Flutter

Gestures and User Input

Carmelo Villegas Cruz

¿Qué son gestos (Gestures)?

Cualquier evento producido por la interacción del usuario con la aplicación.

Entre lo más conocidos están.

- **Tap**, no es más que un toque en la pantalla.
- **Double Tap**, igual que el anterior pero haciendo un doble toque.
- **Long Press**, cuando pulsamos sin levantar por un periodo de tiempo.
- ...

¿Qué son gestos (Gestures)?

Toda la información referente a los gestos está en la documentación oficial.

Tarea
Acceder al enlace anterior y hacer un chuletario (cheatsheet) en formato pdf donde se describan todos los gestos que Flutter puede reconocer.

Manejando los gestos de los usuarios

En Flutter existen dos capas para la gestión de los gestos introducidos por el usuario.

- **Pointer layer**, son las capas que recogen los eventos del puntero (pointer), por ejemplo las coordenadas y el movimiento sobre la pantalla.

- **Gestures**, esta capa está por encima en cuanto abstracción a la anterior y recoge gestos predefinidos como tap, drags, scale …

Pointers

De las dos capas que existen para el manejo de gestos está es de un menor nivel de abstracción.

En esta capa, somos nosotros a través de la información del puntero lo que decidimos como controlar el evento.

Por ejemplo si recogemos la información sobre un toque de un usuario podríamos decidir si pensamos que un tap o un drag.

Pointers

Tenemos cuatro tipo de pointer events:

1. **PointerDownEvent**, es donde la interacción comienza.
2. **PointerMoveEvent**, si no levantamos el dedo y lo desplazamos por la pantalla.
3. **PointerUpEvent**, cuando levantamos el dedo.
4. **PointerCancelEvent**, por algún motivo se cancela la gestión del evento.

¿Pero cómo recogemos esos eventos?

Tenemos que hacer uso de **Listener.**

Veamos un ejemplo, para ellos usaremos:

```
flutter create --sample=widgets.Listener.1 nombre_proyecto
```

Gestures

Para manejar gestos *comunes*, **Flutter** añade una segunda capa donde se reconocen múltiples *pointer events:*

- Tap y Double Tap.
- Long press.
- Drag (horizontal y vertical) and pan
- Scale

Para manejar dichos eventos Flutter nos provee de GestureDectector.

Tap

```
@override
Widget build(BuildContext context) {
 return GestureDetector(
   onTap: () {
     setState(() {
      _counter++;});
   },
   child: Container(
     color: Colors.grey,
     child: Center(
       child: Text(
         "Tap contador: $_counter",
         style: Theme.of(context).textTheme.headlineLarge,
       ), ),
   ), );
} }
```

Double Tap

```
@override
Widget build(BuildContext context) {
 return GestureDetector(
   onDoubleTap: () {
     setState(() {
      _counter++;});
   },
   child: Container(
     color: Colors.grey,
     child: Center(
       child: Text(
         "Double Tap contador: $_counter",
         style: Theme.of(context).textTheme.headlineLarge,
       ), ),
   ), );
} }
```

Long Press

```
@override
Widget build(BuildContext context) {
 return GestureDetector(
   onLongPress: () {
     setState(() {
      _counter++;});
   },
   child: Container(
     color: Colors.grey,
     child: Center(
       child: Text(
         "Double Tap contador: $_counter",
         style: Theme.of(context).textTheme.headline1,
       ), ),
   ), );
} }
```

Drag - Horizontal

```
GestureDetector(
   onHorizontalDragStart: (DragStartDetails details) {
     setState(() {
          _move = Offset.zero;
          _dragging = true;
   },
   onHorizontalDragUpdate: (DragUpdateDetails details) {
     setState(() {
          _move += details.delta;
   },
   onHorizontalDragEnd: (DragEndDetails details) {
     setState(() {
          _dragging = false;
          _dragCount++;
   },
   child: …
```

Pan

```
GestureDetector(
    onPanStart: (details) {
        //do something
    },
    onPanUpdate: (details) {
        //do something
    },
    onPanEnd: (details) {
        //do something
    },
    child: …
```

Scale

```
GestureDetector(
   onScaleStart: (ScaleStartDetails details) {
     setState(() {
          _scale = 1.0;
          _scaling = true;
   },
   onScaleUpdate: (ScaleUpdateDetails details) {
     setState(() {
          _scale = details.scale;
   },
   onScaleEnd: (ScaleEndDetails details) {
     setState(() {
          _scaleCount++;
          _scaling = false;
   },
   child: …
```

Tarea

Para entender los conceptos vistos anteriormente vamos a implementar una App que solo tiene una pantalla donde muestra una lista de elementos (ListView). Para cada elemento añadiremos la funcionalidad que al hacer un desplazamiento de derecha a izquierda, dicho elemento se borre del listado. Los pasos a seguir serían:

1. Crear una lista de elementos.
2. Habilitar un gesto de desplazamiento horizontal a cada uno de los elementos de la lista cambiando el color.
3. Envolver cada elementos en un Dismissible widget.
4. El elemento se pinta de rojo cuando desplazamos.