Build Better Mobile Apps with ReactiveUI

Andy Lech

Code Mash 2026 Sponsors

Unobtanium Exhibitor

InfernoRed Technology

Adamantium Exhibitor

AWS

Flexjet

TuxCare

Platinum Exhibitor

Umbraco

Codelogic

Jumpmind

Red Hat AI

Text Control, LLC

Gold Exhibitor

Callibrity

CGI

KidzMash Key Sponsor

NimblePros

Non-Exhibiting Sponsors

Boardgame Room

Waterpark Party

Waterpark Party

Waterpark Party

18th Annual Orlando Code Camp | 2026

April 11th - Seminole State College

Seminole State College, Sanford, FL

https://www.orlandocodecamp.com/

Audience Polls

- Written a mobile or desktop app before?
- Used the MVVM pattern in their app?
- Used Reactive Extensions (Rx.NET) before?
- Used ReactiveUI (RxUI) or related libraries before?

Topics

- What is ReactiveUI?
- My Path to ReactiveUI
- Architecting for Smart Data Consumption
- How Reactive UI Helps

Part 1

What is ReactiveUI?

ReactiveUI Ecosystem

- Reactive Extensions (Rx or Rx.NET) - reactivex.io
  An API for asynchronous programming with observable streams
- ReactiveUI (RxUI) - www.reactiveui.net
  A composable Model-View-ViewModel framework for all .NET platforms!
- Dynamic Data - github.com/reactivemarbles/DynamicData
  A library for managing complex operation on collections through Rx.NET
- Refit - reactiveui.github.io/refit
  An automatic type-safe REST library for .NET Core, Xamarin and .NET
- Akavache - github.com/reactiveui/Akavache
  An asynchronous Key-Value store for cross-platform applications
- Splat - github.com/reactiveui/splat
  A cross-platform library for service location, unit testing, and logging

Reactive Extensions

- Based on IObservable<T>
- Data is pushed to subscribers
- Subscribers hook into a pipeline
- Values can be processed like LINQ

```
using System.Reactive.Linq;

IObservable<long> ticks =
  Observable.Timer(
    dueTime: TimeSpan.Zero(),
    period: TimeSpan.FromSeconds(1)
  );

ticks.Subscribe(
  tick => Console.WriteLine("$Tick {tick}");
)

Console.ReadLine();
```

ReactiveUI

- Light MVVM framework over Rx.NET
- Smart property-change notifications
- Control of thread scheduling
- Complex command functionality
- Supports ViewModel navigation
- Allow automatic data persistence

```
this.WhenAnyValue(x => x.SearchQuery)
    .Throttle(TimeSpan.FromSeconds(0.8),
      RxApp.TaskpoolScheduler)
    .Select(query => query?.Trim())
    .DistinctUntilChanged()
    .Where(query =>
      !string.IsNullOrWhiteSpace(query))
    .ObserveOn(RxApp.MainThreadScheduler)
    .InvokeCommand(ExecuteSearch);
```

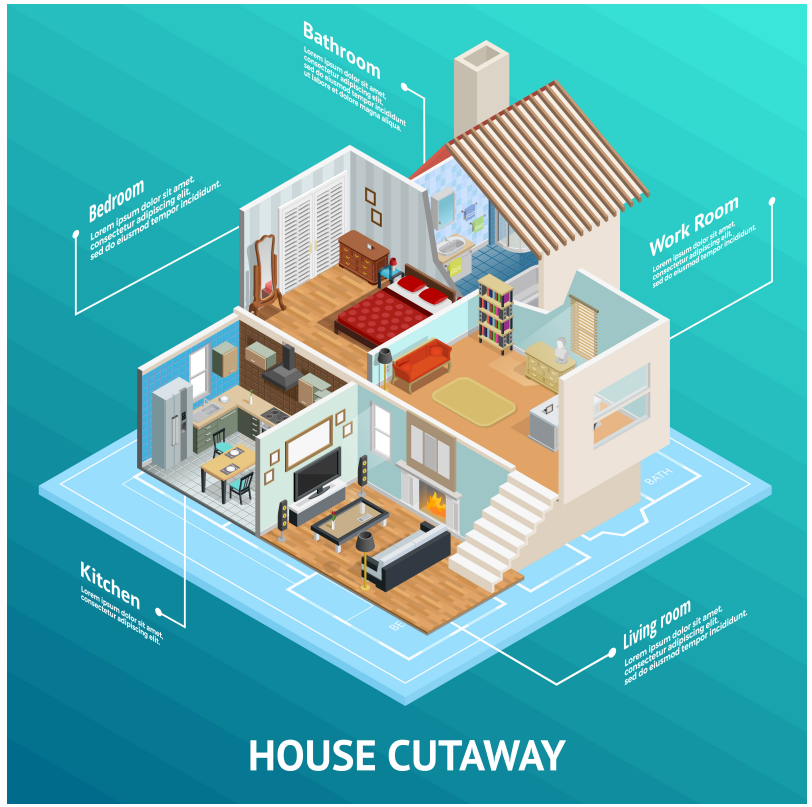
My Path to ReactiveUI

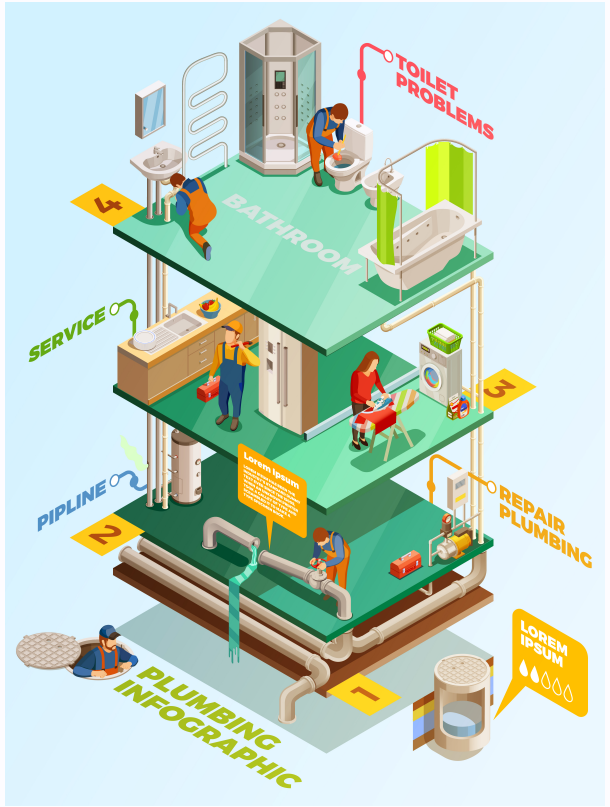
- 2009 - Reactive Extensions released by Microsoft
- Early 2011 - ReactiveUI created
- May 2014 - Xamarin.Forms launched
- August 2015 - Hired to create apps for Android, iOS, Windows Phone
- Late 2015 - Started using Refit for API mapping and Akavache for caching
- Early 2016 - Started using VM navigation library derived from ReactiveUI
- May 2020 - .NET MAUI announced
- September 2020 - Saw Michael Stonis talk on ReactiveUI
- October 2020 - Started using RxUI for VM navigation, testing, and more
- May 2022 - .NET MAUI released

Part 2

Architecting for Smart Data Consumption

Perspectives on Data and Testing



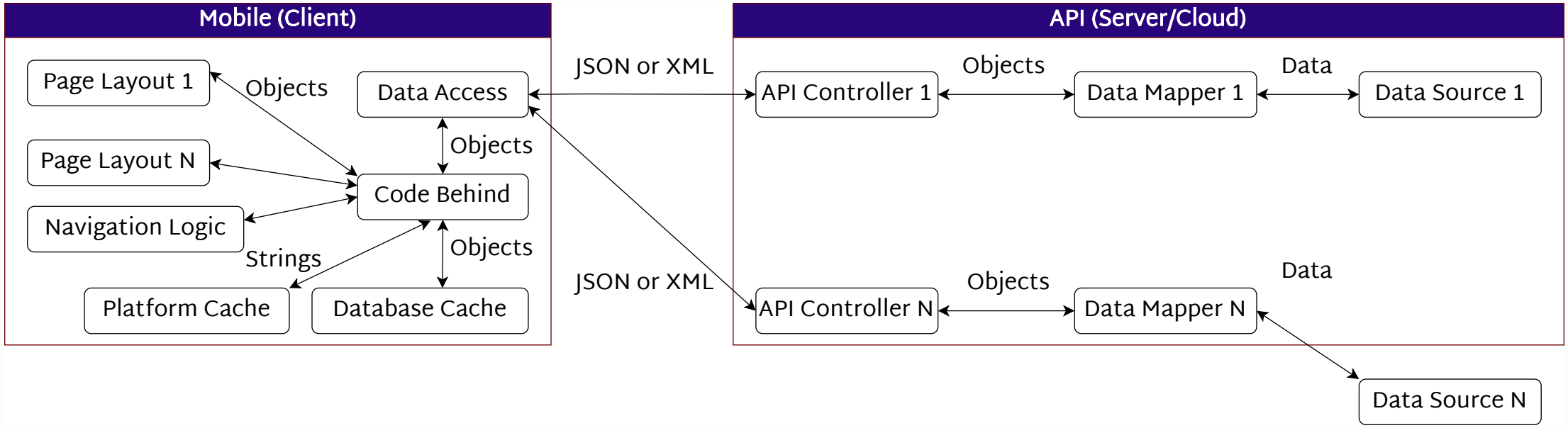


User

Web Site

Mobile App

Mobile Architecture - Components



- App handles interactivity, page layouts, local caching, and API calls
- API stack delivers only data or status codes in response to app requests
- App pages tend to be focused on single tasks that call the API selectively
- **Mobile/API devs focus more on reliable, just-in-time data delivery**

MVVM Pattern - Theory

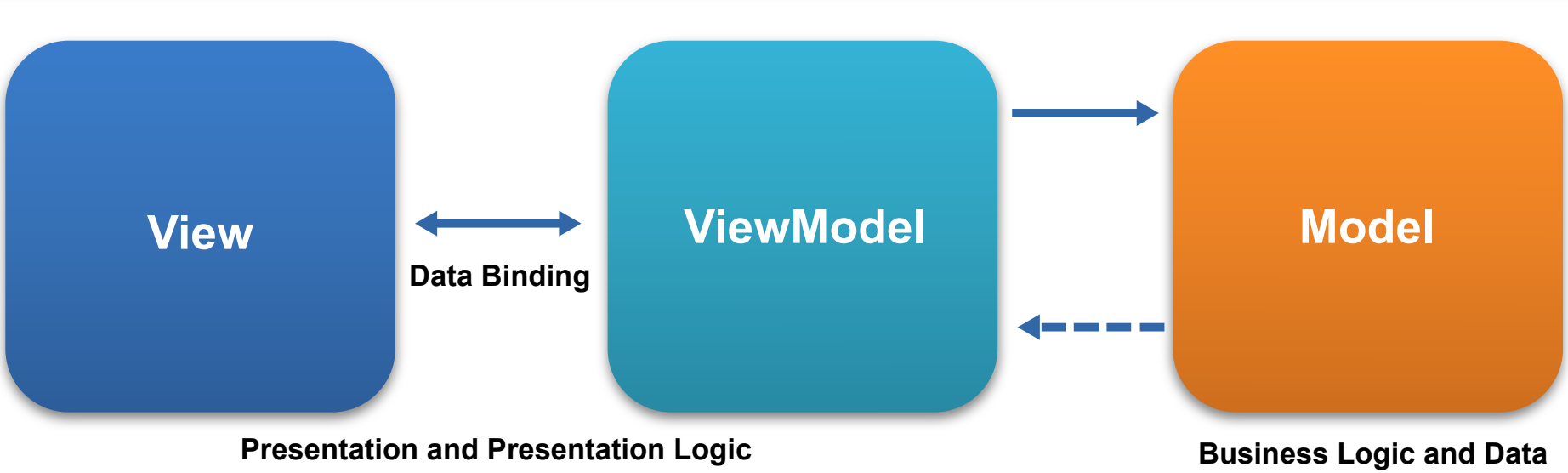
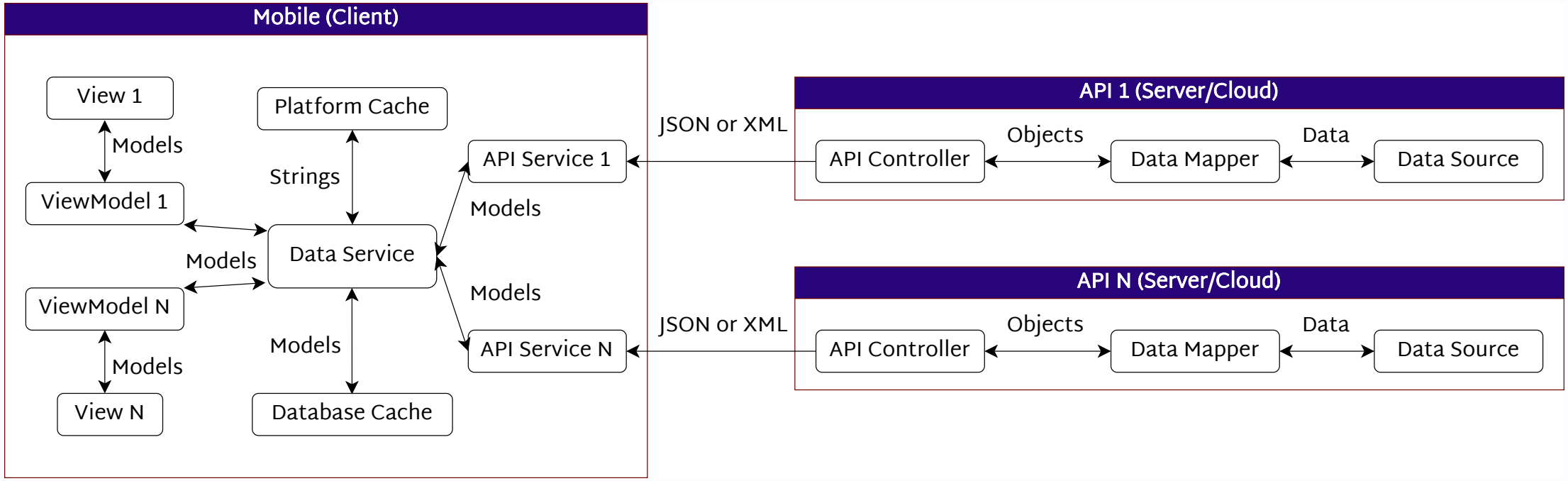


Image from Wikipedia

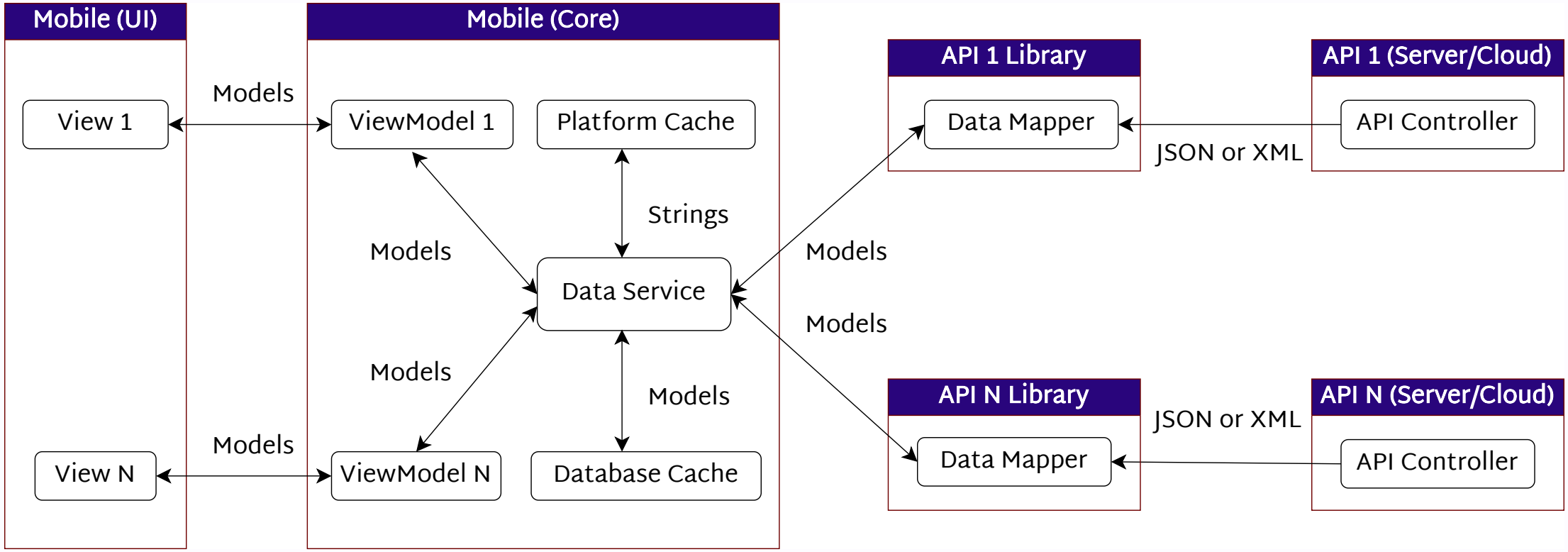
Mobile Architecture - Real-world

Component	Architecture	Purpose
Page Layouts	Views	Presentation
Navigation Logic	ViewModels	Navigation Router
Code Behind	ViewModels	Presentation Logic
Code Behind	ViewModels	Data Manipulation Logic
Local Caches	Data Service	Data Source Logic
Data Access	API Services	Data Translation Logic
Data Access	Models	Internal Data Definitions
Data Access	Data Transfer Objects	External Data Definitions

Mobile Architecture - Network



Mobile Architecture - Solution



Part 3

How Reactive UI Helps

Login - XAML

```
<VerticalStackLayout Grid.Row="1" Spacing="6">
  <Label Text="First Name" TextColor="{StaticResource PrimaryDarkText}" />
  <Entry x:Name="FirstName"
    TextChanged="Entry_OnTextChanged"
    TextColor="{StaticResource Primary}" />
</VerticalStackLayout>
<VerticalStackLayout Grid.Row="2" Spacing="6">
  <Label Text="Last Name" TextColor="{StaticResource PrimaryDarkText}" />
  <Entry x:Name="LastName"
    TextChanged="Entry_OnTextChanged"
    TextColor="{StaticResource Primary}" />
</VerticalStackLayout>
<Button x:Name="LoginButton"
  BackgroundColor="{StaticResource Tertiary}"
  Clicked="LoginButton_OnClicked"
  IsEnabled="False"
  Text="Login"
  TextColor="{StaticResource White}" />
```

.NET - Event Handlers

```
private void Entry_OnTextChanged(object? sender, TextChangedEventArgs e)
{
    LoginButton.IsEnabled =
        !IsNullOrWhiteSpace(FirstName.Text)
        && !IsNullOrWhiteSpace(LastName.Text);
}

private async void LoginButton_OnClicked(object? sender, EventArgs e)
{
    var firstName = FirstName.Text;
    if (Users.TryGetValue(firstName, out var lastName))
    {
        if (lastName == LastName.Text)
        {
            UnauthenticatedMessage.IsVisible = false;
            var homePageRoute = $"home?firstName={firstName}&lastName={lastName}";
            await Shell.Current.GoToAsync(homePageRoute, animate: true);
        }
    }
    UnauthenticatedMessage.IsVisible = true;
}
```

ReactiveUI - Property Assignment

```
public class ViewModel : ReactiveObject {
    [Reactive] public string FirstName { get; set; }
    [Reactive] public string LastName { get; set; }

    public string FormattedName { [ObservableAsProperty] get; }

    public BetterViewModel()
    {
      this.WhenAnyValue(
          vm => vm.FirstName,
          vm => vm.LastName,
          (first, last) => $"{last}, {first}")
        .ToPropertyEx(this, x => x.FormattedName);
    }
  }
```

ReactiveUI - Commands

```
ReactiveCommand
  .CreateFromTask(
    async () =>
    {
      this.Log().Info("Starting Login");
      var result = await ProcessLogin();
      this.Log().Info($"Finished Login");
      return result;
    },
    this.WhenAnyValue(x => x.Username, x => x.Password,
      (username, password) =>
        !string.IsNullOrWhiteSpace(username) &&
        !string.IsNullOrWhiteSpace(password)));
```

Dynamic Data

```
ReadOnlyObservableCollection<TradeProxy> list;

var myTradeCache = new SourceCache<Trade, long>(trade => trade.Id);
var myOperation = myTradeCache.Connect()
  .Filter(trade=>trade.Status == TradeStatus.Live)
  .Transform(trade => new TradeProxy(trade))
  .Sort(SortExpressionComparer<TradeProxy>.Descending(t => t.Timestamp))
  .ObserveOnDispatcher()
  .Bind(out list)
  .DisposeMany()
  .Subscribe();
```

Refit

```
public interface IGitHubApi
{
    [Get("/users/{user}")]
    Task<User> GetUser(string user,
      [Authorize("Bearer")] string token);
}

var gitHubApi = RestService.For<IGitHubApi>("https://api.github.com");

// Will add the header "Authorization: Bearer OAUTH-TOKEN}" to the request
var octocat = await gitHubApi.GetUser("octocat", "OAUTH-TOKEN");
```

Akavache

```
// Store an object
var user = new User { Name = "John", Email = "john@example.com" };
await CacheDatabase.UserAccount.InsertObject("current_user", user);

// Retrieve an object
var cachedUser =
  await CacheDatabase.UserAccount.GetObject<User>("current_user");

// Store with expiration
await CacheDatabase.LocalMachine.InsertObject("temp_data",
  someData, DateTimeOffset.Now.AddHours(1));

// Get or fetch pattern
var data =
  await CacheDatabase.LocalMachine.GetOrFetchObject("api_data",
    async () => await httpClient.GetFromJsonAsync<ApiResponse>(
      "https://api.example.com/data"));
```

Splat

- Service Locator

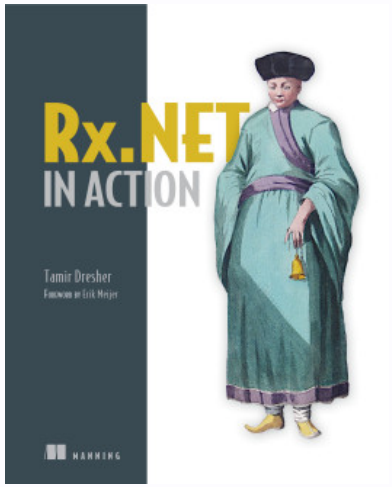
```
// Register services at application startup
Locator.CurrentMutable.Register<IToaster>(() => new Toaster());
Locator.CurrentMutable.RegisterConstant<IConfiguration>(myConfig);
Locator.CurrentMutable.RegisterLazySingleton<ILogger>(() => new FileLogger());

// Resolve services anywhere in your application
var toaster = Locator.Current.GetService<IToaster>();
var config = Locator.Current.GetService<IConfiguration>();
```

References

Media

- Rx.NET in Action by Tamir Dresher (Manning)
- You, I, and ReactiveUI by Kent Boogart (self-published)
- Michael Stonis - Rx, RxUI, and Xamarin.Forms (2 PDFs)
  github.com/TheEightBot/Reactive-Examples



Frameworks

- Reactive Extensions (Rx) - reactivex.io
- ReactiveUI (RxUI) - www.reactiveui.net
- Refit - reactiveui.github.io/refit
- Akavache - github.com/reactiveui/Akavache
- Splat - github.com/reactiveui/splat
- Dynamic Data - github.com/reactivemarbles/DynamicData