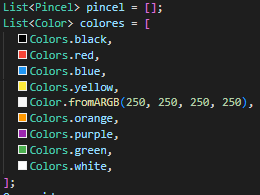
Hola buenos dias.

Este es el documento correspondiente al trabajo de Flutter en el cual se explica el proceso de creación del programa que desarrollamos mi compañero Cristopher Alexander Flores Miranda y yo Jorge Alberto Maldonado Castillo una vez aclarado esto, los dejamos con la explicación.

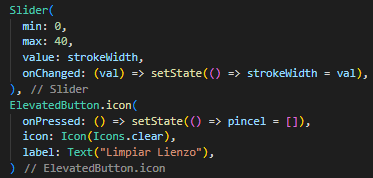
El proceso para la creación de la aplicación fue iniciando primero por temas estéticos simples. Empezamos creando una clase llamada Lienzo en la que daremos un diseño a la página, una vez creado y para tenerlo ya programado, vamos a definir un color predeterminado para lo que es el pincel con las siguientes 2 líneas de código:

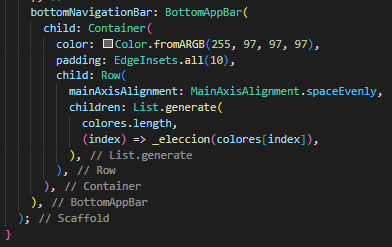


La variable selección y strokeWidth tendrán funcionalidad más adelante. Una vez hecho esto procedemos a crear una lista (Colors) con los colores que el usuario tendrá a su disposición a su vez que creamos la lista Pincel. 

En el centro de la lista está el color que cumplirá la función de un borrador más adelante, ahora solo queda crear el widget que permitirá la interacción con la pantalla, creamos un body tipo Stack para tener capas donde trabajar y vamos a crear otras 3 funciones (OnPanStart/Update/End) dentro de un GestureDetector, que detectara interacciones con la pantalla y que indicaremos más adelante que hacer.

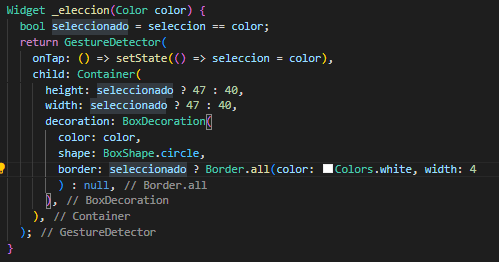
Ya teniendo esto crearemos 2 funciones vitales para un lienzo digital, primero crearemos un Slider que nos permitirá controlar el grosor del pincel con el que dibujamos (maximo de 40) y un BottomAppBar qué limpiará el lienzo con un .clear() mediante el siguiente código:



Teniendo esto podemos crear el apartado donde el usuario podrá escoger sus colores, este será de tono grisáceo para evitar un conflicto estético con los otros colores. En la misma vamos a invocar la lista mediante un children para que muestr los colores que añadimos anteriormente:

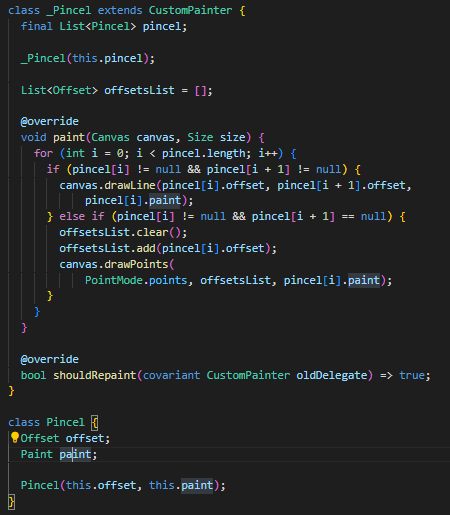
Ahora tenemos la interfaz creada, es hora de darle funcionalidad el 95% de la funcionalidad a la misma.

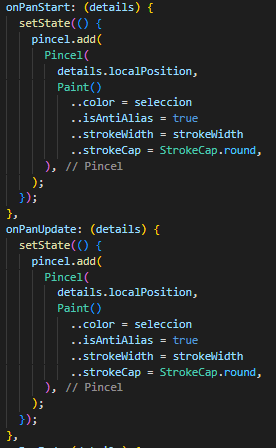
Creamos el widget \_eleccion que interactuará directamente con la lista de colores que creamos y dimos un espacio en la interfaz, una vez hecho esto con un GestureDetector vamos a definir el tamaño de los iconos, luego con un BoxDecorations haremos que extraiga los colores de la lista, los ponga en forma de círculo y añadiremos la separación entre cada uno tomando en consideración el centro, y para un toque estético añadimos una orilla de color blanco. Para que esto se muestre en pantalla fuera del GestureDetector vamos a añadir una variable tipo bool llamada seleccionado, una vez hecho esto y al inicio del Gesture Detector crearemos una función onTap() para que detecte los toque en la pantalla y así los cambios realizados anteriormente surtan efecto:



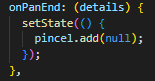
Ahora estamos con la parte más importante del código. Crearemos la clase Pincel y dentro de esta la lista Offset con la que evitaremos un problema más adelante. Creamos un void donde estará un ciclo anidado, que permitirá que el programa repita la función de onTap() indefinidamente mientras tengamos presionada la pantalla donde le indicamos.

Luego de eso creamos una clase Pincel para que resguarde algunas variables precedidas con this. para evitar posibles conflictos de nombre:

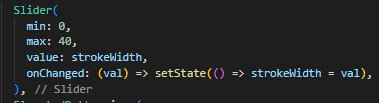


Ya con esto podemos darle funcionalidad a las 3 funciones que creamos casi al inicio del programa. Para onPanStar y onPanUpdate invocamos la variable pincen, que con un .add usamos un details.localPosition para detectar el lugar donde estamos tocando la pantalla, seguido de un Paint donde estaremos especificando el grosor, el color con el que se está pintando y con strokeCap se dará la instrucción para que detecte el momento donde dejamos de tocar la pantalla. 

Para onPanUpdate basta con añadir un pincel.add() y declararlo como “null”, para que a la hora de tocar otra zona de la pantalla no se generen lineas rectas:



Todo este proceso se hace mediante un setState en las 3 funciones.

Como último paso vamos a configurar el slider que añadimos con anterioridad, en esta está definida la variable strokeWidth con la que se modificará el grosor del pincel, que va creciendo de menor a mayor hasta un máximo de 40.

Con esto el paint con flutter estará 100% funcional.