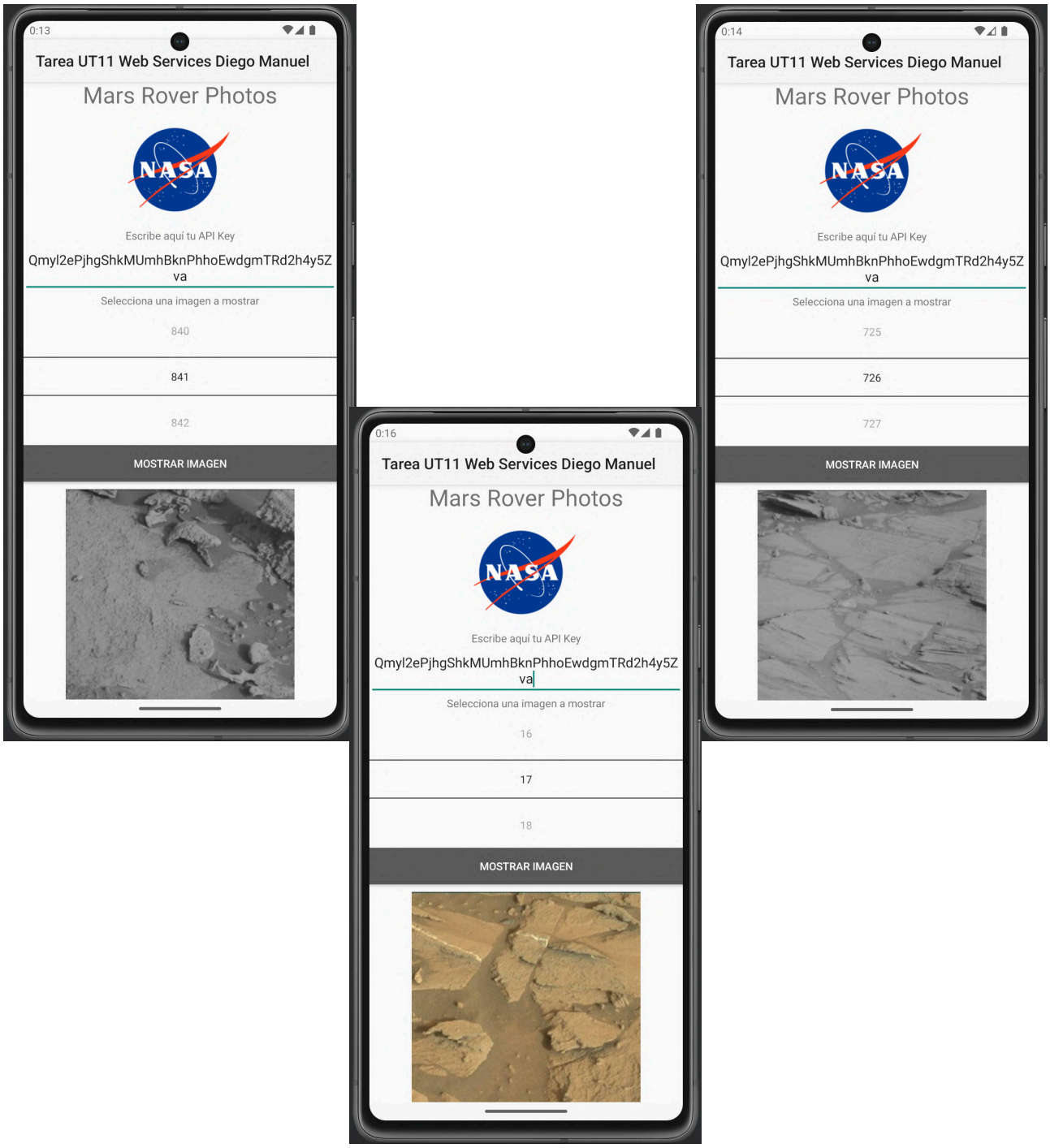
# Tarea 11 para Programación Multimedia y Dispositivos Móviles




Diego Manuel Carrasco Castañares

PMDM11.

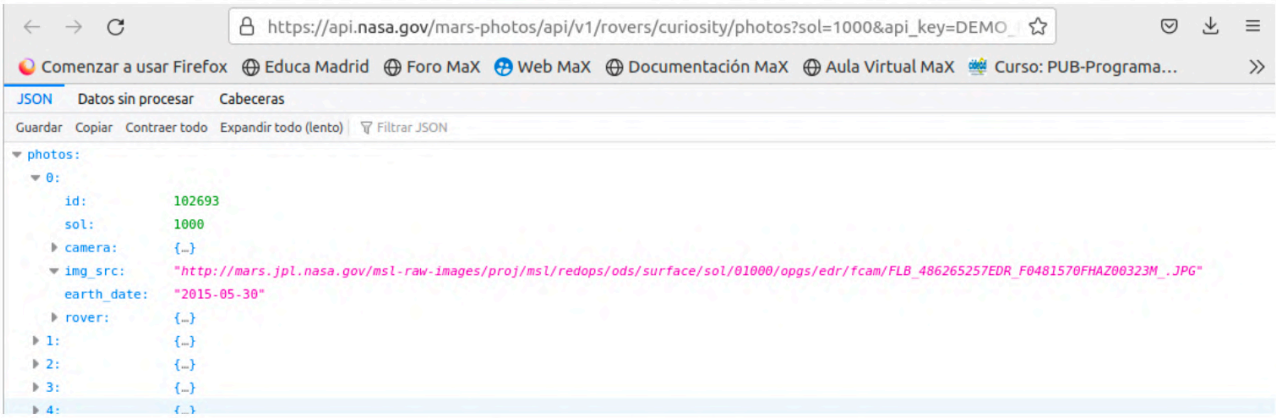
- Web Services. Tarea

Detalles de la tarea de esta unidad.

Enunciado.

Tarea UT11: Retrofit y Picasso para acceder a imágenes de servicios web.

Crea una aplicación que se conecte mediante Retrofit al servicio web con el que la NASA ofrece imágenes de Marte publicado en esta URL de ejemplo https://api.nasa.gov/mars-photos/api/v1/rovers/curiosity/photos?sol=1000&api_key=DEMO_KEY
Este servicio devuelve, con el parámetro indicado en la URL sol=1000 y una clave de API (api_key), un array JSON (que puedes manejar con el convertidor GSon) con información de numerosas imágenes (actualmente 855 imágenes) incluyendo la URL de la misma y que tiene el aspecto mostrado en la siguiente figura:




Google Developers (Uso educativo nc)

La aplicación que debes desarrollar solicitará al usuario dos parámetros y mostrará en la parte inferior la imagen mediante Picasso:

- ✓ Tu App Key que debes conseguir registrando tu correo electrónico en esta web https://api.nasa.gov/
- ✓ El número de imagen (entre 0 y 855) que el usuario elija. Se puede utilizarNumberPicker (como aparece en la imagen de muestra) para limitar la elección del usuario a ese rango de números. Más información sobre su uso: https://developer.android.com/reference/android/widget/NumberPicker

El aspecto esperado de la aplicación al seleccionar la imagen en la posición 1 del array devuelto por el web service, es el

mostrado en la captura de pantalla del móvil (la imagen solicitada tarda unos instantes en aparecer).


Google Developers (Uso educativo nc)

NOTA 1: Para que sea más sencilla la creación de las clases POJO usadas por Retrofit, puedes copiar desde un navegador como Firefox el texto JSON devuelto por el web service con la llamada de ejemplo indicada en la captura anterior, y mediante esta web: http://pojo.sodhanalibrary.com/ o esta: https:/ www.jsonschema2pojo.org/ (marcando JSON y Gson)ayudarte a crear la clase principal Curiosity (con la propiedad ArrayList<Photo>), y la clase Photo para definir la propiedad img_src que es lo que necesitamos para cargar la imagen con Picasso.

NOTA 2: Ten cuidado con las URLs de las imágenes porque el web service a día de hoy las devuelve con HTTP, lo cual genera un problema de seguridad. Puedes sustituir el HTTP por HTTPS de forma programática con la función replaceFirst() o utilizar la clase Java URL para hacerlo.

## Criterios de puntuación. Total 10 puntos.

La valoración total de la tarea es de 10 puntos repartidos del siguiente modo:

- ✓ Aspecto: la aplicación se muestra de un modo similar al mostrado. 1 punto.
- ✓ Funcionamiento de la app:

- La aplicación se conecta y muestra la imagen seleccionada: 2 puntos.
- La aplicación usa correctamente Retrofit definiendo las clases POJO de forma adecuada: 6 puntos.
- La aplicación usa Picasso para mostrar la imagen: 1 punto.

Puesto que no existe una evaluación por unidades la evaluación se realiza en base a los criterios generales del módulo.

## Recursos necesarios para realizar la Tarea.

✓ Ordenador con Android Studio instalado y suficientes recursos para ejecutar el emulador.
✓ Contenidos de la Unidad, muy importantes los ejercicios resueltos de la misma.
✓ Páginas web de los desarrolladores de los sistemas operativos para móviles.
✓ Uso de retrofit: https://square.github.io/retrofit/
✓ Más información sobre retrofit: https://github.com/square/retrofit

## Consejos y recomendaciones.

✓ Lee atentamente el enunciado y asegúrate de haber entendido lo que has de hacer.
✓ Intenta reproducir en tu Android Studio los ejercicios resueltos previamente.
✓ No dudes en comentarle a tu tutor o tutora cualquier duda que te pueda surgir.
✓ Envíasela a tu tutor o tutora a través del sistema establecido en la plataforma.
✓ Las capturas y el contenido de los ficheros deben aparecer en perfecto orden para que esté claro lo que deseas mostrar.
✓ Los diseños deben mostrarse lo más parecidos posibles a lo que se pide.
✓ Debe llevarse a cabo sobre una versión actual de Android.

## Indicaciones de entrega.

Una vez realizada la tarea elaborarás un documento PDF.
✓ El documento pdf mostrará capturas de la app:
- Captura del aspecto de tu aplicación al arrancar la primera vez.
- Captura del aspecto de tu aplicación cuando cargues una de las imágenes.

✓ Añade el contenido de los ficheros editados: java, xml, archivo de manifiesto, graddle, etc.

✓ Las capturas que no sean de tu aplicación, sino de tu Android Studio, deben mostrar el fondo de pantalla y la fecha y hora de la barra de estado de tu sistema operativo para garantizar la originalidad del material.

También se deberá entregar la carpeta comprimida del proyecto en formato .zip (File → Export → Export to Zip file...)

Entrega los archivos a través de la plataforma. Se nombrarán siguiendo las siguientes pautas:

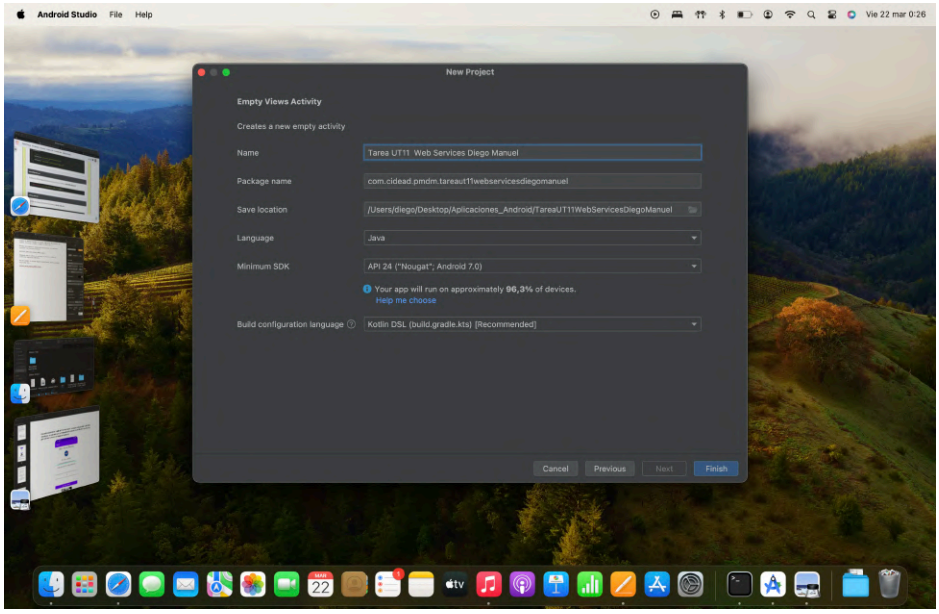
apellido1_apellido2_nombre_PMDM11_Tarea

Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños.

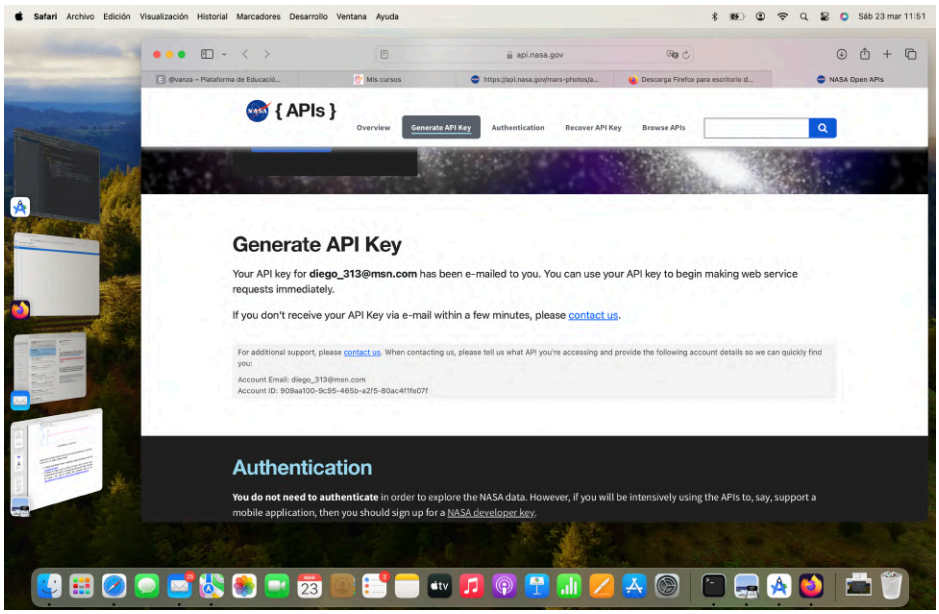
Así por ejemplo la alumna Begoña Sánchez Mañas, debería nombrar esta tarea como...

sanchez_manas_begona_PMDM11_Tarea

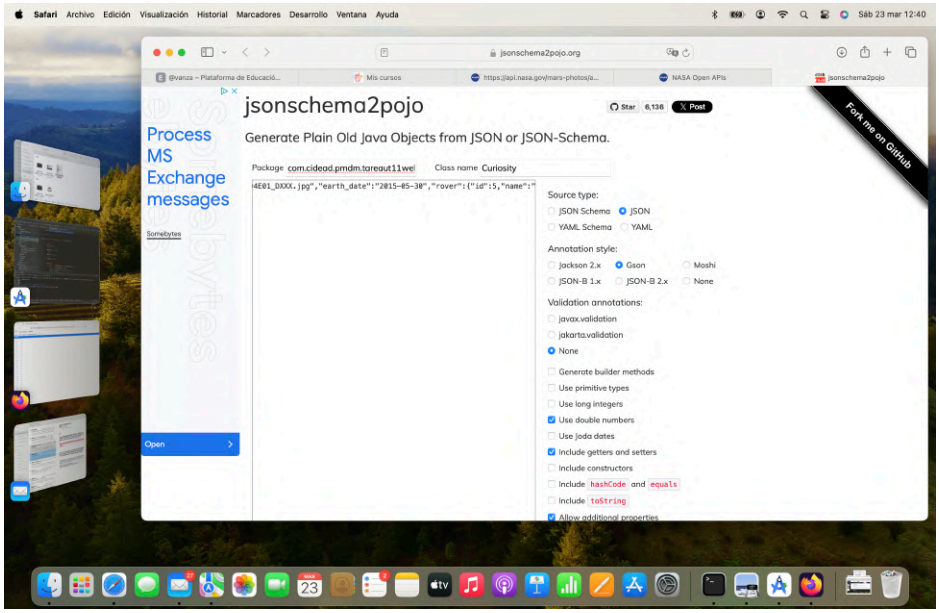
Empezaremos creando el proyecto, al cual llamaremos Tarea UT11 Web Services Diego Manuel.



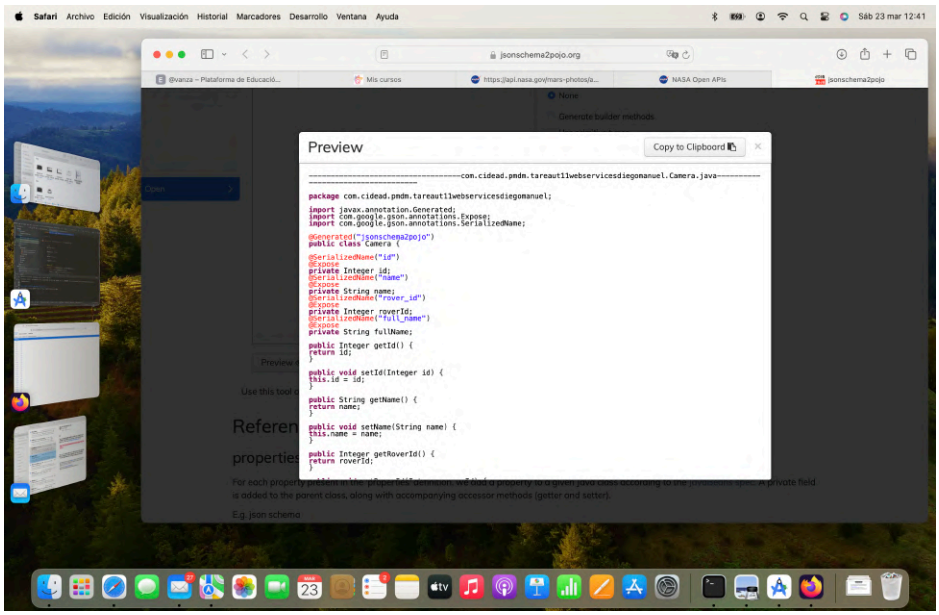

Ahora pasaremos a registrarnos en la web de la nasa para generar el Api Key y poder acceder a las imágenes.



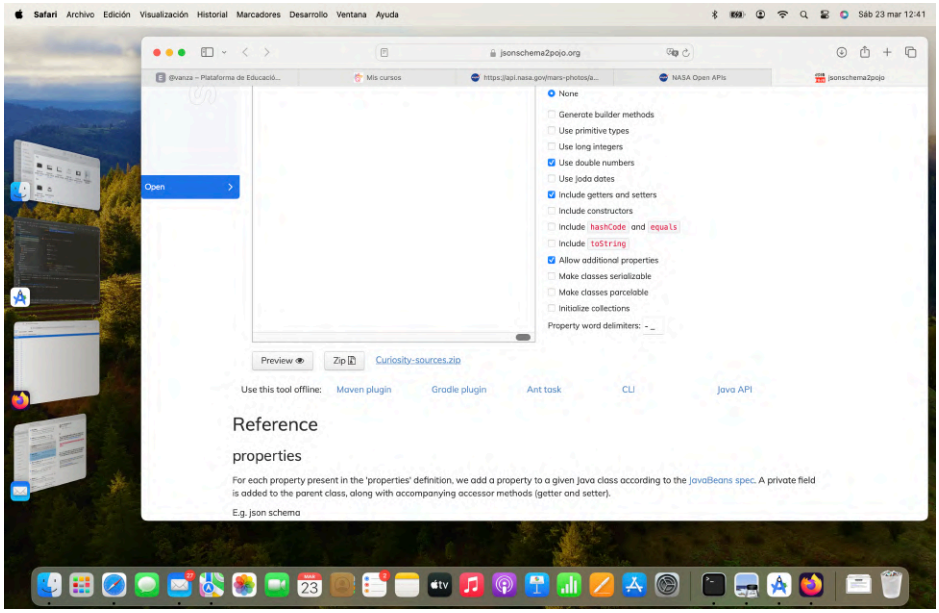

Tras ello accederemos a la web indicada en la tarea para generar los POJO, en la cual copiaremos parte del json (ya que entero me da un error de exceso de caracteres), elegiremos como tipo de archivo JSON y como estilo Gson. Los demás campos los dejaremos como están.



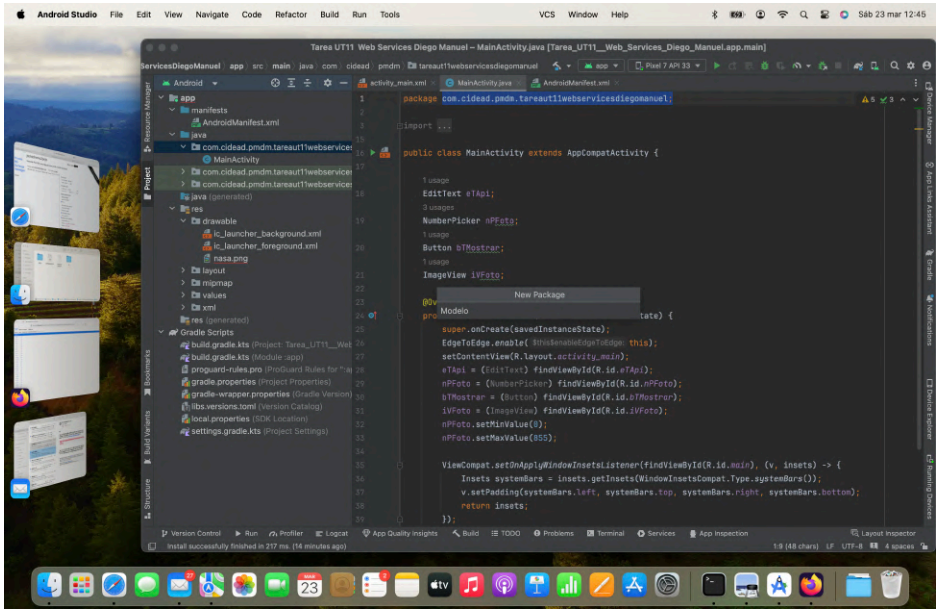

En la siguiente imagen podemos ver un preview de como quedaría.

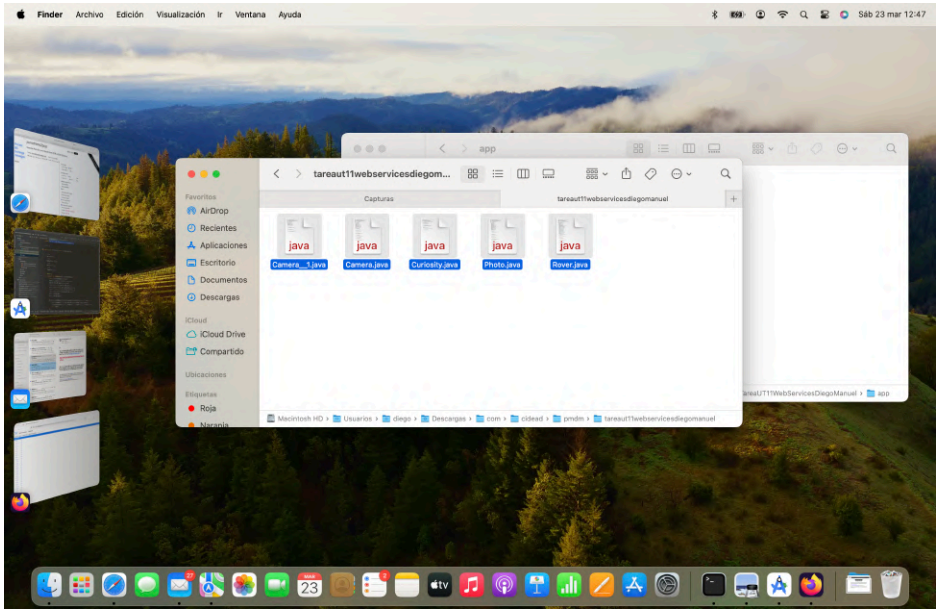



Tras ello clicaremos en zip y en el enlace que nos abre para descargarlo en formato zip.



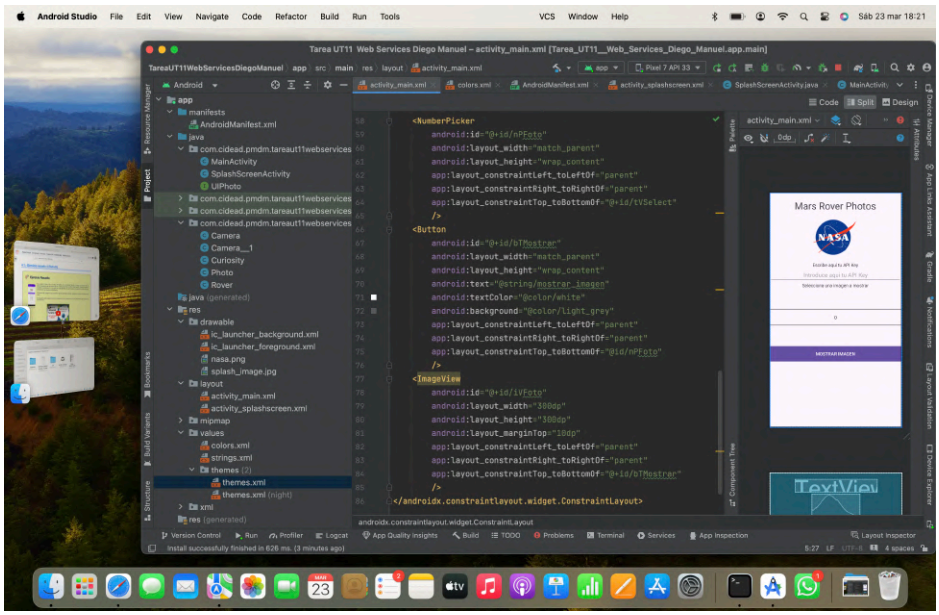

Tras ello, en el proyecto, crearemos una carpeta llamada Modelo, en la cual vamos a meter los archivos generados.



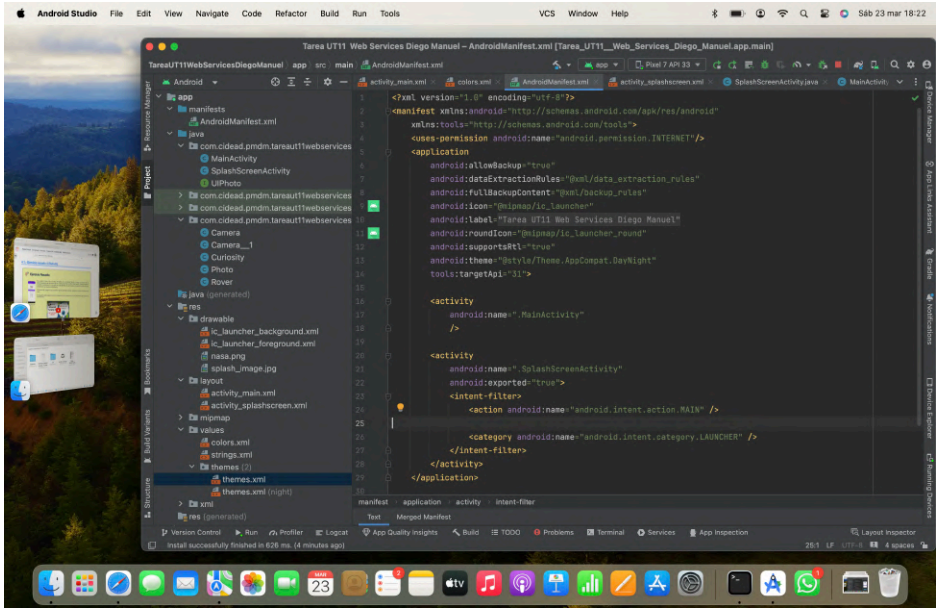



Ahora pasaremos a componer el activity_main.

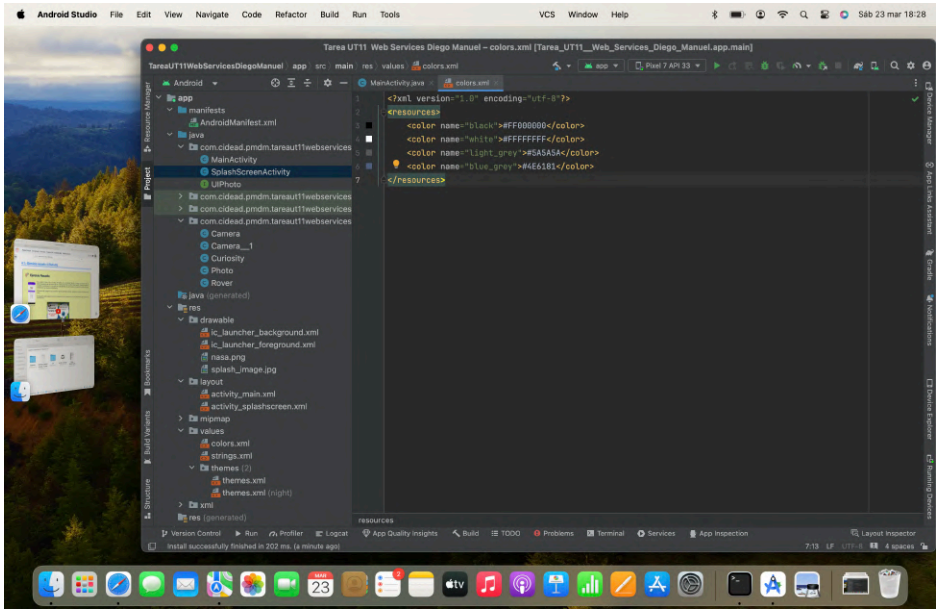
Dicho activity esta compuesto por un TextView, que será el encargado de ejercer de título con un tamaño de letra más grande, un ImageView que contendrá la imagen de la nasa, otro TextView que indicará al cliente que introduzca el API Key, un EditText que recogerá el API Key introducido por el cliente, otro TextView que indicará al cliente que elija la foto a mostrar, un NumberPicker que contendrá las imágenes contenidas en la página de la nasa, un Button que contendrá el código para mostrar la imagen elegida en el NumberPicker y un ImageView que contendrá mostrará la imagen recibida.




En el AndroidManifest cambiaremos el tema para que nos muestre el título en la ActionBar y modificaremos para que nos muestre el SplashScreen.



En el archivo colors.xml añadiremos dos colores mas, un gris claro para ponerlo de fondo en el botón y un azul grisáceo para el fondo del SplashScreen.




Ahora vamos a pasar a crear el código en el MainActivity.

Dicha clase contendrá seis atributos, un EditText que contendrá el api introducido por el cliente, un NumberPicker que contendrá el número de imagen seleccionada por el usuario, un Button que contendrá el código para mostrar la foto, un ImageView que mostrará la imagen, un String que contendrá un mensaje que se mostrará en caso de que el usuario no introduzca el API Key y otro string, pero de tipo final, que contendrá la url hasta las imágenes.

En el método onCreate asociaremos cada atributo a su homónimo en el xml, estableceremos que el valor mínimo es 0 y el máximo 855 en el NumberPicker (que son las fotos del Rover de Marte contenidas en la web).

Capturaremos el evento del botón, el cual comprobará mediante un if si el usuario no ha introducido el API Key, en cuyo mostrara un Toast indicándolo.

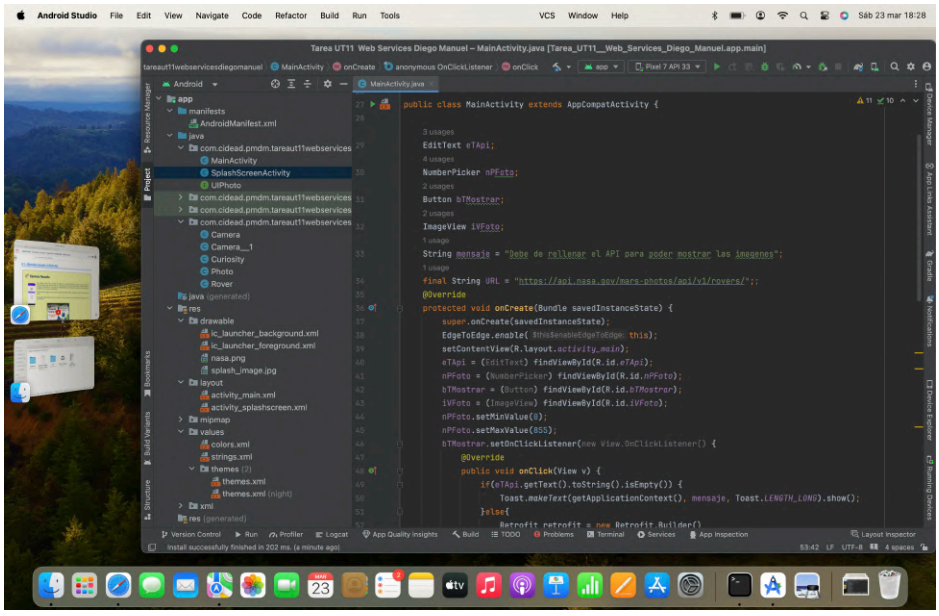
Si por el contrario si se ha introducido el API Key crearemos una instancia de Retrofit al cual le pasaremos como baseURL la url de las imágenes, agregará un factory para serializar y deserializar los objetos JSON usando la biblioteca Gson y el build que construirá la instancia como tal.

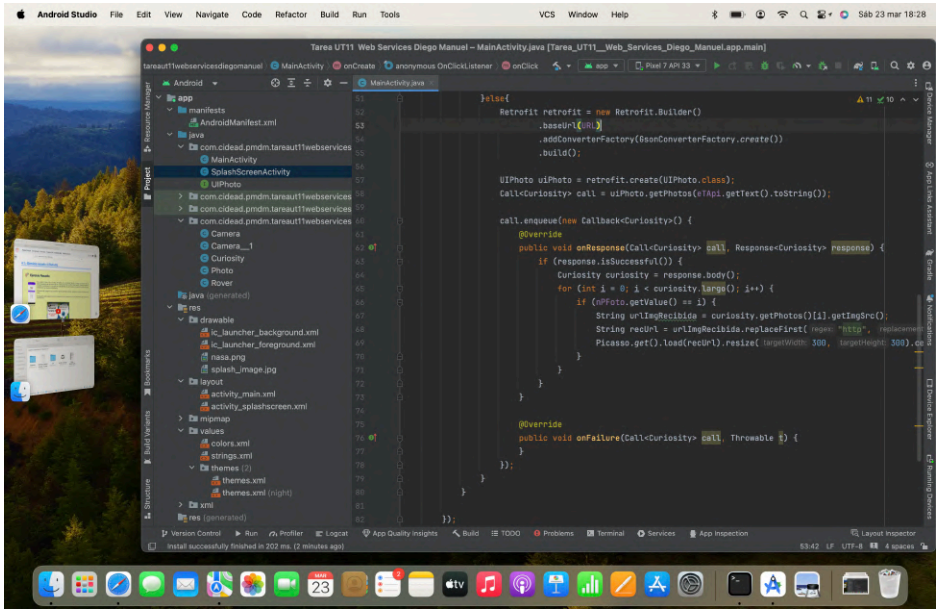
Tras ello crearemos un objeto de la interfaz UIFoto, que usará la instancia del Retrofit creado anteriormente para obtener las fotos requeridas por el usuario.

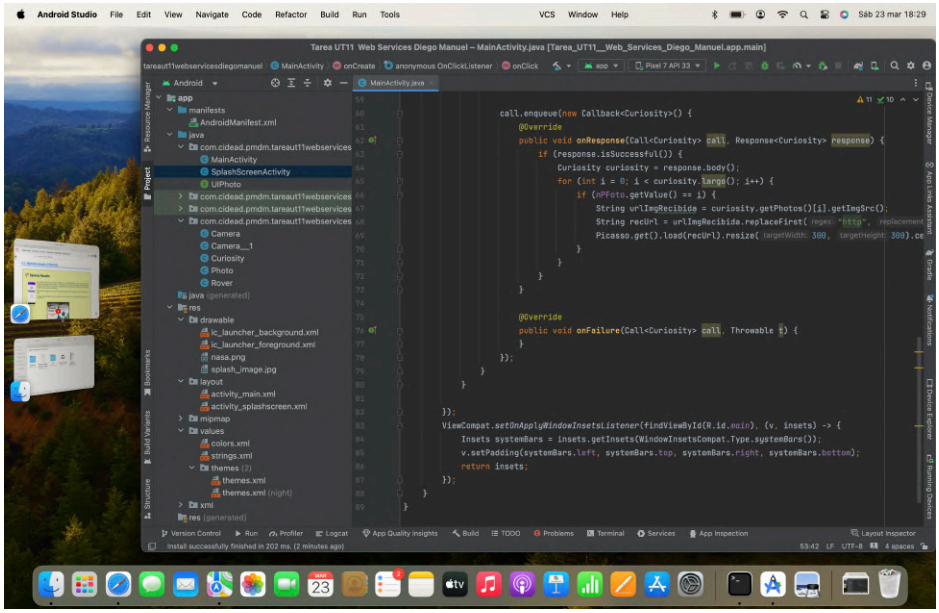
Para poder extraerlas crearemos un Call que contendrá un array de contendrá las fotos y devolverá la seleccionada por el usuario en el NumberPicker.

A le pasaremos dicho Call al enqueue para que realice la llamada.

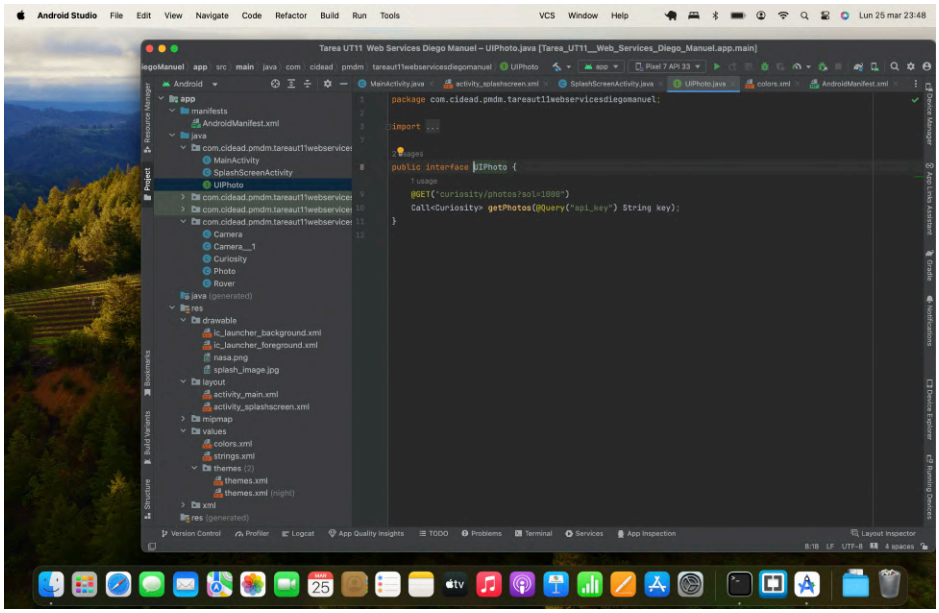
Para ello sobre escribirá el método onResponse, el cual contendrá un if, que comprobará si ha recibido de forma correcta la repuesta a la llamada Creará una nueva instancia de la clase Curiosity creada desde la web indicada en la tarea, recorrerá mediante un bucle for la longitud de la misma, y mediante un if contenido en el for anterior, comprobará si coincide la foto en la que se encuentra el for con la seleccionada por el usuario en el NumberPicker, en cuyo caso extraerá y guardara en un string la url de la imagen, reemplazara el protocolo “http” por “https” para que no nos genere Android Studio ningún error de seguridad, y mediante la librería Picasso extraerá y mostrará la imagen en el ImageView.







La clase UIFoto será del tipo interface y establecerá un punto de acceso a la api web, donde se alojan las fotos del Rover, mediante Retrofit. La interfaz dispone de un método llamado getPhotos() que realizará una solicitud GET a la URL especificada con un parámetro de consulta que será el api_key obtenido de dicha página.



Y con esto damos por finalizada la tarea.