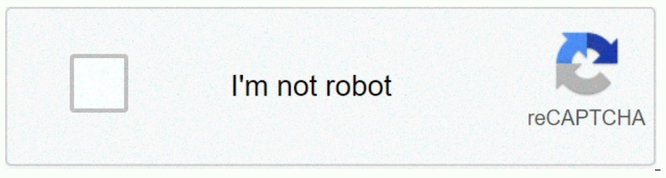

Continue

Android 8. 1 0 features

Written by Christian ColladoAndroid 8.1 Oreo landed in our lives earlier this month of the year, with some news under the hood trying to make our experience with Google's mobile platform much more complete. However, to date only owners of the latest Nexus and Google Pixels can enjoy the latest iteration of the operating system, or what is the same, less than 0.5% of Android users.Perhaps that is why many of the features of Android 8.1, however useful or revolutionary they are, have been overlooked by a good part of the community of fans and users of the operating system. Today we are going to collect some of these new features that came with the latest version of the system, but that for some reason were almost completely ignored. The best Android 8.1 news you didn't know might be of interest: Android 8.0 Oreo, in-depth analysis and opinionThe battery level of Bluetooth devicesThe customization levels of some companies, such as OnePlus OxygenOS, have for some time integrated an option through which to quickly display the battery level of devices connected via Android 8.1 Bluetooth.Su, this option is finally included in the Android warehouse , and just open the quick settings panel to see at a glance how much battery our speaker, headphones or any other device connected via Bluetooth has.Dynamic themes Android users in their purest edition have waited years for the arrival of an integrated themed engine, which allows us to change the appearance of the system to our liking without relying on external applications. Although Android 8.1 still does not offer us this possibility, it takes an important step forward through the new dynamic themes, a function that consists in switching between a dark or light appearance depending on the wallpaper used, all automatically without the user having to change the theme manually. Neural network APIID intelligence is not just the future. With Android 8.1, it's also the current one, which is that app developers already have an API through which to introduce machine learning systems into their apps. WallpaperColors API API for API and shooting because it touches me. If I used to talk to you about the Neural Networks API, now it's your turn for another novelty introduced in Android 8.1, which few users have noticed. WallpaperColors is an API that allows developers, or the operating system itself, to extract the main color from the background, to apply it at different points in the interface. In the images on these lines, you can see several examples with colors, which play in the shadow that appears when you turn on the power menu – also new). Restriction of background applicationsDo after android version 6.0 Marshmallow, we enjoyed an energy saving system called DOE. Over time, this feature has been improved, and little by little, it has become a great ally when it comes to saving battery. However, more advanced users have lost the ability to make the SYSTEM a little more aggressive. In Android 8.1, a new background app restriction method has been introduced, which allows the user to prevent them from running when they are not open. You can read more about how this feature works in the article we dedicate to you at that time. Burn-in protection Google Pixel 2 XL screen issues have sparked a controversy that, while silently, takes several years with us: oled panel flaws. As we explained at the moment, this technology is subject to burns, which occur when a static image remains for long periods of screen time. With Android 8.1, Google decided to take action, introducing various protection systems. The first is to change the style of the virtual navigation bar in some of the system apps, such as settings, instead of a white bar with black buttons, instead of a black bar with white buttons. However, that is not all. With Android 8.1, when accessing any application, the navigation buttons are obscured, in order to reduce the contrast between the background and the buttons themselves, and thus avoid the burn-in effect as much as possible. Keep in mind that gray buttons don't return to their original shape until the navigation bar is reused to go back, go home, or open recent apps. Android 8.1 (API level 27) introduces a variety of new features and features for users and developers. This document highlights what's new for developers. Android Oreo (Go edition) Android Go is our initiative to optimize the Android experience for billions of people coming online around the world. Starting with Android 8.1, we are making Android a great platform for entry-level devices. Android Oreo (Go Edition) configuration features include: Memory optimizations. Improve memory usage across the platform to ensure apps can run efficiently on devices with 1 GB or less RAM. Flexible targeting options. New hardware feature constants that allow you to direct app deployment to normal or low-RAM devices through Google Play. Google Play will give visibility to apps optimized by developers to provide a great experience for billions of people with building billions of guidelines. We've updated the creation of guidelines for billions with more guidance on how to optimize your app for devices running Android Oreo (Go edition). For most developers, optimizing your existing APK or using Google Play's multiple APK feature to target a version of your APK to low-RAM devices is the best way to prepare for devices running Android Oreo (Go edition). Remember that making your app lighter and more efficient gives benefits to your entire audience, regardless of device. NEURAL Networks API The Neural Networks API provides accelerated calculations and inference for on-device machine learning frameworks such as TensorFlow Lite, Google's cross-platform ML library for mobile devices, as well as Caffe2 and others. Visit the TensorFlow Lite open source search file to download and documents. TensorFlow Lite works with the Neural Networks API to run models like MobileNets, Inception v3, and Smart Reply efficiently on your mobile device. Android 8.1 AutoFill Framework Updates (API Level 27) provides several improvements to the AutoFill Framework that you can incorporate into apps. The BaseAdapter class now includes the setAutofillOptions() method, which provides string representations of values in an adapter. This is useful for selection box controls that dynamically generate values in their adapters. For example, you can use the setAutofillOptions() method to provide a string representation of the list of years that users can choose as part of a credit card expiration date. Autofill services can use string representation to appropriately populate views that require data. In addition, the AutofillManager class includes the notifyViewVisibilityChanged(View, int, boolean) method that you can call to notify the framework of changes in the visibility of a view in a virtual structure. There is also an overload of the method for non-virtual structures. However, non-virtual structures typically do not require explicit notification to the framework because the method is already called by the View class. Custom descriptions are useful to help the autofill service clarify what is saved; For example, when the screen contains a credit card, it might display a credit card bank logo, the last four digits of the credit card number, and its expiration number. For more information, see the CustomDescription class. Validator objects are used to avoid displaying the autosave UI when the validation condition is not met. For more information, see the Validator class along with its subclasses, LuhnChecksumValidator and RegexValidator. Android 8.1 notifications includes includes The following changes to notifications: Apps can now only ring a notification alert once a second. Warning sounds that exceed this frequency are not queued and are lost. This change does not affect other aspects of notification behavior, and notification messages are still sent as expected. NotificationListenerService and ConditionProviderService are not supported on low-RAM Android devices that return true when ActivityManager.isLowRamDevice() is called. EditText update starting at API level 27, the EditText.getText() method returns an Editable object; he previously returned a CharSequence. This change is compatible with previous versions, because Editable implements CharSequence. The editable interface provides valuable additional functionality. For example, because Editable also implements the Spannable interface, you can markup content within an Instance of EditText. Secure navigation actions at the code level Using the WebView implementation of the Secure Browsing API, your app can detect when an instance of WebView attempts to navigate to a URL that Google has classified as a known threat. By default, the WebView shows an interstitial that alerts users to the known threat. This screen gives users the ability to upload the URL anyway or return to a secure previous page. In Android 8.1, you can define the way your app responds to a known threat: You can control whether your app reports known threats to Safe Browsing. You can have your app automatically take a specific action, such as returning to security, whenever it encounters a URL classified as a known threat. Note: For optimal protection against known threats, wait until secure browsing has been initialized before invoking the loadUrl() method of a WebView object. The following code snippets show how you can indicated that your app's WebView instances can always safely return after detecting a known threat: AndroidManifest.xml <manifest> <application> ... <android meta-data:name=android.webkit.WebView.EnableSafeBrowsing android:value=true></meta-data> </application> </manifest> MyWebActivity.java private var superSafeWebView: WebView? null private var safeBrowsingIsInitialized: Boolean : false // ... override fun onCreate(savedInstanceState: Bundle?) super.onCreate(savedInstanceState) superSafeWebView (WebView(this).apply - webViewClient , MyWebViewClient() safeBrowsingIsInitialized , false startSafeBrowsing(this@SafeBrowsingActivity, - success -> safeBrowsingIsInitialized - true if (!success) - Log.e(MY_APP_TAG, Failed to initialize sicura!) webView privato superSafeWebView; private booleano safeBrowsingIsInitialized; ... @Override protected void onCreate(Bundle savedInstanceState) - super.onCreate(savedInstanceState); superSafeWebView - new WebView(this); superSafeWebView.setWebViewClient(new MyWebViewClient()); safeBrowsingIsInitialized - false; false; new ValueCallback<Boolean>() - @Override public void onReceiveValue(Boolean success) - safeBrowsingIsInitialized : true; if (!success) - Log.e(MY_APP_TAG, Safe Browsing could not be initialized!); MyWebViewClient.java MyWebViewClient class : WebViewClient() - // Automatically returns to security when you try to load a Web site that // Safe Browsing has identified as a known threat. An instance of WebView // calls this method only after Safe Browsing has been initialized, so no conditional logic is required // in this case. override fun onSafeBrowsingHit(view: WebView, request: WebResourceRequest, threatType: Int, callback: SafeBrowsingResponse) - // The true argument indicates that the app reports incidents such as // this to Safe Browsing. callback.backToSafety(true) Toast.makeText(view.context, Unsafe web page blocked., Toast.LENGTH_LONG).show() - public class MyWebViewClient extends WebViewClient - // Automatically returns to security when you try to load a Web site that // Safe Browsing has identified a known threat. An instance of WebView // calls this method only after Safe Browsing has been initialized, so no conditional logic is required // in this case. @Override void public onSafeBrowsingHit(WebView view, WebResourceRequest request, int threatType, SafeBrowsingResponse callback) - // The true argument indicates that the app reports incidents such as // this to Safe Browsing. callback.backToSafety(true); Toast.makeText(view.getContext(), Unsafe Web Page locked., Toast.LENGTH_LONG).show(); The MediaMetadataRetriever class has a new method, getScaledFrameAtTime(), that finds a frame near a certain time position and returns a bitmap with the same aspect ratio as the source frame, but resized to fit

a specified rectangle of width and height. This is useful for generating preview images from the video. We recommend that you use this method instead of getFrameAtTime() which can waste memory because it returns a bitmap with the same resolution as the source video. For example, a frame of a 4K video would be a 16 MB bitmap, much larger than would be needed for a thumbnail image. Android 8.1 (API level 27) introduces a new SharedMemory API. This class creates, maps, and manages an anonymous SharedMemory instance. Set memory protection to a SharedMemory object for reading and/or writing, and because the SharedMemory object is Parcelable, you can easily navigate to another process through AIDL. The SharedMemory API interacts with the ASharedMemory feature in the NDK. ASharedMemory provides access to a file descriptor, which can then be mapped to read and write. It's a great way share large amounts of data between apps or between multiple processes within a single app. WallpaperColors API Android 8.1 (API level 27) allows the live background to provide color information to the system UI. To do this, create a </Boolean> </Boolean> object from a bitmap, drawable, or using three manually selected colors. You can also retrieve this color information. To create a WallpaperColors object, do one of the following: to create a WallpaperColors object using three colors, create an instance of the WallpaperColors class by passing in the primary, secondary, and tertiary colors. The primary color must not be null. To create a WallpaperColors object from a bitmap, call the fromBitmap() method passing the bitmap source as a parameter. To create a WallpaperColors object from a drawable object, call the fromDrawable() method passing the drawable source as a parameter. To retrieve primary, secondary, or tertiary color details from the background, call the following methods: To notify the system of any significant color changes in the live background, call the notifyColorsChanged() method. This method triggers an onComputeColors() lifecycle event where you can provide a new WallpaperColors object. To add a listener for color changes, you can call the addOnColorsChangedListener() method. You can also call the getWallpaperColors() method to retrieve the primary colors of a background. Fingerprint Updates The FingerprintManager class introduced the following error codes: FINGERPRINT_ERROR_LOCKOUT_PERMANENT – The user tried too many times to unlock the device using the fingerprint reader. FINGERPRINT_ERROR_VENDOR: A vendor-specific fingerprint reader error occurred. Encryption updates Some changes have been made to encryption with Android 8.1: new algorithms have been implemented in Conscrypt. The Implementation of Conscrypt is used preferably over the existing Bouncy Castle implementation. New algorithms include: AlgorithmParameters:GCM KeyGenerator:AES KeyGenerator:DESEDE KeyGenerator:HMACMD5 KeyGenerator:HMACSHA1 KeyGenerator:HMACSHA224 KeyGenerator:HMACSHA256 KeyGenerator:HMACSHA384 KeyGenerator:HMACSHA512 SecretKeyFactory:DESEDE Signature:NONEWITHECDSA Cipher.getParameters().getParameterSpec(IvParameterSpec.class) no longer works for algorithms that use GCM. Use getParameterSpec(GCMParameterSpec.class). Many internal Conscrypt classes associated with TLS have been managed. Since developers sometimes access these reflections, shim have been left in place to support previous usage, but some details have changed. For example, sockets were previously of type OpenSSLSocketImpl, but are now of type ConscryptFileDescriptorSocket or ConscryptEngineSocket, both extendIng OpenSSLSocketImpl. SSLSession methods used to throw IllegalArgumentException when passed null reference, now throw NullPointerException. RSA KeyFactory no longer allows keys to be converted from byte arrays larger than the encoded key. Calls to generatePrivate() and generatePublic() that provide a KeySpec where the key structure does not the entire buffer will result in an InvalidKeySpecException. When a socket read is interrupted by the socket to close, Conscrypt is used to return -1 from reading. Reading now throws SocketException. The root CA certificate set has been changed, mostly removing a large number of obsolete certificates, but also removing root certificates for WoSign and StartCom. For more information about this decision, see Google's security blog post, Final Removal of Trust in WoSign and StartCom Certificates. Certificates.