**Отчет по Главе 8: Навигация (Navigation)**

**1. Введение**

Данная глава посвящена изучению механизмов навигации в .NET MAUI. Навигация является неотъемлемой частью любого многостраничного приложения, позволяя пользователям перемещаться между различными представлениями и функциями. .NET MAUI предлагает несколько подходов к управлению навигацией, включая стековую навигацию, модальные страницы и мощную систему навигации на основе URI, предоставляемую Shell. Понимание этих механизмов и способов передачи данных между страницами критически важно для построения логичной и удобной пользовательской структуры настольных и мобильных приложений.

**2. Основная часть**

**2.1. Основные концепции и принципы**

* **Стековая навигация (Stack Navigation):**
  + Страницы помещаются в стек по принципу "последним пришел - первым вышел" (LIFO).
  + Navigation.PushAsync(new MyPage()): Добавляет новую страницу в стек и отображает ее.
  + Navigation.PopAsync(): Удаляет текущую страницу из стека и возвращает пользователя на предыдущую.
  + Доступна через свойство Navigation у Page.
* **Модальная навигация (Modal Navigation):**
  + Страница отображается поверх текущего стека навигации, блокируя взаимодействие с нижележащими страницами до ее закрытия.
  + Navigation.PushModalAsync(new MyModalPage()): Отображает страницу модально.
  + Navigation.PopModalAsync(): Закрывает текущую модальную страницу.
* **Shell Навигация:**
  + Мощная система навигации, основанная на URI (строковых маршрутах). Предоставляет возможности для создания сложной иерархии UI (всплывающие меню, вкладки).
  + Shell.Current.GoToAsync("route"): Переход по указанному маршруту. Маршруты могут быть абсолютными (начинаются с //) или относительными.
  + **Регистрация маршрутов:** Страницы, на которые осуществляется переход по URI и которые не определены явно в AppShell.xaml как ShellContent, должны быть зарегистрированы с помощью Routing.RegisterRoute("routeName", typeof(PageType)) в конструкторе AppShell или в другом подходящем месте.
  + **Передача данных через параметры запроса URI:** Данные можно передавать как часть URI: "routeName?param1=value1&param2=value2".
  + **Получение данных на целевой странице:** С помощью атрибута [QueryProperty(nameof(PropertyNameInPage), "uriQueryParamName")] свойство страницы связывается с параметром из URI.
* **Передача данных между страницами:**
  + **Через конструктор:** При стековой навигации данные можно передать в конструктор новой страницы.
  + **Через параметры запроса URI (Shell):** Как описано выше.
  + **MessagingCenter**: Система обмена сообщениями по подписке/публикации, позволяющая слабо связанным компонентам обмениваться данными. Подходит для передачи данных "назад" или между несвязанными напрямую страницами.
  + **Статические свойства или сервисы (Dependency Injection):** Для обмена более сложными или глобальными данными.
  + **Прямой доступ к свойствам предыдущей страницы (не рекомендуется для сильной связности):** При PopAsync можно получить доступ к экземпляру предыдущей страницы из стека навигации.

**2.2. Примеры кода с пояснениями**

Для демонстрации навигации были созданы страницы NavigationHomePage.xaml, NavigationDetailPage.xaml и NavigationModalPage.xaml.

**2.2.1. NavigationHomePage.xaml и NavigationHomePage.xaml.cs**  
Стартовая страница с кнопками для различных типов навигации.

<!-- Фрагмент NavigationHomePage.xaml -->

<Button Text="Go to Detail Page (PushAsync)" Clicked="GoToDetailPage\_Clicked"/>

<Button Text="Go to Detail Page (Shell URI)" Clicked="GoToDetailPageShell\_Clicked"/>

<Button Text="Open Modal Page" Clicked="OpenModalPage\_Clicked"/>

<Label x:Name="ResultLabel" Text="Result from Detail Page will appear here."/>

// Фрагмент NavigationHomePage.xaml.cs

private async void GoToDetailPage\_Clicked(object sender, EventArgs e)

{

await Navigation.PushAsync(new NavigationDetailPage("Hello from Home Page!"));

}

private async void GoToDetailPageShell\_Clicked(object sender, EventArgs e)

{

await Shell.Current.GoToAsync($"{nameof(NavigationDetailPage)}?message={Uri.EscapeDataString("Navigated via Shell URI!")}");

}

private async void OpenModalPage\_Clicked(object sender, EventArgs e)

{

await Navigation.PushModalAsync(new NavigationModalPage());

}

// ... обработка MessagingCenter и UpdateResult ...

*Пояснение:* Демонстрируется PushAsync с передачей данных через конструктор, GoToAsync с параметром URI и PushModalAsync.

**2.2.2. NavigationDetailPage.xaml и NavigationDetailPage.xaml.cs**  
Страница для приема данных и возврата.

<!-- Фрагмент NavigationDetailPage.xaml -->

<Label x:Name="ReceivedDataLabel" Text="No data received."/>

<Entry x:Name="DataToReturnEntry" Placeholder="Enter data to return" />

<Button Text="Go Back (PopAsync)" Clicked="GoBack\_Clicked"/>

<Button Text="Go Back and Send Data (via MessagingCenter)" Clicked="GoBackAndSendData\_Clicked"/>

// Фрагмент NavigationDetailPage.xaml.cs

[QueryProperty(nameof(MessageFromShell), "message")]

public partial class NavigationDetailPage : ContentPage

{

// ... свойство MessageFromShell и конструкторы ...

private async void GoBack\_Clicked(object sender, EventArgs e)

{

// ... логика передачи данных напрямую (менее предпочтительно) ...

await Navigation.PopAsync();

}

private async void GoBackAndSendData\_Clicked(object sender, EventArgs e)

{

MessagingCenter.Send(this, "DataFromDetail", DataToReturnEntry.Text ?? "Data");

await Navigation.PopAsync();

}

}

*Пояснение:* Страница принимает данные через конструктор или QueryProperty. Возврат данных демонстрируется через прямой доступ (менее предпочтительно) и MessagingCenter.

**2.2.3. NavigationModalPage.xaml и NavigationModalPage.xaml.cs**  
Простая модальная страница.

// Фрагмент NavigationModalPage.xaml.cs

private async void CloseModal\_Clicked(object sender, EventArgs e)

{

await Navigation.PopModalAsync();

}

**2.2.4. Регистрация маршрута в AppShell.xaml.cs**

// Фрагмент AppShell.xaml.cs

public AppShell()

{

InitializeComponent();

Routing.RegisterRoute(nameof(NavigationDetailPage), typeof(NavigationDetailPage));

}

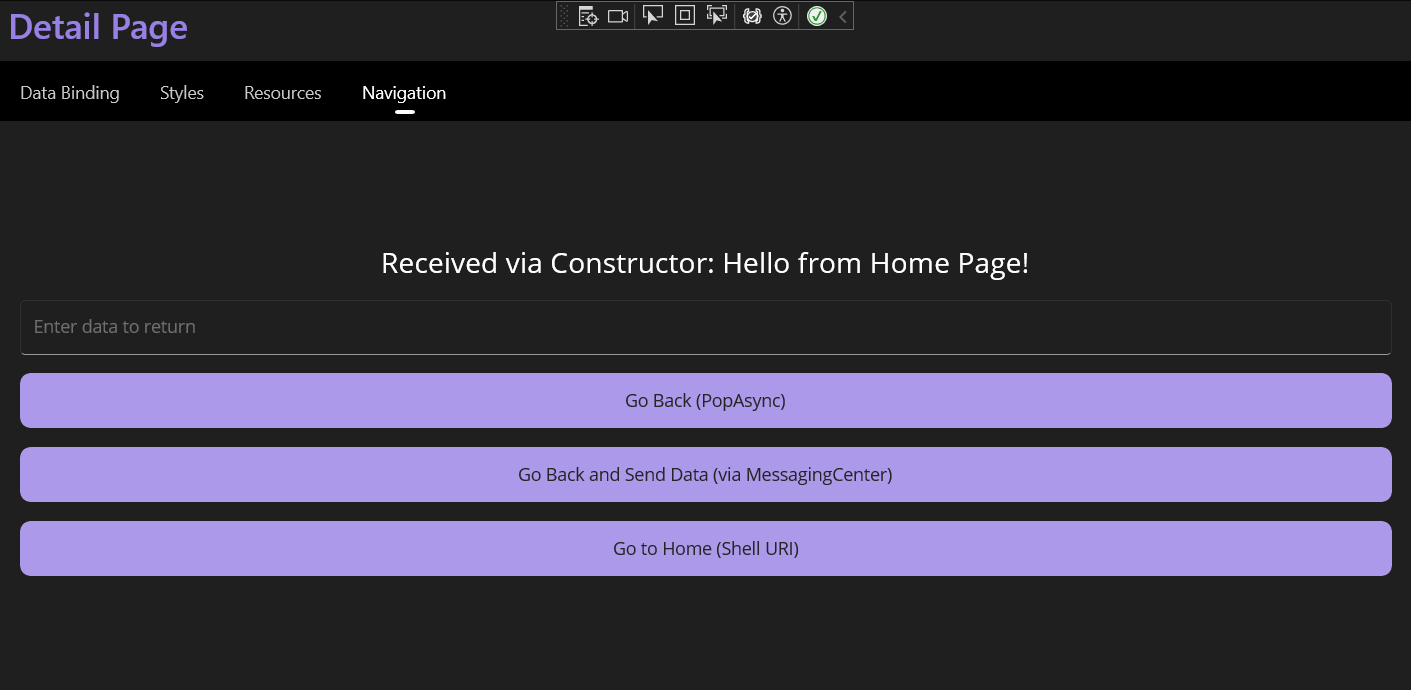
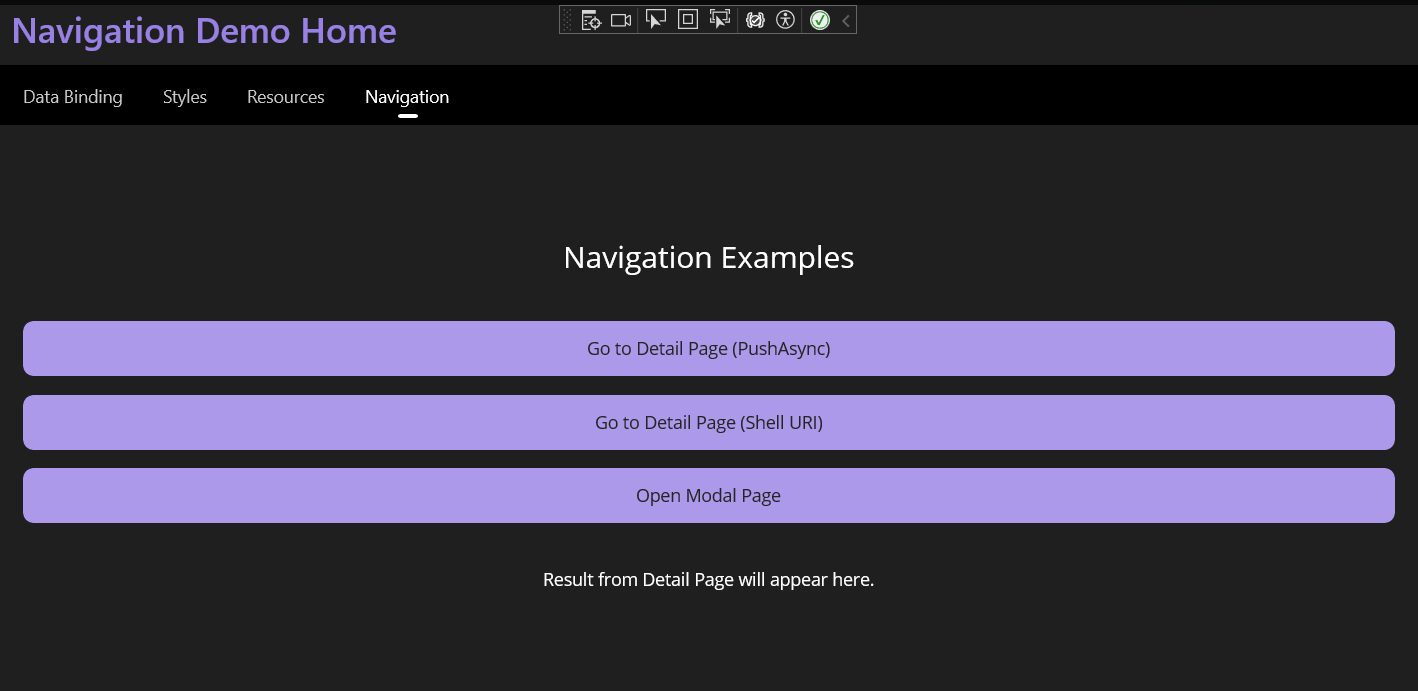
*Пояснение:* Необходимо для работы навигации Shell.Current.GoToAsync к NavigationDetailPage.

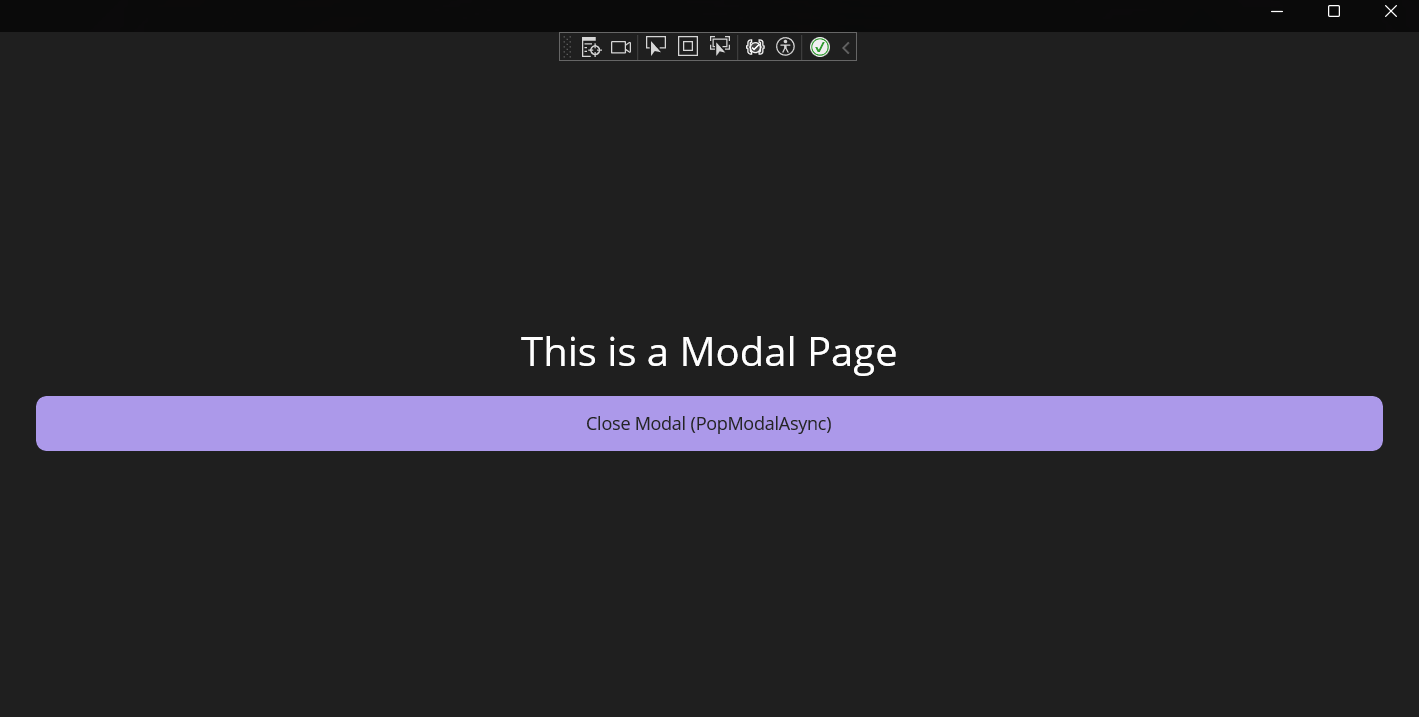
**3. Заключение**

Навигация является ключевым аспектом разработки интерактивных приложений. .NET MAUI предоставляет гибкие механизмы для ее реализации, включая традиционную стековую и модальную навигацию, а также мощную систему маршрутизации Shell. Выбор конкретного подхода зависит от архитектуры приложения. Умение передавать данные между страницами различными способами (конструктор, параметры URI, MessagingCenter) позволяет создавать сложные потоки взаимодействия с пользователем. Эти знания важны для построения логичной структуры настольных приложений.

**4. Скриншоты и примеры**

Демонстрация различных типов навигации была выполнена с использованием трех страниц и запущена на платформе Windows.



*Рис. 8.1 - 8.3. Демонстрация стековой, Shell URI и модальной навигации, а также передачи данных.*

**Процесс выполнения примеров:**

1. Созданы страницы NavigationHomePage, NavigationDetailPage и NavigationModalPage.
2. Реализованы переходы между страницами с использованием Navigation.PushAsync, Shell.Current.GoToAsync и Navigation.PushModalAsync.
3. Продемонстрированы способы передачи данных: через конструктор, параметры URI (QueryProperty) и MessagingCenter.
4. Маршрут для NavigationDetailPage зарегистрирован в AppShell.xaml.cs.
5. NavigationHomePage добавлена в меню AppShell.xaml.
6. Проведено тестирование всех сценариев навигации и передачи данных.  
   Все типы навигации и способы передачи данных работали корректно, позволяя перемещаться между страницами и обмениваться информацией.

**5. Документация**

* **Ссылка на GitHub с исходным кодом:** <https://github.com/Nikita-Levuskin/C-sharp-practic/tree/main/1>
* **Описание выполненных шагов:**
  1. Изучены материалы по теме "Навигация" (глава 8) с ресурса [metanit.com/sharp/maui/](https://www.google.com/url?sa=E&q=https%3A%2F%2Fmetanit.com%2Fsharp%2Fmaui%2F).
  2. Созданы три страницы для демонстрации навигационных сценариев.
  3. Реализованы примеры стековой, модальной и Shell URI-навигации.
  4. Продемонстрированы различные методы передачи данных между страницами.
  5. Выполнена регистрация маршрута для Shell-навигации.
  6. Проведено тестирование на платформе Windows и подготовлены скриншоты.