SMAATSDK

REQUISITOS Y DOCUMENTACIÓN DEL

MÓDULO DE POST-PROCESADO EN ANDROID

RELEASE v1.0

Serimag

Índice de contenido

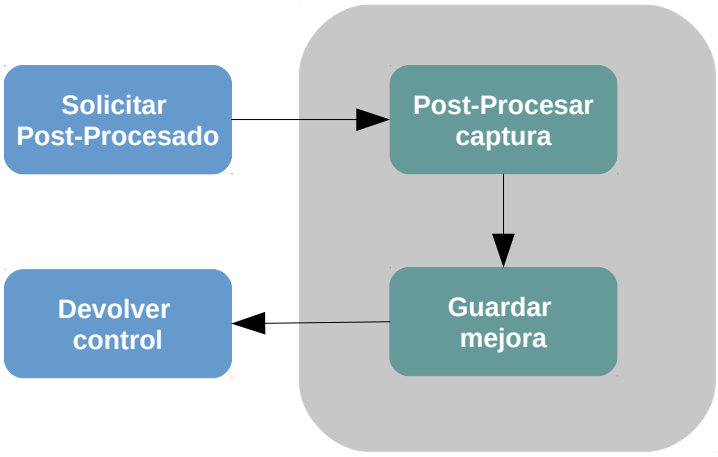
Alcance

Este documento contiene una explicación detallada sobre las funcionalidades y los requisitos del Módulo de Post-Procesado. Para empezar se mostrará un diagrama de funcionamiento general y se explicarán que ficheros se necesitan para el post-procesado y todos los pasos para llevar a cabo su encarpetado. A continuación, se explicarán todas las funciones proporcionadas y sus parámetros. Finalmente, se proporcionará un ejemplo de uso.

Objetivos

Los objetivos de este documento es explicar de manera detallada el funcionamiento y los requisitos del Módulo de Post-Procesado y facilitar su integración en nuevos proyectos o proyectos ya existentes.

Diagrama de funcionamiento general



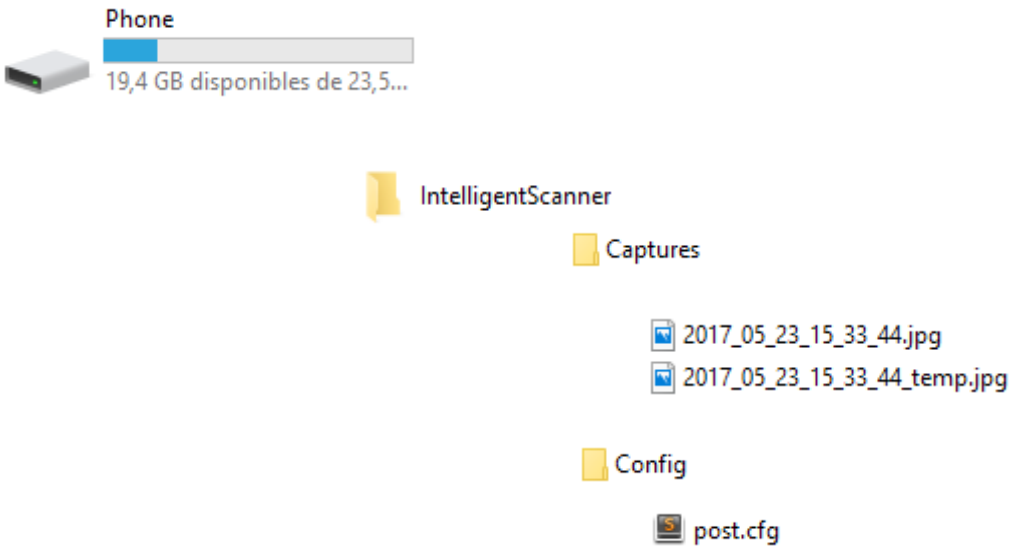
Bloques propios de la aplicación

Bloques encapsulados por el SDK

Bloques propios del motor de post-procesado

Ejemplo de encarpetado y ficheros necesarios para la captura de documentos

En el almacenamiento de la tarjeta de memoria (SD) se crea la carpeta IntelligentScanner que dentro contiene la carpeta Captures donde se guardarán tanto las imágenes a post-procesar como las imágenes post-procesadas. Este módulo puede post-procesar cualquier imagen, pero en caso de querer post-procesar una imagen generada por el motor, se debe activar previamente la función generatePostProcessImage() del Módulo IntelligentScanner, que generará una imagen en la carpeta Captures. Una vez post-procesada la imagen, se sobrescribe la de entrada. Además es necesario crear una carpeta Config donde se colocará el fichero *post.cfg* .



Funciones proporcionadas y parámetros a utilizarlas

Funciones necesarias Estas funciones se encuentran definidas en el fichero PostProcessInterface.java	
PostProcessInterface postProcessInterface = PostProcessInterface.getInstance()	Asigna la instancia del motor de post-procesado. La instancia de este objeto sigue el patrón Singleton.
postProcessInterface.setConfigPath(String configPath)	Fija el path de la carpeta Config, donde se encuentra el fichero de configuración, “post.cfg”.
postProcessInterface.setInputPath(String inputPath)	Fija el path de la carpeta donde se halla la imagen a post-procesar.
postProcessInterface.setOutputPath(String outputPath)	Fija el path de la carpeta donde se guardará la imagen post-procesada.
*Ver funciones opcionales	

String postProcessInterface.run(String documentType, String imageName)	Arrancar el post-procesado de la imagen especificando tipo de documento a post-procesar. Excepciones: • ArgumentException • ConfigFileNotFound • ConfigFileMalformatted • Exception

***Funciones opcionales** Estas funciones se pueden utilizar para cambiar las propiedades de captura.	
void postProcessInterface.setImageQuality(int width, int height, int quality)	Fijar propiedades (ancho, alto, calidad) de la imagen post-procesada. Si no se llama a esta función, se aplicarán los parámetros por defecto. Excepciones: • IllegalArgumentException

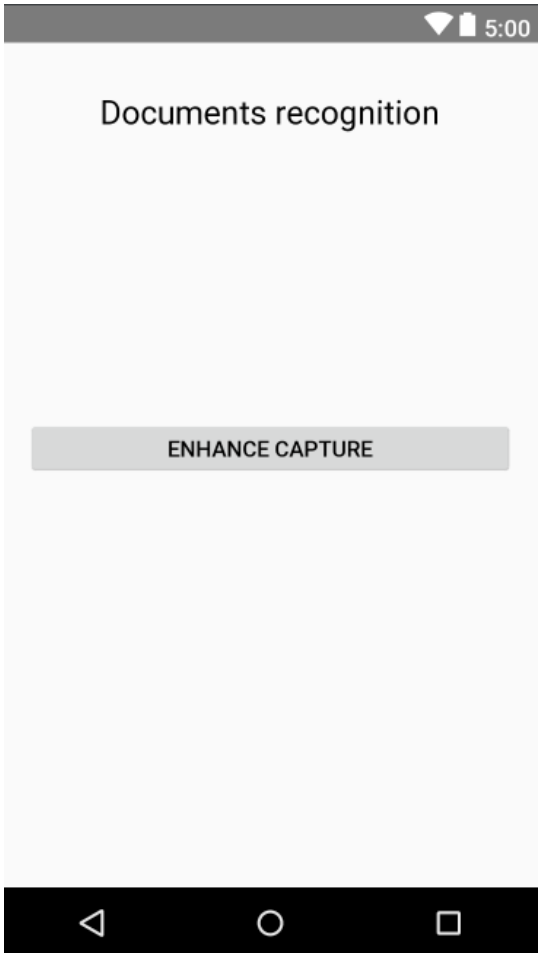
Parámetros a utilizar Parámetros a utilizar en las funciones anteriores	
String configPath	Este parámetro especifica la ruta hasta la carpeta "Config". Es un parámetro OBLIGATORIO para el funcionamiento del SDK.
String inputPath	Este parámetro especifica la ruta hasta la carpeta donde se halla la imagen a post-procesar.
String outputPath	Este parámetro especifica la ruta hasta la carpeta donde se guardará la imagen post-procesada.
String documentType	Este parámetro puede tomar seis valores que corresponden con el nombre de los modelos de documento a capturar: • ES_ID_FRONT • ES_ED_BACK • ES_RESIDENCE_FRONT • ES_RESIDENCE_BACK • INT_PASSPORT • EU_CHECK_FRONT • STRANDARD_CARD
String imageName	Este parámetro contiene el nombre de la imagen a post-procesar alojada en la carpeta Captures almacenada en la memoria SD. Este parámetro debe ser la salida de la función getPostProcessImageName().

int width	Este parámetro especifica el ancho de la captura en píxeles. • Tamaño mínimo permitido: 1 píxel • Tamaño máximo permitido: ∞ píxeles • Tamaño por defecto en ID/RESIDENCE/STANDARD_CARD: 670 píxeles • Tamaño recomendado en ID/RESIDENCE/STANDARD_CARD: 670 píxeles • Tamaño por defecto en INT_PASSPORT: 475 píxeles • Tamaño recomendado en INT_PASSPORT: 475 píxeles • Tamaño por defecto en EU_CHECK: 692 píxeles • Tamaño recomendado en EU_CHECK: 692 píxeles
int height	Este parámetro especifica el ancho de la captura en píxeles. • Tamaño mínimo permitido: 1 píxel • Tamaño máximo permitido: ∞ píxeles • Tamaño por defecto en ID/RESIDENCE/STANDARD_CARD: 425 píxeles • Tamaño recomendado en ID/RESIDENCE/STANDARD_CARD: 425 píxeles • Tamaño por defecto en INT_PASSPORT: 475 píxeles • Tamaño recomendado en INT_PASSPORT: 475 píxeles • Tamaño por defecto en EU_CHECK: 315 píxeles • Tamaño recomendado en EU_CHECK: 315 píxeles
int quality	Este parámetro especifica el porcentaje de calidad de captura: • Porcentaje mínimo permitido: 0 • Porcentaje máximo permitido: 100 • Porcentaje por defecto: 100 • Porcentaje recomendado: 100

Ejemplo de uso

Una vez instalado el AAR ImagePostProcessor, ya podemos empezar a utilizarlo.

Suponiendo que tenemos un botón que nos permite realizar el post-procesado a una imagen de manera asíncrona, en la siguiente imagen se muestra el código a implementar. En primer lugar, en la línea 37 se crea el objeto de la clase que permite ejecutar el post-procesado. En las siguientes tres lineas se finjan las rutas hasta las carpetas donde se encuentran el fichero de configuración, "*post.cfg*", y las capetas de input y output. En este caso tanto el input como el output se encuentran en la misma carpeta llamada Captures. A continuación, en la linea 82 se fijan el alto, el ancho y la calidad de la imagen. Finalmente, en la linea 84 se ejecuta el post-procesado de manera asíncrona esperando el nombre de la imagen mejorada.



```
01:     package serimagmedia.capturaautomatica;
02:
03:     import android.graphics.Color;
04:     import android.os.Bundle;
05:     import android.os.Environment;
06:     import android.support.v7.app.AppCompatActivity;
07:     import android.view.View;
08:     import android.widget.TextView;
09:     import java.io.File;
10:
11:     import automaticdocumentcapturesdk.PostProcessInterface;
12:
13:
14:     public class StartMenu extends AppCompatActivity {
15:
16:         private File path_sd = Environment.getExternalStorageDirectory();
17:         private File path_captures = new File(path_sd.getAbsolutePath() +
18:     "//IntelligentScanner//Captures");
19:         private File path_config = new File(path_sd.getAbsolutePath() +
20:     "//IntelligentScanner//Config");
21:         public static PostProcessInterface postProcessInterface;
22:         static final int PERMISSION_REQUEST = 1;
23:
24:         public StartMenu(){
25:         }
26:
27:         @Override
28:         protected void onCreate(Bundle savedInstanceState) {
29:             super.onCreate(savedInstanceState);
30:             requestPermissions();
31:             setContentView(R.layout.activity_start_menu);
32:
33:     }
```

```
34:     public void initSMAAT_SDK(){
35:
36:         postProcessInterface = PostProcessInterface.getInstance();
37:         postProcessInterface.setConfigPath(path_config.getPath());
38:         postProcessInterface.setInputPath(path_captures.getPath());
39:         postProcessInterface.setOutputPath(path_captures.getPath());
40:
41:      }
42:
43:      public void requestPermissions(){
44:         ActivityCompat.requestPermissions(this, new String[]{
45:
46:                 // Only if you implement the example using external
47:                    //storage (SD)
48:                 Manifest.permission.WRITE_EXTERNAL_STORAGE,
49:
50:         },PERMISSION_REQUEST);
51:      }
52:
53:
54:      public void onRequestPermissionsResult(int requestCode,
55:                         String permissions[], int[] grantResults) {
56:         switch (requestCode) {
57:             case PERMISSION_REQUEST: {
58:           // If request is cancelled, the result arrays are empty.
59:                 if (grantResults.length > 0
60:                         && grantResults[0] ==
61:                               PackageManager.PERMISSION_GRANTED) {
62:
63:            // Initialize SDK only if permissions are granted
64:                      initSMAAT_SDK();
65:
66:                 }
67:                 return;
68:             }
69:         }
70:      }
71:
72:
73:
74:
75:
76:
```

```
77:   public void onClickPost(View v){
78:
79:           Runnable runnable = new Runnable() {
80:               @Override
81:               public void run() {
82:
83:                   postProcessInterface.setImageQuality(670,425,100);
84:                   postProcessInterface.run("ES_ID_BACK",
85:   "2018_04_24_04_34_45_temp.jpg");
86:
87:               }
88:           };
89:           new Thread(runnable).start();
90:
91:       }
92:   }
```