Desarrollo de una aplicación móvil para la digitalización e identificación de eventos en calendarios

G. Coronel, L. Gianinetto, S. Cernotto, C. Martínez, E. Albornoz

sinc(i): Instituto de investigación en señales, sistemas e inteligencia computacional
Facultad de Ingeniería y Ciencias Hídricas, Universidad Nacional del Litoral -CONICET
(*)emalbornoz@sinc.unl.edu.ar

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (sinc.unl.edu.ar)
G. Coronel, L. Gianinetto, S. Cernotto, C. E. Martínez & E. M. Albornoz; "Desarrollo de una aplicación móvil para la digitalización e identificación de eventos en calendarios"
Anales del 8vo. Congreso Nacional de Ingeniería Informática – Sistemas de Información (CONAIISI), 2020.

Resumen *En la actualidad existen una gran variedad de herramientas que permiten administrar citas, tareas y diferentes tipos de eventos a través de un calendario. En general, estos están asociados a direcciones de mail y/o sistemas de notificaciones que proveen un recordatorio flexible y muy útil. Los estudiantes utilizan estas aplicaciones ya que deben administrar clases, turnos de examen, periodos de inscripción, etc. y esta información se publica habitualmente en calendarios en formato papel. Aquí se presenta el desarrollo de un sistema que permite digitalizar esta información y exportarla automáticamente a un calendario digital.*

Palabras clave: digitalización de calendarios, procesamiento de imágenes, aplicación móvil

1. Introducción

A pesar del paso de los años existen prácticas que se mantienen bastante arraigadas en el ecosistema de la Universidad. Quién haya transitado por los pasillos de una Facultad y más aún los estudiantes, seguramente han observado calendarios pegados en paredes y pizarrones con eventos anotados manualmente. Aún hoy en día, al inicio de cada cuatrimestre, es muy común que a los estudiantes se les entreguen distintas versiones de calendarios en papel que ya vienen marcados con diversos colores para identificar a simple vista diferentes eventos, como por ejemplo: fechas de los turnos de exámenes, comienzo y fin del cuatrimestre, fechas de inscripción y feriados, entre otros. Además, cada estudiante lleva su propio registro de fechas importantes, como pueden ser fechas de parciales, entregas de trabajos prácticos, o cualquier otro evento personal; que también marcan sobre el calendario de papel.

En los últimos años, casi todos los estudiantes han podido acceder a dispositivos móviles y adoptaron de forma natural la utilización de calendarios digitales, ya que pueden llevarlo en el teléfono y generan recordatorios de los eventos mediante el envío de alertas o notificaciones. Estos calendarios digitales incorporan cada día más funcionalidades y se integran por ejemplo a las casillas de e-mail[1]. Como ambas formas de registro coexisten, los estudiantes deben cargar manualmente cada evento del calendario en papel en el calendario digital. En este sentido, y para nuestro conocimiento, no existe una herramienta que permita realizar este proceso de forma simple y automática.

En este trabajo se presenta el desarrollo de una aplicación que permite tomar una fotografía de un calendario en papel con los eventos marcados, identificar cada uno de estos, extraerlos, estandarizarlos y generar la migración a un calendario digital. El diseño y desarrollo del sistema se puede dividir en dos grandes partes: una en la que se implementan las rutinas de procesamiento de imágenes, y se acceden a partir de un servicio web que recibe una imagen y devuelve la lista con los eventos detectados; mientras que la segunda parte consiste en el desarrollo de la aplicación móvil con una interfaz agradable que permite capturar la imagen, comunicarse con el servicio, enviar la imagen e interpretar la información obtenida para poder cargarla al calendario.

2. Marco Teórico

En esta sección presentaremos algunos conceptos teóricos de procesamiento digital de imágenes que fueron utilizados en este proyecto.

[1] Algunos de los más conocidos pueden encontrarse en https://www.google.com/intl/es-419/calendar/about/ y https://calendar.yahoo.com.

Operador Laplaciano

El Laplaciano permite resaltar las regiones de una imagen que contiene cambios rápidos de intensidad (bordes) [1]. Se define el Laplaciano de una función 2D como:

$$\nabla^2 f = 4z_5 - (z_2 + z_4 z_6 + z_8)$$

Considerando una aproximación con N_4 es posible calcularlo con una convolución con una máscara como:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Es posible utilizar la varianza de este resultado para estimar el nivel de desenfoque de la imagen. Si el Laplaciano de la imagen presenta una varianza grande, existe una amplia variedad de respuestas (bordes y partes homogéneas), y esto es representativo de una imagen enfocada. Por el contrario, si la varianza es baja, estaríamos en presencia de una imagen con muy pocos o sin bordes, lo cual es indicio de desenfoque.

Transformada de Hough

Este método permite identificar el contenido de puntos colineales en una imagen, que pueden o no ser parte de una recta o un segmento de recta [1]. Esta propuesta considera a cada punto (x, y) de la imagen en término de sus coordenadas polares (ρ, θ), donde ρ representa la distancia entre el origen de coordenadas y el punto (x, y), mientras que θ es el ángulo del vector <x,y>. Cada punto genera, en el espacio transformado, una sinusoidal según $\rho = x\cos\{\theta\} + y\sin\{\theta\}$. Al transformar dos puntos de la imagen, las sinusoidales correspondientes se intersectan en un punto (ρ_R, θ_R) qué son los parámetros que definen la recta soporte que pasa por ambos puntos en la imagen (ρ_R es la distancia perpendicular a la recta soporte y θ_R el ángulo de la perpendicular). Haciendo el cómputo de todos los puntos es posible determinar las rectas principales de la imagen[2], buscado los acumuladores (puntos de intersección) más grandes en el espacio transformado a partir de los cuales se puede obtener su orientación en la imagen.

Operaciones morfológicas binarias

Estos métodos tienen su origen en la teoría de conjuntos y se implementan a través de operaciones unión e intersección con pequeñas imágenes (elementos estructurantes, EE) definidas según la aplicación lo requiera [2]. El EE suele ser una pequeña imagen (3x3 px) que se desplaza sobre la imagen original, se realiza una operación entre las imágenes solapadas y se deposita el resultado habitualmente donde coincide el centro del EE con la imagen. Aquí se trabaja con:

Erosión binaria: si el EE no está completamente contenido en la región de la imagen que se está analizando, el pixel que coincide con el centro del EE es eliminado en la imagen resultado [2].

Dilatación binaria: si la intersección entre el EE y la región de la imagen que se está analizando es distinta de del conjunto vacío, el pixel que coincide con el centro del EE es activado en la imagen resultado [2].

Método Otsu para umbralización

Este método genera un umbral en intensidad que permite separar los píxeles en dos clases: primer plano y fondo. El umbral se determina minimizando la varianza de intensidades dentro de la clases o, de manera equivalente, maximizando la varianza entre clases [3]. La varianza dentro de la clases se define como la suma de las dos varianzas, σ_0^2 y σ_1^2, multiplicadas por pesos asociados, w_0 y w_1

$$\sigma_w^2(t) = w_0(t)\sigma_0^2(t) + w_1(t)\sigma_1^2(t)$$

Los pesos son las probabilidades de las clases separadas por un umbral *t*, calculados a partir del histograma de la imagen, donde *L* es la cantidad de clases en los cuales se agrupan las intensidades

$$w_0 = \sum_{i=0}^{t-1} p(i)$$

$$w_1 = \sum_{i=t}^{L-1} p(i)$$

Como ya se mencionó, minimizar la varianza dentro de las clases es equivalente a maximizar la varianza entre clases

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t) = w_0(\mu_0(t) - \mu_T)^2 + w_1(\mu_1(t) - \mu_T)^2$$
$$= w_0(t)w_1(t)\left[\mu_0(t) - \mu_1(t)\right]^2$$

expresándose en términos de las probabilidades *w* y medias μ de las clases

$$\mu_0(t) = \frac{\sum_{i=0}^{t-1} ip(i)}{w_0(t)}$$

[2] Se hace la salvedad de que no necesariamente es una recta, puede ser la dirección en que coexistan más puntos colineales.

$$\mu_1(t) = \frac{\sum_{i=t}^{L-1} ip(i)}{w_1(t)}$$

$$\mu_T = \sum_{i=0}^{L-1} ip(i)$$

El algoritmo se hace iterativo y converge al encontrar la máxima varianza entre clases.

Algoritmo de medias no locales

Este algoritmo fue introducido por Buades [4] y ha demostrado ser un método simple pero eficaz para eliminar ruido en imágenes. Para eliminar el ruido en un píxel dado, utiliza un promedio ponderado de otros píxeles en la imagen ruidosa. En particular, dada una imagen ruidosa $u = u_i$, la imagen sin ruido $\widehat{u} = \widehat{u}_i$ en el píxel *i,* se calcula mediante la ecuación

$$\hat{u}_i = \frac{\sum_j w_{ij} u_j}{\sum_j w_{ij}}$$

donde w_{ij} es un peso asignado a los píxeles *i* y *j*, calculado sobre una vecindad geométrica centrada en los mismos dado de la siguiente manera

$$w_{ij} = e^{\frac{1}{h^2}\|V_i - V_j\|^2}$$

Aquí, V_i y V_j representan las vecindades centradas en los píxeles *i* y *j* correspondientes a secciones de imagen de tamaño $k \times k$, $\|V_i - V_j\|$ es la norma euclídea de la diferencia de ambos vectores y *h* un parámetro de suavizado. Así, los pesos cumplen las siguientes condiciones

$$0 < w_{ij} < 1 \quad \text{y} \quad \sum_j w_{ij} = 1$$

y permiten que los píxeles con vecindades similares reciban un mayor peso en comparación a los píxeles cuyas vecindades se ven diferentes. De esta manera, se hace uso explícito del hecho de que en la mayoría de las imágenes aparecen patrones repetitivos.

Filtro de mediana

Este filtro pertenece al grupo de los filtros no lineales de orden, es bastante útil para eliminar ruidos del tipo gaussiano; sal y pimienta; y su combinación. Se define a partir de una ventana que se desplaza por la imagen y en cada paso se calcula un resultado que se asigna al pixel que coincide con el centro de la ventana. El proceso consiste en ordenar los valores de intensidad de los píxeles y seleccionar el valor que está en el centro de esa lista [1].

3. Materiales y métodos

Definiciones y requerimientos del sistema

La funcionalidad básica del sistema puede resumirse como una serie de tareas: recibir una foto de un calendario ya sea desde la memoria del dispositivo o capturando una nueva con la cámara, identificar el formato del calendario, reconocer las fechas marcadas, generar una lista de eventos especificando sus días y meses, y articular la integración con la aplicación de calendario.

Las tareas específicas se agruparon en 3 bloques generales:

1. Pre-procesamiento de la imagen: se deben considerar y resolver situaciones donde, el calendario no está centrado en la foto, la foto fue capturada con cualquier distancia, se tienen distintos niveles de iluminación, la foto está desenfocada y la información no es nítida, existen rotaciones no deseadas en la foto, etc.
2. Identificación y reconocimiento: se debe reconocer el formato del calendario para identificar días y meses, se deben identificar las marcas que corresponden a eventos y reconocer si existen eventos de más de un día de duración.
3. Aplicación móvil: debe implementar funcionalidades para permitir al usuario tomar una foto o bien cargar una existente desde la memoria del dispositivo, enviar la misma a un servicio web que la procese y recibir los eventos enviados como respuesta para finalmente exportarlos a un calendario digital.

Para la implementación del sistema, atendiendo a las definiciones previas, se utilizan las siguientes tecnologías:

- Python + OpenCV [5, 6]: para el desarrollo del procesamiento de imágenes que será alojada en el servidor.
- Python + Django [7]: el servidor fue desarrollado funciona en python y se utilizó Django como framework, mientras que se encuentra alojado gratuitamente para pruebas en Heroku[3].
- Android SDK + Java [8]: la app móvil es una aplicación nativa Android por lo que el desarrollo se codificó en Java.

[3] Disponible en www.heroku.com.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (sinc.unl.edu.ar)
G. Coronel, L. Gianinetto, S. Cernotto, C. E. Martínez & E. M. Albornoz; "Desarrollo de una aplicación móvil para la digitalización e identificación de eventos en calendarios"
Anales del 8vo. Congreso Nacional de Ingeniería Informática – Sistemas de Información (CONAIISI), 2020.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (sinc.unl.edu.ar)
G. Coronel, L. Gianinetto, S. Cernotto, C. E. Martínez & E. M. Albornoz; "Desarrollo de una aplicación móvil para la digitalización e identificación de eventos en calendarios"
Anales del 8vo. Congreso Nacional de Ingeniería Informática – Sistemas de Información (CONAIISI), 2020.

- Retrofit[4]: para la comunicación desde la aplicación con la API del servidor. Permite programar la conexión HTTP para el envío de la imagen, y manejar/administrar la respuesta del servidor.

Se establecieron algunas pautas respecto de los modelos de calendarios permitidos, a saber: los meses deberán estar ubicados 4 por fila y en orden creciente de izquierda a derecha con su correspondiente nombre, las semanas podrán comenzar en lunes o domingo y deben estar las iniciales de los días identificando las columnas. La aplicación será lo suficientemente robusta para aceptar un calendario tanto en idioma español como en inglés. En la Figura 1 se presenta un modelo de calendario posible.

Figura 1. Modelo de calendario

Para las pruebas del sistema se generó un pequeño conjunto de imágenes con diferentes formatos. Los calendarios deberán tener un fondo uniforme, el calendario no estará necesariamente centrado en la imagen, puede presentar rotaciones, la imagen puede contener un fondo distinto al color del calendario, puede tener diferentes zoom y puede estar borrosa.

Diseño y esquema general

La app móvil tiene una serie de layout principales que se presentan en la Figura 2 y se esquematiza el flujo de la información. En la primer pantalla se solicita ingresar las etiquetas (colores + título) que representan los eventos en el calendario, por ejemplo, los eventos marcados en azul representan días de exámenes, los de color verde son los días de inscripción, etc. La siguiente pantalla permite capturar una fotografía o cargar una desde la memoria.

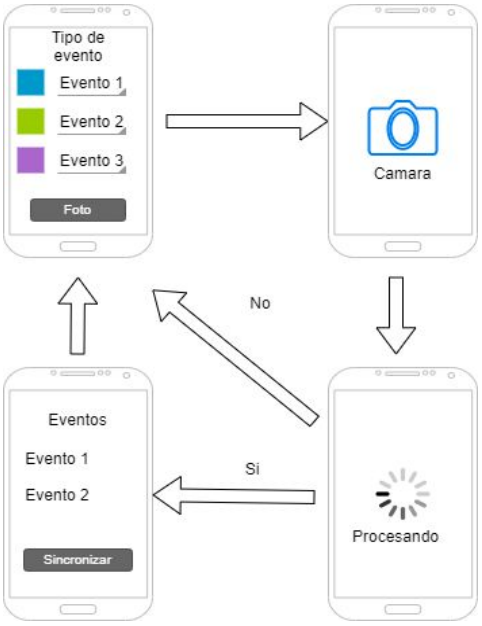

Figura 2. Diagrama de la app móvil

En la siguiente etapa la imagen es enviada al servidor para ser procesada y, si todo va bien, devuelve los eventos reconocidos; mientras que, si la imagen no cumple con los requisitos mínimos, se pide al usuario que repita el proceso previo. Finalmente, la última pantalla muestra la lista de eventos y permite sincronizarlos automáticamente con el calendario digital.

Desarrollo

Inicialmente, el procesamiento de la imagen se realizó en una versión desktop que, una vez testeada, se migró a una app móvil cliente-servidor. La primer etapa consiste en la validación de la imagen para poder continuar con el procesamiento, y se considera:

- que el nivel de brillo de la imagen esté dentro de los parámetros esperados (utilizando el nivel de gris medio).
- que la foto no esté desenfocada (utilizando el cálculo del Laplaciano).
- que la resolución del calendario dentro de la imagen sea apropiado (depende de la resolución y el zoom de la foto).
- que la calidad de la imagen permita identificar la información importante (p.e. los caracteres).

Luego de evaluar la validez de la imagen que elige el usuario, se procesa la imagen. El diagrama general del procesamiento de la información se presenta en la Figura 3. La primera tarea es obtener los bloques de los 12 meses y asignarles su etiqueta correspondiente. Los modelos de

[4] Disponible en https://square.github.io/retrofit/

calendarios considerados presentan la información de los meses con una distancia significativa (ver Fig. 1).

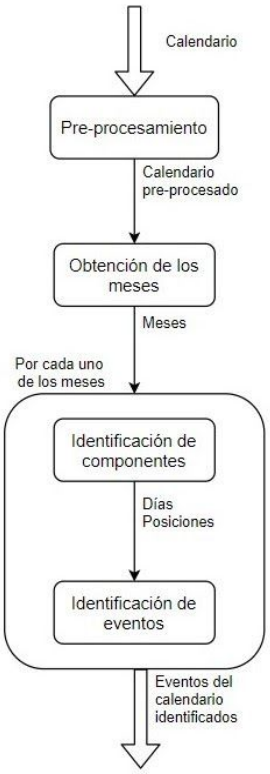

Figura 3. Diagrama de bloques de procesos

La imagen se procesa para corregir rotaciones (utilizando la transformada de Hough) y escalados. Luego, esta imagen se segmenta, umbraliza (mediante Otsu), recorta y mejora con morfología matemática y filtrado. A este resultado se le aplican operaciones de dilatación para uniformizar los bloques de los meses. Se descartan los bloques que no sean rectangulares y que su tamaño este entre (1/12) del tamaño del calendario (límite superior) y (1/12)*0.3 (límite inferior). Finalmente, se obtienen los bloques de los 12 meses.

Sobre cada bloque, se determinarán los días que integran ese mes y si la semana comienza en domingo o lunes. Esto se hace extrayendo la cabecera del bloque y reconociendo el nombre de los días a través de la biblioteca *pytesseract* [9]. Se consideran situaciones donde los días son expresados por su inicial, por dos o tres letras iniciales. Además, se procesa morfológicamente el bloque para obtener una grilla con la localización de cada día en particular. Luego, se procede a identificar eventos para cada mes, a partir de la información de color mediante segmentación.

Una vez resuelto el procesamiento general de la imagen, se procede a implementar las funcionalidades necesarias para que la app funcione en el esquema cliente-servidor. Este proceso se agrupa en 3 etapas:

1. *Definiciones y permisos*: se desarrollaron funciones para chequear si la app cuenta con los permisos para funcionar correctamente y, en caso de que no, se pide autorización al usuario. Los permisos corresponden al uso de internet, escritura/lectura del almacenamiento y escritura/lectura del calendario. En la pantalla inicial, definir las etiquetas y colores de los eventos es obligatorio, no se puede continuar si no se define.

2. *Aplicación Web*: esta aplicación permite recibir una imagen vía HTTP POST, realizar todo el procesamiento de la imagen y luego, devolver un archivo JSON.

3. *Administración de resultados*: se desarrollaron las funciones para recibir la información de eventos desde el servicio web, y manejar los errores. Según la documentación oficial de Android Studio [10], las inserciones y actualizaciones de eventos del calendario deben realizarse en un subproceso asíncrono para trasladar la acción a segundo plano, para lo que se desarrolló una subclase AsyncTask. De esta manera, la información recibida se combina con las etiquetas almacenadas y se crean los eventos finales.

4. Resultados y discusiones

Las pruebas iniciales se realizaron con calendarios bastante “ideales”, editados con herramientas gráficas, con el fin de comprobar el funcionamiento general y la coherencia de los resultados. Luego, continuamos realizando pruebas con calendarios "reales", tomando imágenes con la cámara de un teléfono móvil. Tanto para la versión standalone como para la versión web, la metodología de evaluación fue similar. En la Figura 4 y 5 se presenta un ejemplo de cada tipo de calendario respectivamente. Aquí podemos encontrar imágenes reales que presenten algún tipo de ruido, lo cual se podrá solucionar aplicando filtro de medias no locales o filtro de medianas, lo que permitirá limpiar la imagen y seguir su proceso. Otra de las diferencias que se podrá encontrar entre ambas imágenes es la nitidez y el enfoque que presenta cada una, lo cual puede ser detectado a través del operador Laplaciano, mencionado en secciones anteriores, el cual permitirá decidir si la imagen supera el umbral estimado y continuar el proceso, o de lo contrario descartar la misma.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (sinc.unl.edu.ar)
G. Coronel, L. Gianinetto, S. Cernotto, C. E. Martínez & E. M. Albornoz; "Desarrollo de una aplicación móvil para la digitalización e identificación de eventos en calendarios"
Anales del 8vo. Congreso Nacional de Ingeniería Informática – Sistemas de Información (CONAIISI), 2020.

Figura 4. Calendario "ideal"

Figura 5. Calendario "real"

Luego de las primeras etapas de procesamiento se logra obtener los bloques dilatados (Fig. 6) y en el paso posterior se identifican los meses (en color verde, Fig. 7). Para cada bloque de mes es posible identificar la posición de los días, el resultado puede verse en la Figura 8.

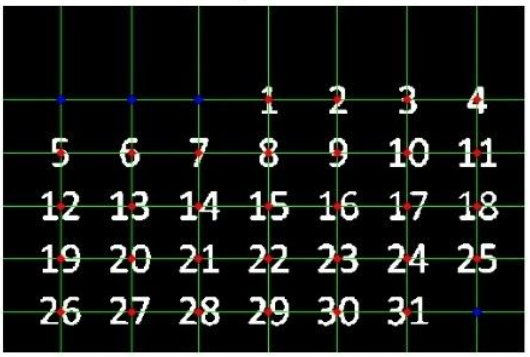

Figura 8. Identificación de la posición de los días

En lo que respecta a las pruebas cliente-servidor, se probó su funcionamiento utilizando Postman para poder enviar un HTTP POST con la imagen al servidor. El resultado es una cadena JSON que está compuesto por el set (mes, color y días para cada evento) o por un mensaje de error (ver Figura 9).

Para ilustrar el funcionamiento de la aplicación final, se presentan capturas de pantalla de la aplicación ejecutándose en un dispositivo móvil. En la Figura 10, se

Figura 6. Filtrado de bloques

Figura 7. Detección de los meses

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (sinc.unl.edu.ar)
G. Coronel, L. Gianinetto, S. Cernotto, C. E. Martínez & E. M. Albornoz: "Desarrollo de una aplicación móvil para la digitalización e identificación de eventos en calendarios"
Anales del 8vo. Congreso Nacional de Ingeniería Informática – Sistemas de Información (CONAIISI), 2020.

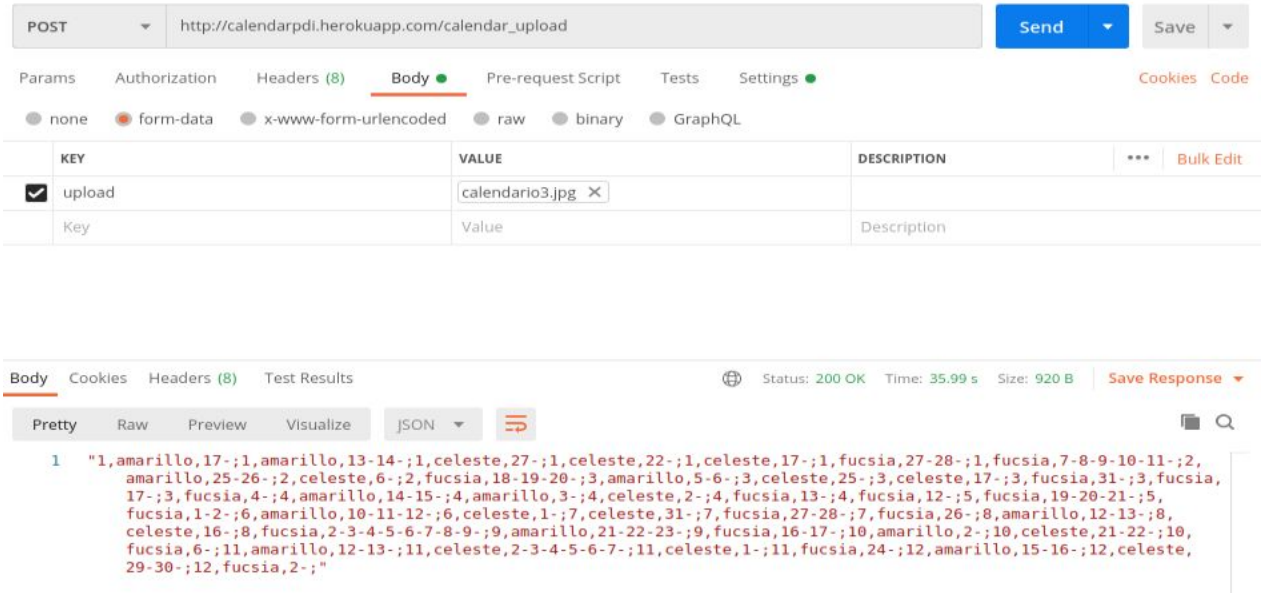

Figura 9: Resultado servicio web

presenta la pantalla inicial de la aplicación donde se solicitan los permisos necesarios y luego, la configuración de los eventos. Dicha configuración es representada a través de etiquetas las cuales serán ingresadas por el usuario para identificar los títulos de cada evento que se corresponderá a cada color. En la figura 10, podemos apreciar un ejemplo de las etiquetas que proporciona como sugerencia la aplicación. En la pantalla siguiente se accede a la carga/captura de la imagen. Luego, se procesa (esto demanda algunos segundos en la comunicación y procesamiento en el servidor) y la aplicación nos informará al respecto tal como se muestra en la Figura 11. Una vez finalizado este proceso, en caso de que haya sido exitoso, la aplicación mostrará los eventos que ha obtenido desde el servidor y se dispondrán a través de una lista desplegada en pantalla. Aquí se detalla la información obtenida de la imagen cargada, enumerando para cada conjunto de eventos, el mes al que pertenece, su etiqueta correspondiente al evento designado para dicho color, y los días correspondientes al evento marcado. Luego, el usuario tendrá la opción de sincronizarlos al calendario de Google y la aplicación mostrará la información que se observa en la Figura 13. En caso contrario, se le informará al usuario que ha ocurrido un error (Figura 14).

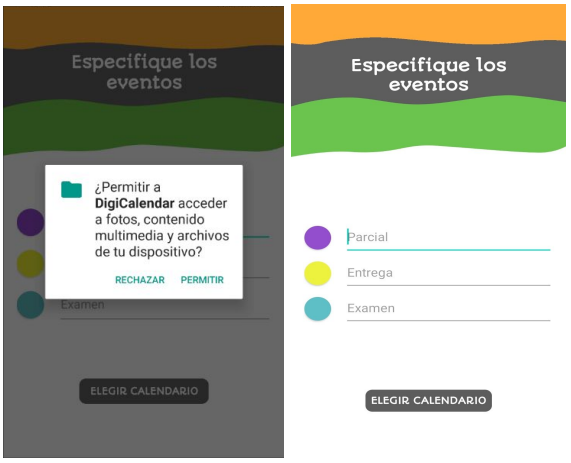

Figura 10: Pantalla de inicio

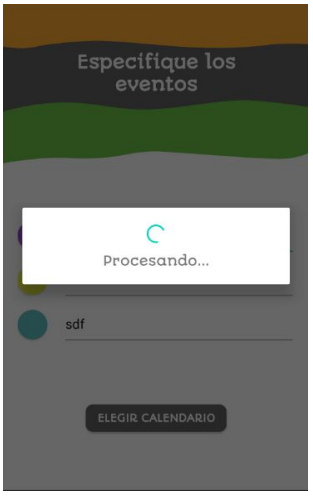

Figura 11: Pantalla de espera

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (sinc.unl.edu.ar)
G. Coronel, L. Gianinetto, S. Cernotto, C. E. Martínez & E. M. Albornoz; "Desarrollo de una aplicación móvil para la digitalización e identificación de eventos en calendarios"
Anales del 8vo. Congreso Nacional de Ingeniería Informática – Sistemas de Información (CONAIISI), 2020.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (sinc.unl.edu.ar)
G. Coronel, L. Gianinetto, S. Cernotto, C. E. Martínez & E. M. Albornoz; "Desarrollo de una aplicación móvil para la digitalización e identificación de eventos en calendarios"
Anales del 8vo. Congreso Nacional de Ingeniería Informática – Sistemas de Información (CONAIISI), 2020.

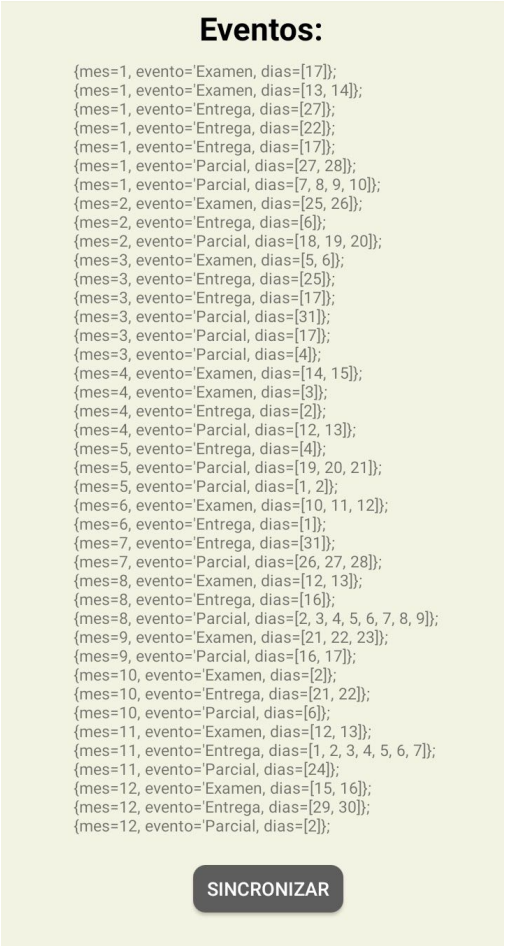

Figura 12: Eventos obtenidos

Figura 13: Pantalla para importar eventos

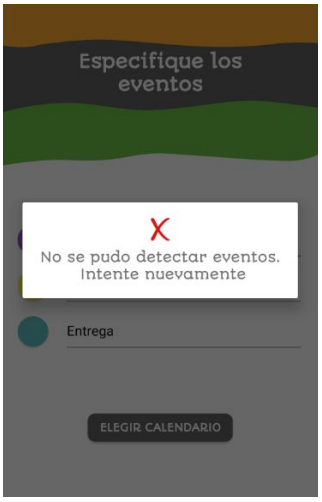

Figura 14: Posible mensaje de error

5. Conclusiones y Trabajos futuros

En este trabajo se presentó el diseño y desarrollo de un sistema que permite la identificación automática de eventos en calendarios impresos para su posterior sincronización con herramientas de calendarios digitales. Se ha podido alcanzar una implementación que contempla desde la adquisición de la imagen hasta la sincronización con el calendario de Google, y se realizó utilizando técnicas de procesamiento digital de imágenes con lo que se lograron resultados muy satisfactorios. Se implementaron métodos para identificar, resolver y/o notificar diferentes situaciones de adquisición de imágenes: poca iluminación, desenfoque, etc. En una segunda etapa fue posible desarrollar una app Android que funciona en un esquema cliente-servidor de forma consistente con la aplicación standalone. Esta aplicación está disponible de forma gratuita en https://github.com/gcoron/CalendarDigitizerApp.

En esta versión, se debe tener en cuenta que la fotografía capturada debe tener buenas condiciones para que los eventos sean detectados de forma correcta. Los colores que se utilizan para resaltar los distintos eventos, un calendario con el papel arrugado o si existe otro elemento en la foto además del calendario se generan "ruidos" en el proceso y pueden pueden llevar a una detección incorrecta del calendario, de los meses, de cada evento y los días.

Como trabajo futuro se podría implementar métodos que permitan corregir/reducir el desenfoque que pueda introducir el usuario al momento de tomar la fotografía y no solo detectarlo; de esta forma se podría extender el uso incluso a aquellas imágenes que estén borrosas. Además,

se pretende dotar a la aplicación de funciones para interacción con los usuarios, a fin de abordar correcciones o mejoras que vayan detectando.

Referencias

[1] Gonzalez, R. C., & Woods, R. E. (2007). Image processing. Digital image processing, 2, 1.

[2] Serra, J., & Soille, P. (Eds.). (2012). Mathematical morphology and its applications to image processing (Vol. 2). Springer Science & Business Media.

[3] Otsu, N. (1979). A threshold selection method from gray-level histograms. IEEE transactions on systems, man, and cybernetics, 9(1), 62-66.

[4] Buades, A., Coll, B., & Morel, J. M. (2011). Non-local means denoising. Image Processing On Line, 1, 208-212.

[5] Minichino, Joe, and Joseph Howse. Learning OpenCV 3 Computer Vision with Python. Packt Publishing Ltd, 2015.

[6] Laganière, Robert. OpenCV 3 Computer Vision Application Programming Cookbook. Packt Publishing Ltd, 2017.

[7] Dauzon, S., Bendoraitis, A., & Ravindran, A. (2016). Django: Web Development with Python. Packt Publishing Ltd.

[8] Studio, A. (2017). Android Studio. The Official IDE for Android.

[9] Matthias Lee. Pytesseract . Accedido en 19-06-2020 a url- https://pypi.org/project/pytesseract/. 2020.

[10] Developer Android Studio. “Calendar provider overview”. Accedido en 18/08/2020 a url https://developer.android.com/guide/topics/providers/calendar-provider.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (sinc.unl.edu.ar)
G. Coronel, L. Gianinetto, S. Cernotto, C. E. Martínez & E. M. Albornoz; "Desarrollo de una aplicación móvil para la digitalización e identificación de eventos en calendarios"
Anales del 8vo. Congreso Nacional de Ingeniería Informática – Sistemas de Información (CONAIISI), 2020.