Phương pháp tách biệt View và Logic

Ý nghĩa của việc tách việt View và Logic.

Nhìn chung, giao diện người dùng (UI) của game thường rất phức tạp với nhiều yếu tố và trạng thái quan trọng. Hơn nữa, chúng còn sử dụng nhiều animation, đòi hỏi sự tinh chỉnh tỉ mỉ và thường xuyên phải làm lại do thay đổi yêu cầu.

Trong bối cảnh đó, để xây dựng UI một cách hiệu quả và dễ bảo trì, chúng ta cần phải xây dựng một workflow cho phép kiểm tra ngay lập tức các UI đã tạo, chỉnh sửa rồi lại kiểm tra.

Áp dụng điều này vào bối cảnh chuyển màn hình "out-game" (các phần ngoài gameplay chính như menu, shop...) điều đó có nghĩa là chúng ta cần xây dựng một môi trường cho phép dùng dữ liệu giả (dummy data) để kiểm tra hoạt động ngay lập tức ở cấp độ từng màn hình mà không cần phải chạy ứng dụng từ đầu.

Để làm được điều đó, cần phải tách biệt View khỏi Logic và xây dựng một nền tảng cho phép truyền dummy data vào View.

Vì vậy trong bài viết này, Haruki Yano giới thiệu về phương pháp tách biệt View và Logic bằng cách sử dụng USN

Mô tả màn hình sẽ tạo trong bài viết này.

Trong bài viết này chúng ta sẽ tạo một màn hình với 3 ảnh được bố trí như sau:

- Ảnh bên trái (FIXED) là hình ảnh luôn được hiển trị ở trang này.
- Ảnh ở giữa (LOCAL) sẽ ngẫu nhiên tải và hiển thị một hình ảnh từ LOCAL 01 đến LOCAL 03 được lưu trên máy
- Ảnh bên phải (REMOTE) sẽ tải và hiển thị ngẫu nhiên một hình ảnh từ REMOTE 01 đến REMOTE 03 được lấy trên mạng về.

f:id:halya_11:20220104210541p:plain
Mỗi hình ảnh là một button, khi nhấn vào sẽ mở ra một modal tương ứng.

f:id:halya_11:20211212233450g:plain

Đầu tiên, viết toàn bộ vào View.

Trước tiên hãy thử triển khai mô tả trên bằng cách viết tất cả vào View.

```
using System.Collections;
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.UI;
using UnityScreenNavigator.Runtime.Core.Modal;
using UnityScreenNavigator.Runtime.Core.Sheet;

namespace ViewLogicSeparation.Scripts
{
    public class MainSheet : Sheet
    {
        [SerializeField] private Button _fixedButton;
        [SerializeField] private Button _localButton;
        [SerializeField] private Button _remoteButton;
        [SerializeField] private RawImage _fixedImage;
        [SerializeField] private RawImage _localImage;
        [SerializeField] private RawImage _remoteImage;

        // Initialization process
        public override IEnumerator Initialize()
        {
            _fixedImage.texture = Resources.Load<Texture>("tex_button_fixed");
            _fixedButton.onClick.AddListener(OnFixedButtonClicked);
            _localButton.onClick.AddListener(OnLocalButtonClicked);
            _remoteButton.onClick.AddListener(OnRemoteButtonClicked);
            yield break;
        }

        // Cleanup process
        public override IEnumerator Cleanup()
        {
            _fixedImage.texture = null;
            _fixedButton.onClick.RemoveListener(OnFixedButtonClicked);
            _localButton.onClick.RemoveListener(OnLocalButtonClicked);
            _remoteButton.onClick.RemoveListener(OnRemoteButtonClicked);
            yield break;
        }

        // Processing before this screen is displayed
        public override IEnumerator WillEnter()
        {
            _localImage.texture = Resources.Load<Texture>($"tex_button_local_{Random.Range(1, 4):D2}");

            // Download and get the texture
            var url = $"http://foo/bar/tex_button_remote_{Random.Range(1, 4):D2}.png";
            var uwr = UnityWebRequestTexture.GetTexture(url);
            yield return uwr.SendWebRequest();
            _remoteImage.texture = ((DownloadHandlerTexture)uwr.downloadHandler).texture;
        }

        // Processing before this screen is hidden
        public override void DidExit()
        {
```

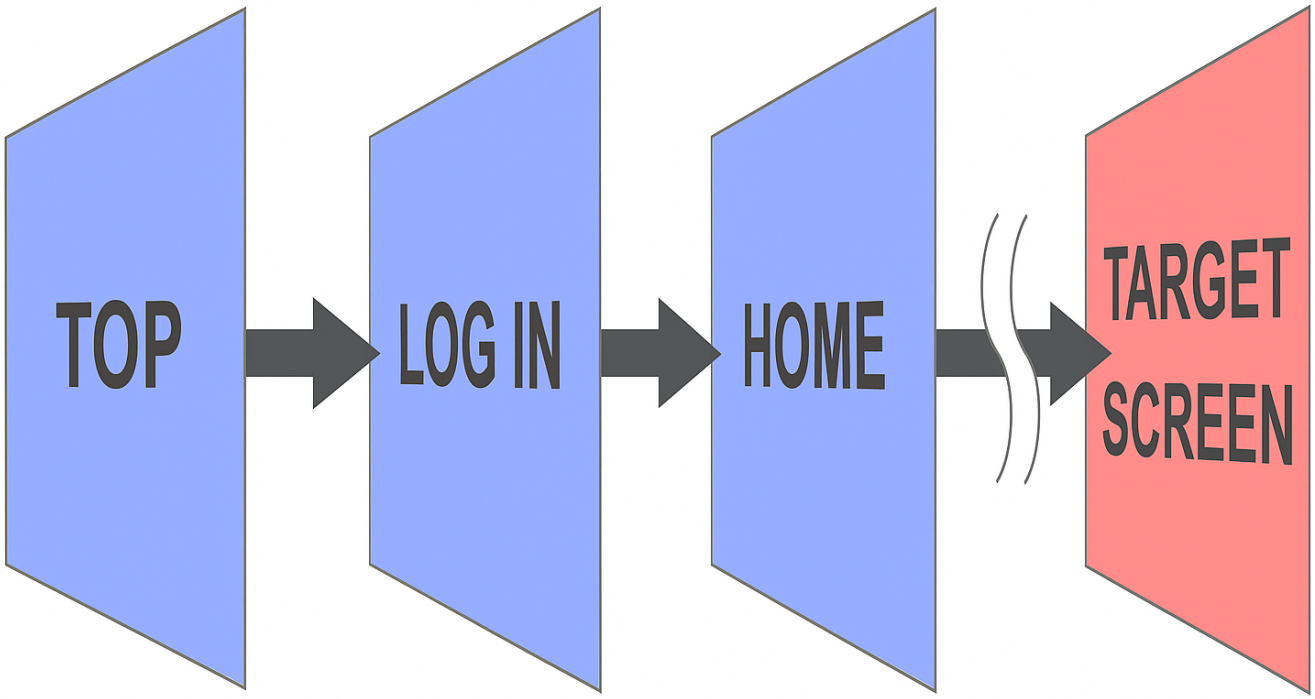
```
            _localImage.texture = null;
            _remoteImage.texture = null;
        }

        private void OnFixedButtonClicked()
        {
            // When the Fixed button is clicked, open Modal01
           ModalContainer.Find("Main").Push("Modal01", true);
        }

        private void OnLocalButtonClicked()
        {
            // When the Local button is clicked, open Modal02
           ModalContainer.Find("Main").Push("Modal02", true);
        }

        private void OnRemoteButtonClicked()
        {
            // When the Remote button is clicked, open Modal03
            ModalContainer.Find("Main").Push("Modal03", true);
        }
    }
}
```

Tạo View và `ModalContainer` một cách phù hợp sau đó gán các tham chiếu của chúng vào các trường `[SerializeField]`

Tiếp theo, chỉ cần dùng `SheetContainer.Show()` để hiển thị sheet này là bạn có thể xác nhận rằng nó hoạt động bình thường

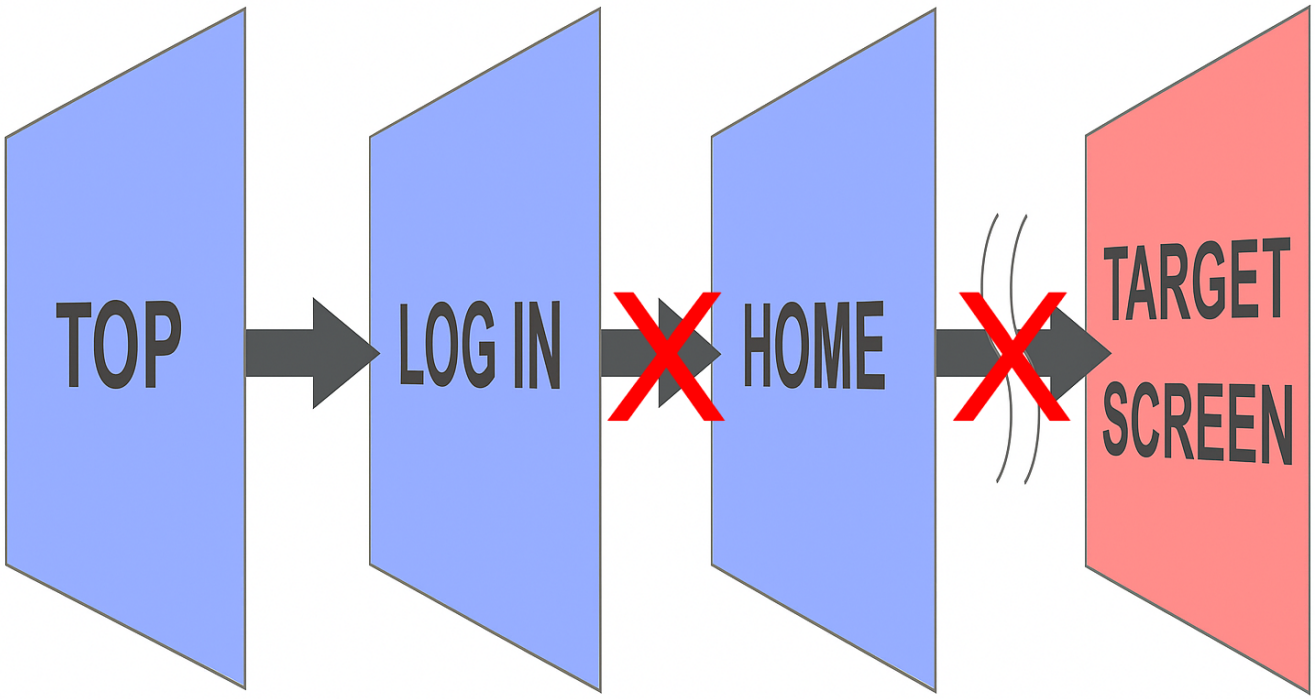
Những vấn đề của cách triển khai trên

Bây giờ, hãy giả sử rằng màn hình chúng ta đã tạo ở phần trước chỉ là một trong số rất nhiều màn hình của một ứng dụng lớn, và đó là một màn hình mà phải đi qua nhiều màn hình mới có thể chuyển tới được.



Khi muốn kiểm tra hoạt động của màn hình này, nếu lần nào cũng phải đi từ màn hình đầu tiên của ứng dụng thì sẽ rất tốn thời gian.

Hơn nữa, ví dụ nếu như màn hình đăng nhập không hoạt động bình thường, chúng ta thậm chí còn không thể đến được màn hình mục tiêu.



Đây là một vấn đề về mặt thiết kế do đã không cân nhắc đến workflow và quy trình phát triển. Có thể tưởng tượng rằng nó sẽ trở thành một món nợ kỹ thuật lớn khi quy mô phát triển ngày càng tăng. Ngay cả khi đã viết unit test hoàn hảo cho logic thì phần view như thế này vẫn là một giải pháp nửa vời.

Để ngăn chặn tình huống này, chúng ta sẽ xây dựng một môi trường cho phép đặt Prefab đã gắn MainScreen ở trên vào một Scene riêng lẻ và chạy nó một cách độc lập.

Tuy nhiên, khi kiểm tra hoạt động một cách độc lập, cách triển khai MainScreen hiện tại có những vấn đề sau:

- Không thể kiểm tra hoạt động nếu tất cả LOCAL 1 - 3 không tồn tại
- Không thể kiểm tra hoạt động nếu các modal đích bị lỗi hoặc chưa được triển khai.
- Không thể kiểm tra hoạt động nếu taatsa cả các ảnh được chỉ định bởi URL (REMOTE 1 - 3) không tồn tại.

Điều này có nghĩa là việc kiểm tra đang bị ảnh hưởng bởi môi trường xung quanh. Do đó, trong phần tiếp theo chúng ta sẽ tách View và Logic để việc kiểm tra các hoạt động của màn hình diễn ra một cách độc lập hơn.

Tách biệt View và Logic

Trước tiên, chúng ta sẽ bắt đầu tách phần logic đang cản trở việc kiểm tra hoạt động ra khỏi View.

Để giải quyết vấn đề đã nêu trên, chúng ta sẽ thiết kế sao cho URL và image path của hình ảnh được cung cấp từ bên ngoài, Khi người dùng nhấp chuột, nó chỉ thông báo một sự kiện ra bên ngoài.

```
using System.Collections;
using UnityEngine;
using UnityEngine.Events;
using UnityEngine.Networking;
using UnityEngine.UI;
using UnityScreenNavigator.Runtime.Core.Sheet;

namespace ViewLogicSeparation.Scripts
{
    public class MainSheet : Sheet
    {
        [SerializeField] private Button _fixedButton;
        [SerializeField] private Button _localButton;
        [SerializeField] private Button _remoteButton;
        [SerializeField] private RawImage _fixedImage;
        [SerializeField] private RawImage _localImage;
        [SerializeField] private RawImage _remoteImage;

        public event UnityAction FixedButtonClicked;
        public event UnityAction LocalButtonClicked;
        public event UnityAction RemoteButtonClicked;

        private string _localImagePath;
        private string _remoteImageUrl;

        // Set the path for the LOCAL image and the URL for the REMOTE image from the outside.
        public void Setup(string localImagePath, string remoteImageUrl)
        {
            _localImagePath = localImagePath;
            _remoteImageUrl = remoteImageUrl;
        }

        public override IEnumerator WillEnter()
        {
            _localImage.texture = Resources.Load<Texture>(_localImagePath);

            // Download and get the texture
            var uwr = UnityWebRequestTexture.GetTexture(_remoteImageUrl);
            yield return uwr.SendWebRequest();
            _remoteImage.texture = ((DownloadHandlerTexture)uwr.downloadHandler).texture;
        }

        // Initialization process
        public override IEnumerator Initialize()
        {
            _fixedImage.texture = Resources.Load<Texture>("tex_button_fixed");
            // Just notify the click event
            _fixedButton.onClick.AddListener(FixedButtonClicked);
            _localButton.onClick.AddListener(LocalButtonClicked);
            _remoteButton.onClick.AddListener(RemoteButtonClicked);
            yield break;
        }

        // Cleanup process
        public override IEnumerator Cleanup()
        {
            _fixedImage.texture = null;
            _fixedButton.onClick.RemoveListener(FixedButtonClicked);
            _localButton.onClick.RemoveListener(LocalButtonClicked);
            _remoteButton.onClick.RemoveListener(RemoteButtonClicked);
```

```
            yield break;
        }

        // Processing before this screen is hidden
        public override void DidExit()
        {
            _localImage.texture = null;
            _remoteImage.texture = null;
        }
    }
}
```

Tiếp theo, chúng ta sẽ tạo ra một Presenter để truyền path và URL vào view này, cũng như để nhận các sự kiện từ View.

Trong chuyển đổi màn hình, các sự kiện vòng đời của mỗi màn hình có thể được xem như một loại sự kiện đến từ View. Presenter sẽ bắt (hook) vào các sự kiện vòng đời của màn hình này để thực hiện các xử lý phù hợp tại mỗi thời điểm.

Trong USN, chúng ta thực hiện điều này bằng cách thêm một `ISheetLifecycleEvent` vào mỗi màn hình thông qua phương thức `AddLifecycleEvent()` như sau.

```
using System;
using System.Collections;
using UnityScreenNavigator.Runtime.Core.Modal;
using UnityScreenNavigator.Runtime.Core.Sheet;
using Random = UnityEngine.Random;

namespace ViewLogicSeparation.Scripts
{
    public class MainSheetPresenter : ISheetLifecycleEvent, IDisposable
    {
        private readonly MainSheet _sheet;

        public MainSheetPresenter(MainSheet sheet)
        {
            _sheet = sheet;
            // Adding this Presenter's lifecycle events to the screen.
            // If a value less than 0 is given as the second argument, it will be executed before the
screen's lifecycle events.
            // If a value of 1 or more is given as the second argument, it will be executed after the
screen's lifecycle events.
            _sheet.AddLifecycleEvent(this, -1);
            _sheet.FixedButtonClicked += OnFixedButtonClicked;
            _sheet.LocalButtonClicked += OnLocalButtonClicked;
            _sheet.RemoteButtonClicked += OnRemoteButtonClicked;
        }

        public void Dispose()
        {
            _sheet.RemoveLifecycleEvent(this);
            _sheet.FixedButtonClicked -= OnFixedButtonClicked;
            _sheet.LocalButtonClicked -= OnLocalButtonClicked;
            _remoteButton.onClick.RemoveListener(OnRemoteButtonClicked);
        }

        public IEnumerator Initialize()
        {
            yield break;
        }
```

```
        public IEnumerator WillEnter()
        {
            // Set the paths and URLs of the images to be used in the Sheet.
            var localImagePath = $"tex_button_local_{Random.Range(1, 4):D2}";
            var remoteImageUrl = $"http://foo/bar/tex_button_remote_{Random.Range(1, 4):D2}.png";
            _sheet.Setup(localImagePath, remoteImageUrl);
            yield break;
        }

        public void DidEnter()
        {
        }

        public IEnumerator WillExit()
        {
            yield break;
        }

        public void DidExit()
        {
        }

        public IEnumerator Cleanup()
        {
            yield break;
        }

        private void OnFixedButtonClicked()
        {
            // When the Fixed button is clicked, open Modal01.
            ModalContainer.Find("Main").Push("Modal01", true);
        }



        private void OnLocalButtonClicked()
        {
            // When the Local button is clicked, open Modal02.
            ModalContainer.Find("Main").Push("Modal02", true);
        }

        private void OnRemoteButtonClicked()
        {
            // When the Remote button is clicked, open Modal03.
            ModalContainer.Find("Main").Push("Modal03", true);
        }
    }
}
```

Sau đó, chỉ cần khởi tại Presenter này trong khi cung cấp cho nó MainSheet là bạn có thể xác nhận rằng nó hoạt động bình thường.

20211212233450.gif
Lưu ý, vì mục đích của chúng ta là tách biệt khỏi View nên tác giả đã không suy nghĩ phức tạp thêm và viết tất cả các phần xử lý vào Presenter

Chạy thử với dữ liệu giả (dummy data)

Bây giờ, khi chúng ta đã có thể tách biệt View và Logic để chạy kiểm thử và kiểm tra màn hình một cách độc lập hãy kiểm tra nó bằng dummy data.

Lần này, chúng ta sẽ cung cấp một vài đường dẫn cố định của ảnh giả, mỗi khi nhấn nút chúng ta sẽ cho hiển thị một modal giả tương ứng.

Dưới đây là một Presenter dùng cho việc thử nghiệm.

```
using System;
using System.Collections;
using UnityScreenNavigator.Runtime.Core.Modal;
using UnityScreenNavigator.Runtime.Core.Sheet;

namespace ViewLogicSeparation.Scripts
{
    public class FakeMainSheetPresenter : ISheetLifecycleEvent, IDisposable
    {
        private readonly MainSheet _sheet;

        public FakeMainSheetPresenter(MainSheet sheet)
        {
            _sheet = sheet;
            _sheet.AddLifecycleEvent(this, -1);
            _sheet.FixedButtonClicked += OnFixedButtonClicked;
            _sheet.LocalButtonClicked += OnLocalButtonClicked;
            _sheet.RemoteButtonClicked += OnRemoteButtonClicked;
        }

        public void Dispose()
        {
            _sheet.RemoveLifecycleEvent(this);
            _sheet.FixedButtonClicked -= OnFixedButtonClicked;
            _sheet.LocalButtonClicked -= OnLocalButtonClicked;
            _sheet.RemoteButtonClicked -= OnRemoteButtonClicked;
        }

        public IEnumerator Initialize()
        {
            yield break;
        }

        public IEnumerator WillEnter()
        {
            // Set the paths and URLs of the images to be used in the Sheet.
            var localImagePath = "tex_button_dummy";
            var remoteImageUrl = $"http://foo/bar/tex_button_dummy.png";
            _sheet.Setup(localImagePath, remoteImageUrl);
            yield break;
        }

        public void DidEnter()
        {
        }

        public IEnumerator WillExit()
        {
            yield break;
        }

        public void DidExit()
        {
        }
```

```
        public IEnumerator Cleanup()
        {
            yield break;
        }

        private void OnFixedButtonClicked()
        {
            OpenFakeModal();
        }

        private void OnLocalButtonClicked()
        {
            OpenFakeModal();
        }

        private void OnRemoteButtonClicked()
        {
            OpenFakeModal();
        }

        private void OpenFakeModal()
        {
            ModalContainer.Find("Main").Push("FakeModal", true);
        }
    }
}
```

Khi khởi tạo Presenter này truyền vào MainSheet (đóng vai trò là View), kết quả thực thi như sau:

f:id:halya_11:20211221225035g:plain
Như vậy, giờ đây chúng ta đã có thể sử dụng dữ liệu giả để kiểm tra hoạt động của màn hình kể cả khi hình ảnh local và remote chưa có sẵn hoặc khi các modal chưa được làm xong.

Lời kết

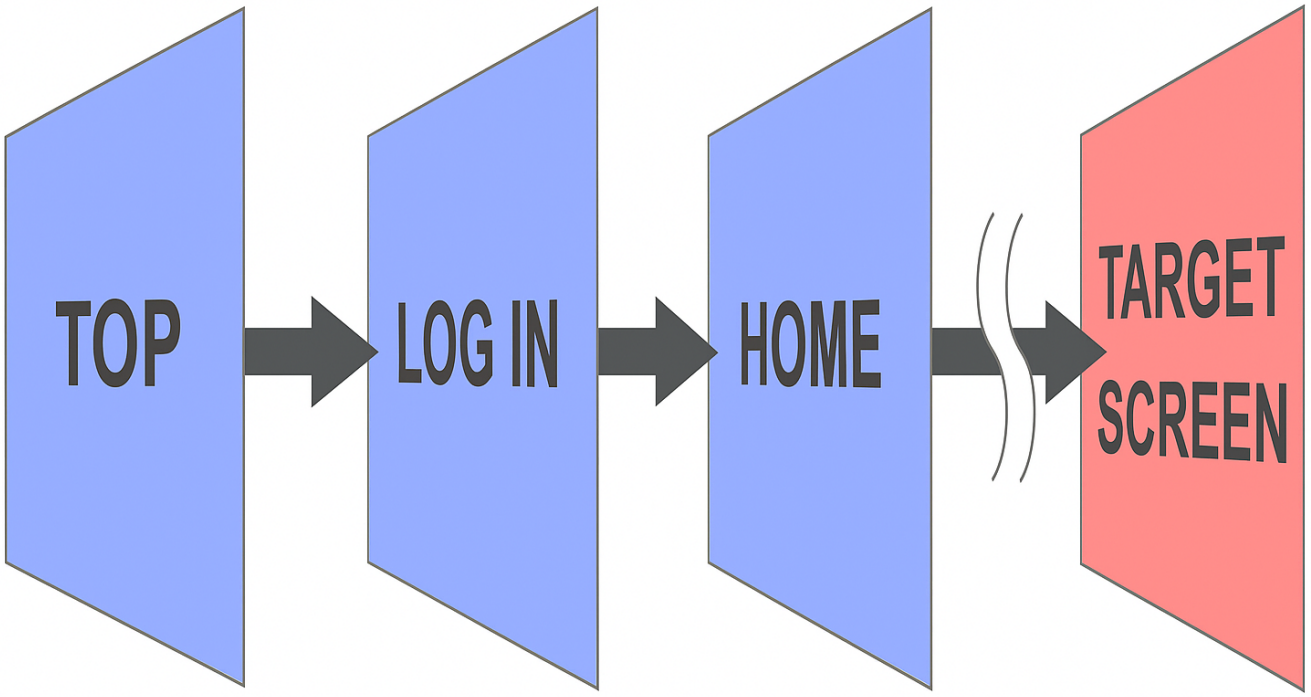
Vậy là Haruki Yano đã tổng hợp về phương pháp tách biệt View và Logic bằng cách sử dụng USN.

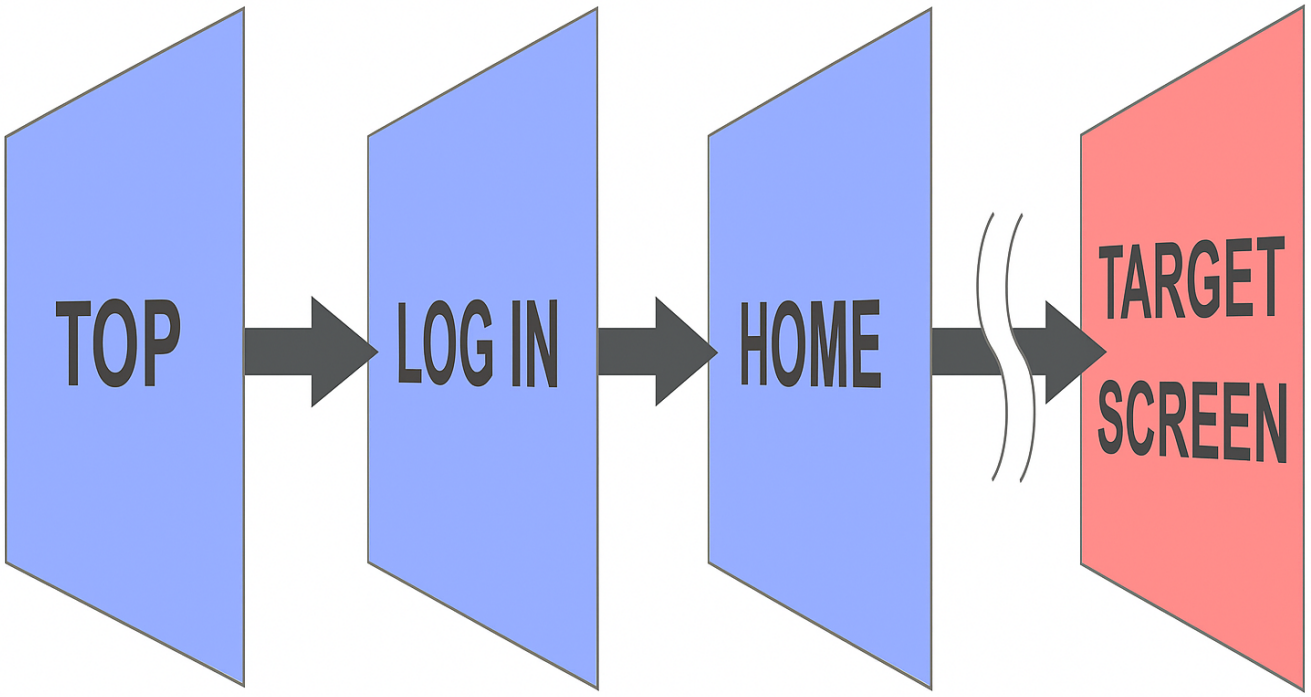
Lưu ý rằng ví dụ trong bài viết chỉ là một trường hợp và không có nghĩa đây là câu trả lời đúng và tối ưu. Ví dụ, trong bài viết này tác giả đã tách các popup thành một View riêng biệt, nhưng tùy vào yêu cầu mà việc tích hợp nó vào View như một phần của màn hình có thể sẽ tốt hơn. Chúng ta cũng có thể xây dựng một môi trường kiểm thử hoạt động từng cụm gồm nhiều màn hình thay vì từng màn hình riêng lẻ.

Ngoài ra, các ứng dụng thực tế còn phức tạp hơn nhiều và việc xây dựng một môi trường hoạt động sử dụng dummy data cũng tốn rất nhiều công sức. Cũng có thể có người cho rằng nên ưu tiên triển khai trước mắt hơn là xây dựng môi trường.

Tuy nhiên nếu *công sức xây dựng môi trường < công sức tiết kiệm được* thì dĩ nhiên chúng ta nên làm. Và Haruki Yano nghĩ rằng chính việc đưa ra phán đoán đó mới là một công việc quan trọng.

Đừng để bị các lý thuyết về kiến trúc chi phối mà rơi vào chủ nghĩa nguyên tắc, hãy nhìn về tương lai và tự mình đưa ra các phán đoán tổng thể.


TOP
LOG IN
HOME
TARGET
SCREEN


TOP
LOG IN
HOME
TARGET
SCREEN

Revision #3
Created 2025-06-23 16:47:44 UTC by ThanhDV
Updated 2025-06-24 17:01:13 UTC by ThanhDV