Módulo 4: Elementos Visuales y Flujo de Interacción entre Pantallas

Lección: Captura de Firma Digital en Flutter: Uso del `SignaturePad`

Objetivos de la Lección

- Comprender el propósito y funcionamiento del widget `SignaturePad` para la captura de firmas digitales en Flutter.
- Conocer las propiedades principales y personalización de `SignaturePad`.
- Aprender casos de uso para la captura de firmas digitales, como autenticación y documentos legales.
- Aplicar un ejemplo básico de implementación del widget en una aplicación Flutter.

Introducción de la Lección

El `SignaturePad` es un widget en Flutter que permite capturar firmas digitales dibujadas en pantalla, guardándolas como una imagen. Este widget es útil en aplicaciones que requieren autenticación mediante firma o la aceptación de documentos legales y contratos. La captura de firma digital es una herramienta importante para aplicaciones de negocios y servicios digitales, facilitando la validación de identidades y el acuerdo de términos desde cualquier dispositivo móvil.

Uso del Package `signature` en Flutter

El widget `SignaturePad` no es parte del conjunto de widgets predeterminados de Flutter, por lo que necesitamos un paquete externo llamado `signature` para agregar esta funcionalidad. El package `signature` proporciona el widget `Signature` y el controlador `SignatureController`, que permiten capturar la firma como un trazo dibujado en pantalla y convertirlo en una imagen.

Este paquete es esencial cuando queremos capturar, personalizar y manipular firmas digitales en nuestra aplicación, permitiendo configuraciones como el grosor de la firma, su color y su conversión a formato de imagen para almacenarla o enviarla. En el momento de escribir esta lección el paquete `signature` estaba en la `signature: ^5.5.0`.

¿Cómo Agregar el Package signature al Proyecto?

Para usar el `SignaturePad` en tu proyecto Flutter, sigue estos pasos para añadir el paquete `signature`:

1. **Abrir el archivo `pubspec.yaml`**:
    - En la raíz de tu proyecto Flutter, abre el archivo `pubspec.yaml`, que contiene las dependencias de tu proyecto.
2. **Agregar la Dependencia**:
    - Dentro de `dependencies`, agrega el paquete `signature`. Asegúrate de incluir la versión más reciente. Puedes encontrar la versión actualizada en pub.dev.

El archivo debe verse similar a este ejemplo:

```
dependencies:
  flutter:
    sdk: flutter
  signature: ^5.5.0  # Versión de ejemplo; revisa pub.dev
                     # para obtener la última versión.
```

Funcionalidades y Propiedades Principales de `SignaturePad`

El widget `SignaturePad` permite a los usuarios firmar usando el dedo o un lápiz digital, almacenando la firma como una imagen. Sus principales propiedades y funcionalidades incluyen:

- **backgroundColor**: Define el color de fondo del área de la firma.
- **penColor**: Configura el color de la firma (el trazo de la firma).
- **penStrokeWidth**: Establece el grosor del trazo de la firma.
- **clear()**: Borra la firma actual, permitiendo al usuario reiniciar el área de firma.
- **toImage()**: Convierte la firma dibujada en un archivo de imagen, facilitando el almacenamiento o envío de la firma capturada.

Casos de Uso de `SignaturePad`:

- **Autenticación**: Firmar para validar la identidad del usuario en la aplicación.
- **Aceptación de Documentos Legales**: Firmar para acordar términos de servicios, contratos o formularios digitales.
- **Recepción de Entregas**: Firmar para confirmar la recepción de pedidos o documentos.

Ejemplo Simple de `SignaturePad`

En este ejemplo, implementaremos un área de firma que permite al usuario dibujar su firma en pantalla. Incluye un botón para borrar la firma y otro para guardar la imagen de la firma.

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

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

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

class SignaturePadExample extends StatefulWidget {
  @override
  _SignaturePadExampleState createState() =>
_SignaturePadExampleState();
}

class _SignaturePadExampleState extends
State<SignaturePadExample> {
  final SignatureController _controller = SignatureController(
    penStrokeWidth: 3,
    penColor: Colors.black,
```

```
  exportBackgroundColor: Colors.grey[200]!,
);

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Ejemplo de SignaturePad'),
    ),
    body: Column(
      children: [
        Container(
          height: 200,
          width: double.infinity,
          color: Colors.grey[200],
          child: Signature(
            controller: _controller,
          ),
        ),
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              onPressed: () {
                _controller.clear();
              },
              child: Text('Borrar'),
            ),
            SizedBox(width: 10),
            ElevatedButton(
```

```
                onPressed: () async {
                  if (_controller.isNotEmpty) {
                    final signatureImage = await
_controller.toImage();
                    // Aquí podrías guardar o compartir la
imagen de la firma
                    print("Firma capturada en formato de
imagen");
                  }
                },
                child: Text('Guardar'),
              ),
            ],
          ),
        ],
      ),
    );
  }
}
```

En este ejemplo:

- **SignatureController**: Controla el estado del `SignaturePad`, incluyendo el color del trazo y el fondo.
- **penStrokeWidth** y **penColor**: Configuran el grosor y el color del trazo de la firma.
- **clear()**: El botón "Borrar" llama a este método para limpiar el área de firma.
- **toImage()**: El botón "Guardar" convierte la firma en una imagen, permitiendo su uso en otras partes de la aplicación.

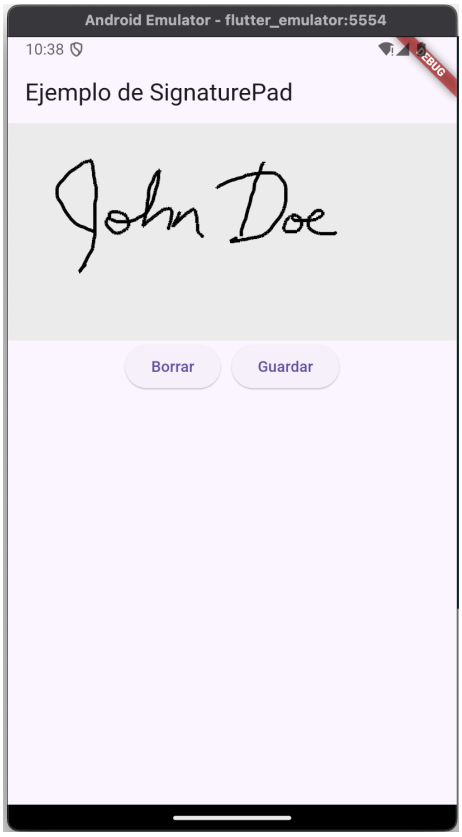

Imagen de la pantalla que incluye el Widget `SignaturePad`.

Creado por Javier A. Dastas (2024)

Relación con Otros Conceptos

El widget `SignaturePad` está estrechamente relacionado con la funcionalidad de captura de datos en aplicaciones móviles, lo que permite a los usuarios interactuar de manera más segura y confiable. En términos de diseño UX/UI, la captura de firma digital proporciona una experiencia de usuario simplificada para la validación y autenticación. Además, este widget es clave en aplicaciones de ingeniería de software que requieren acuerdos digitales de forma segura y conveniente.

Resumen de la Lección

En esta lección, aprendimos sobre el widget `SignaturePad` en Flutter, explorando sus propiedades y su implementación para capturar firmas digitales. Vimos cómo configurar

las opciones de personalización, como color y grosor del trazo, y cómo convertir la firma en una imagen para su almacenamiento o envío. Este widget es esencial para aplicaciones que requieren validación de identidad o aceptación de términos legales.

Actividad de la Lección

En esta actividad, aplicaremos lo aprendido sobre el widget `SignaturePad` en Flutter para crear una aplicación que permita capturar y guardar una firma digital. Esto permitirá a los estudiantes comprender cómo implementar la autenticación por firma y cómo personalizar el área de firma para que el usuario pueda interactuar fácilmente.

La actividad tiene dos partes:

1. **Captura de Firma con `SignaturePad`**:
    - Configura un área de firma en la aplicación donde el usuario pueda dibujar su firma.
    - Incluye un botón que permita al usuario borrar el área de firma en caso de error.
    - Esta parte refuerza el uso de `SignaturePad` para la captura de firma digital en aplicaciones de autenticación y validación.
2. **Guardar Firma como Imagen**:
    - Implementa un botón para guardar la firma en un archivo de imagen.
    - Verifica que el archivo de la firma capturada pueda ser guardado para su uso posterior en la aplicación.
    - Esta parte permite trabajar con la conversión de la firma en imagen y su manejo dentro de la aplicación.

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 explicación de cómo el `SignaturePad` puede mejorar la experiencia del usuario en la autenticación y aceptación de documentos.

Con esta actividad, se busca consolidar el conocimiento en la creación de interfaces interactivas y seguras que permitan la captura de firmas digitales, proporcionando una experiencia de usuario enfocada en la autenticación y validación de identidad.