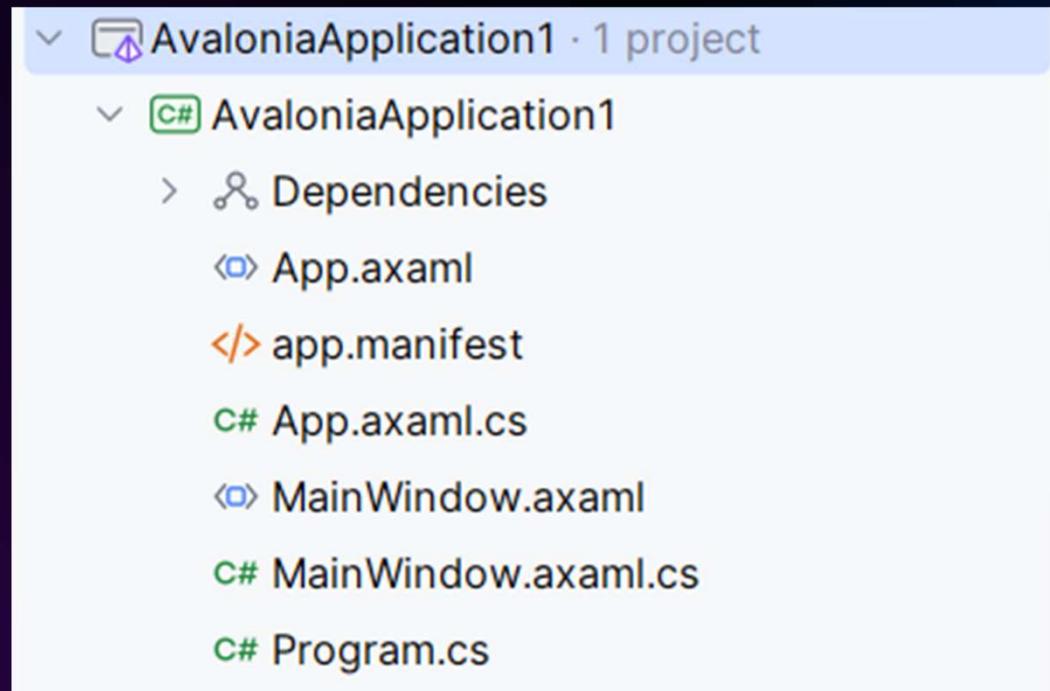
Avalonia .NET App

Минимальный базовый шаблон.

Подходит, когда нужно «чистое» приложение без заранее навязанной архитектуры.

Особенности:

1. Только инфраструктура Avalonia (App.axaml, MainWindow.axaml).
2. Нет MVVM, нет дополнительных библиотек.
3. Создаётся как стартовая точка для ручной сборки архитектуры.
4. Максимальная гибкость — всё пишешь сам.



АРХИТЕКТУРА: ШАБЛОНЫ

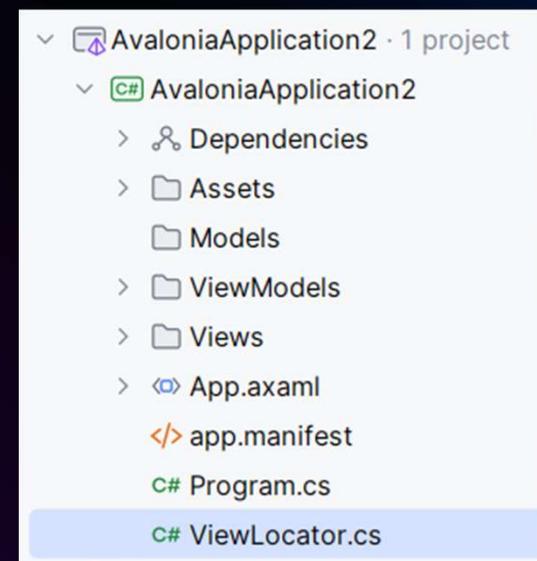
Avalonia .NET MVVM App

Готовый шаблон с архитектурой MVVM.

Подходит для типичных приложений сразу с разделением View / ViewModel.

Особенности:

- Реализован MVVM «из коробки».
- Добавлены базовые ViewModels, привязки, команды.
- Использует **один из двух официальных MVVM-подходов Avalonia** (обычно CommunityToolkit.Mvvm).
- Логика и представление изначально разделены.



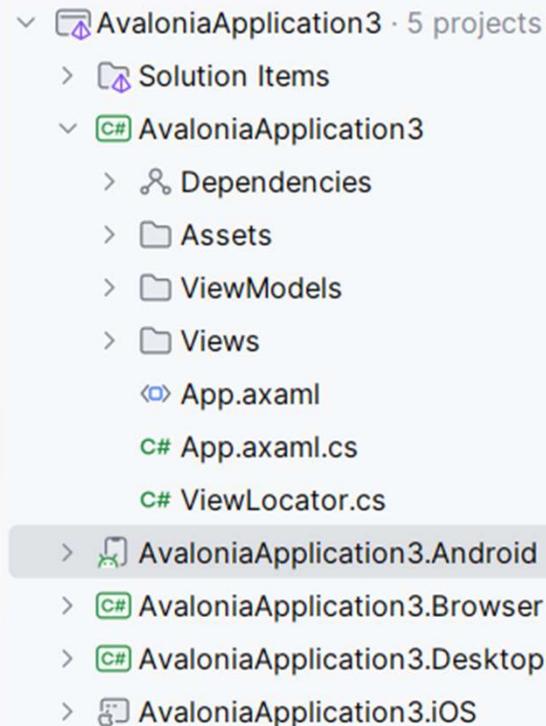
АРХИТЕКТУРА: ШАБЛОНЫ

Avalonia Cross Platform Application

Расширенная версия базового шаблона с фокусом на мультиплатформенности.

Особенности:

1. Более подробная структура проекта под Windows, Linux, macOS, Android, iOS.
2. Уже настроены платформенные стартовые точки (например, AppBuilder для разных ОС).
3. Может содержать заготовки под MVVM (но не всегда включает полноценный MVVM-набросок как MVVM-App).
4. Оптимальный выбор, если цель — выпуск на разные платформы.



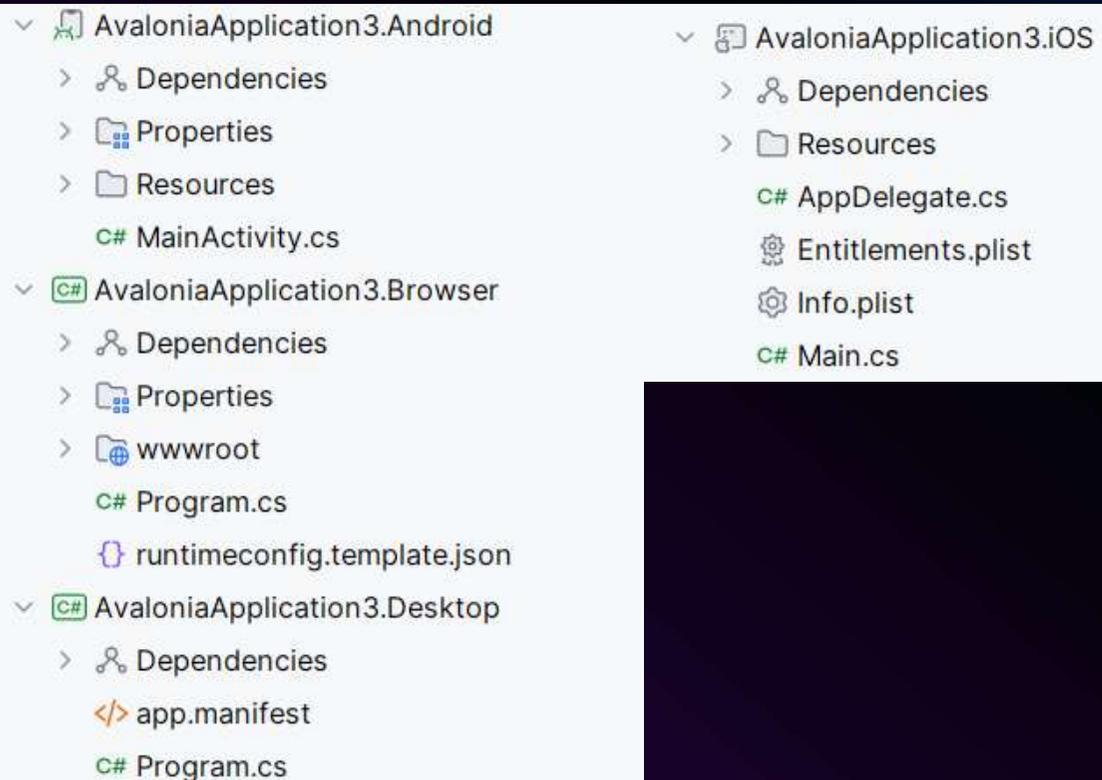
АРХИТЕКТУРА: ШАБЛОНЫ

Avalonia Cross Platform Application

Расширенная версия базового шаблона с фокусом на мультиплатформенности.

Особенности:

1. Более подробная структура проекта под Windows, Linux, macOS, Android, iOS.
2. Уже настроены платформенные стартовые точки (например, AppBuilder для разных ОС).
3. Может содержать заготовки под MVVM (но не всегда включает полноценный MVVM-набросок как MVVM-App).
4. Оптимальный выбор, если цель — выпуск на разные платформы.



АРХИТЕКТУРА: ИНСТРУМЕНТЫ

Инструмент	Простыми словами	Для кого	Сложность
CommunityToolkit.Mvvm	«Пиши как обычно, а мы сделаем за тебя рутину»	Новичков и обычных приложений	Лёгкий
ReactiveUI	«Всё — поток событий, UI сам реагирует»	Сложные, динамичные приложения	Сложнее

АРХИТЕКТУРА: ИНСТРУМЕНТЫ

ReactiveUI

```
using ReactiveUI;
using System.Reactive.Linq;

public class PersonViewModel : ReactiveObject
{
    private int _age;
    public int Age
    {
        get => _age;
        set => this.RaiseAndSetIfChanged(ref _age, value);
    }

    public bool IsAdult => _isAdult.Value;
    private readonly ObservableAsPropertyHelper<bool> _isAdult;

    public PersonViewModel()
    {
        // Реактивный поток: IsAdult обновляется автоматически
        this.WhenAnyValue(x => x.Age)
            .Select(age => age >= 18)
            .ToProperty(this, x => x.IsAdult, out _isAdult);
    }
}
```

CommunityToolkit

```
using CommunityToolkit.Mvvm.ComponentModel;

public partial class PersonViewModel : ObservableObject
{
    [ObservableProperty]
    private int age;

    public bool IsAdult => Age >= 18;
}
```

АРХИТЕКТУРА: MVVM

MVVM — Model–View–ViewModel

Архитектурный паттерн для разделения логики и интерфейса.

View

UI — окна, кнопки, разметка.
Ничего не знает о логике.

ViewModel

Логика: данные, команды, состояния.
Содержит свойства и команды, к которым привязывается View (binding).
Не содержит UI-кода.

Model

Бизнес-данные и работа с ними: структуры данных, сервисы, доступ к БД, API.



АРХИТЕКТУРА: MVVM

View пример: Кнопка “Перестроить сеть” вызывает команду GenerateGraphCommand

The screenshot illustrates the MVVM architecture in Visual Studio. The Solution Explorer on the left shows the project structure, with the **ViewModels** folder circled in red. The code in **MainView.axaml** (line 51) shows a button with the text "Перестроить сеть" and the command `Command="{Binding (MainViewModel). Path=GenerateGraphCommand}"`. The **MainViewModel.cs** file is also visible in the background. On the right, a control panel titled "Проектировщик сетей" (Network Designer) shows a "Перестроить сеть" button circled in red, along with other buttons like "Создать сеть" and "Обновить сеть". A network diagram with 5 nodes (0-4) is shown to the right of the control panel.

АРХИТЕКТУРА: MVVM

ViewModel пример:

GenerateGraphCommand Задаётся в модели представления для доступа из представления.

К команде привязывается метод GenerateGraph() который использует Graph связанный с моделью графа сети.

```
27 public class MainViewModel : ViewModelBase
69 public ICommand GenerateGraphCommand { get; }
70
71
c# MainViewModel.cs x
27 public class MainViewModel : ViewModelBase
200 //ShowMainViewCommand.Subscribe(vm => CurrentView = vm);
201 //ShowGenerateGraphViewCommand.Subscribe(vm => CurrentView = vm);
202 GenerateGraphCommand = ReactiveCommand.Create(GenerateGraph);
203 CreateGraphCommand = ReactiveCommand.Create(CreateGraph);
204
c# MainViewModel.cs x c# MainModel.cs
27 public class MainViewModel : ViewModelBase
555 private void GenerateGraph()
557 //console.WriteLine(">>> команда GenerateGraphCommand вызвана")
558 Graph.Edges.Clear();
559
560 //if (Graph.Nodes.Count == 0) return;
561
562 switch (SelectedTopology)
563 {
564     case "Полносвязная":
565         for (int i = 0; i < Graph.Nodes.Count; i++)
566             for (int j = i + 1; j < Graph.Nodes.Count; j++)
567                 {
568                     Graph.Edges.Add(new Edge { From = Graph.Nodes[i]
```

```
27 public class MainViewModel : ViewModel
48     /// </summary>
49     public GraphModel Graph { get; } =
50     /// </summary>
51     /// </summary>
52     public class Edge : ReactiveObj
102 }
103
104     /// <summary>
105     /// Модель графа, содержащая уз
106     /// </summary>
107     public class GraphModel
108     {
109         /// <summary>
110         /// Коллекция узлов графа.
111         /// </summary>
112         public ObservableCollection
113
114         /// <summary>
115         /// Коллекция рёбер графа.
116         /// </summary>
117         public ObservableCollection
118
119         /// <summary>
120         /// Словарь потоков на рёбр
```

АРХИТЕКТУРА: MVVM

Model пример:

Graph связанный с моделью графа сети GraphModel который описывает модель в виде списка узлов и рёбер.

ПРЕИМУЩЕСТВА И НЕДОСТАТКИ

Недостатки:

- Молодая экосистема по сравнению с WPF, и как следствие недостаток обучающих материалов(нет учебников) и форумов для разработчиков авалония.
- Не все компоненты и SDK пока доступны на мобильных платформах.

Преимущества:

- Кроссплатформенность.
- Привычный XAML и MVVM
- Гибкая стилизация и темы
- Открытый исходный код
- Активное сообщество

Проектировщик сетей

Топология: Полносвязная

Количество узлов (N): 4

Размер пакета (p): 8

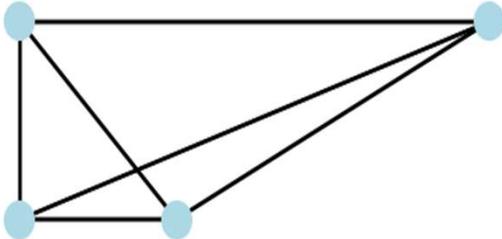
ПС каналов связи: 16 байт

Перестроить сеть

Создать сеть

Обновить сеть

Редактирование таблиц



```
graph LR;
  N1(( )) --- N2(( ))
  N1 --- N3(( ))
  N1 --- N4(( ))
  N2 --- N3
  N2 --- N4
  N3 --- N4
```

ЗАКЛЮЧЕНИЕ

Avalonia UI — перспективный фреймворк для создания кроссплатформенных .NET-приложений. Он объединяет знакомый XAML-подход, гибкий дизайн и поддержку всех ОС. Благодаря активному сообществу и открытому коду, Avalonia быстро развивается и становится серьезной альтернативой WPF и MAUI.

СПАСИБО

Благодарю за внимание!

Источники:

<https://docs.avaloniaui.net/>

<https://github.com/AvaloniaUI/Avalonia.Samples/>

<https://git.kpfu.ru/GAKutasov/avalonianetapp>