Conceptos en Dart y Flutter

Para el resguardo de información al cerrar la aplicación: modificar el pubscep.ymal y poner la dependencia de los datos.

```
dependencies:
  flutter:
    sdk: flutter
  shared_preferences: ^2.1.1
```

Después de pegarlo, guardar y pulsar en la barra que aparece arriba *Pub upgrade* y *Pub get* (en ese orden).

Widget: piezas de interfaz reutilizables. Pueden contener otros widgets.

Build(): se encarga de construir la interfaz visual (UI) del widget (qué se muestra, en qué orden y cómo se organiza).

Async: se usa para marcar una función como asíncrona, es decir, la función puede esperar resultados que tardan un tiempo sin bloquear la interfaz de usuario.

Await: complemento para la función asíncrona, pausa la ejecución de la función hasta que la operación termina, pero sin bloquear el resto de la app.

RunApp: es la función que arranca la aplicación, siendo el punto de entrada principal (similar al main() en otros lenguajes). Coge un widget raíz y lo muestra en la pantalla.

StatelessWidget: widget que no cambia después de crearse. Muestra información fija (o calculada una sola vez) y si hubiera algún cambio, el widget debe ser reconstruido desde fuera. No tiene variables que cambien con el tiempo, sólo tiene el método build(), es ideal para componentes estáticos.

StatefulWidget: a diferencia del anterior, este widget si que cambia con el tiempo, tiene una parte dinámica llamada State donde se almacenan las variables que se guardan. Cuando el estado cambia, se reconstruye la interfaz con los nuevos datos.
Tiene dos clases:

- MyStateFulWidget: la estructura.
- _MyStatefulWidgetState: el estado (las variables que cambian).

setState() avisa a flutter que ha habido un cambio y debe actualizar la pantalla.

ShowDialog: muestra un cuadro de diálogo modal encima de la interfaz actual (una ventana emergente). Bloquea temporalmente la interacción con el resto de la app y devuelve un valor cuando se cierra.

Por Agustín Alcalde Reyes y Olga Peláez Calvo

Context: es un objeto que le dice a un widget dónde está ubicado dentro del árbol de widgets, sirve para acceder a información o funcionalidades que dependen del árbol de widgets.

GestureDetector: es un widget invisible que detecta gestos del usuario sobre cualquier parte de la interfaz. Al ser invisible, no muestra nada por sí mismo, pero envuelve otros widgets para hacerlos interactivos.

onTap: es una función callback que se ejecuta cuando el usuario da un toque (pulsación corta) sobre él, no detecta arrastres ni pulsaciones largas.

Scaffold: como el esqueleto de tu pantalla, es un widget base que proporciona una estructura visual básica con espacios predefinidos para appBar (barra superior), Body (contenido principal), FloatingActionButton (botón flotante), Drawer (menú lateral), BottonNavigationBar (barra de navegación inferior), Snackbars (notificación temporal en la pantalla que desaparece tras unos segundos y no interrumpe la interacción del usuario)…

ElevatedButton: es un botón normal, pero el diseño es elevado y crea sombra. Se suele usar para botones que hacen funciones importantes.

CrossAxisCount: es una propiedad de GridView para indicar cuántas columnas quieres que tenga la cuadrícula.

EdgeInsets: es una clase que se usa para definir márgenes o espacios internos (padding) en un widget.

InitState: es un método del ciclo de vida de un StatefulWidget que se llama una sola vez cuando se crea el estado del widget. Suele usarse para inicializar variables, controladores, listeners o hacer tareas antes de que se construya la interfaz por primera vez.

Timer.periodic: es un constructor de la clase Timer que permite ejecutar una función de forma repetida cada cierto intervalo de tiempo.

dispose: es un método del ciclo de vida de un StatefulWidget que se llama cuando el widget se elimina la pantalla para limpiar recursos.

Future.delayed: es una función que permite ejecutar código después de un tiempo determinado, sirve para esperar un tiempo antes de hacer algo sin bloquear el hilo principal.

SizedBox: es un widget que se usa principalmente para dar un tamaño fijo a un widget hijo y agregar espacio vacío entre widgets.

DraggableScrollableSheet: te permite envolver contenido dentro de una hoja que se puede arrastrar para ajustar su tamaño dentro de ciertos límites.

Physics(BouncingScrollPhysics): physics controla cómo se comporta el desplazamiento (scroll) de widgets desplazables (BouncingScrollPhysics añade un rebote elástico al llegar al final o al inicio del contenido).

ListTile: es un widget que representa un único elemento de lista con estructura prediseñada (leading (icono o avatar al inicio), title (texto principal), subtitle (texto secundario debajo del título), trailing (icono o widget al final), onTap/onLongPress (gestos para interacción)).

MaterialApp: punto de inicio de una aplicación Flutter que usa el estilo Material Design (el diseño creado por Google); es el contenedor principal que organiza la navegación, el tema, y la estructura general de la app.

debugShowCheckedModeBanner: en una app de Flutter, de forma predeterminada aparece una etiqueta roja de “debug” en la esquina superior derecha, para indicar que la app sigue en desarrollo. Esta función (propiedad de MaterialApp) hace que la etiqueta no aparezca si es falso (*debugSowCheckedModeBanner : false;*).

home: define cuál será la pantalla predeterminada, es decir, la primera que se mostrará al iniciar la aplicación.

PageController: te permite moverte entre páginas manualmente o por código, saber en qué página estás actualmente y escuchar los cambios de página.

SharedPreferences: es un almacenamiento local clave-valor que se guarda en el dispositivo, no es una base de datos, es para datos simples y ligeros.

mainAxisAlignment: es una propiedad de Row o Column, controla cómo se distribuyen los hijos dentro del contenedor en la dirección principal.

WidgetsBindingObserver: es una interfaz que te permite escuchar eventos del sistema y del framework desde tu widget (como cambios en el estado de la app, de la orientación de la pantalla, visibilidad del teclado…), notifica y da control cuando algo cambia fuera de tu widget.

WidgetsBinding: actúa como puente entre Flutter y el motor de renderizado, es decir, conecta tu código de widgets con el framework y el motor nativo.

_loadProgress: es una variable que representa el progreso de carga de un recurso (indica cuánto del recurso se ha cargado.

Mounted: es una propiedad booleana que pertenece a la clase State. Indica si el State todavía está insertado en el árbol de widgets. Se suele utilizar para evitar errores al actualizar el estado de un widget que ya fue eliminado.

AppLifecycleState: indica en qué estado se encuentra tu app respecto al sistema operativo:

- resumed: la app está en foreground y activa, el usuario puede interactuar.
- inactive: la app está en foreground pero inactiva (como cuando llega una llamada).
- paused: la app está en background, el usuario no la ve pero sigue en memoria.
- detached: la app ya no está en el árbol de widgets.

addPostFrameCallback: es un método de WidgetsBinding que permite registrar un callback que se ejecuta después de que el framework haya terminado de construir y renderizar el widget actual (se podría usar para obtener el tamaño o posición de un widget tras renderizar, mostrar diálogos o snackbars justo después de construir la pantalla…)

BarrierDismissible: es un booleano que define si el usuario puede cerrar el diálogo o modal tocando fuera de él.

.pop: es un método de Navigator que elimina la pantalla actual del stack de navegación (regresa a la pantalla anterior).