Instalación y uso de Android Studio

Ernestas Jurkus

1ºDAM

Contenidos

[1. Instalación 3](#_Toc26878175)

[2. Introducción y personalización 9](#_Toc26878176)

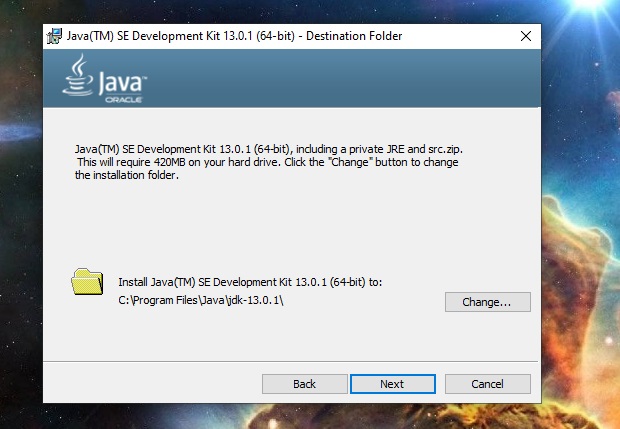
[3. Plug-ins 20](#_Toc26878177)

[4. Proyecto básico 23](#_Toc26878178)

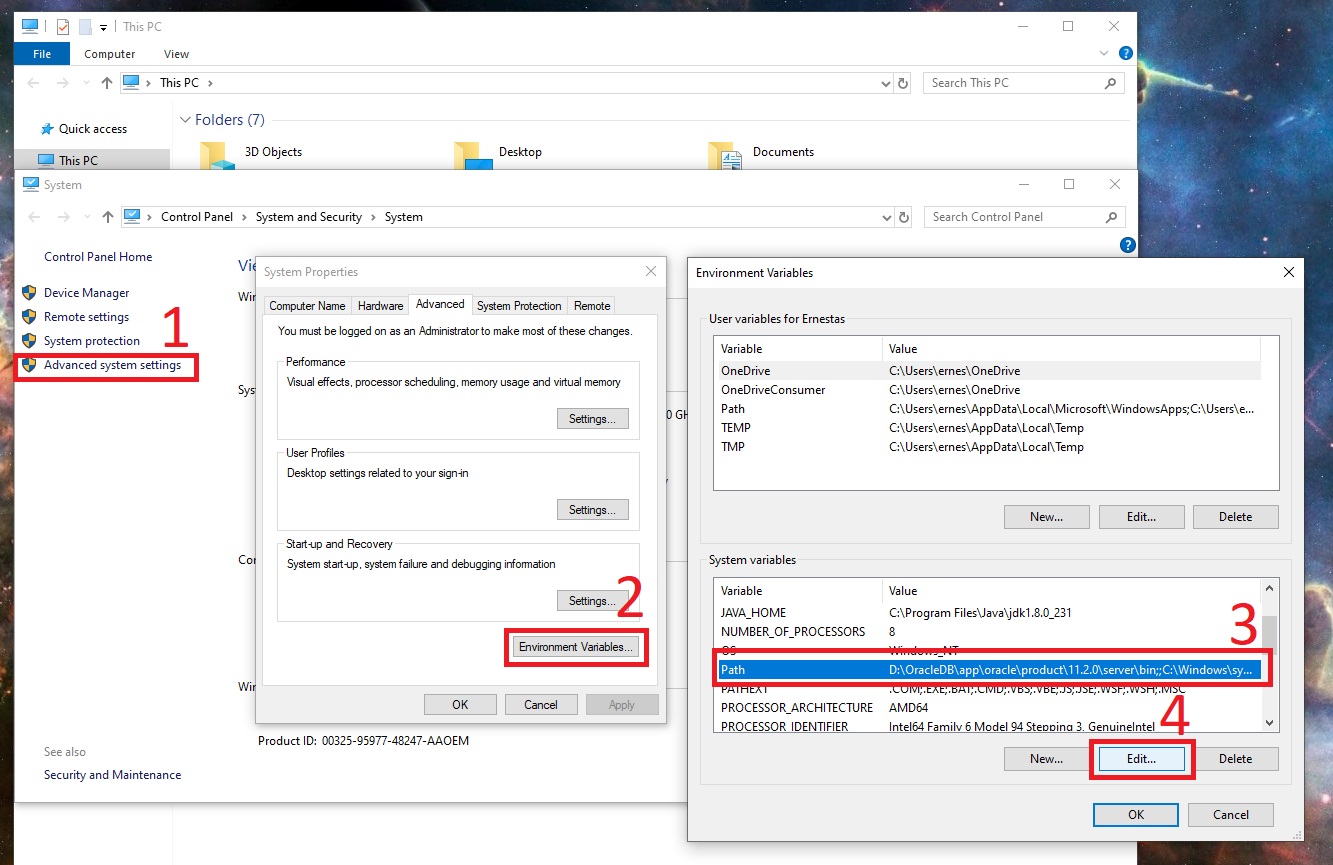
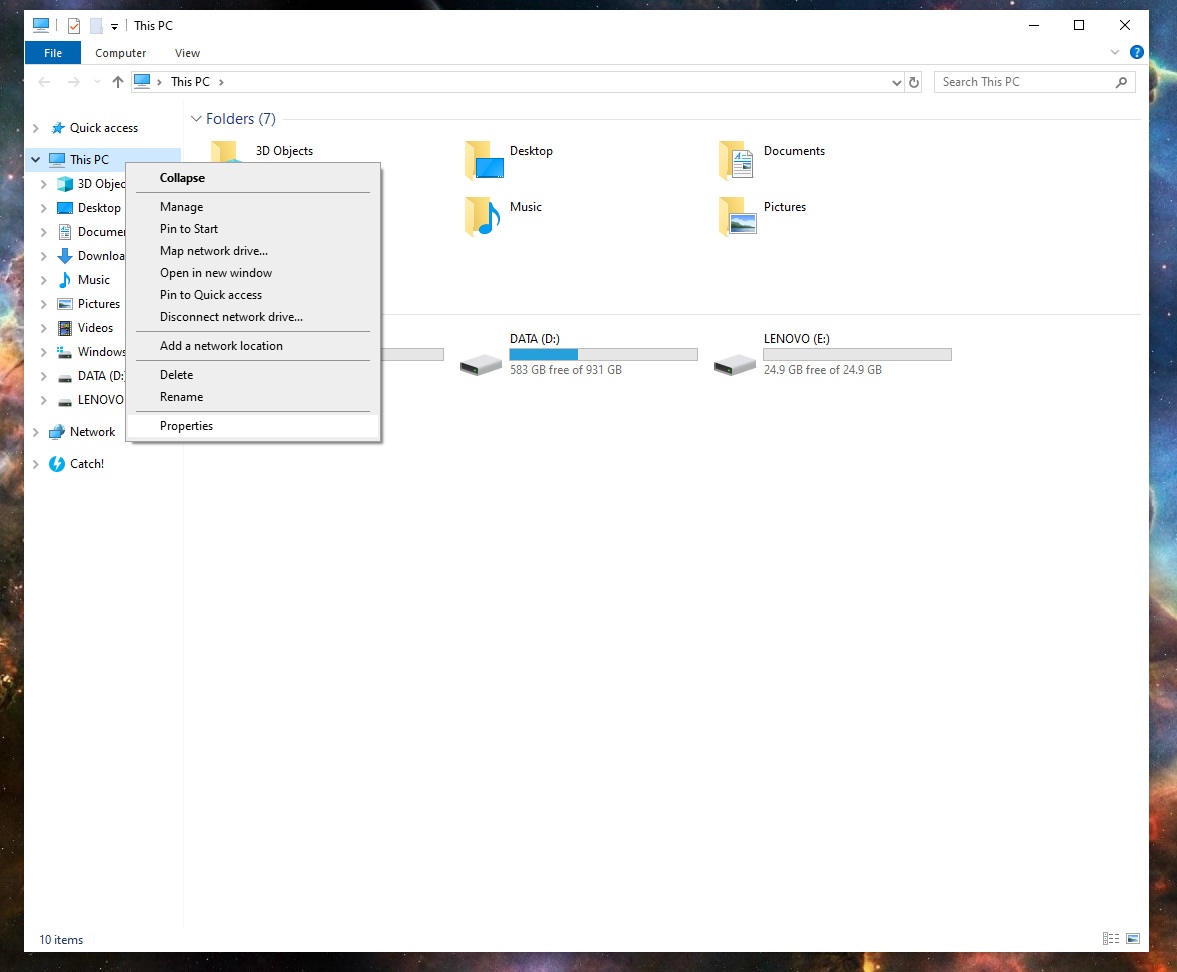
## Instalación

Primero de todo, debemos instalar el Kit de Desarrollo de Java JDK para poder desarrollar las aplicaciones de Java.

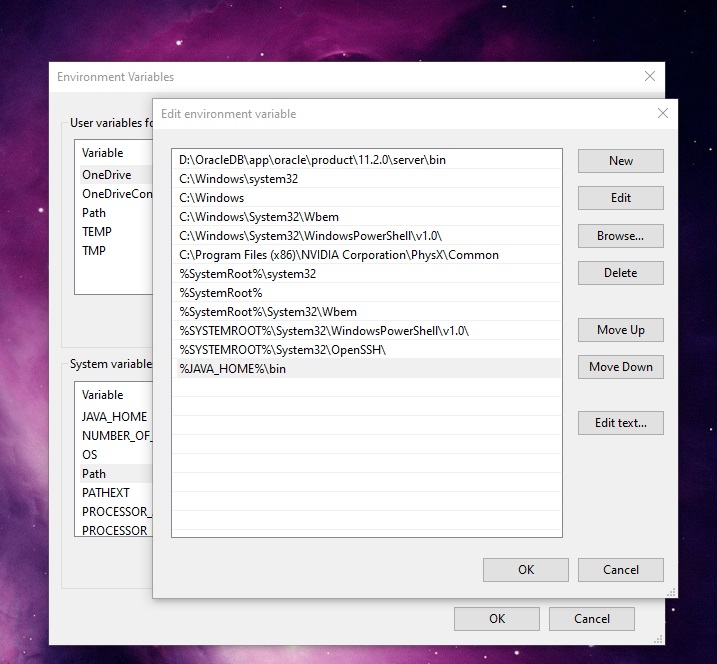
<https://www.oracle.com/technetwork/java/javase/downloads/index.html>



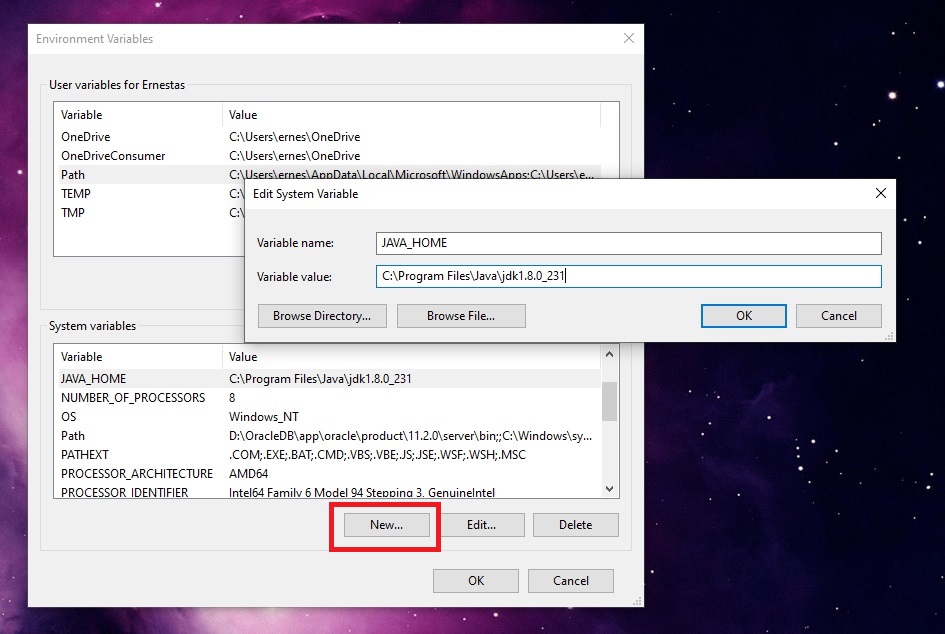
En la instalación, es importante mirar donde se instala el JDK, ya que necesitaremos el directorio para modificar unas variables de entorno:



Con el cursor sobre Path, le damos a Editar y añadimos ‘%JAVA\_HOME%\bin’, como se muestra en la imagen:



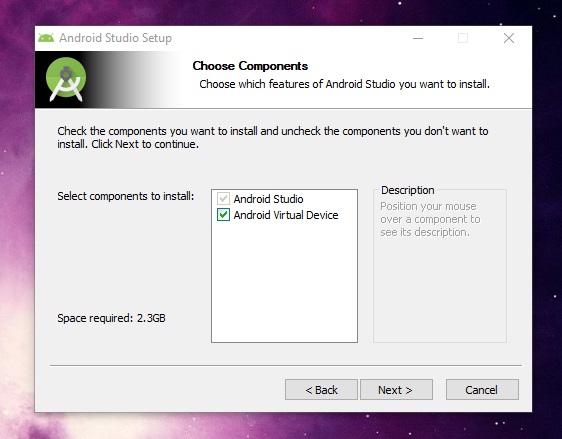
Y también creamos una variable llamada JAVA\_HOME con el directorio en el que hemos instalado el JDK:



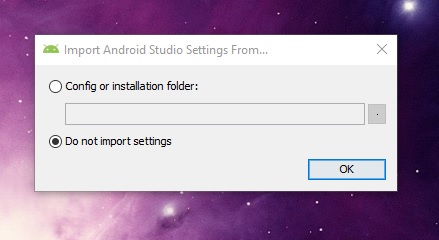
Una vez finalizada la instalación y la configuración, podemos proceder a la instalación de Android Studio en sí.

<https://developer.android.com/studio>

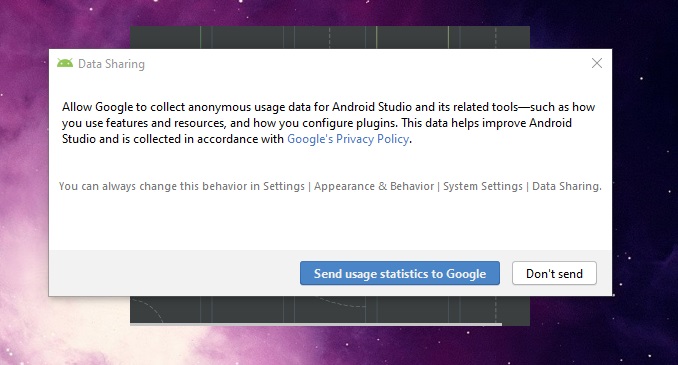
Al iniciar la instalación, elegiremos los componentes a instalar. Aunque el Android Virtual Device es opcional, es muy recomendable para la prueba del software a desarrollar, ya que nos permitirá emular dispositivos de prueba para el mismo.



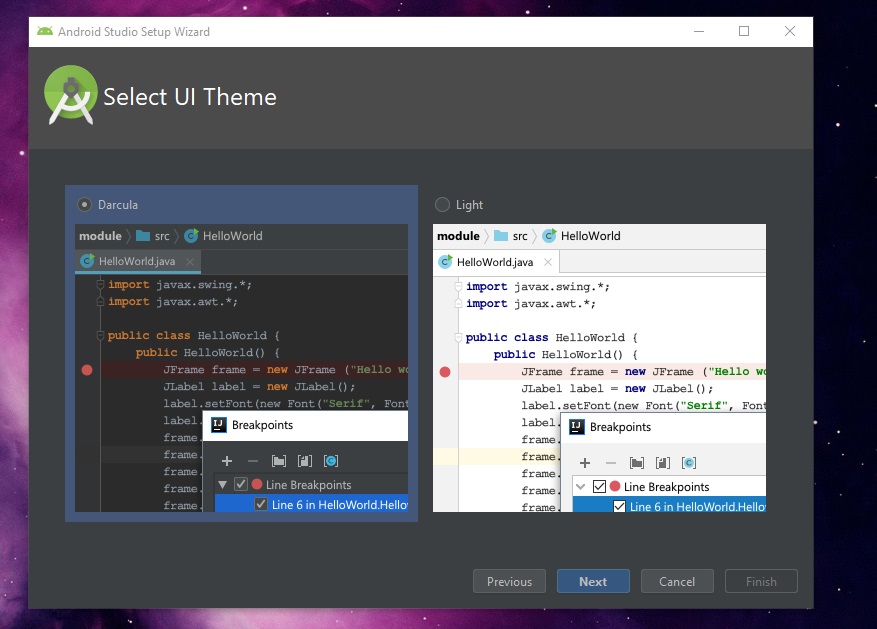
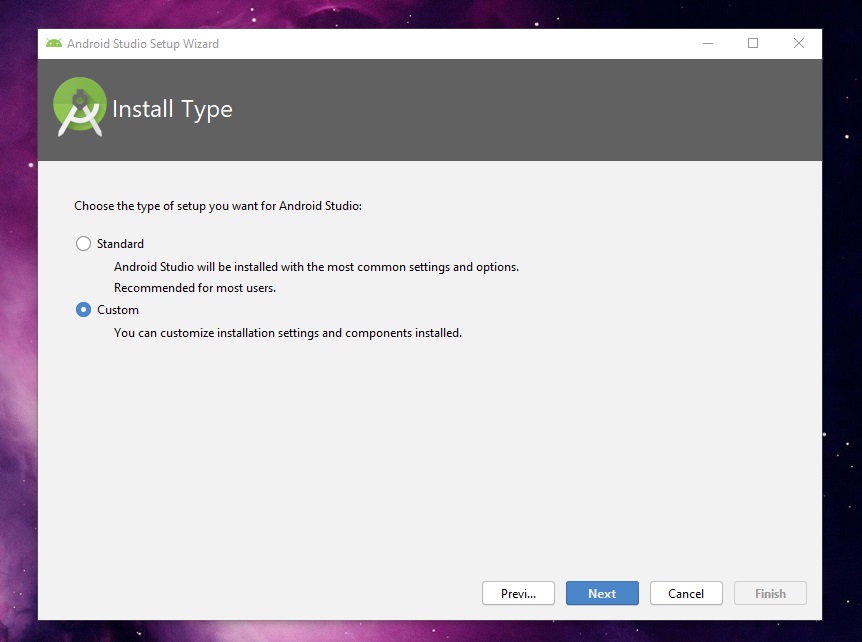
Al terminar de instalarse, marcaremos la opción de ejecutar Android Studio y le daremos a Finalizar para iniciar el entorno.



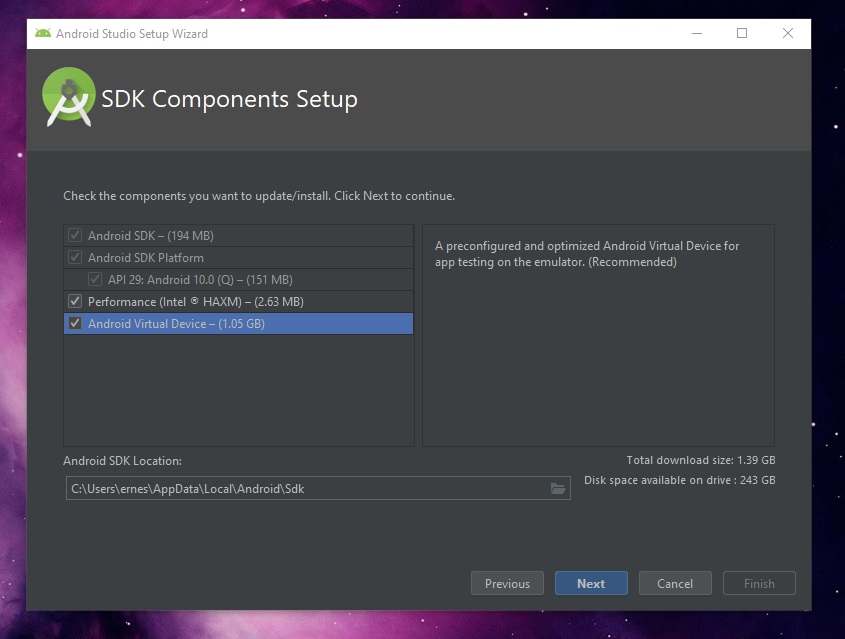
La primera vez, se nos preguntará si queremos importar configuraciones, pero si no hemos desarrollado en Android Studio anteriormente, probablemente esto no nos interese.



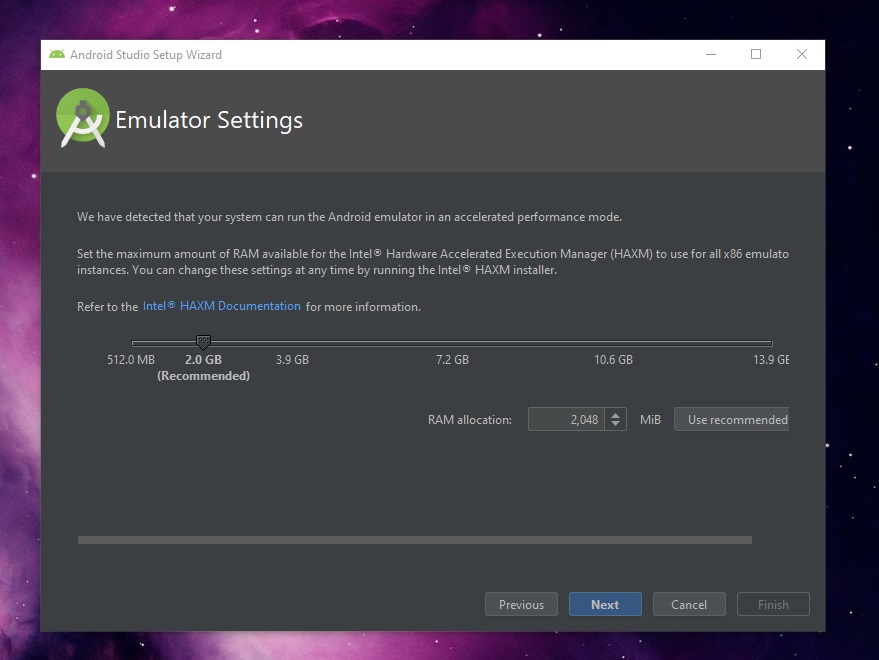
También se preguntará sobre nuestra preferencia a la hora de enviar datos de uso y configuración de nuestros plugins, lo cual es personal y no afectará al uso del entorno. Así, continuando con la instalación, podemos elegir entre una configuración estándar o una personalizada. Exploraremos la opción personalizada:



Desde un principio, Android Studio nos deja elegir entre dos temas comunes para los colores ‘Darcula’ (tema de noche) o ‘Light’ (tema estándar).

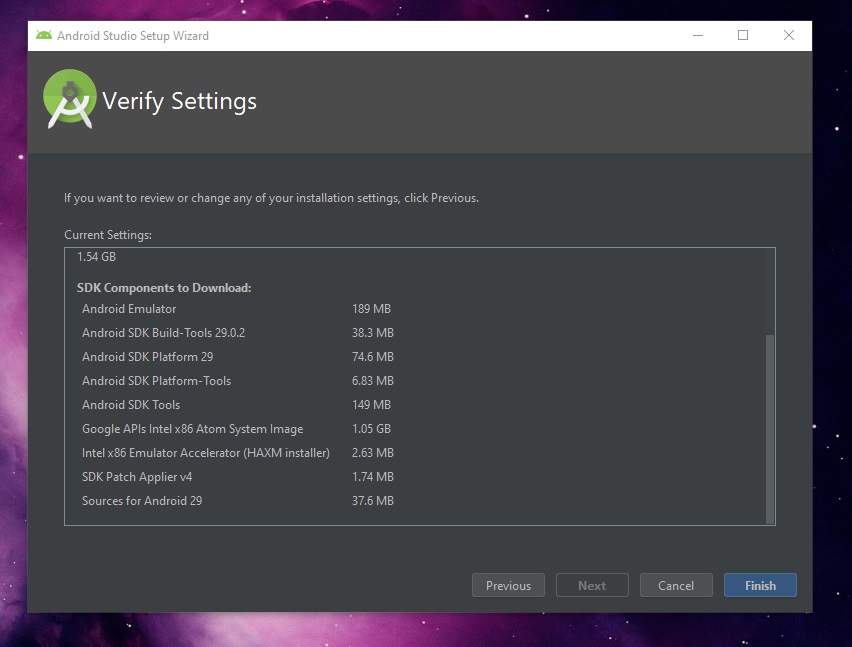


También nos puede aparecer una lista de elementos a instalar o actualizar del Software Development Kit (como se ve, se vuelve a hacer mención a Android Virtual Device). Es conveniente marcar todas las casillas y continuar sin modificar el directorio.

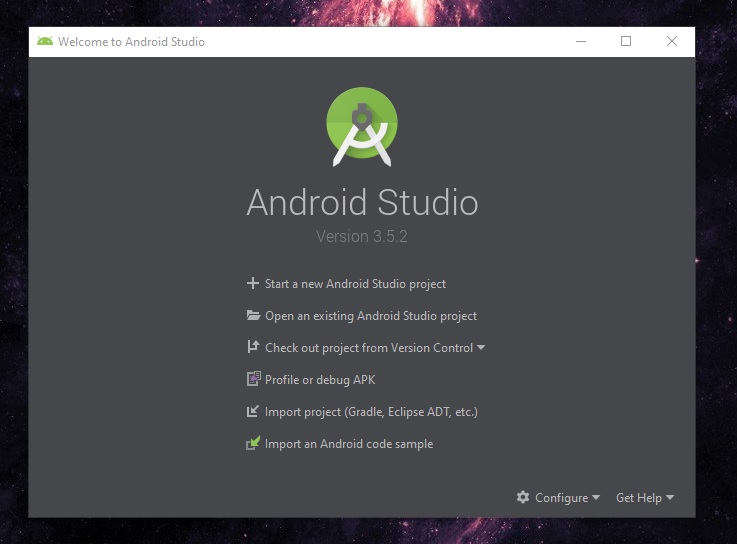


Convenientemente y si está disponible, el instalador detectará que nuestro procesador puede emular en un modo de máximo rendimiento las máquinas virtuales de Android que creemos y nos permite establecer un máximo de RAM para esta función. Por conveniencia y para el primer uso, lo dejaremos en recomendado aunque, como se indica en el instalador, es posible modificar posteriormente esta configuración.

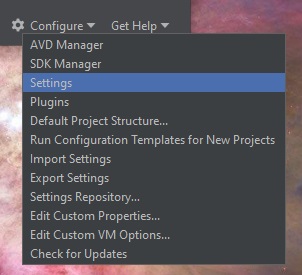
Solo quedará una pantalla en la que confirmamos nuestra configuración y le damos a finalizar.



## Introducción y personalización



Por defecto, al iniciar Android Studio (o si se está siguiendo la instalación), aparecerá este diálogo, que nos permitirá abrir un proyecto desde múltiples fuentes o crear uno nuevo. En la esquina inferior derecha nos aparece así también, la opción de configuración y la de ayuda.

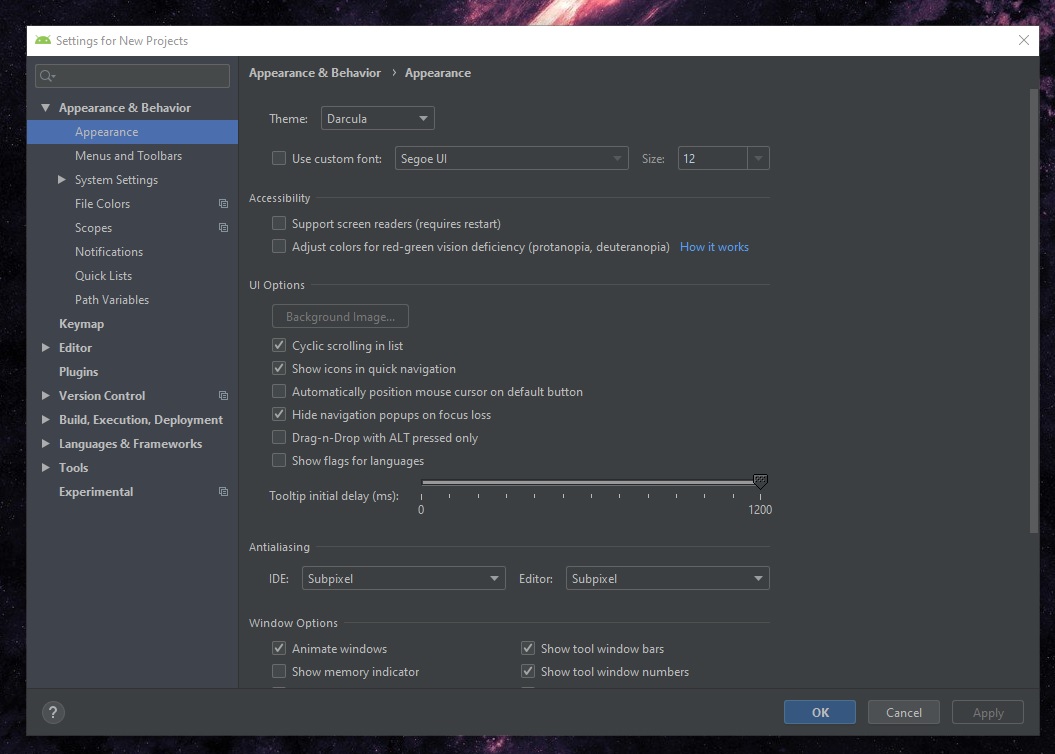


Desplegando el menú de configuración, podemos ver numerosas opciones: La administración de nuestras máquinas virtuales de Android, la administración de las API que usamos, los plugins, configuración para formatos de los nuevos proyecto, exportación e importación de configuraciones y la búsqueda de actualizaciones para nuestro entorno.

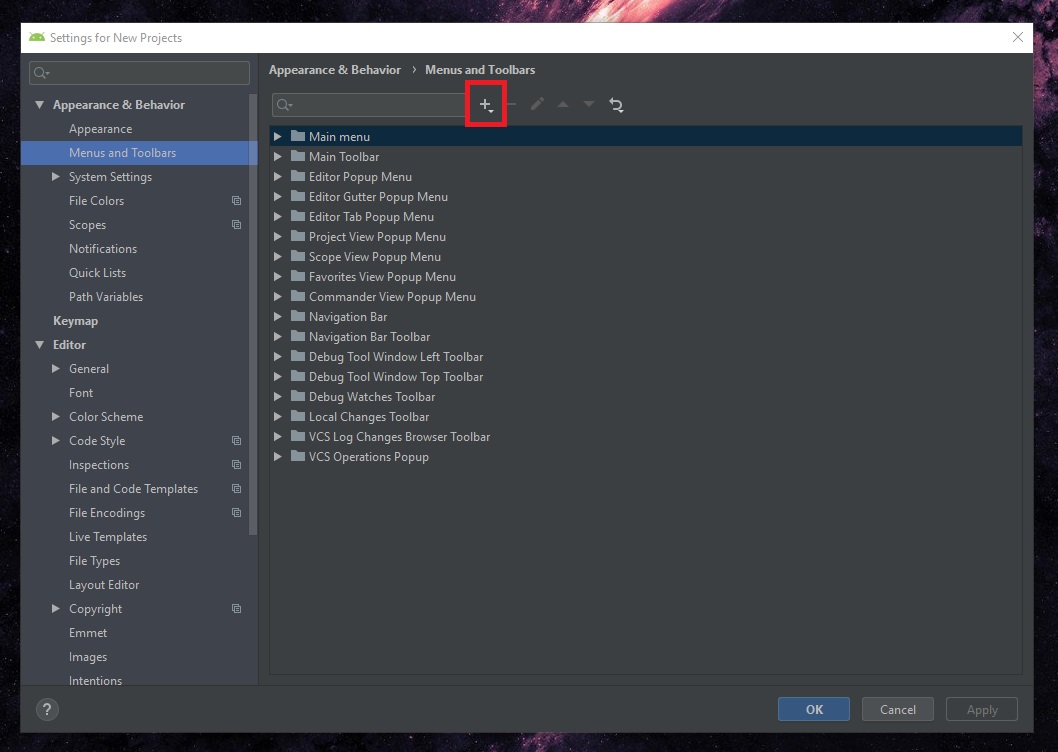
Por ahora y para presentar unas cuantas configuraciones básicas, nos centraremos en la tercera opción...



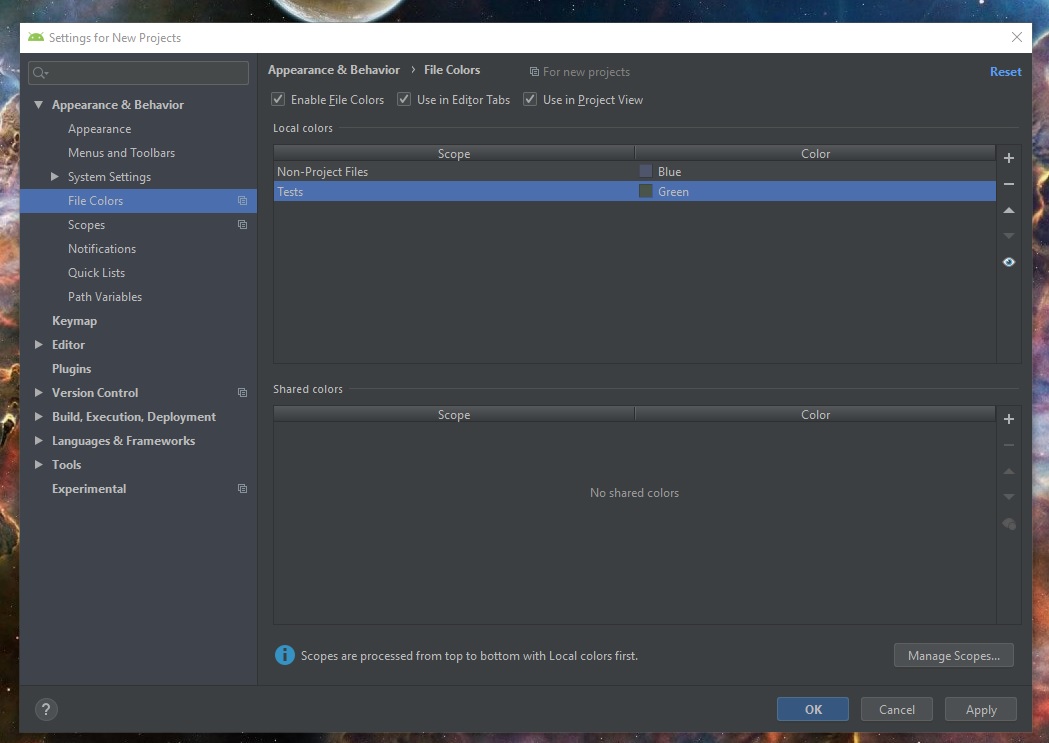
Aquí desplegamos un gran menú de configuraciones, que incluye algunas a las que nos referíamos antes. Para un primer uso, revisaremos las configuraciones más básicas de apariencia, accesibilidad y atajos de código.



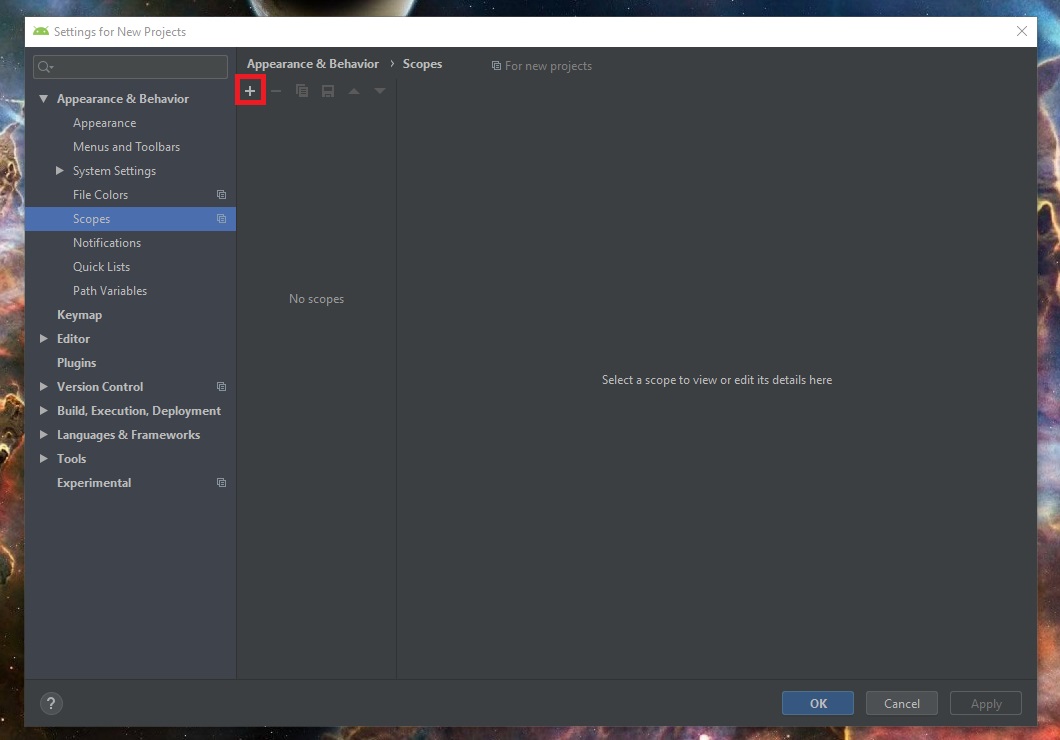
Al desplegar la primera pestaña, tenemos la opción de cambiar de nuevo el tema (entre los instalados) y la fuente, un par de opciones de accesibilidad y la apariencia de todo el entorno y de las ventanas que abrimos en él. Muchas de estas opciones son simplemente estéticas y de preferencia personal, pero, por ejemplo, algunas de las opciones de ventanas como las animaciones y el desplazamiento suave puede ser adecuado desactivarlas en equipos lentos.



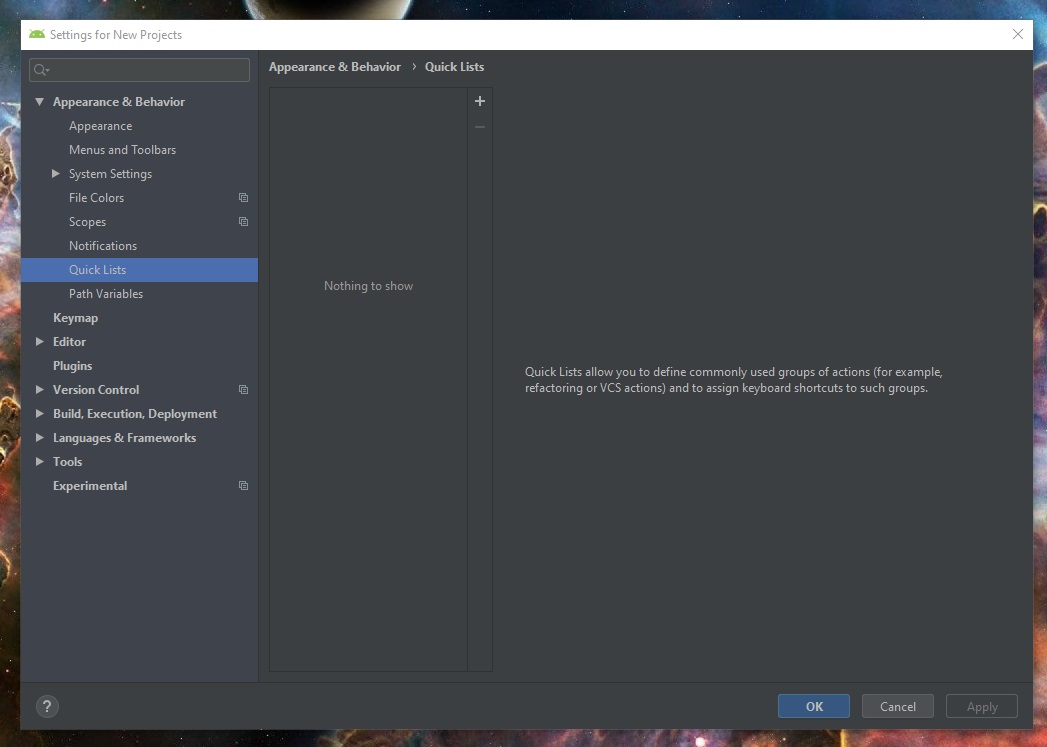
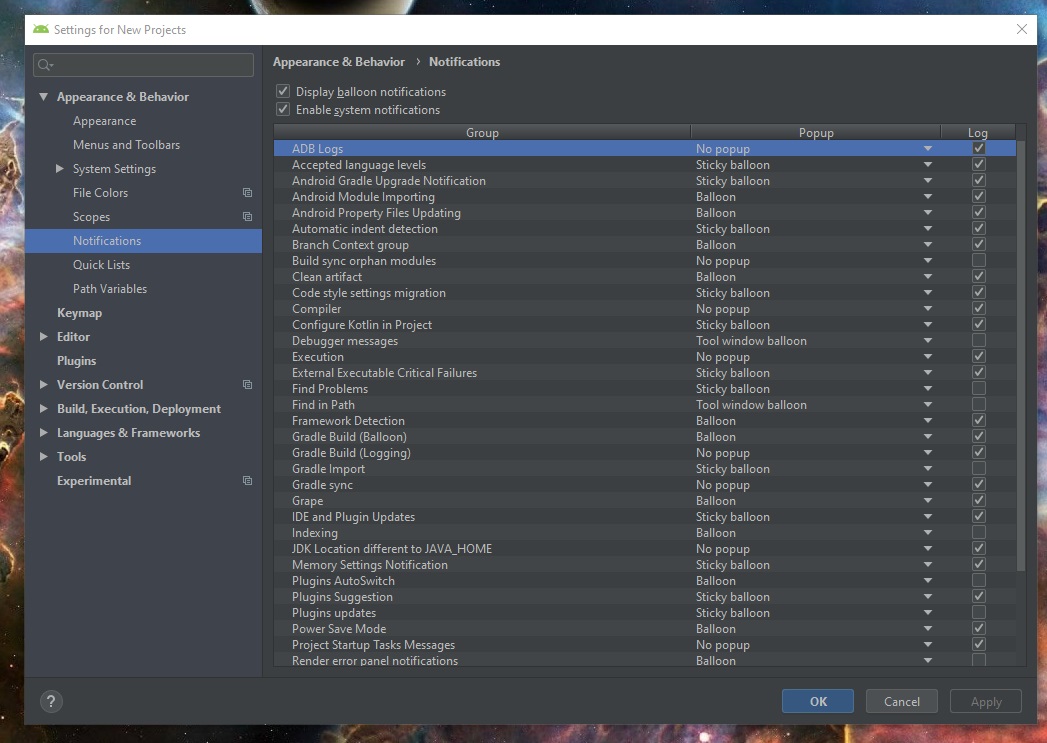
Debajo de apariencias, tenemos la configuración del menú. El botón en rojo nos irá permitiendo añadir acciones y separadores para personalizar el menú, que desplegamos y modificamos en la misma ventana.



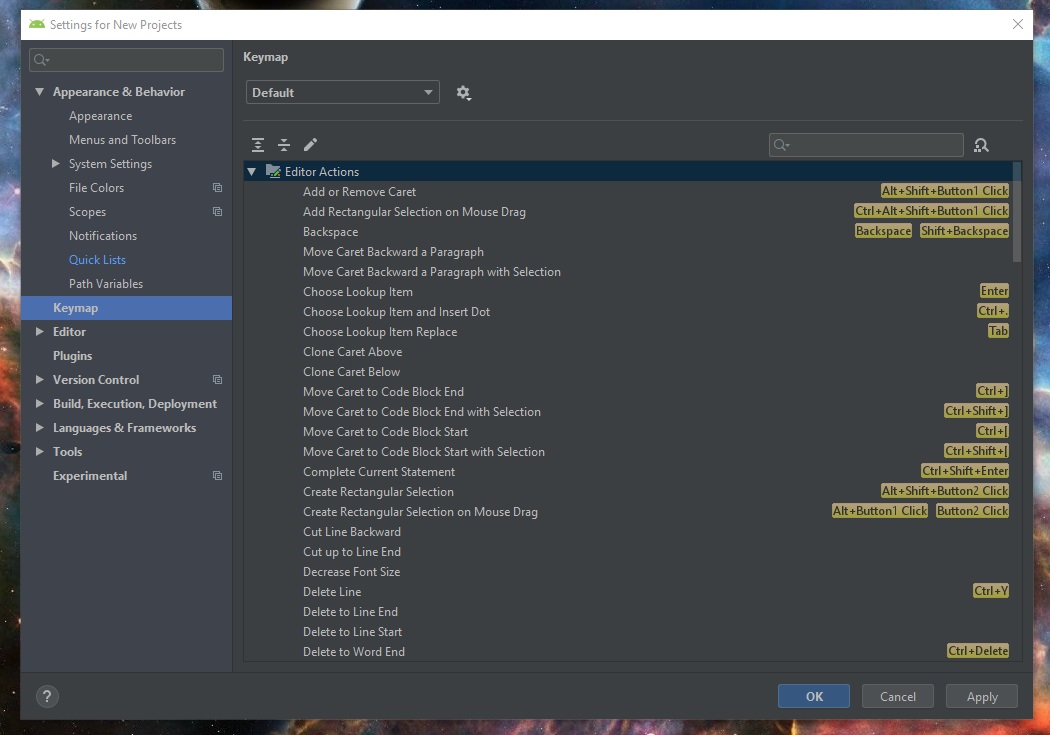
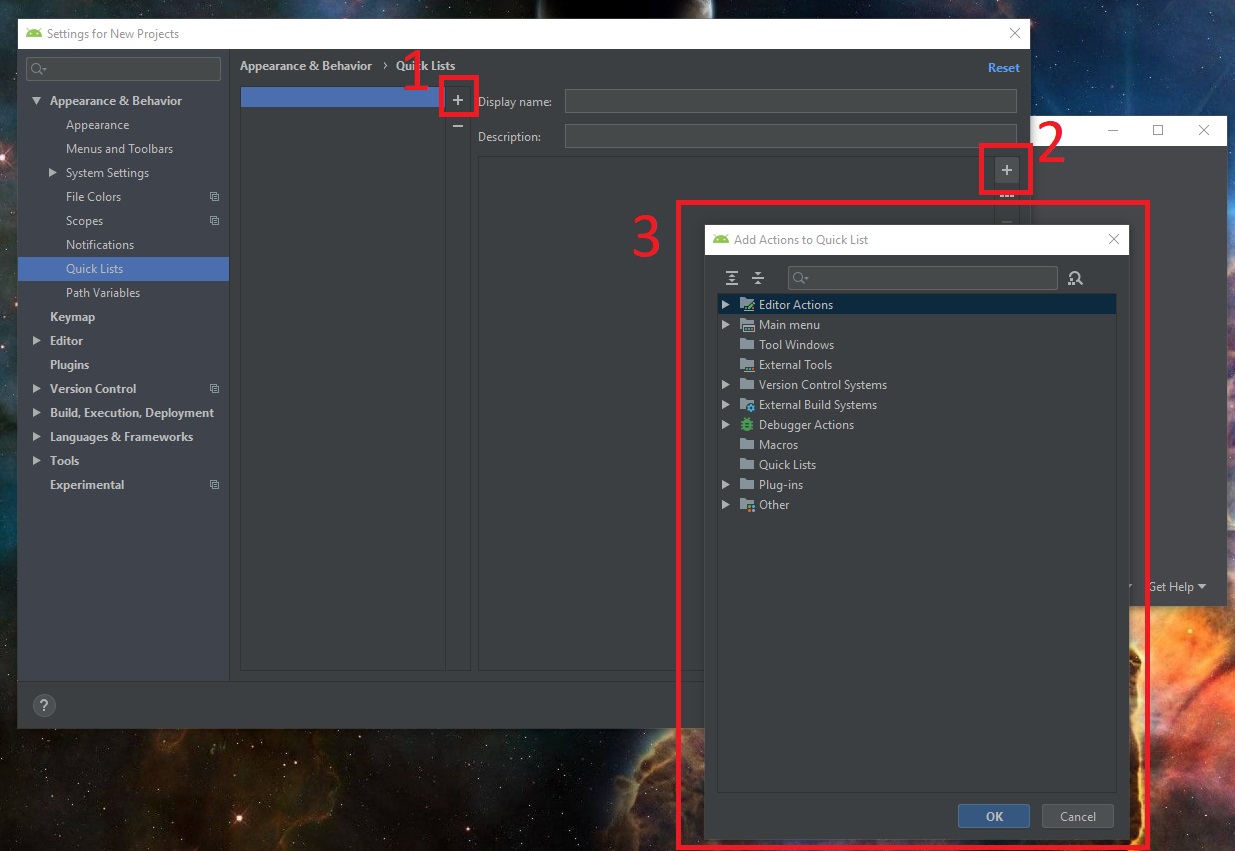
Debajo de configuración de sistemas, podemos modificar los colores en los que aparecerán distintos tipos de archivos a los que hagamos referencia en los códigos fuente, permitiendo las opciones siguientes a estas modificar el “rango” o los tipos de archivos sobre los que aplicamos estos colores.



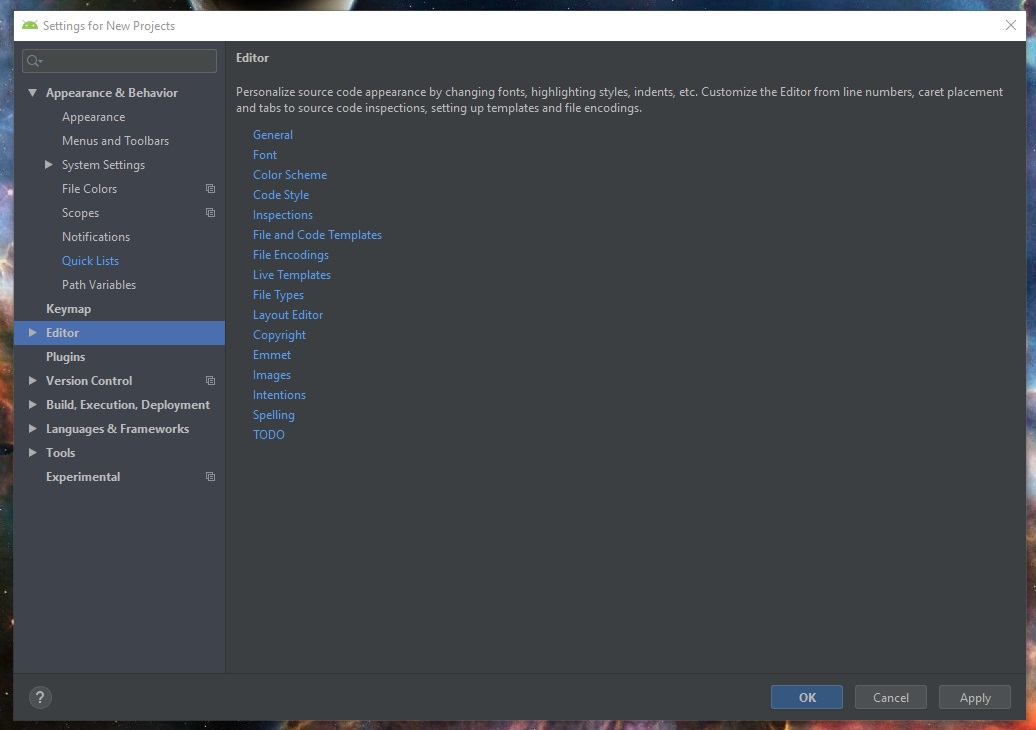
La opción que le sigue nos permitirá configurar las notificaciones que recibimos durante el desarrollo, así como el tipo de notificación que recibiremos ante los eventos:



En Listas Rápidas, tendremos la posibilidad de agrupar varias acciones que usemos de manera recurrente y, como se indica, asignarlas a un atajo en el teclado:

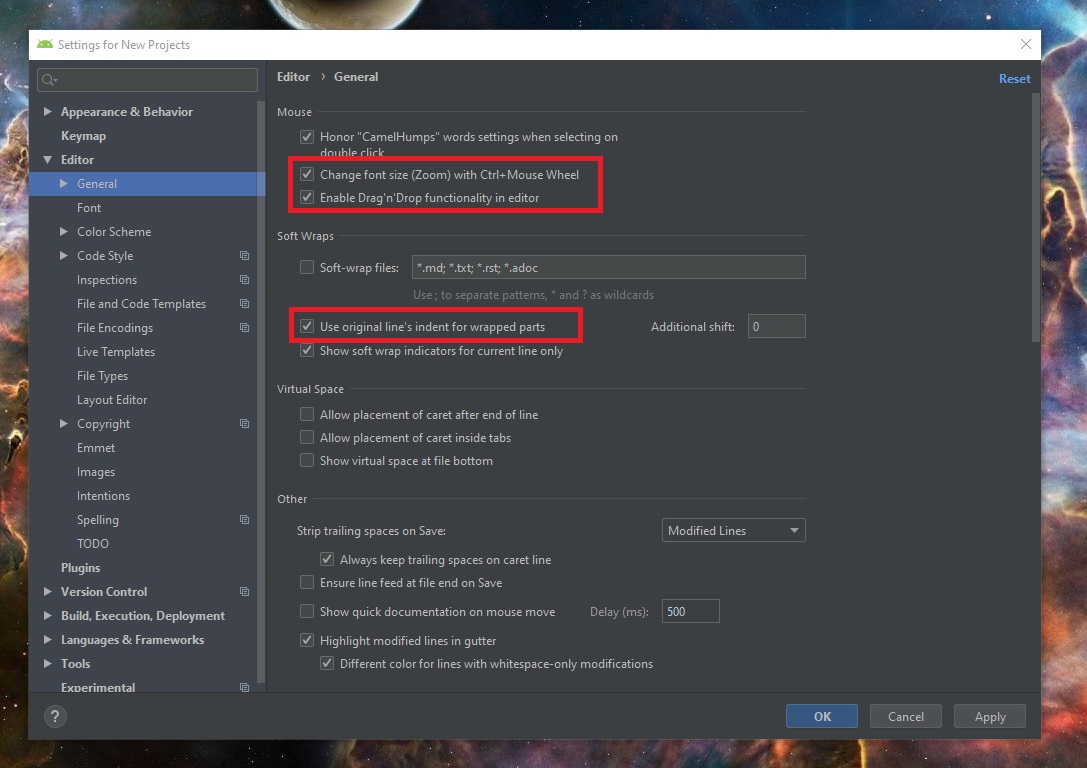


Tras terminar con la apariencia y el comportamiento, nos podemos encontrar la configuración de los atajos de determinadas acciones, que también estaban en la configuración de Listas Rápidas. Y justo después de esto, tenemos la configuración del editor:

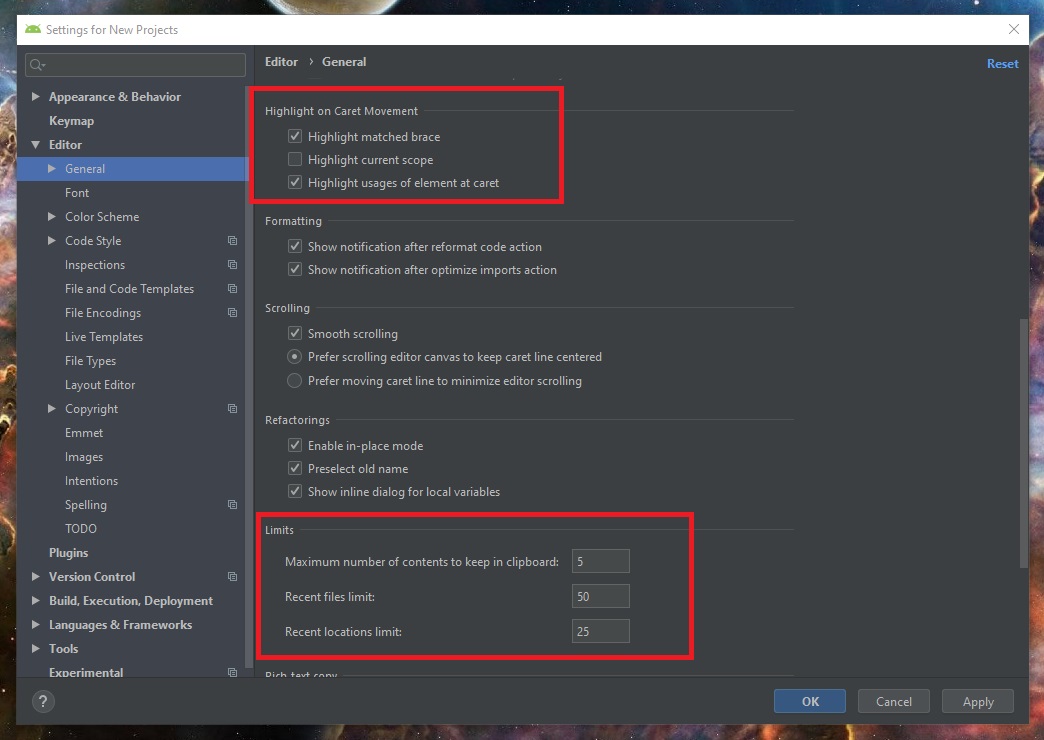


Aquí, de nuevo, nos encontramos con numerosas configuraciones referentes a la edición del código fuente.

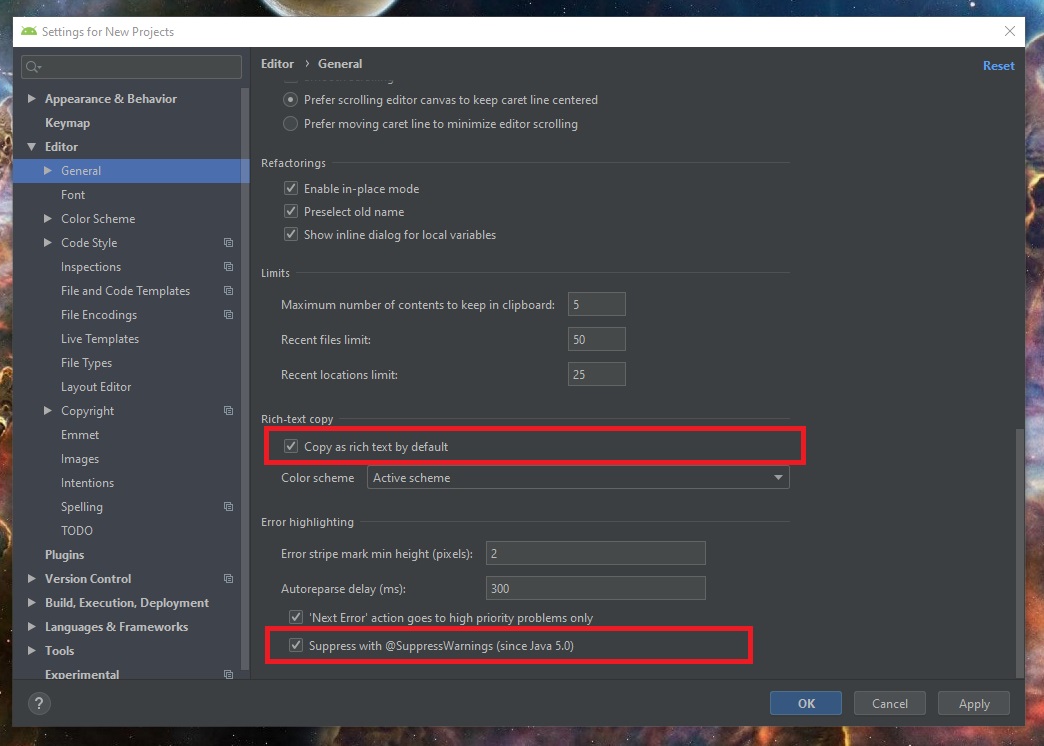
En “general”, encontraremos las opciones más básicas, muchas de ellas bastante interesantes si hemos desarrollado con otro entorno y estamos acostumbrados a ciertas facilidades. En opciones del ratón, dos funciones muy importantes aparecen. La de modificar el zoom con cntl+rueda del ratón (desactivada por defecto) y la opción que en el desarrollo de interfaces nos permitirá arrastrar elementos directamente al editor (esta sí está activada por defecto). La siguiente opción marcada es la de usar la misma tabulación empleada en el archivo desde el que copiamos unas líneas de texto. De nuevo, desactivada por defecto, puede ser interesante para algunos usuarios.



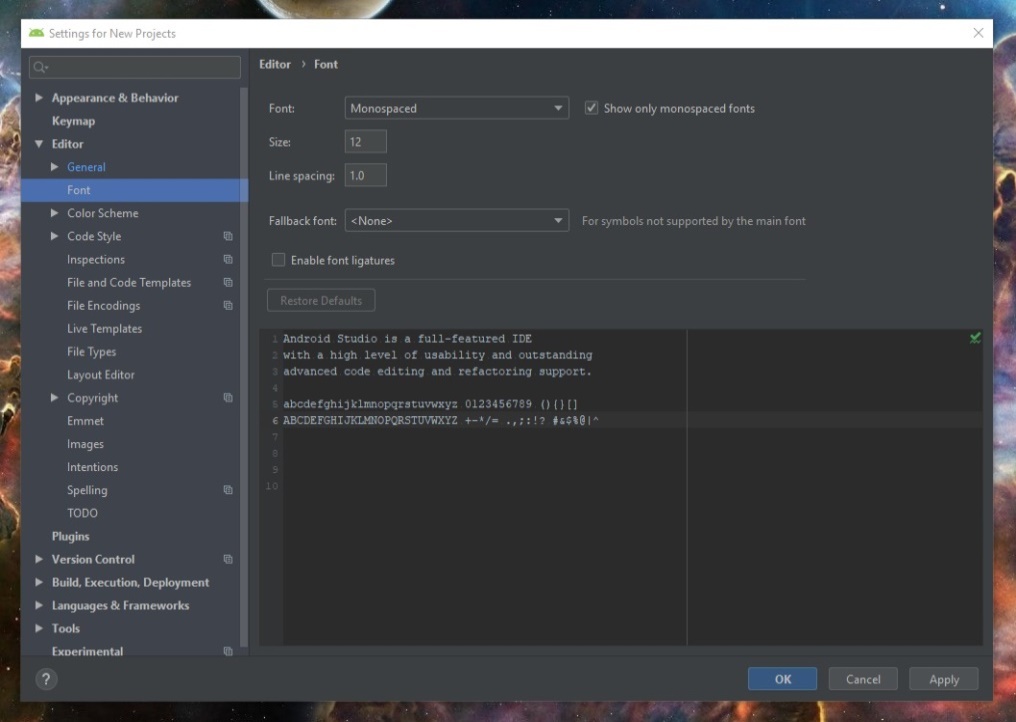
Un poco más abajo, están las opciones de destacar ciertos elementos al tener el cursor sobre ellos, permitiendo, en orden, destacar los paréntesis de cierre y apertura asociados entre sí, los bloques (normalmente esta opción no se activaría si se escoge la primera) y destacar los elementos del mismo nombre, ya sean clases, instancias, variables…



También muy útiles son las opciones destacadas en la parte inferior, sobre todo el primer número. AS, por defecto, guarda varios elementos en el portapapeles, permitiéndonos llevar tantos trozos de código como indiquemos en la primera opción. Y aunque no tan importante, optimiza la navegación tener acceso a los archivos y localizaciones recientes, cuyo límite por defecto suele ser suficiente, pero de nuevo se puede modificar.

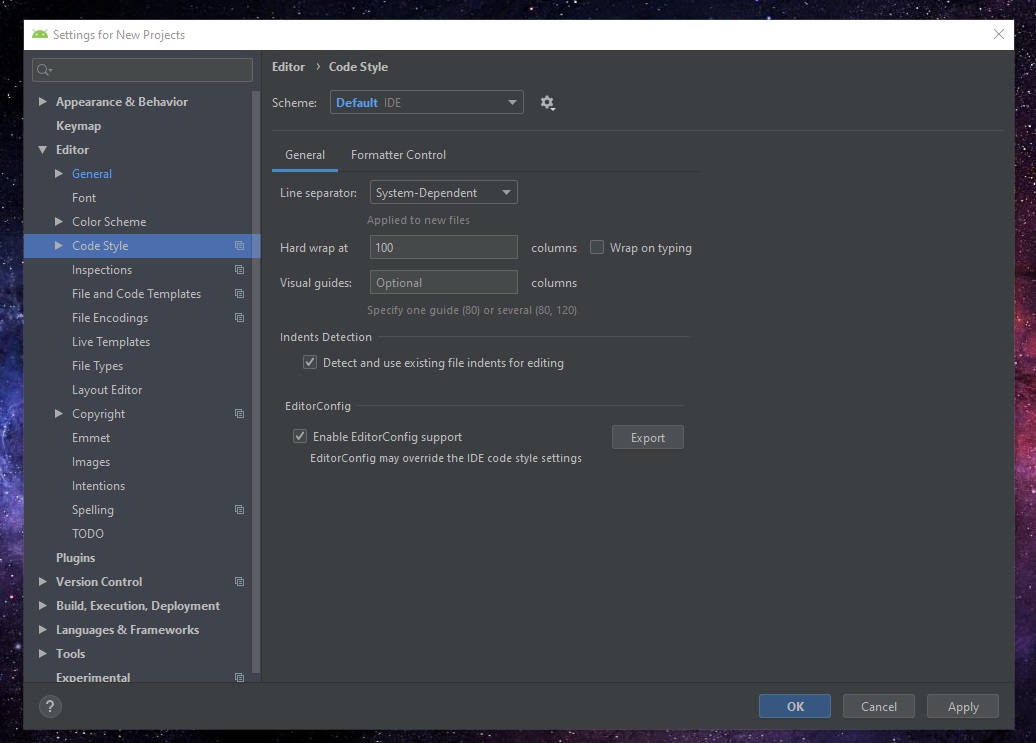


Por último, si seguimos desplazándonos a través de las opciones generales, nos encontramos otras dos opciones muy útiles. Una que copia por defecto el texto del editor como RTF (útil si vemos que trasladando el código de un lugar a otro nos dan problemas ciertos símbolos) y la supresión de avisos mediante la etiqueta indicada (si la versión de Java que usamos es compatible).

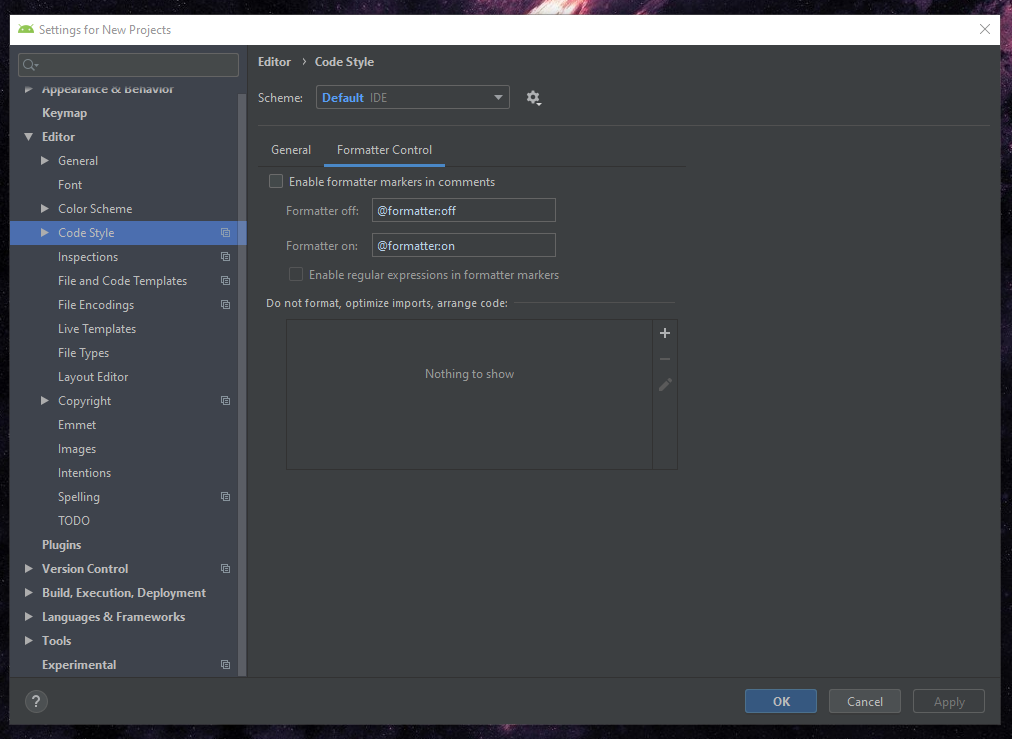


La siguiente opción nos permite modificar tamaño y estilo de la fuente del editor.

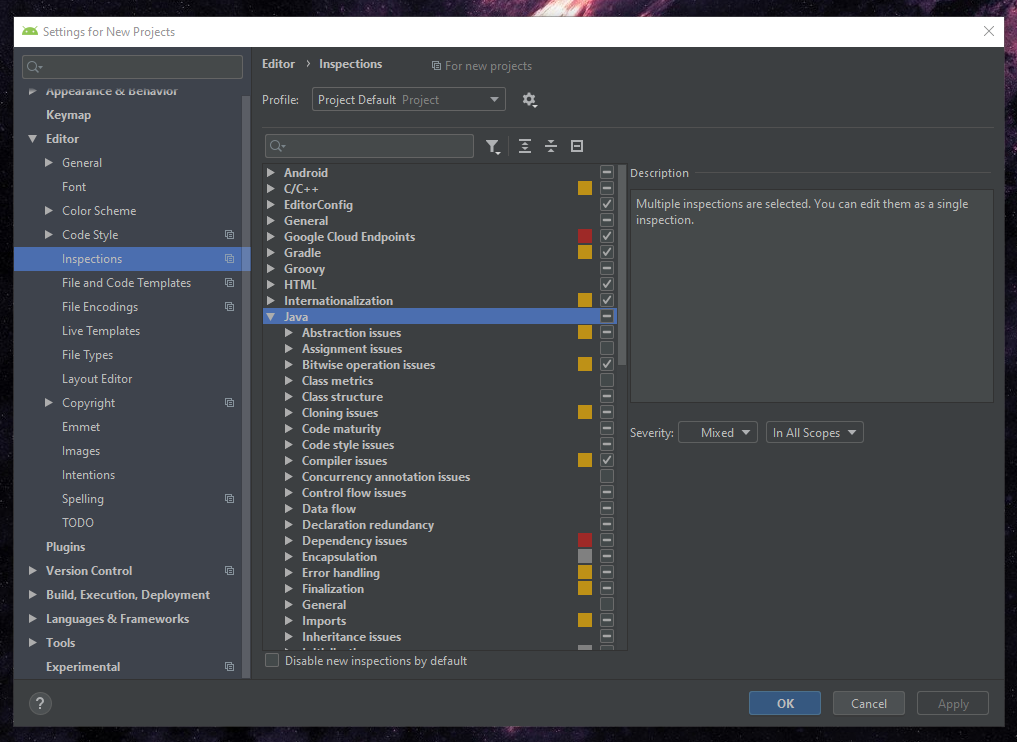
La de a continuación es simplemente la elección del tema de nuevo, así que nos la saltamos y pasamos a estilo de código directamente.



Aquí podemos elegir el separador de líneas, por defecto el que usa el sistema. También el número de columnas a partir del cual el texto pasaría a la siguiente línea (se requiere activar la casilla de al lado para que lo haga mientras escribamos). Esta opción, sin embargo, no funciona para todas las extensiones, entre ellas .java.

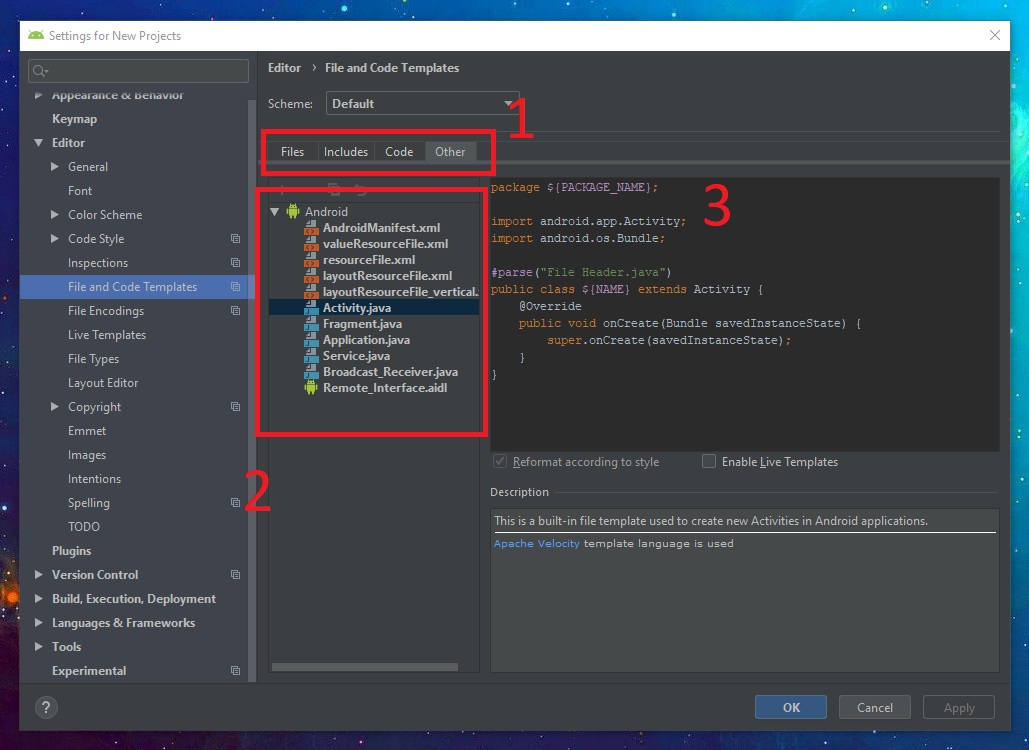


En la misma opción, la segunda pestaña nos permite establecer las etiquetas de formateo automático del código o elementos que directamente no queremos que AS formatee.

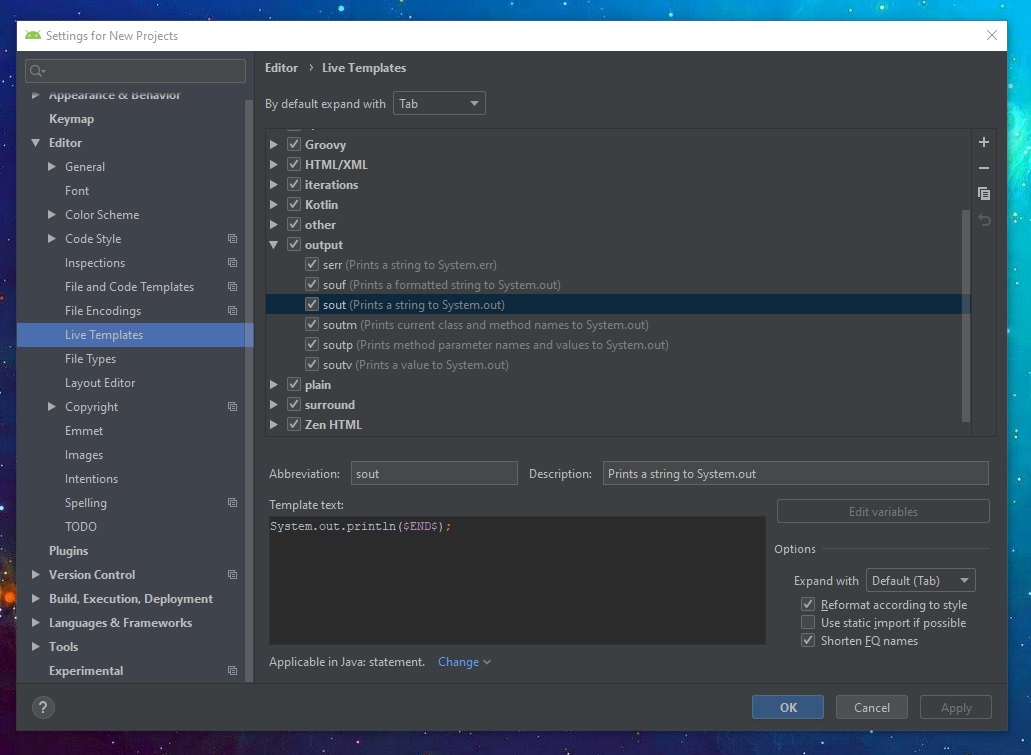


Otra opción es la de inspecciones, que nos permite establecer los errores que el editor destacará de manera automática y cómo de severos los considerará. Con la flecha podemos desplegar los errores referentes a los numerosos lenguajes en los que trabaja Android Studio. Para un primer uso, no lo tocaremos, pero es una opción interesante que conocer.

Para terminar con la sección del Editor y con las configuraciones más básicas, veremos cómo podemos acceder a dos opciones muy importantes: plantillas de archivos y código y atajos de código.



En la modificación de plantillas, elegiremos para qué tipo de archivo o código queremos modificar la plantilla, después el archivo específico (con java por ejemplo están las plantillas para la actividad principal, servicios…) y en el recuadro haremos directamente las modificaciones pertinentes.

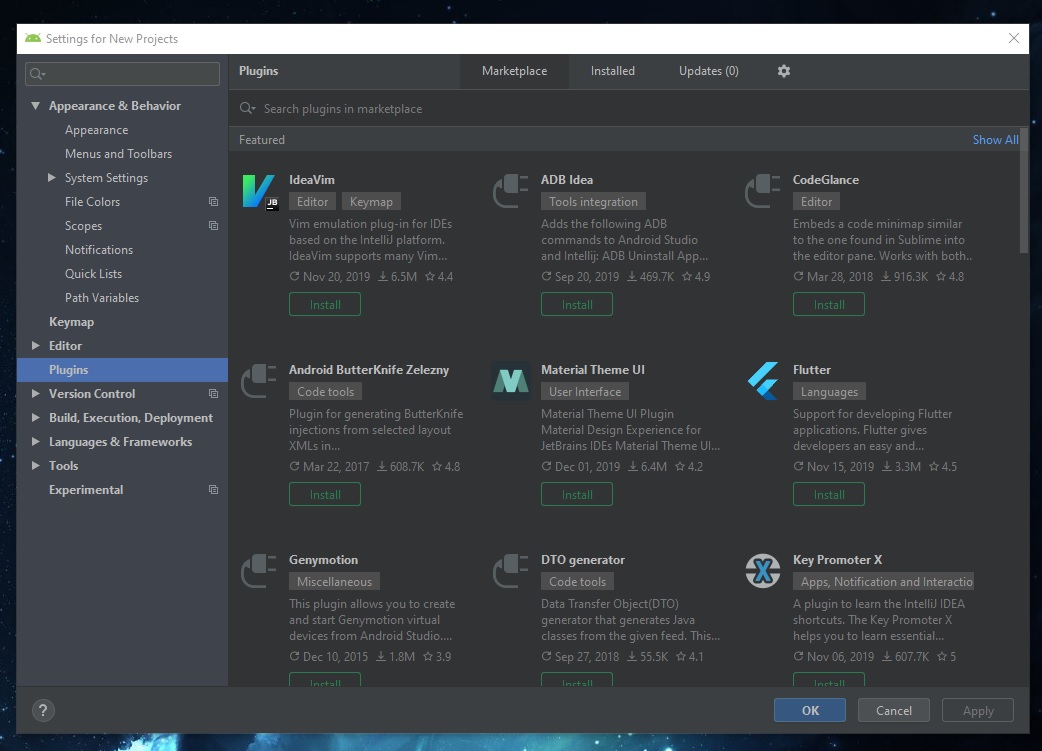


Y en plantillas “en vivo”, tendremos los numerosos atajos de código, en listas desplegables y también directamente modificables, como se puede ver.

## Plug-ins

Una parte importante de los entornos de desarrollo son los plugins, funciones adicionales que le podemos añadir al entorno para facilitar nuestro trabajo o con algún otro fin determinado.

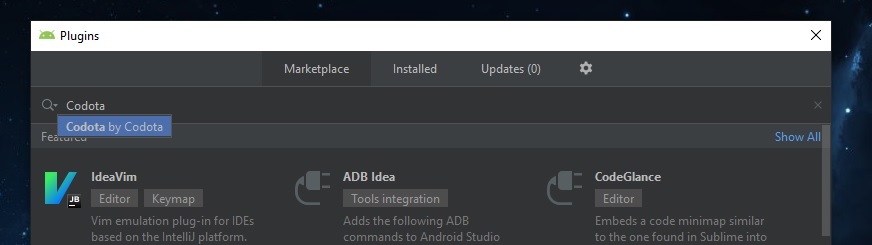
Volvemos a la pantalla de configuración y hacemos clic sobre Plugins en el menú desplegado.



Android Studio nos da una gran facilidad para administrar los plugins instalados. En la imagen ya podemos ver algunos como IdeaVim, Flutter o Key Promoter X, su icono coloreado nos indica que está instalado, con una pequeña etiqueta y una descripción de lo que hacen al lado.

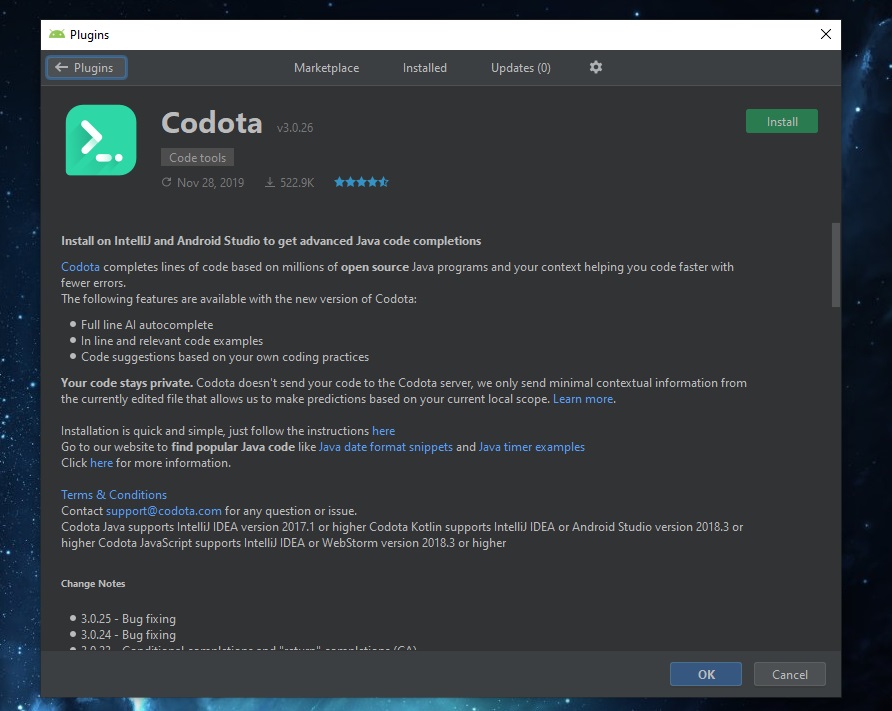
Las pestañas de arriba nos permiten navegar entre todos los plugin disponibles, solo los plugin instalados, y aquellos pendientes de actualización.

Para probar la instalación de plugins, buscaremos uno usando el buscador en la pestaña principal, que nos permite buscar plugins que no están destacados en esta página. En este caso seleccionamos el buscador y buscamos “Codota”.

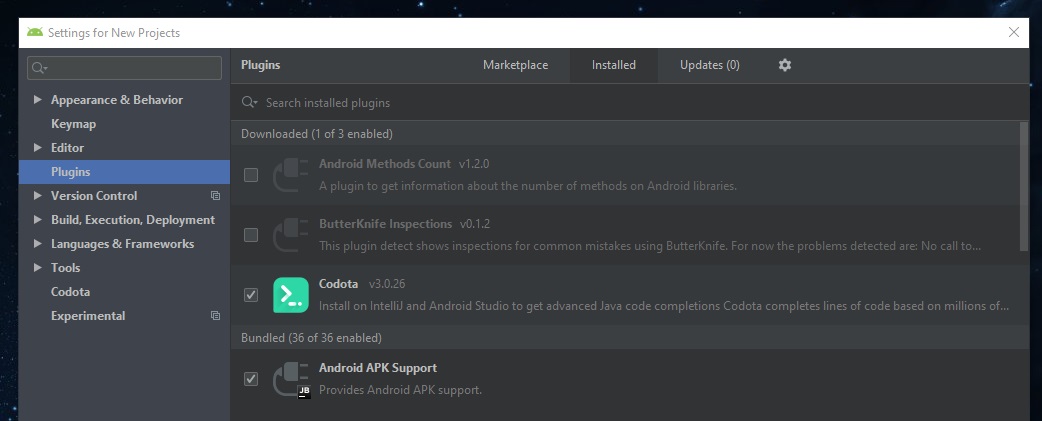


Hacemos clic en la lista y se abrirá la pestaña de instalación.

En la siguiente pantalla, hacemos clic sobre el botón verde de instalación. A veces nos aparecerán avisos de tratamiento de datos de terceros en este paso. Aceptamos si lo consideramos conveniente y se instalará. Tras esto, el botón de instalación cambiará a reinicio del entorno. Clic de nuevo para hacerlo y terminar la instalación.

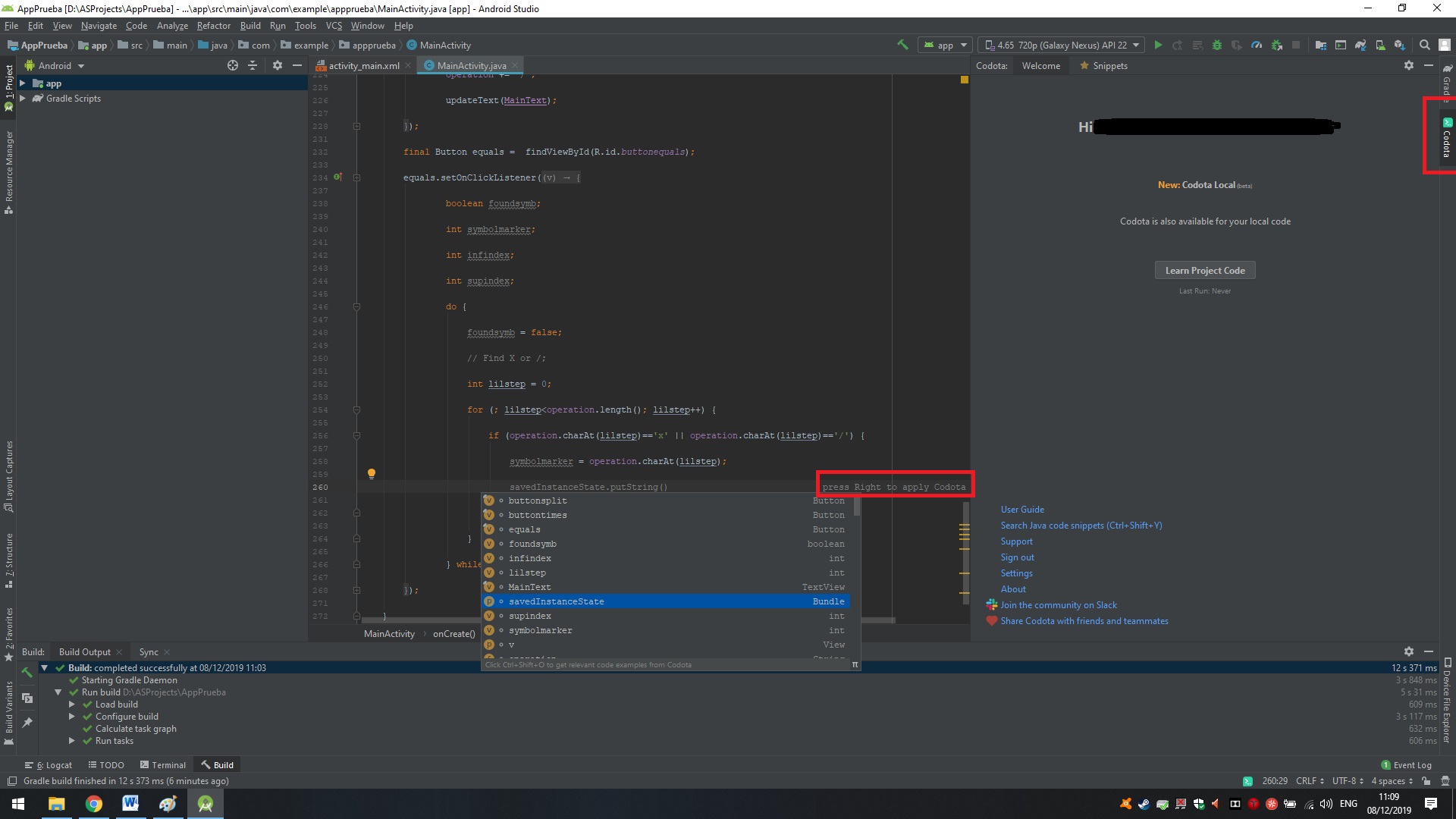


Si vamos ahora de nuevo a la ventana de administración de los plugins, veremos el plugin recién instalado, con una casilla al lado para activarlo o desactivarlo con facilidad:



Volviendo a la página de instalación, como se ha visto antes, hay links a instrucciones de instalación, términos de uso y a veces también a guía, lo cual ya depende de cada plugin específico.

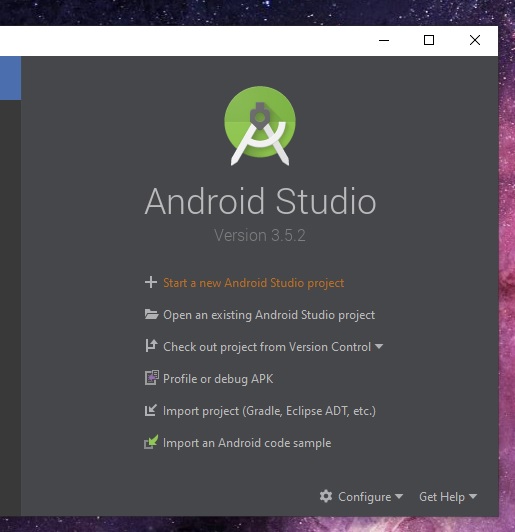
En el caso de Codota, además, hay que registrarse en su página con nuestro correo, pero esto no es el caso en todos los plugins. Tras todo el proceso de instalación, Codota nos permite acceder al menú que aquí se muestra apretando ctrl+espacio:



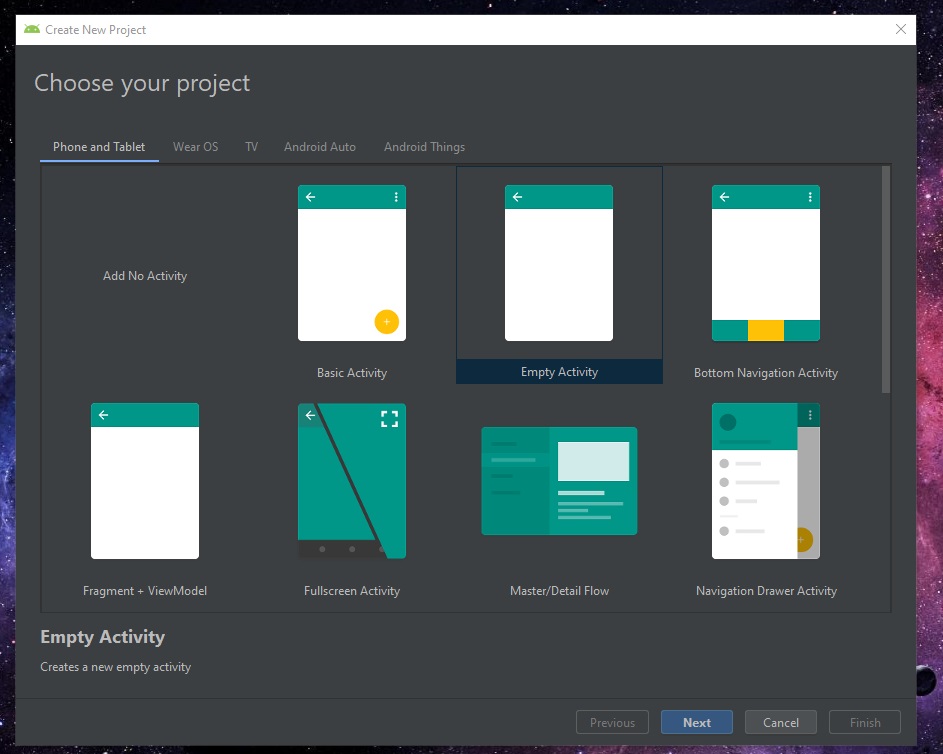
Este menú nos despliega una larga lista de elementos y llamadas a métodos que podemos insertar directamente en el código destacando el trozo que nos interesa y presionando flecha derecha.

También se puede ver a la derecha el menú creado dentro del entorno y links a los que ya hacía referencia el propio instalador, ahora a mano fuera del menú de administración de plugins.

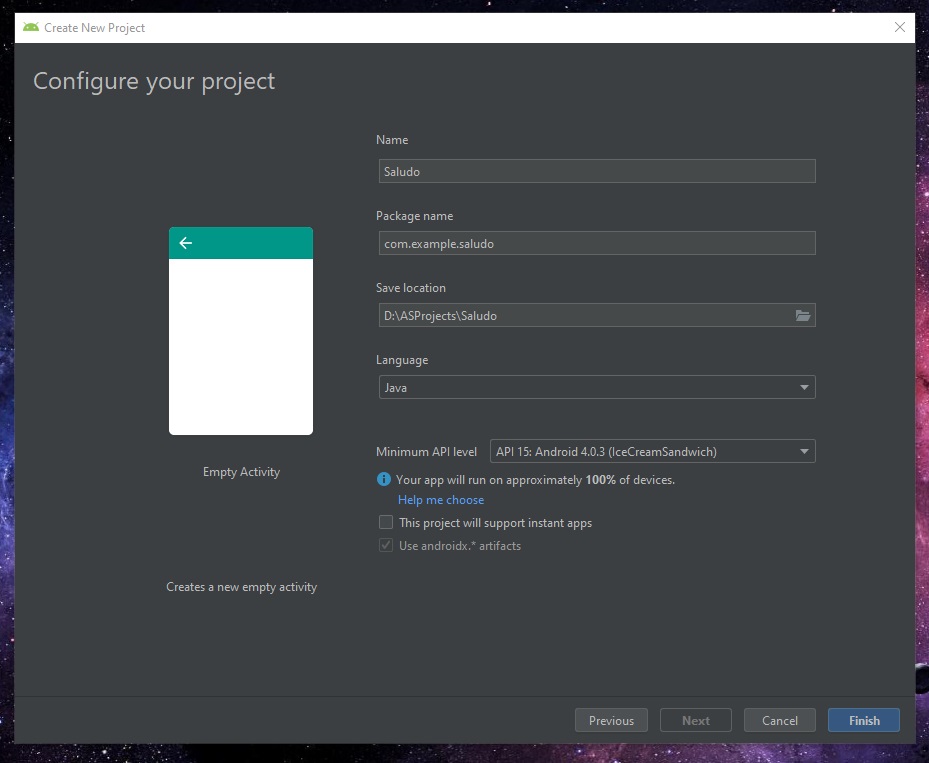
## Proyecto básico



A no ser que ya hubiéramos creado un proyecto con anterioridad, Android Studio se inicia en esta pantalla (desde la que ya habíamos accedido a la configuración). Seleccionando la primera opción, podremos crear un nuevo proyecto:



Lo primero que se nos pedirá es elegir una plantilla para la aplicación, para nuestra aplicación de prueba elegiremos la denominada “actividad vacía”, que nos dará una base suficiente de código y distribución de la vista para crear algo sencillo sin sobrecargarnos con funciones que no vayamos a usar.

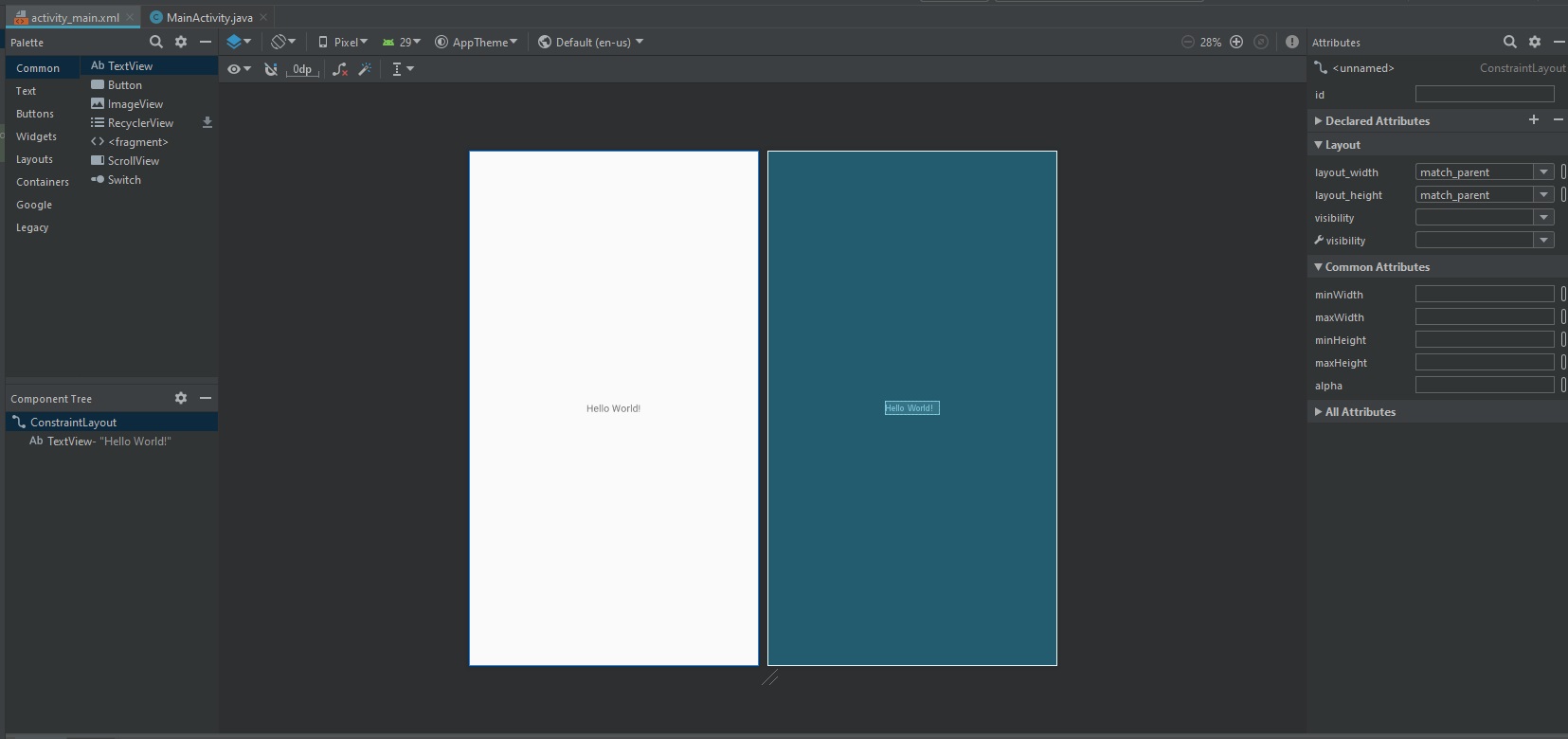


Elegimos un nombre para el proyecto y AS establecerá automáticamente un nombre para el paquete que contendrá todo el código (se puede cambiar pero hay ciertas restricciones que el mismo entorno impone antes de crear el proyecto). Se puede cambiar dónde se guardará el proyecto, siendo conveniente hacerlo en una carpeta propia, ya que se generarán varios archivos nada más empezar. Se elige el lenguaje y la API básica, así como un par de opciones adicionales que tampoco nos interesan ahora. Solo estableceremos nosotros el nombre y la localización y le daremos al botón azul de finalizar.



En la configuración por defecto, nada más abrir un proyecto, tendremos un archivo .java correspondiente a lo que en sí hace nuestra aplicación (aunque claro, más tarde no será el único .java que tendremos) y un .xml, que se refiere a la vista de la actividad principal de la aplicación (esto tarda un rato en aparecer, hasta que el entorno “construye” la aplicación). También tendremos numerosos menús “ocultos” en los bordes del entorno y, en la esquina superior derecha, el botón de “ejecutar” y el menú de elegir el sistema en el que se prueba la aplicación.

La configuración de este último elemento, para el cual es importante que en la instalación se haya marcado el Android Virtual Device, se explicará al final. Por el momento, miraremos la creación de la vista y los elementos más importantes.



Al abrir el XML, los elementos más importantes que nos encontramos son los siguientes:

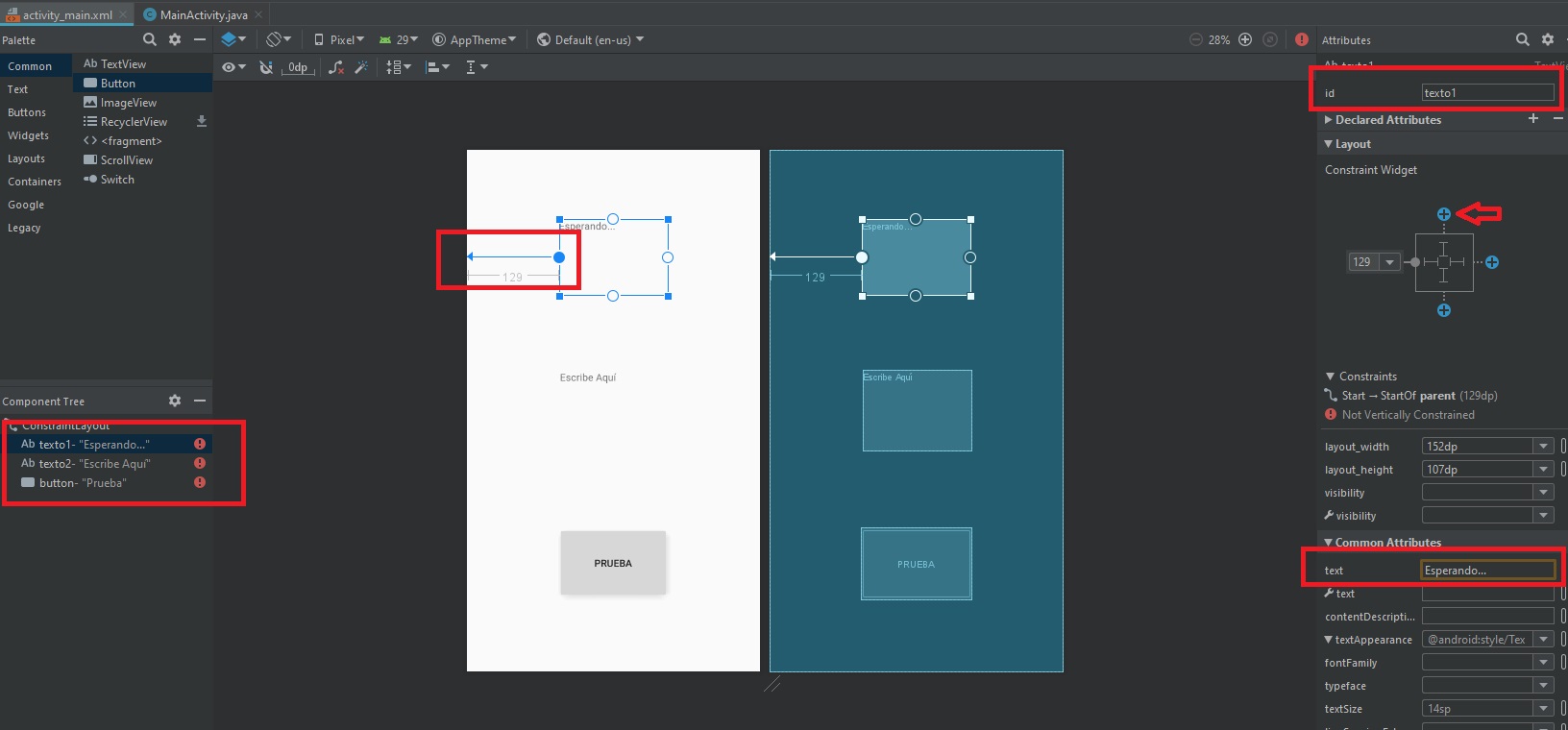
-Una vista de la aplicación en el centro. La que se encuentra en blanco es la “de diseño”, en la que podemos ver cómo queda la aplicación al final. La vista en azul solo contiene la arquitectura de la vista, sin mostrar, por ejemplo, las imágenes o los iconos y normalmente usando el id del elemento y no su contenido (trabajando solo con texto como haremos, esto no es notable).

-En la esquina superior izquierda, tenemos los elementos, divididos en categorías, siendo por defecto la categoría activada la de “elementos comunes”. Si en la configuración tenemos el “Drag&Drop” activado, podemos hacer doble clic sobre un elemento de la lista y arrastrarlo y colocarlo directamente en la vista.

-Justo debajo de la lista de elementos, tenemos la lista de todas las instancias de elementos que ya contiene nuestra vista.

-A la derecha, se encuentran en listas desplegables todos los atributos asociados al elemento que seleccionemos en la vista. El primero que siempre nos aparece es el atributo de “id”, identificación, sobre el cual hay que mantener control continuo a la hora de programar.

Lo primero que haremos será eliminar el elemento de ejemplo “Hello World”, haciendo clic sobre el elemento en la vista o la lista de instancias y dándole a suprimir. A continuación crearemos dos instancias de texto (la instancia que será de entrada debe ser PlainText, este se encuentra en la categoría “Text”, no en la de “Common”) y una instancia de botón.

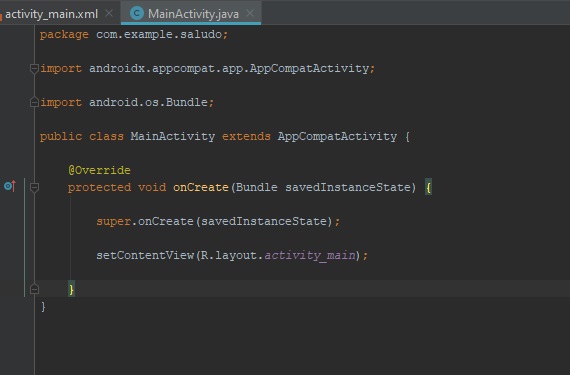


Tras crear cada elemento, se recomienda o casi requiere, hacer dos cosas. Primero, AS creará nombres genéricos para cada elemento de la vista, que podremos ver en el id. Para poder trabajar adecuadamente, cambiaremos los nombres a todos los elementos que introduzcamos, para poder reconocerlos y hacer referencia a ellos en el .java. Segundo, como se ve en la lista de instancias hacia la izquierda, tenemos varios avisos.

Estos avisos nos advierten que, de iniciarse la aplicación en este estado, todos los elementos se verán arrastrados al punto 0,0 de la pantalla. Esto es porque no tienen una posición fija establecida. Para evitar esto, se han de anclar los elementos en vertical u horizontal (en nuestro caso lo anclaremos de ambas maneras), a algún otro elemento de la vista o a la vista entera, arrastrando los círculos blancos que hay sobre el recuadro de cada elemento o con los botones azules que señala la flecha.

En nuestro caso también hemos puesto un texto inicial en los atributos, para mostrar que la mayoría de atributos son directamente modificables.

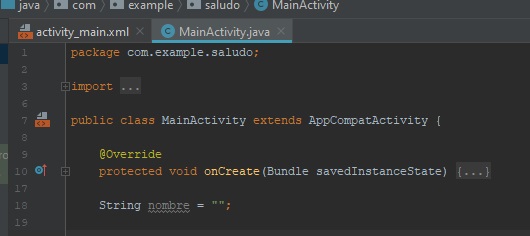
Una vez anclados y renombrados todos los elementos, podemos pasar directamente al .java



La pestaña del .java de la actividad no es demasiado compleja: es el texto plano en el que se programa en java con los colores predefinidos para palabras clave, métodos, etc… Tendremos algunos signos más y flechas con menos entre el código y el número de línea. Esto pliega y despliega algunos bloques del código para hacer más cómoda la vista.

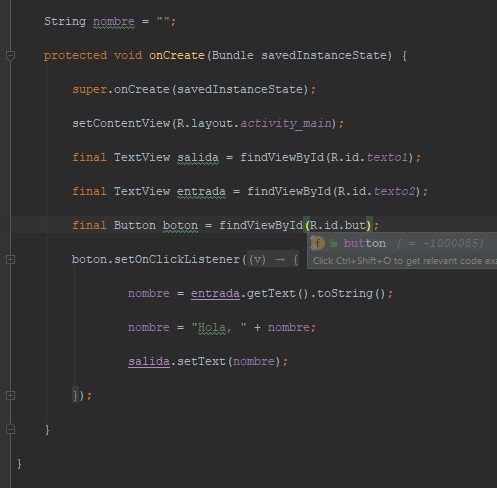
Con la plantilla que hemos elegido, tenemos un código de base que nos permitiría ejecutar la aplicación y mostrar la vista que acabamos de crear con el método onCreate (que básicamente carga los elementos establecido). Pero esta aplicación no haría absolutamente nada, sería una imagen fija. Hay que darle instrucciones a la aplicación para hacer algo.

Nosotros comenzaremos a escribir justo después de cerrarse (o antes de abrirse, se verá que se ha cambiado de sitio por comodidad en otra imagen) el método onCreate y declararemos una sola variable llamada nombre.



Ahora uno de los pasos más importantes en el uso de este entorno: La creación de referencias a los elementos de las vistas y la escritura del código a ejecutar, que ha de ir en su propio método.

Primero, la creación de las referencias. Aquí es donde cobra importancia lo de tener ids bien definidos y que no nos confundan a la hora de asociar los elementos de la vista con variables de nuestro código. La estructura básica (y el código que se refiere a los tres elementos antes creados) es la siguiente:

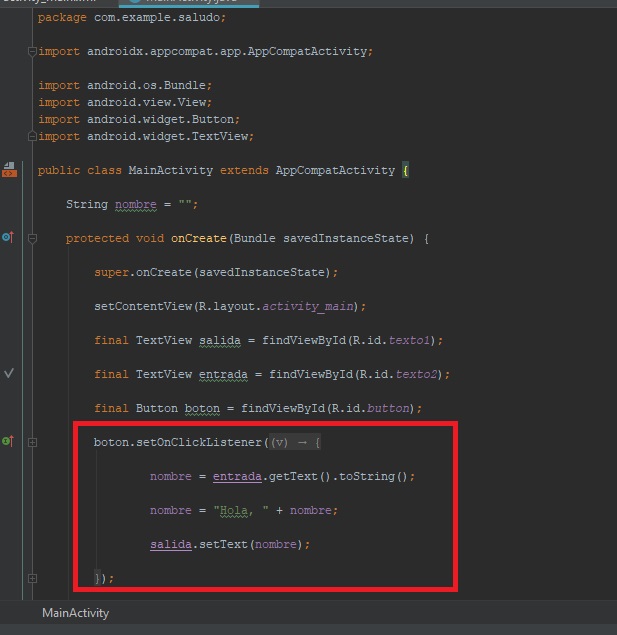


Nótese que se ha creado dentro de OnCreate, para cargar estas referencias en la misma creación de la aplicación (si no, puede cesar la ejecución nada más iniciarse). Se declara una variable “final” (ya que no modificaremos en sí la referencia a una vista), de la clase a la que pertenece el elemento al que nos queremos referir (normalmente es exactamente el mismo nombre que tienen en la lista de elementos que podemos crear) y un nombre. Esta referencia se inicializa, y esto ya es más propio de AS, con el método findViewById. Este método llama en sus atributos a R (una clase creada automáticamente por el sistema para asociar los recursos del XML con el Java) y al campo id, justo el que antes hemos establecido y que, después de otro punto, hacer referencia al valor que hemos establecido en la configuración de la vista. A partir de ahora, nos podremos referir a los elementos de la vista con el nombre de la referencia, en nuestro ejemplo “salida”, “entrada”, “boton”.

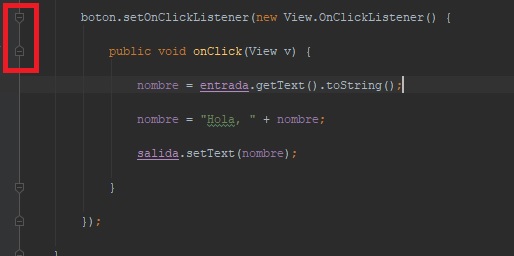
AS y algunos plugins nos ayudarán en mayor o menor medida a rellenar las líneas de código y poner los nombre correctos, como se muestra en la imagen, pero es muy aconsejable llevar identificadores reconocibles, sobre todo pensando en la posterior corrección de errores.

La segunda construcción básica es la de “escucha de eventos”. Los eventos son los cambios que tienen lugar en la pantalla de la aplicación (en sí OnCreate es un método disparado por un evento, que es el de la apertura de la aplicación). Estableciendo un escucha, podemos crear un bloque de código java que se ejecute cada vez que tiene lugar un evento y construir así nuestras aplicaciones. Para la de prueba, crearemos un escucha para nuestro botón y usaremos la referencia a entrada para leer el texto que luego introduzcamos en la aplicación. El botón, ante el evento de ser pulsado, ejecutará un código que toma el texto de “entrada”, construye una cadena con la concatenación estándar de java y la saca por el elemento “salida”.

Estos escuchas, además, deben ir dentro del método onCreate, para ser creados justo al iniciar la aplicación:

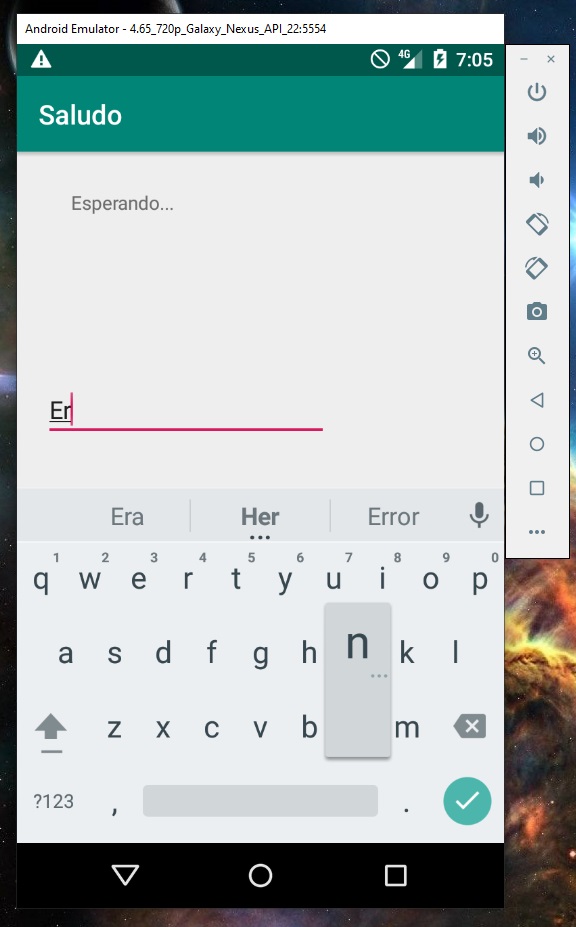
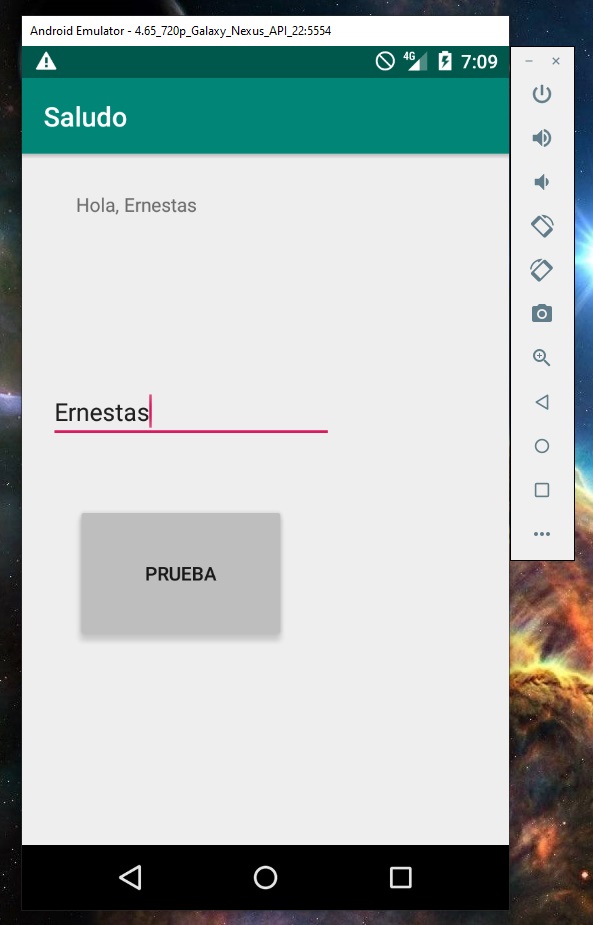


Un apunte adicional: se puede ver que hemos añadido unos import extras, que de faltar harán que algunas palabras clave del nuevo bloque aparezcan en rojo por no encontrarse disponibles los métodos (android.view.View, por ejemplo, es el paquete que controla los elementos de la vista). Como vemos, iniciamos el bloque haciendo referencia al control por la referencia ya creada en el código, en este caso “boton” y la llamada a setOnClickListener, que asocia el objeto que lo ha llamado con un escucha. La forma del bloque viene predeterminada por los propios atributos del primer método y contiene en él el onClick (la declaración completa es la de la siguiente imagen, pero AS lo recortará a esta forma cuando se guarde y se vuelva a abrir el proyecto). En onClick es donde va el código a ejecutar cuando se le dé al botón.



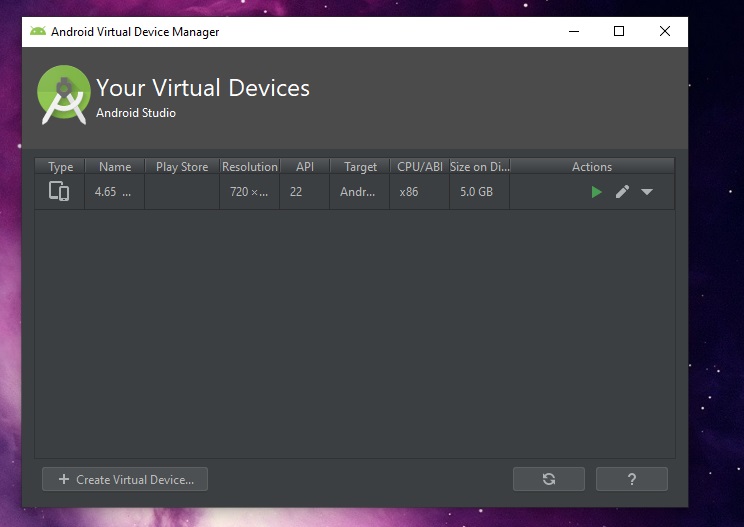
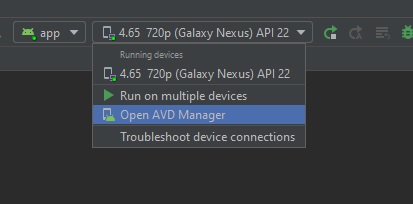
Nosotros asignamos a nombre el valor que tiene entrada al pulsar el botón, leyendo el texto introducido con los métodos getText() y toString(). Lo concatenamos con un saludo como haríamos en java y ejecutamos un pequeño método, setText(), sobre la salida, para mostrarlo en la aplicación.

Poniendo a prueba nuestra aplicación en un emulador de Samsung Galaxy Nexus, con el botón verde de ejecutar de la esquina superior derecha, veremos que se comporta como esperábamos:

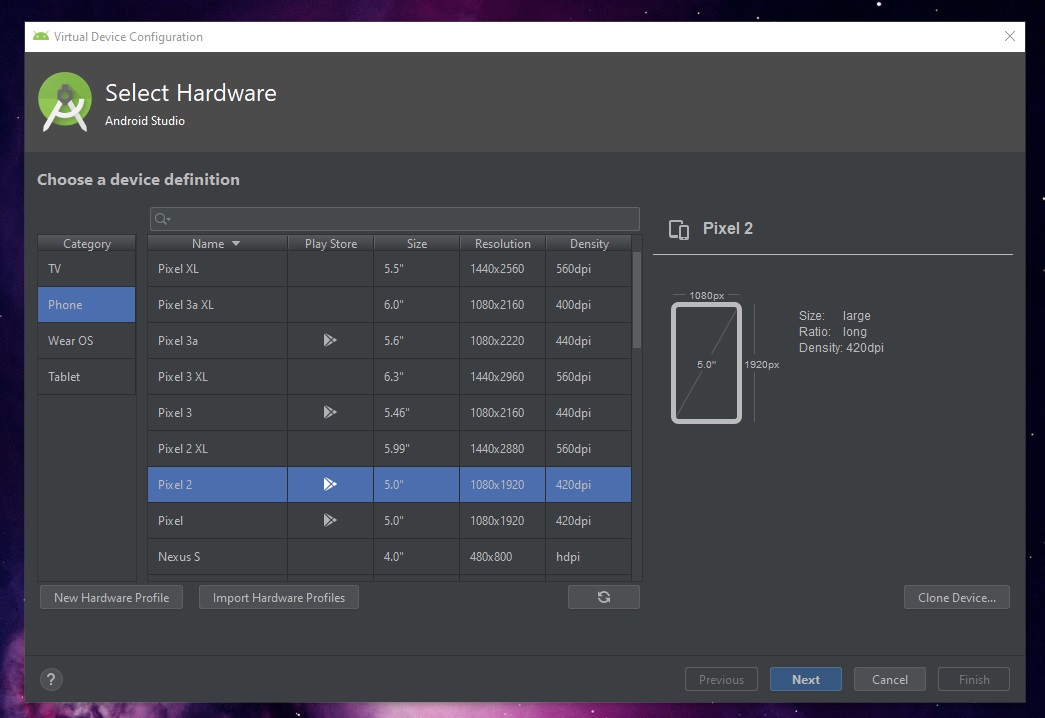


Ahora, queda un pequeño problema que resolver: La primera vez que iniciemos AS, es muy posible que no haya creado ningún emulador y no podamos poner a prueba la aplicación con el botón indicado. Es posible que nuestro ordenador ni siquiera pueda emular o que la función haya sido desactivada con la UEFI, en cuyo caso puede usarse nuestro propio dispositivo Android, siempre que tenga activadas las opciones de desarrollador y tengamos los drivers de USB adecuados (si todo está en orden, nuestro propio dispositivo aparecerá en la lista una vez conectado por cable y solo habrá que elegirlo y darle al botón de ejecución).

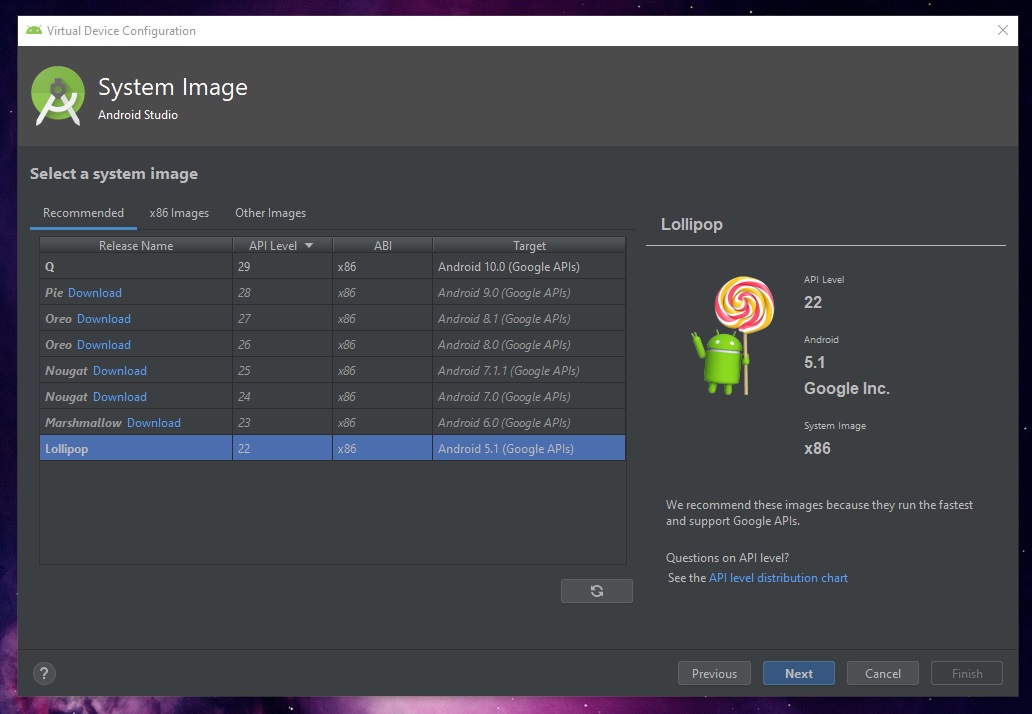
Si podemos y nos interesa a la hora de desarrollar, podemos probar nuestras aplicaciones en varios dispositivos emulándolos con el Android Virtual Device. Para usarlo, primero abriremos el administrador:



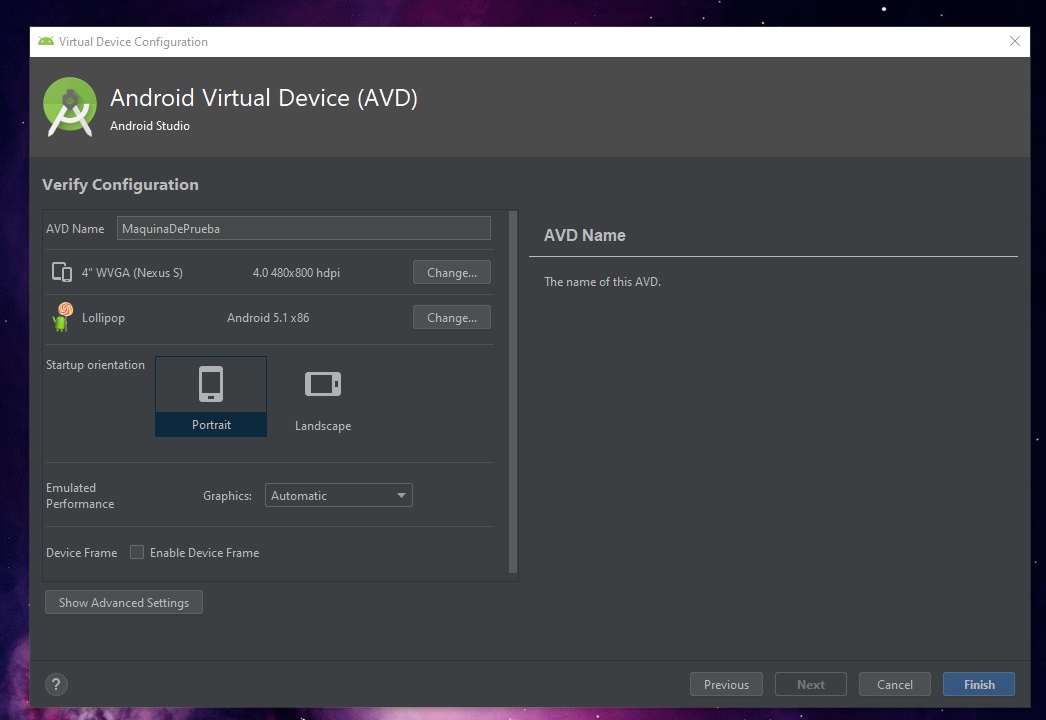
Aquí seleccionamos en la esquina superior izquierda “Crear Máquina Virtual”



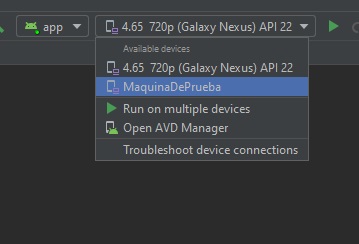
Aquí podemos, bien elegir entre una lista de hardware con sus especificaciones o crear nuestro propio perfil de hardware en el que probar las aplicaciones. Elegimos uno de los perfiles predefinidos y le damos a siguiente.



Se nos pedirá elegir una versión de Android a instalar. Si es nuestra primera máquina es muy posible que no tengamos ninguna disponible y que tengamos que descargarla con el botón de descarga.



A continuación le ponemos un nombre, revisamos la configuración y le damos al botón azul para finalizar.



Una vez finalizada la instalación, podremos ver nuestra máquina en la lista desplegable y usarla para la prueba de software, teniendo todos los elementos necesarios para el desarrollo preparados.