Módulo 4: Elementos Visuales y Flujo de Interacción entre Pantallas

Lección: Selección de Fechas y Horas en Flutter: Uso de DatePicker y TimePicker

Objetivos de la Lección

- Comprender el uso del widget `DatePicker` para la selección de fechas, así como su personalización de formato.
- Aprender a implementar el widget `TimePicker` para la selección de hora y la configuración de tiempo en formatos de 12 y 24 horas.
- Conocer las propiedades y funcionalidades principales de cada widget.
- Aplicar ejemplos básicos en Flutter para cada uno de los widgets.

Introducción de la Lección

En Flutter, los widgets `DatePicker` y `TimePicker` son útiles para permitir que los usuarios seleccionen fechas y horas dentro de la aplicación. El `DatePicker` permite elegir una fecha específica, con opciones de personalización en el formato de visualización, y es ideal para aplicaciones que requieren la selección de fechas de eventos o recordatorios. Por otro lado, el `TimePicker` permite seleccionar una hora, con la posibilidad de configurarlo en formatos de 12 o 24 horas, siendo perfecto para aplicaciones que manejan horarios específicos. En esta lección, exploraremos cómo usar ambos widgets, sus propiedades, y cómo personalizarlos según las necesidades de la aplicación.

`DatePicker`: Selección de Fechas

El widget `DatePicker` permite que el usuario seleccione una fecha de un calendario emergente. Este widget es especialmente útil para aplicaciones que requieren la entrada de fechas específicas, como reservas, eventos o tareas.

Propiedades Principales de `DatePicker`:

- **initialDate**: La fecha que aparece seleccionada al abrir el `DatePicker`.
- **firstDate** y **lastDate**: Establecen el rango de fechas que el usuario puede seleccionar.
- **locale**: Define el formato y el idioma del `DatePicker`.

Ejemplo Simple de `DatePicker`:

En este ejemplo, se muestra cómo usar un botón que al presionarlo despliega un `DatePicker`. La fecha seleccionada se muestra en pantalla.

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: DatePickerExample(),
    );
  }
```

```
}

class DatePickerExample extends StatefulWidget {
  @override
  _DatePickerExampleState createState() =>
_DatePickerExampleState();
}

class _DatePickerExampleState extends State<DatePickerExample> {
  DateTime? selectedDate;

  Future<void> _selectDate(BuildContext context) async {
    final DateTime? picked = await showDatePicker(
      context: context,
      initialDate: DateTime.now(),
      firstDate: DateTime(2000),
      lastDate: DateTime(2101),
    );
    if (picked != null && picked != selectedDate)
      setState(() {
        selectedDate = picked;
      });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Ejemplo de DatePicker'),
      ),
```

```
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text(
              selectedDate == null
                  ? 'No se ha seleccionado ninguna fecha'
                  : 'Fecha seleccionada:
${selectedDate!.toLocal()}'.split(' ')[0],
            ),
            SizedBox(height: 20.0),
            ElevatedButton(
              onPressed: () => _selectDate(context),
              child: Text('Seleccionar fecha'),
            ),
          ],
        ),
      ),
    );
  }
}
```

En este ejemplo:

- **initialDate** establece la fecha actual como predeterminada.
- **firstDate** y **lastDate** limitan la selección entre los años 2000 y 2101.
- **showDatePicker** es la función que despliega el selector de fechas, mostrando la fecha seleccionada en la interfaz.

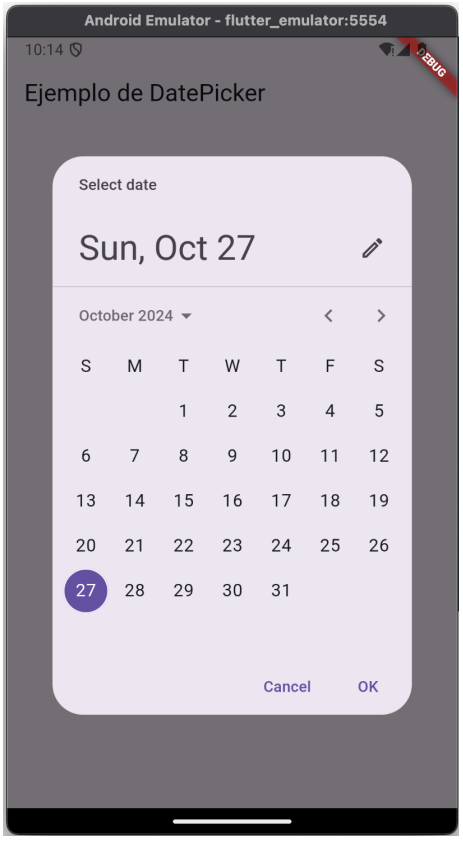

Imagen de la pantalla que incluye el Widget `DatePicker`.

Creado por Javier A. Dastas (2024)

`TimePicker`: Selección de Hora

El widget `TimePicker` permite que el usuario seleccione una hora, con la opción de visualizarla en formato de 12 o 24 horas. Este widget es ideal para aplicaciones que requieren la entrada de una hora específica, como recordatorios, alarmas o citas.

Propiedades Principales de `TimePicker`:

- **initialTime**: La hora que aparece seleccionada al abrir el `TimePicker`.
- **use24HourFormat**: Define si el formato de tiempo es de 24 horas o de 12 horas.
- **locale**: Define el formato y el idioma del `TimePicker`.

Ejemplo Simple de `TimePicker`:

En este ejemplo, se muestra cómo usar un botón que al presionarlo despliega un `TimePicker`. La hora seleccionada se muestra en pantalla.

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: TimePickerExample(),
    );
  }
}

class TimePickerExample extends StatefulWidget {
  @override
  _TimePickerExampleState createState() =>
_TimePickerExampleState();
}

class _TimePickerExampleState extends State<TimePickerExample> {
  TimeOfDay? selectedTime;

  Future<void> _selectTime(BuildContext context) async {
    final TimeOfDay? picked = await showTimePicker(
```

```
      context: context,
      initialTime: TimeOfDay.now(),
    );
    if (picked != null && picked != selectedTime)
      setState(() {
        selectedTime = picked;
      });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Ejemplo de TimePicker'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text(
              selectedTime == null
                  ? 'No se ha seleccionado ninguna hora'
                  : 'Hora seleccionada:
${selectedTime!.format(context)}',
            ),
            SizedBox(height: 20.0),
            ElevatedButton(
              onPressed: () => _selectTime(context),
              child: Text('Seleccionar hora'),
            ),
```

```
            ],
          ),
        ),
      );
    }
}
```

En este ejemplo:

- **initialTime** define la hora actual como predeterminada.
- **showTimePicker** es la función que despliega el selector de hora, mostrando la hora seleccionada en la interfaz.
- **format** permite visualizar la hora seleccionada en el formato correcto.

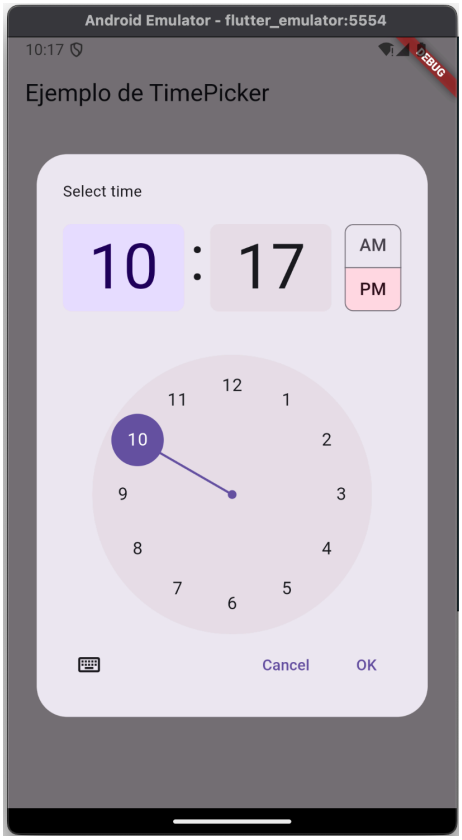

Imagen de la pantalla que incluye el Widget `TimePicker`.

Creado por Javier A. Dastas (2024)

Relación con Otros Conceptos

El uso de `DatePicker` y `TimePicker` es fundamental en aplicaciones que requieren la entrada de fechas y horas. Estos widgets son comúnmente utilizados en aplicaciones de calendario, eventos, y recordatorios, permitiendo a los usuarios realizar selecciones de tiempo de manera eficiente. En el diseño de UX/UI, su implementación es clave para una experiencia de usuario intuitiva y directa, especialmente cuando se trata de configurar alarmas o programar eventos.

Resumen de la Lección

En esta lección, exploramos los widgets `DatePicker` y `TimePicker` en Flutter. Aprendimos cómo permitir que los usuarios seleccionen fechas y horas, así como personalizar el formato de cada uno. Estos widgets son esenciales para aplicaciones que requieren entrada de tiempo y proporcionan una experiencia de usuario interactiva y personalizada.

Actividad de la Lección

En esta actividad, aplicaremos lo aprendido sobre los widgets `DatePicker` y `TimePicker` en Flutter para crear una aplicación que permita seleccionar fechas y horas, facilitando la personalización de eventos o recordatorios. Al integrar la selección de fechas y horas, los estudiantes podrán construir una interfaz que ofrece una experiencia completa de planificación.

La actividad tiene dos partes:

1. **Selección de Fecha con `DatePicker`**:
    - Configura un `DatePicker` que permita al usuario seleccionar una fecha para un evento.
    - Muestra la fecha seleccionada en pantalla, permitiendo ver la fecha del evento programado.
    - Esta parte permite practicar el uso de `DatePicker` en el contexto de planificación de eventos.
2. **Selección de Hora con `TimePicker`**:
    - Configura un `TimePicker` que permita al usuario seleccionar una hora para el evento previamente creado.
    - Muestra la hora seleccionada en pantalla para visualizar el tiempo exacto del evento.
    - Esta parte permite comprender el uso de `TimePicker` y trabajar con configuraciones de tiempo de manera interactiva.

Entrega de la Actividad:

- Desarrolla un documento en formato PDF que incluya el código desarrollado para esta actividad, capturas de pantalla de la aplicación en ejecución, y una breve descripción de cómo `DatePicker` y `TimePicker` ayudan a mejorar la funcionalidad de planificación en una aplicación de Flutter.

Con esta actividad, se busca consolidar el conocimiento en la creación de interfaces de usuario interactivas que combinan fechas y horas, proporcionando una experiencia de usuario enfocada en la gestión de eventos y recordatorios.