**Отчет по Главе 10: Диалоги (Dialogs)**

**1. Введение**

Данная глава посвящена изучению встроенных механизмов для отображения диалоговых окон в .NET MAUI. Диалоговые окна являются неотъемлемой частью пользовательского взаимодействия, позволяя информировать пользователя о важных событиях, запрашивать подтверждение критических действий, предлагать выбор из нескольких опций или получать простой текстовый ввод. Умение использовать стандартные диалоговые окна важно для создания интуитивно понятных и соответствующих гайдлайнам платформы настольных и мобильных приложений.

**2. Основная часть**

**2.1. Основные концепции и принципы**

.NET MAUI предоставляет три основных типа асинхронных диалоговых окон, которые вызываются из любого объекта Page. Эти методы отображают нативные для каждой платформы диалоги.

* **DisplayAlert**:
  + Отображает простое модальное окно с уведомлением.
  + **Перегрузки:**
    - DisplayAlert(title, message, cancel): Показывает окно с заголовком, сообщением и одной кнопкой для закрытия.
    - DisplayAlert(title, message, accept, cancel): Показывает окно с двумя кнопками (например, "Да" и "Нет"). Метод возвращает Task<bool>, где true соответствует нажатию кнопки accept.
* **DisplayActionSheet**:
  + Предоставляет пользователю список вариантов действий. Внешний вид зависит от платформы (например, на мобильных устройствах это может быть "лист", выдвигающийся снизу).
  + DisplayActionSheet(title, cancel, destruction, ...buttons):
    - title: Заголовок окна.
    - cancel: Текст кнопки отмены (нажатие на нее или закрытие окна вернет null или этот текст).
    - destruction: Текст кнопки для "разрушительного" действия (например, "Удалить"). На некоторых платформах эта кнопка может выделяться красным цветом.
    - ...buttons: Перечисление остальных вариантов действий в виде строк.
  + Метод возвращает Task<string>, содержащий текст нажатой кнопки.
* **DisplayPromptAsync**:
  + Отображает диалоговое окно, запрашивающее у пользователя ввод текстовой строки.
  + DisplayPromptAsync(title, message, accept, cancel, placeholder, maxLength, keyboard): Позволяет настроить заголовок, сообщение, текст кнопок, текст-заполнитель, максимальную длину ввода и тип клавиатуры.
  + Метод возвращает Task<string>, содержащий введенный пользователем текст, или null, если пользователь нажал "Отмена".

**2.2. Примеры кода с пояснениями**

Демонстрация работы с диалогами была реализована на странице DialogsDemoPage.xaml, в code-behind которой (DialogsDemoPage.xaml.cs) вызывались соответствующие методы.

// Фрагменты из DialogsDemoPage.xaml.cs

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

{

await DisplayAlert("Simple Alert", "This is a simple alert message.", "OK");

ResultLabel.Text = "Simple Alert was shown.";

}

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

{

bool answer = await DisplayAlert("Confirmation", "Do you want to proceed?", "Yes", "No");

ResultLabel.Text = $"Confirmation result: {answer}";

}

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

{

string action = await DisplayActionSheet("Choose an option", "Cancel", "Delete", "Option 1", "Option 2", "Option 3");

ResultLabel.Text = $"Action Sheet result: {action}";

}

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

{

string result = await DisplayPromptAsync("Question", "What is your name?", "OK", "Cancel", "Your name...", 20, Keyboard.Text);

ResultLabel.Text = $"Prompt result: {result}";

}

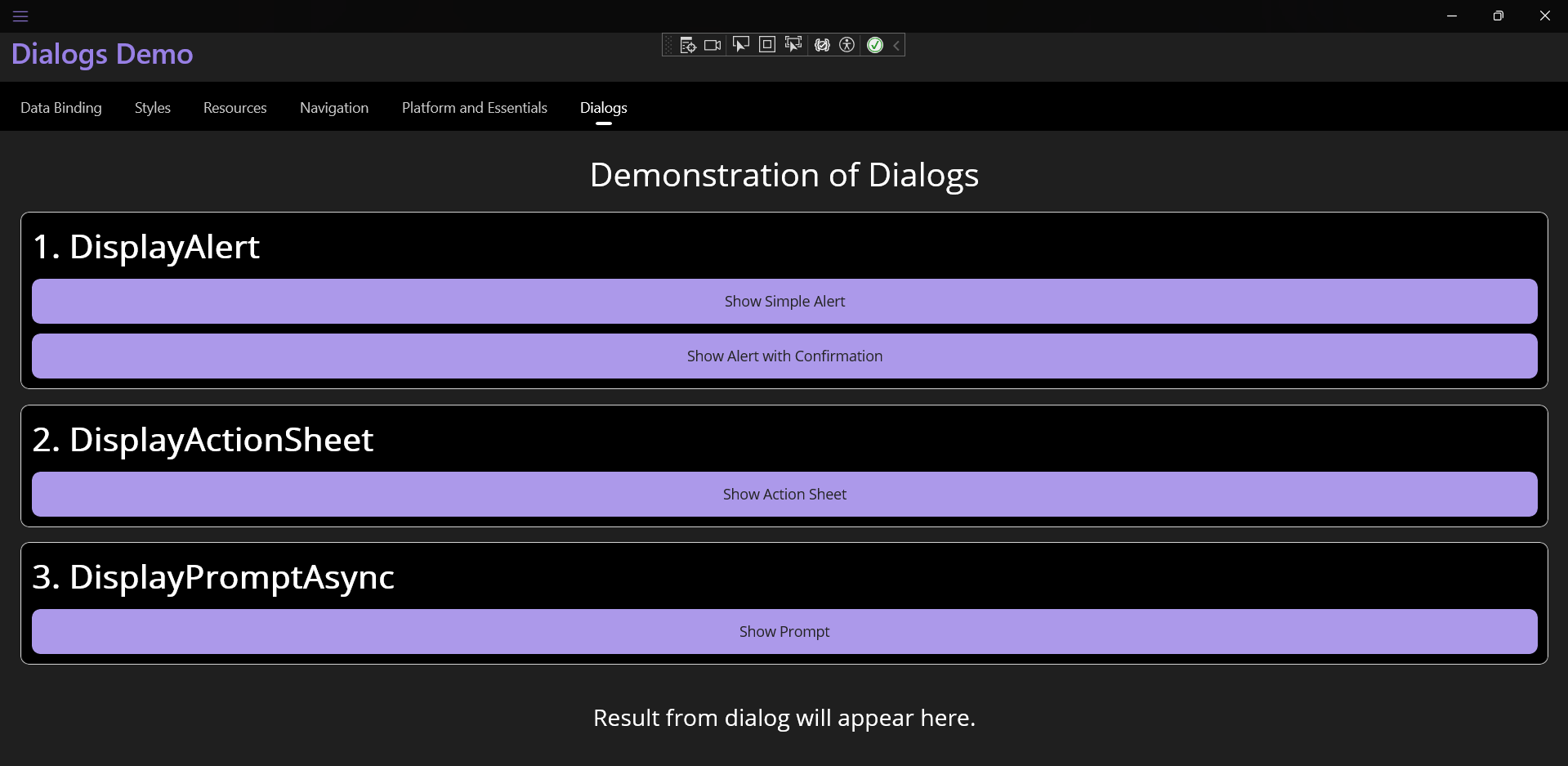
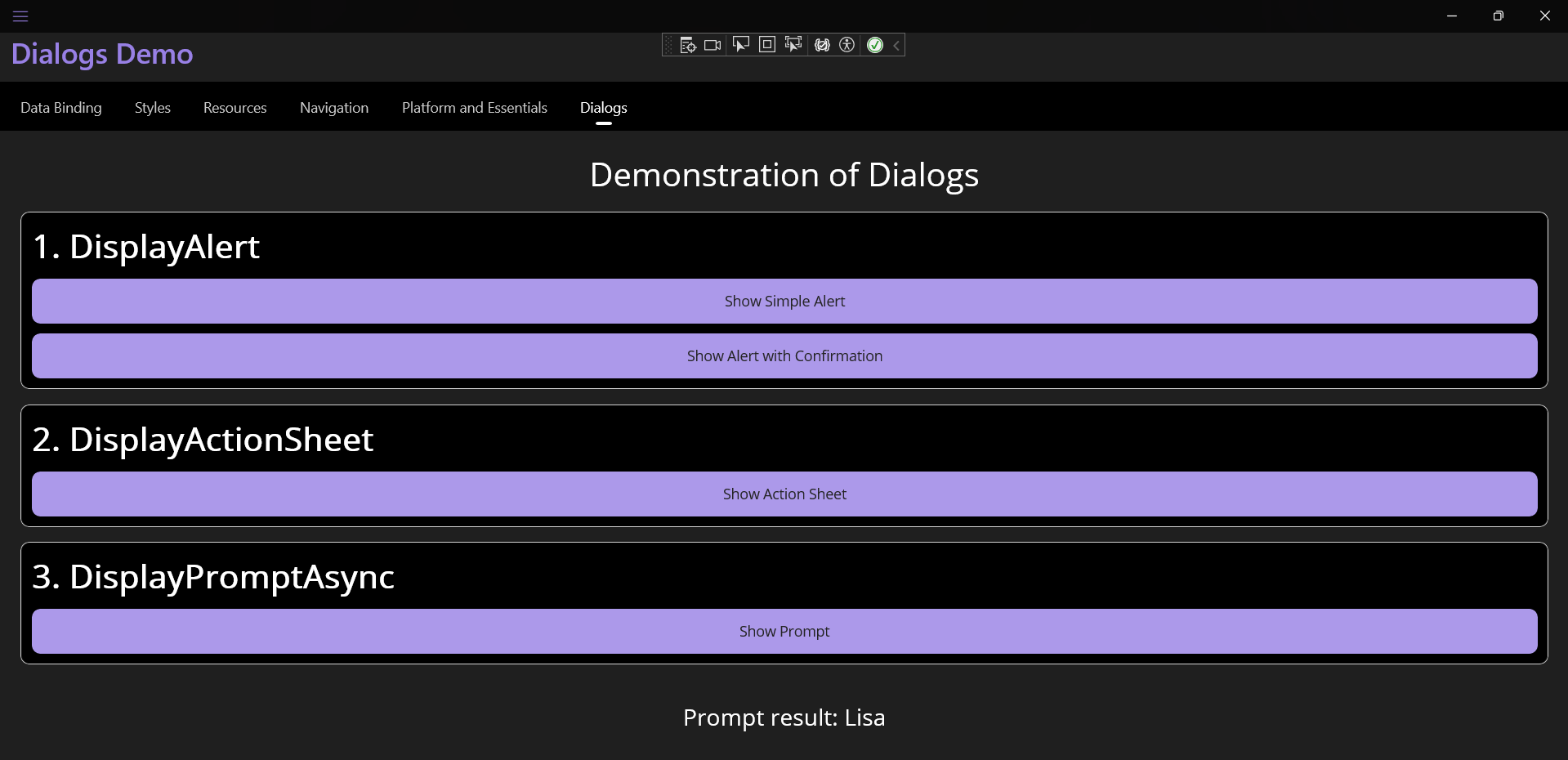
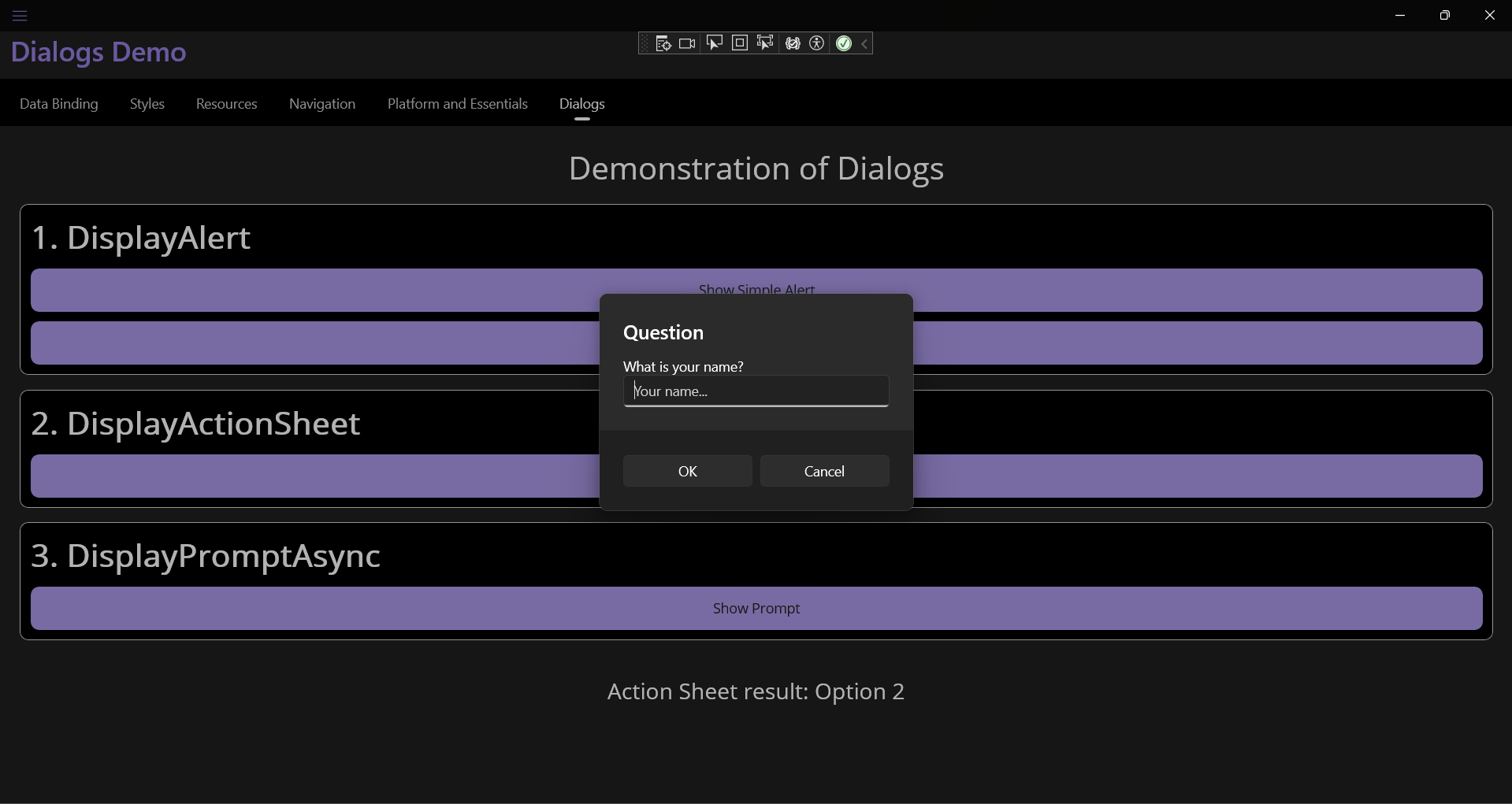
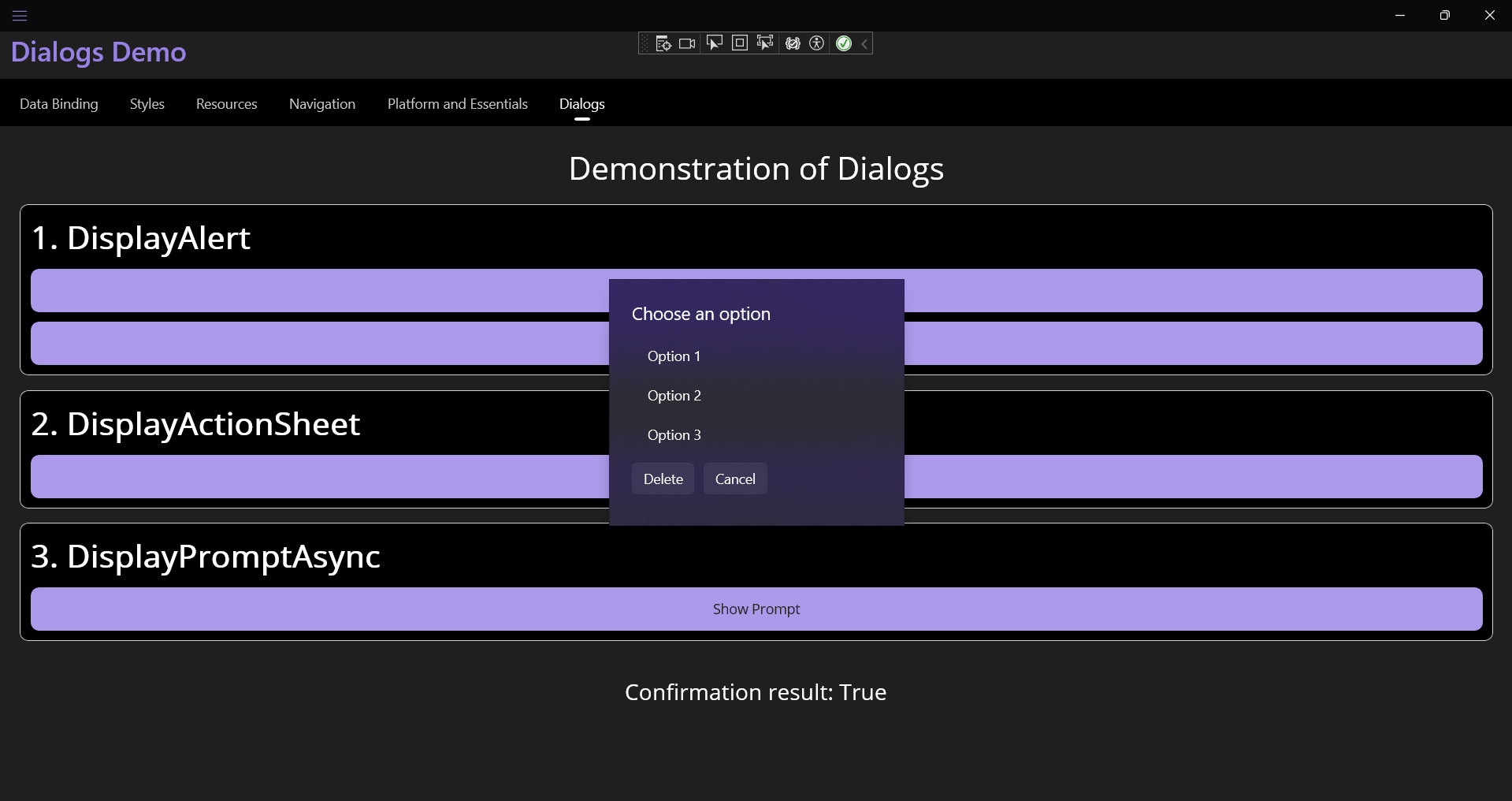
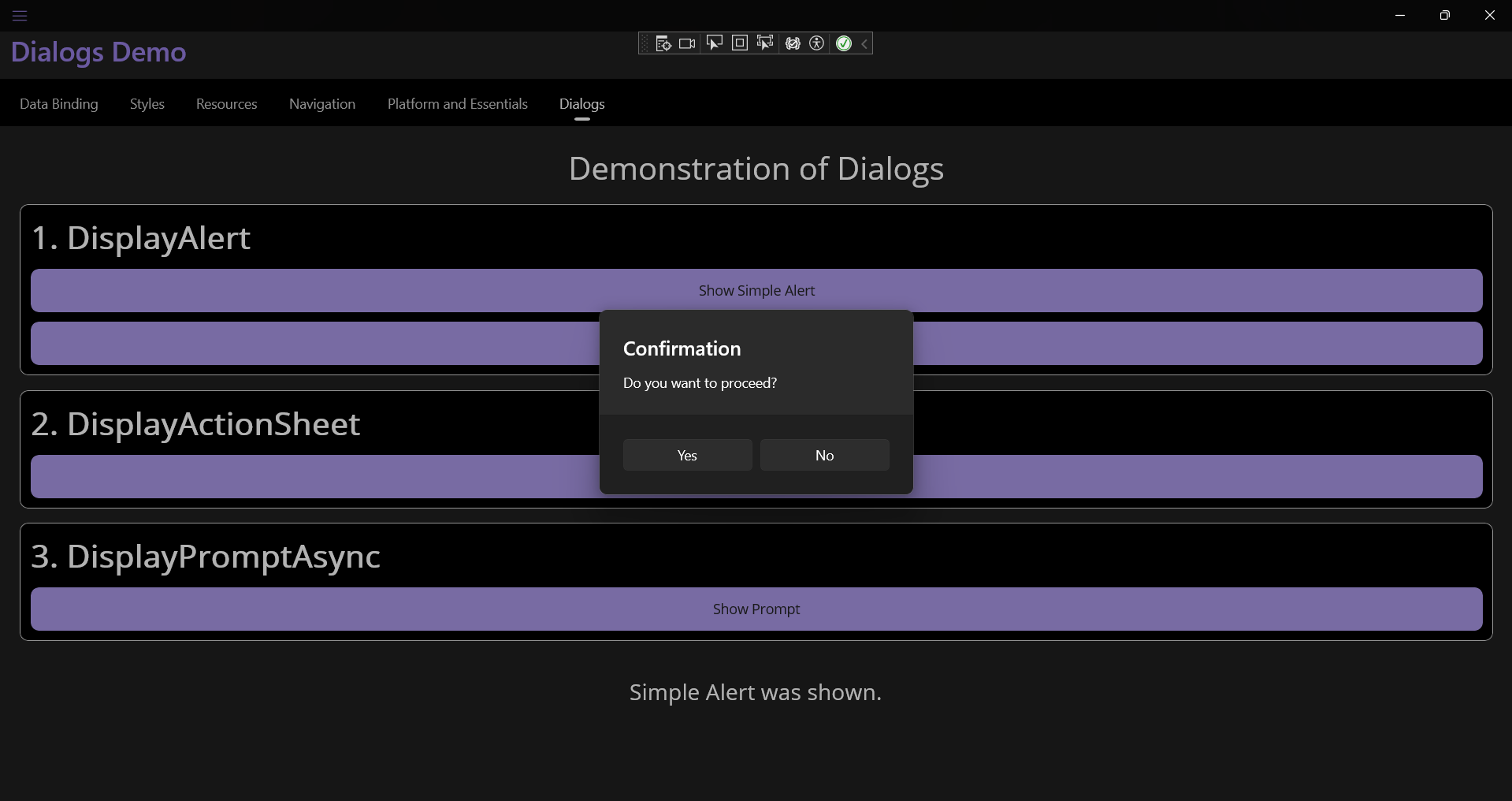
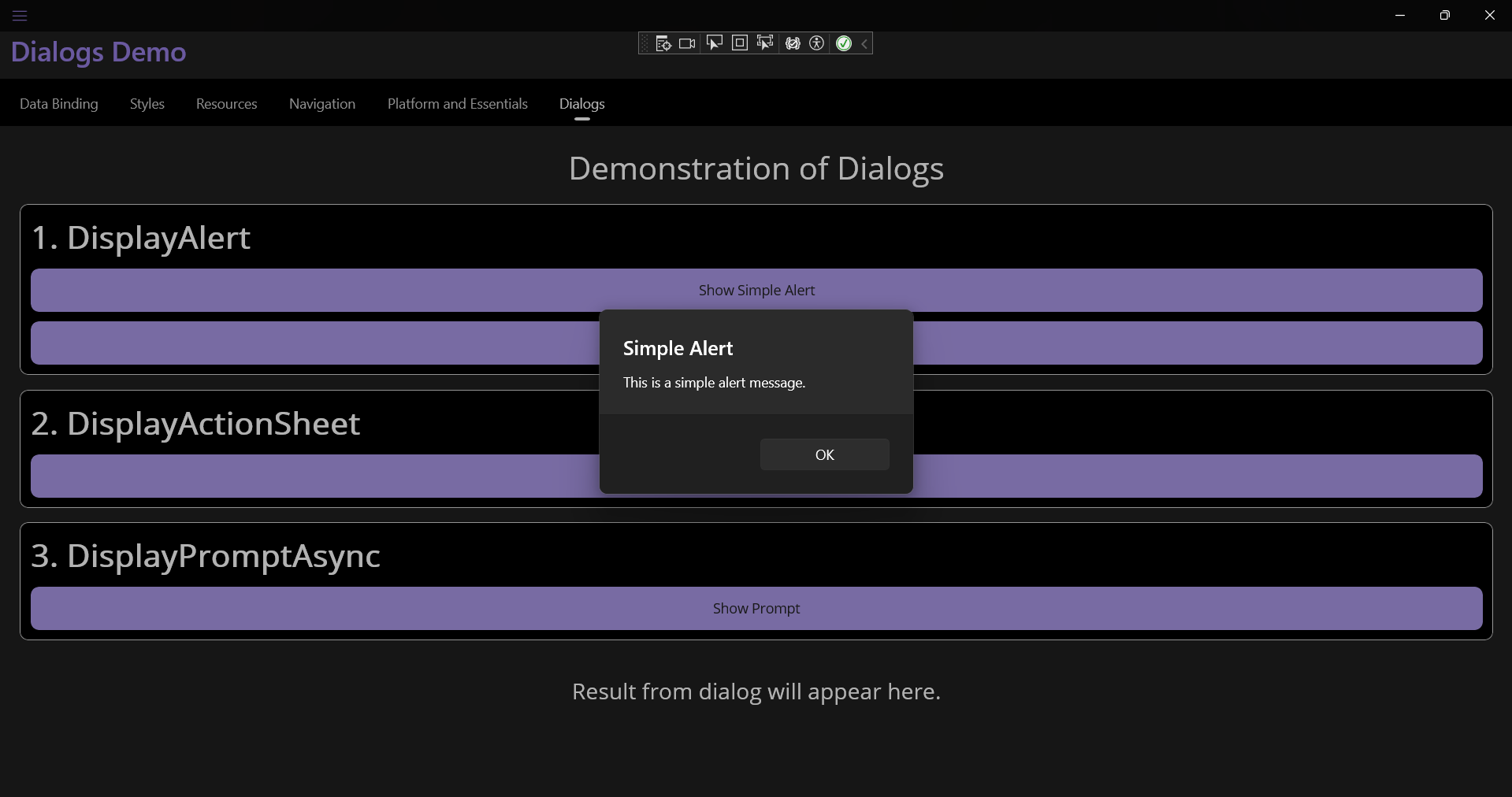
*Пояснение:* Каждый метод-обработчик является асинхронным (async), так как вызов диалоговых окон (Display...) является асинхронной операцией, возвращающей Task. Мы ожидаем результат с помощью await и затем обновляем Label на странице, чтобы показать, какое действие выполнил пользователь.

**3. Заключение**

Встроенные диалоговые окна .NET MAUI предоставляют простой и эффективный способ взаимодействия с пользователем для решения стандартных задач. Они обеспечивают нативное для каждой платформы поведение, что улучшает пользовательский опыт. Использование DisplayAlert, DisplayActionSheet и DisplayPromptAsync является базовым навыком при разработке любых приложений, включая настольные, где необходимо подтверждать действия, информировать об ошибках или запрашивать у пользователя простые данные.

**4. Скриншоты и примеры**

Демонстрация работы диалоговых окон была выполнена на странице DialogsDemoPage, запущенной на платформе Windows.

*******Рис. 10.1 - 10.6. Демонстрация работы DisplayAlert, DisplayActionSheet и DisplayPromptAsync.*

**Процесс выполнения примеров:**

1. Создана страница DialogsDemoPage с кнопками для вызова каждого типа диалога.
2. В code-behind реализованы асинхронные обработчики нажатия кнопок.
3. Каждый обработчик вызывает соответствующий метод (DisplayAlert, DisplayActionSheet, DisplayPromptAsync) и ожидает результат.
4. Полученный от пользователя результат (или факт закрытия окна) отображается в Label на странице.
5. Страница добавлена в AppShell.xaml и протестирована.  
   Все диалоговые окна успешно вызывались и корректно возвращали результат взаимодействия с пользователем.

**5. Документация**

* **Ссылка на GitHub с исходным кодом:** https://github.com/RaniZee/.NET-MAUI-Showcase-Cross-Platform-Applications
* **Описание выполненных шагов:**
  1. Изучены материалы по теме "Диалоги" (глава 10) с ресурса [metanit.com/sharp/maui/](https://www.google.com/url?sa=E&q=https%3A%2F%2Fmetanit.com%2Fsharp%2Fmaui%2F).
  2. Создана страница DialogsDemoPage для демонстрации.
  3. Реализованы примеры вызова DisplayAlert, DisplayActionSheet и DisplayPromptAsync.
  4. Продемонстрирована обработка результатов, возвращаемых этими асинхронными методами.
  5. Проведено тестирование на платформе Windows и подготовлены скриншоты.