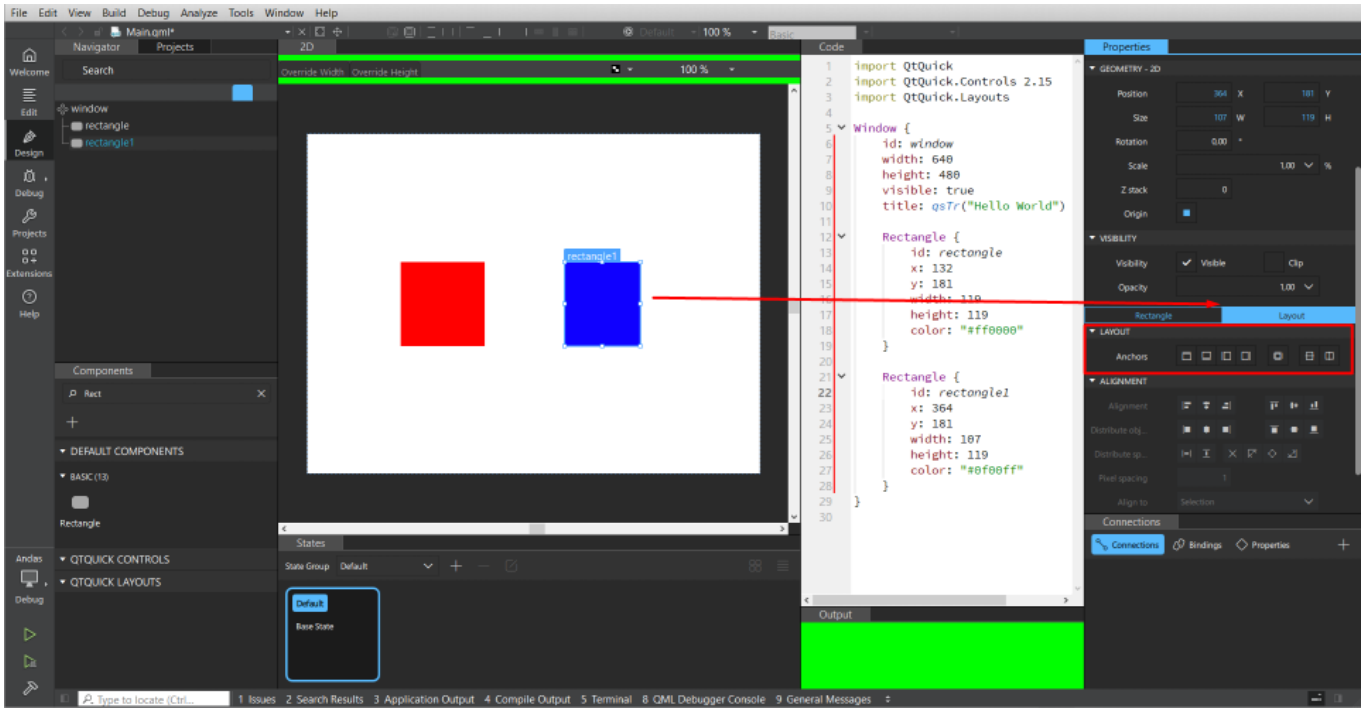
Posicionamientos en Qt

Índice

Posicionamientos en Qt

Anclas

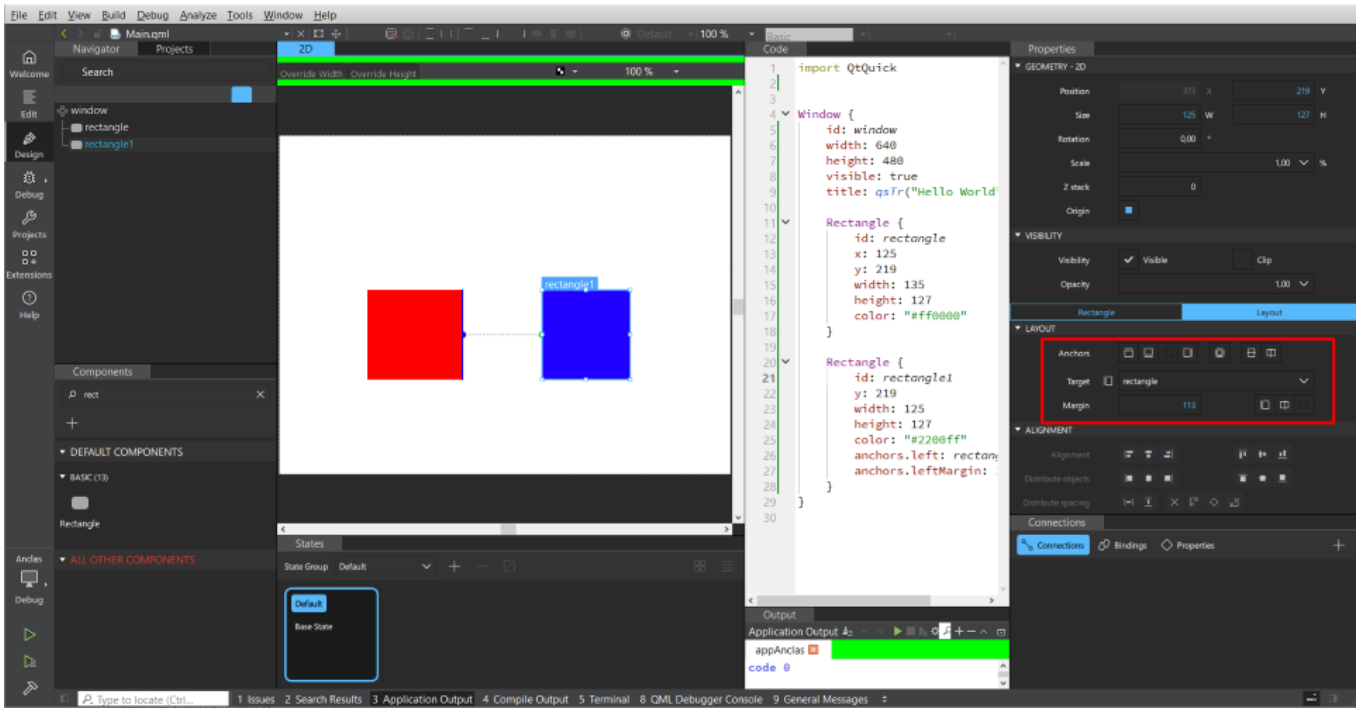
Las *anclas* son mecanismos que nos permiten posicionar elementos respecto a su contenedor o a otros elementos adyacentes. De esta manera, cuando cambie la posición de uno de ellos, la posición del que está anclado también cambiará.



Disponemos de varios tipos de anclas

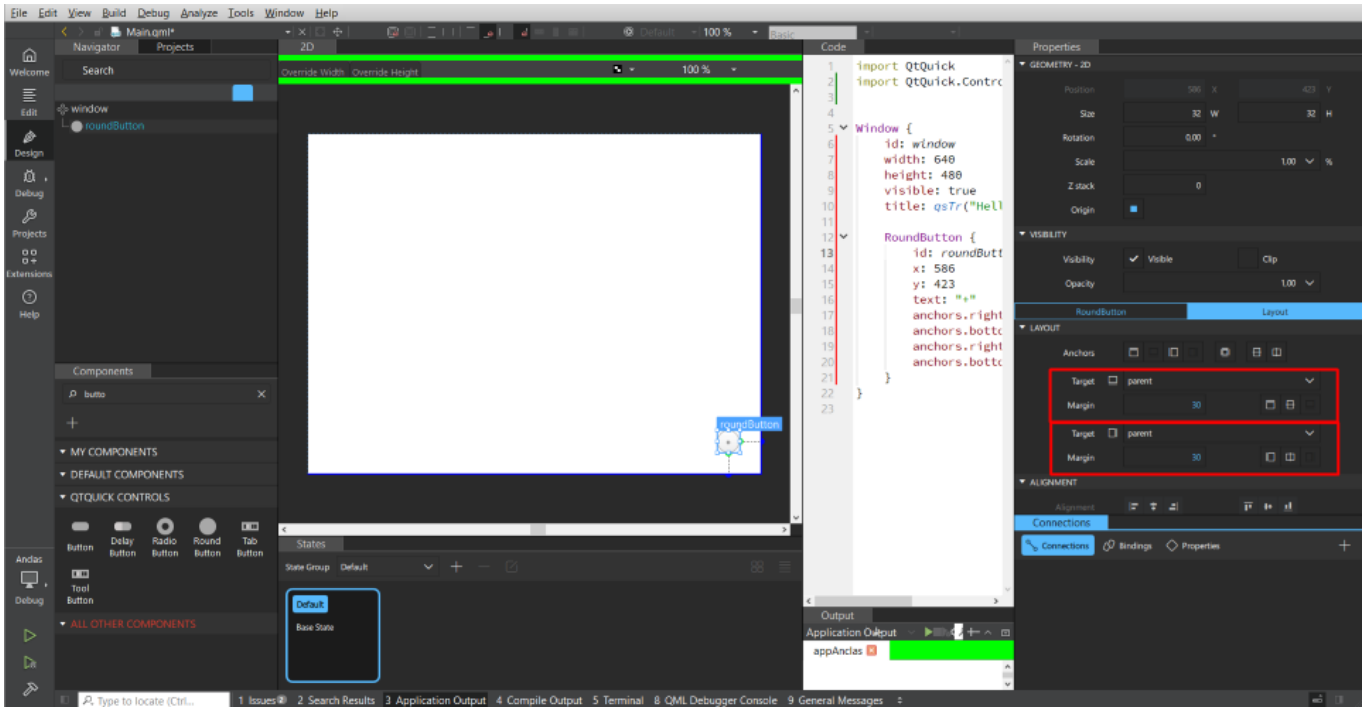
- **Left, top, right, bottom**. El ancla hará que la parte de elemento especificada se alinee con el otro elemento.
- **Fill Parent.** Hará que el elemento se ancle a su elemento padre, por lo que ocupará todo su espacio.
- **Vertical y Horizontal.** Alineará vertical u horizontalmente el elemento sobre el que esté anclado.

Creamos un ancla izquierda desde el cuadrado azul al cuadrado rojo.



Esta ancla implicará que el cuadrado azul estará a 113 píxeles del lado izquierdo del cuadrado rojo **siempre**.

También podemos anclar los elementos respecto a su elemento padre (en este caso la Ventana). En este caso, aunque hagamos crecer la ventana, el botón estará siempre en la misma posición respecto a ella. **Target: parent**, hace referencia al elemento inmediatamente superior.

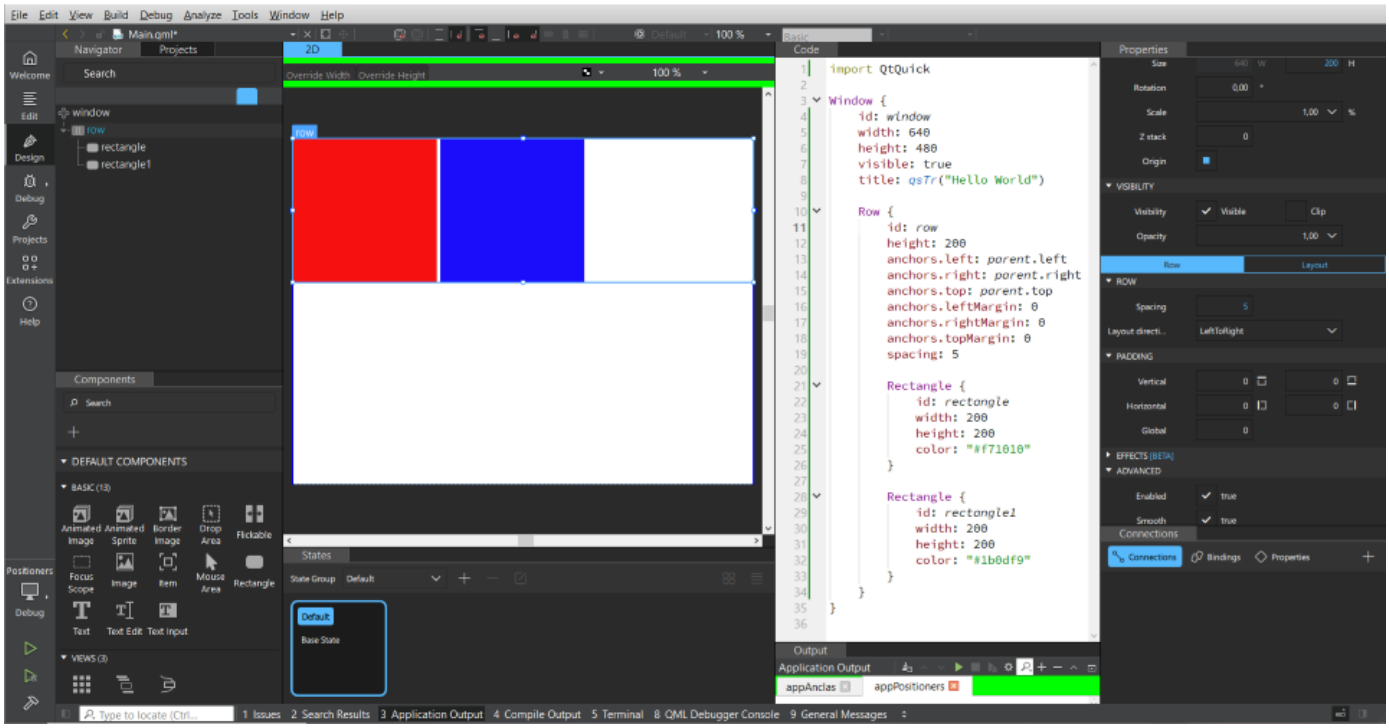


Posicionadores

Los posicionadores son contenedores que nos ayudan a gestionar el posicionamiento de los elementos dentro de la interfaz.

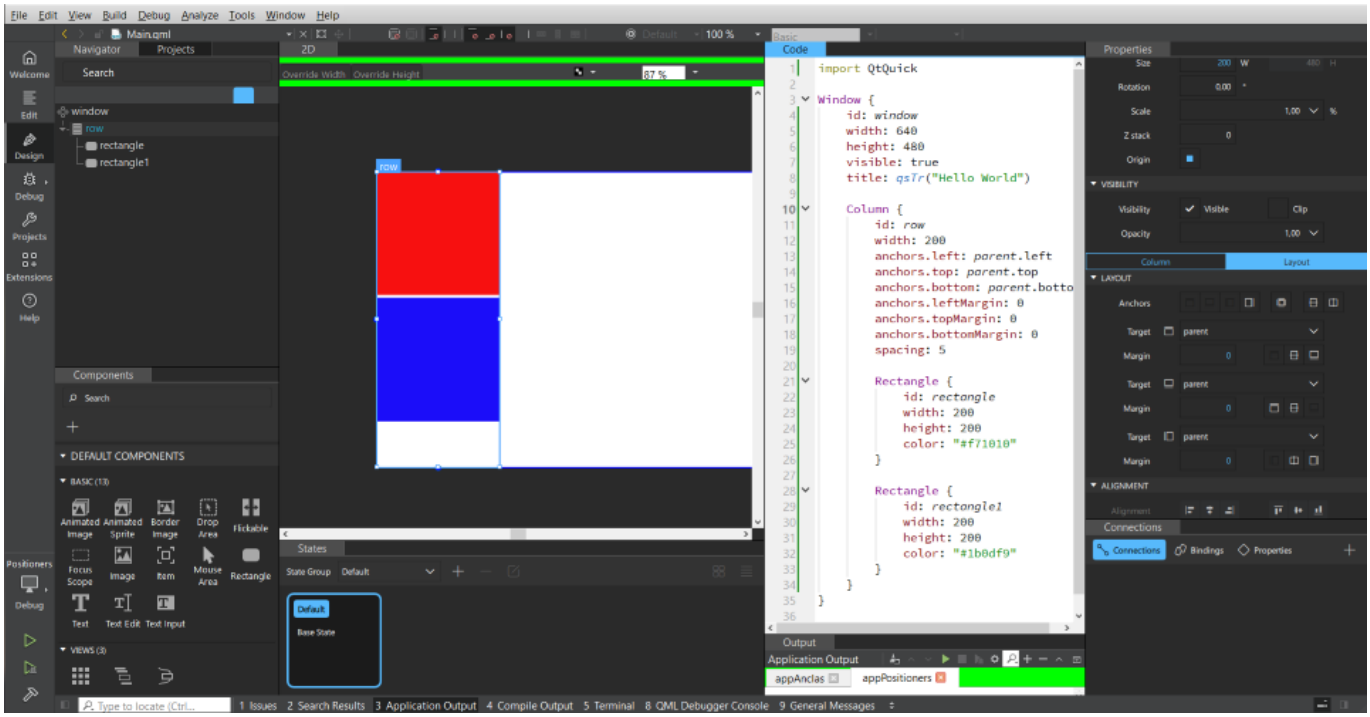
Row

El componente fila mostrará los elementos en disposición horizontal. Dentro de la fila, se ha definido la propiedad *spacing*, que creará automáticamente una pequeña distancia entre los elementos



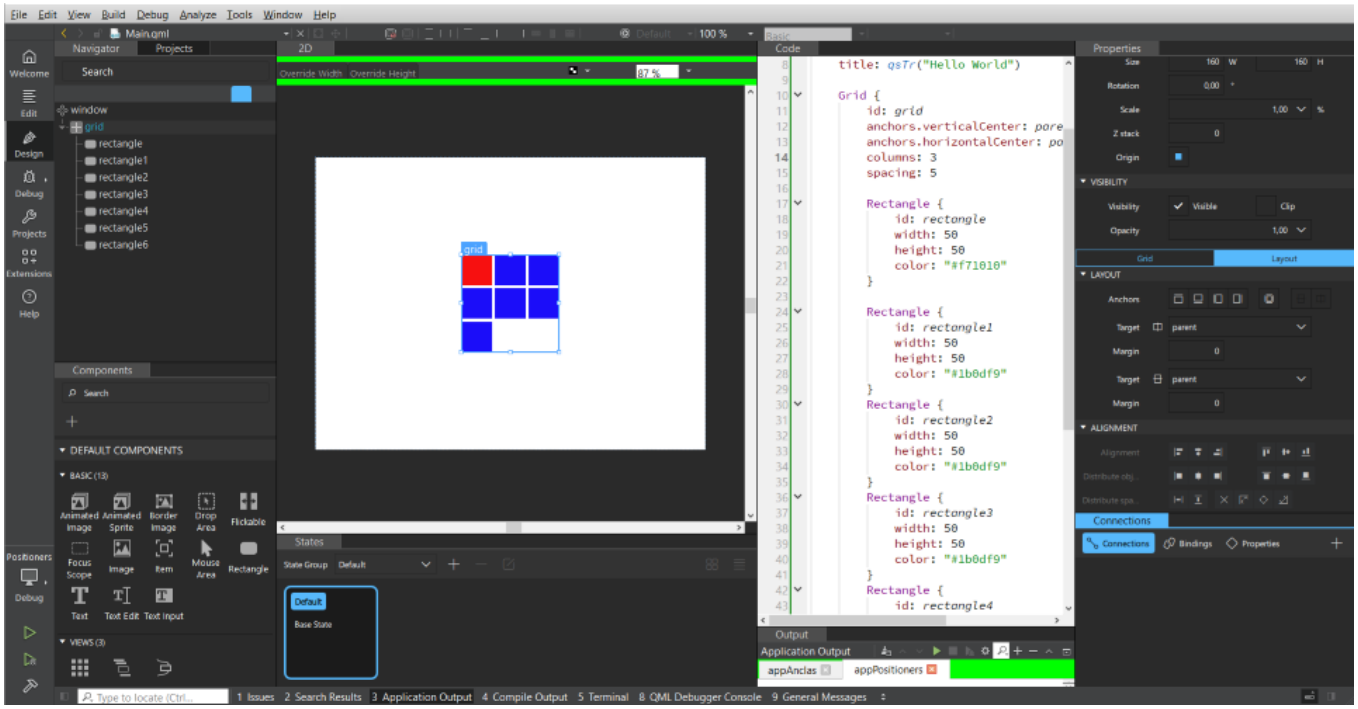
Column

El componente columna dispondrá los elementos de forma vertical.



Grid

El componente rejilla dispondrá los elementos de forma horizontal y vertical, según definamos los elementos que deben aparecer por fila/columna. Podemos definir las propiedades *columns* y *rows*, de forma que la rejilla mostrará tantos elementos como hayamos definido

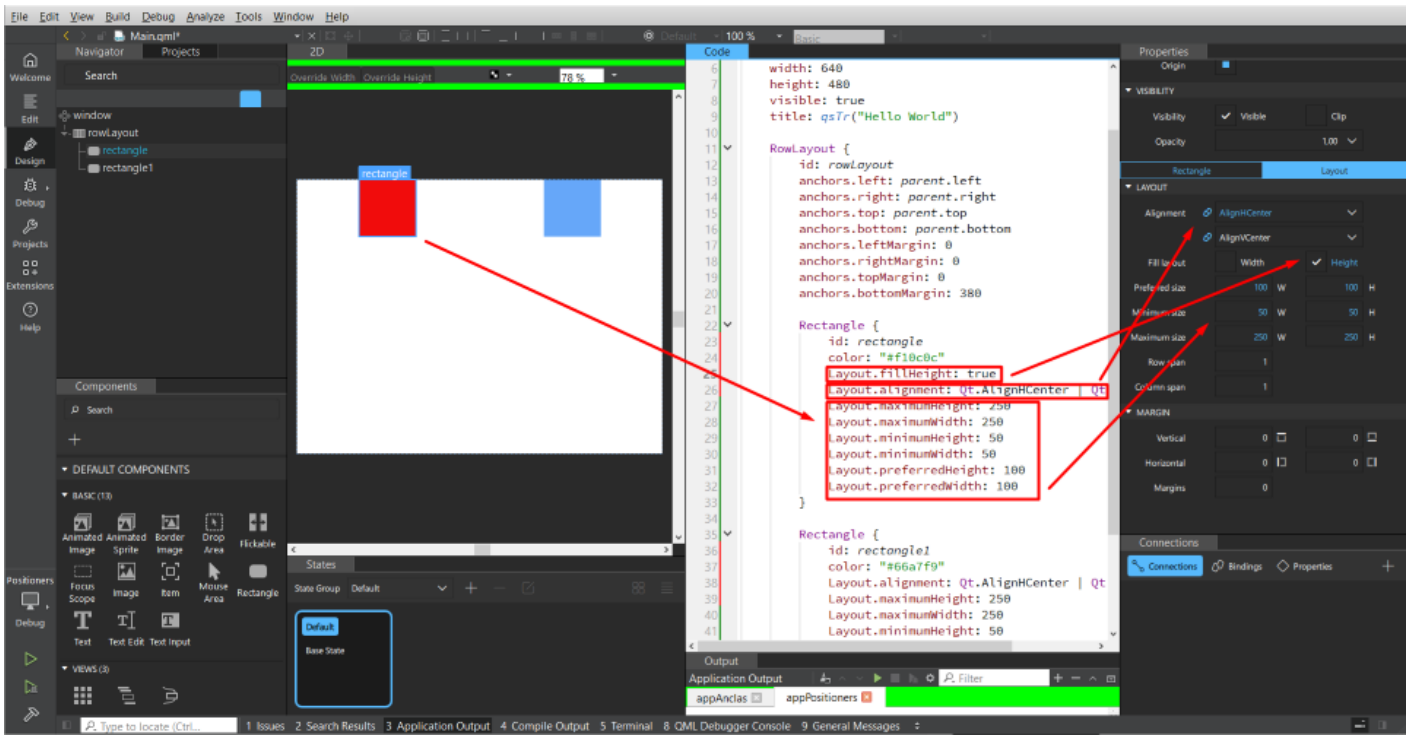


Layouts

Los layouts son componentes utilizados para elegir la disposición de los elementos en nuestra interfaz. A diferencia de los posicionadores, los layouts permiten redimensionar sus elementos hijos, lo que los hace perfectos para interfaces que puedan cambiar de tamaño.

RowLayout

Para todos los tipos de Layouts que vamos a ver, hay que tener en cuenta que **no** tenemos que especificar el alto y el ancho como height y width como en otros componentes. Las dimensiones de los hijos dentro de la fila se gestionan desde la pestaña de Layout.



Mediante anclas, hemos definido que nuestra Row esté ligada a la parte de arriba/izquierda/derecha de la ventana.

El rectángulo ocupa todo el alto del Row y está alineado tanto vertical como horizontalmente dentro del espacio que se le da.

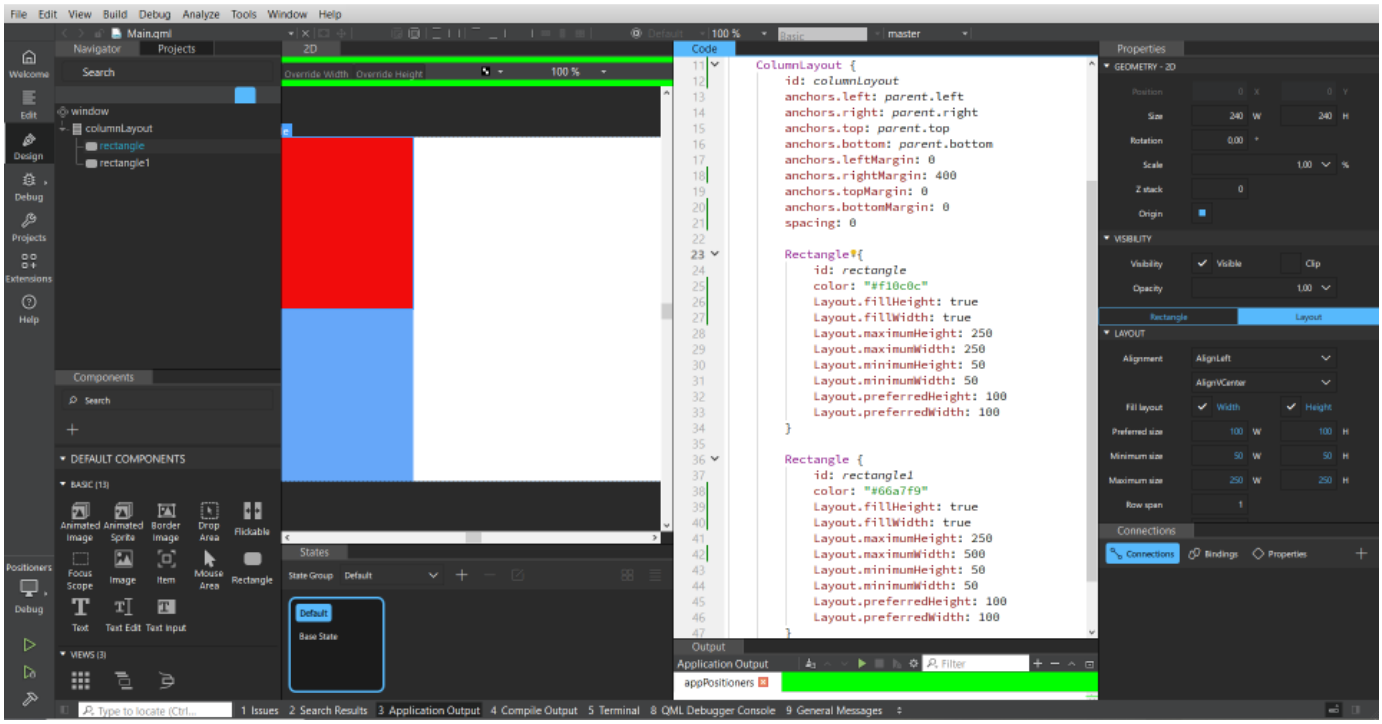
En los layouts podemos definir distintos tamaños para nuestro elemento.

- **Prefered width/height**. Es lo que ocupará el elemento por defecto.

- **Minimum width/height.** Si nuestra fila se hace más pequeña (por ejemplo, si se hace más pequeña la pantalla), nuestro elemento rectángulo decrecerá, pero hasta el mínimo especificado.
- **Maximum width/height.** Si nuestra fila se hace más grande, nuestro elemento rectángulo crecerá, pero hasta el máximo especificado.

ColumnLayout

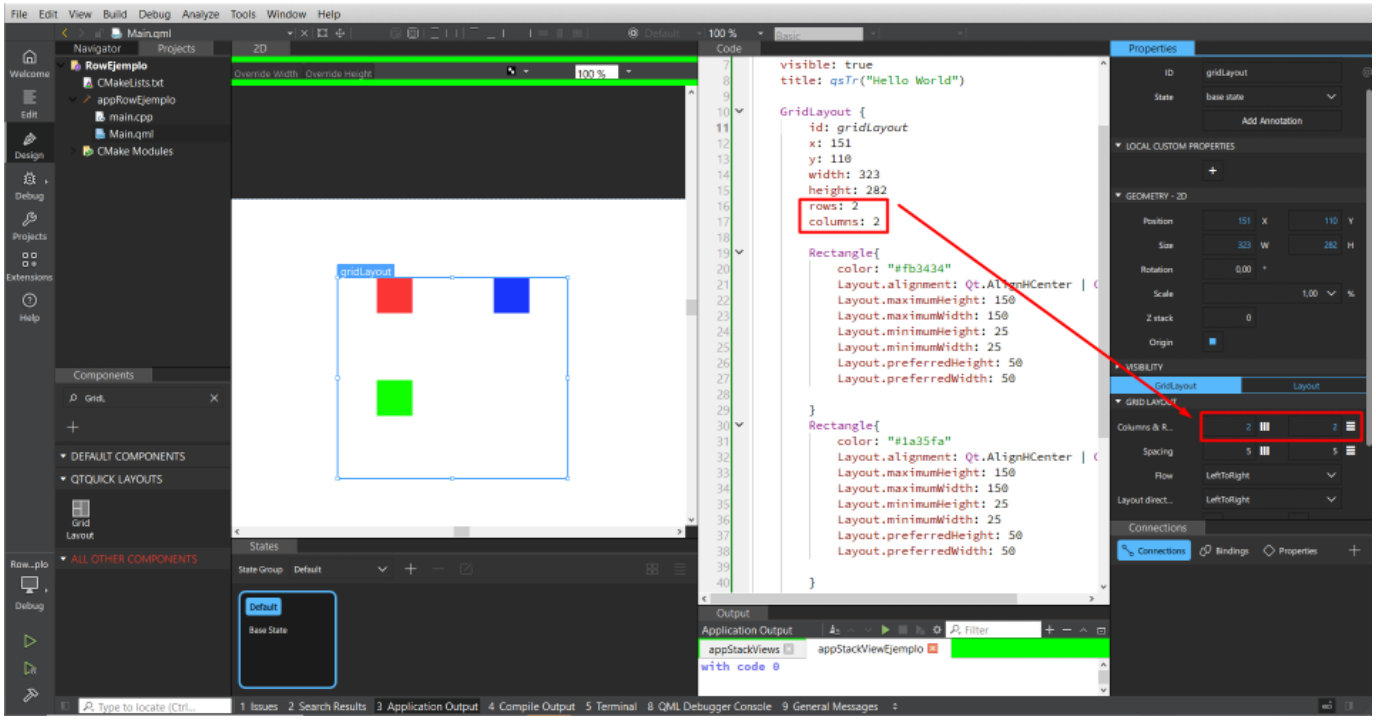
Para el componente ColumnLayout será igual que RowLayout pero con una disposición horizontal.

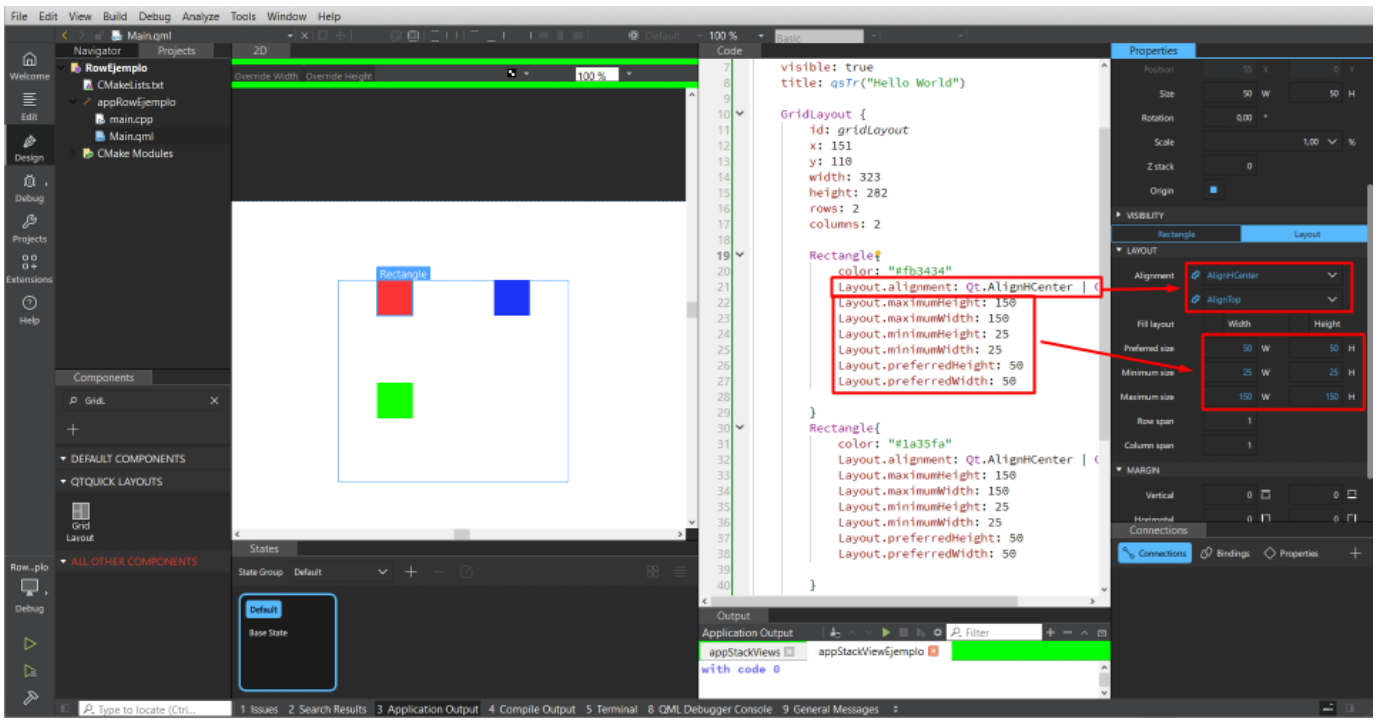


GridLayout

Con GridLayout se dispondrán sus elementos hijos en forma de rejilla. Podremos definir distintos tamaños para sus elementos y que cambien según el tamaño del contenedor.

Además, mediante la propiedad *columns* y *rows*, definimos cuantos elementos queremos por columna/fila.





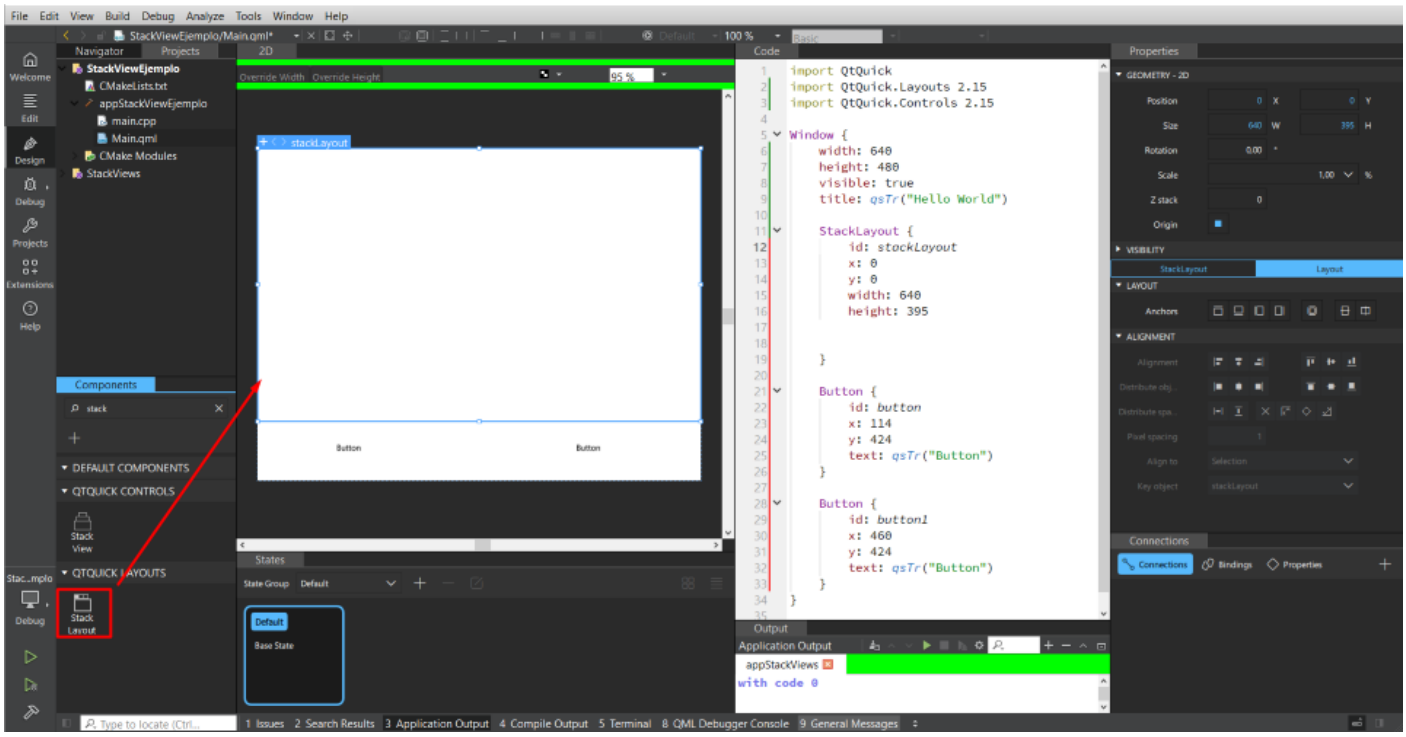
**En el código, después del segundo rectángulo hay un tercero

StackLayout

El componente StackLayout proporciona una pila de Items donde sólo uno de ellos se mostrará a la vez.

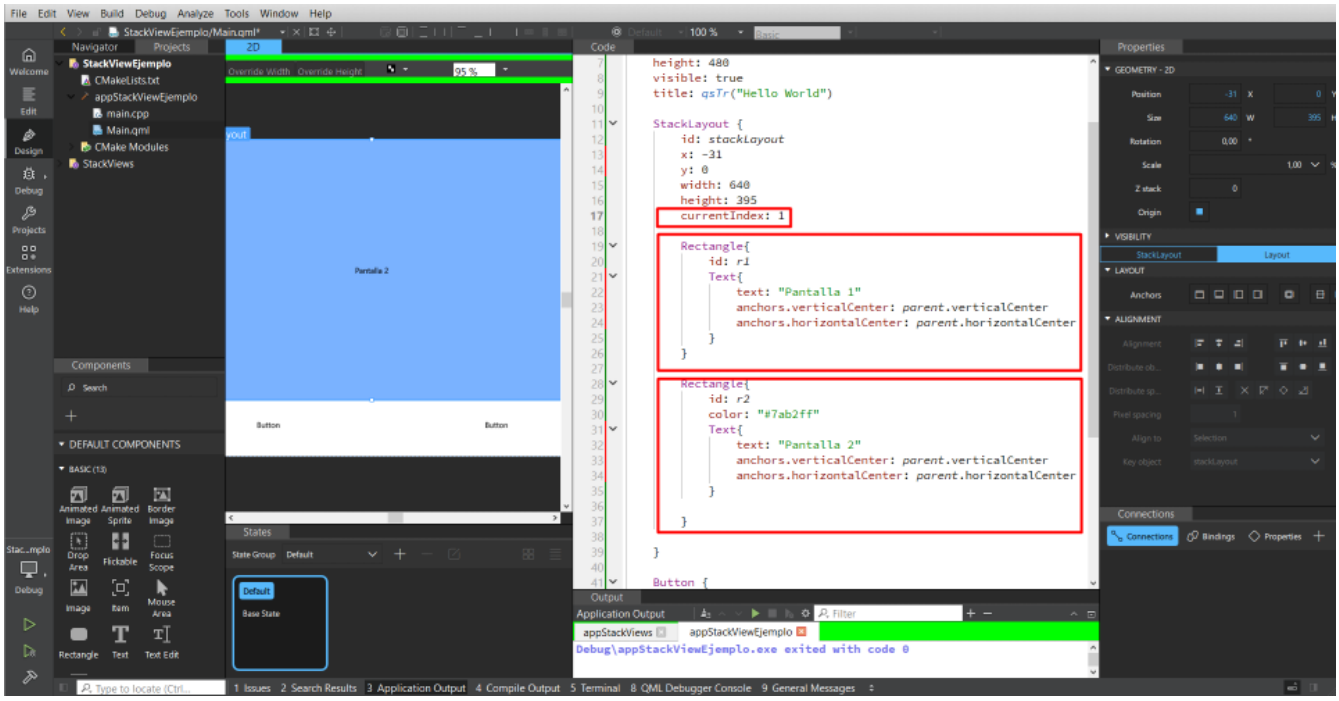
El espacio que ocupe el StackLayout será donde se posicione la vista que seleccionemos en el momento.

A parte de añadir el StackLayout a nuestra pantalla, añadiremos dos botones con los que gestionaremos la vista que se muestre.



Añadimos dos componentes de tipo rectángulo que actuarán como nuestras vistas. Mediante la propiedad *currentIndex* se gestiona la vista que se muestra.

En el caso de la imagen, StackLayout tiene la propiedad currentIndex como 1, por lo que la vista que saldrá por defecto será el segundo rectángulo.



Para añadir la navegación, haremos que los botones cambien el *currentIndex* de nuestro StackLayout