

ỦY BAN NHÂN DÂN TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC SÀI GÒN

TRẦN ANH TUẤN
NGUYỄN CÔNG MINH

NGHIÊN CỨU VỀ .NET MAUI
VÀ XÂY DỰNG ỨNG DỤNG MINH HOẠ

KHÓA LUẬN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN
TRÌNH ĐỘ ĐÀO TẠO: ĐẠI HỌC

TP. HỒ CHÍ MINH, THÁNG 5 NĂM 2024

ỦY BAN NHÂN DÂN TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC SÀI GÒN

TRẦN ANH TUẤN
NGUYỄN CÔNG MINH

NGHIÊN CỨU VỀ .NET MAUI
VÀ XÂY DỰNG ỨNG DỤNG MINH HOẠ

KHÓA LUẬN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN
TRÌNH ĐỘ ĐÀO TẠO: ĐẠI HỌC

NGƯỜI HƯỚNG DẪN: TS. CAO THÁI PHƯƠNG THANH

TP. HỒ CHÍ MINH, THÁNG 5 NĂM 2024

LỜI CAM ĐOAN

Chúng tôi xin cam đoan đây là công trình nghiên cứu của riêng chúng tôi, các số liệu và kết quả nghiên cứu đều trong luận văn là trung thực, được các đồng tác giả cho phép sử dụng và chưa từng được công bố trong bất kì một công trình nào khác.

Tác giả khóa luận 1

Tác giả khóa luận 2

Trần Anh Tuấn

Nguyễn Công Minh

LỜI CẢM ƠN

Khóa luận tốt nghiệp đóng vai trò quan trọng trong quá trình học tập của sinh viên năm cuối, đòi hỏi sự chú tâm và đầu tư không chỉ về kiến thức mà còn về thời gian và tâm huyết. Việc thực hiện khóa luận là một giai đoạn quan trọng, đóng vai trò tiền đề quan trọng trong việc phát triển kỹ năng nghiên cứu cho sinh viên để sẵn sàng cho con đường phía trước.

Trước hết, chúng em xin bày tỏ lòng biết ơn đặc biệt đến giảng viên hướng dẫn là thầy Cao Thái Phương Thanh, người đã hỗ trợ chúng tôi không ngừng trên hành trình nghiên cứu và thực hiện khóa luận. Những kiến thức mà thầy chia sẻ đã làm nền tảng vững chắc cho quá trình hoàn thành bài luận văn này.

Chúng em cũng muốn bày tỏ lòng biết ơn đặc biệt đến thầy Phạm Thị Vương, người đã hỗ trợ chúng tôi một cách tận tình, hướng dẫn chúng tôi về cách tư duy và phương pháp làm việc khoa học. Những góp ý quý báu của thầy không chỉ giúp đỡ trong quá trình thực hiện luận văn này mà còn là nguồn động viên quan trọng cho hành trang học thuật của chúng em trong tương lai.

Chúng em kính chúc những điều tốt đẹp nhất sẽ luôn đi kèm và đồng hành cùng quý thầy, cô và mọi người.

MỤC LỤC

	Trang
TRANG PHỤ BÌA.....	i
LỜI CAM ĐOAN.....	ii
LỜI CẢM ƠN.....	iii
MỤC LỤC.....	1
DANH MỤC CÁC CHỮ VIẾT TẮT.....	8
DANH MỤC CÁC BẢNG.....	9
DANH MỤC CÁC SƠ ĐỒ.....	10
DANH MỤC CÁC HÌNH ẢNH.....	13
LỜI MỞ ĐẦU.....	16
0.1. Tính cấp thiết của đề tài.....	16
0.2. Lý do chọn đề tài.....	16
0.3. Mục đích của khóa luận tốt nghiệp.....	17
0.4. Phạm vi nghiên cứu.....	18
0.5. Phương pháp nghiên cứu.....	18
0.6. Cấu trúc của khóa luận tốt nghiệp.....	19
CHƯƠNG 1: GIỚI THIỆU VỀ .NET MAUI	
1.1. Đa nền tảng.....	20
1.1.1. Định nghĩa.....	20
1.1.2. Tâm quan trọng.....	20
1.2. .NET MAUI.....	21
1.2.1. Định nghĩa.....	21

1.2.2. Đối tượng mục tiêu của .NET MAUI..... 22

1.2.3. Lịch sử phát triển..... 23

1.2.4. Các tính năng chính .NET MAUI..... 25

1.2.5. Các nền tảng hỗ trợ..... 25

1.2.6. So sánh .NET MAUI với các Framework đa nền tảng khác..... 26

CHƯƠNG 2: KIẾN THỨC NỀN TẢNG CỦA .NET MAUI

2.1. Nền tảng hình thành .NET MAUI..... 27

2.2. Cách hoạt động của .NET MAUI..... 30

2.3. Cấu trúc của một project .NET MAUI..... 31

 2.3.1. Single Project..... 31

 2.3.2. Cấu trúc chung..... 32

 2.3.3. Thư mục Platforms..... 34

 2.3.3.1. Android..... 35

 2.3.3.2. iOS..... 36

 2.3.3.3. MacCatalyst..... 36

 2.3.3.4. Tizen..... 37

 2.3.3.5. Windows..... 38

 2.3.4. Thư mục Resources..... 39

 2.3.5. Code-behind..... 40

2.4. App Lifecycle..... 41

CHƯƠNG 3: THIẾT KẾ GIAO DIỆN NGƯỜI DÙNG VỚI XAML TRONG .NET MAUI

3.1. Alignment và vị trí..... 46

3.1.1. Alignment.....	46
3.1.2. Điều chỉnh vị trí.....	47
3.2. View.....	49
3.2.1. Hiện thị dữ liệu.....	50
3.2.2. Khởi tạo lệnh.....	53
3.2.3. Cài đặt giá trị.....	56
3.2.4. Chính sửa văn bản.....	58
3.2.5. Biểu thị hoạt động.....	59
3.2.6. Hiển thị tập hợp.....	60
3.3. Layout.....	64
3.3.1. StackLayout.....	64
3.3.1.1. VerticalStackLayout.....	65
3.3.1.2. HorizontalStackLayout.....	66
3.3.2. Grid.....	67
3.3.3. FlexLayout.....	70
3.3.4. AbsoluteLayout.....	73
3.4. Page.....	75
3.4.1. ContentPage.....	75
3.4.2. FlyoutPage.....	76
3.4.3. NavigationPage.....	79
3.4.4. TabbedPage.....	85
3.5. Shell.....	87
3.5.1. Flyout.....	88

3.5.1.1. Mục Flyout.....	88
3.5.1.2. Menu.....	89
3.5.1.3. Header.....	89
3.5.1.4. Footer.....	89
3.5.2. Tab.....	90
3.5.3. Navigation.....	90
3.5.4. Search.....	90

CHƯƠNG 4: MÔ HÌNH MVVM VÀ KẾT NỐI DỮ LIỆU TRONG .NET MAUI

4.1. Mô hình MVVM.....	92
4.1.1. View.....	93
4.1.2. ViewModel.....	94
4.1.3. Model.....	95
4.1.4. Các giải pháp khác.....	95
4.2. Kết nối dữ liệu.....	95

CHƯƠNG 5: PHÁT TRIỂN ỨNG DỤNG MINH HỌA CHO .NET MAUI

5.1. Sơ đồ Use-case và đặc tả cho từng chức năng.....	98
5.1.1. Chức năng đăng nhập.....	99
5.1.2. Chức năng đăng ký.....	102
5.1.3. Chức năng đăng xuất.....	103
5.1.4. Chức năng quản lý bài viết.....	105
5.1.5. Chức năng xem bài viết.....	107

5.2. Sơ đồ hoạt động (Activity Diagram).....	110
5.2.1. Chức năng đăng nhập.....	111
5.2.2. Chức năng đăng ký.....	112
5.2.3. Chức năng quản lý bài viết.....	113
5.2.4. Chức năng xem bài viết.....	114
5.2.5. Sơ đồ hoạt động của chức năng đăng xuất.....	115
5.3. Sơ đồ tuần tự	115
5.3.1. Chức năng đăng nhập.....	113
5.3.1.1. Lựa chọn đăng nhập thẳng.....	117
5.3.1.2. Lựa chọn quên mật khẩu.....	118
5.3.2. Chức năng đăng ký.....	119
5.3.3. Chức năng quản lý bài viết.....	120
5.3.3.1. Lựa chọn đăng bài viết.....	121
5.3.3.2. Lựa chọn sửa bài viết.....	122
5.3.3.3. Lựa chọn xóa bài viết.....	123
5.3.4. Chức năng xem bài viết.....	124
5.3.4.1. Lựa chọn thêm bình luận.....	125
5.3.4.2. Lựa chọn sửa bình luận.....	126
5.3.4.3. Lựa chọn xóa bình luận.....	127
5.3.4.4. Lựa chọn đánh giá bài viết.....	128
5.3.4.5. Lựa chọn lưu bài viết.....	129
5.4. Sơ đồ lớp sau phân tích thiết kế.....	130
5.4.1. Đăng nhập và đăng ký.....	130

5.4.2. Chức năng quản lý bài viết	130
5.4.3. Chức năng xem bài viết	131
5.4.4. Các lớp DTO.....	132
5.5. Bản thiết kế cho cơ sở dữ liệu.....	132
5.5.1. Sơ đồ thực thể kết hợp.....	132
5.5.2. Sơ đồ ảnh xạ quan hệ các thực thể trong sơ đồ thực thể kết hợp.....	133
5.5.3. Cài đặt cơ sở dữ liệu.....	134
5.6. Các công nghệ áp dụng.....	134
5.7. Giao diện người dùng.....	136
5.7.1. Màn hình Đăng Nhập.....	137
5.7.2. Màn hình Đăng Ký.....	138
5.7.3. Màn hình Trang Chủ.....	139
5.7.4. Màn hình Thông Tin Người Dùng.....	140
5.7.5. Màn hình Chính Sửa Thông Tin Người Dùng.....	141
5.7.6. Màn hình Quản Lý Blog.....	142
5.7.7. Màn hình Chính Sửa Blog.....	143
5.7.8. Màn hình Thêm Blog.....	144
5.7.9. Màn hình Blog Đã Lưu.....	145
5.7.10. Màn hình Chi Tiết Blog.....	146
5.7.11. Màn hình Bình Luận.....	147
5.7.12. Màn hình Flyout.....	148
5.7.13. Giao diện được triển khai trên các nền tảng khác nhau.....	149

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. Kết quả đạt được.....	150
6.2. Hạn chế.....	150
6.3. Hướng phát triển.....	151
Tài liệu tham khảo.....	152

DANH MỤC CÁC CHỮ VIẾT TẮT

.NET MAUI	:	.NET Multi-platform App UI
MVVM	:	Model-View-ViewModel
XAML	:	Extensible Application Markup Language

DANH MỤC CÁC BẢNG

STT	Tên	Trang
1	Bảng 1.1: So sánh .NET MAUI với Flutter và React Native	26
2	Bảng 2.1: Các sự kiện vòng đời được ánh xạ đến các nền tảng khác nhau	43
3	Bảng 5.1: Đặc tả chức năng đăng nhập	97
4	Bảng 5.2: Đặc tả chức năng đăng ký	100
5	Bảng 5.3: Đặc tả chức năng đăng xuất	102
6	Bảng 5.4: Đặc tả chức năng quản lý bài viết	103
7	Bảng 5.5: Đặc tả chức năng xem bài viết	106

DANH MỤC CÁC SƠ ĐỒ

STT	Tên	Trang
1	Sơ đồ 2.1: Cách hoạt động của .NET MAUI	30
2	Sơ đồ 2.2: Mô tả App Lifecycle	42
3	Sơ đồ 4.1: Mô tả mô hình MVVM	92
4	Sơ đồ 5.1: Use-case tổng thể của ứng dụng	98
5	Sơ đồ 5.2: Use-case đăng nhập	99
6	Sơ đồ 5.3: Use-case đăng ký	102
7	Sơ đồ 5.4: Use-case đăng xuất	103
8	Sơ đồ 5.5: Use-case quản lý bài viết	105
9	Sơ đồ 5.6: Use-case xem bài viết	107
10	Sơ đồ 5.7: Sơ đồ hoạt động của chức năng đăng nhập	111
11	Sơ đồ 5.8: Sơ đồ hoạt động của chức năng đăng ký	112
12	Sơ đồ 5.9: Sơ đồ hoạt động của chức năng quản lý bài viết	113
13	Sơ đồ 5.10: Sơ đồ hoạt động của chức năng xem bài viết	114
14	Sơ đồ 5.11: Sơ đồ hoạt động của chức năng đăng xuất	115
15	Sơ đồ 5.12: Sơ đồ tuần tự tổng thể của chức năng đăng nhập	116
16	Sơ đồ 5.13: Sơ đồ tuần tự của lựa chọn đăng nhập thẳng	117
17	Sơ đồ 5.14: Sơ đồ tuần tự của lựa chọn quên mật khẩu	118

18	Sơ đồ 5.15: Sơ đồ tuần tự của chức năng đăng ký	119
19	Sơ đồ 5.16: Sơ đồ tuần tự tổng thể của chức năng quản lý bài viết	120
20	Sơ đồ 5.17: Sơ đồ tuần tự của lựa chọn đăng bài viết	121
21	Sơ đồ 5.18: Sơ đồ tuần tự của lựa chọn sửa bài viết	122
22	Sơ đồ 5.19: Sơ đồ tuần tự của lựa chọn xóa bài viết	123
23	Sơ đồ 5.20: Sơ đồ tuần tự tổng thể của chức năng xem bài viết	124
24	Sơ đồ 5.21: Sơ đồ tuần tự của lựa chọn thêm bình luận	125
25	Sơ đồ 5.22: Sơ đồ tuần tự của lựa chọn sửa bình luận	126
26	Sơ đồ 5.23: Sơ đồ tuần tự của lựa chọn xóa bình luận	127
27	Sơ đồ 5.24: Sơ đồ tuần tự của lựa chọn đánh giá bài viết	128
28	Sơ đồ 5.25: Sơ đồ tuần tự của lựa chọn lưu bài viết	129
29	Sơ đồ 5.26: Sơ đồ lớp thu được trong phần phân tích chức năng đăng nhập và đăng ký	130
30	Sơ đồ 5.27: Sơ đồ lớp thu được trong phần phân tích chức năng quản lý bài viết	130
31	Sơ đồ 5.28: Sơ đồ lớp thu được trong phần phân tích chức năng xem bài viết	131
32	Sơ đồ 5.29: Sơ đồ lớp DTO trong ứng dụng	132
33	Sơ đồ 5.30: Sơ đồ thực thể kết hợp	133
34	Sơ đồ 5.31: Sơ đồ ảnh xạ quan hệ các thực thể trong sơ đồ thực thể kết hợp	133

35	Sơ đồ 5.32: Sơ đồ quan hệ thực thể được thể hiện trong SQL Server	134
36	Sơ đồ 5.33: Sơ đồ thể hiện quan hệ giữa các màn hình	136

DANH MỤC CÁC HÌNH ẢNH

STT	Tên	Trang
1	Hình 2.1: Cấu trúc chung của một project .NET MAUI	32
2	Hình 2.2: Cấu trúc của thư mục Platforms	34
3	Hình 2.3: Cấu trúc của thư mục Android	35
4	Hình 2.4: Cấu trúc của thư mục iOS	36
5	Hình 2.5: Cấu trúc của thư mục MacCatalyst	36
6	Hình 2.6: Cấu trúc của thư mục Tizen	37
7	Hình 2.7: Cấu trúc của thư mục Windows	38
8	Hình 2.8: Cấu trúc của thư mục Resources	39
9	Hình 2.9: Code minh họa cho Code-behind	41
10	Hình 3.1: Minh họa các control trong giao diện người dùng	45
11	Hình 3.2: View được căn chỉnh bằng HorizontalOptions	46
12	Hình 3.3: View được căn chỉnh bằng VerticalOptions	47
13	Hình 3.4: Minh họa margin và padding	48
14	Hình 3.5: Margin và padding	49
15	Hình 3.6: Shape, Label và BoxView	50
16	Hình 3.7: Frame, Label và Image	51
17	Hình 3.8: WebView	52
18	Hình 3.9: Border	53
19	Hình 3.10: Button và SwipeView	54

20	Hình 3.11: ImageButton	54
21	Hình 3.12: RadioButton	55
22	Hình 3.13: SearchBar	55
23	Hình 3.14: CheckBox, Switch và Slider	56
24	Hình 3.15: Stepper	57
25	Hình 3.16: TimePicker và DatePicker	58
26	Hình 3.17: Entry và Editor	59
27	Hình 3.18: ActivityIndicator và ProgressBar	60
28	Hình 3.19: CarouselView và IndicatorView	61
29	Hình 3.20: ListView	62
30	Hình 3.21: CollectionView	62
31	Hình 3.22: Picker	63
32	Hình 3.23: TableView	64
33	Hình 3.24: Minh họa StackLayout	64
34	Hình 3.25: VerticalStackLayout	66
35	Hình 3.26: HorizontalStackLayout	67
36	Hình 3.27: Minh họa Grid	67
37	Hình 3.28: Grid	70
38	Hình 3.29: FlexLayout	72
39	Hình 3.30: Minh họa AbsoluteLayout	73
40	Hình 3.31: AbsoluteLayout	74
41	Hình 3.32: Minh họa ContentPage	75
42	Hình 3.33: Minh họa FlyoutPage	76

43	Hình 3.34: FlyoutPage	78
44	Hình 3.35: Minh họa NavigationPage	79
45	Hình 3.36: NavigationPage	81
46	Hình 3.37: Minh họa hoạt động thêm trang	82
47	Hình 3.38: Minh họa hoạt động quay lại trang trước	83
48	Hình 3.39: Minh họa hoạt động xóa trang	83
49	Hình 3.40: Minh họa hoạt động chèn trang	83
50	Hình 3.40: Minh họa TabbedPage	84
51	Hình 3.41: TabbedPage	85
52	Hình 5.1: Màn hình đăng nhập	137
53	Hình 5.2: Màn hình đăng ký	138
54	Hình 5.3: Màn hình trang chủ	139
55	Hình 5.4: Màn hình thông tin người dùng	140
56	Hình 5.5: Màn hình chỉnh sửa thông tin người dùng	141
57	Hình 5.6: Màn hình quản lý blog	142
58	Hình 5.7: Màn hình chỉnh sửa blog	143
59	Hình 5.8: Màn hình thêm blog	144
61	Hình 5.9: Màn hình blog đã lưu	145
62	Hình 5.10: Màn hình chi tiết blog	146
63	Hình 5.11: Màn hình bình luận	147
64	Hình 5.12: Màn hình flyout hỗ trợ điều hướng giữa các màn hình trong ứng dụng	148
65	Hình 5.13: Ứng dụng được triển khai trên các nền tảng khác nhau	149

LỜI MỞ ĐẦU

0.1. Tính cấp thiết của đề tài

Trong thời đại hiện nay, đa nền tảng là xu hướng quan trọng cho sự phát triển của ngành công nghệ thông tin tại Việt Nam cũng như toàn thế giới. Đa nền tảng giúp tiết kiệm thời gian và nguồn lực, với khả năng phát triển một lần và triển khai trên nhiều hệ điều hành.

Các Framework như React Native, Flutter và Xamarin đáp ứng nhu cầu linh hoạt và nhanh chóng của doanh nghiệp phát triển ứng dụng trên nhiều nền tảng. Tuy nhiên, nhiều Framework đa nền tảng hiện nay lại gặp vấn đề hiệu suất hoặc khả năng kém đồng bộ giao diện người dùng giữa các nền tảng khác nhau.

Vì vậy, ngành công nghệ thông tin nói chung và ngành lập trình đa nền tảng nói riêng cần một giải pháp mới, một giải pháp thay thế, một Framework đa nền tảng có hiệu suất ổn định cùng với khả năng tùy chỉnh giao diện người dùng linh hoạt. Và đó là .NET MAUI - một Framework đa nền tảng mới được xây dựng trên Xamarin đến từ Microsoft.

Việc được xây dựng trên Xamarin - một Framework đa nền tảng chiếm trung bình 15% thị trường lập trình đa tảng trong những năm trở lại đây đã mang lại lợi thế rất lớn cho .NET MAUI. Bên cạnh đó, cùng với sự phát triển của thị trường ứng dụng di động, desktop và tỷ lệ ứng dụng đa nền tảng ngày càng gia tăng, có thể nói, khả năng .NET MAUI sẽ chiếm lĩnh thị phần đáng kể trong tương lai là rất cao.

0.2. Lý do chọn đề tài

.NET MAUI, được phát triển và đầu tư bởi Microsoft, dù mới xuất hiện, nhưng lại mang đến tiềm năng cho ngành lập trình đa nền tảng. Dù sinh sau đẻ muộn, .NET MAUI lại có lợi thế ké thừa những điểm mạnh từ các Framework tiền bối, đặc biệt là Xamarin - "đứa con cưng" của Microsoft trong lĩnh vực lập trình đa nền tảng.

Việc áp dụng Single Project khiến khả năng bảo trì của .NET MAUI trở nên dễ dàng hơn so với người tiền nhiệm là Xamarin. Đồng thời, việc sử dụng trình biên dịch AOT (Ahead-of-Time) để chuyển đổi mã C# sang mã máy, mang lại tốc độ ứng dụng nhanh hơn so với React Native, thậm chí là có thể sánh ngang với Flutter, tuy nhiên, so với ngôn ngữ Dart được sử dụng trong Flutter, .NET MAUI khả năng tiếp cận dễ dàng hơn khi sử dụng C#.

Và việc khám phá và ứng dụng những tiềm năng to lớn của .NET MAUI là lý do chính của đề tài, qua đó, có thể quảng bá Framework này đến cộng đồng lập trình Việt Nam trong bối cảnh sự hiện diện của Framework này đã có nhưng lại không được bàn luận nhiều giới công nghệ Việt Nam.

0.3. Mục đích của khóa luận tốt nghiệp

Chọn .NET MAUI làm đề tài cho khóa luận không chỉ xuất phát từ những lợi ích cụ thể mà Framework này mang lại, mà còn do sự độc đáo và tiềm năng lớn từ Microsoft. .NET MAUI đang là mảng công nghệ mới, đầy tính năng chưa khám phá, làm cho nó trở thành sân chơi sáng tạo cho nhà phát triển. Điều này cung cấp cơ hội thú vị để thử nghiệm và đóng góp vào sự phát triển của Framework.

Thêm vào đó, thị trường sử dụng .NET MAUI hiện tại còn hạn chế, đặc biệt tại Việt Nam, tạo cơ hội để nghiên cứu và đóng góp vào phổ biến Framework này. Việc .NET MAUI được phát triển bởi Microsoft, một công ty hàng đầu thế giới, mang lại giá trị và tiềm năng lớn.

Với mục tiêu là cung cấp cho người đọc cái nhìn tổng quan về .NET MAUI và các khái niệm cốt lõi của Framework này, bao gồm lịch sử phát triển, kiến trúc, thiết kế giao diện, tính năng và lợi ích chính thay vì đi sâu vào các chi tiết kỹ thuật phức tạp. Bên cạnh đó là các kiến thức liên qua như .NET, C#, Xamarin,... giúp người đọc nắm bắt được kiến thức nền tảng vững chắc và bản chất của Framework để họ có thể tự tin khám phá và ứng dụng .NET MAUI trong các dự án thực tế.

Đồng thời, việc xây dựng ứng dụng minh họa đa nền tảng bằng .NET MAUI sử dụng khả năng thiết kế giao diện người dùng mạnh mẽ cùng với các mô hình xây dựng ứng dụng khác nhau giúp cho người đọc có thể thấy được sự linh hoạt của Framework này.

0.4. Phạm vi nghiên cứu

Với tư cách là một Framework mới, .NET MAUI hiện vẫn chưa phổ biến rộng rãi và chưa được áp dụng nhiều trong đời sống hàng ngày. Do đó chúng tôi muốn đánh giá tổng quan về sức linh hoạt của .NET MAUI và cách nó có thể đáp ứng đa dạng nhu cầu của cộng đồng người sử dụng. Qua việc nghiên cứu từng khía cạnh của .NET MAUI, từ khả năng viết ứng dụng chạy trên nhiều nền tảng chỉ từ một code-base, cùng khả năng thiết kế giao diện người dùng mạnh mẽ, đến sự tương thích cũng như khả năng áp dụng các mô hình ứng dụng và các công nghệ lập trình hiện đại, chúng tôi hy vọng mang đến cái nhìn toàn diện về tiềm năng và ứng dụng của Framework này trong thực tế, giúp cộng đồng hiểu rõ hơn về lợi ích và hạn chế của .NET MAUI.

0.5. Phương pháp nghiên cứu

Phân tích, nghiên cứu tài liệu, sách, video hướng dẫn,... để bước đầu có được những kiến thức cơ bản về .NET MAUI. Bên cạnh đó là tìm hiểu về ý kiến của người dùng, cũng như các Framework tương tự khác, từ đó có thể đánh giá .NET MAUI một cách toàn diện. Ngoài ra còn nghiên cứu thêm về các lý thuyết, dù không trực tiếp liên quan, nhưng là nền tảng của .NET MAUI. Và cuối cùng là triển khai xây dựng ứng dụng minh họa.

0.6. Cấu trúc của khóa luận tốt nghiệp

Cấu trúc của khóa luận bao gồm 6 chương chính:

- Chương 1: Giới Thiệu Về .NET MAUI**
- Chương 2: Nền Tảng Của .NET MAUI**
- Chương 3: Thiết Kế Giao Diện Người Dùng Với XAML Trong .NET MAUI**
- Chương 4: Mô Hình MVVM Và Kết Nối Dữ Liệu Trong .NET MAUI**
- Chương 5: Phát Triển Ứng Dụng Minh Họa Cho .NET MAUI**
- Chương 6: Kết Luận Và Hướng Phát Triển**

CHƯƠNG 1: GIỚI THIỆU VỀ .NET MAUI

1.1. Đa nền tảng

1.1.1. Định nghĩa

Đa nền tảng, hay còn gọi là multiplatform, là một khái niệm quan trọng trong lĩnh vực công nghệ thông tin, đặc biệt là trong việc phát triển phần mềm và ứng dụng. Thuật ngữ này ám chỉ khả năng của một sản phẩm hoặc ứng dụng có thể hoạt động một cách linh hoạt trên nhiều hệ điều hành và nền tảng khác nhau mà không yêu cầu các sửa đổi lớn trong mã nguồn hay thiết kế.

Nói cách khác, với tính đa nền tảng, nhà phát triển chỉ cần viết một lần code duy nhất, sau đó có thể triển khai ứng dụng trên nhiều hệ điều hành phổ biến như Android, iOS, macOS, Windows, Linux, web,...

1.1.2. Tầm quan trọng

Trong bối cảnh công nghệ ngày nay, sự đa dạng của hệ điều hành, đặc biệt là các thiết bị di động, đã trở thành một thách thức lớn cho các nhà phát triển phần mềm. Nhu cầu ngày càng cao về các ứng dụng tương thích trên nhiều nền tảng khác nhau như Windows, macOS, Linux, iOS, Android và hơn thế nữa đòi hỏi các giải pháp sáng tạo và hiệu quả.

Sự bùng nổ của điện thoại thông minh đã biến chúng thành thiết bị không thể thiếu trong cuộc sống hàng ngày. Tuy nhiên, khi làm việc, con người vẫn sử dụng các thiết bị máy tính bảng, laptop và PC để đảm bảo năng suất và hiệu quả công việc. Hơn nữa, nhiều tác vụ đòi hỏi sự đồng bộ hóa dữ liệu và thông tin trên cả điện thoại và máy tính, tạo nên nhu cầu cấp thiết cho các ứng dụng đa nền tảng.

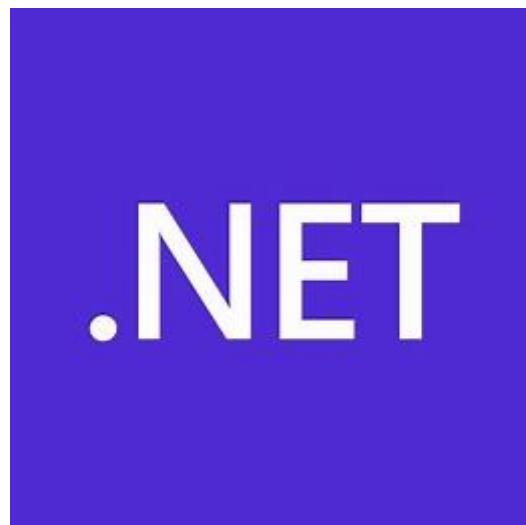
Để đối mặt với thách thức này, người phát triển thường áp dụng các kỹ thuật và phương pháp nhất định để đảm bảo tính đa nền tảng của sản phẩm. Một trong những cách phổ biến là sử dụng ngôn ngữ lập trình chung có khả năng chạy trên nhiều hệ điều hành, như Java, Python, hoặc JavaScript. Sự linh hoạt của những

ngôn ngữ này giúp giảm bớt gánh nặng khi phát triển ứng dụng cho nhiều môi trường khác nhau.

Ngoài ra, các Framework đa nền tảng cũng được phát triển để hỗ trợ việc xây dựng ứng dụng linh hoạt trên nhiều hệ điều hành. Các ví dụ bao gồm React Native, Xamarin và Flutter, những công cụ giúp tối ưu hóa quá trình phát triển và đồng thời giảm bớt khả năng phải duy trì nhiều phiên bản riêng biệt của ứng dụng.

Tổng quát, khả năng đa nền tảng của một ứng dụng không chỉ giúp tiết kiệm thời gian và chi phí trong quá trình phát triển mà còn mở rộng khả năng tiếp cận đối với một lượng người dùng đa dạng, không bị ràng buộc bởi sự đa dạng của hệ điều hành và thiết bị. Điều này đóng vai trò quan trọng trong việc tạo ra trải nghiệm người dùng tích cực và thú vị trên mọi nền tảng.

1.2. .NET MAUI



1.2.1. Định nghĩa

.NET Multi-platform App UI (.NET MAUI) là một Framework được phát triển bởi Microsoft để xây dựng ứng dụng đa nền tảng trên môi trường .NET sử dụng ngôn ngữ lập trình C# và ngôn ngữ đánh dấu Extensible Application Markup Language (XAML).

Ngôn ngữ lập trình và các Framework đa nền tảng đóng một vai trò quan trọng trong quá trình phát triển ứng dụng có khả năng hoạt động trên nhiều hệ điều hành và thiết bị. Trong số các ngôn ngữ phổ biến được đề cập ở trên như Java, Python và JavaScript thường được ưu chuộng do khả năng chạy trên nhiều nền tảng khác nhau. Tuy nhiên, xuất hiện của .NET MAUI đã mang đến một giải pháp mới bằng việc sử dụng ngôn ngữ C#. Mặc dù C# không phải là ngôn ngữ phổ biến nhưng .NET MAUI đã tận dụng sức mạnh của C# để tạo ra một trải nghiệm phát triển ứng dụng đa nền tảng hiệu quả dành cho các nhà phát triển muốn tìm đến một ngôn ngữ lập trình đa nền tảng khác.

Bên cạnh đó, việc .NET MAUI sử dụng Xamarin như là một phần quan trọng của nền tảng sẽ có ích cho các nhà phát triển đã có kinh nghiệm với Xamarin, giúp các nhà phát triển có sẵn kinh nghiệm phát triển ứng dụng đa nền tảng bằng Xamarin có thể dễ dàng chuyển đổi và tiếp tục sử dụng kiến thức đã có, tạo ra sự liên tục và sự thuận tiện trong quá trình chuyển đổi sang phát triển ứng dụng đa nền tảng.

1.2.2. Đối tượng mục tiêu của .NET MAUI

- **Nhà phát triển phần mềm:** .NET MAUI là lựa chọn lý tưởng cho các nhà phát triển phần mềm có kinh nghiệm lập trình C# và .NET, muốn phát triển ứng dụng di động và desktop đa nền tảng hiệu quả. .NET MAUI cũng hỗ trợ tích hợp với các dịch vụ và công nghệ Microsoft khác như Azure, Visual Studio và .NET Core, giúp nhà phát triển dễ dàng xây dựng các giải pháp phần mềm toàn diện.
- **Doanh nghiệp:** .NET MAUI mang lại giải pháp tiết kiệm chi phí và hiệu quả cho doanh nghiệp muốn phát triển ứng dụng di động và desktop đa nền tảng. Doanh nghiệp có thể sử dụng .NET MAUI để xây dựng một lần và triển khai trên nhiều nền tảng, giúp giảm thiểu chi phí phát triển và bảo trì.

Framework này cũng hỗ trợ tích hợp với các hệ thống doanh nghiệp hiện có, giúp doanh nghiệp dễ dàng triển khai ứng dụng vào quy trình hoạt động.

- **Sinh viên ngành công nghệ thông tin:** .NET MAUI là công nghệ mới mẻ và đầy tiềm năng, giúp sinh viên tiếp cận và cập nhật kiến thức và kỹ năng lập trình đa nền tảng mới nhất. Ngoài ra, việc học tập và sử dụng .NET MAUI sẽ giúp sinh viên nâng cao khả năng cạnh tranh trên thị trường lao động. .NET MAUI còn là chủ đề nghiên cứu thú vị cho các sinh viên muốn tìm hiểu về các xu hướng phát triển phần mềm mới nhất.

1.2.3. Lịch sử phát triển

Năm 2019:

- **Tháng 11:** Microsoft công bố dự án .NET MAUI trong khuôn khổ hội nghị Build 2019. Mục tiêu của dự án là tạo ra một nền tảng ứng dụng di động đa nền tảng duy nhất, cho phép lập trình viên viết mã một lần để triển khai trên nhiều hệ điều hành như Android, iOS, macOS và Windows.

Năm 2020:

- **Tháng 5:** Microsoft phát hành bản xem trước đầu tiên của .NET MAUI. Phiên bản này bao gồm các công cụ cơ bản để xây dựng ứng dụng di động với giao diện người dùng gốc, khả năng điều hướng trang và truy cập các tính năng của thiết bị.
- **Tháng 9:** Microsoft ra mắt bản xem trước thứ hai của .NET MAUI, bổ sung thêm nhiều tính năng mới như hỗ trợ WebView, khả năng tích hợp với bản đồ và hỗ trợ cho các thiết bị đeo được.

Năm 2021:

- **Tháng 3:** Microsoft tung ra bản xem trước thứ ba của .NET MAUI, tập trung vào cải thiện hiệu suất và độ ổn định. Phiên bản này cũng giới thiệu

một số tính năng mới như hỗ trợ cho Blazor Hybrid và khả năng tạo các ứng dụng đa cửa sổ.

- **Tháng 11:** Microsoft phát hành .NET MAUI Preview 4, mang đến nhiều cải tiến cho hiệu suất, khả năng truy cập và hỗ trợ cho các nền tảng mục tiêu. Phiên bản này cũng đánh dấu lần đầu tiên Microsoft chính thức khuyến khích các nhà phát triển sử dụng .NET MAUI để xây dựng các ứng dụng di động mới.

Năm 2022:

- **Tháng 5:** Microsoft ra mắt .NET MAUI 6, phiên bản chính thức đầu tiên của nền tảng này. .NET MAUI 6 bao gồm tất cả các tính năng từ các bản xem trước trước đó, đồng thời bổ sung thêm nhiều cải tiến khác như hỗ trợ tốt hơn cho các thiết bị gập, hiệu suất đồ họa được tăng tốc và khả năng tích hợp với các dịch vụ đám mây của Microsoft.
- **Tháng 11:** Microsoft phát hành .NET MAUI 7, giới thiệu một số tính năng mới như hỗ trợ cho Apple Watch và Samsung Galaxy Watch, khả năng tạo các ứng dụng desktop liền mạch và cải thiện hiệu suất cho các thiết bị Android.

Năm 2023:

- **Tháng 3:** Microsoft tung ra .NET MAUI 8, bổ sung thêm nhiều tính năng mới cho các nhà phát triển ứng dụng doanh nghiệp, bao gồm hỗ trợ cho các ứng dụng LOB, khả năng tích hợp với Microsoft Dataverse và các cải tiến bảo mật.
- **Tháng 11:** Microsoft phát hành .NET MAUI 9, tập trung vào cải thiện trải nghiệm nhà phát triển với các công cụ mới, tài liệu được cập nhật và hỗ trợ cộng đồng tốt hơn. Phiên bản này cũng giới thiệu một số tính năng mới như hỗ trợ cho các thiết bị AR/VR và khả năng tạo các ứng dụng web di động.

Năm 2024:

- **Tháng 4:** .NET MAUI tiếp tục phát triển với các bản cập nhật thường xuyên, bổ sung thêm tính năng mới và cải thiện hiệu suất. Nền tảng này ngày càng được sử dụng rộng rãi bởi các nhà phát triển để xây dựng các ứng dụng di động đa nền tảng mạnh mẽ và linh hoạt.

1.2.4. Các tính năng chính .NET MAUI

- Hỗ trợ đa nền tảng giúp xây dựng ứng dụng có thể chạy trên nhiều hệ điều hành khác nhau như Windows, Android, iOS, macOS,... mà chỉ sử dụng một code-base chung.
- Hệ thống các thành phần giao diện từ đơn giản đến phức tạp giúp người dùng có thể dễ dàng thiết kế giao diện mong muốn, bên cạnh đó còn có thể tùy chỉnh handler giúp nâng cao khả năng thiết kế giao diện.
- Hỗ trợ data-binding (kết nối dữ liệu) giữa các thành phần giao diện và dữ liệu một cách nhanh chóng và hiệu quả.
- Tích hợp tốt với các mô hình phát triển phần mềm dễ bảo trì như MVVM.
- Các API đa nền tảng giúp truy cập vào tính năng của thiết bị gốc.
- Hỗ trợ Hot Reload giúp dễ dàng chỉnh sửa code khi ứng dụng vẫn đang chạy.

1.2.5. Các nền tảng hỗ trợ

Với .NET MAUI, các nhà phát triển có thể dễ dàng tạo ra các ứng dụng di động và desktop hoàn chỉnh, hoạt động mượt mà trên nhiều hệ điều hành phổ biến:

- **Android:** Khám phá tiềm năng to lớn của thị trường di động Android với .NET MAUI, hỗ trợ từ Android 5.0 (API 21) trở lên.

- **iOS:** Mang đến trải nghiệm người dùng iOS tuyệt vời bằng .NET MAUI, tương thích hoàn toàn với iOS 11 và các phiên bản mới hơn.
- **macOS:** Tận dụng sức mạnh của macOS với .NET MAUI, xây dựng ứng dụng macOS 10.15 trở lên bằng MacCatalyst.
- **Windows:** Tiếp cận thị trường Windows rộng lớn với .NET MAUI, hỗ trợ Windows 11 và Windows 10 phiên bản 1809 trở lên, sử dụng Windows UI Library (WinUI) 3.
- **Tizen:** Mở rộng thị trường sang các thiết bị Tizen với .NET MAUI, được hỗ trợ bởi Samsung, mang đến trải nghiệm ứng dụng đa dạng và phong phú.

* *MacCatalyst cho phép xây dựng và chia sẻ các ứng dụng Mac với các ứng dụng iPad.*

1.2.6. So sánh .NET MAUI với các Framework đa nền tảng khác

Bảng 1.1: So sánh .NET MAUI với Flutter và React Native

Tiêu chí	.NET MAUI	Flutter	React Native
Ngôn ngữ	C# và XAML	Dart	JavaScript
Ngày ra đời	Được công bố vào tháng 5 năm 2020	Phát hành lần đầu vào tháng 5 năm 2017	Phát hành lần đầu vào năm 2015
Khung làm việc	.NET Core	Flutter SDK	ReactJS
Công nghệ cơ sở	.NET Core	Flutter Engine	ReactJS
Hiệu suất	Tối ưu cho từng nền tảng cụ thể, hiệu suất tốt	Hiệu suất cao, mượt mà trên cả iOS và Android	Hoạt động ổn định, có thể giảm nếu được tùy chỉnh hợp lý

Tích hợp	Tích hợp mạnh mẽ với các dịch vụ của Microsoft	Cung cấp các gói tích hợp cho các dịch vụ phổ biến	Tích hợp dễ dàng với các thư viện và dịch vụ web
Hiệu quả phát triển	Hỗ trợ công cụ phát triển như Visual Studio và VS Code, tích hợp dễ dàng với các dịch vụ của Microsoft	Công cụ phát triển mạnh mẽ, hỗ trợ đầy đủ tính năng hot reload	Có thư viện và công cụ phong phú cho phát triển, tích hợp với các công cụ phát triển JavaScript
Hệ thống UI	Sử dụng XAML hoặc mã nguồn mở	Sử dụng Widget Tree	Sử dụng JSX
Độ phổ biến	Đang phát triển, nhận được sự quan tâm từ cộng đồng	Ngày càng phổ biến, được Google hậu thuẫn	Phổ biến, có nhiều ứng dụng nổi tiếng sử dụng
Hỗ trợ cộng đồng	Hỗ trợ mạnh mẽ từ cộng đồng .NET, tài liệu đầy đủ	Cộng đồng ngày càng lớn, tài liệu đầy đủ và hỗ trợ mạnh mẽ	Có một cộng đồng lớn với nhiều tài liệu và gói mở rộng

CHƯƠNG 2: KIẾN THỨC NỀN TẢNG CỦA .NET MAUI

2.1. Nền tảng hình thành .NET MAUI

.NET:

- Là nền tảng phát triển phần mềm đa nền tảng do Microsoft phát triển.
- Cung cấp các thư viện, công cụ và runtime để xây dựng ứng dụng web, desktop, di động, game, IoT, ...
- NET là nền tảng cốt lõi cho .NET MAUI, cung cấp các tính năng cơ bản như quản lý bộ nhớ, xử lý ngoại lệ, lập trình mạng,...

.NET CLI:

- Là giao diện dòng lệnh đa nền tảng cho .NET (command-line interface).
- Cho phép tạo, xây dựng, chạy và quản lý ứng dụng .NET từ dòng lệnh.
- .NET CLI được sử dụng trong .NET MAUI để thực hiện các tác vụ như tạo dự án mới, cài đặt thư viện, biên dịch mã,...

Xamarin:

- Là Framework mã nguồn mở để phát triển ứng dụng di động cho iOS, Android và Windows Phone bằng C#.
- Cung cấp lớp trừu tượng để truy cập các tính năng và API native của từng nền tảng di động.
- Xamarin là tiền thân của .NET MAUI và đóng vai trò quan trọng trong việc phát triển các ứng dụng di động đa nền tảng trước đây.

C#:

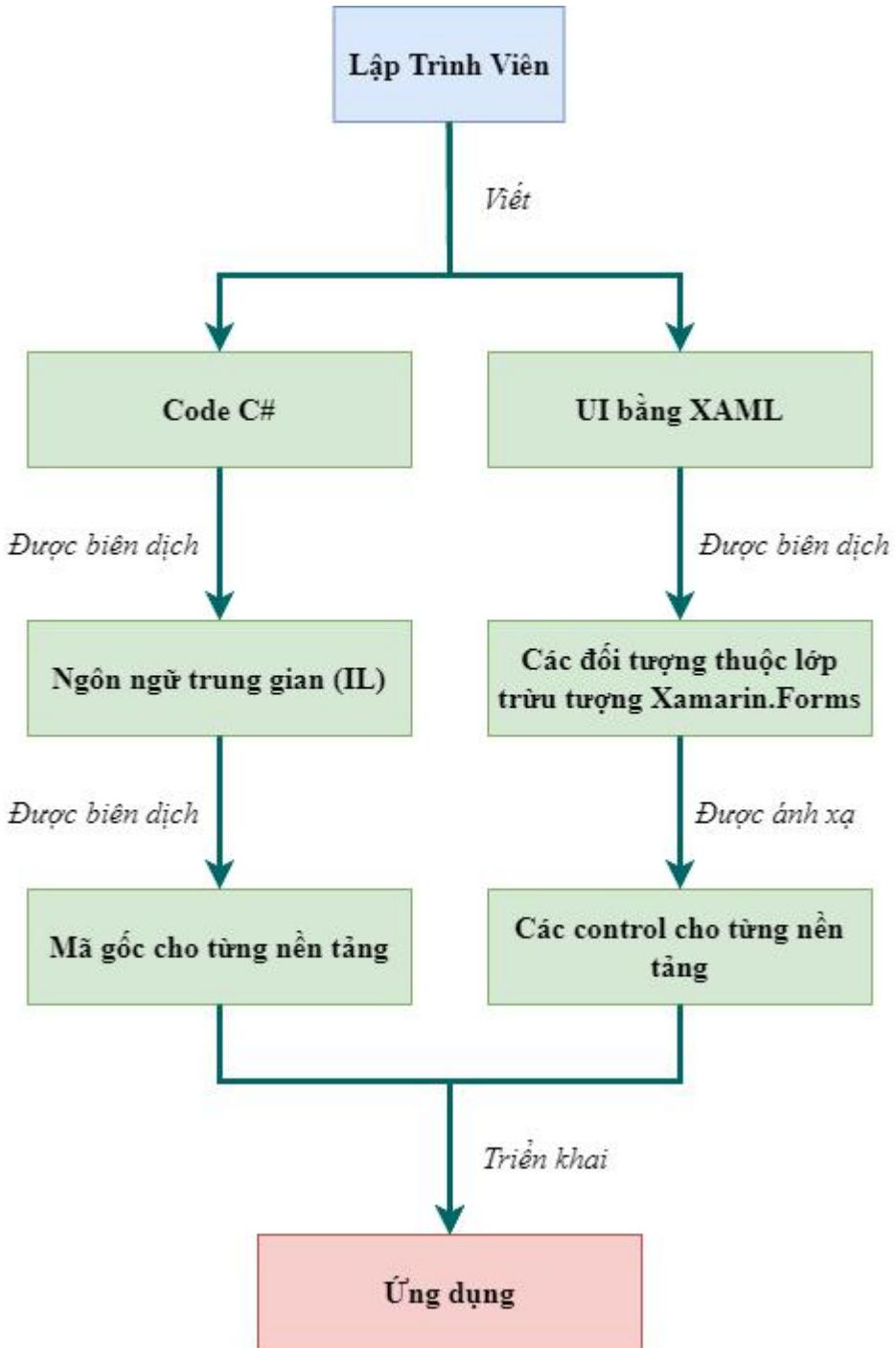
- Là ngôn ngữ lập trình đa năng do Microsoft phát triển.
- Dễ học, dễ sử dụng và được hỗ trợ rộng rãi bởi cộng đồng phát triển lớn.

- C# là ngôn ngữ chính được sử dụng để viết logic ứng dụng trong .NET MAUI.

XAML:

- Là ngôn ngữ đánh dấu mở rộng (Extensible Markup Language) để thiết kế giao diện người dùng cho ứng dụng.
- Cung cấp cú pháp đơn giản, linh hoạt và dễ học để tạo giao diện đẹp mắt và đáp ứng.
- XAML được sử dụng trong .NET MAUI để mô tả các thành phần giao diện, bộ cục và thuộc tính của chúng.

2.2. Cách hoạt động của .NET MAUI



Cách hoạt động của .NET MAUI:

- 1) Lập trình viên viết code C# định nghĩa logic ứng dụng, xử lý dữ liệu và định nghĩa, mô tả giao diện người dùng bằng XAML.
- 2) Trình biên dịch C# biên dịch code C# thành ngôn ngữ trung gian (IL). Trình biên dịch XAML biên dịch các thành phần giao diện được viết bằng XAML thành các đối tượng thuộc lớp trừu tượng Xamarin.Forms, các lớp trừu tượng này không phụ thuộc vào nền tảng cụ thể nào.
- 3) Trên từng nền tảng cụ thể, trình biên dịch AOT (Ahead-of-Time) hoặc JIT (Just-in-Time) sẽ biên dịch ngôn ngữ trung gian thành mã gốc trên nền tảng đó. Các lớp giao diện cụ thể trên nền tảng đó cũng được tải và sẽ chịu trách nhiệm ánh xạ các lớp trừu tượng thành các thành phần giao diện tương đương.
- 4) Ứng dụng được triển khai trên các thiết bị và nền tảng khác nhau.

2.3. Cấu trúc của một project .NET MAUI

2.3.1. Single Project

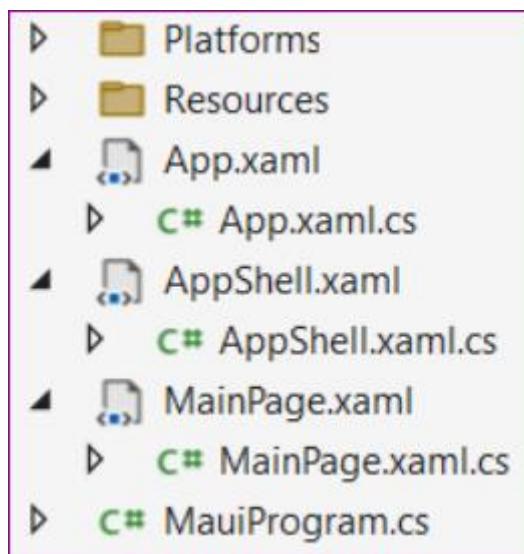
.NET MAUI loại bỏ những rắc rối thường gặp khi phát triển ứng dụng đa nền tảng đó là thay vì phải tạo và quản lý nhiều dự án riêng biệt cho từng hệ điều hành, ta chỉ cần tạo một dự án duy nhất có thể nhắm mục tiêu Android, iOS, macOS, Tizen và Windows, đây gọi một khái niệm khá mới .NET và là một phần quan trọng của .NET MAUI.

Dù ta đang xây dựng ứng dụng cho thiết bị di động hay máy tính để bàn, Single Project sẽ mang đến trải nghiệm phát triển liền mạch và đồng nhất. Nhờ vậy, ta có thể tập trung vào việc viết mã chức năng cốt lõi thay vì lo lắng về những khác biệt của các nền tảng khác nhau.

Lợi ích của Single Project:

- Một code-base chung: Viết code một lần và chạy trên nhiều nền tảng, tiết kiệm thời gian và công sức phát triển.
- Đơn giản hóa debug: Chọn mục tiêu debug dễ dàng để kiểm tra ứng dụng trên từng nền tảng.
- Tệp tài nguyên chung: Quản lý tập trung hình ảnh, tệp cấu hình và các tài nguyên khác cho tất cả các nền tảng.
- Truy cập API từng nền tảng: Sử dụng các API dành riêng cho từng nền tảng khi cần thiết.
- Điểm vào ứng dụng duy nhất: Khởi động ứng dụng trên mọi nền tảng chỉ với một điểm vào.

2.3.2. Cấu trúc chung



Hình 2.1: Cấu trúc chung của một project .NET MAUI

- **Platforms:** Thư mục Platforms đóng vai trò quan trọng trong .NET MAUI, là nơi chứa mã nguồn cụ thể cho từng nền tảng được hỗ trợ. Nó giúp nhà phát triển tối ưu hóa ứng dụng cho từng hệ điều hành, mang lại hiệu suất và

trải nghiệm người dùng tốt nhất. Bên trong thư mục Platforms là một tập hợp các thư mục con, mỗi thư mục đại diện cho một nền tảng cụ thể. Tên thư mục thường trùng với tên của hệ điều hành mà nó hỗ trợ, ví dụ thư mục Platforms/Android hỗ trợ cho nền tảng Android.

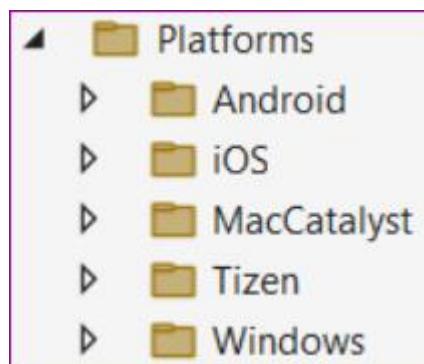
- **Resources:** Thư mục Resources đóng vai trò quan trọng trong .NET MAUI, là nơi lưu trữ tất cả các tài nguyên cần thiết cho ứng dụng. Tài nguyên bao gồm bất kỳ dữ liệu nào nhúng vào ứng dụng mà không phải là mã, chẳng hạn như hình ảnh, font chữ,...
- **MauiProgram.cs:** Lớp MauiProgram.cs là nơi khởi tạo và cấu hình ứng dụng. Nó sử dụng Generic Host Builder, một phương pháp do Microsoft phát triển để gói gọn và quản lý các yêu cầu của ứng dụng một cách hiệu quả.
- **App.xaml.cs:** Lớp App.xaml.cs là điểm truy cập chính cho ứng dụng đa nền tảng. Nó chịu trách nhiệm khởi tạo các thành phần cốt lõi của ứng dụng, định cấu hình các dịch vụ và thiết lập giao diện người dùng gốc cho từng nền tảng.
- **App.xaml:** Tệp App.xaml là nơi lưu trữ các tài nguyên giao diện người dùng (UI) chung có thể được sử dụng trong toàn bộ ứng dụng. Nó giúp định nghĩa các yếu tố UI cơ bản như màu sắc, kiểu chữ, kích thước, bố cục và các thuộc tính giao diện khác, đảm bảo sự nhất quán về mặt hình ảnh và trải nghiệm người dùng trên mọi trang và thành phần trong ứng dụng.
- **MainPage.xaml:** Nơi thiết kế giao diện người dùng (UI) cho trang chủ ứng dụng bằng ngôn ngữ XAML. Tập tin này chịu trách nhiệm bố cục các yếu tố UI như hình ảnh, nút, nhãn,... và định nghĩa cách chúng sẽ hiển thị trên màn hình.
- **MainPage.xaml.cs:** Nơi viết code C# để điều khiển hành vi của trang chủ ứng dụng. Tập tin này xử lý các sự kiện người dùng như nhấp chuột, thay

đổi văn bản,... và thực hiện logic ứng dụng cần thiết để mang lại trải nghiệm người dùng mong muốn.

- **AppShell.xaml:** Nơi thiết kế giao diện người dùng (UI) cho Shell của ứng dụng bằng ngôn ngữ XAML. Tập tin này chịu trách nhiệm định nghĩa cấu trúc Shell, bao gồm các Flyout (menu bên), TabBar (thanh tab) và các trang nội dung.
- **AppShell.xaml.cs:** Nơi viết code C# để điều khiển hành vi của Shell và quản lý việc điều hướng giữa các trang trong ứng dụng. Tập tin này xử lý các sự kiện người dùng như nhấp chuột vào tab, thay đổi mục menu,... và sử dụng API Shell Navigation để điều hướng người dùng đến các trang mong muốn.

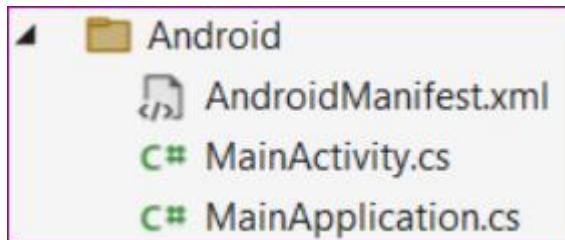
* Khi làm việc với .NET MAUI, ta nhận thấy tệp .xaml thường đi kèm với lớp .xaml.cs, sự kết hợp này xuất phát từ những hạn chế của XAML. Mặc dù XAML xuất sắc trong việc thiết kế giao diện, nó không thể đáp ứng tất cả các nhu cầu của lập trình ứng dụng. Do đó, .xaml.cs ra đời để lấp đầy khoảng trống, cung cấp khả năng viết mã C# mạnh mẽ để kiểm soát hành vi, nghiệp vụ và logic ứng dụng. Tệp C# đi kèm này được gọi là **Code-behind**, sẽ được nói rõ hơn tại mục 2.3.5.

2.3.3. Thư mục Platforms



Hình 2.2: Cấu trúc của thư mục Platforms

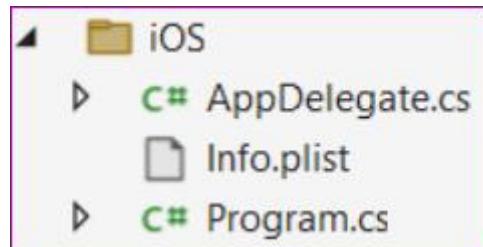
2.3.3.1. Android



Hình 2.3: Cấu trúc của thư mục Android

- **MainApplication.cs:** Là điểm truy cập cho ứng dụng Android. Tạo ra một đối tượng MauiApp bằng cách sử dụng lớp MauiProgram. Nó cung cấp cầu nối giữa ứng dụng Android và code .NET MAUI đa nền tảng.
- **MainActivity.cs:** Hoạt động trong phát triển Android là một loại thành phần ứng dụng cung cấp giao diện người dùng. MainActivity bắt đầu khi ứng dụng được tải. Việc này thường được thực hiện bằng cách nhấn vào biểu tượng ứng dụng; tuy nhiên, nó cũng có thể được kích hoạt bởi một thông báo hoặc nguồn khác. Thường chứa code để thiết lập giao diện người dùng ban đầu và xử lý các sự kiện người dùng cơ bản.
- **AndroidManifest.xml:** Là tệp cấu hình ứng dụng Android có vai trò quan trọng trong việc triển khai ứng dụng Android. Giúp khai báo các quyền mà ứng dụng yêu cầu, cung cấp thông tin phiên bản ứng dụng, SDK mục tiêu và tối thiểu, liệt kê các tính năng phần cứng và phần mềm cần thiết cho ứng dụng.

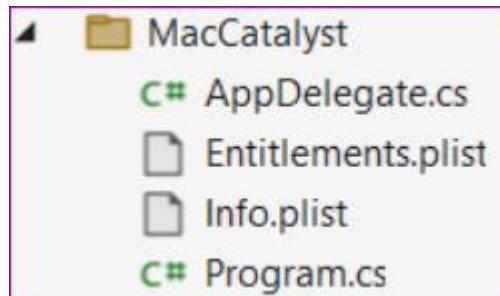
2.3.3.2. iOS



Hình 2.4: Cấu trúc của thư mục iOS

- **AppDelegate.cs:** Quản lý vòng đời ứng dụng, đóng vai trò quan trọng trong việc đảm bảo ứng dụng iOS hoạt động chính xác và hiệu quả. Cho phép phản hồi các sự kiện quan trọng trong vòng đời ứng dụng iOS, chẳng hạn như khởi chạy, tạm ngưng và tiếp tục. Cung cấp điểm truy cập để cấu hình các dịch vụ nền tảng iOS và quản lý các tài nguyên ứng dụng.
- **Info.plist:** Tương tự như AndroidManifest.xml trong nền tảng Android, tệp này chứa cấu hình ứng dụng iOS như tên, phiên bản, biểu tượng, quyền yêu cầu và các cài đặt cấu hình khác. Được sử dụng bởi hệ thống iOS để xác định và quản lý ứng dụng.
- **Program.cs:** Điểm truy cập chính cho ứng dụng iOS. Tạo ra một đối tượng MauiApp bằng cách sử dụng lớp MauiProgram. Giúp cung cấp cầu nối giữa ứng dụng iOS và mã .NET MAUI đa nền tảng.

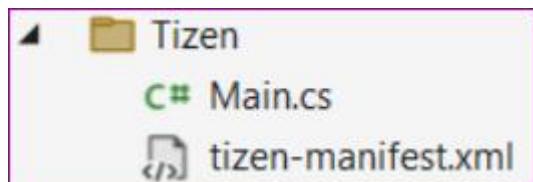
2.3.3.3. MacCatalyst



Hình 2.5: Cấu trúc của thư mục MacCatalyst

- **AppDelegate.cs:** Quản lý vòng đời ứng dụng, đóng vai trò quan trọng trong việc đảm bảo ứng dụng MacCatalyst hoạt động chính xác và hiệu quả. Cho phép phản hồi các sự kiện quan trọng trong vòng đời ứng dụng MacCatalyst, chẳng hạn như khởi chạy, tạm ngưng và tiếp tục. Cung cấp điểm truy cập để cấu hình các dịch vụ nền tảng MacCatalyst và quản lý các tài nguyên ứng dụng.
- **Info.plist:** Tương tự như AndroidManifest.xml trong nền tảng Android, tệp này chứa cấu hình ứng dụng MacCatalyst như tên, phiên bản, biểu tượng, quyền yêu cầu và các cài đặt cấu hình khác. Được sử dụng bởi hệ thống MacCatalyst để xác định và quản lý ứng dụng.
- **Program.cs:** Điểm truy cập chính cho ứng dụng MacCatalyst. Tạo ra một đối tượng MauiApp bằng cách sử dụng lớp MauiProgram. Giúp cung cấp cầu nối giữa ứng dụng MacCatalyst và mã .NET MAUI đa nền tảng.
- **Entitlements.plist:** Là tệp cấu hình quyền hạn cho ứng dụng chạy trên macOS trong project .NET MAUI. Nó cho phép ứng dụng truy cập vào các tài nguyên hệ thống và thực hiện các tác vụ nhất định như truy cập Keychain, Camera, vị trí,...

2.3.3.4. Tizen

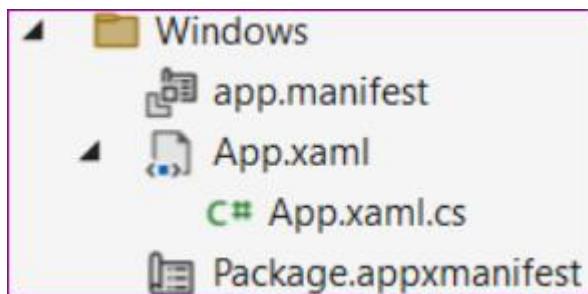


Hình 2.6: Cấu trúc của thư mục Tizen

- **Main.cs:** Điểm truy cập chính cho ứng dụng Tizen. Tạo ra một đối tượng MauiApp bằng cách sử dụng lớp MauiProgram. Giúp cung cấp cầu nối giữa ứng dụng Tizen và mã .NET MAUI đa nền tảng.

- **tizen-manifest.xml:** Tương tự như AndroidManifest.xml trong nền tảng Android, tệp này chứa cấu hình ứng dụng Tizen như tên, phiên bản, biểu tượng, quyền yêu cầu và các cài đặt cấu hình khác. Được sử dụng bởi hệ thống Tizen để xác định và quản lý ứng dụng.

2.3.3.5. Windows



Hình 2.7: Cấu trúc của thư mục Windows

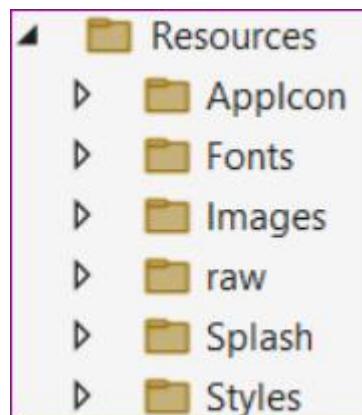
- **app.manifest:** Là thành phần bắt buộc cho mỗi gói ứng dụng Windows, chứa thông tin cần thiết cho hệ thống Windows để triển khai, hiển thị hoặc cập nhật ứng dụng. Bao gồm danh tính gói, phần phụ thuộc, khả năng cần thiết, hình ảnh và điểm mở rộng.
- **App.xaml:** Thiết kế giao diện người dùng cho cửa sổ chính của ứng dụng Windows.
- **App.xaml.cs:** Viết mã C# để điều khiển hành vi của cửa sổ chính và khởi tạo các thành phần ứng dụng khác.
- **Package.appxmanifest:** Mô tả và xác định các tập hợp song song được chia sẻ và riêng tư mà ứng dụng sẽ liên kết trong thời gian chạy. Chỉ rõ phiên bản lắp ráp được sử dụng để kiểm tra ứng dụng. Cung cấp siêu dữ liệu cho các tệp riêng tư của ứng dụng. Giúp hệ thống Windows quản lý các tài nguyên ứng dụng hiệu quả.

* *Cùng nhau, App.xaml và App.xaml.cs tạo thành các điểm truy cập chính cho ứng dụng Windows .NET MAUI.*

2.3.4. Thư mục Resources

Phát triển ứng dụng đa nền tảng thường gặp rào cản trong việc quản lý tài nguyên do mỗi nền tảng có yêu cầu riêng biệt. Ví dụ, hình ảnh cần được lưu trữ ở nhiều định dạng và độ phân giải khác nhau, dẫn đến sự lộn xộn và khó khăn trong việc theo dõi.

Single Project của .NET MAUI mang đến giải pháp hiệu quả cho vấn đề này. Nó cho phép lưu trữ tất cả các tệp tài nguyên như hình ảnh, font, biểu tượng... ở một vị trí duy nhất, đơn giản hóa việc tổ chức và truy cập. Hơn nữa, .NET MAUI sử dụng một tệp tài nguyên hình ảnh duy nhất làm nguồn và tự động chuyển đổi nó thành các phiên bản có độ phân giải phù hợp cho từng nền tảng khi xây dựng ứng dụng.



Hình 2.8: Cấu trúc của thư mục Resources

Thư mục Resources chứa các thư mục con khác nhau tương ứng với các loại tài nguyên khác nhau:

- **AppIcon:** Chứa các tệp biểu tượng ứng dụng được sử dụng cho nhiều mục đích khác nhau, bao gồm màn hình khởi động, thanh tiêu đề.
- **Fonts:** Chứa các tệp phông chữ tùy chỉnh được sử dụng trong giao diện người dùng của ứng dụng.
- **Images:** Chứa các tệp hình ảnh được sử dụng trong giao diện người dùng của ứng dụng, chẳng hạn như biểu tượng, ảnh minh họa và hình ảnh nền.

Với việc hình ảnh là thành phần thiết yếu trong mỗi ứng dụng, thư mục này thường chứa nhiều tệp nên việc quản lý tên tệp cần được ưu tiên.

- **raw:** Chứa các tệp dữ liệu thô, chưa được xử lý như HTML, JSON, CSV,...
- **Splash:** Chứa các tệp hình ảnh màn hình khi ứng dụng được khởi động hoặc khi đang tải, đóng vai trò như màn hình chờ. Raw: Chứa các tệp dữ liệu thô không được định dạng, chẳng hạn như tệp JSON, tệp XML, tệp cấu hình và tệp video.
- **Styles:** Chứa các tệp định dạng CSS được sử dụng để tạo kiểu cho các thành phần giao diện người dùng trong ứng dụng.

** Kích thước hình ảnh trong các thư mục AppIcon, Images và Splash được tự động điều chỉnh khi chạy ứng dụng để tương thích với từng nền tảng và thiết bị khác nhau.*

2.3.5. Code-behind

Code-behind là một phần quan trọng trong việc phát triển ứng dụng .NET MAUI. Nó đóng vai trò như lớp nền tảng cho giao diện người dùng (UI) XAML, cung cấp logic nghiệp vụ và xử lý các sự kiện cho ứng dụng.

Mối quan hệ giữa XAML và Code-behind:

- **Tệp XAML:** Định nghĩa giao diện người dùng của ứng dụng, bao gồm các thành phần UI như Button, Label, Image,...
- **Tệp Code-behind:** Cung cấp code C# để xử lý logic nghiệp vụ và các sự kiện cho ứng dụng. Tệp này được liên kết với tệp XAML tương ứng.

```
<Button
    x:Name="testBtn"
    BackgroundColor="#LightBlue"
    Clicked="testBtn_Clicked"
    HorizontalOptions="Fill"
    Text="Click me to see a random number"
    TextColor="White" />
```

Code xaml xác định giao diện của nút testBtn

```
private void testBtn_Clicked(object sender, EventArgs e)
{
    // Create a Random instance
    Random random = new Random();
    // Generate a random number between -100 and 100
    int randomNumber = random.Next(-100, 101);
    // Set the button text to the random number
    testBtn.Text = randomNumber.ToString();
}
```

Code-behind là C# xác định sự kiện cho nút testBtn

Hình 2.9: Code minh họa cho Code-behind

Lợi ích của Code-behind:

- Dễ dàng viết mã logic nghiệp vụ:** Code-behind cung cấp môi trường thuận lợi để viết mã logic nghiệp vụ cho ứng dụng.
- Kiểm soát chặt chẽ giao diện người dùng:** Code-behind cho phép truy cập và thao tác trực tiếp các thành phần UI, giúp kiểm soát giao diện người dùng chặt chẽ hơn.

Hạn chế của Code-behind:

- Khó bảo trì:** Khi ứng dụng phát triển về quy mô, việc bảo trì code-behind có thể trở nên phức tạp do sự kết hợp chặt chẽ giữa logic nghiệp vụ và giao diện người dùng.
- Khó khăn trong việc kiểm tra đơn vị:** Việc kiểm tra đơn vị logic nghiệp vụ trong code-behind có thể khó khăn do sự phụ thuộc vào các thành phần UI cụ thể.
- Thay đổi giao diện người dùng tốn kém:** Do logic nghiệp vụ được nhúng trong code-behind, việc thay đổi giao diện người dùng có thể dẫn đến thay đổi mã logic, tăng chi phí phát triển.

2.4. App Lifecycle

Vòng đời ứng dụng (App Lifecycle) là một khái niệm quan trọng trong lập trình ứng dụng, mô tả các trạng thái khác nhau mà một ứng dụng trải qua trong suốt

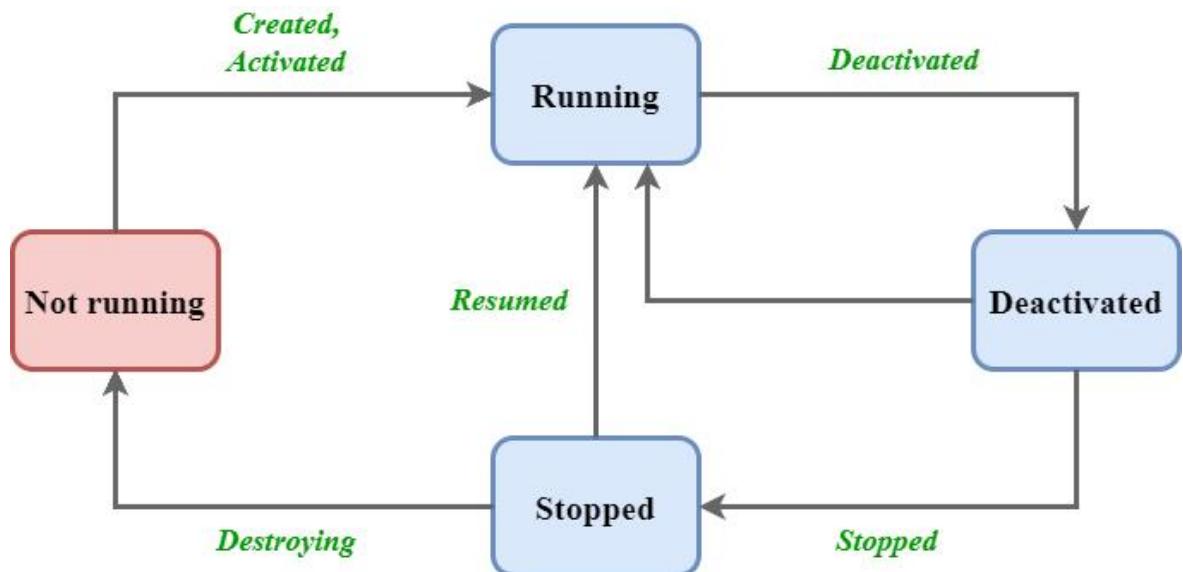
thời gian hoạt động. Hiểu rõ vòng đời ứng dụng giúp nhà phát triển viết code hiệu quả, quản lý tài nguyên và tạo trải nghiệm người dùng tốt hơn.

Các ứng dụng .NET MAUI trải qua bốn trạng thái chính trong suốt vòng đời của chúng:

- 1) **Not running:** Ứng dụng chưa được tải vào bộ nhớ và không hoạt động.
- 2) **Running:** Ứng dụng đã được tải vào bộ nhớ và đang hoạt động trên thiết bị.
- 3) **Deactivated:** Ứng dụng vẫn đang trong bộ nhớ nhưng không hoạt động ở chế độ nền.
- 4) **Stopped:** Ứng dụng đã được gỡ khỏi bộ nhớ và không còn hoạt động.

Mỗi ứng dụng .NET MAUI sẽ chuyển đổi qua bốn trạng thái trên và sẽ kích hoạt các sự kiện vòng đời (lifecycle event) xảy ra khi chuyển đổi trạng thái.

* Để có thể truy cập vào các sự kiện vòng đời, ta phải truy cập thông qua lớp *Window*, vì .NET MAUI Framework đã nền tảng nên cách truy cập thông qua lớp *Window* phù hợp với cả máy tính để bàn và thiết bị di động.



Sơ đồ 2.2: Mô tả App Lifecycle

Các sự kiện vòng đời bao gồm:

- **Created:** Sự kiện được kích hoạt khi cửa sổ ứng dụng được tạo, quá trình xử lý của cửa sổ đã được thiết lập nhưng có thể chưa hiện thị.
- **Activated:** Sự kiện được kích hoạt khi cửa sổ ứng dụng đã được kích hoạt và trở thành cửa sổ đang được lấy tiêu điểm (focus - trạng thái đang được người dùng tương tác). Ở sự kiện này, các hoạt động ban đầu của ứng dụng thường được bắt đầu, chẳng hạn như tải dữ liệu hoặc khởi tạo giao diện người dùng.
- **Deactivated:** Sự kiện này được kích hoạt khi cửa sổ ứng dụng không còn là cửa sổ được lấy tiêu điểm, tuy nhiên cửa sổ vẫn có thể còn được hiện thị.
- **Stopped:** Sự kiện này được kích hoạt khi cửa sổ ứng dụng không còn hiển thị nữa, ứng dụng có thể bị chấm dứt để giải phóng tài nguyên.
- **Resumed:** Sự kiện này được kích hoạt khi ứng dụng được tiếp tục hoạt động sau khi đã dừng, chỉ xảy ra khi sự kiện Stopped đã được kích hoạt trước đó. Có thể sử dụng sự kiện này để khôi phục trạng thái ứng dụng và tiếp tục các hoạt động đã bị tạm dừng.
- **Destroying:** Sự kiện này được kích hoạt khi cửa sổ gốc của ứng dụng đang bị hủy và giải phóng tài nguyên.

.NET MAUI cung cấp cho ta các sự kiện vòng đời để theo dõi các trạng thái của ứng dụng trên nhiều nền tảng, tuy nhiên nếu hiểu được sự kiện nào được kích hoạt và cách xử lý chúng trên từng nền tảng riêng biệt sẽ giúp ta dễ dàng chuẩn đoán cũng như khắc phục các sự cố liên qua đến sự kiện vòng đời của ứng dụng.

Bảng 2.1: Các sự kiện vòng đời được ánh xạ đến các nền tảng khác nhau

Sự kiện	Android	iOS	Windows
Created	OnPostCreate	FinishedLaunching	Created
Activated	OnResume	OnActivated	Activated (CodeActivated và PointerActivated)
Deactivated	OnPause	OnResignActivation	Activated (Deactivated)
Stopped	OnStop	DidEnterBackground	VisibilityChanged
Resumed	OnRestart	WillEnterForeground	Resumed
Destroying	OnDestroy	WillTerminate	Closed

Việc sử dụng các sự kiện vòng đời ứng dụng cho phép nhà phát triển:

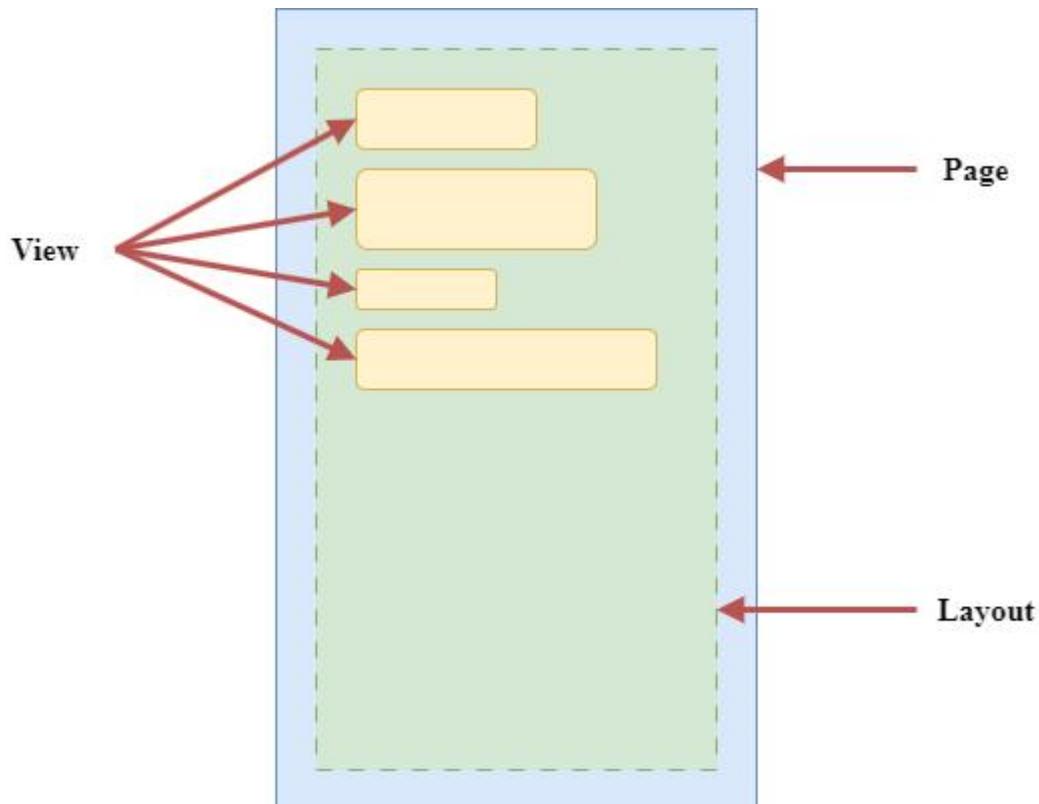
- **Lưu trạng thái ứng dụng:** Khi ứng dụng bị Deactivated hoặc Stopped, nhà phát triển có thể lưu trạng thái hiện tại của ứng dụng để phục hồi sau.
- **Giải phóng tài nguyên:** Khi ứng dụng bị Deactivated hoặc Stopped, nhà phát triển có thể giải phóng các tài nguyên hệ thống không cần thiết để tiết kiệm pin và cải thiện hiệu suất.
- **Thực hiện các tác vụ nền:** Ứng dụng có thể thực hiện các tác vụ nền như tải dữ liệu hoặc đồng bộ hóa khi bị Deactivated.

CHƯƠNG 3: THIẾT KẾ GIAO DIỆN NGƯỜI DÙNG VỚI XAML TRONG .NET MAUI

Giao diện người dùng của một ứng dụng .NET MAUI được tạo ra từ các phần tử tương ứng với các control khác nhau trên mỗi nền tảng mục tiêu.

Các nhóm control chính được sử dụng để tạo giao diện người dùng trong các ứng dụng .NET MAUI bao gồm các thành phần hiện thị (view), bố cục (layout) và các trang (page).

Một trang trong ứng dụng thường chiếm toàn bộ màn hình hoặc cửa sổ, mỗi trang thường chứa ít nhất một bố cục. Bố cục cho phép sắp xếp vị trí và nhóm các thành phần hiện thị, một bố cục có thể chứa các bố cục khác để tạo ra một bố cục mới. Thành phần hiện thị là các phần tử UI như các nút bấm, các nhãn,... là hiện thị chính và tương tác với người dùng.



Hình 3.1: Minh họa các control trong giao diện người dùng

3.1. Alignment và vị trí

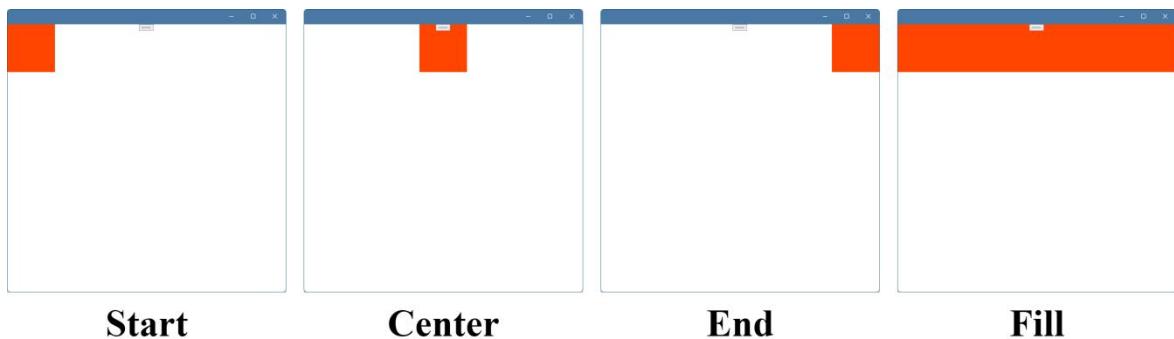
Các view và layout nằm trong layout cha có thể được căn chỉnh (align), xác định vị trí (position) và kích thước trong trường hợp layout cha có dư khoảng trống - layout cha có kích thước lớn hơn tổng kích thước của các view và layout con nằm trong nó.

3.1.1. Alignment

Các view và layout có các thuộc tính ***HorizontalOptions*** và ***VerticalOptions*** để căn chỉnh theo lần lượt theo chiều ngang và chiều dọc.

Đối với thuộc tính căn chỉnh theo chiều ngang ***HorizontalOptions***:

- ***Start***: Xác định vị trí view ở phía bên trái của layout cha.
- ***Center***: Xác định vị trí view ở giữa theo chiều ngang của layout cha.
- ***End***: Xác định vị trí view ở phía bên phải của layout cha.
- ***Fill*** (mặc định): Xác định view sẽ lấp đầy chiều rộng (width) của layout cha.

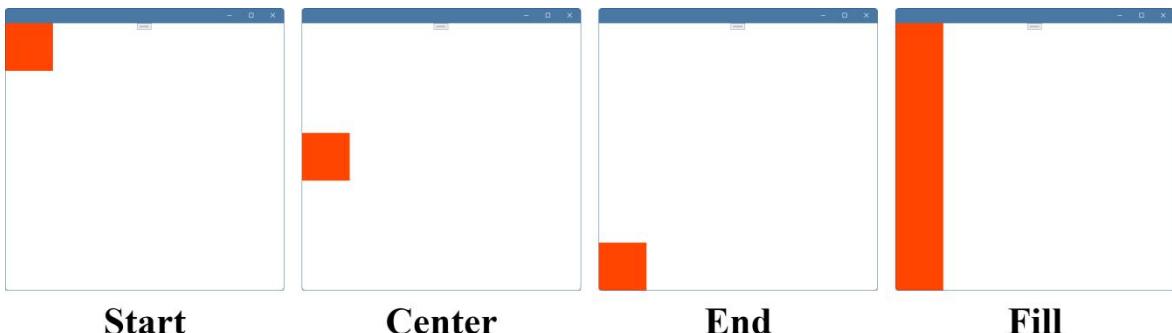


Hình 3.2: View được căn chỉnh bằng ***HorizontalOptions***

Đối với thuộc tính căn chỉnh theo chiều dọc ***VerticalOptions***:

- ***Start***: Xác định vị trí view ở phía bên trên của layout cha.
- ***Center***: Xác định vị trí view ở giữa theo chiều dọc của layout cha.

- **End:** Xác định vị trí view ở phía bên dưới của layout cha.
- **Fill** (mặc định): Xác định view sẽ lấp đầy chiều cao (height) của layout cha.



Hình 3.3: View được căn chỉnh bằng VerticalOptions

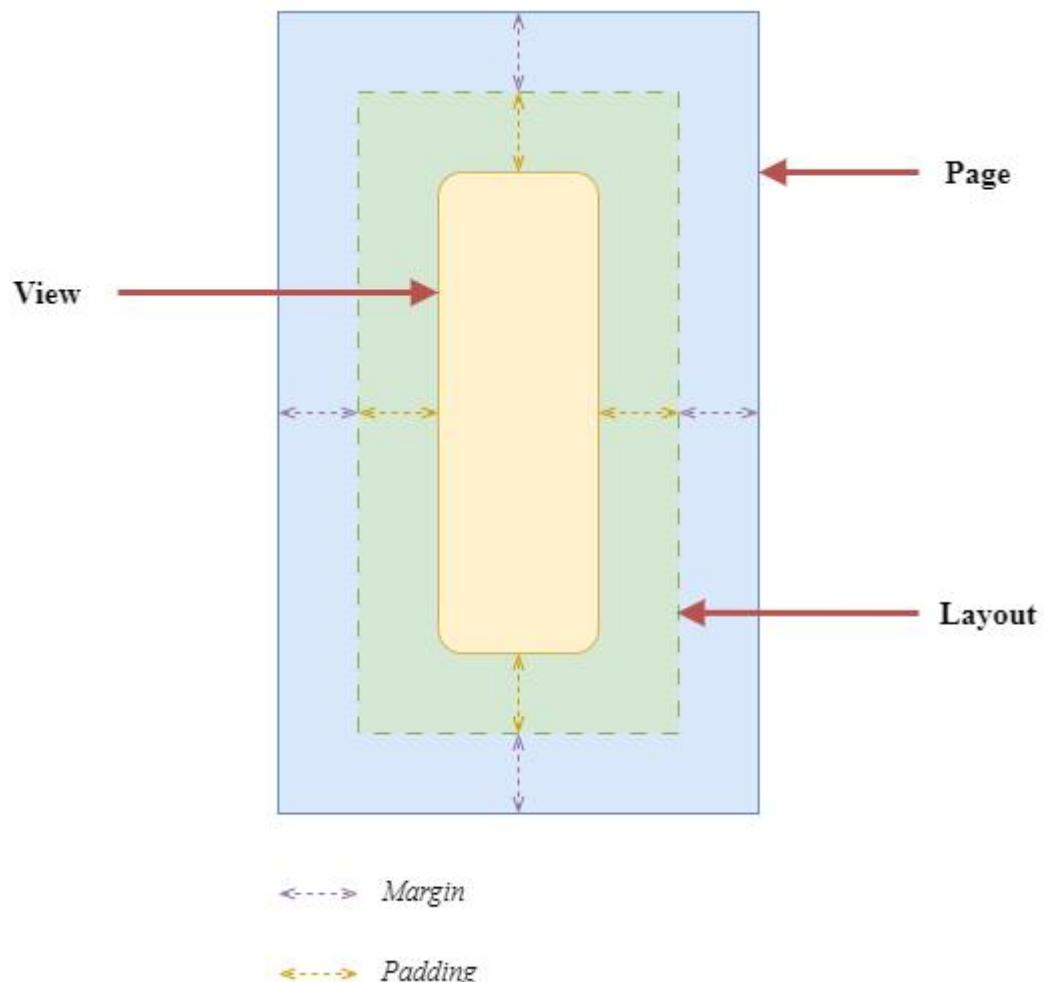
* Kích thước của view từ **HeightRequest** và **WidthRequest** sẽ bị ghi đè (override) nếu như sử dụng **Fill**.

3.1.2. Điều chỉnh vị trí

Margin và **Padding** là hai thuộc tính dùng để điều chỉnh vị trí của các view và layout liền kề nhau hoặc liên quan đến việc điều chỉnh các control con.

- **Margin:** Thể hiện khoảng cách giữa một phần tử và các phần tử lân cận với nó, qua đó giúp điều chỉnh vị trí của phần tử cũng như các phần tử lân cận. Dùng cho view và layout.
- **Padding:** Thể hiện khoảng cách giữa một phần tử và các phần tử con của nó, qua đó giúp tách control khỏi các control con hoặc nội dung của nó. Dùng cho view, layout và page.

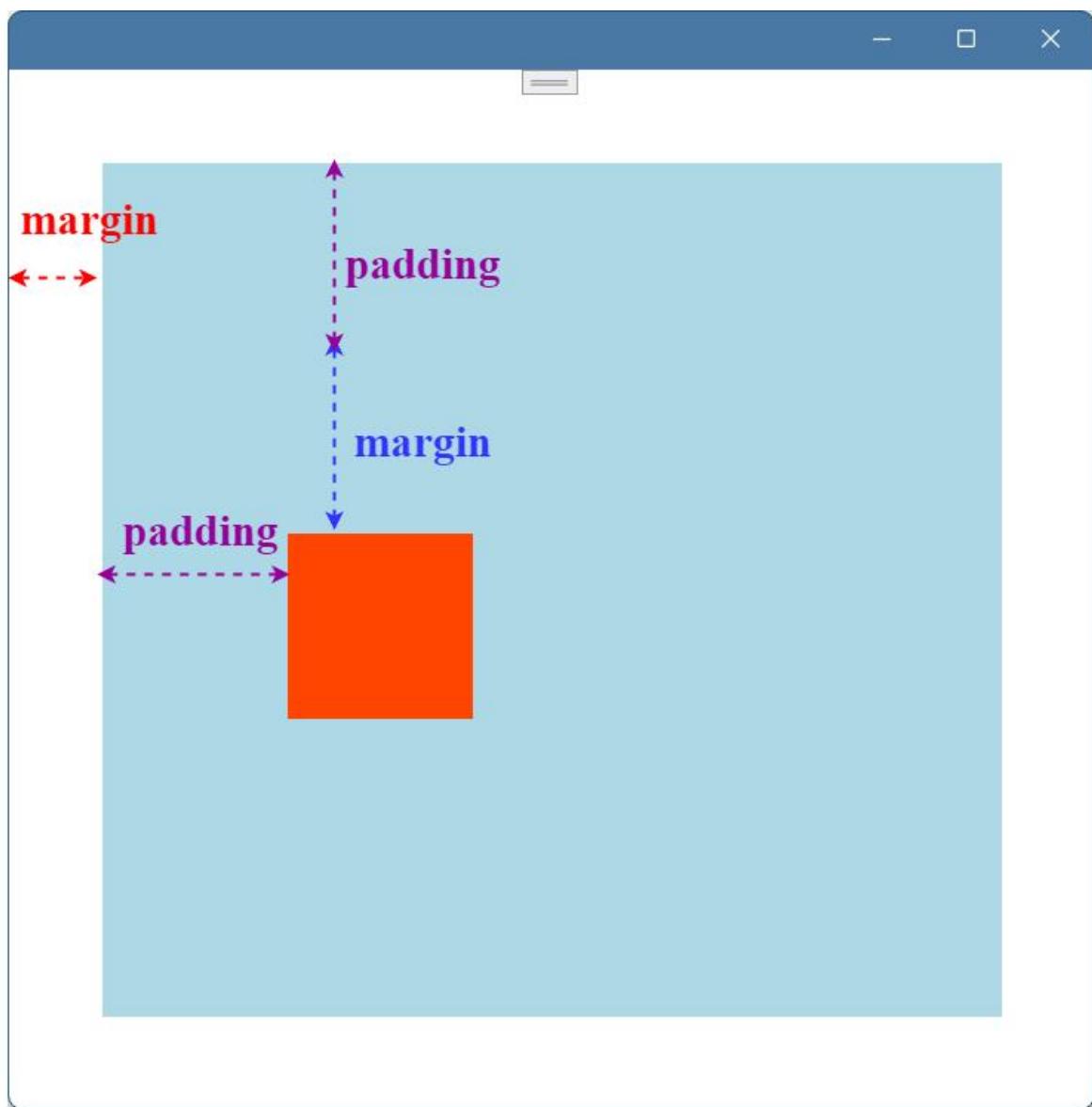
* **Margin** và **Padding** có tính cộng dồn nếu được áp dụng với các phần tử liền kề nhau và khi được áp dụng đồng thời.



Hình 3.4: Minh họa margin và padding

Margin và Padding là các thuộc tính loại Thickness (độ dày):

- **Nếu được xác định bởi một giá trị:** Giá trị này sẽ được áp dụng cho các cạnh trái, trên, phải và dưới của phần tử.
- **Nếu được xác định bởi hai giá trị:** Giá trị đầu tiên sẽ được áp dụng cho cạnh trái và phải của phần tử, giá trị thứ hai sẽ được áp dụng cho cạnh trên và dưới của phần tử.
- **Nếu được xác định bởi bốn giá trị:** Các giá trị lần lượt sẽ được áp dụng cho cạnh trái, trên, phải và dưới của phần tử.



Hình 3.5: Margin và padding

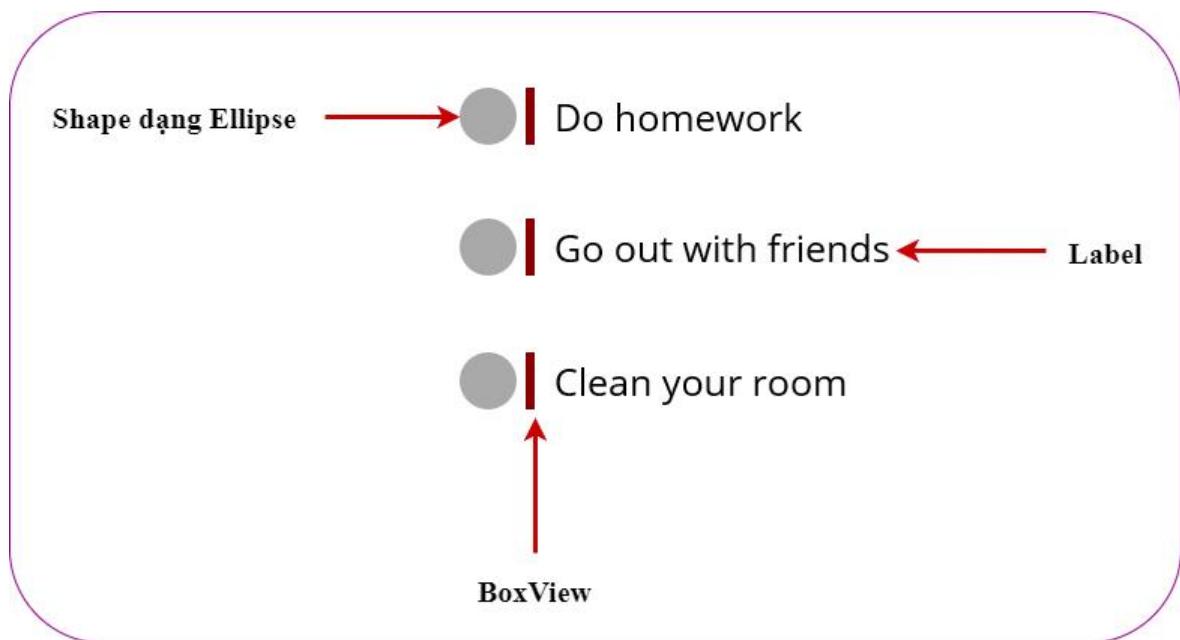
3.2. View

Với hơn 30 view khác nhau, .NET MAUI cung cấp một kho tàng công cụ mạnh mẽ để xây dựng giao diện người dùng đa nền tảng. Mỗi view sở hữu chức năng hiển thị riêng biệt, nhưng nhiều view lại có chung mục đích hoặc cách thức hoạt động tương đồng. Việc nhóm các view này theo chức năng giúp dễ dàng lựa chọn và sử dụng view phù hợp cho từng trường hợp cụ thể trong ứng dụng.

3.2.1. Hiện thị dữ liệu

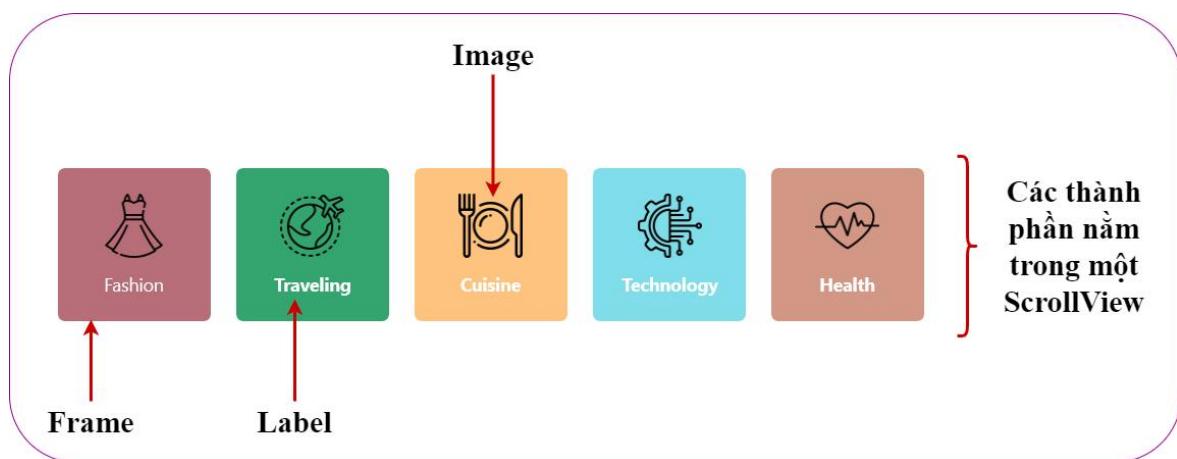
Nhóm view này cung cấp các công cụ mạnh mẽ để hiển thị nhiều loại dữ liệu cho người dùng bao gồm cả dữ liệu trực quan trong ứng dụng .NET MAUI:

- **Label:** Hiển thị văn bản đơn giản, định dạng cơ bản, là một trong những view phổ biến nhất, dùng để hiển thị tiêu đề, nhãn, thông tin,... Có thể tùy chỉnh phông chữ, màu sắc, kích thước, căn chỉnh và các thuộc tính khác.
- **BoxView:** Hiển thị hình nền màu sắc hoặc hình ảnh đơn giản, thường dùng để tạo nền cho các view khác hoặc tạo hiệu ứng thị giác. Có thể tùy chỉnh màu sắc, hình ảnh, độ mờ, góc tròn và các thuộc tính khác.
- **Shapes:** Hiển thị các hình dạng vectơ 2D như hình vuông, hình tròn, đường thẳng,... Dùng để tạo các biểu tượng, đồ thị, trang trí và các thành phần giao diện tùy chỉnh. Có thể tùy chỉnh màu sắc, độ dày đường viền, kiểu tô và các thuộc tính khác.



Hình 3.6: Shape, Label và BoxView

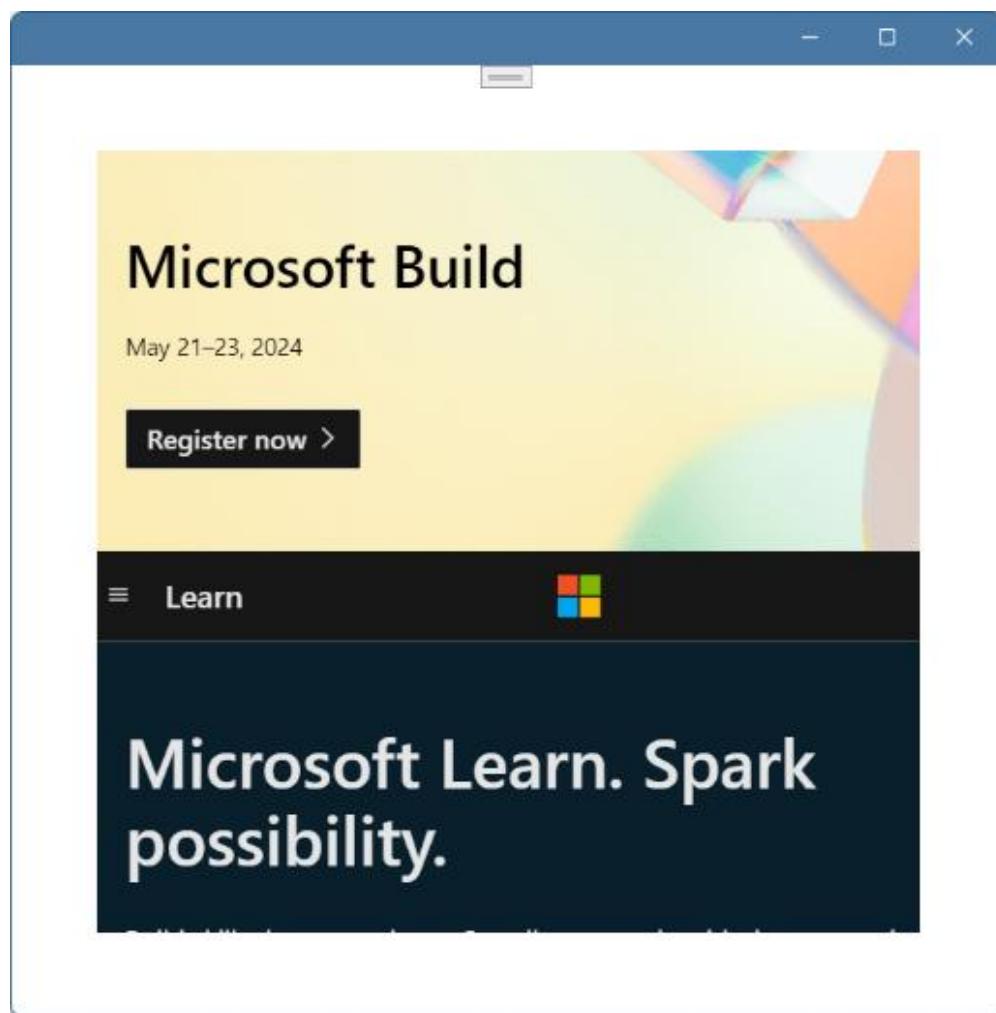
- **Frame:** Hiển thị nội dung của một view khác trong một khung có viền và bóng đổ, tạo hiệu ứng 3D, giúp phân biệt, và làm nổi bật các phần nội dung trong giao diện. Có thể tùy chỉnh màu sắc, độ dày viền, bóng đổ và các thuộc tính khác.
- **Image:** Hiển thị hình ảnh tĩnh từ tệp tin hoặc URL, dùng để hiển thị ảnh sản phẩm, logo, avatar, ảnh bìa,... Có thể tùy chỉnh kích thước, tỷ lệ khung hình, chế độ lấp đầy và các thuộc tính khác.



Hình 3.7: Frame, Label và Image

- **GraphicsView:** Hiển thị đồ họa 2D, dùng để tạo các biểu đồ, hình ảnh động, hiệu ứng đặc biệt và các nội dung đồ họa phức tạp. Cung cấp hiệu suất cao và khả năng tùy chỉnh mạnh mẽ.
- **Map:** Hiển thị bản đồ tương tác với các vị trí, đường đi và thông tin địa lý, tích hợp với các dịch vụ bản đồ như Bing Maps hoặc Mapbox. Cho phép người dùng tìm kiếm địa điểm, đặt dấu mốc, theo dõi tuyến đường và thực hiện các thao tác khác.
- **ScrollView:** Cho phép cuộn nội dung của một view khi nội dung vượt quá kích thước màn hình, dùng để hiển thị danh sách dài, nội dung văn bản nhiều,... Có thể tùy chỉnh hướng cuộn, thanh cuộn và các thuộc tính khác.

- **WebView:** Hiển thị nội dung web từ URL bên trong ứng dụng .NET MAUI, dùng để tích hợp các trang web, ứng dụng web và các nội dung trực tuyến khác. Hỗ trợ JavaScript, HTML và CSS.



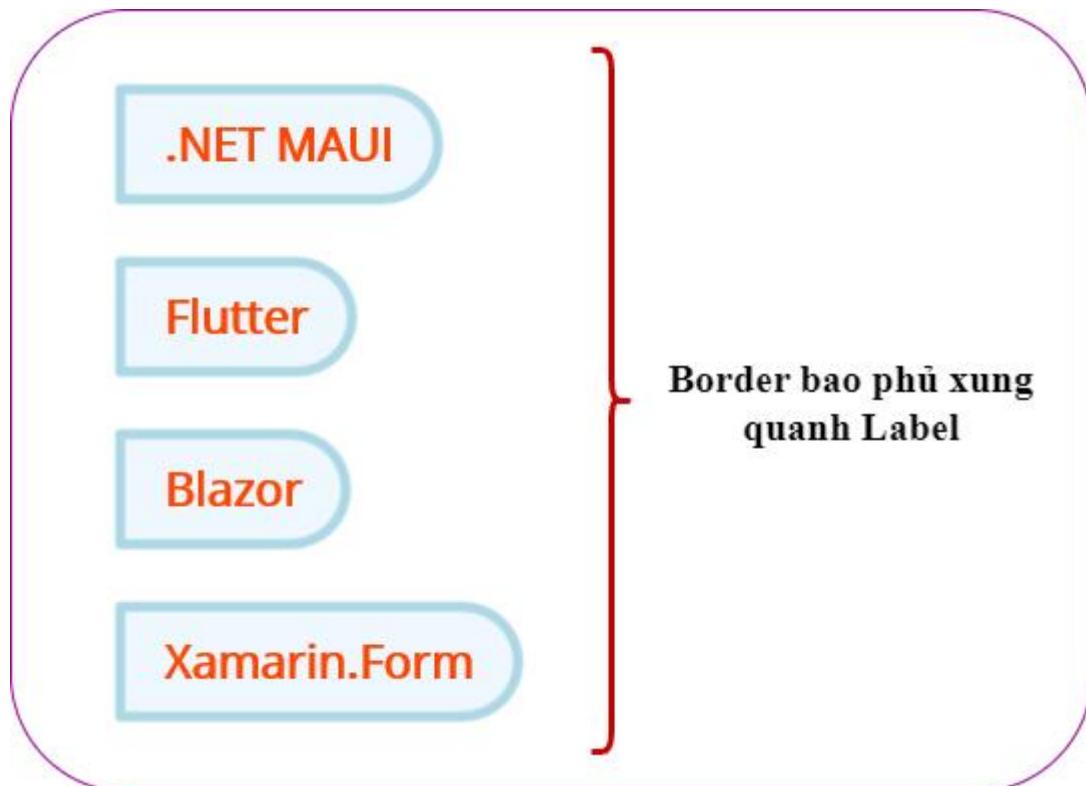
WebView với nguồn là trang chủ của Microsoft Learn

Hình 3.8: WebView

- **BlazorWebView:** Hiển thị nội dung web được tạo bằng Blazor, Framework lập trình web dựa trên .NET. Tích hợp liền mạch với các view .NET MAUI khác, cho phép xây dựng ứng dụng .NET MAUI Blazor Hybrid (ứng dụng lai giữa .NET MAUI và Blazor) mạnh mẽ.

- **Border:** Vẽ đường viền xung quanh các view khác, tạo điểm nhấn và phân chia bố cục. Có thể tùy chỉnh màu sắc, độ dày, kiểu đường viền và các thuộc tính khác.

* *ScrollView có thể không hoạt động nếu được sử dụng lồng trong nhiều control khác.*



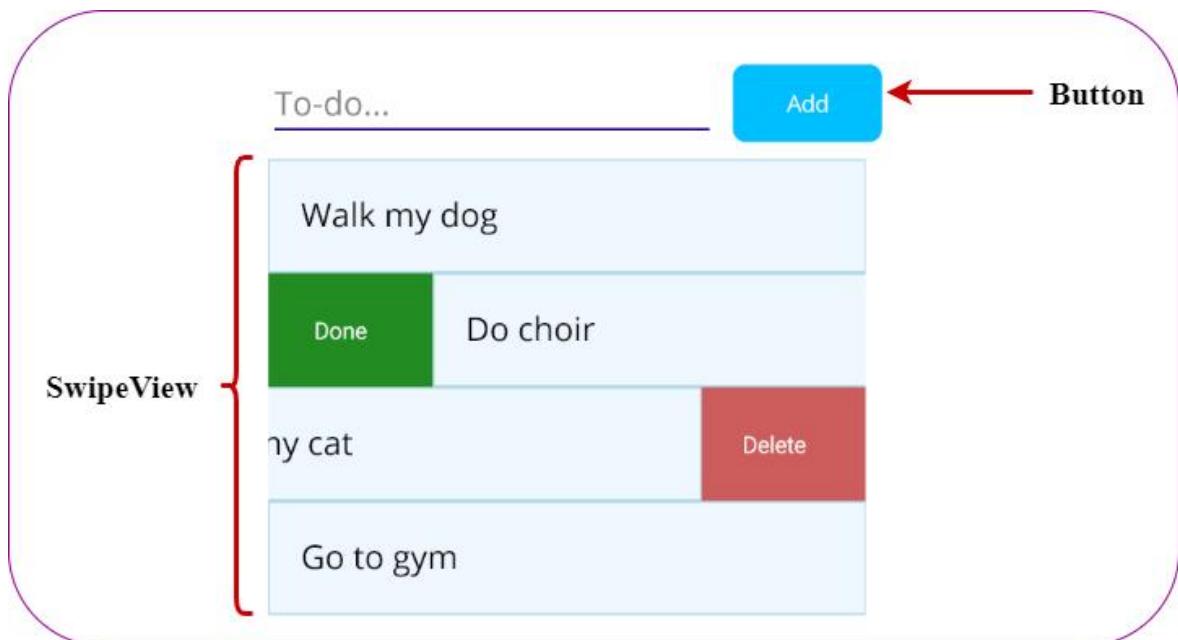
Hình 3.9: Border

3.2.2. Khởi tạo lệnh

Nhóm view này cung cấp các công cụ để người dùng tương tác với ứng dụng và kích hoạt hành động, sự kiện:

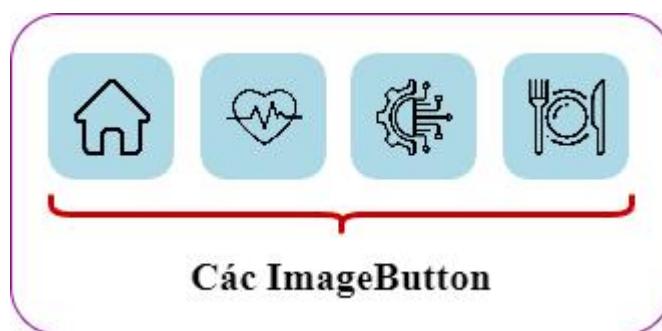
- **Button:** Nút bấm để người dùng thực hiện hành động khi nhấp chuột hoặc chạm, dùng để tính toán, mở màn hình mới, gửi dữ liệu hoặc thực hiện các thao tác khác. Có thể tùy chỉnh văn bản, hình ảnh, màu sắc, kích thước và các thuộc tính khác.

- **SwipeView:** Cho phép người dùng vuốt sang trái hoặc sang phải để mở lộ nội dung ẩn hoặc thực hiện hành động, dùng để tạo menu ẩn, hiển thị thêm thông tin hoặc thay đổi trạng thái của một view. Có thể tùy chỉnh hướng vuốt, nội dung ẩn và các thuộc tính khác.



Hình 3.10: Button và SwipeView

- **ImageButton:** Nút bấm với hình ảnh thay vì văn bản. Chức năng tương tự như Button nhưng giúp truyền tải thông tin một cách trực quan hơn và thu hút sự chú ý của người dùng trong nhiều trường hợp nhất định. Có thể tùy chỉnh hình ảnh, màu sắc, kích thước và các thuộc tính khác.



Hình 3.11: ImageButton

- **RadioButton:** Nút radio cho phép người dùng lựa chọn một tùy chọn từ nhiều lựa chọn có sẵn, dùng để thu thập thông tin đầu vào hoặc cài đặt cấu hình. Chỉ có thể chọn một RadioButton trong một nhóm.



Hình 3.12: RadioButton

- **RefreshView:** Cho phép người dùng tải lại dữ liệu trong ứng dụng bằng cách kéo xuống, dùng để cập nhật dữ liệu thường xuyên hoặc tải nội dung mới. Có thể tùy chỉnh hiệu ứng kéo xuống, màu sắc và các thuộc tính khác. Chỉ có thể dùng cho các phần tử con có thể cuộn được như ScrollView, CollectionView hay ListView.
- **SearchBar:** Thanh tìm kiếm cho phép người dùng tìm kiếm thông tin trong ứng dụng, dùng để lọc dữ liệu, tìm kiếm sản phẩm, tìm kiếm nội dung hoặc thực hiện các thao tác tìm kiếm khác. Có thể tùy chỉnh placeholder, văn bản nút tìm kiếm, kiểu dáng và các thuộc tính khác.

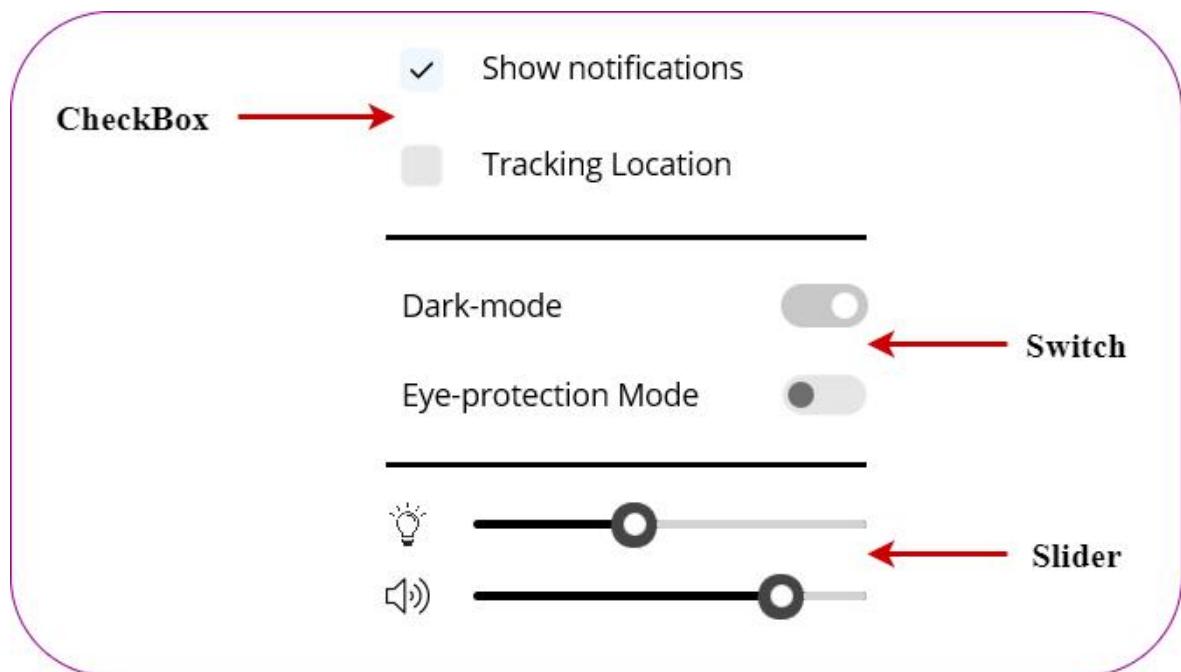


Hình 3.13: SearchBar

3.2.3. Cài đặt giá trị

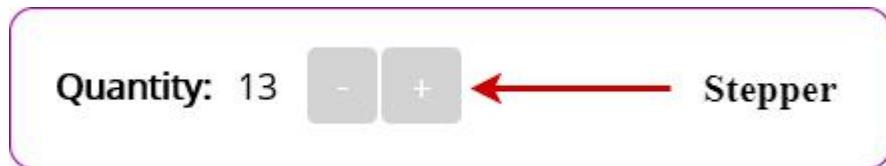
Nhóm view này cung cấp các công cụ để người dùng nhập và chọn giá trị cho các thuộc tính và cài đặt trong ứng dụng:

- **CheckBox:** Ô chọn cho phép người dùng bật/tắt một tùy chọn, dùng để thu thập thông tin đầu vào hoặc cài đặt cấu hình. Có thể chọn nhiều CheckBox trong cùng một nhóm.
- **Slider:** Thanh trượt cho phép người dùng chọn giá trị trong một phạm vi xác định, dùng để điều chỉnh cài đặt, chọn mức độ, nhập giá trị số hoặc thực hiện các thao tác chọn giá trị khác. Có thể tùy chỉnh phạm vi giá trị, bước nhảy, kiểu hiển thị và các thuộc tính khác.
- **Switch:** Công tắc cho phép người dùng bật/tắt một tính năng, dùng để cài đặt cấu hình, quản lý quyền truy cập hoặc bật/tắt các tính năng khác. Có thể tùy chỉnh nhãn, màu sắc và các thuộc tính khác.



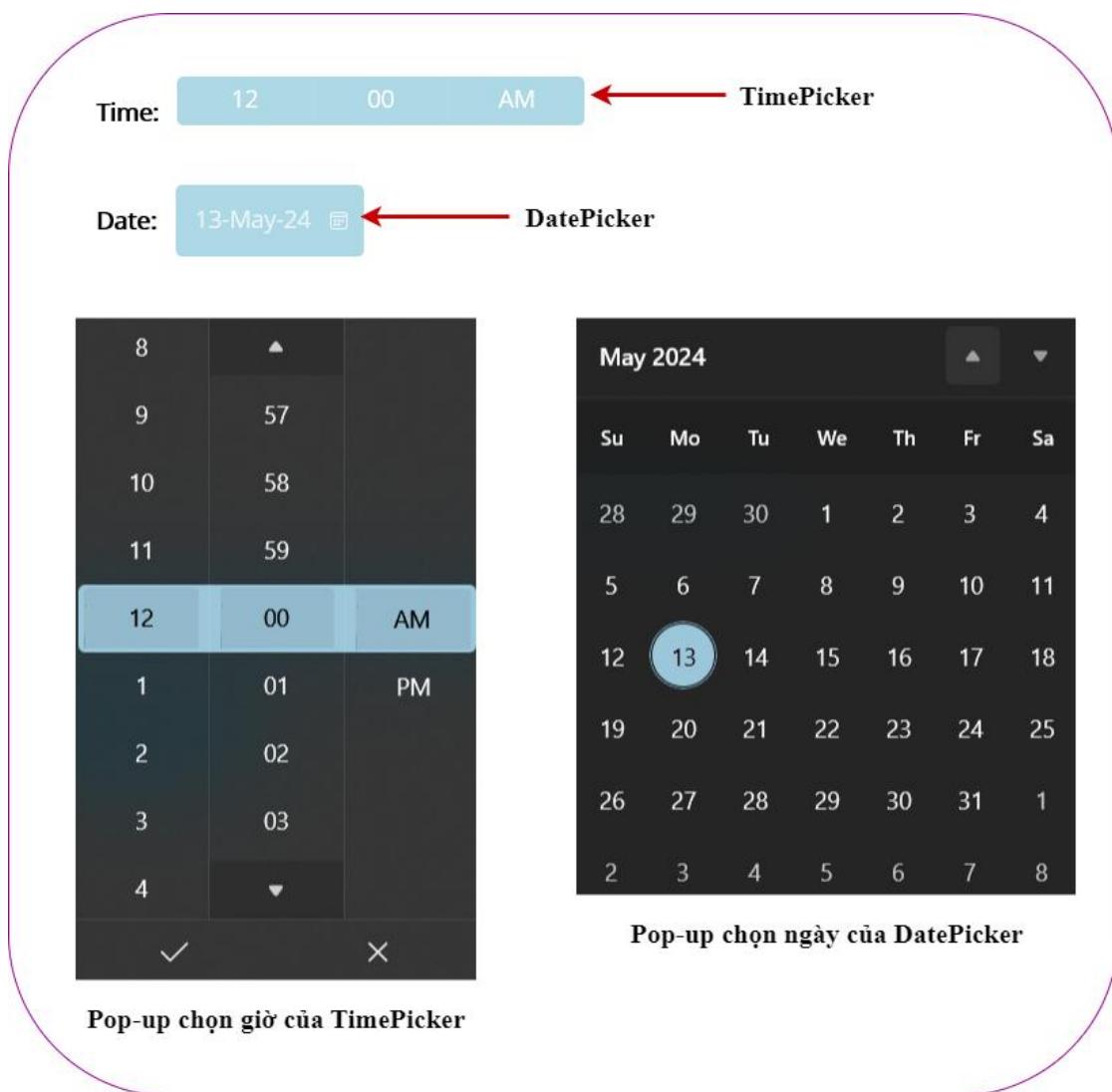
Hình 3.14: CheckBox, Switch và Slider

- **Stepper:** Bộ tăng/giảm cho phép người dùng tăng hoặc giảm giá trị từng bước, bao gồm 2 nút với nhãn “+” và “-”, dùng để nhập số lượng, điều chỉnh giá trị hoặc thực hiện các thao tác tăng/giảm giá trị khác. Có thể tùy chỉnh bước nhảy, giá trị tối thiểu và tối đa, và các thuộc tính khác.



Hình 3.15: Stepper

- **TimePicker:** Bộ chọn thời gian cho phép người dùng chọn giờ và phút, dùng để thu thập thông tin giờ hẹn, lên lịch, đặt thời gian báo thức hoặc thực hiện các thao tác chọn thời gian khác. Có thể tùy chỉnh định dạng thời gian, kiểu hiển thị đồng hồ và các thuộc tính khác.
- **DatePicker:** Bộ chọn ngày cho phép người dùng chọn ngày tháng năm, dùng để thu thập thông tin ngày sinh, lịch hẹn, ngày tháng quan trọng hoặc thực hiện các thao tác chọn ngày khác. Có thể tùy chỉnh định dạng ngày, kiểu hiển thị lịch và các thuộc tính khác.



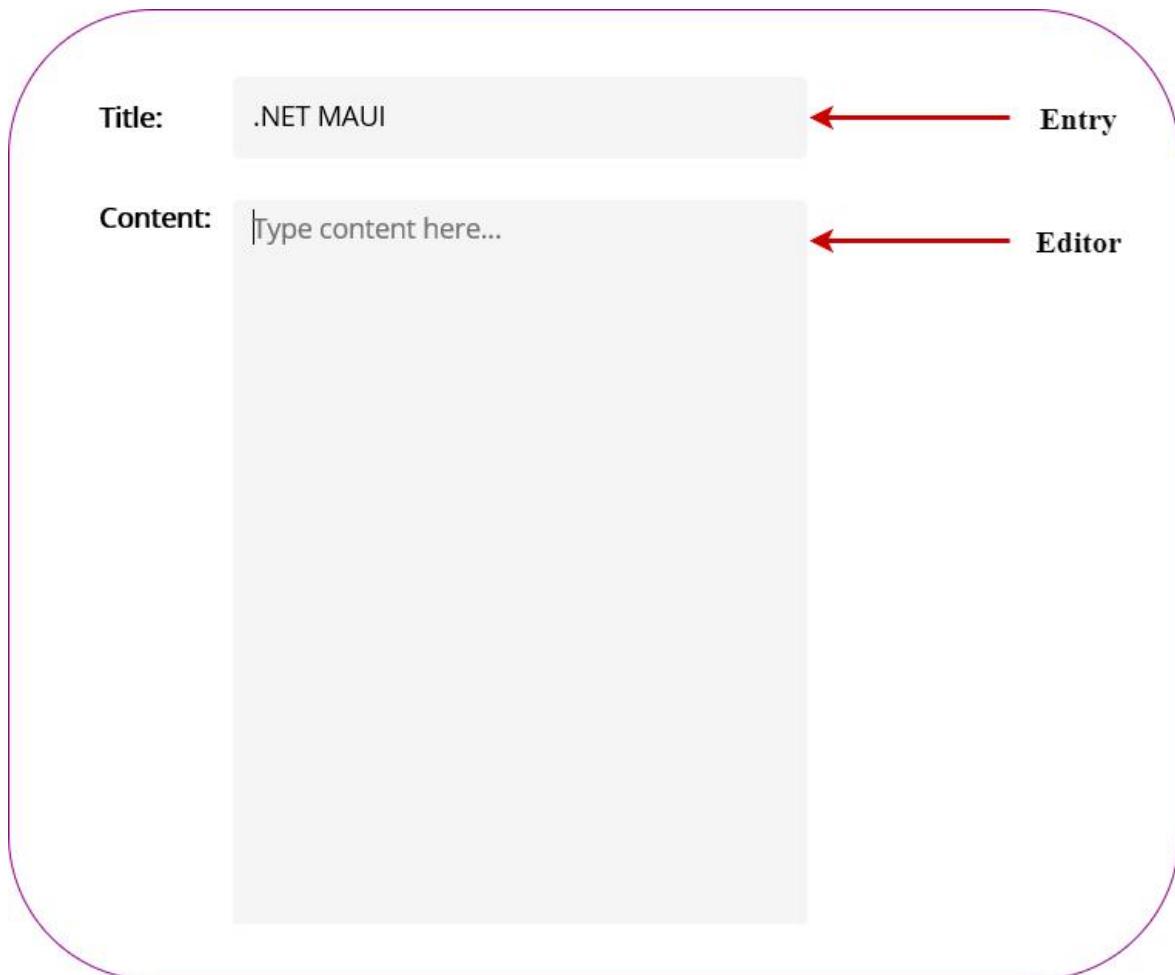
Hình 3.16: TimePicker và DatePicker

3.2.4. Chính sửa văn bản

Nhóm view này cung cấp các công cụ để người dùng nhập, chỉnh sửa và quản lý văn bản trong ứng dụng:

- **Entry:** Ô nhập văn bản đơn giản cho phép người dùng nhập văn bản một dòng, dùng để thu thập thông tin đầu vào, nhập tên, nhập email, nhập địa chỉ hoặc thực hiện các thao tác nhập văn bản cơ bản khác. Có thể tùy chỉnh kích thước, kiểu chữ, màu sắc, placeholder và các thuộc tính khác.

- **Editor:** Trình chỉnh sửa văn bản đa dòng cho phép người dùng nhập, chỉnh sửa và định dạng văn bản, dùng để viết ghi chú, soạn thảo email, tạo nội dung hoặc thực hiện các thao tác chỉnh sửa văn bản phức tạp hơn. Hỗ trợ định dạng văn bản cơ bản như in đậm, in nghiêng, gạch chân, căn chỉnh,... và tùy chỉnh thanh công cụ, phím tắt và các thuộc tính khác.

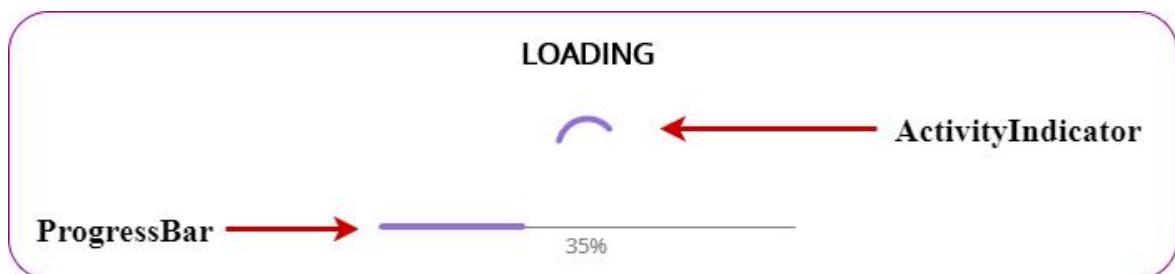


Hình 3.17: Entry và Editor

3.2.5. Biểu thị hoạt động

Nhóm view này cung cấp các công cụ để thể hiện trạng thái hoạt động và cung cấp phản hồi cho người dùng khi ứng dụng đang thực hiện thao tác hoặc tải dữ liệu:

- **ActivityIndicator:** Biểu tượng hoạt động xoay tròn cho biết ứng dụng đang thực hiện thao tác và sẽ hoàn thành trong thời gian ngắn (tương tự như biểu thị loading - đang tải), dùng để thể hiện tiến trình tải dữ liệu, thực hiện thao tác hoặc chờ đợi phản hồi từ máy chủ. Có thể tùy chỉnh kích thước, màu sắc, kiểu hoạt động và các thuộc tính khác.
- **ProgressBar:** Thanh tiến trình cho biết tỷ lệ hoàn thành của một thao tác hoặc quá trình tải dữ liệu, dùng để thể hiện tiến trình tải file, cập nhật phần mềm hoặc thực hiện các thao tác tốn thời gian khác. Có thể tùy chỉnh màu sắc, kiểu hiển thị, giá trị tiến trình và các thuộc tính khác.



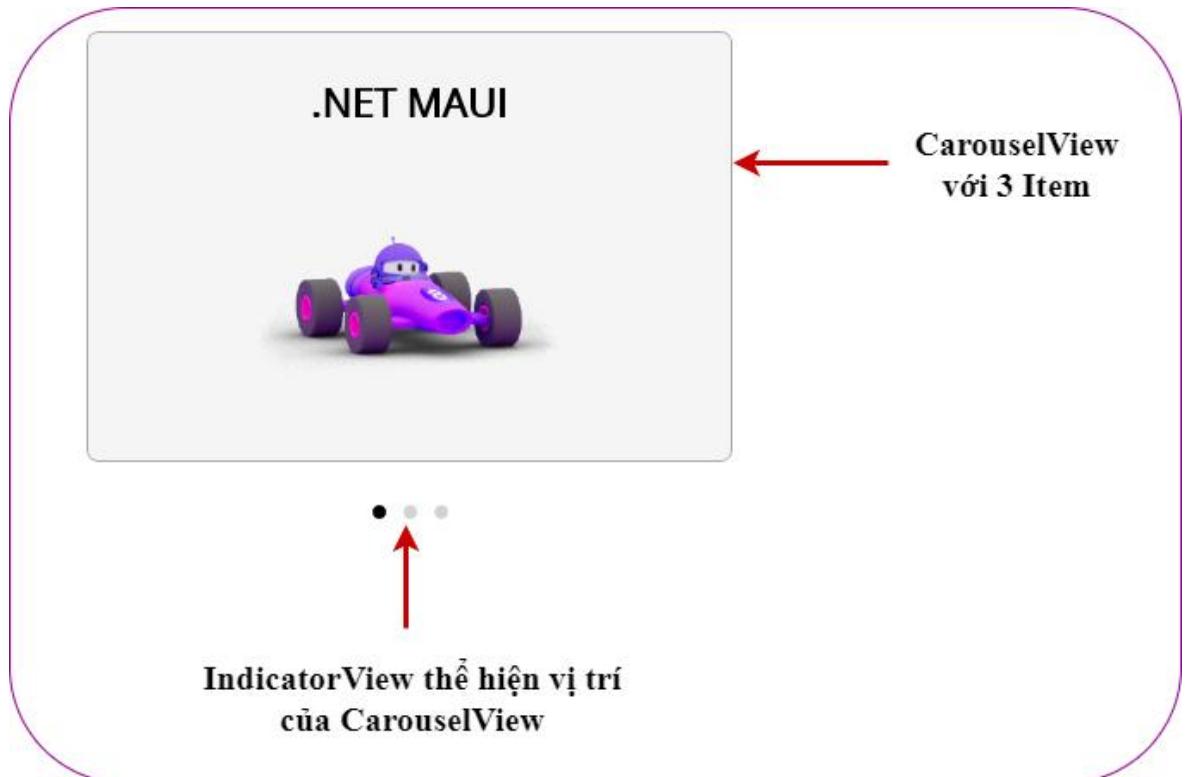
Hình 3.18: ActivityIndicator và ProgressBar

3.2.6. Hiển thị tập hợp

Nhóm view này cung cấp các công cụ mạnh mẽ để hiển thị và quản lý các tập hợp dữ liệu trong ứng dụng .NET MAUI:

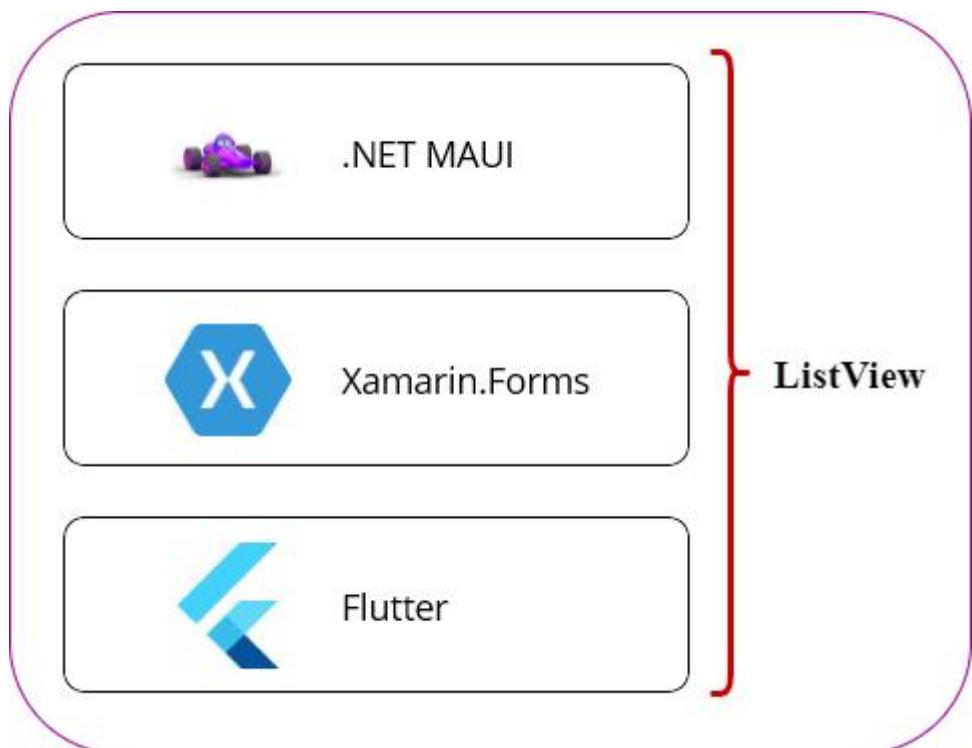
- **CarouselView:** Tương tự như một băng chuyền, cho phép người dùng vuốt sang trái hoặc sang phải để xem nhiều mục trong một không gian nhỏ gọn, dùng để hiển thị hình ảnh sản phẩm, bài đăng blog, tin tức hoặc nội dung khác theo kiểu cuộn liên tục. Có thể tùy chỉnh hiệu ứng cuộn, chỉ báo trang, bố cục và các thuộc tính khác.
- **IndicatorView:** Hiển thị một tập hợp các chỉ báo để thể hiện tiến trình hoặc trạng thái của một thứ gì đó, dùng để hiển thị số trang trong CarouselView,

vị trí hiện tại hoặc mức độ hoàn thành của một thao tác. Có thể tùy chỉnh số lượng, kiểu dáng, màu sắc và vị trí của các chỉ báo.



Hình 3.19: `CarouselView` và `IndicatorView`

- **ListView:** View đơn giản để hiển thị danh sách các mục dữ liệu với bố cục dạng danh sách, dùng để hiển thị danh sách email, danh sách ghi chú, danh sách lịch sử hoặc nội dung dạng danh sách khác. Hỗ trợ nhiều kiểu hiển thị cho từng mục dữ liệu như văn bản, hình ảnh, biểu tượng,... và có thể tùy chỉnh giao diện danh sách, thêm tiêu đề, footer và các thành phần khác.



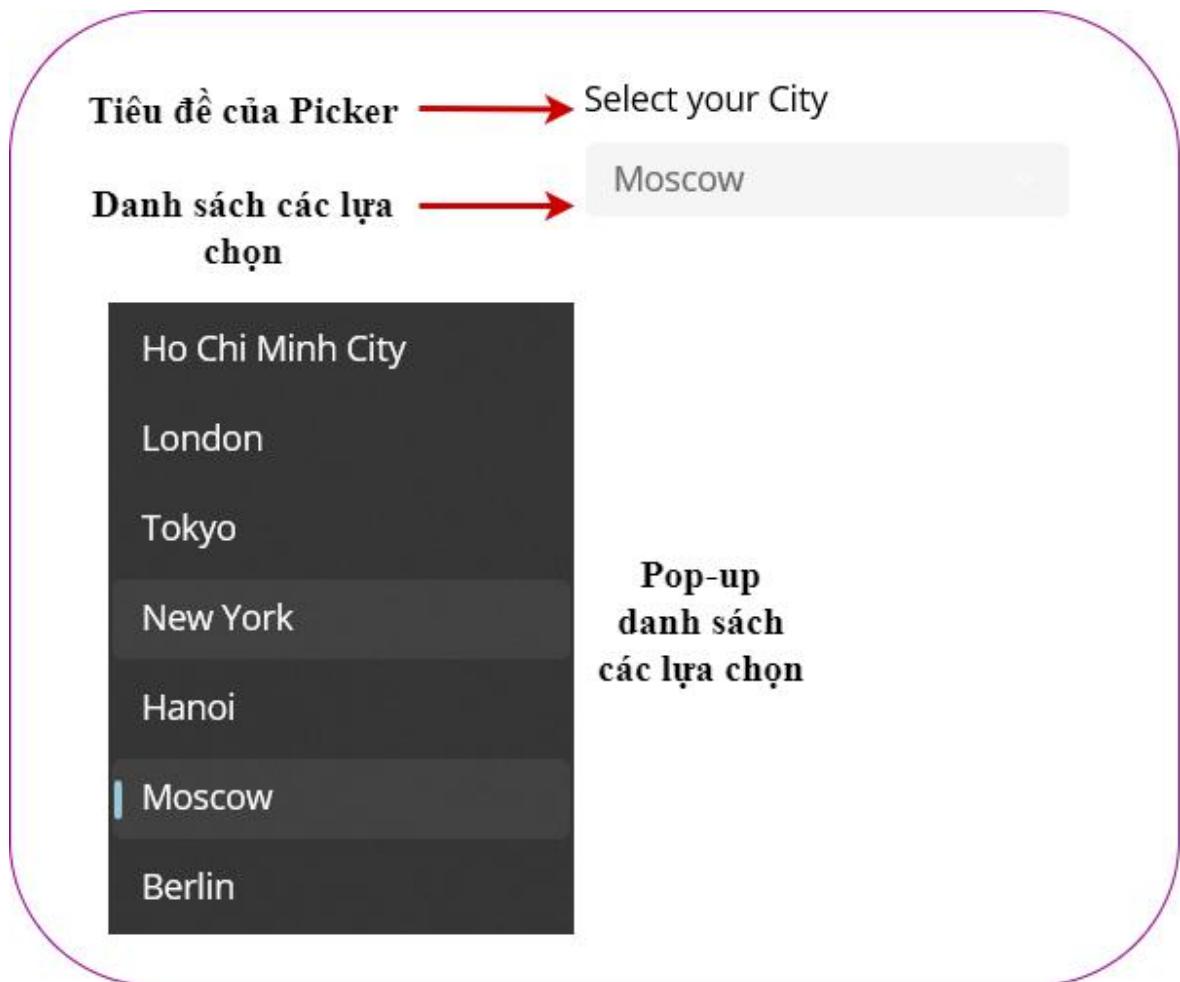
Hình 3.20: ListView

- **CollectionView:** View linh hoạt để hiển thị danh sách hoặc lưới dữ liệu với nhiều tùy chỉnh bố cục đa dạng hơn so với ListView, dùng để hiển thị danh sách sản phẩm, danh bạ, lịch sử trò chuyện hoặc nội dung có cấu trúc khác. Hỗ trợ nhiều kiểu bố cục như lưới, danh sách, thẻ,... và có thể tùy chỉnh giao diện cho từng mục dữ liệu, thêm tiêu đề, chân trang và các thành phần khác.



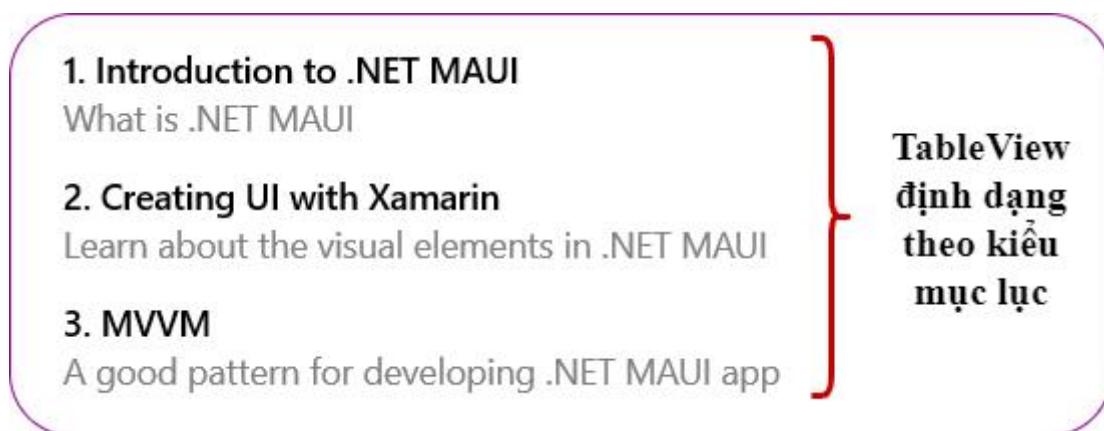
Hình 3.21: CollectionView

- **Picker:** Cho phép người dùng chọn một mục từ danh sách các tùy chọn, dùng để thu thập thông tin đầu vào, lựa chọn cài đặt hoặc chọn một giá trị từ danh sách. Có thể tùy chỉnh danh sách các tùy chọn, văn bản hiển thị, giá trị được chọn và các thuộc tính khác.



Hình 3.22: Picker

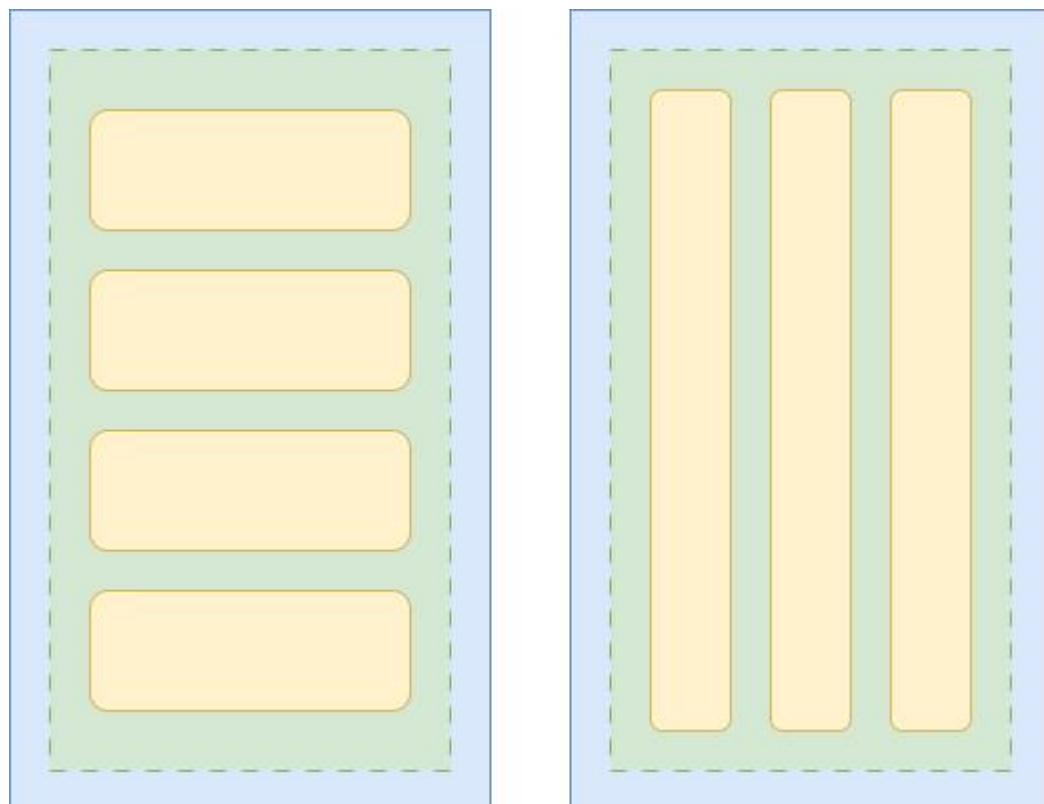
- **TableView:** View linh hoạt để hiển thị dữ liệu dạng bảng với nhiều tùy chỉnh bố cục và định dạng, dùng để hiển thị bảng tính, báo cáo, số liệu thống kê hoặc nội dung dạng bảng khác. Hỗ trợ nhiều kiểu cột như văn bản, số, hình ảnh,... và có thể tùy chỉnh tiêu đề cột, định dạng dữ liệu, thêm hàng và cột, và các thuộc tính khác.



Hình 3.23: TableView

3.3. Layout

3.3.1. StackLayout



Hình 3.24: Minh họa StackLayout

StackLayout là một layout cơ bản sắp xếp các view con ở bên trong theo ngắn xếp một chiều, có thể theo chiều ngang hoặc chiều dọc. Mặc định, StackLayout được xác định theo chiều dọc.

StackLayout có thể được sử dụng làm layout cha chứa các view hoặc layout khác, hoặc có thể làm layout con nằm bên trong layout khác.

StackLayout gồm các thuộc tính chính sau:

- **Orientation:** Xác định hướng của các view và layout con, giá trị mặc định của là *Vertical* (dọc).
- **Spacing:** là một số thực xác định khoảng cách giữa các view và layout con, giá trị mặc định là 0.

* *Spacing có thể là một giá trị âm, điều này sẽ làm các view con overlap (đè lên nhau)*

Kích thước của các view con trong StackLayout phụ thuộc vào giá trị của các thuộc tính **HeightRequest** và **WidthRequest** của chúng, trong khi vị trí thì phụ thuộc vào giá trị của các thuộc tính **HorizontalOptions** và **VerticalOptions**.

Đối với StackLayout theo chiều dọc, các view con sẽ tự động mở rộng theo chiều ngang (chiều rộng) nếu như kích thước của chúng chưa được xác định. Tương tự, đối với StackLayout theo chiều chiều ngang, các view con sẽ tự động mở rộng theo chiều dọc (chiều cao) nếu như kích thước của chúng chưa được xác định.

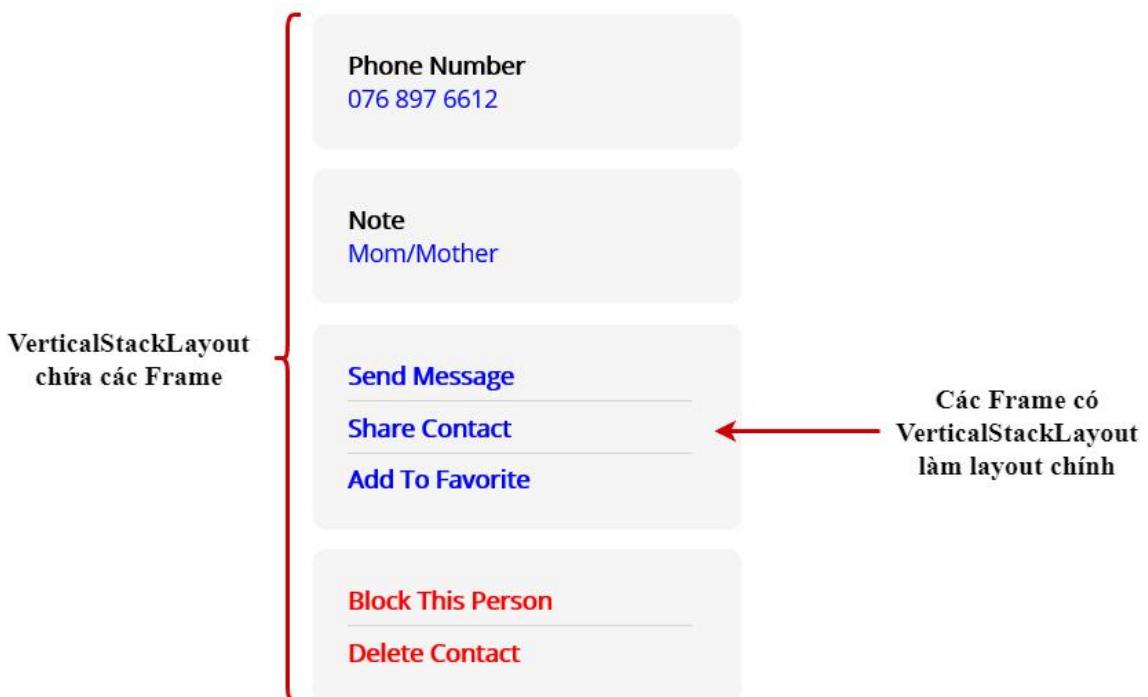
Đối với StackLayout theo chiều dọc, chỉ có thể áp dụng **HorizontalOptions**. Trong khi StackLayout theo chiều ngang, chỉ có thể áp dụng **VerticalOptions**.

3.3.1.1. VerticalStackLayout

VerticalStackLayout có bản chất tương tự như StackLayout, sắp xếp các view con ở bên trong theo ngắn xếp dọc một chiều, qua đó có hiệu suất tốt hơn StackLayout và không có thuộc tính Orientation.

VerticalStackLayout giống với StackLayout theo chiều dọc vì vậy chỉ có thể áp dụng HorizontalOptions cho các view và layout con.

** Giá trị mặc định của HorizontalOptions của VerticalStackLayout là Fill giúp cho bối cục được tối ưu nhất. Thay đổi thuộc tính này sẽ tiêu tốn thêm bộ nhớ ngay cả khi sau đó thuộc tính này được đổi lại về mặc định.*

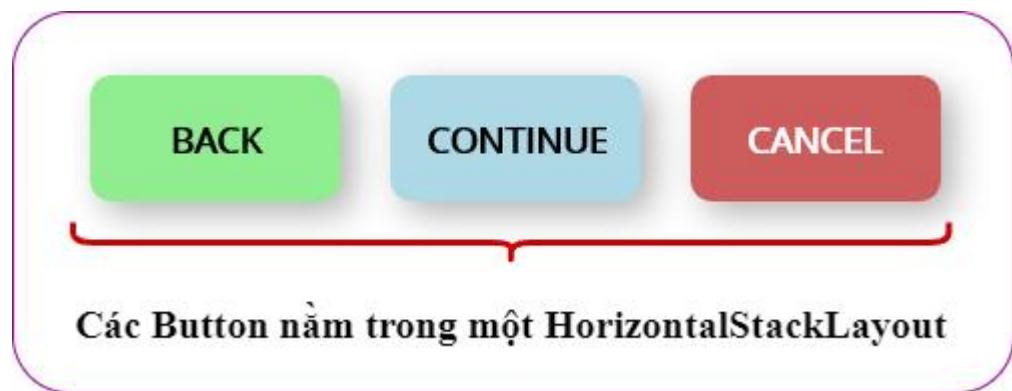


Hình 3.25: VerticalStackLayout

3.3.1.2. HorizontalStackLayout

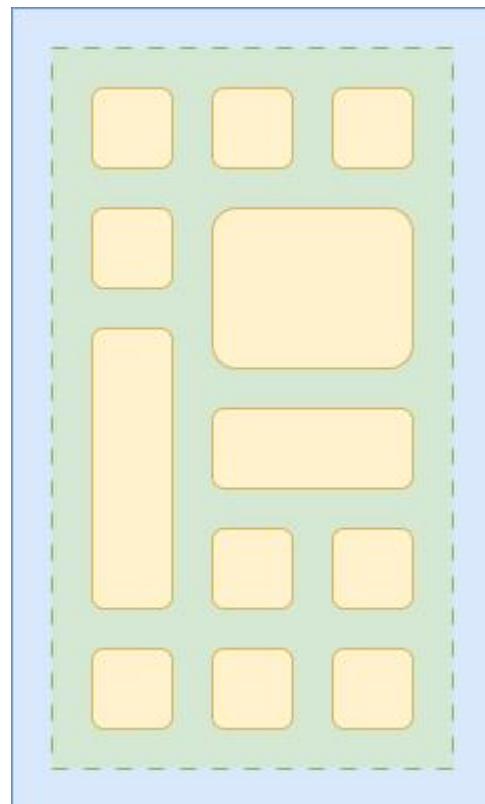
HorizontalStackLayout có bản chất tương tự như StackLayout, sắp xếp các view con ở bên trong theo ngăn xếp ngang một chiều, qua đó có hiệu suất tốt hơn StackLayout và không có thuộc tính Orientation.

HorizontalStackLayout giống với StackLayout theo chiều ngang vì vậy chỉ có thể áp dụng VerticalOptions cho các view và layout con.



Hình 3.26: HorizontalStackLayout

3.3.2. Grid



Hình 3.27: Minh họa Grid

Grid là một layout dạng lưới sắp xếp các phần tử con nằm bên trong thành các hàng và cột có kích thước theo tỉ lệ hoặc tuyệt đối. Grid có thể làm layout cha cho các layout con khác.

* *Mặc định Grid sẽ có một hàng và một cột. Số thứ tự của hàng hoặc cột sẽ bắt đầu từ số 0, có nghĩa là một grid có 5 cột, thì số thứ tự của từng cột lần lượt là 0, 1, 2, 4.*

Grid còn có nhiều thuộc tính đính kèm (attached property). Các thuộc tính đính kèm là các thuộc tính không trực tiếp thuộc phần tử cha, thay vào đó được đính kèm và được xác định giá trị bởi các phần tử con nằm trong lớp cha, thông thường, các phần tử con có thể sử dụng thuộc tính đính kèm để thông báo cho phần tử cha về cách chúng được hiện thị trên giao diện.

Các thuộc tính đính kèm:

- **Row:** Xác định vị trí của các phần tử con theo dòng, là một giá trị số nguyên lớn hơn 0 với giá trị mặc định là 0.
- **Column:** Xác định vị trí của các phần tử con theo cột, là một giá trị số nguyên lớn hơn 0 với giá trị mặc định là 0.
- **RowSpan:** Xác định số dòng mà các phần tử con trải rộng (span) bên trong grid, là một giá trị số nguyên lớn hơn một với giá trị mặc định là 1.
- **ColumnSpan:** Xác định số cột mà các phần tử con trải rộng bên trong grid, là một giá trị số nguyên lớn hơn một với giá trị mặc định là 1.

Các thuộc tính của grid:

- **RowDefinitions:** Một danh sách các RowDefinition xác định chiều cao (height) của các dòng. Danh sách này chứa một RowDefinition cho mỗi dòng trong grid.

- **ColumnDefinitions:** Một danh sách các ColumnDefinition xác định chiều rộng (weight) của các cột. Danh sách này chứa một ColumnDefinition cho mỗi cột trong grid.
- **RowSpacing:** Xác định khoảng cách giữa các hàng trong grid, là một giá trị số thực với giá trị mặc định là 0.
- **ColumnSpacing:** Xác định khoảng cách giữa các cột trong grid, là một giá trị số thực với giá trị mặc định là 0.

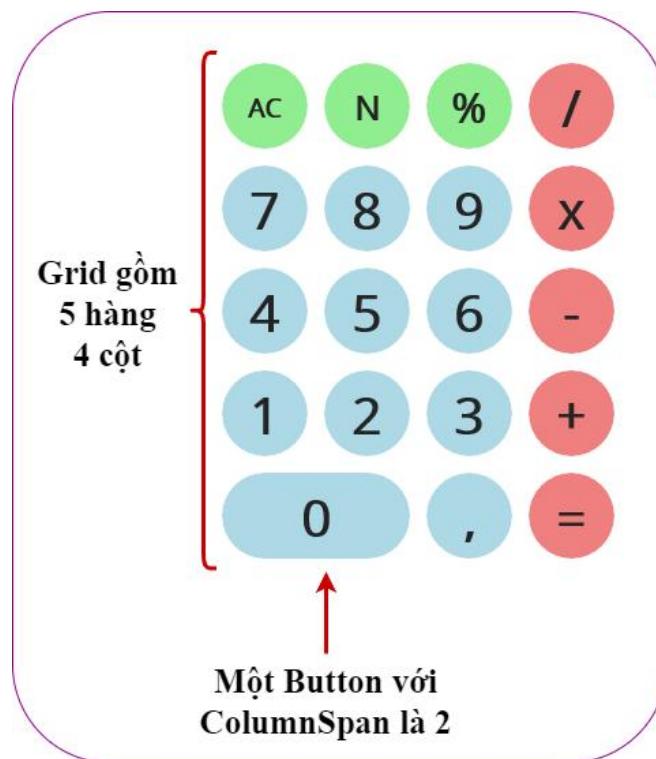
Giá trị chiều cao của RowDefinition và chiều rộng của ColumnDefinition có thể được xác định bằng 3 cách sau:

- **Absolute:** Chiều cao của dòng và chiều rộng của cột là một giá trị tính bằng đơn vị đo trên các thiết bị.
- **Auto:** Chiều cao của dòng và chiều rộng của cột tự động được điều chỉnh dựa trên các nội dung bên trong các ô (cell).
- **Star:** Chiều cao của dòng và chiều rộng của cột còn lại được phân bổ theo tỉ lệ.

* **Các phần tử con trong grid có thể cùng nằm trong một cell. Thứ tự xuất hiện của chúng là thứ tự được thêm vào cell, phần tử được thêm vào sau sẽ đè lên phần tử trước.**

* **RowSpacing và ColumnSpacing có thể là giá trị âm, điều này sẽ làm các cell overlap.**

* **Các phần tử con trong grid có thể được căn chỉnh (align) trong cell của chúng bằng HorizontalOptions và VerticalOptions.**



Hình 3.28: Grid

3.3.3. FlexLayout

FlexLayout là một layout sắp xếp phần tử con theo chiều ngang hoặc dọc trong một ngăn xếp. Nếu có quá nhiều phần tử con để có thể xếp vào trong một hàng hoặc một cột thì FlexLayout có thể bao bọc (wrap) chúng lại.

Điểm mạnh của FlexLayout là nó có thể thích nghi với nhiều kích cỡ màn hình và có thể tùy ý điều chỉnh hướng và alignment.

Trong FlexLayout, main axis là trục chính của bố cục, xác định cách phân bổ các phần tử con trong bố cục, cross axis là trục vuông góc với main axis.

*** FlexLayout dựa trên CSS Flexible Box Layout Module**

Các thuộc tính của FlexLayout:

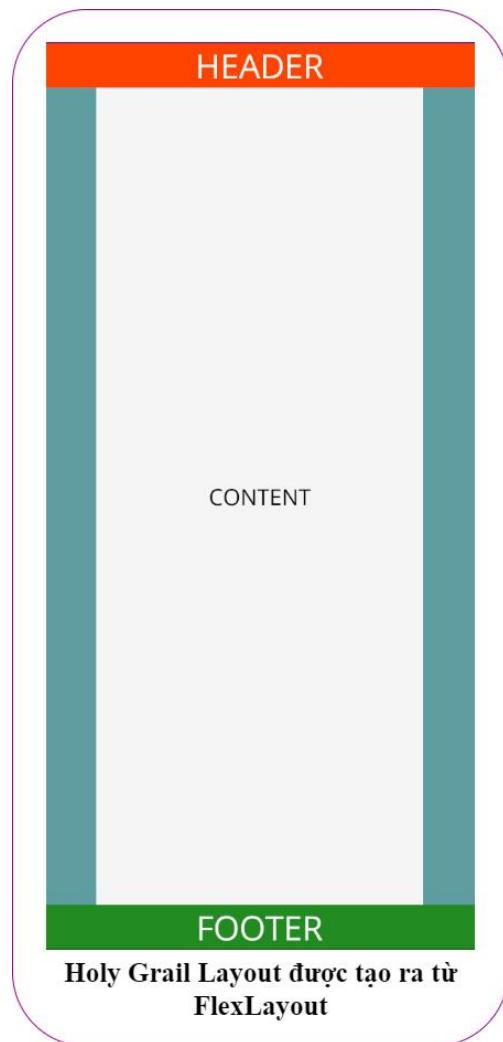
- ***AlignContent***: Xác định cách bố cục sẽ phân bổ không gian giữa và xung quanh các phần tử con đã được bố trí trên nhiều dòng. Giá trị mặc định là Stretch.
- ***AlignItems***: Cho biết cách công cụ bố cục sẽ phân bổ không gian giữa và xung quanh các phần tử con theo cross axis. Giá trị mặc định là Stretch.
- ***Direction***: Xác định hướng và main axis của các phần tử con. Giá trị mặc định là Row.
- ***JustifyContent***: Xác định cách phân bố không gian giữa và xung quanh các phần tử con dọc theo main axis. Giá trị mặc định là Start.
- ***Position***: Xác định xem vị trí của các phần tử con có tương đối với nhau hay là sử dụng các giá trị cố định. Giá trị mặc định Relative.
- ***Wrap***: Kiểm soát xem các phần tử con được sắp xếp thành một dòng hay nhiều dòng. Giá trị mặc định là NoWrap.

Bên cạnh đó FlexLayout còn có các thuộc tính đính kèm sau:

- ***AlignSelf***: Cho biết cách bố cục sẽ phân bổ không gian giữa và xung quanh các phần tử con cho một phần tử con cụ thể dọc theo cross axis. Giá trị mặc định là Auto.
- ***Basis***: Xác định kích thước chính ban đầu của phần tử con trước khi không gian trống được phân bổ theo các giá trị thuộc tính khác. Giá trị mặc định là Auto.
- ***Grow***: Là một giá trị số thực chỉ định lượng không gian có sẵn mà phần tử con nên sử dụng trên main axis. Giá trị mặc định là 0,0.
- ***Order***: Là một giá trị số nguyên xác định xem phần tử con đó nên được xếp trước hay sau các phần tử con khác trong vùng chứa. Giá trị mặc định là 0.

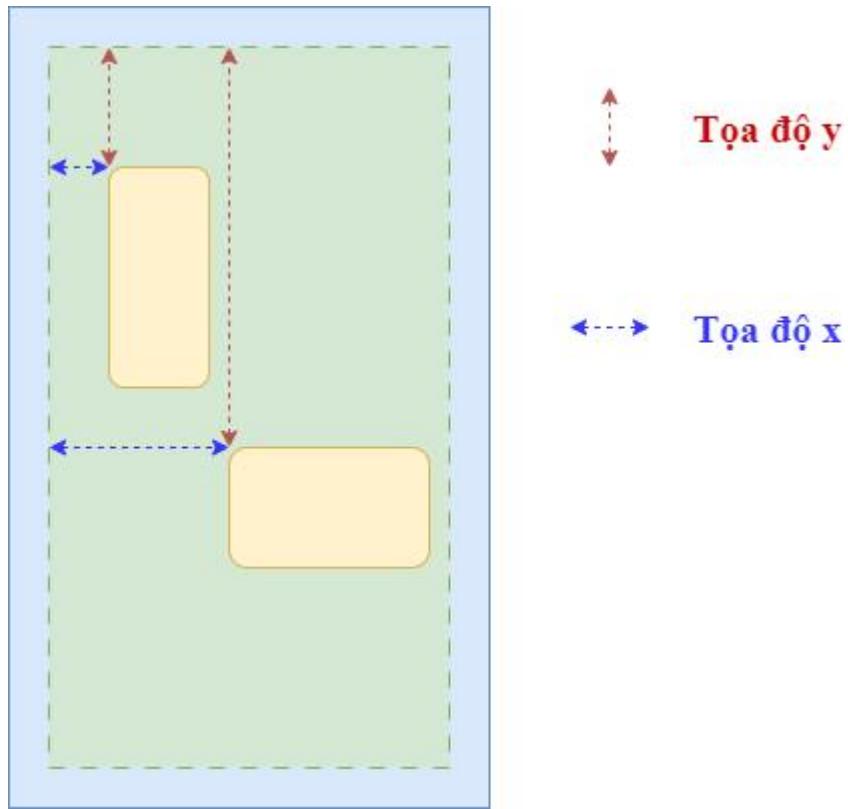
- **Shrink:** Là một giá trị số thực kiểm soát cách một phần tử con nên co lại như thế nào để các phần tử con khác có thể vừa bên trong vùng chứa. Giá trị mặc định là 1.0.

* **Sự phức tạp của FlexLayout mang lại sự linh hoạt cho layout này, cũng vì thế, một trong các layout phổ biến là Holy Grail Layout (tạm dịch: “Bố cục chén thánh”) có thể dễ dàng được tạo ra từ FlexLayout.**



Hình 3.29: FlexLayout

3.3.4. AbsoluteLayout



Hình 3.30: Minh họa AbsoluteLayout

AbsoluteLayout được sử dụng để xác định vị trí và kích thước của các phần tử con bằng các giá trị rõ ràng.

Vị trí được xác định bởi góc trên bên trái của phần tử con với góc trên bên trái của AbsoluteLayout theo đơn vị độc lập của thiết bị.

AbsoluteLayout còn có thể điều chỉnh vị trí và kích thước của các phần tử con theo tỉ lệ. Bên cạnh đó bộ cục này còn có thể điều chỉnh vị trí sao cho các phần tử con có thể nằm đè lên nhau.

AbsoluteLayout có thể được xem là một bộ cục đặc biệt chỉ được dùng trong vài trường hợp.

AbsoluteLayout không có thuộc tính cho riêng nó, thay vào đó là hai thuộc tính đính kèm:

- **LayoutBounds:** Là thuộc tính đính kèm thể hiện vị trí và kích thước của phần tử con. Giá trị mặc định là (0,0,AutoSize,AutoSize).
- **LayoutFlags:** Cho biết liệu vị trí và kích thước phần tử con có được thể hiện theo tỷ lệ hay không. Giá trị mặc định là None.

Thuộc tính LayoutBounds có hai format:

- *Nếu chỉ được xác định bởi hai giá trị x, y:* x và y xác định vị trí của góc trên bên trái của phần tử con so với phần tử cha của nó. Phần tử sẽ tự động điều chỉnh kích thước.
- *Nếu chỉ được xác định bởi bốn giá trị x, y, width và height:* x và y xác định vị trí của góc trên bên trái của phần tử con so với phần tử cha của nó, trong khi width và height xác định kích thước.

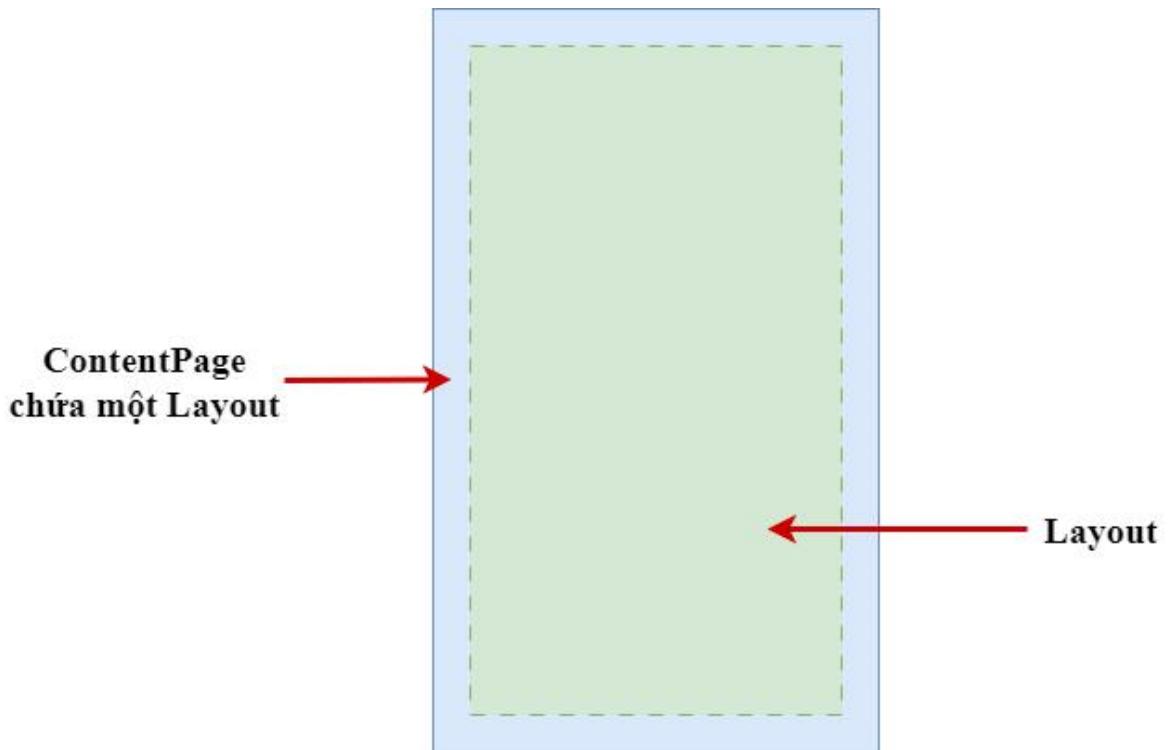
* **HorizontalOptions** và **VerticalOptions** không thể áp dụng cho các phần tử con nằm trong **AbsoluteLayout**.



Hình 3.31: *AbsoluteLayout*

3.4. Page

3.4.1. ContentPage



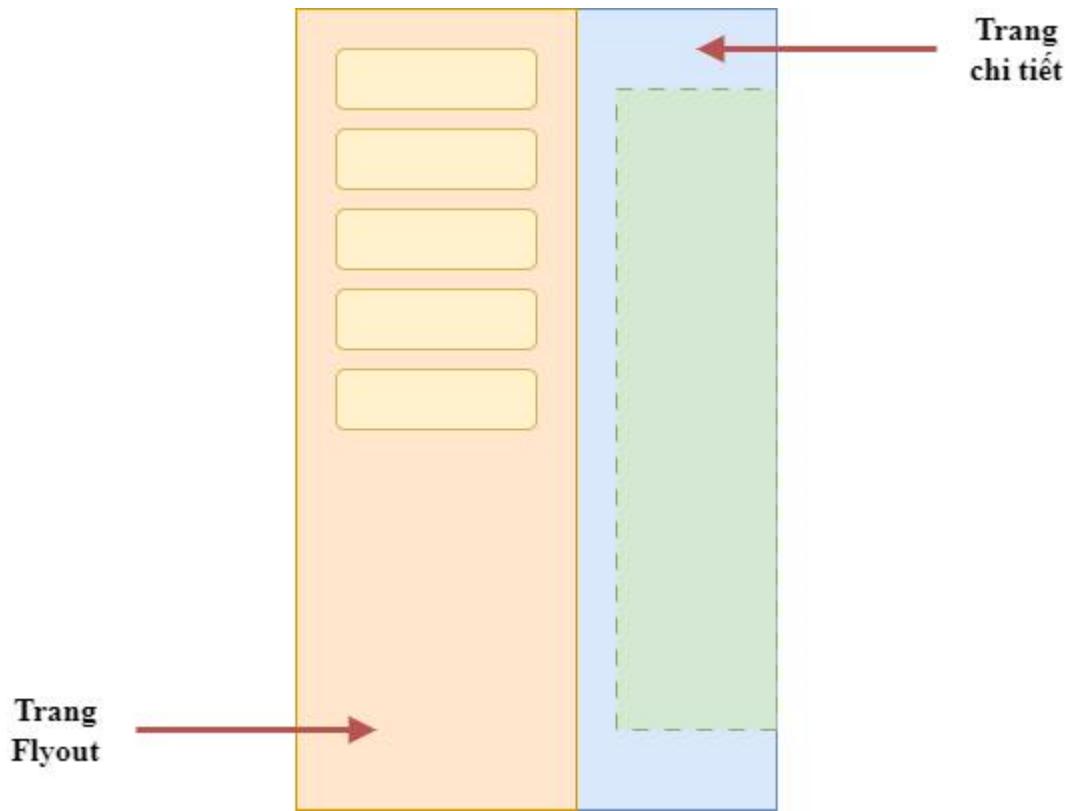
Hình 3.32: Minh họa ContentPage

ContentPage là một loại trang phổ biến và cơ bản, dùng để hiện thị một layout duy nhất, thường là StackLayout hoặc Grid.

ContentPage gồm các thuộc tính cơ bản của một trang như: Title, IconImageSource, BackgroundImageSource, IsBusy, and Padding, bên cạnh đó còn có thêm thuộc tính HideSoftInputOnTapped là một giá trị bool xác định rằng nếu nhấn vào bất cứ đâu trên trang sẽ làm ẩn đi bàn phím (trường là trên các thiết bị di động) nếu như nó đang xuất hiện.

ContentPage tuy đơn giản, nhưng thường được dùng làm trang khuôn mẫu. Một ứng dụng .NET MAUI thường có nhiều trang khuôn mẫu bắt nguồn từ ContentPage và sự định vị giữa các trang này có thể được thực hiện.

3.4.2. FlyoutPage



Hình 3.33: Minh họa FlyoutPage

FlyoutPage trong .NET MAUI giúp tổ chức và quản lý thông tin hiệu quả bằng cách kết hợp hai trang:

- **Trang Flyout:** Hiển thị danh sách các mục, cho phép người dùng dễ dàng lựa chọn và điều hướng, đóng vai trò như một menu định hướng giữa các trang khác nhau.
- **Trang Chi tiết:** Hiển thị thông tin chi tiết về mục được chọn từ trang Flyout.

FlyoutPage cung cấp hai hành vi bố cục linh hoạt để phù hợp với từng thiết bị và nhu cầu sử dụng:

- **Popover:** Trang chi tiết che phủ toàn bộ hoặc che phủ một phần trang Flyout. Chọn một mục trên trang Flyout sẽ điều hướng đến trang chi tiết

tương ứng. Hành vi này thường được sử dụng trên điện thoại, nơi không gian màn hình hạn chế.

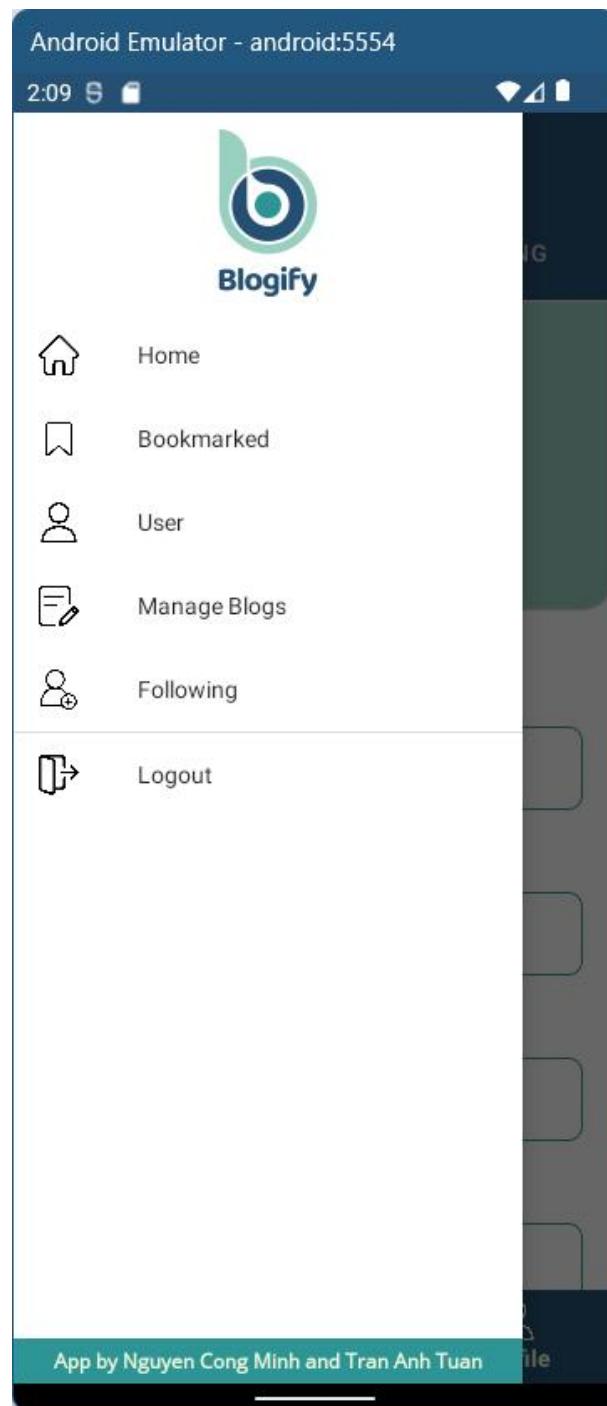
- **Split:** Trang Flyout được hiển thị cố định ở bên trái màn hình, trong khi trang chi tiết chiếm phần còn lại. Hành vi này phù hợp cho máy tính bảng hoặc máy tính để bàn, nơi có nhiều không gian màn hình hơn. Bên cạnh đó, hành vi này còn cung cấp thêm hai chế độ bao gồm:
 - **SplitOnLandscape:** Split chỉ được áp dụng khi thiết bị được xoay ngang. Trên màn hình dọc, nó sẽ chuyển sang Popover.
 - **SplitOnPortrait:** Split chỉ được áp dụng khi thiết bị được xoay dọc. Trên màn hình ngang, nó sẽ chuyển sang Popover.

FlyoutPage cung cấp một số thuộc tính quan trọng giúp tùy chỉnh và điều khiển cách thức hoạt động của nó:

- **Detail:** Là một trang (page), xác định trang chi tiết sẽ được hiển thị khi người dùng chọn một mục trong trang Flyout.
- **Flyout:** Là một trang (page), xác định trang Flyout, nơi hiển thị danh sách các mục.
- **FlyoutLayoutBehavior:** Xác định hành vi bối cảnh của FlyoutPage.
- **IsGestureEnabled:** Cho phép hoặc tắt cử chỉ vuốt để chuyển đổi giữa trang Flyout và trang chi tiết.
- **IsPresented:** Xác định xem trang Flyout hay trang chi tiết có được hiển thị hay không. Giá trị true thì trang Flyout được hiển thị, ngược lại, giá trị false thì trang chi tiết được hiển thị.

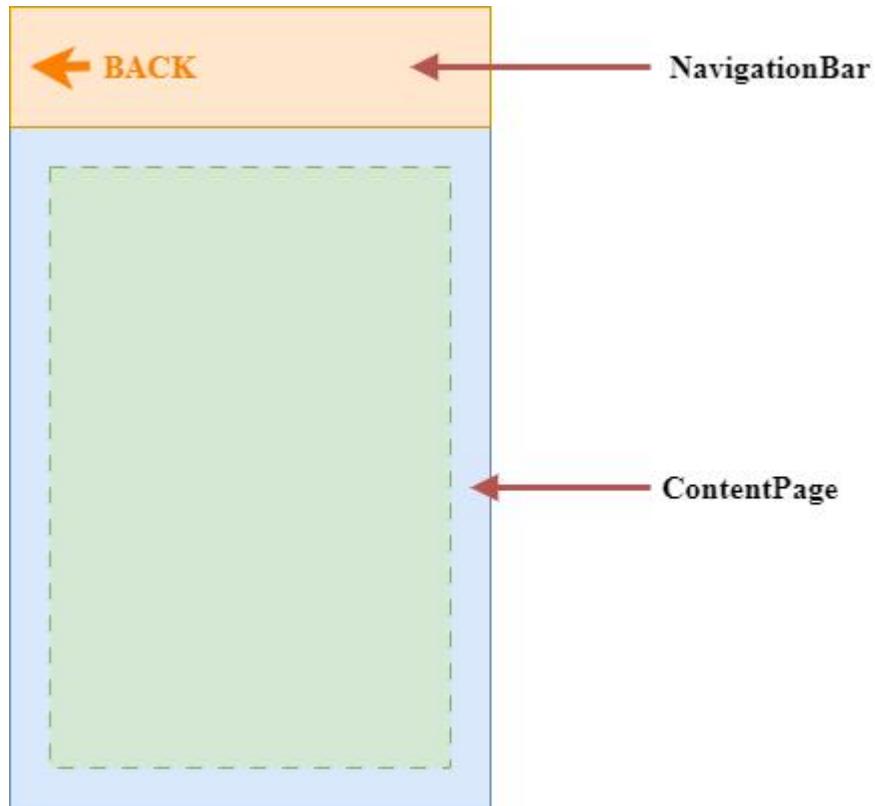
* *Nếu giá trị của thuộc tính FlyoutLayoutBehavior là Default thì hành vi của FlyoutPage sẽ được xác định dựa vào nền tảng mà thiết bị đang chạy trên. Ví dụ như trên Windows, hành vi bối cảnh của FlyoutPage sẽ là Split.*

* *Thuộc tính FlyoutLayoutBehavior chỉ ảnh hưởng đến các ứng dụng chạy trên máy tính bảng hoặc máy tính để bàn, ứng dụng chạy trên các thiết bị di động luôn có hành vi là Popover.*



Hình 3.34: FlyoutPage

3.4.3. NavigationPage



Hình 3.35: Minh họa NavigationPage

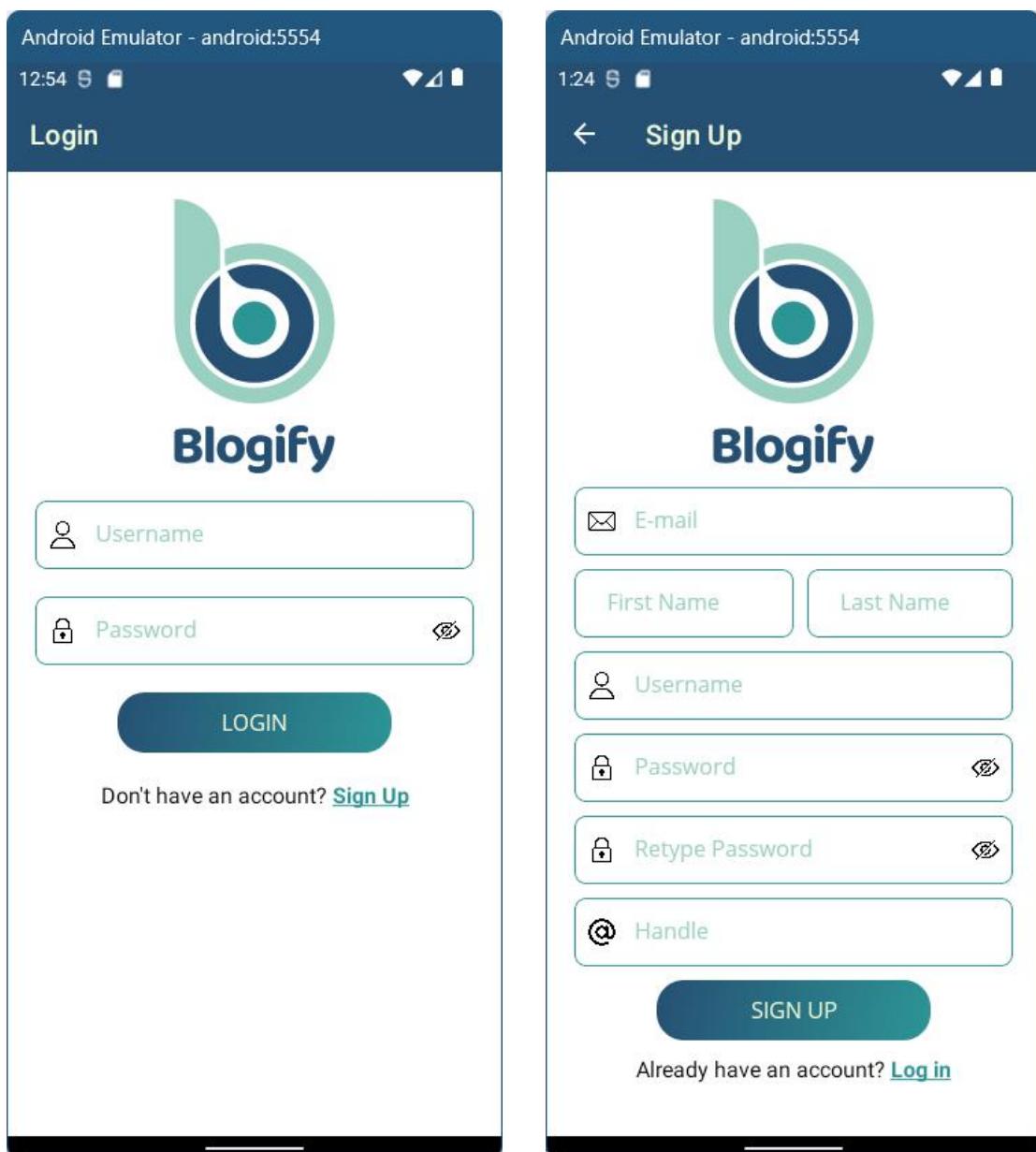
NavigationPage trong .NET MAUI cung cấp trải nghiệm điều hướng phân cấp giúp dễ dàng điều hướng tiến hoặc lùi qua các trang.

NavigationPage hoạt động như một ngăn xếp dạng last-in first-out (LIFO) với đối tượng là các trang. Trang vào ngăn xếp sau cùng là trang được đưa ra đầu tiên.

NavigationPage gồm các thuộc tính sau:

- **BarBackground:** Xác định nền của thanh điều hướng, có thể sử dụng ColorBrush, LinearGradientBrush, ImageBrush,... để tạo ra các hiệu ứng nền độc đáo.
- **BarBackgroundColor:** Chỉ định màu nền trực tiếp cho thanh điều hướng.

- **BackButtonTitle:** Là thuộc tính đính kèm, giúp thay đổi văn bản hiển thị trên nút quay lại trong thanh điều hướng. Mặc định, văn bản là "Back".
- **BarTextColor:** Chỉ định màu của văn bản hiển thị trên thanh điều hướng, bao gồm tiêu đề và nút quay lại.
- **CurrentPage:** Đại diện cho trang nằm trên cùng của ngăn xếp điều hướng. Đây là thuộc tính read-only.
- **HasNavigationBar:** Là thuộc tính đính kèm, xác định liệu thanh điều hướng có hiển thị trên NavigationPage hay không. Mặc định, thanh điều hướng được hiển thị.
- **HasBackButton:** Là thuộc tính đính kèm, xác định liệu thanh điều hướng có bao gồm nút quay lại hay không. Mặc định, nút quay lại được hiển thị.
- **IconColor:** Là thuộc tính đính kèm, chỉ định màu nền của biểu tượng trong thanh điều hướng. Biểu tượng này thường được sử dụng để hiển thị logo hoặc hình ảnh đại diện cho ứng dụng.
- **RootPage:** Đại diện cho trang gốc của ngăn xếp điều hướng. Đây là thuộc tính read-only.
- **TitleIconImageSource:** Là thuộc tính đính kèm, xác định biểu tượng được hiển thị bên cạnh tiêu đề trên thanh điều hướng. Có thể sử dụng ImageSource để tải hình ảnh từ tệp tin hoặc URL.
- **TitleView:** Là thuộc tính đính kèm, thay thế tiêu đề mặc định trên thanh điều hướng bằng một view tùy chỉnh. View này có thể là một Label, Image,...



Hình 3.36: NavigationPage

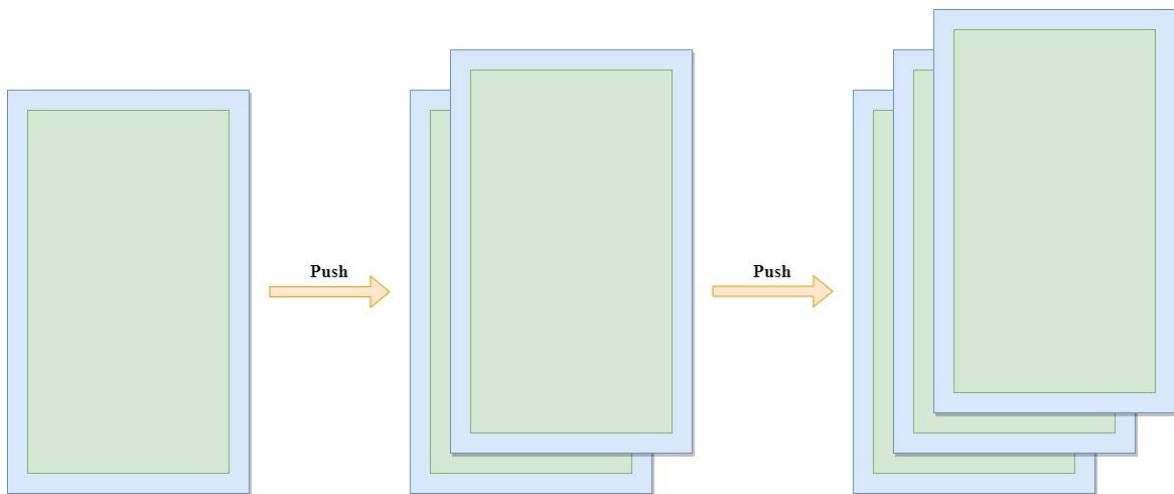
NavigationPage cung cấp ba sự kiện quan trọng giúp theo dõi trạng thái điều hướng và thực hiện hành động phù hợp trong ứng dụng:

- **Pushed:** Được kích hoạt khi một trang mới được đẩy lên ngăn xếp điều hướng.

- **Popped:** Được kích hoạt khi một trang được bật ra khỏi ngăn xếp điều hướng.
- **PoppedToRoot:** Được kích hoạt khi trang không phải gốc cuối cùng được bật ra khỏi ngăn xếp điều hướng, hay nói cách khác, là khi người dùng đã điều hướng quay lại trang gốc.

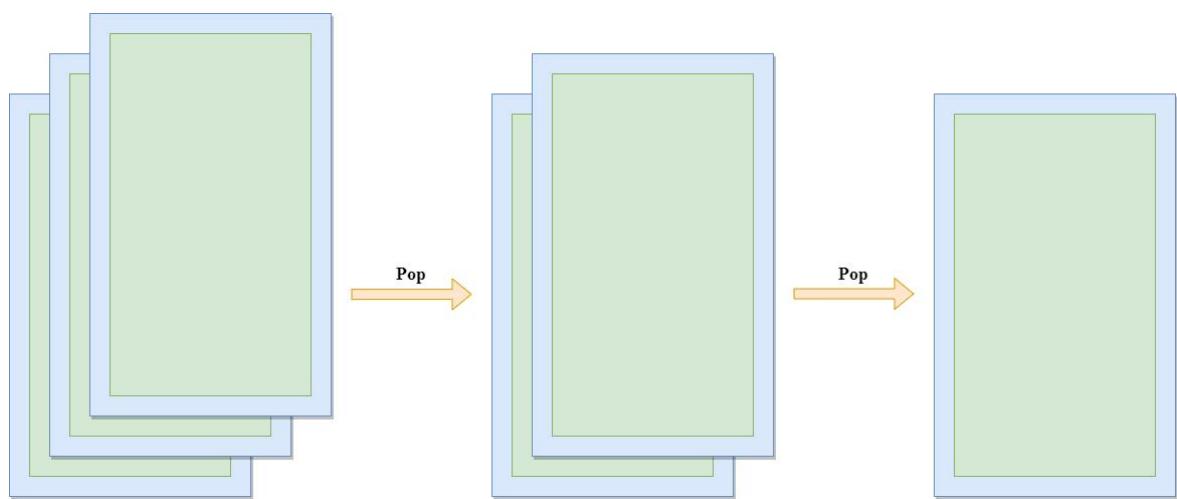
Cách thức hoạt động:

- **Thêm trang:** Khi thêm một trang mới vào NavigationPage bằng phương thức PushAsync(), trang đó sẽ được đẩy lên ngăn xếp và trở thành trang hoạt động. Trang này sẽ hiển thị trên màn hình và người dùng có thể tương tác với nó.



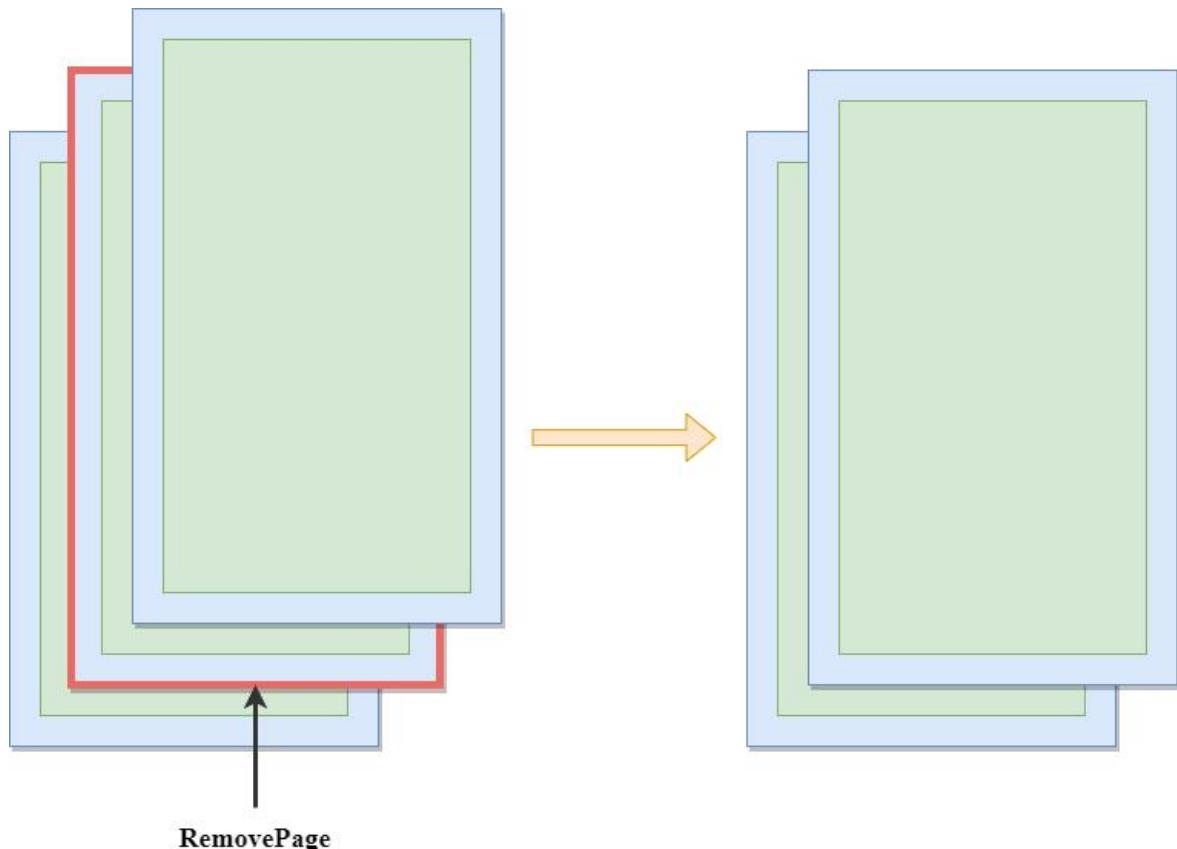
Hình 3.37: Minh họa hoạt động thêm trang

- **Quay lại trang trước:** Khi người dùng nhấp vào nút "Back" trong ứng dụng, sử dụng nút quay lại vật lý hoặc cảm ứng trên các thiết bị, trang hoạt động hiện tại sẽ được bật ra khỏi ngăn xếp. Trang tiếp theo trong ngăn xếp sẽ trở thành trang hoạt động mới và được hiển thị trên màn hình.



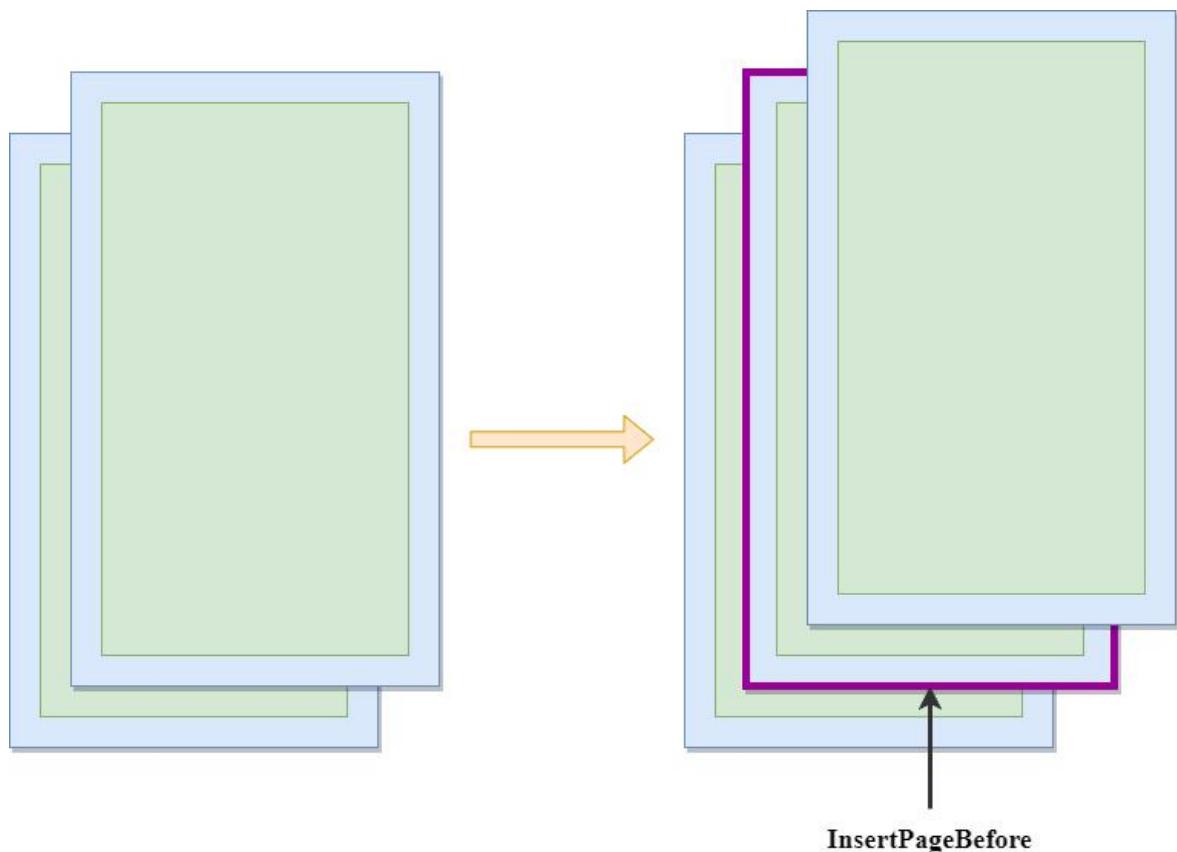
Hình 3.38: Minh họa hoạt động quay lại trang trước

- **Xóa trang:** Có thẻ sử dụng phương thức RemovePage() để xóa một trang cũ thẻ khỏi ngăn xếp điều hướng. Trang đã xóa không thể quay trở lại.



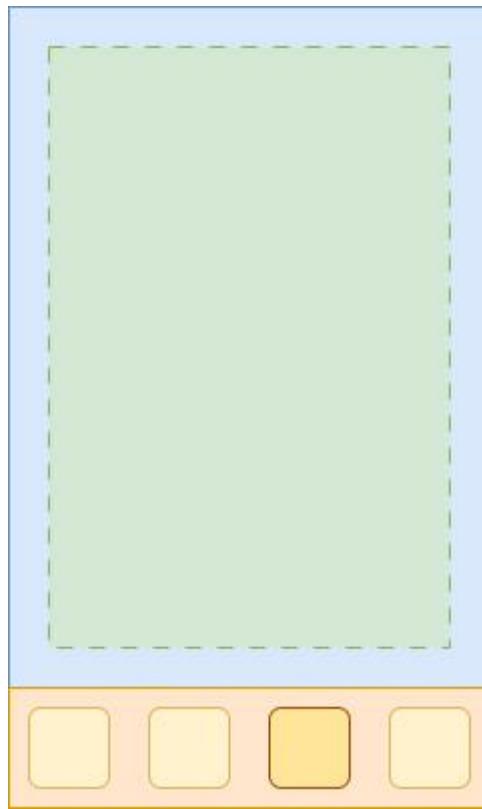
Hình 3.39: Minh họa hoạt động xóa trang

- **Chèn trang:** Phương thức InsertPageBefore() cho phép chèn một trang mới vào ngăn điều hướng trước một trang hiện có. Điều này giúp định cấu trúc thứ tự điều hướng mong muốn cho ứng dụng.



Hình 3.40: Minh họa hoạt động chèn trang

3.4.4. TabbedPage



Hình 3.40: Minh họa TabbedPage

TabPage trong .NET MAUI cho phép hiển thị nhiều trang con dưới dạng các tab, giúp người dùng dễ dàng chuyển đổi giữa các nội dung khác nhau trong cùng một giao diện.

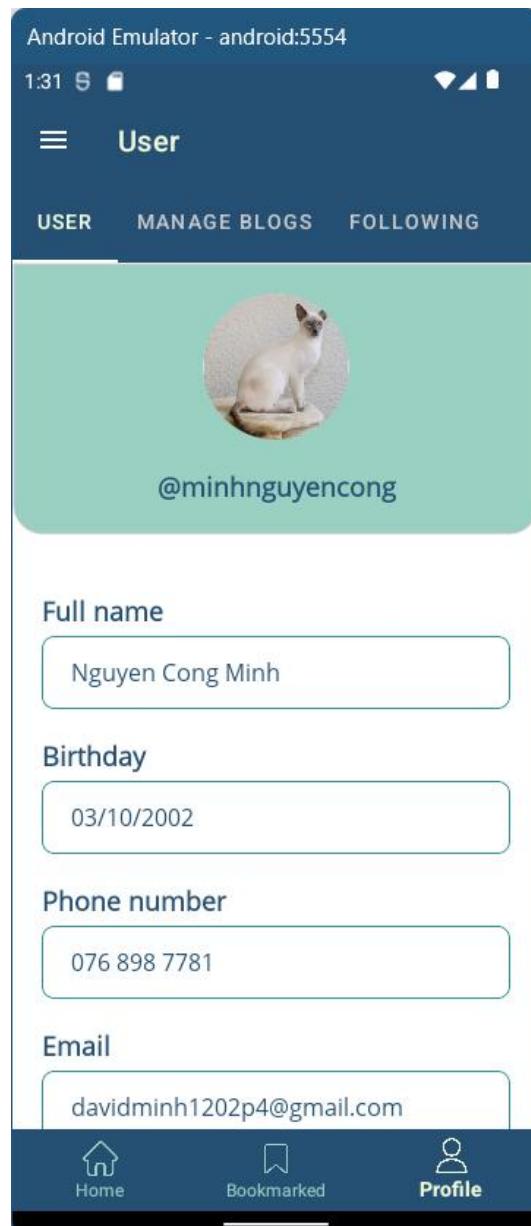
TabPage chứa một tập hợp các trang con, mỗi trang đại diện cho một nội dung riêng biệt. Các tab được hiển thị ở trên cùng hoặc dưới cùng của trang, cho phép người dùng trực quan lựa chọn trang con họ muốn xem, khi người dùng chọn một tab, trang con tương ứng sẽ được hiển thị, che khuất các trang con khác.

TabPage có các thuộc tính sau:

- **BarBackground:** Xác định nền của thanh tab, có thể sử dụng ColorBrush, LinearGradientBrush, ImageBrush,... để tạo ra các hiệu ứng nền độc đáo.

- **BarBackgroundColor:** Chỉ định màu nền trực tiếp cho thanh tab.
- **BarTextColor:** Chỉ định màu của văn bản hiển thị trên thanh tab.
- **SelectedTabColor:** Chỉ định màu của tab đang được chọn.
- **UnselectedTabColor:** Chỉ định màu của các tab đang không được chọn.

* *Tiêu đề và biểu tượng của mỗi tab là tiêu đề và biểu tượng được xác định của mỗi trang con tương ứng.*



Hình 3.41: TabbedPage

3.5. Shell

Shell là một mô hình giao diện người dùng mới trong .NET MAUI, đóng vai trò như một giải pháp thay thế hiệu quả cho các mô hình truyền thống như NavigationPage, FlyoutPage và TabbedPage. Shell cung cấp một cấu trúc linh hoạt và dễ sử dụng để xây dựng giao diện người dùng cho các ứng dụng đa nền tảng, giúp đơn giản hóa quá trình phát triển và mang lại trải nghiệm người dùng nhất quán.

Shell đóng vai trò như bộ khung cơ bản cho giao diện người dùng ứng dụng .NET MAUI, mang đến các chức năng thiết yếu mà hầu hết các ứng dụng đều cần:

- **Mô tả thứ bậc ứng dụng trực quan:** Giúp người dùng dễ dàng nắm bắt cấu trúc ứng dụng bằng cách hiển thị thứ bậc các trang một cách rõ ràng. Nhờ vậy, người dùng có thể di chuyển giữa các phần khác nhau của ứng dụng một cách trực quan và hiệu quả.
- **Trải nghiệm điều hướng phổ biến:** Cung cấp các phương thức điều hướng phổ biến như thanh tab, thanh điều hướng, Flyout, menu,... Nhờ vậy, người dùng có thể di chuyển giữa các trang một cách dễ dàng và nhanh chóng, nâng cao trải nghiệm người dùng tổng thể.
- **Sơ đồ điều hướng URI linh hoạt:** Sử dụng sơ đồ điều hướng URI để xác định vị trí của từng trang trong ứng dụng. Điều này cho phép người dùng truy cập trực tiếp vào bất kỳ trang nào bằng cách nhập địa chỉ URI tương ứng, mang đến sự linh hoạt và tiện lợi.
- **Tích hợp trình xử lý tìm kiếm hiệu quả:** Tích hợp sẵn trình xử lý tìm kiếm, giúp người dùng nhanh chóng tìm kiếm nội dung mong muốn trong ứng dụng. Nhờ vậy, người dùng có thể tiết kiệm thời gian và dễ dàng truy cập thông tin cần thiết.

* **Shell không tưởng thích với NavigationPage, FlyoutPage và TabbedPage.**
Người dùng chỉ có thể chọn dùng Shell hoặc các trang điều hướng trên tùy vào như cầu sử dụng cũng như độ phức tạp của ứng dụng.

3.5.1. Flyout

Flyout trong Shell có chức năng và hành vi tương tự như FlyoutPage là sử dụng các mục để điều hướng giữa các trang trong ứng dụng, tuy nhiên có một số điểm khác biệt là FlyoutPage có cấu trúc cố định, còn Flyout trong Shell linh hoạt hơn, cho phép tùy chỉnh nhiều hơn về mặt hiện thị như Header, Footer, Menu.

3.5.1.1. Mục Flyout

Tương tự như FlyoutPage, các mục này cho phép người dùng điều hướng giữa các trang, điều khác biệt lớn nhất là Flyout trong Shell cho phép người dùng xác định cách hiện thị các mục Flyout và các mục con của nó bằng cách sử dụng thuộc tính FlyoutDisplayOptions:

- **AsSingleItem:** Mục Flyout sẽ được hiện thị dưới dạng một mục đơn lẻ.
- **AsMultipleItems:** Mục Flyout và các mục con của nó sẽ được hiện thị dưới dạng một nhóm các mục.

Thuộc tính này mang đến sự linh hoạt mà FlyoutPage không có được. Người dùng có thể tùy chọn giá trị cho thuộc tính này tùy vào số lượng mục hoặc chức năng của ứng dụng. Đối với các ứng dụng có nhiều mục hoặc mục con thì AsMultipleItems giúp nhóm những mục có điểm tương đồng lại để tiết kiệm không gian trong giao diện cũng như tránh làm phức tạp việc điều hướng của ứng dụng, trong khi AsSingleItem, tương tự như FlyoutPage, lại phù hợp hơn với các ứng dụng đơn giản có số lượng mục ít, giúp người dùng tập trung vào từng mục riêng lẻ.

3.5.1.2. Menu

Là một chức năng tùy chọn của Flyout trong Shell, cho phép thêm các mục Menu vào Flyout bên cạnh các mục vốn có. Khác với các mục Flyout, mục Menu đóng vai trò thực thi các hành động, sự kiện.

Các mục Menu thường được dùng để mở các trang URL trong trình duyệt web của hệ thống mà ứng dụng đang chạy trên.

3.5.1.3. Header

Flyout Header trong Shell cho phép thêm nội dung tùy chỉnh vào phần đầu của Flyout, giúp tạo điểm nhấn và cung cấp thông tin bổ sung cho người dùng.

Flyout Header hiển thị ở phía trên các mục trong Flyout. Nội dung của nó được xác định bởi một đối tượng có thẻ được đặt bằng thuộc tính Shell.FlyoutHeader hoặc có thẻ xác định trực tiếp bằng DataTemplate. Đối tượng này có thể là bất kỳ loại nào có thể được hiển thị trong giao diện người dùng, ví dụ như Label, Image, StackLayout,...

Thường dùng để cung cấp các thông tin bổ sung như Logo, tiêu đề của ứng dụng hay tạo điểm nhất cho ứng dụng.

3.5.1.4. Footer

Flyout Footer tương tự như Header nhưng nằm ở phía dưới cùng của Flyout.

Thường dùng để cung cấp các thông tin bổ sung như thông tin liên hệ, bản quyền, phiên bản của ứng dụng,...

3.5.2. Tab

Tab trong Shell cho phép tạo các tab điều hướng trong phần nội dung chính của ứng dụng tương tự như TabbedPage. Mỗi Tab trong thanh Tab tương ứng với một trang riêng biệt, mang đến trải nghiệm điều hướng mượt mà và trực quan.

Một thanh Tab có thể chứa một hoặc nhiều Tab, nếu có nhiều hơn một Tab, thì thanh Tab sẽ được hiện thị phía dưới màn hình. Mỗi Tab lại có thể chứa một hoặc nhiều đối tượng nội dung của Shell, với mỗi đối tượng này là một ContentPage. Nếu một tab có nhiều hơn một đối tượng nội dung của Shell thì các đối tượng này sẽ được điều hướng thông qua các thành Tab phía trên màn hình. Nếu như có nhiều hơn 5 Tab trong thanh Tab, thì một Tab mới là Tab “More” (thêm) dùng để truy cập các Tab còn lại sẽ xuất hiện.

3.5.3. Navigation

Navigation (điều hướng) trong Shell tương tự như NavigationPage.

Shell cung cấp trải nghiệm điều hướng dựa trên URI, có thể dễ dàng điều hướng đến bất cứ trang nào trong ứng dụng mà không cần phải tuân theo thứ bậc điều hướng của các trang, bên cạnh đó, còn cung cấp khả năng điều hướng lùi mà không cần phải đi qua tất cả các trang trong ngăn xếp điều hướng.

3.5.4. Search

Ngoài các tính năng điều hướng mạnh mẽ, Shell còn cung cấp thanh tìm kiếm được tích hợp sẵn ở phía trên cùng của màn hình. Thanh tìm kiếm này giúp người dùng dễ dàng tìm kiếm nội dung, trang hoặc chức năng trong ứng dụng một cách nhanh chóng và hiệu quả.

Khi người dùng nhập văn bản vào thanh tìm kiếm, Shell sẽ tự động gợi ý các kết quả phù hợp, giúp người dùng tìm kiếm nội dung mong muốn nhanh chóng mà không cần phải điều hướng qua nhiều trang.

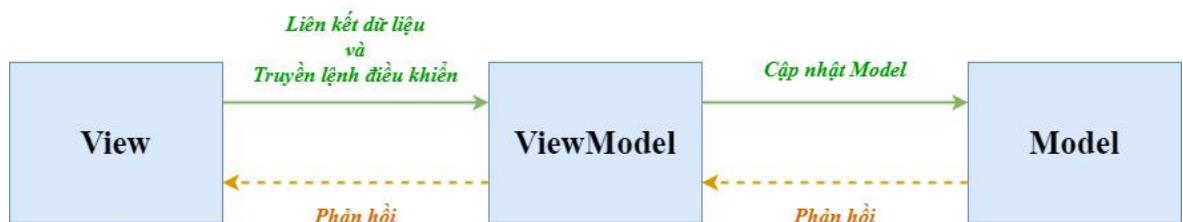
CHƯƠNG 4: MÔ HÌNH MVVM VÀ KẾT NỐI DỮ LIỆU TRONG .NET MAUI

4.1. Mô hình MVVM

Việc phát triển ứng dụng .NET MAUI thường liên quan đến việc tạo giao diện người dùng trong XAML và sau đó thêm mã code-behind để xử lý logic nghiệp vụ. Cách làm này có thể dẫn đến một số vấn đề khi ứng dụng phát triển về quy mô và phức tạp như đã đề cập tại mục 2.3.5.

Mô hình thiết kế phần mềm Model-View-ViewModel (MVVM) tách biệt rõ ràng lớp logic và nghiệp vụ của ứng dụng khỏi giao diện người dùng. Từ đó giúp việc bảo trì, kiểm thử, cũng như phát triển và nâng cấp phần mềm dễ dàng hơn. Bên cạnh đó còn cung cấp khả năng tái sử dụng lại các đoạn code, vốn là một vấn đề lớn trong ngành lập trình. Và điều quan trọng nhất mà mô hình MVVM đem lại đó cho phép các nhà lập trình và các nhà thiết kế giao diện đồ họa cộng tác và làm việc song song cũng như dễ dàng hơn.

Mô hình MVVM gồm 3 thành phần chính là lớp View, lớp Model và lớp View-Model:



Sơ đồ 4.1: Mô tả mô hình MVVM

Trước khi đi sâu vào từng thành phần của MVVM, chúng ta cũng cần phải hiểu các thành phần trên tương tác với nhau như thế nào bằng cách sử dụng mối quan hệ “biết về”:

- View “biết về” ViewModel.
- Model “biết về” ViewModel.

- View không “biết về” Model.

- Model không “biết về” View.

→ ViewModel ngăn cách View và Model, cho phép 2 thành phần này có thể phát triển riêng biệt mà không ảnh hưởng đến nhau.

4.1.1. View

View là thành phần chịu trách nhiệm hiển thị dữ liệu và tương tác với người dùng. Dưới đây là các đặc điểm chính của lớp View:

- **Hiển thị dữ liệu từ ViewModel:** View nhận dữ liệu từ ViewModel thông qua kết nối dữ liệu. Kết nối dữ liệu tự động đồng bộ hóa dữ liệu giữa ViewModel và View, giúp View luôn hiển thị dữ liệu mới nhất. Lớp này có thể sử dụng các phần tử giao diện người dùng khác nhau để hiển thị dữ liệu, ví dụ như Label, Image, ListView,...
- **Phản hồi tương tác của người dùng:** View lắng nghe các sự kiện tương tác của người dùng, ví dụ như nhập chuột, thay đổi giá trị... Khi sự kiện xảy ra, View sẽ thực hiện các hành động thích hợp, ví dụ như cập nhật dữ liệu trong ViewModel, kích hoạt Command,... View có thể sử dụng các sự kiện để truyền thông tin về tương tác của người dùng đến ViewModel.
- **Tách biệt khỏi logic nghiệp vụ:** View chỉ nên chứa code liên quan đến giao diện người dùng, không nên chứa logic xử lý dữ liệu. Logic nghiệp vụ nên được đặt trong ViewModel để đảm bảo tính tách biệt và dễ bảo trì. View chỉ nên tương tác với ViewModel thông qua kết nối dữ liệu và sự kiện. Tuy vậy, điều này không có nghĩa không được sử dụng code-behind để xử lý các tương tác của người dùng.

4.1.2. ViewModel

ViewModel là thành phần chịu trách nhiệm làm cầu nối giữa giao diện người dùng - lớp View và dữ liệu - lớp Model. Dưới đây là các đặc điểm chính của lớp ViewModel:

- **Cung cấp dữ liệu cho View:** ViewModel lấy dữ liệu từ Model và xử lý nó thành dạng phù hợp để View hiển thị. Lớp này có thể thực hiện các thao tác trên dữ liệu như lọc, sắp xếp, tổng hợp trước khi cung cấp cho View. View sử dụng data binding để kết nối với ViewModel và tự động cập nhật khi dữ liệu thay đổi.
- **Quản lý trạng thái ứng dụng:** ViewModel lưu trữ trạng thái hiện tại của ứng dụng, ví dụ như vị trí hiện tại trong danh sách, trạng thái của một nút bấm,... View có thể truy cập trạng thái ứng dụng thông qua các thuộc tính của ViewModel, khi trạng thái ứng dụng thay đổi, ViewModel sẽ cập nhật các thuộc tính liên quan và thông báo cho View để View tự động cập nhật giao diện.
- **Xử lý logic nghiệp vụ:** ViewModel có thể chứa logic nghiệp vụ liên quan đến giao diện người dùng, ví dụ như logic xác thực dữ liệu đầu vào, logic xử lý sự kiện,... ViewModel có thể tương tác với Model để truy xuất hoặc cập nhật dữ liệu khi cần thiết.
- **Cung cấp Commands cho View:** ViewModel có thể cung cấp Commands cho View để thực hiện các hành động, ví dụ như thêm, sửa, xóa dữ liệu,... View có thể sử dụng data binding để kết nối các nút, menu,... với Commands của ViewModel. Khi người dùng kích hoạt một nút hoặc menu, Command tương ứng sẽ được thực thi và ViewModel sẽ thực hiện logic xử lý hành động đó.

4.1.3. Model

Model là thành phần chịu trách nhiệm đại diện cho dữ liệu của ứng dụng. Đặc điểm chính của lớp Model là lưu trữ dữ liệu thô của ứng dụng, thường được định nghĩa dưới dạng các lớp đối tượng đơn giản. Mỗi lớp đối tượng trong Model đại diện cho một thực thể trong miền của ứng dụng, ví dụ như sản phẩm, khách hàng, đơn hàng,...

Đôi khi mô hình MVVM không cần đến lớp Model khi có mô hình dữ liệu đơn giản.

4.1.4. Các giải pháp khác

Mô hình MVVM thích hợp và được sử dụng rộng rãi vào các sản phẩm của Microsoft, trong đó có .NET MAUI. Tuy nhiên MVVM không phải là giải pháp bắt buộc và duy nhất để thiết kế các ứng dụng .NET MAUI. Những nhà lập trình vốn quen với các mô hình khác MVC, Web API vẫn có thể sử dụng những mô hình này vào ứng dụng .NET MAUI của họ, điều này cho thấy sự linh hoạt cũng như tương thích của .NET MAUI đối với các mô hình thiết kế phần mềm khác nhau.

Việc lựa chọn mô hình thiết kế phù hợp cho ứng dụng .NET MAUI phụ thuộc vào nhiều yếu tố như độ phức tạp của ứng dụng, nhu cầu của người dùng, sở thích và kinh nghiệm của nhà phát triển.

4.2. Kết nối dữ liệu

Ứng dụng .NET MAUI bao gồm nhiều trang, mỗi trang chứa các view hiển thị dữ liệu và cho phép người dùng tương tác. data binding (kết nối dữ liệu) là kỹ thuật quan trọng giúp đồng bộ hóa dữ liệu giữa view và nguồn dữ liệu cơ bản, đảm bảo giao diện người dùng luôn hiển thị thông tin chính xác và luôn cập nhật.

Trước data binding, việc đồng bộ hóa dữ liệu giữa view và nguồn dữ liệu thường được thực hiện bằng cách viết mã thủ công để xử lý các sự kiện thay đổi dữ liệu. Khi view thay đổi, dữ liệu cần được cập nhật và ngược lại. Việc viết code thủ công cho nhiều view có thể dẫn đến mã phức tạp, khó bảo trì và dễ xảy ra lỗi.

Data binding tự động hóa việc đồng bộ hóa dữ liệu giữa view và nguồn dữ liệu, giúp đơn giản hóa mã và giảm thiểu lỗi. Data binding có thể được triển khai trong XAML hoặc mã C#. Tuy nhiên, XAML phổ biến hơn vì nó giúp giảm kích thước code và dễ đọc hơn.

Data binding liên kết các thuộc tính của view với các thuộc tính của nguồn dữ liệu. Khi giá trị của một thuộc tính thay đổi, giá trị của thuộc tính liên kết cũng được cập nhật tự động. Data binding có thể là một chiều hoặc hai chiều:

- **Một chiều:** Dữ liệu chảy từ nguồn dữ liệu đến view. view hiển thị dữ liệu nhưng không thể thay đổi nó.
- **Hai chiều:** Dữ liệu có thể chảy cả hai chiều giữa view và nguồn dữ liệu. view có thể hiển thị và thay đổi dữ liệu.

Trong mô hình MVVM, ViewModel là lớp chứa dữ liệu được lấy từ Model. Data binding thường được sử dụng để liên kết View với ViewModel, giúp đảm bảo giao diện người dùng luôn hiển thị dữ liệu mới nhất từ ViewModel.

CHƯƠNG 5: PHÁT TRIỂN ỨNG DỤNG MINH HỌA CHO .NET MAUI

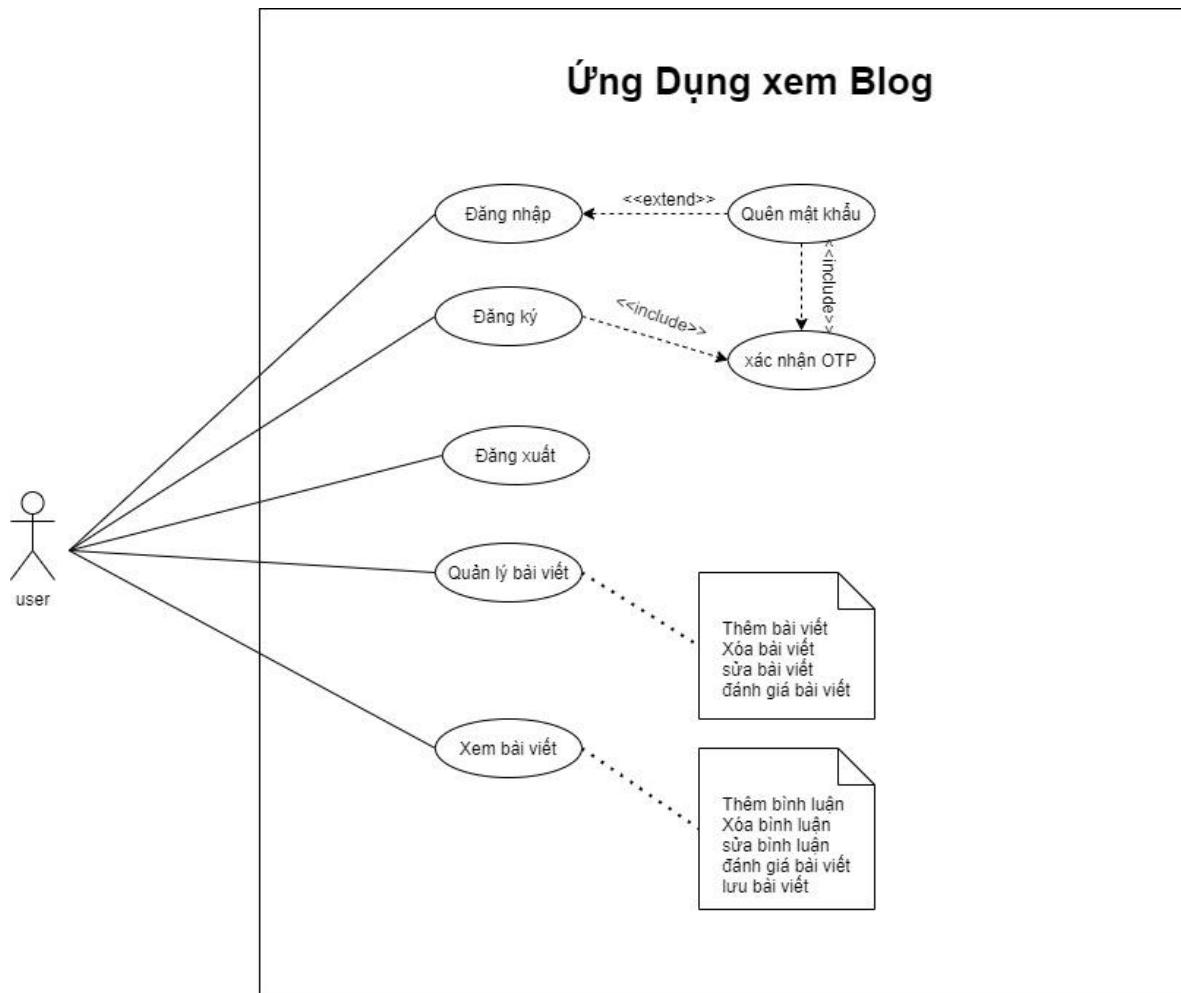
Ứng dụng minh họa cho .NET MAUI là một ứng dụng Blog, cho thấy sức mạnh và khả năng linh hoạt của .NET MAUI trong việc phát triển các ứng dụng. Ứng dụng cung cấp đầy đủ các tính năng cơ bản của một blog, mang đến cho người dùng trải nghiệm đọc và tương tác phong phú.

Chức năng chính:

- **Đăng nhập/Đăng ký:** Hệ thống xác thực người dùng an toàn, cho phép người dùng tạo tài khoản và truy cập nội dung blog.
- **Trang chủ:** Hiển thị danh sách các bài blog mới nhất hoặc theo chủ đề, giúp người dùng dễ dàng khám phá nội dung mong muốn.
- **Xem bài blog:** Mở từng bài blog, hiển thị nội dung chi tiết, hình ảnh và các thông tin liên quan.
- **Lưu (bookmark):** Cho phép người dùng lưu lại những bài blog yêu thích để truy cập nhanh chóng sau này.
- **Bình luận:** Hệ thống bình luận cho phép người dùng chia sẻ suy nghĩ và thảo luận về các bài viết.
- **Quản lý thông tin người dùng:** Người dùng có thể cập nhật thông tin cá nhân, thay đổi mật khẩu và quản lý cài đặt tài khoản.
- **Quản lý bài blog đã đăng:** Tác giả có thể đăng tải bài viết mới, chỉnh sửa nội dung, xóa bài viết và quản lý danh mục bài viết.
- **Theo dõi:** Người dùng có thể theo dõi các tác giả yêu thích.

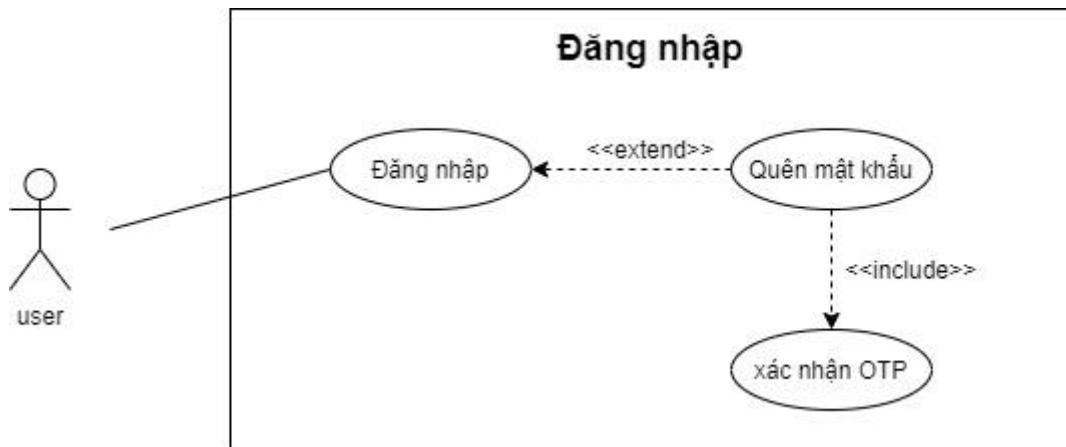
Ứng dụng được triển khai trên Android, iOS và Windows.

5.1. Sơ đồ Use-case và đặc tả cho từng chức năng



Sơ đồ 5.1: Use-case tổng thể của ứng dụng

5.1.1. Chức năng đăng nhập



Sơ đồ 5.2: Use-case đăng nhập

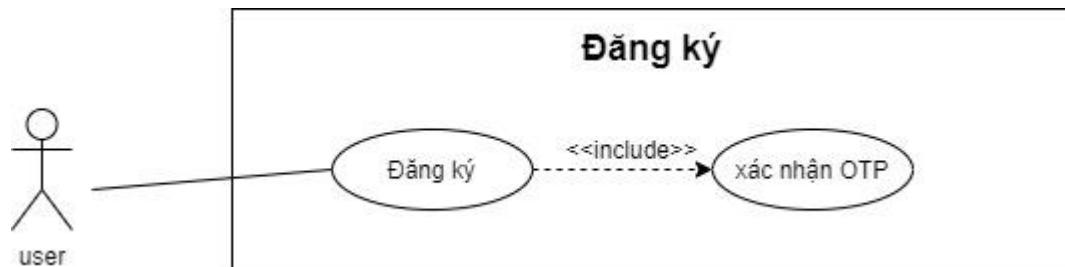
Bảng 5.1: Đặc tả chức năng đăng nhập

Use-case Name:	Đăng nhập	
Actor(s):	Người dùng (user)	
Maturity:	Focused	
Summary:	Người dùng đăng nhập vào hệ thống	
Basic Course of Events	Actor action	System Response
	1. Người dùng truy cập vào hệ thống và chọn ‘Đăng nhập’	
		2. Hệ thống hiển thị giao diện ‘Đăng nhập’
	3. Người dùng có thể chọn chức năng ‘Quên mật khẩu’ A1	
	4. Người dùng có thể ‘Đăng nhập’ A2	
	5. Người dùng có thể ‘Tạo tài	

Alternative Paths:	khoản' A3	
		6. Hệ thống thực hiện yêu cầu của người dùng
	A1	
	Actor action	System Response
	1. Người dùng nhấn vào ‘Quên mật khẩu’ trên giao diện Đăng nhập	
		2. Hệ thống hiển thị giao diện yêu cầu người dùng nhập tên tài khoản người dùng đã quên mật khẩu.
	3. Người dùng nhập tên tài khoản và nhấn ‘Yêu cầu gửi mã OTP’	
		4. Hệ thống sẽ gửi mã OTP về email của tài khoản E1.
		5. Hệ thống hiển thị giao diện xác nhận OTP
	6. Người dùng nhập mã OTP vừa nhận sau đó nhấn ‘Xác thực’ hoặc ‘Gửi lại’ E2	
		7 Hệ thống chuyển sang giao diện thay đổi mật khẩu
	8 Người dùng nhập mật khẩu mới và nhập lại mật khẩu sau đó nhấn ‘Xác nhận’ E3	
		9 Hệ thống chuyển về giao diện đăng nhập
	A2	

	Actor action	System Response
	1. Người dùng điền tên đăng nhập và mật khẩu sau đó chọn ‘Đăng nhập’ E4, E5	
		2. Hệ thống chuyển đến giao diện ‘Trang chủ’ tương ứng
A3		
	Actor action	System Response
	1 Người dùng nhấn ‘Tạo tài khoản’	
		2 Hệ thống hiển thị giao diện đăng ký tài khoản
Exception Paths:	E1 Nếu tài khoản không tồn tại hệ thống sẽ thông báo lỗi yêu cầu gửi OTP sẽ bị hủy. E2 Nếu mã OTP không chính xác hệ thống sẽ thông báo lỗi. E3 Nếu mật khẩu và xác nhận mật khẩu không khớp hệ thống sẽ thông báo lỗi. E4 Nếu tên tài khoản hoặc mật khẩu sai cú pháp sẽ báo lỗi E5 Nếu tài khoản không tồn tại hoặc tên đăng nhập, mật khẩu sai hoặc tài khoản đã bị khoá hệ thống sẽ thông báo lỗi.	
Extension Points:	Không có	
Triggers:	Người dùng muốn đăng nhập vào hệ thống	
Assumptions:	Người dùng đã thao tác các chức năng	
Preconditions:	Người dùng đã có tài khoản trên hệ thống	
Post Conditions:	Nếu đăng nhập thành công hệ thống sẽ chuyển người dùng sang giao diện trang chủ tương ứng ngược lại hệ thống sẽ không thay đổi	

5.1.2. Chức năng đăng ký



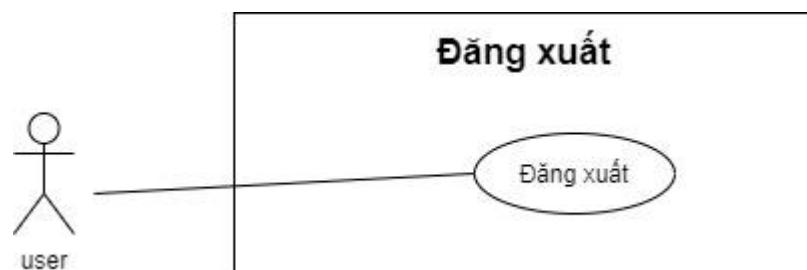
Sơ đồ 5.3: Use-case đăng ký

Bảng 5.2: Đặc tả chức năng đăng ký

Use-case Name:	Đăng ký		
Actor(s):	Người dùng (user)		
Maturity:	Focused		
Summary:	Người dùng đăng ký tài khoản mới trên hệ thống		
Basic Course of Events	Actor action	System Response	
	1. Người dùng truy cập vào hệ thống và chọn ‘Đăng ký’		
		2. Hệ thống hiển thị giao diện ‘Đăng ký tài khoản mới’	
	3. Người dùng điền đầy đủ thông tin và nhấn nút ‘Đăng ký’ để tiếp tục hoặc nhấn ‘Quay lại’ để huỷ bỏ việc tạo tài khoản E1		
		4. Hệ thống gửi mã OTP qua mail người dùng đăng ký	
		5 Hệ thống hiển thị giao diện ‘Xác nhận OTP’	

	6. Người dùng tiến hành điền OTP và nhấn ‘Xác nhận’ hoặc ‘Gửi lại’ E2	
		7 Hệ thống tiến hành tạo mới tài khoản và chuyển user đến giao diện ‘đăng nhập’
Alternative Paths:	Không có	
Exception Paths:	E1. Nếu thông tin của user điền không đầy đủ hoặc không hợp lệ hệ thống sẽ thông báo lỗi (không hợp lệ bao gồm mail bị trùng với tài khoản khác). E2. Nếu mã OTP của user điền không chính xác hệ thống sẽ thông báo lỗi	
Extension Points:	Không có	
Triggers:	Người dùng muốn đăng ký tài khoản trên hệ thống	
Assumptions:	Người dùng đã thao tác các chức năng	
Preconditions:	Không có	
Post Conditions:	Nếu đăng ký thành công tài khoản sẽ được tạo và chuyển user đến trang đăng nhập, ngược lại hệ thống sẽ không thay đổi.	

5.1.3. Chức năng đăng xuất

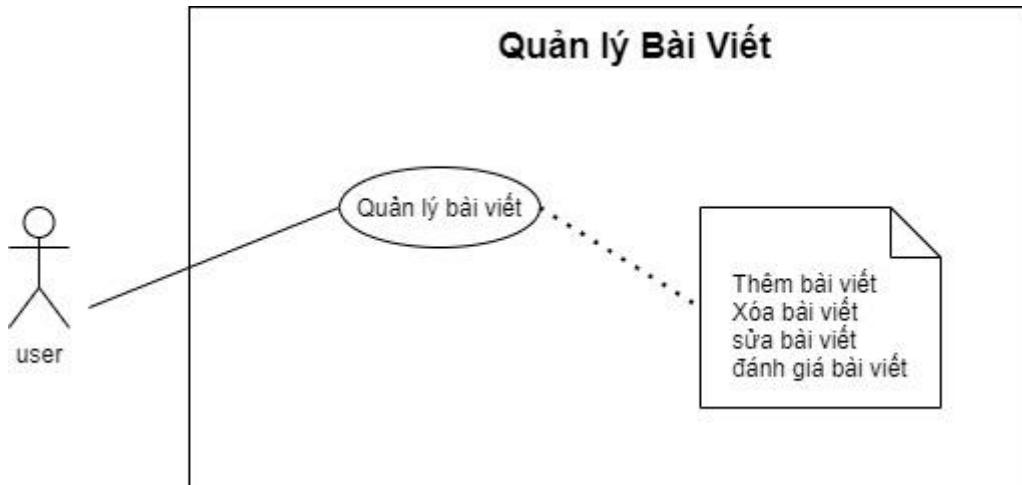


Sơ đồ 5.4: Use-case đăng xuất

Bảng 5.3: *Đặc tả chức năng đăng xuất*

Use-case Name:	Đăng xuất	
Actor(s):	Người dùng (user)	
Maturity:	Focused	
Summary:	Người dùng đăng xuất khỏi hệ thống	
Basic Course of Events	Actor action	System Response
	1. Người dùng chọn ‘Đăng xuất’	
		2. Hệ thống đăng xuất tài khoản của người dùng khỏi hệ thống và chuyển về ‘Trang đăng nhập’
Alternative Paths:	Không có	
Exception Paths:	Không có	
Extension Points:	Không có	
Triggers:	Người dùng muốn đăng xuất khỏi hệ thống	
Assumptions:	Người dùng đã thao tác các chức năng	
Preconditions:	Người dùng đã đăng nhập vào hệ thống	
Post Conditions:	Không có	

5.1.4. Chức năng quản lý bài viết



Sơ đồ 5.5: Use-case quản lý bài viết

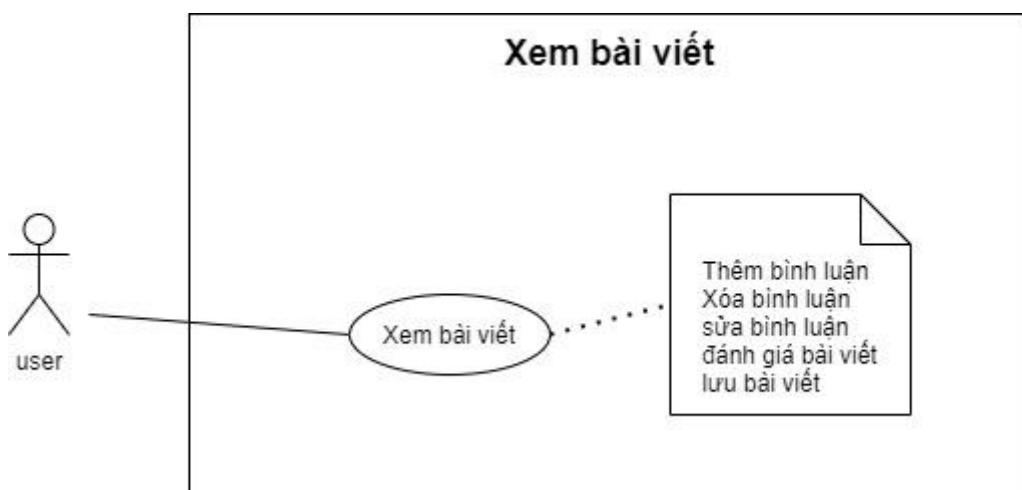
Bảng 5.4: Đặc tả chức năng quản lý bài viết

Use-case Name:	Quản lý bài viết	
Actor(s):	Người dùng (user)	
Maturity:	Focused	
Summary:	Người dùng quản lý các bài viết trong hệ thống	
Basic Course of Events	Actor action 1. Người dùng truy cập vào trang cá nhân 2. Người dùng có thể chọn chức năng ‘thêm bài viết’ A1 3. Người dùng có thể ‘sửa bài viết’ A2 4. Người dùng có thể ‘xóa bài viết’ A3	System Response 5. Hệ thống thực hiện yêu cầu của

		người dùng
	A1	
	Actor action	System Response
	1. Người dùng nhấn vào ‘Thêm bài viết’ trên giao diện trong trang cá nhân	
		2. Hệ thống hiển thị mẫu bài viết, yêu cầu người dùng nhập thông tin
	3. Người dùng nhập đầy đủ thông tin và nhấn ‘đăng bài’	
		4. Hệ thống sẽ kiểm tra thông tin thêm E1.
	A2	
Alternative Paths:	Actor action	System Response
	1. Người dùng nhấn vào ‘Sửa bài viết’ trên 1 bài viết trong giao diện trong trang cá nhân	
		2. Hệ thống hiển thị mẫu bài viết với thông tin đã có, yêu cầu người dùng nhập thông tin cần chỉnh sửa
	3. Người dùng nhập đầy đủ thông tin cần chỉnh sửa và nhấn ‘lưu chỉnh sửa’	
		4. Hệ thống sẽ kiểm tra thông tin sửa E2.
	A3	
	Actor action	System Response
	1. Người dùng nhấn vào ‘Xóa bài viết’ trên 1 bài viết trong	

	giao diện trong trang cá nhân	
		2. Hệ thống hiển thị yêu cầu xác nhận
	3. Người dùng xác nhận yêu cầu	
Exception Paths:	E1 Nếu thông tin bài đăng được thêm không hợp lệ hệ thống sẽ thông báo lỗi. E2 Nếu thông tin bài đăng được chỉnh sửa không hợp lệ hệ thống sẽ thông báo lỗi.	
Extension Points:	Không có	
Triggers:	Người dùng xem hoặc điều chỉnh các bài viết trong trang cá nhân	
Assumptions:	Người dùng đã thao tác các chức năng	
Preconditions:	Người dùng đã chọn vào xem trang cá nhân	
Post Conditions:	Lưu những thay đổi vào CSDL và hiển thị thay đổi cho người dùng biết	

5.1.5. Chức năng xem bài viết



Sơ đồ 5.6: Use-case xem bài viết

Bảng 5.4: Đặc tả chức năng xem bài viết

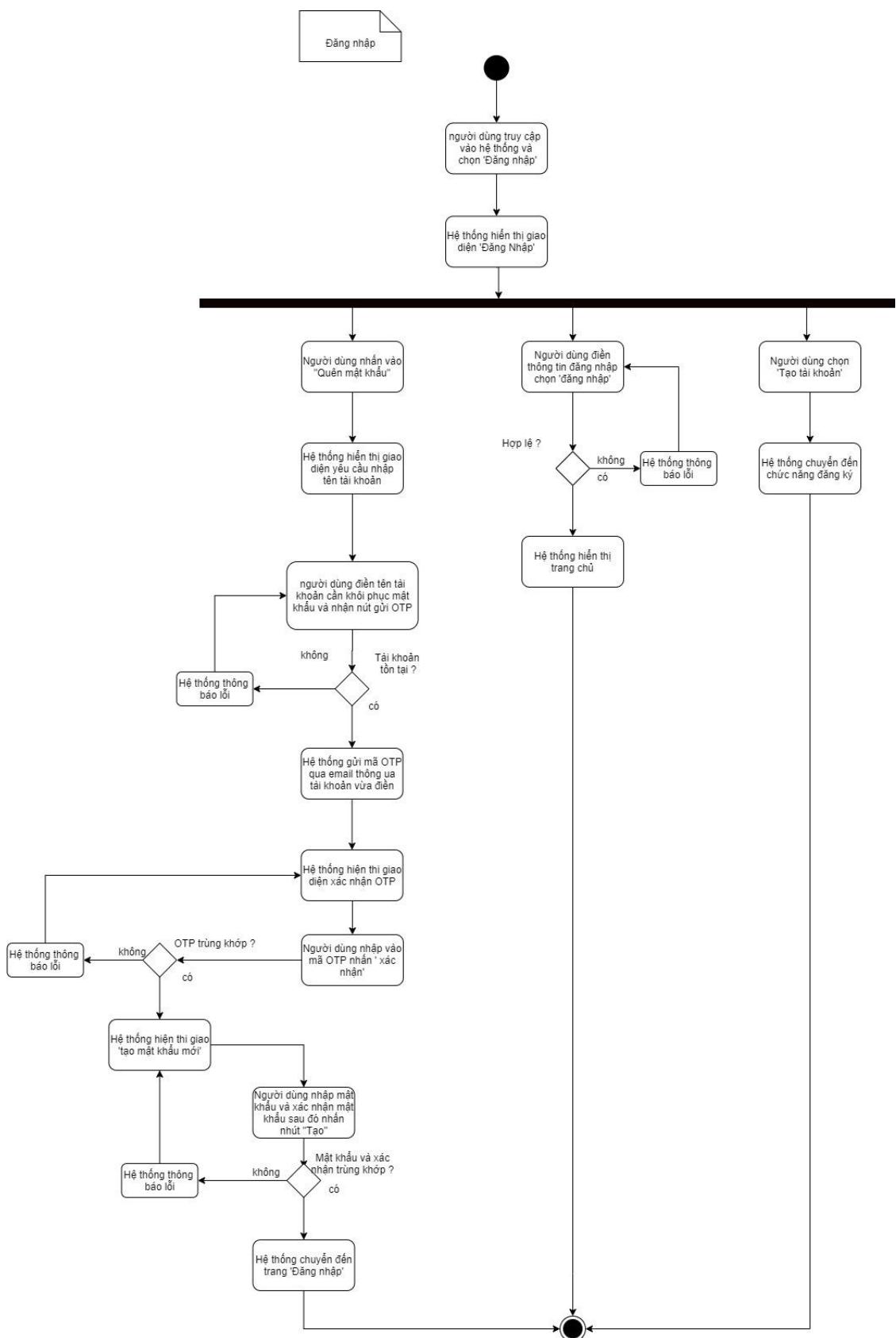
Use-case Name:	Xem bài viết	
Actor(s):	Người dùng (user)	
Maturity:	Focused	
Summary:	Người dùng xem bài viết	
Basic Course of Events	Actor action	System Response
	1. Người dùng chọn 1 bài viết để xem ở trang chủ	
		2. Hệ thống cho xem chi tiết đầy đủ của bài viết
	3. Người dùng có thể chọn chức năng ‘Thêm bình luận’ A1	
	4. Người dùng có thể ‘Sửa bình luận’ trước đó của họ A2	
	5. Người dùng có thể ‘xóa bình luận’ trước đó của họ A3	
	6. Người dùng có thể ‘đánh giá bài viết’ A4	
	7. Người dùng có thể ‘lưu bài viết’ không phải người dùng đăng A5	
Alternative Paths:	A1	
	Actor action	System Response
	1. Người dùng nhấn vào ‘Thêm bình luận’	

		2. Hệ thống hiển thị ô bình luận cho người dùng nhập thông tin
	3. Người dùng nhập đầy đủ thông tin và nhấn ‘đăng bình luận’	
		4. Hệ thống sẽ kiểm tra thông tin thêm E1.
A2		
	Actor action	System Response
	1. Người dùng nhấn vào ‘Sửa bình luận’ trước đó của họ trên bài viết	
		2. Hệ thống hiển thị ô bình luận với thông tin đã có, yêu cầu người dùng nhập thông tin cần chỉnh sửa
	3. Người dùng nhập đầy đủ thông tin cần chỉnh sửa và nhấn ‘lưu chỉnh sửa’	
		4. Hệ thống sẽ kiểm tra thông tin sửa E2.
A3		
	Actor action	System Response
	1. Người dùng nhấn vào ‘Xóa bình luận’ trước đó của họ trên bài viết	
		2. Hệ thống hiển thị thông báo yêu cầu xác nhận
	3. Người dùng xác nhận yêu cầu	

	A4	
	Actor action	System Response
	1. Người dùng chọn ‘Đánh giá bài viết’ (thích hoặc không thích)	
		2. Hệ thống hiển thị thông báo cảm ơn
	A5	
	Actor action	System Response
	1. Người dùng có thẻ ‘lưu bài viết’ không phải người dùng đăng	
		2. Hệ thống lưu bài viết này vào bộ sưu tập của người dùng.
Exception Paths:	E1 Nếu thông tin bình luận được thêm không hợp lệ hệ thống sẽ thông báo lỗi. E2 Nếu thông tin bình luận được chỉnh sửa không hợp lệ hệ thống sẽ thông báo lỗi.	
Extension Points:	Không có	
Triggers:	Người dùng muốn xem bài viết và thực hiện các lựa chọn	
Assumptions:	Người dùng đã thao tác các chức năng	
Preconditions:	Người dùng đã chọn vào xem chi tiết bài viết	
Post Conditions:	Lưu những thay đổi vào CSDL và hiển thị thay đổi cho người dùng biết	

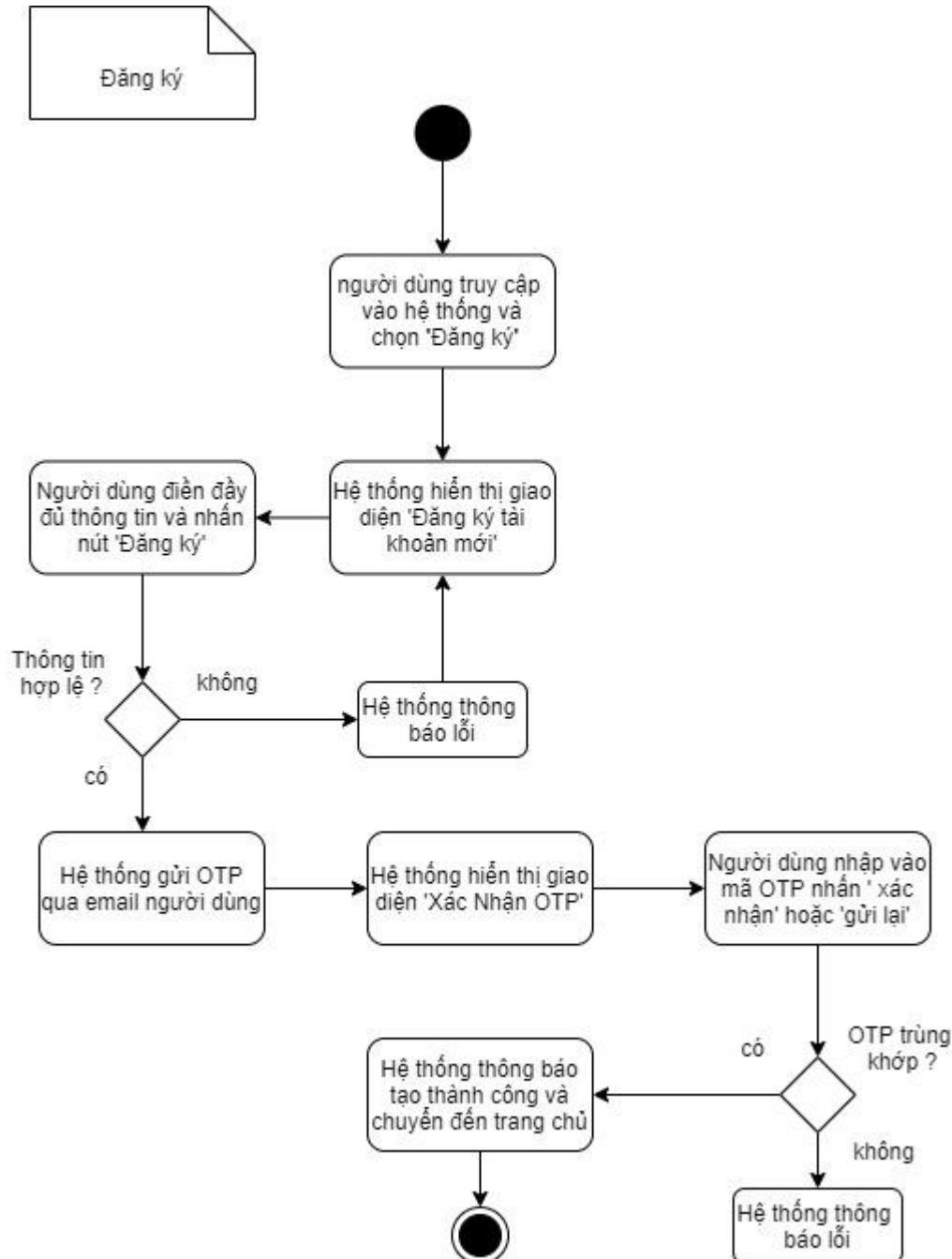
5.2. Sơ đồ hoạt động (Activity Diagram)

5.2.1. Chức năng đăng nhập



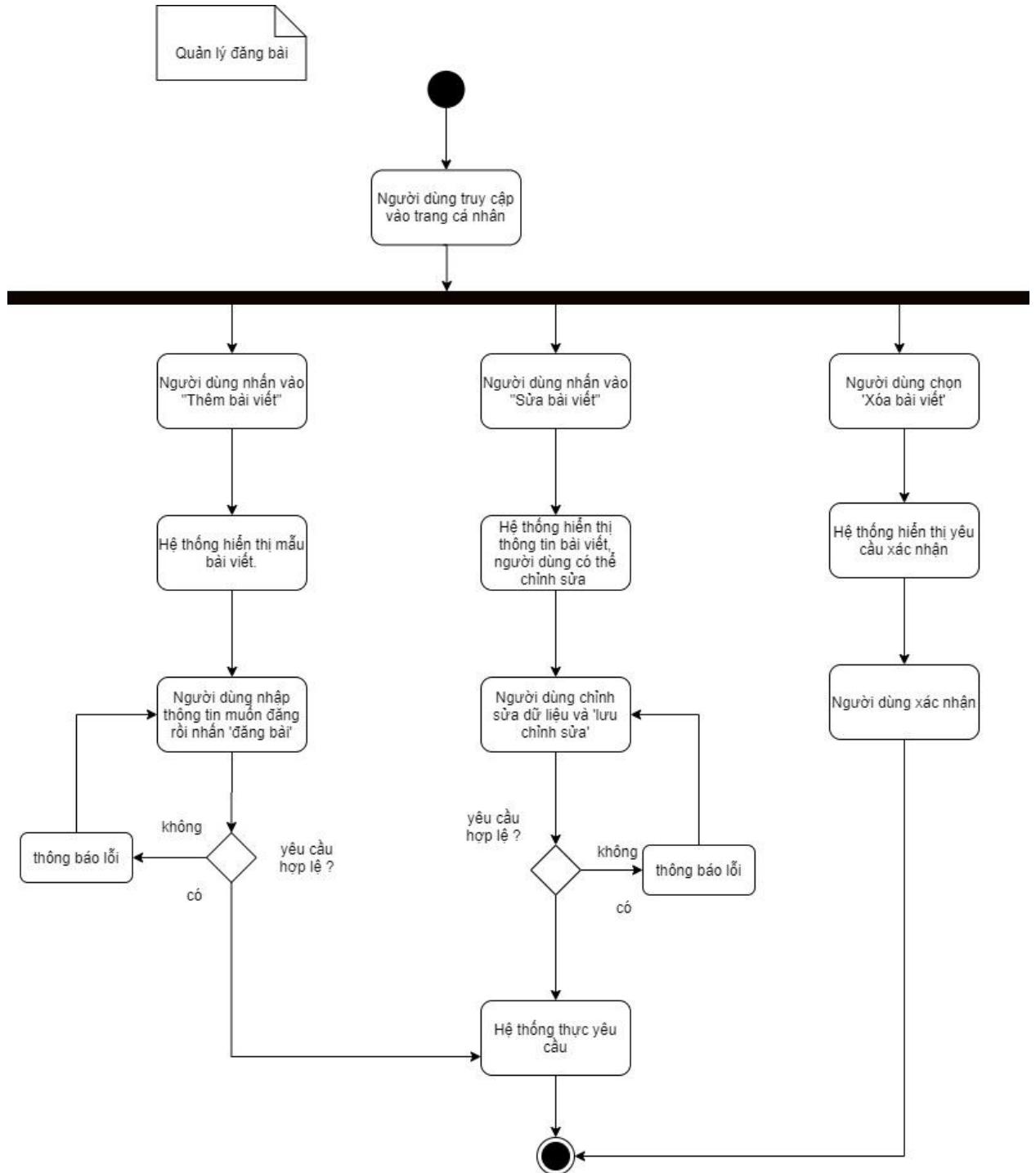
Sơ đồ 5.7: Sơ đồ hoạt động của chức năng đăng nhập

5.2.2. Chức năng đăng ký



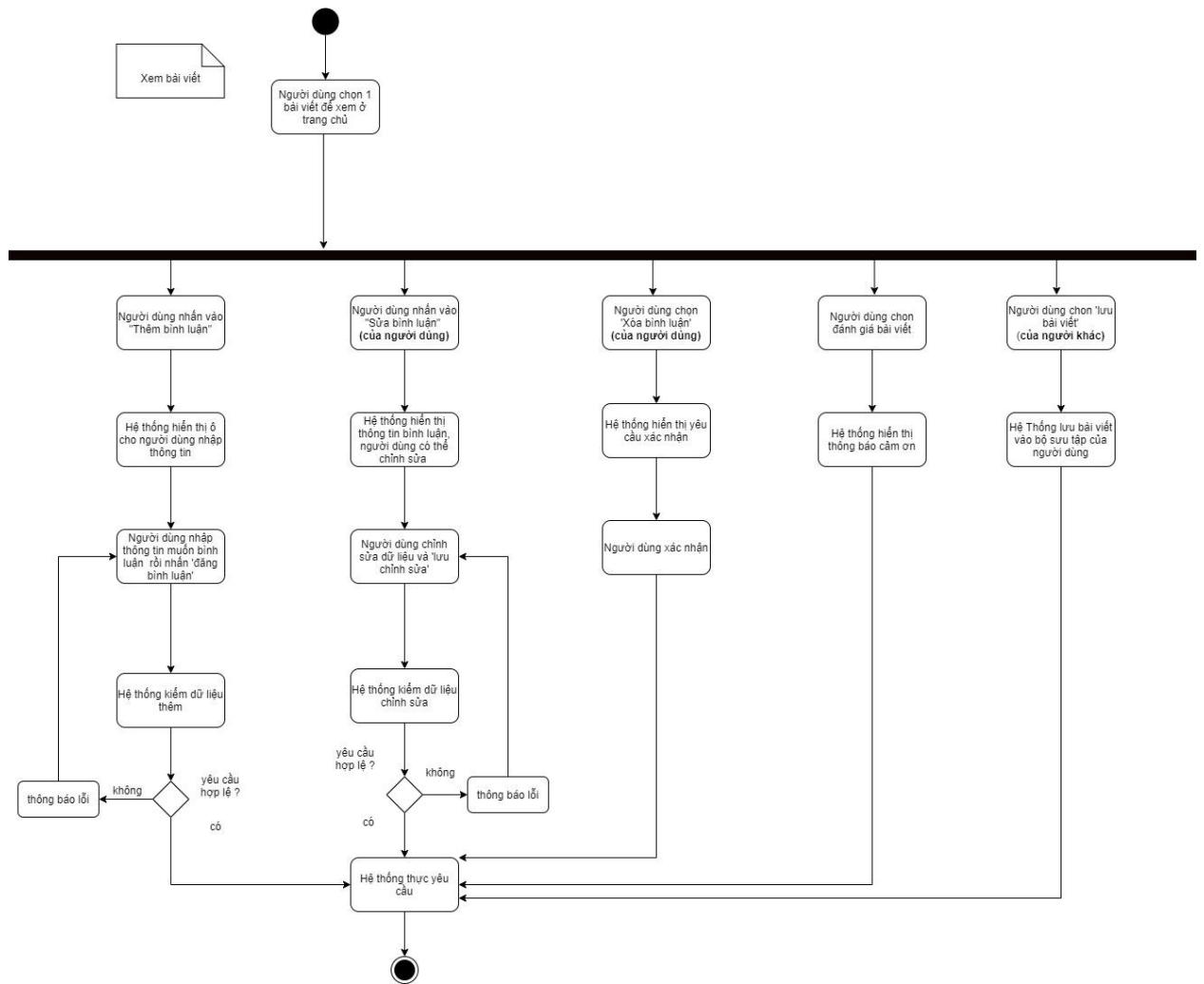
Sơ đồ 5.8: Sơ đồ hoạt động của chức năng đăng ký

5.2.3. Chức năng quản lý bài viết



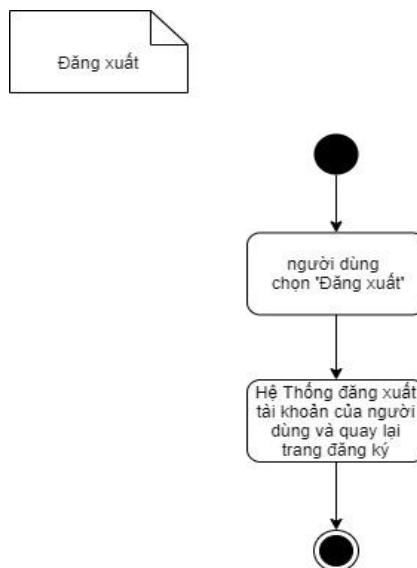
Sơ đồ 5.9: Sơ đồ hoạt động của chức năng quản lý bài viết

5.2.4. Chức năng xem bài viết



Sơ đồ 5.10: Sơ đồ hoạt động của chức năng xem bài viết

5.2.5. Sơ đồ hoạt động của chức năng đăng xuất

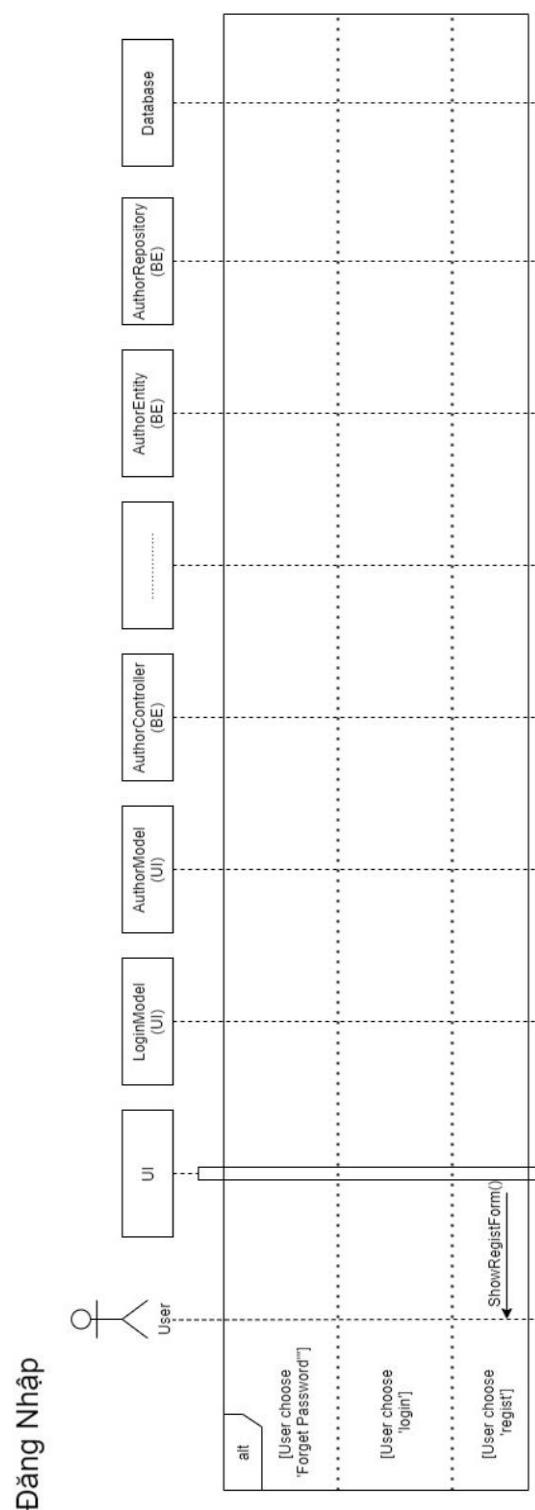


Sơ đồ 5.11: Sơ đồ hoạt động của chức năng đăng xuất

5.3. Sơ đồ tuần tự

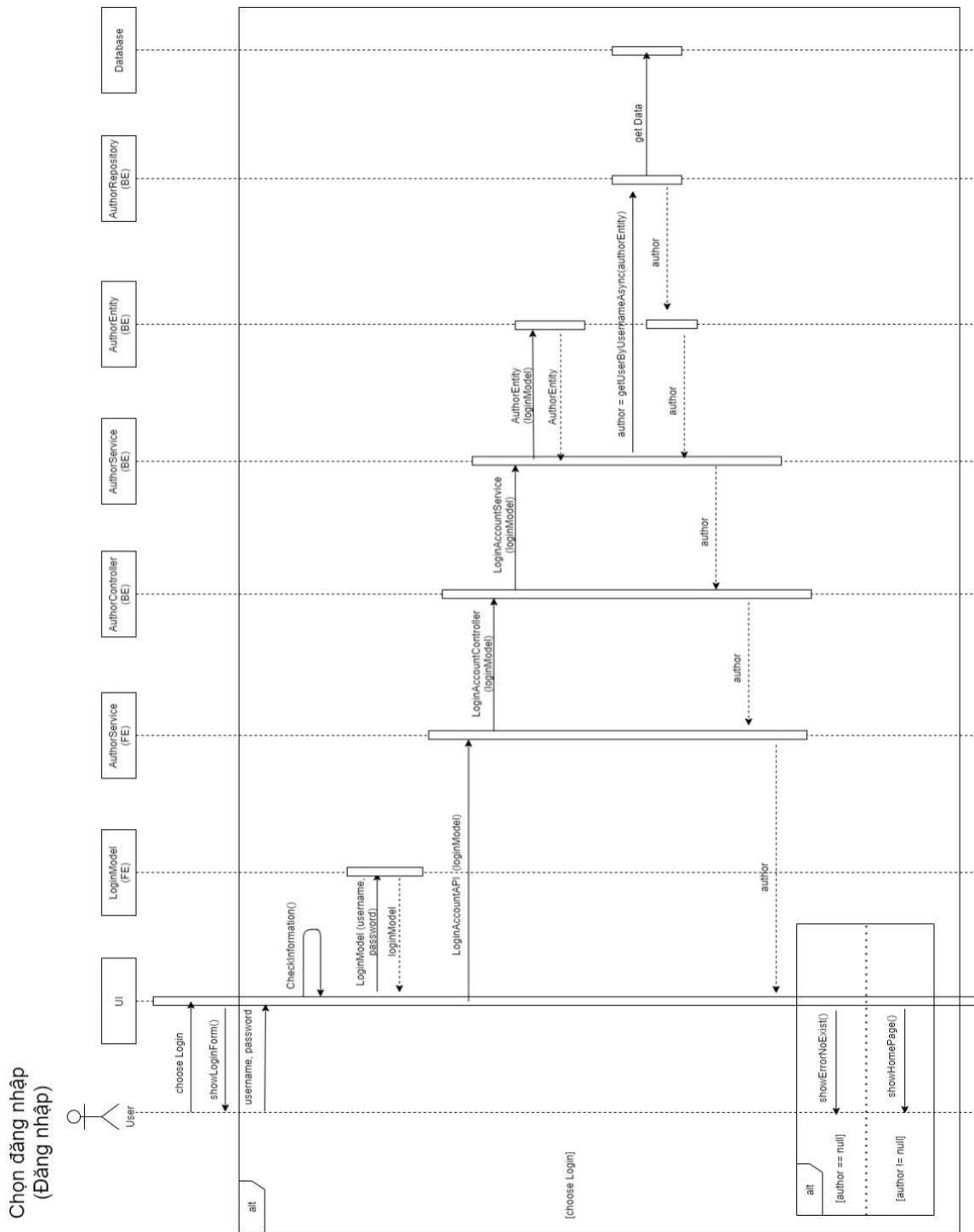
5.3.1. Chức năng đăng nhập

Do chức năng đăng nhập có nhiều bước xử lý và nhiều lựa chọn nên ta có tổng thể sơ đồ tuần tự chức năng đăng nhập như sau.



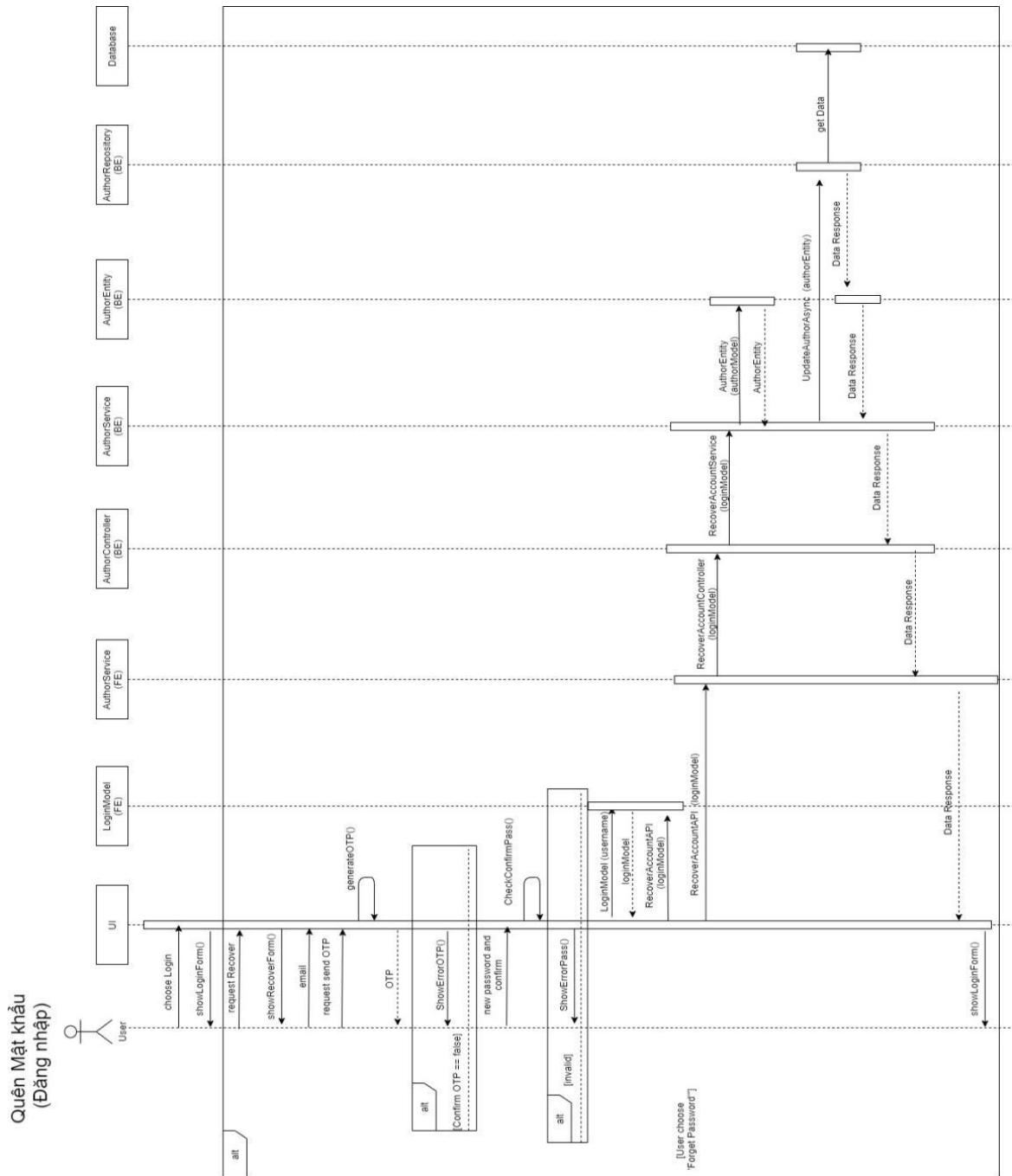
Sơ đồ 5.12: Sơ đồ tuần tự tổng thể của chức năng đăng nhập

5.3.1.1. Lựa chọn đăng nhập thẳng



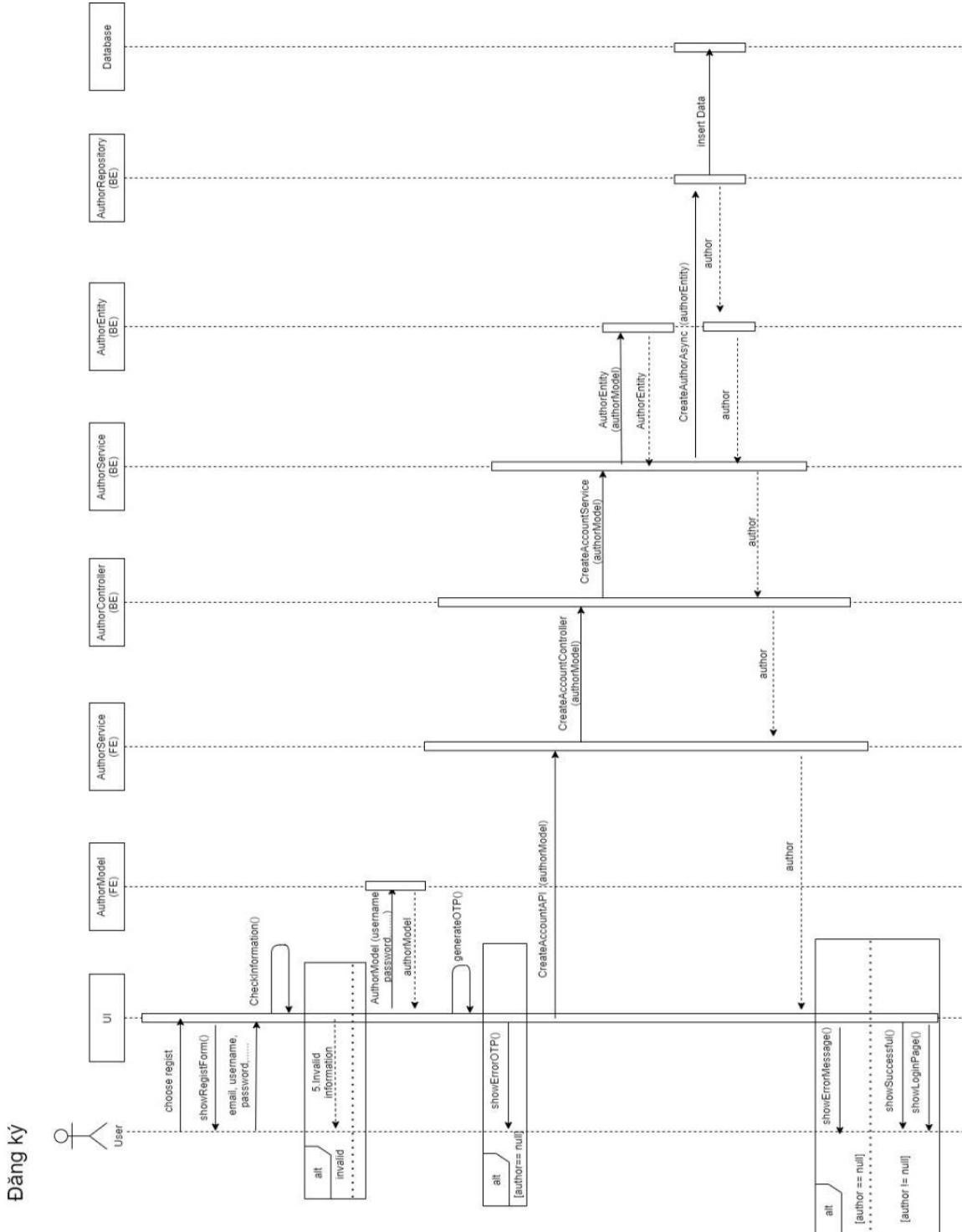
Sơ đồ 5.13: Sơ đồ tuần tự của lựa chọn đăng nhập thẳng

5.3.1.2. Lựa chọn quên mật khẩu



Sơ đồ 5.14: Sơ đồ tuần tự của lựa chọn quên mật khẩu

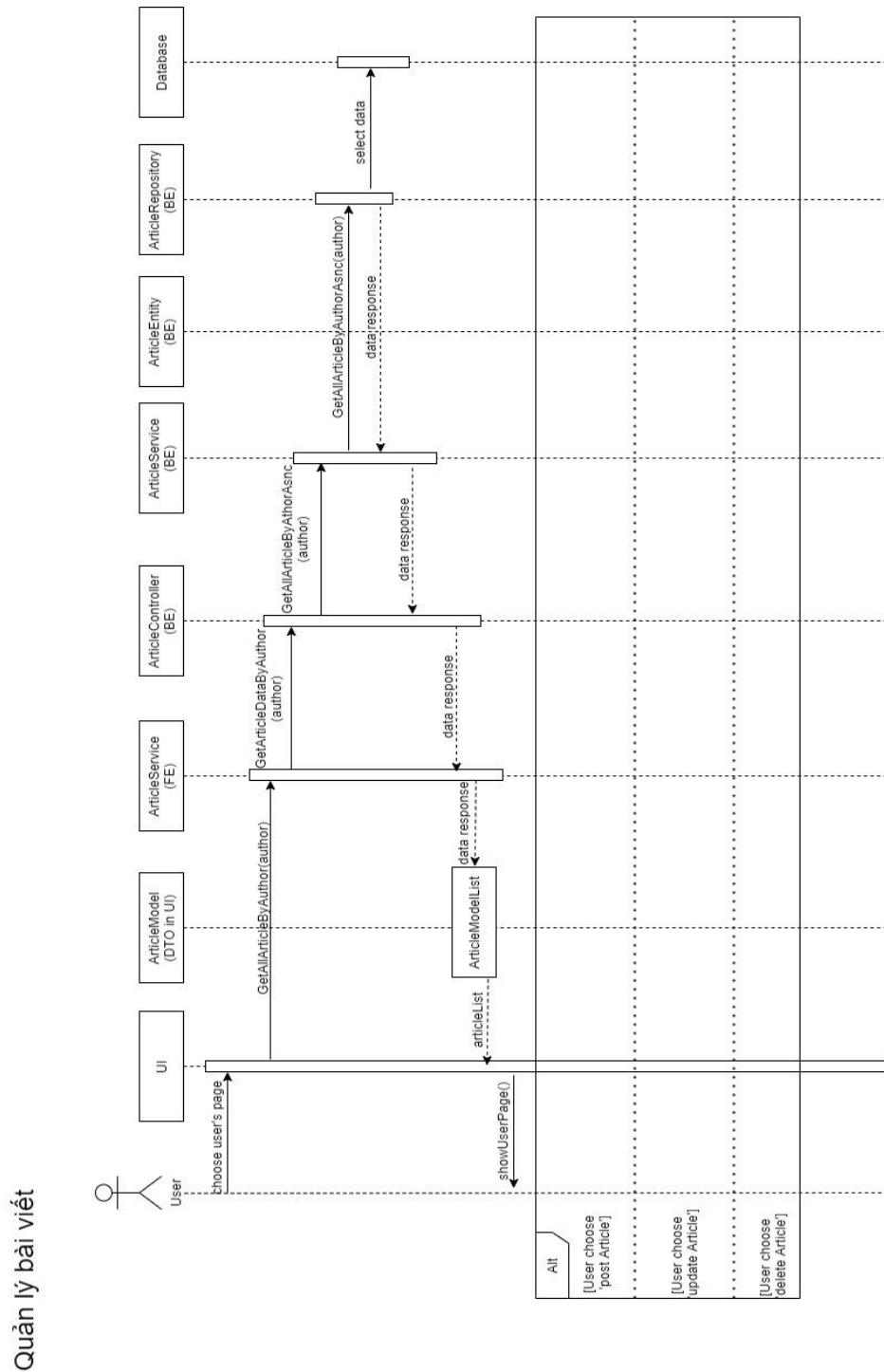
5.3.2. Chức năng đăng ký



Sơ đồ 5.15: Sơ đồ tuần tự của chức năng đăng ký

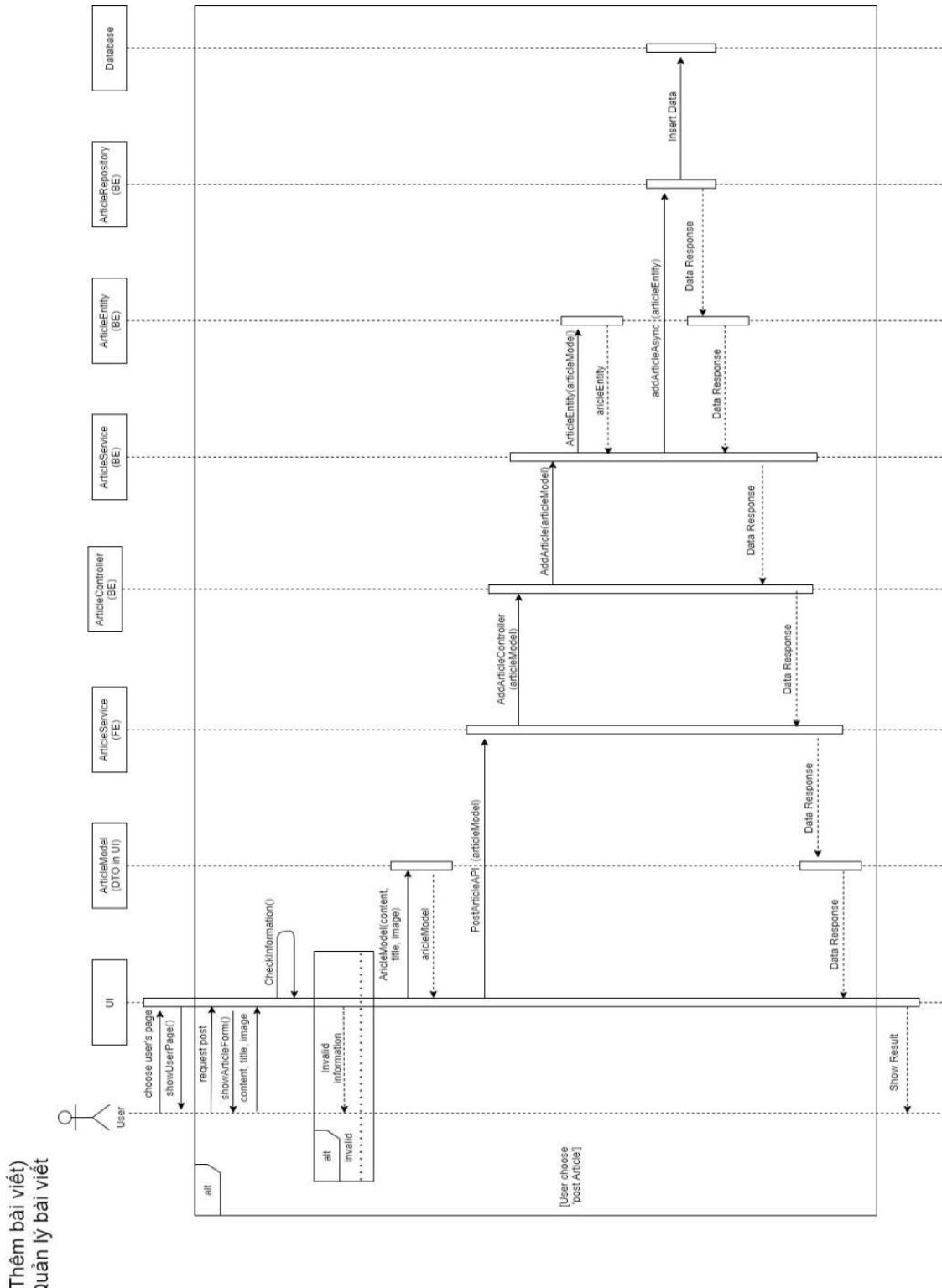
5.3.3. Chức năng quản lý bài viết

Do chức năng quản lý bài viết có nhiều bước xử lý và nhiều lựa chọn nên ta có tổng thể sơ đồ tuần tự chức năng quản lý bài viết như sau.



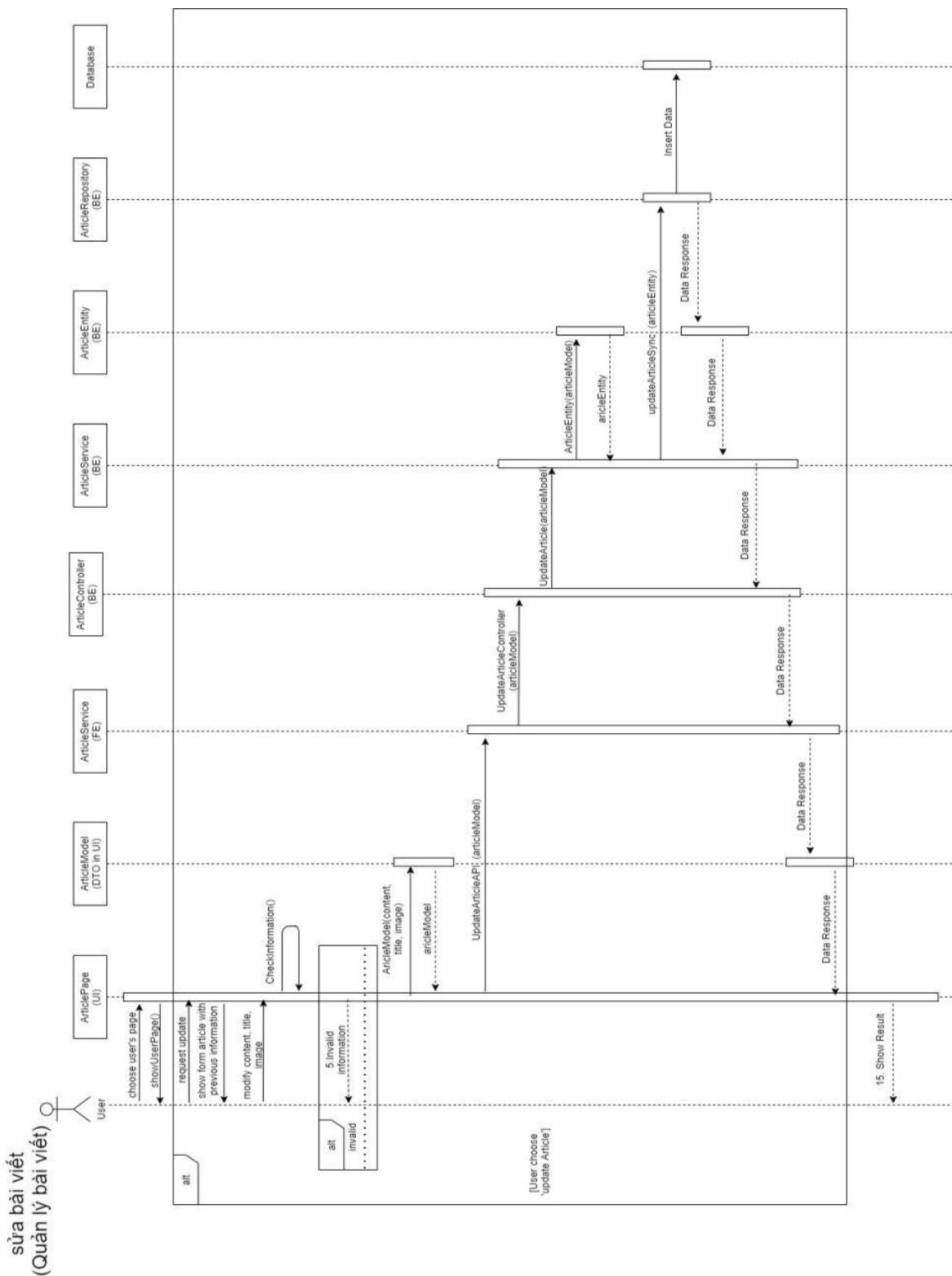
Sơ đồ 5.16: Sơ đồ tuần tự tổng thể của chức năng quản lý bài viết

5.3.3.1. Lựa chọn đăng bài viết



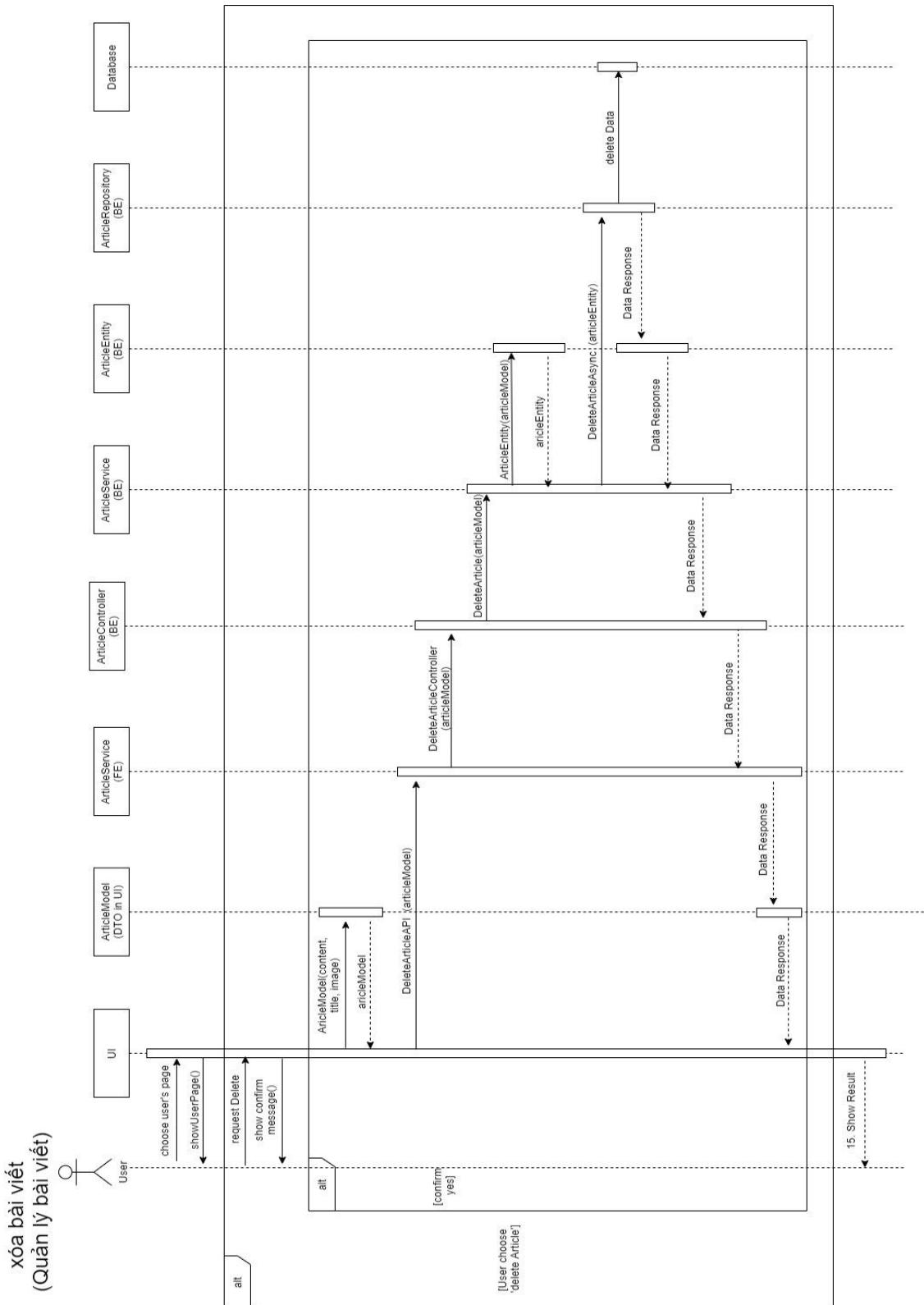
Sơ đồ 5.17: Sơ đồ tuần tự của lứa chon đăng bài viết

5.3.3.2. Lựa chọn sửa bài viết



Sơ đồ 5.18: Sơ đồ tuần tự của lựa chọn sửa bài viết

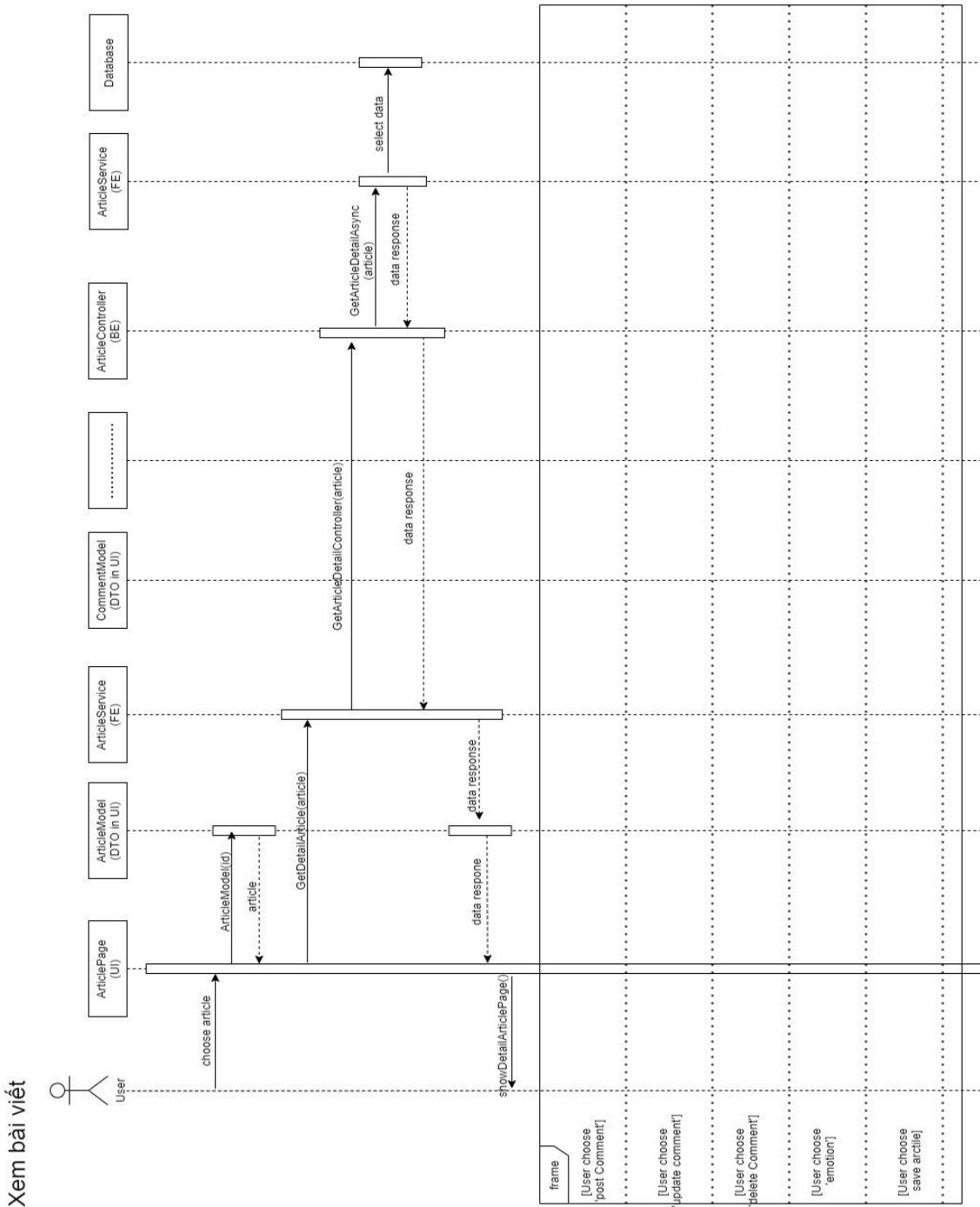
5.3.3.3. Lựa chọn xóa bài viết



Sơ đồ 5.19: Sơ đồ tuần tự của lựa chọn xóa bài viết

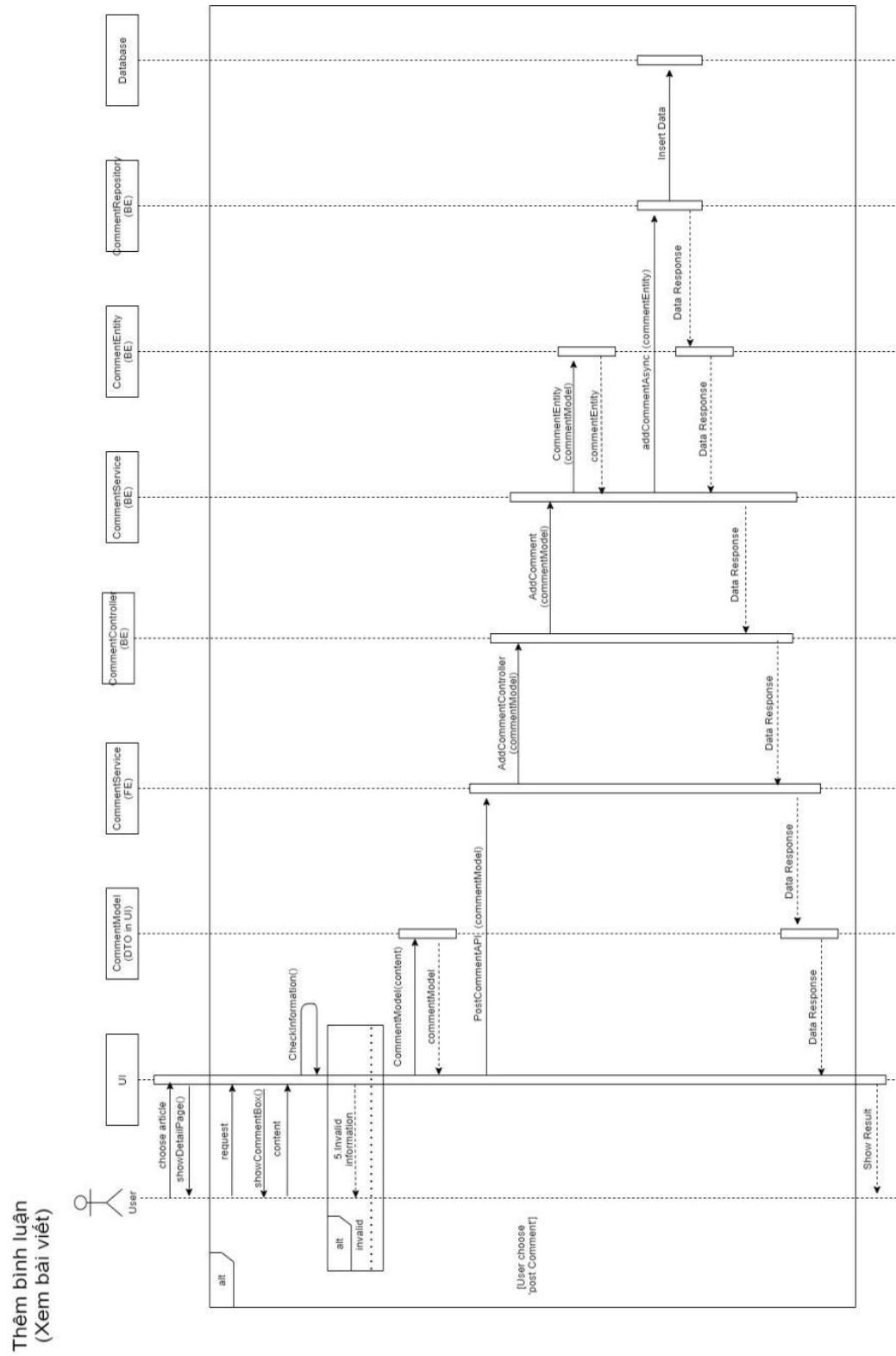
5.3.4. Chức năng xem bài viết

Do chức năng xem bài viết có nhiều bước xử lý và nhiều lựa chọn nên ta có tổng thể sơ đồ tuần tự chức năng xem bài viết như sau.



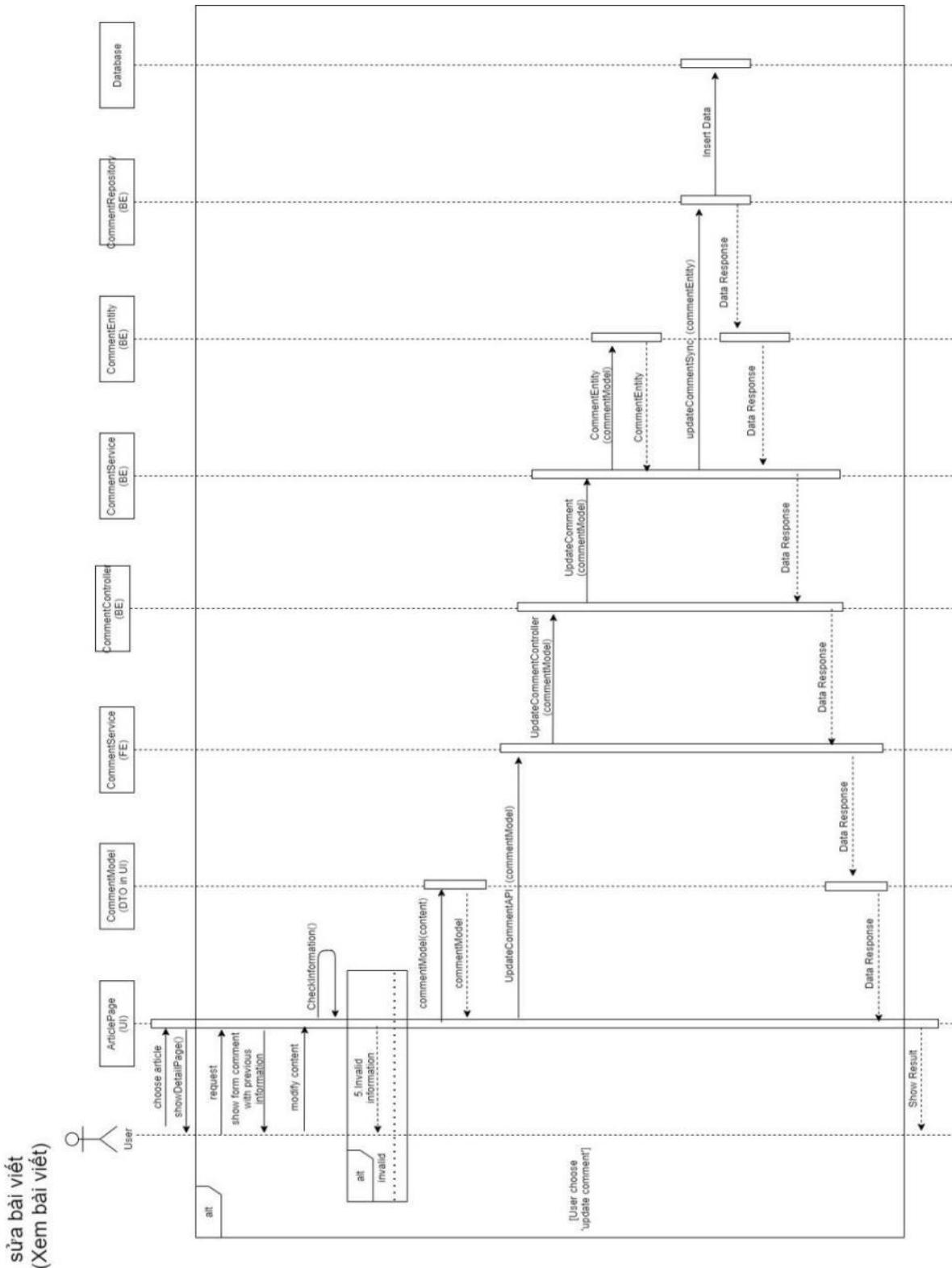
Sơ đồ 5.20: Sơ đồ tuần tự tổng thể của chức năng xem bài viết

5.3.4.1. Lựa chọn thêm bình luận



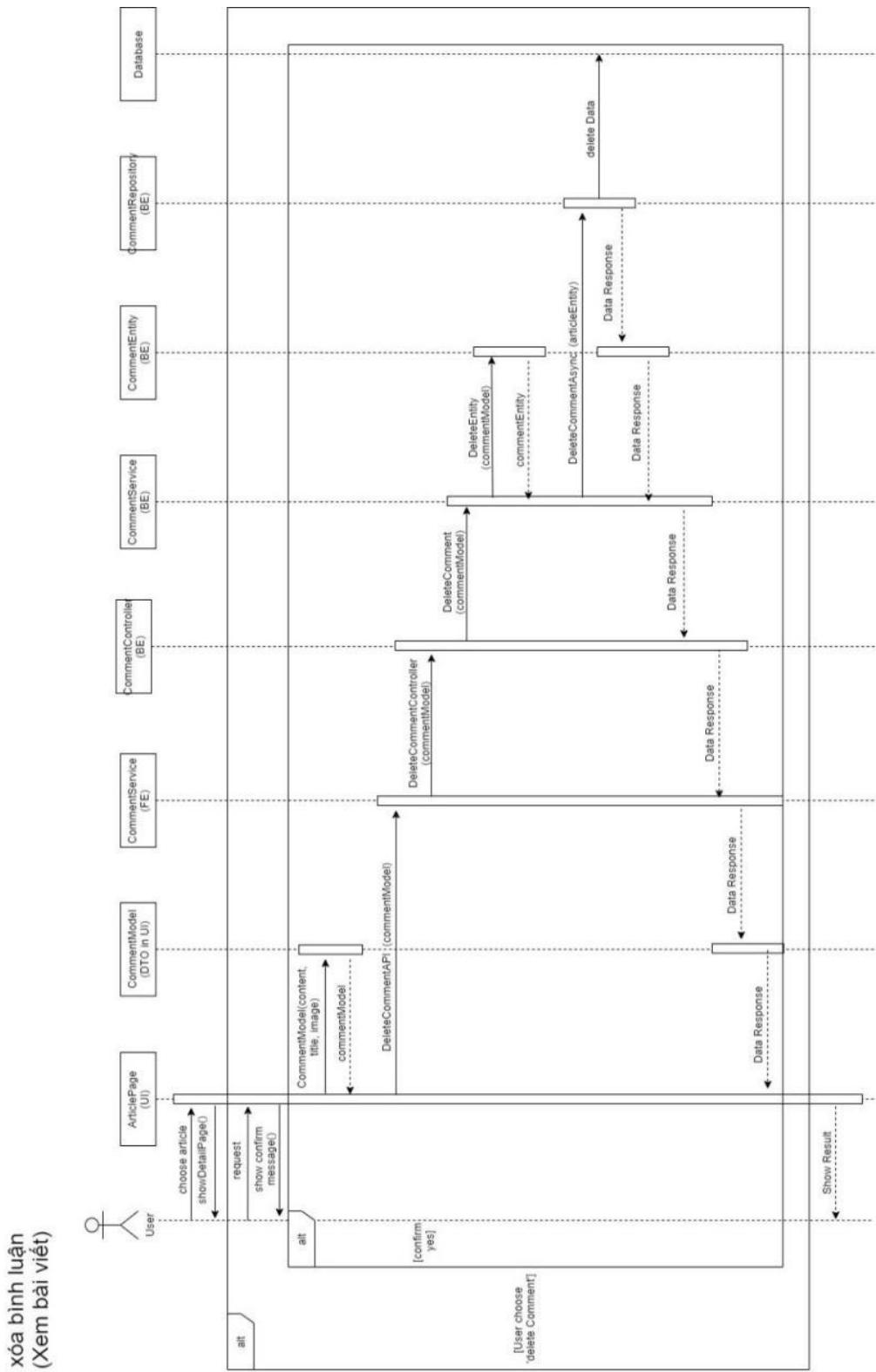
Sơ đồ 5.21: Sơ đồ tuần tự của lựa chọn thêm bình luận

5.3.4.2. Lựa chọn sửa bình luận



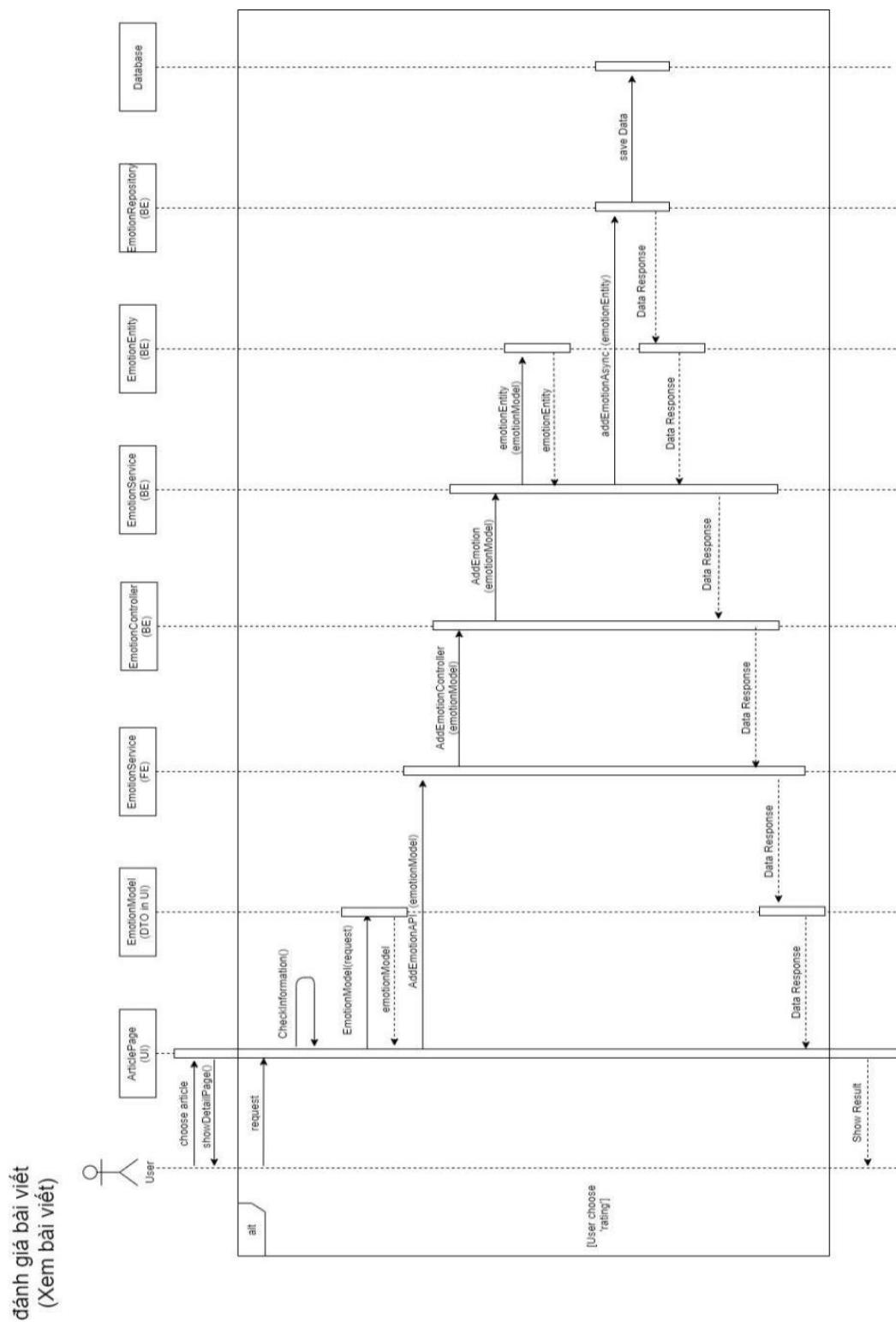
Sơ đồ 5.22: Sơ đồ tuần tự của lựa chọn sửa bình luận

5.3.4.3. Lựa chọn xóa bình luận



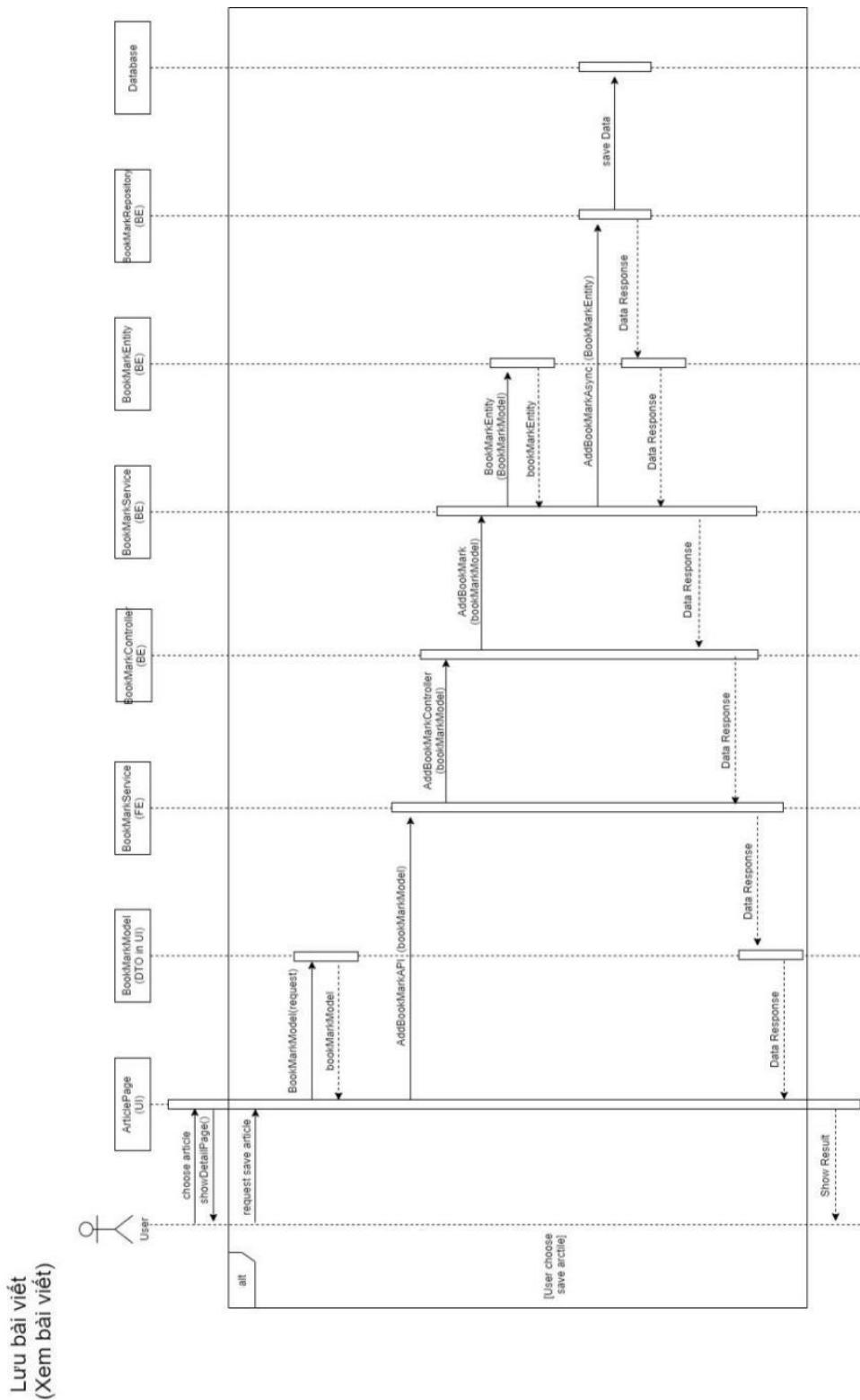
Sơ đồ 5.23: Sơ đồ tuần tự của lựa chọn xóa bình luận

5.3.4.4. Lựa chọn đánh giá bài viết



Sơ đồ 5.24: Sơ đồ tuần tự của lựa chọn đánh giá bài viết

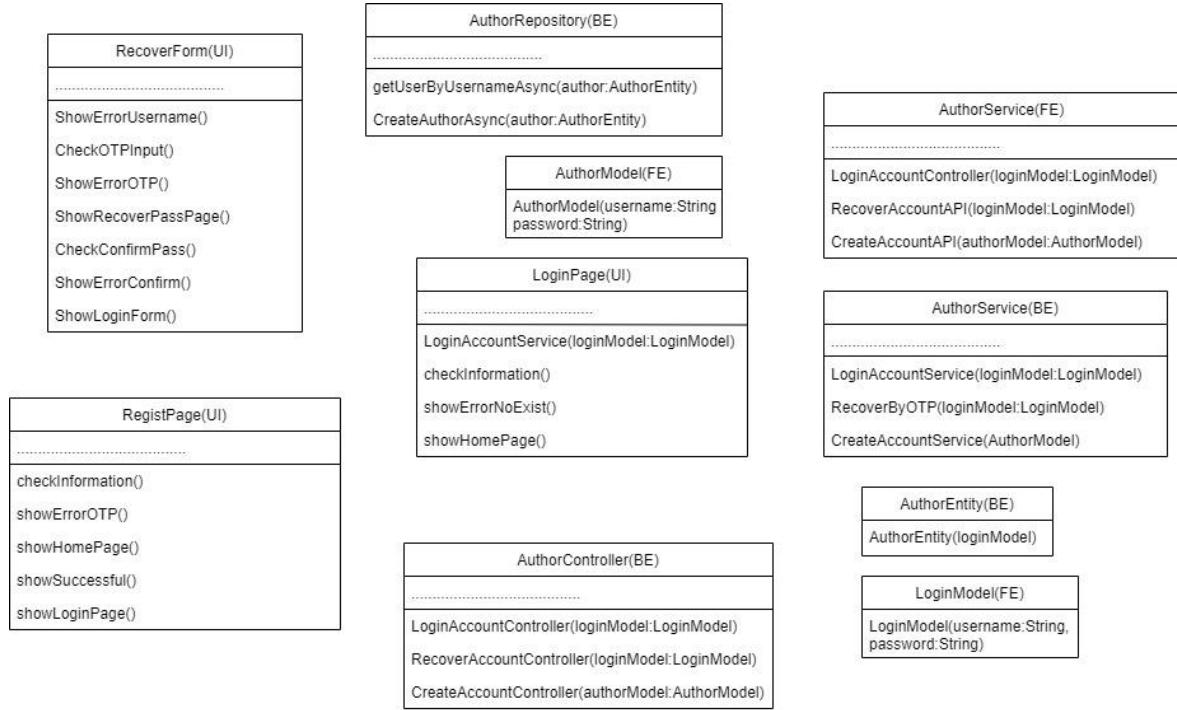
5.3.4.5. Lựa chọn lưu bài viết



Sơ đồ 5.25: Sơ đồ tuần tự của lựa chọn lưu bài viết

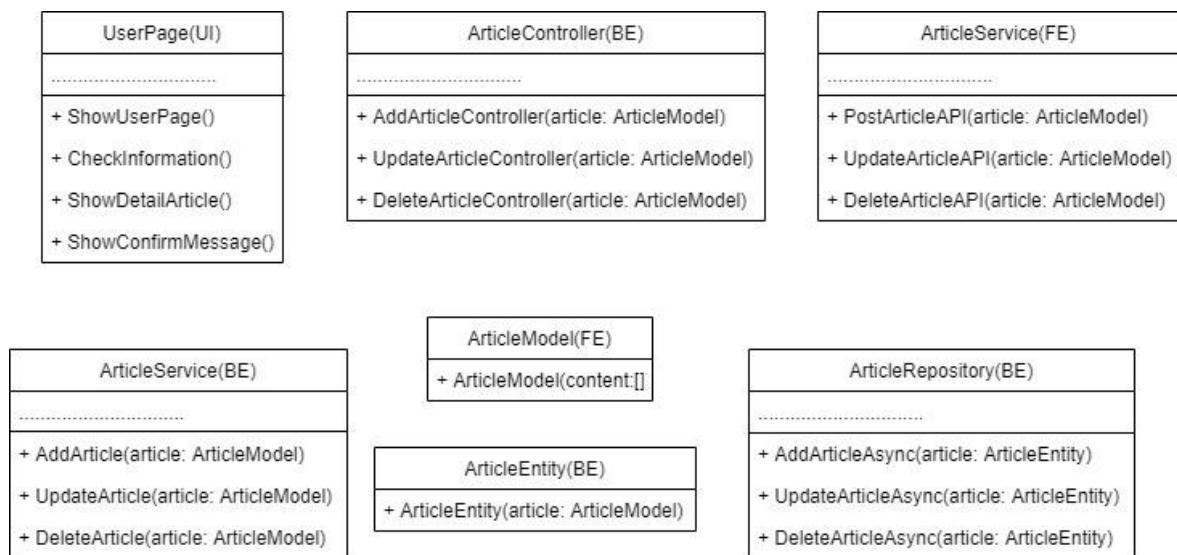
5.4. Sơ đồ lớp sau phân tích thiết kế

5.4.1. Đăng nhập và đăng ký



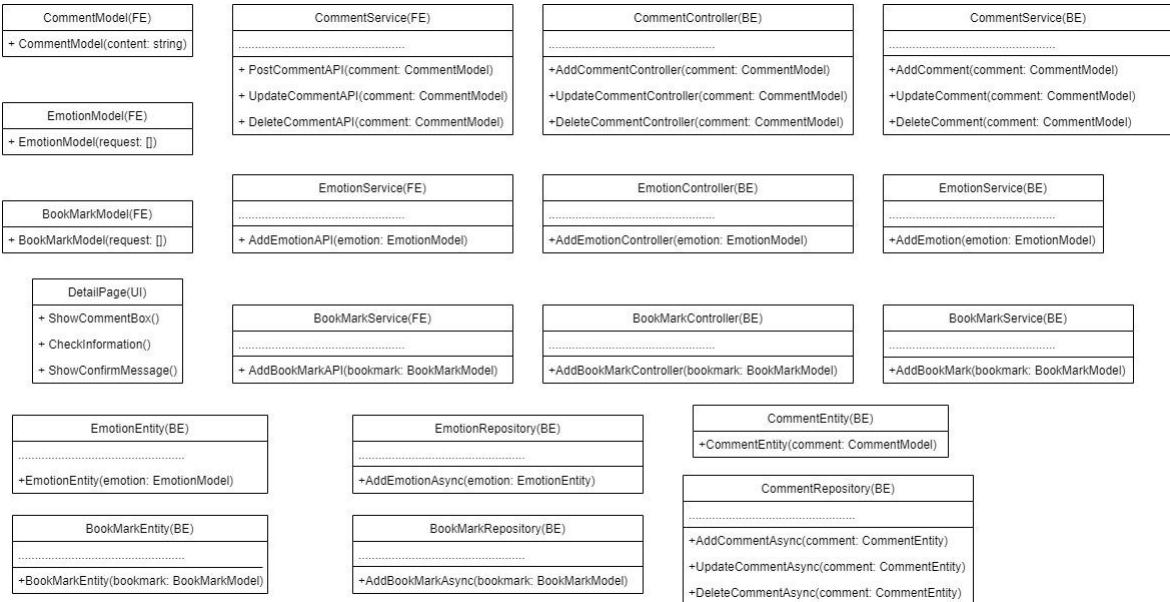
Sơ đồ 5.26: Sơ đồ lớp thu được trong phân tích chức năng đăng nhập và đăng ký

5.4.2. Chức năng quản lý bài viết



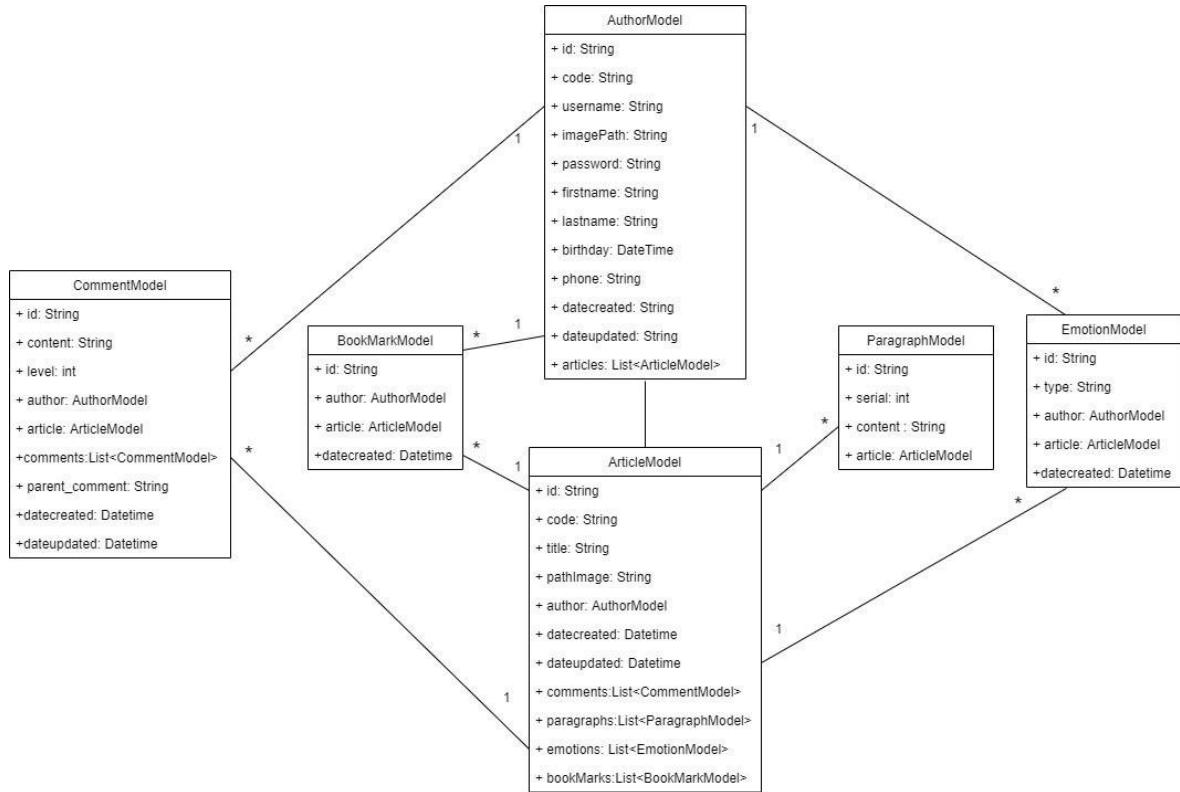
Sơ đồ 5.27: Sơ đồ lớp thu được trong phân tích chức năng quản lý bài viết

5.4.3. Chức năng xem bài viết



Sơ đồ 5.28: Sơ đồ lớp thu được trong phân phân tích chức năng xem bài viết

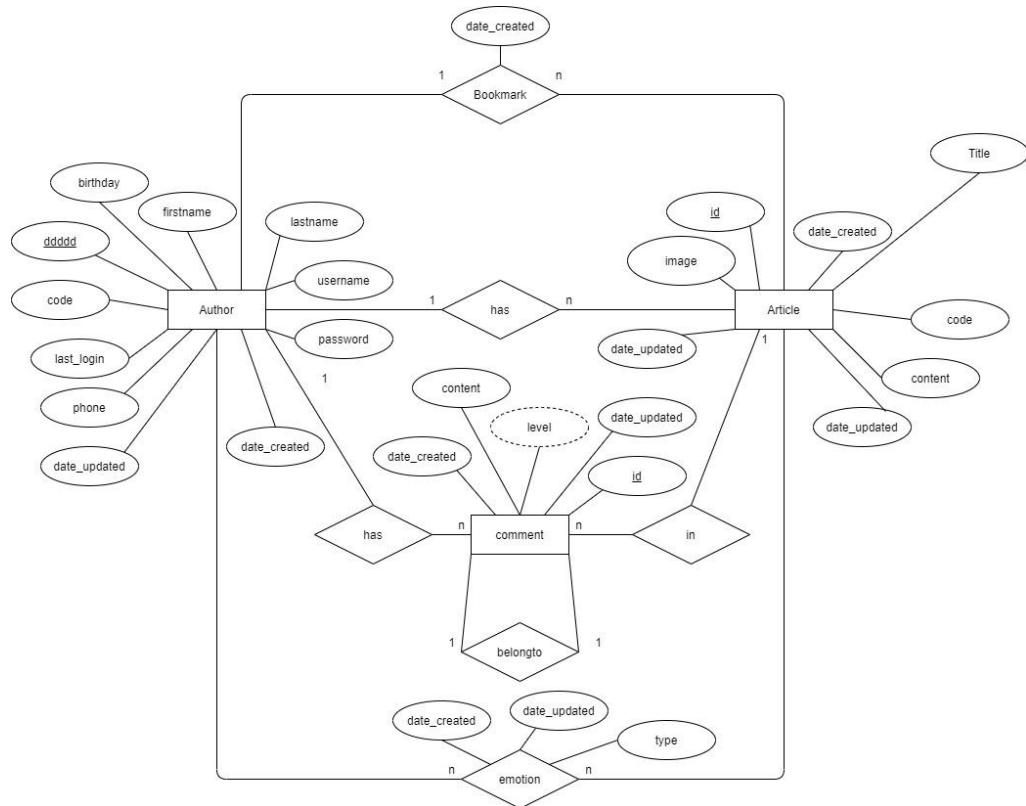
5.4.4. Các lớp DTO



Sơ đồ 5.29: Sơ đồ lớp DTO trong ứng dụng

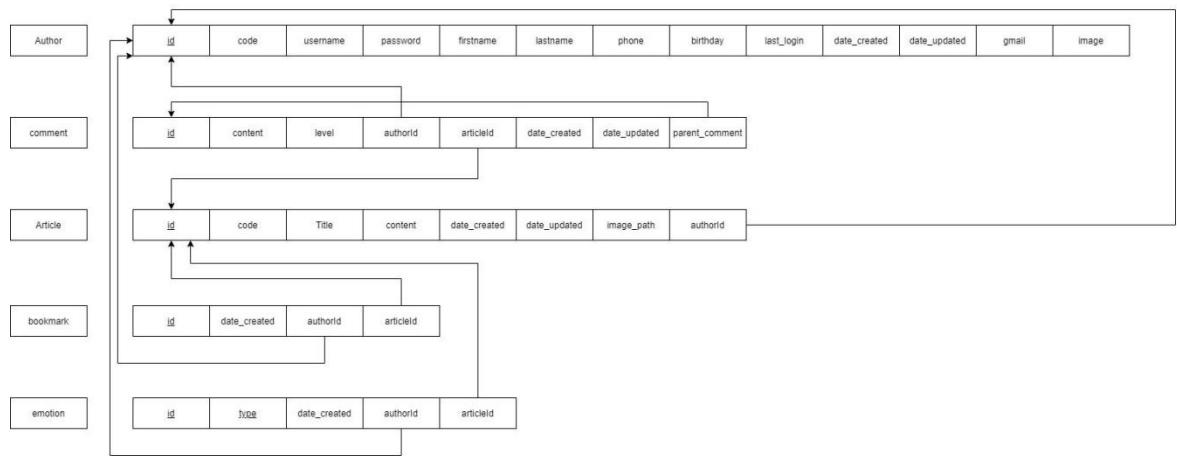
5.5. Bản thiết kế cho cơ sở dữ liệu

5.5.1. Sơ đồ thực thể kết hợp



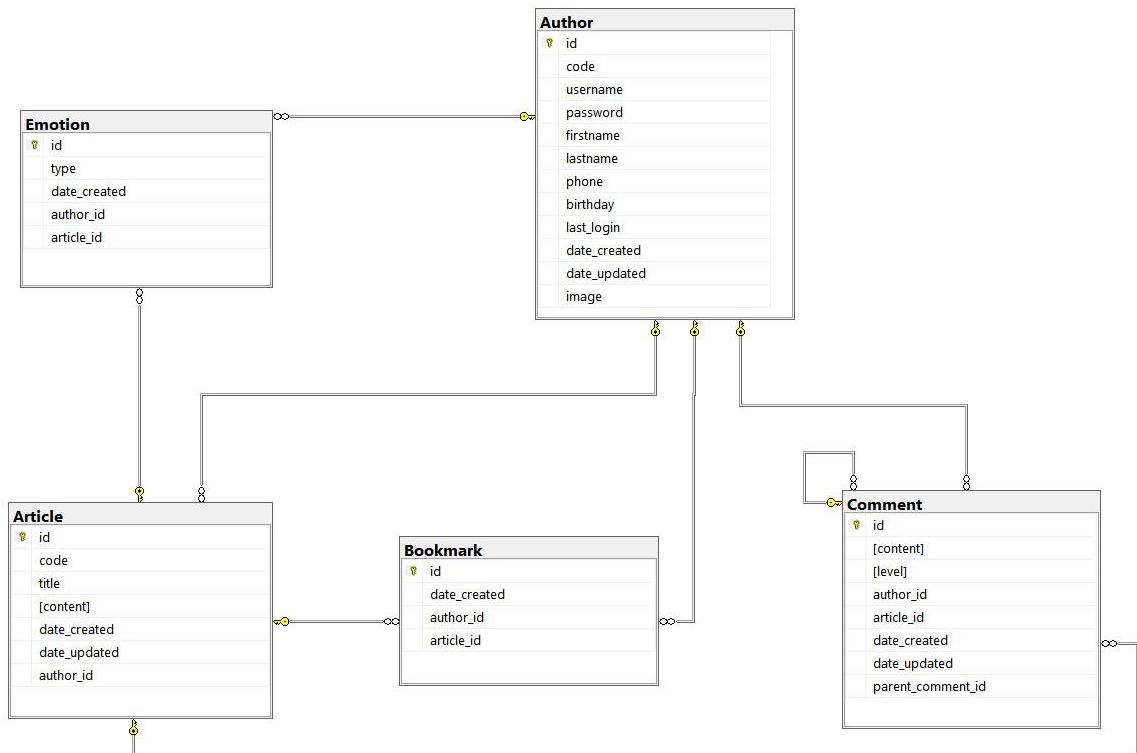
Sơ đồ 5.30: Sơ đồ thực thể kết hợp

5.5.2. Sơ đồ ảnh xạ quan hệ các thực thể trong sơ đồ thực thể kết hợp



Sơ đồ 5.31: Sơ đồ ảnh xạ quan hệ các thực thể trong sơ đồ thực thể kết hợp

5.5.3. Cài đặt cơ sở dữ liệu

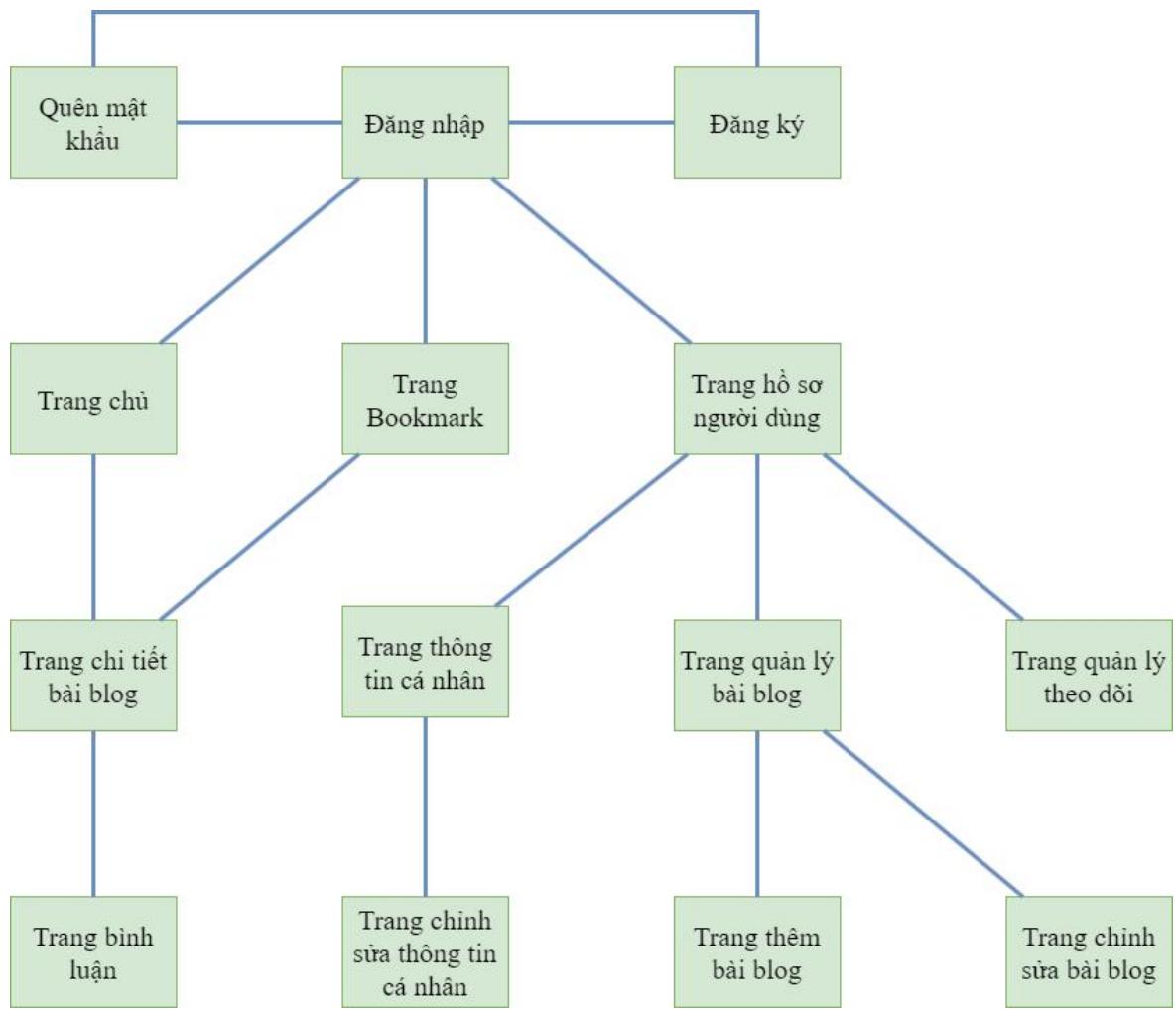


5.6. Các công nghệ áp dụng

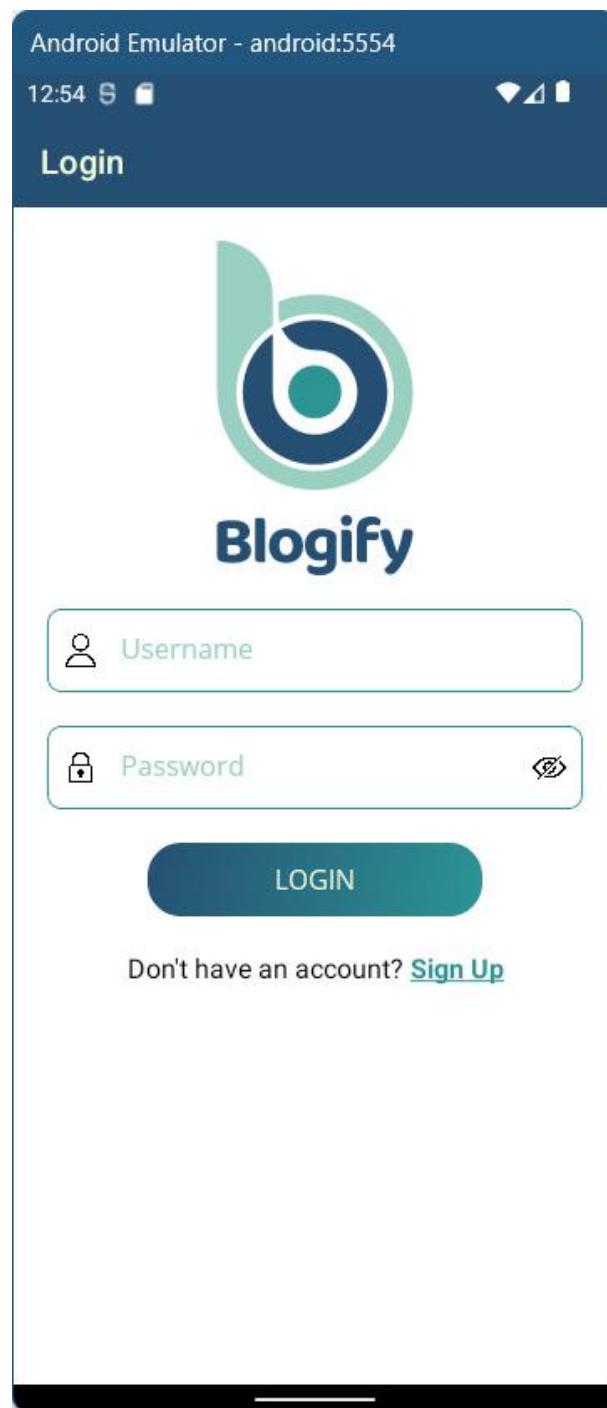
- .NET MAUI:** Sử dụng XAML để thiết kế giao diện người dùng trực quan và linh hoạt. Bên cạnh đó, còn sử dụng Shell thay thế cho các trang truyền thống, cung cấp trải nghiệm điều hướng nhất quán và dễ sử dụng cho ứng dụng, Shell bao gồm các thành phần như thanh điều hướng, thanh tab và flyout, giúp dễ dàng tổ chức và điều hướng giữa các trang trong ứng dụng. Ngôn ngữ lập trình chính cho ứng dụng là C#, được sử dụng để viết logic ứng dụng, xử lý dữ liệu, tương tác với giao diện người dùng và thực hiện các tác vụ khác.

- **Web API .NET Core:** Cung cấp giao diện RESTful để truy cập dữ liệu từ máy chủ back-end. Được sử dụng để tạo API RESTful để ứng dụng tương tác với dữ liệu được lưu trữ trên máy chủ.
- **Firebase:** Nền tảng lưu trữ đám mây cung cấp nhiều dịch vụ bao gồm lưu trữ hình ảnh, cơ sở dữ liệu và xác thực. Được sử dụng để lưu trữ hình ảnh được tải lên ứng dụng.
- **SQL Server:** Hệ quản trị cơ sở dữ liệu quan hệ mạnh mẽ và phổ biến, được sử dụng để lưu trữ và truy xuất dữ liệu có cấu trúc. Được sử dụng để lưu trữ dữ liệu cho ứng dụng.
- **GitHub:** Nền tảng lưu trữ kho mã nguồn phổ biến cho phép lưu trữ, quản lý và theo dõi thay đổi đối với codebase của ứng dụng.

5.7. Giao diện người dùng



5.7.1. Màn hình Đăng Nhập



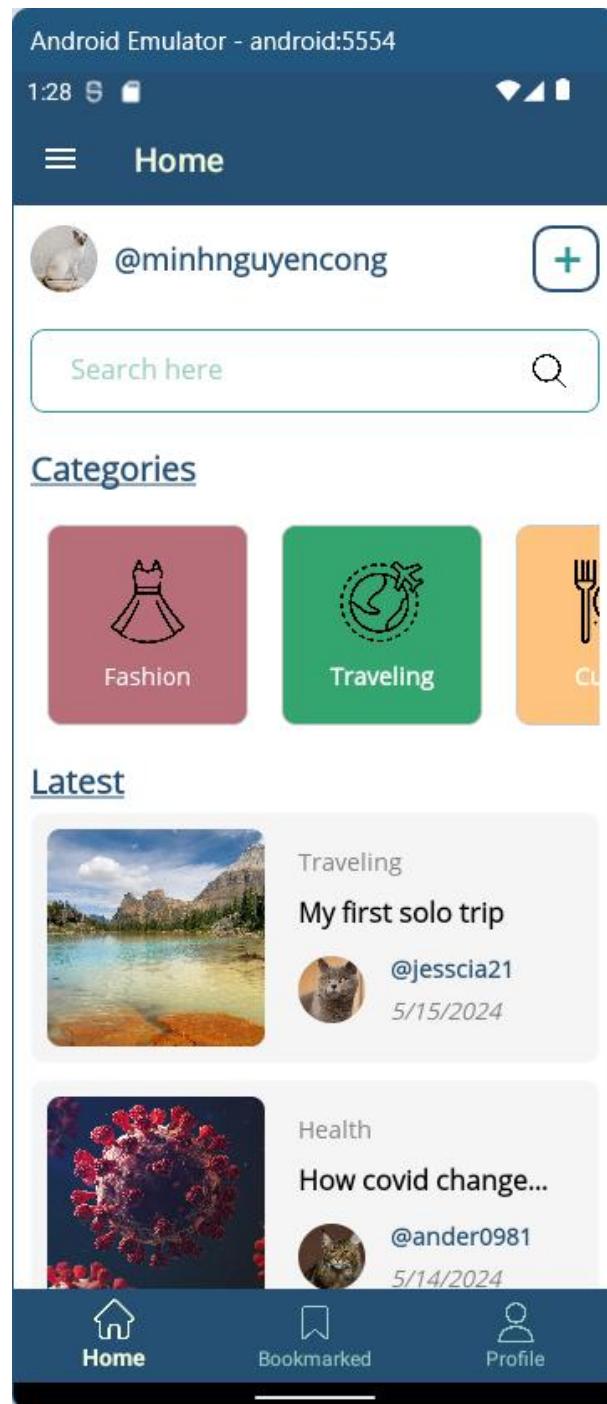
Hình 5.1: Màn hình đăng nhập

5.7.2. Màn hình Đăng Ký



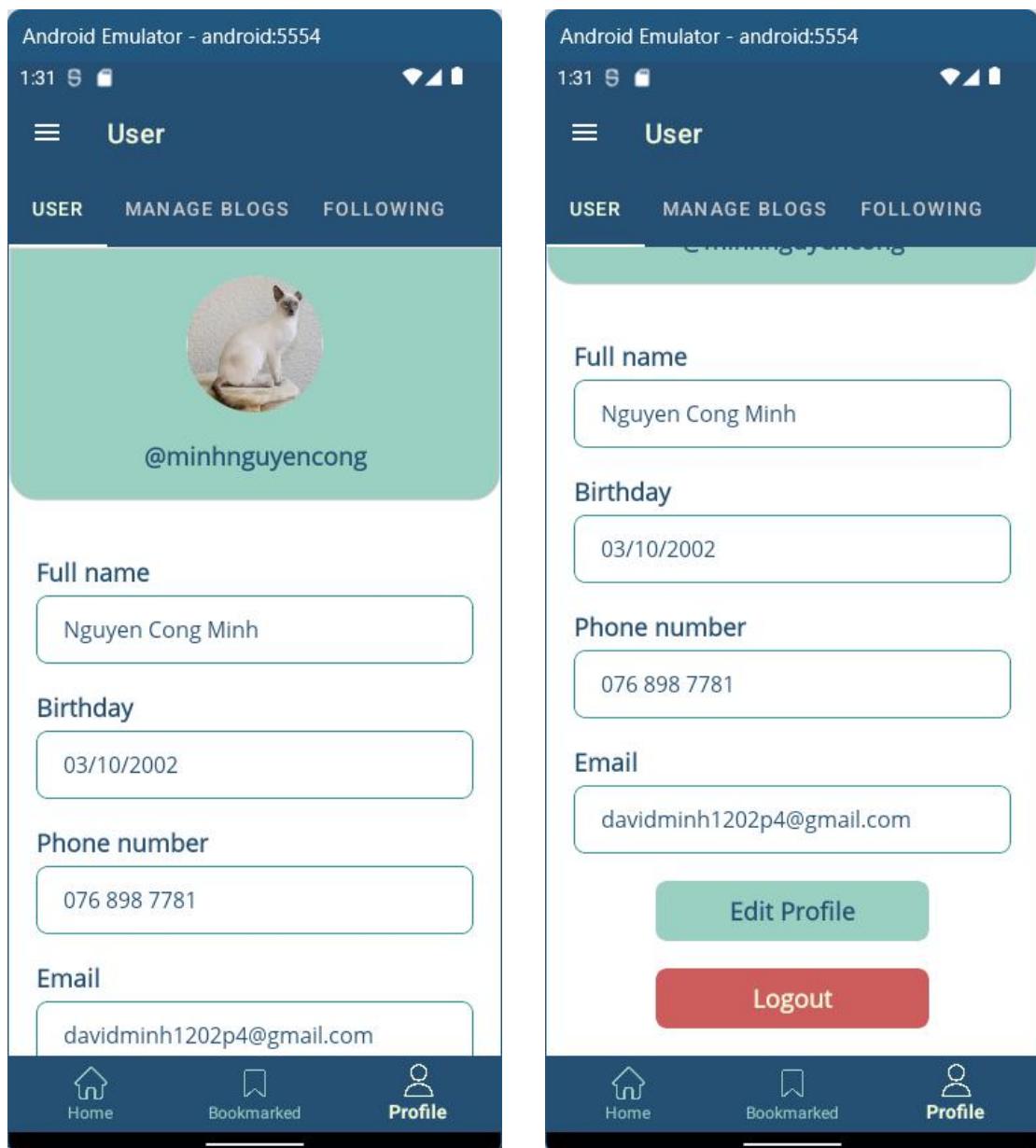
Hình 5.2: Màn hình đăng ký

5.7.3. Màn hình Trang Chủ



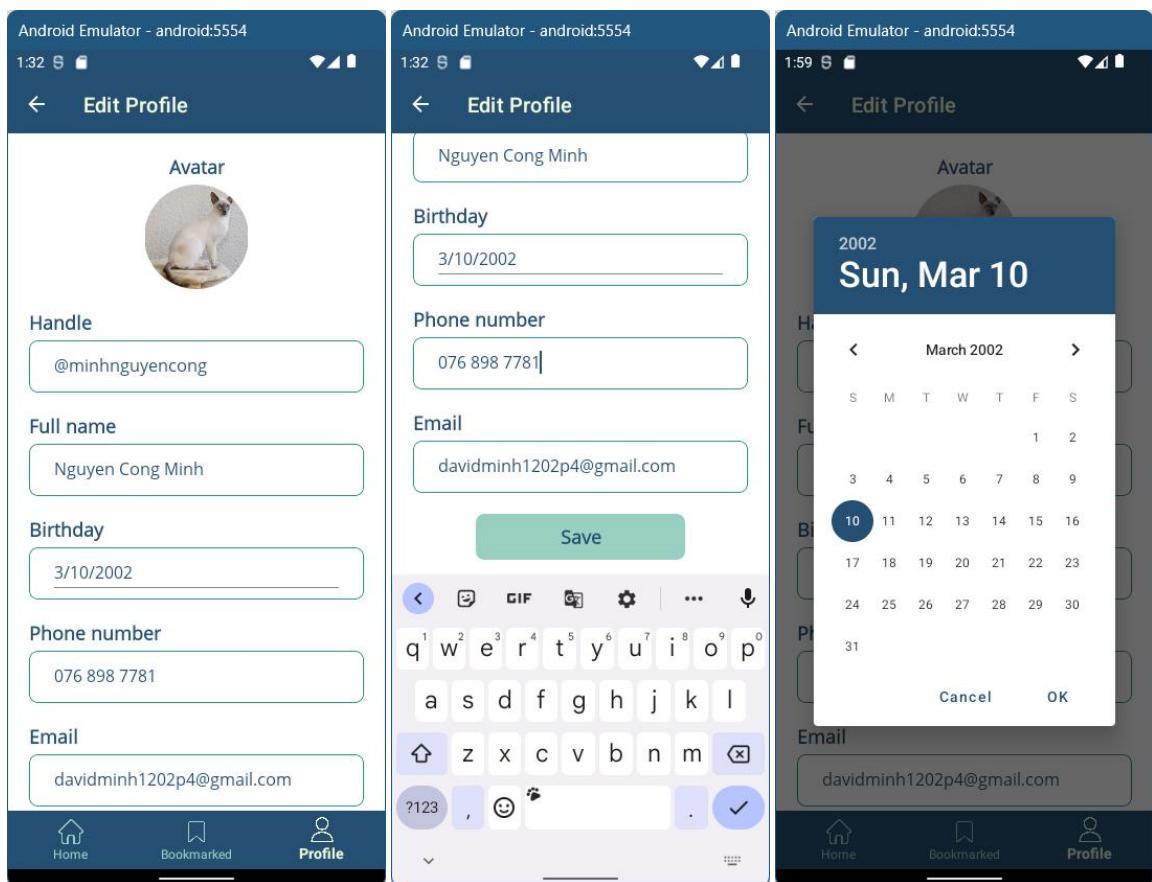
Hình 5.3: Màn hình trang chủ

5.7.4. Màn hình Thông Tin Người Dùng



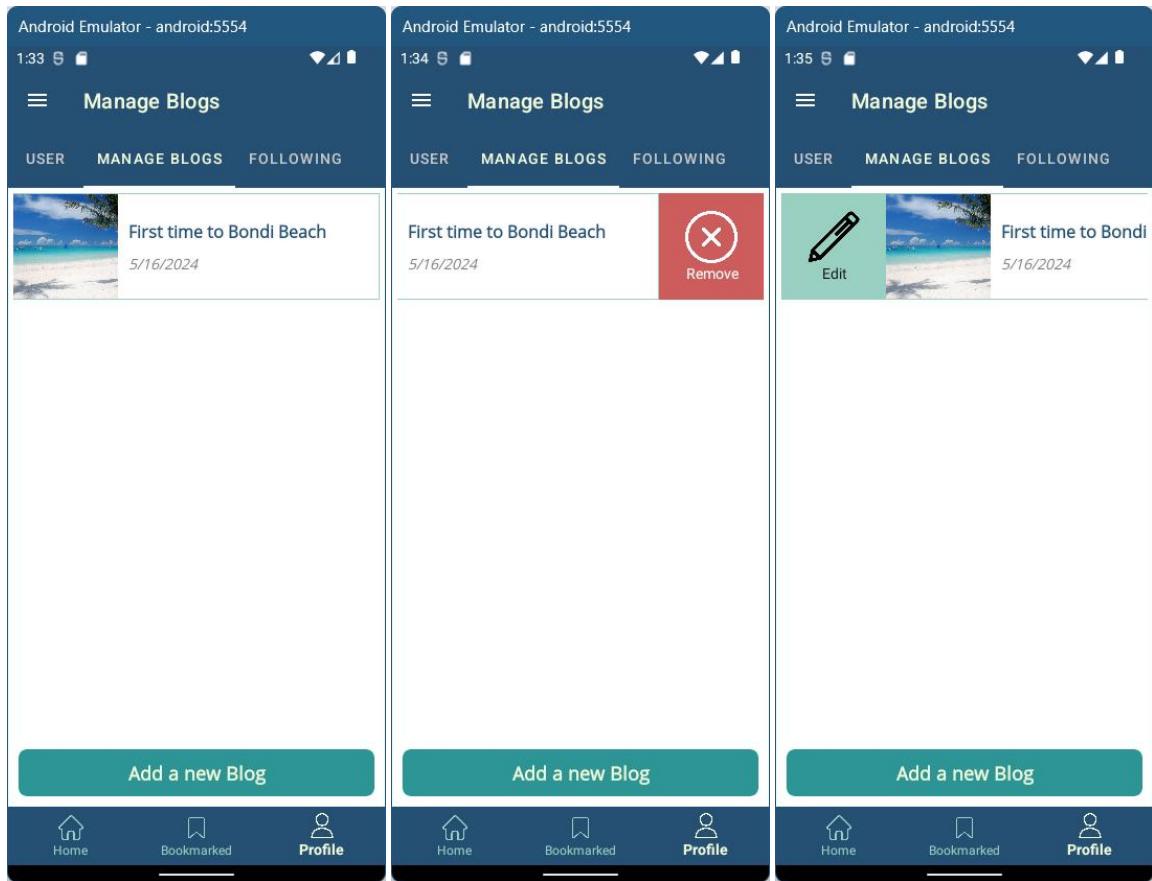
Hình 5.4: Màn hình thông tin người dùng

5.7.5. Màn hình Chỉnh Sửa Thông Tin Người Dùng



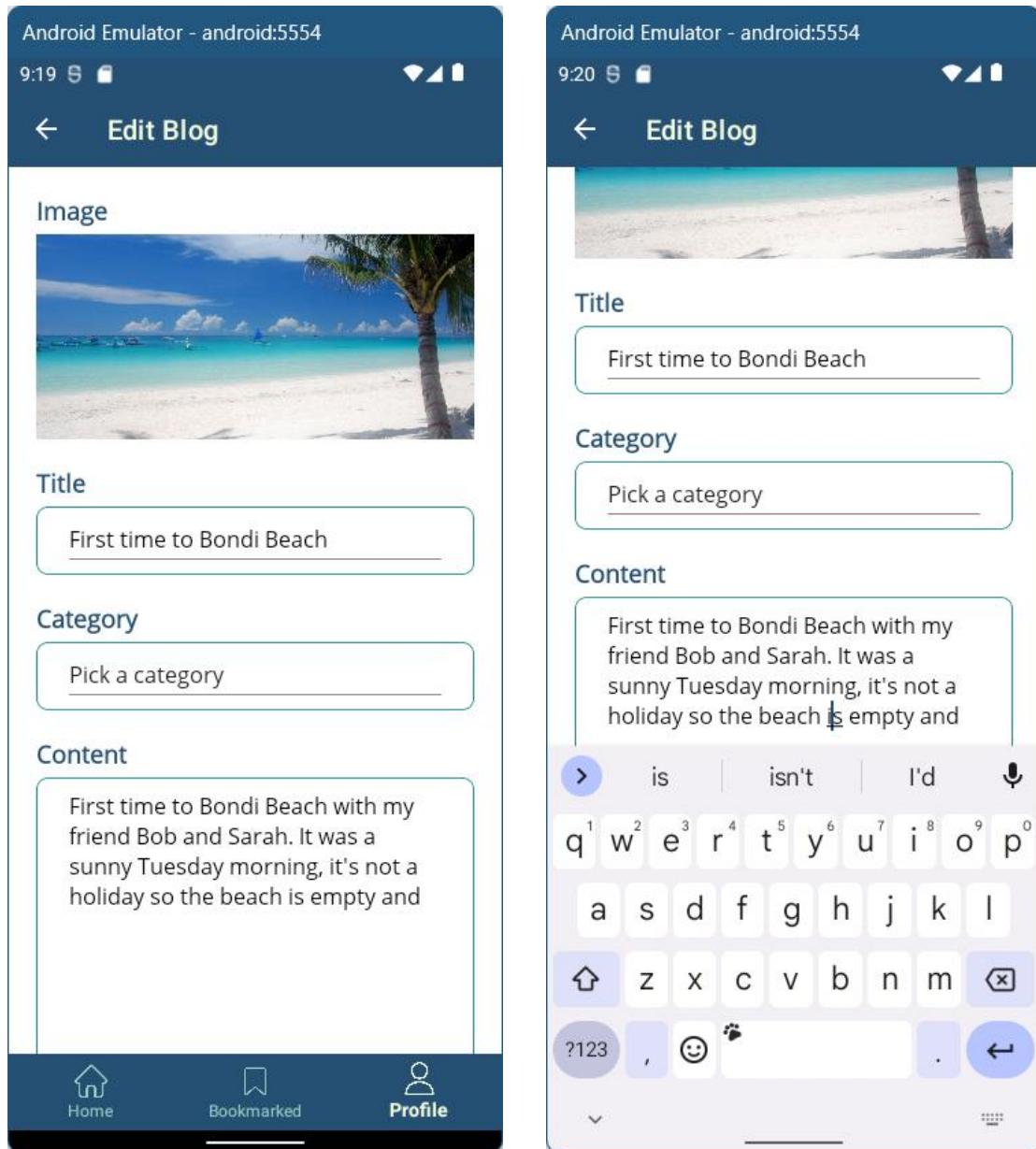
Hình 5.5: Màn hình chỉnh sửa thông tin người dùng

5.7.6. Màn hình Quản Lý Blog



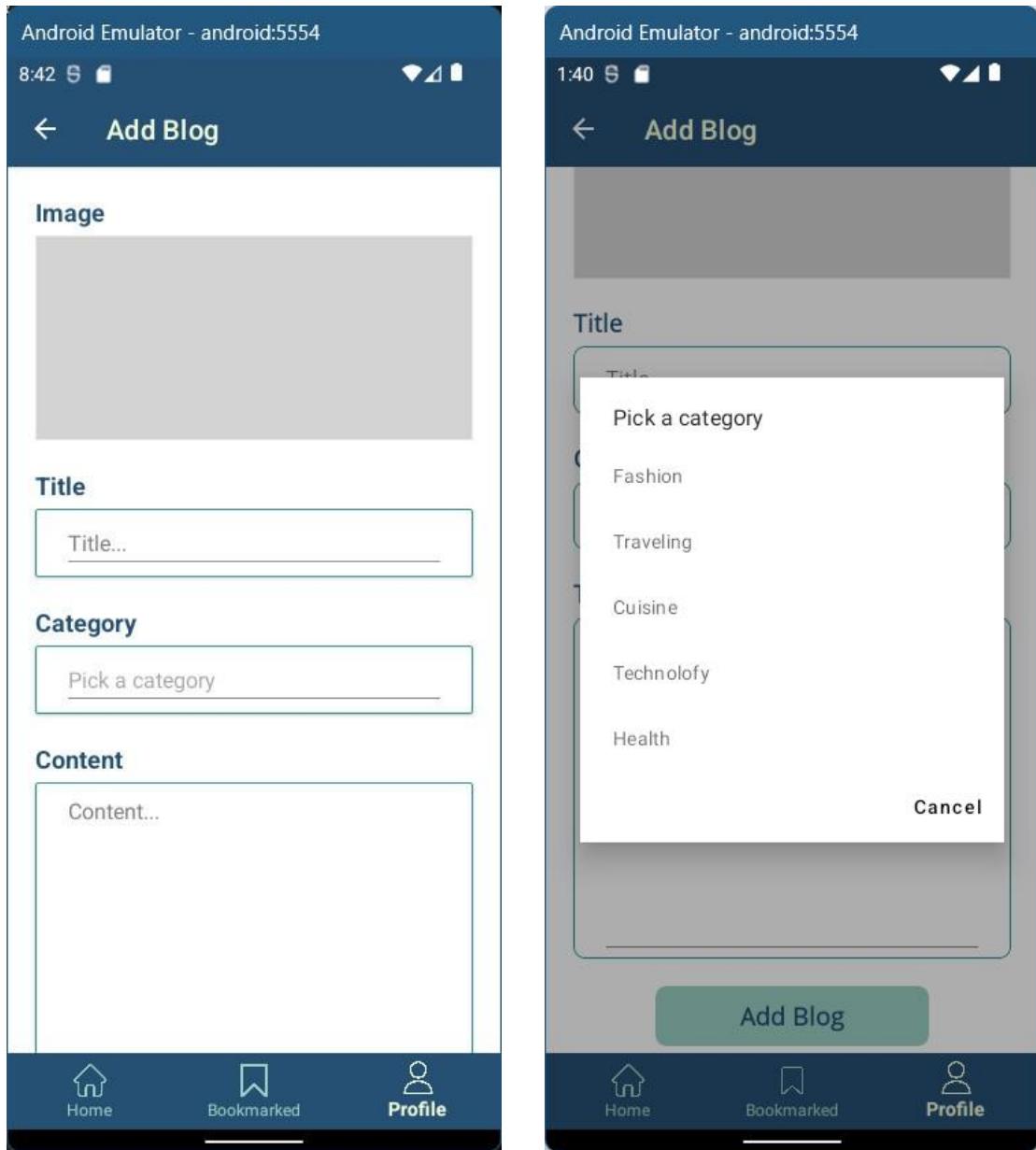
Hình 5.6: Màn hình quản lý blog

5.7.7. Màn hình Chính Sửa Blog



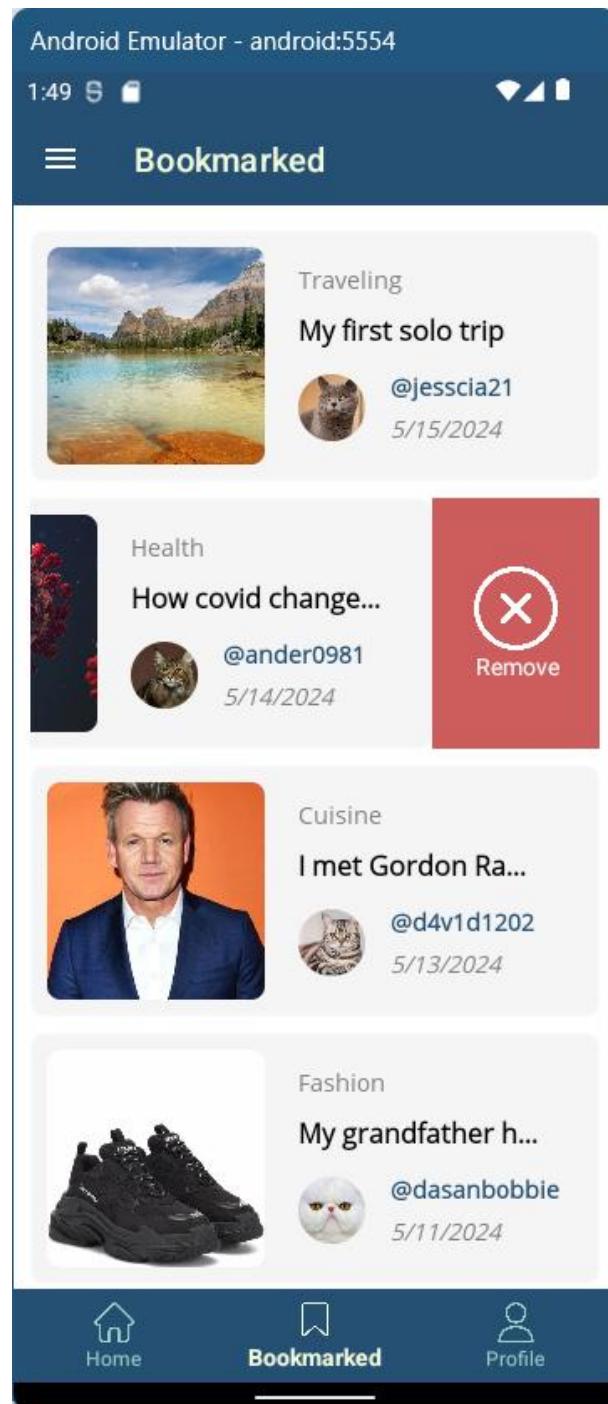
Hình 5.7: Màn hình chỉnh sửa blog

5.7.8. Màn hình Thêm Blog



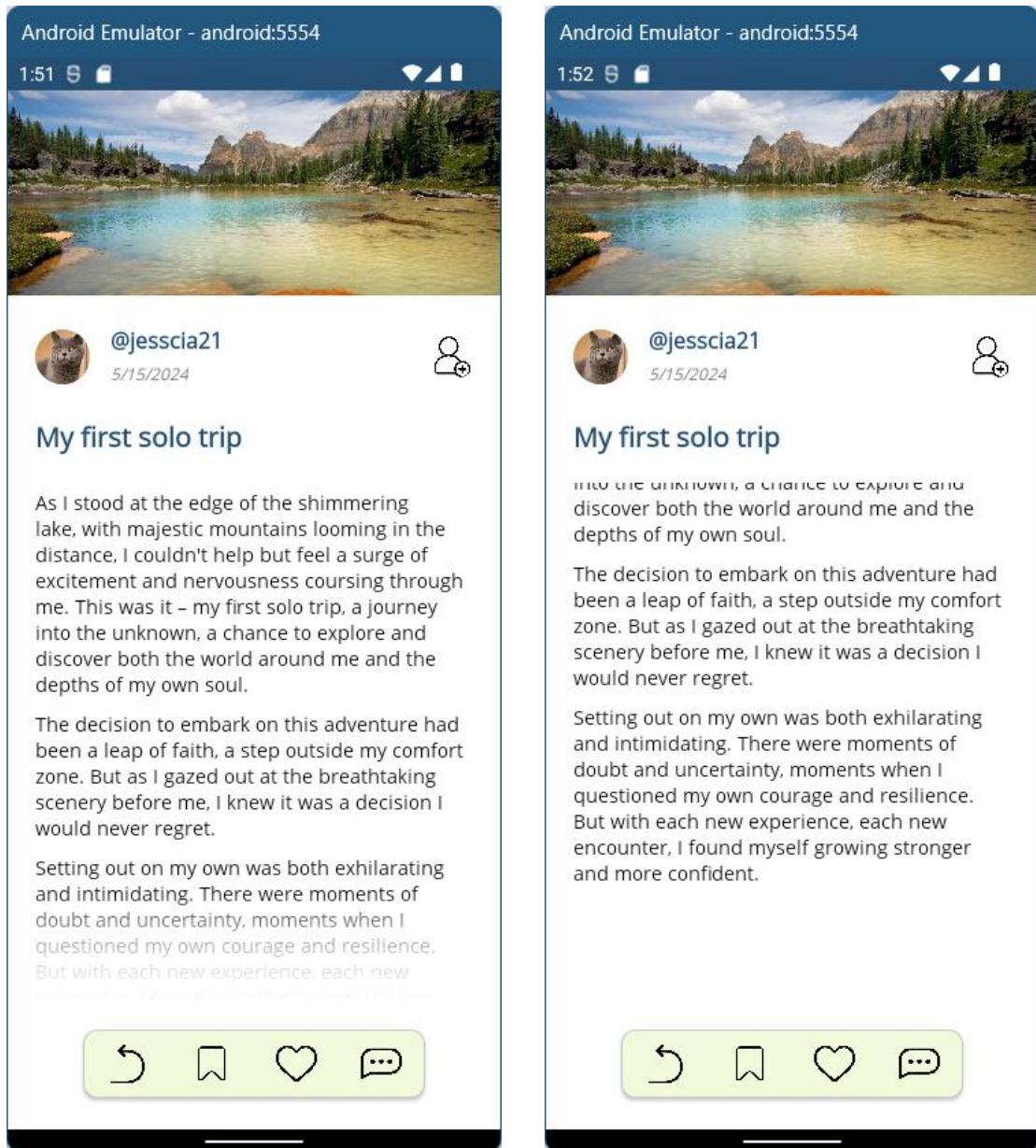
Hình 5.8: Màn hình thêm blog

5.7.9. Màn hình Blog Đã Lưu



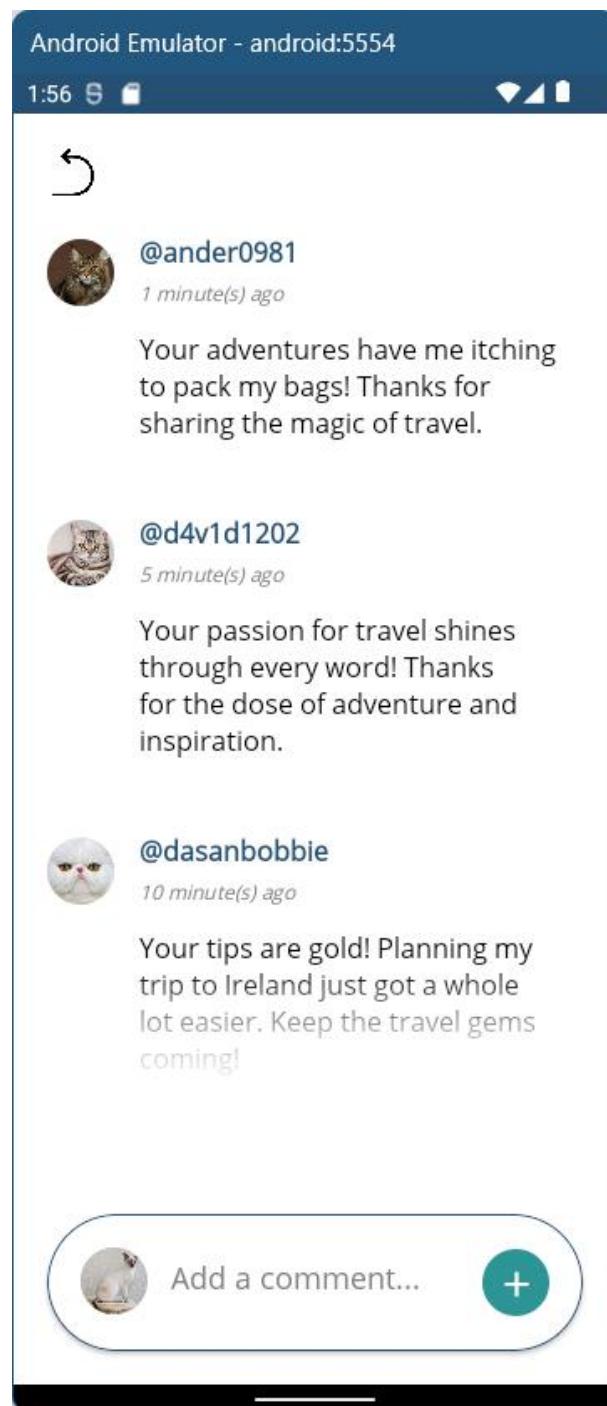
Hình 5.9: Màn hình blog đã lưu

5.7.10. Màn hình Chi Tiết Blog



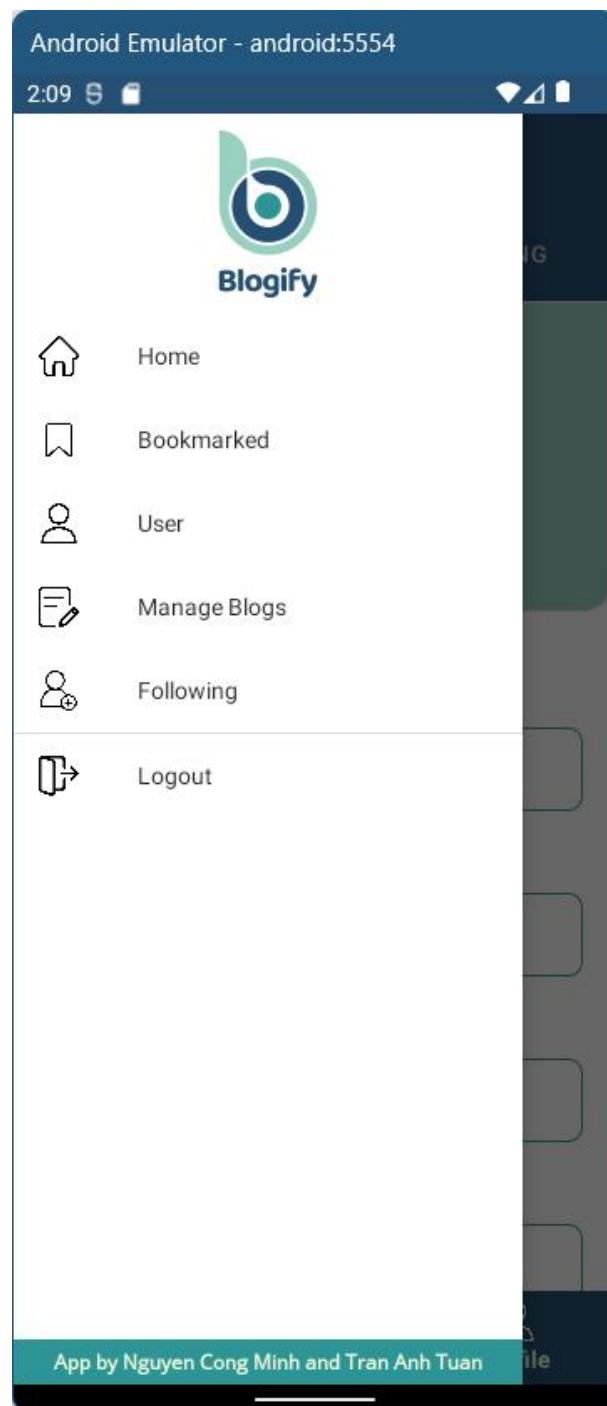
Hình 5.10: Màn hình chi tiết blog

5.7.11. Màn hình Bình Luận



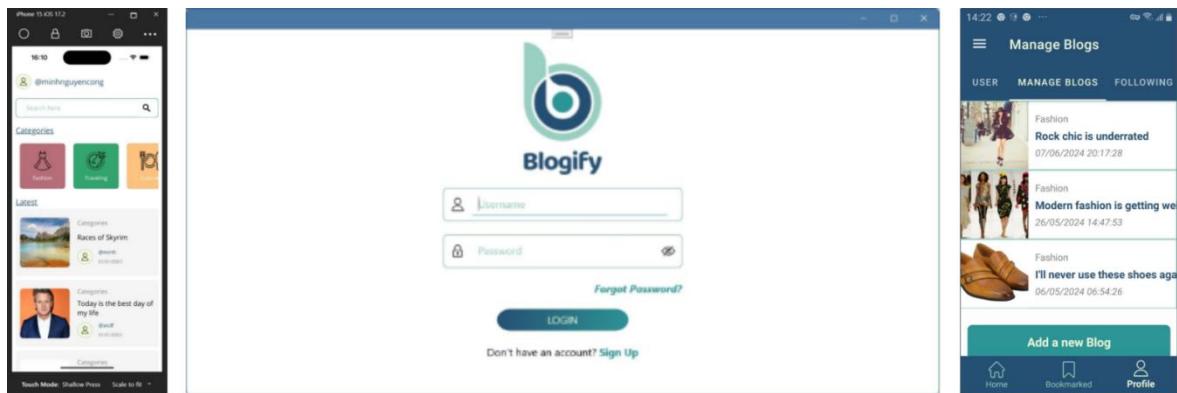
Hình 5.11: Màn hình bình luận

5.7.12. Màn hình Flyout



Hình 5.12: Màn hình flyout hỗ trợ điều hướng giữa các màn hình trong ứng dụng

5.7.13. Giao diện được triển khai trên các nền tảng khác nhau



Hình 5.13: Ứng dụng được triển khai trên các nền tảng khác nhau

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. Kết quả đạt được

- Tìm hiểu được về Framework .NET MAUI và xây dựng ứng dụng minh họa. Ứng dụng minh họa Blog tuy không phải là ứng dụng mang tầm cỡ cao lớn nhưng đủ nổi bật để nói lên đặc điểm của các thành phần cũng như cách thức hoạt động của Framework .NET MAUI qua đó cho thấy được ưu điểm và nhược điểm của công cụ này trong việc xây dựng và phát triển phần mềm.
- Nắm được quy trình, cách phát triển ứng dụng nói riêng và ứng dụng đa nền tảng nói chung.
- Tìm hiểu, sử dụng và kết hợp các công nghệ khác nhau vào ứng dụng .NET MAUI. Tuy không phải là các công nghệ mới nhưng chúng chính là những công nghệ hiện đại phù hợp cho xu thế chung của lĩnh vực phần mềm công nghệ thông tin.
- Học được cách làm việc nhóm cũng như cách phân chia công việc trong dự án nghiên cứu và dự án phát triển phần mềm.

6.2. Hạn chế

- Chưa áp dụng nhiều các tính năng nâng cao vào ứng dụng
- Giao diện còn nhiều thiếu sót.
- Tốc độ ứng dụng còn chậm.
- Code chưa được tối ưu hoàn toàn.
- Hoạt động được trên nhiều nền tảng khác nhau tuy nhiên chưa được tối ưu cho từng nền tảng.

6.3. Hướng phát triển

Từ những hạn chế trên, nhóm nhận thấy có những điểm có thể cải thiện:

- Hoàn thiện các chức năng còn thiếu sót.
- Áp dụng các tính năng nâng cao vào ứng dụng như Handler, Animation, Theme,... để cải thiện giao diện người dùng.
- Sử dụng Web API kết hợp với mô hình MVVM.
- Sử dụng Folder Platforms để thực hiện code riêng biệt cho từng nền tảng, từ đó giúp cải thiện ứng dụng khi chạy trên các nền tảng khác nhau.
- Tối ưu hóa code để tăng tốc độ ứng dụng.
- Tìm giải pháp khác với những lỗi thuộc về Framework đặc biệt là về giao diện người dùng.

Tài liệu tham khảo

1. Matt Goldman, “.NET Maui in Action (MEAP V08)”, Manning Publications (2023), 117-333.
2. Shaun Lawrence, “*Introducing .NET MAUI - Build and Deploy Cross-platform Applications Using C# and .NET Multi-platform App UI*”, Apress (2023), 45-55.
3. Pieter Nijs, “The MVVM Pattern in .NET MAUI - The definitive guide to essential patterns, best practices, and techniques for cross-platform app development (1st Edition)”, Packt Publishing (2023).
4. Mark J. Price, “Apps and Services with .NET 8 - Build practical projects with Blazor, .NET MAUI, gRPC, GraphQL, and other enterprise technologies (2nd Edition)”, Packt Publishing (2023).
5. Michael Stonis, “Enterprise Application Patterns Using .NET MAUI v1.0”, Microsoft Developer Division, .NET, and Visual Studio product teams (2022).
6. Roger Ye, “*.NET MAUI Cross-Platform Application Development - Leverage a first-class cross-platform UI framework to build native apps on multiple platforms (1st Edition)*”, Packt Publishing (2023), 88-98.
7. Tài liệu .NET MAUI, <https://learn.microsoft.com/en-us/dotnet/maui/?view=net-maui-8.0>, khai thác ngày 01/05/2024.
8. Tài liệu về kiến trúc .NET, <https://learn.microsoft.com/en-us/dotnet/architecture/>, khai thác ngày 13/4/2024.
9. Tài liệu về MVVM, <https://www.geeksforgeeks.org/introduction-to-model-view-view-model-mvvm/>, khai thác ngày 04/05/2024.
10. Blog về .NET MAUI và Xamarin, <https://www.xamboy.com/>, khai thác ngày 11/4/2024.
11. Blog về .NET MAUI, <https://devblogs.microsoft.com/dotnet/category/maui/>, khai thác ngày 11/03/2024.