Benemérita Universidad
Autónoma de Puebla

Facultad de Ciencias de la Computación

Análisis y Reconocimiento de Imágenes de Pinturas en un Museo

Para obtener el título de

Licenciado en la Ingeniería en Tecnologías de la Información

Presenta:
Alfredo Loeza Macías

Asesor:
Mario Anzures García

Asesora:
María Luz Adolfina Sánchez Gálvez

Septiembre 2021

Índice General

1. Introducción

En la presente investigación se realiza una aplicación para el procesamiento y reconocimiento de imágenes, que permita la identificación de obras de arte famosas a partir de fotografías tomadas por el usuario en cualquier museo del mundo. Debido a que en ocasiones las personas no conocen a los autores de dichas obras o en su caso son obras poco reconocidas a nivel mundial. Además, esto puede ayudar a las personas a tener un acercamiento con la cultura; ya que tendrían la información más a la mano y en ese momento, es decir, en su dispositivo móvil y mientras se encuentran visitando el museo.

Para la realización de dicha tarea se hará uso de Google Cloud Vision [1], la cual tiene una Interfaz de Programación de Aplicaciones (en inglés *Application Program Interface*, API), que ofrece modelos de aprendizaje automático previamente entrenados y eficaces para ayudar al usuario a entender la imagen seleccionada con la predicción más precisa. Esta API etiqueta el contenido de las imágenes otorgándoles una puntación, la cual determina el nivel de confianza de la etiqueta, y las clasifica en miles de categorías ya definidas, además de reconocer rostros, detectar objetos, leer texto impreso y a mano en diferentes idiomas, entre otras.

Una vez obtenidos los resultados que contiene la imagen, se procesarán y filtrarán dichos datos tales como las etiquetas y la puntuación para convertirlos en información útil (relacionada con la fotografía obtenida de la obra de arte del museo) para el usuario.

Por tanto, en este trabajo de tesis se plantea una aplicación de reconocimiento de información de pinturas que se encuentran en los museos, para que el usuario pueda comprender perfectamente quién hizo la pintura y sobre cual tema se orienta.

Por otro lado, el desarrollo de esta aplicación estará basado en (como ya se mencionó anteriormente) Google Cloud Vision para el reconocimiento de las imágenes. Se utilizará Node Js [2], que es un *framework* que utiliza JavaScript [3] del lado del servidor para hacer el procesamiento de los datos obtenidos por Google Cloud Vision. Por último, se utilizará el kit de herramientas Flutter [4], para crear una interfaz nativa de alta calidad.

Actualmente, el reconocimiento de imágenes [5] implica la creación de una red neuronal [6] que procesa individualmente cada pixel de una imagen. Esta red necesita ser alimentada con muchas imágenes, para que la red “aprenda” a reconocer imágenes similares. Esto requiere una gran inversión en recursos y tiempo para la creación de dicha red, por lo que el reconocimiento de imágenes es una tarea complicada para cualquier persona.

Afortunadamente, Google cuenta con una plataforma llamada Google Cloud, la cual ofrece múltiples soluciones para diferentes necesidades tecnológicas tales como el almacenamiento, macrodatos [7], estadísticas, aprendizaje automático [8], entre otros. Al utilizar la infraestructura de esta gran compañía, la plataforma obtiene datos rápidos, confiables y escalables, además transforma estos datos en información valiosa para el usuario.

Por estos motivos está siendo utilizada por múltiples corporaciones en todo el mundo, creando aplicaciones que ayudan a transformar la enseñanza, la investigación, el aprendizaje y el desarrollo.

También, se puede mencionar que en la mayoría de los museos se encuentra una simple tarjeta o cuadro de texto junto a la obra de arte que proporciona una pequeña descripción de la misma. Lo cual hace que en muchas ocasiones nos quedemos con dudas sobre dicha obra o incluso sobre el autor, la época en que fue realizada y la importancia que la misma tiene. Claro, que hoy en día se puede realizar una búsqueda en internet sobre la obra de arte en la que estemos interesados, pero seguramente, en caso de ser una obra de arte poco conocida o que su creador sea un artista sin renombre, se tendrá como resultado poca información o incluso la misma puede ser confusa.

Por tanto, en este trabajo de tesis se plantea una aplicación que haga uso de tecnologías actuales en el contexto de reconocimiento de imágenes, inteligencia artificial [9], desarrollo web [10] e interfaces de usuario [11], que sean robustas, seguras, usables [12] y responsivas [13], para suministrar información suficiente y adecuada sobre una obra de arte vista en un museo, en el momento que la están apreciando desde el dispositivo móvil que las personas llevan.

Como se mencionó anteriormente, esta aplicación puede ser un servicio que ofrece el museo o una app que cualquier usuario puede descargar y tenerla siempre que la necesite o requiera conocer más sobre cierta obra de arte.

Finalmente, como se puede apreciar este tema es bastante complejo y a la vez importante para cualquier persona que vaya a un museo, e incluso para los propios museos, ya que la aplicación puede ofrecerse como un servicio a los visitantes de dicho museo como un valor agregado del mismo.

El documento contiene: La sección 2 presenta brevemente los fundamentos teóricos del trabajo de tesis. La sección 3 describe las aplicaciones similares destacando las funciones comunes, así como los pros y los contras de las mismas. La sección 4 explica el desarrollo de la aplicación de análisis y reconocimiento de pinturas en un museo. La sección 5 muestra los resultados obtenidos. Y finalmente, la sección 6 proporciona las conclusiones y el trabajo futuro.

2. Marco Teórico

La web en la actualidad es una potente herramienta como espacio de información que día a día se exigen innovadoras funcionalidades para cubrir las complejas exigencias de los usuarios. El desarrollo web es una parte fundamental para poder cubrir estas exigencias. Las herramientas que se utilizan para crear estos complejos sitios cada día evolucionan para dar paso a nuevas tecnologías de desarrollo más potentes, robustas y que incluso se pueden adaptar a un proyecto web específico.

La base de todo proyecto web en cuanto a infraestructura es HTML5 este te provee los elementos estructurales y de maquinación que también te permite incluir otro tipo de tecnología como son las hojas de estilo en cascada CSS3 que te ayuda a dar diseño a la estructura HTML esto para que la web tenga una mejor apariencia visual ante el usuario. Para que la página no sea simple texto plano, tenga interacción con el usuario y sea dinámica se usa la tecnología JavaScript.

En conjunto con todas estas tecnologías mencionadas se pueden desarrollar páginas web informativas estas siempre se van a ejecutar en el navegador del usuario debido a esto dichas paginas estarán limitadas en cuanto a funcionalidades. Por otro lado, existen páginas más complejas que necesitan ejecutarse en un servidor, para esto se usan lenguajes del lado del servidor como PHP. Este tipo de sitios ven la necesidad de incluir bases de datos para que el usuario pueda almacenar y visualizar información, para esto se usan gestores base de datos tales como MYSQL. Además, se hace uso de Brackets para editar los archivos de la

aplicación, Codeigniter para desarrollar la aplicación con PHP y Bootstrap para generar plantillas responsivas intuitivas y usables.

2.1. HTML

HTML5 provee básicamente tres características: estructura, estilo y funcionalidad. Es considerado el producto de la combinación de HTML, CSS y Javascript. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de HTML5, es simplemente un conjunto de nuevas funciones disponibles para el desarrollo de aplicaciones web, que se agregan a las capacidades existentes que encontramos en HTML4. Está especialmente diseñado para mejorar el lenguaje con un soporte mucho mejor para la comunicación multimedia y del servidor, lo que facilita mucho el trabajo de un desarrollador web.

HTML5 no es una nueva versión de HTML4 en comparación con cuando se lanzan nuevas versiones de software. Comprende un conjunto completo de pequeñas adiciones al estándar web existente. Actualmente, cada navegador implementa algunas de estas características, pero no todas. Eventualmente, aunque esperamos que todos los navegadores tengan un conjunto similar de características, lo que significa que no existe tal cosa como ser "compatible con HTML5".

2.2. CSS

CSS es un lenguaje que trabaja junto con HTML5 para proveer estilos visuales a los elementos del documento, como tamaño, color, fondo, bordes, etc. En un intento

por reducir el uso de código JavaScript y para estandarizar funciones populares fue que se creó esta tecnología.

CSS3 no solo cubre diseño y estilos web sino también forma y movimiento. La especificación de CSS3 es presentada en módulos que permiten a la tecnología proveer una especificación estándar por cada aspecto involucrado en la presentación visual del documento.

La especificación de HTML5 fue desarrollada considerando CSS a cargo del diseño. Debido a esta consideración, la integración entre HTML y CSS es ahora vital para el desarrollo web y esta es la razón por la que cada vez que mencionamos HTML5 también estamos haciendo referencia a CSS3, aunque oficialmente se trate de dos tecnologías completamente separadas.

2.3. JavaScript

JavaScript es un lenguaje interpretado usado para múltiples propósitos, pero solo considerado como un complemento hasta ahora. Una de las innovaciones que ayudó a cambiar el modo en que vemos Javascript fue el desarrollo de nuevos motores de interpretación, creados para acelerar el procesamiento de código. La clave de los motores más exitosos fue transformar el código Javascript en código máquina para lograr velocidades de ejecución similares a aquellas encontradas en aplicaciones de escritorio. Esta mejorada capacidad permitió superar viejas limitaciones de rendimiento y confirmar el lenguaje Javascript como la mejor opción para la web.

Las interfaces de programación de aplicaciones (APIs) fueron incorporadas por defecto en cada navegador para asistir al lenguaje en funciones elementales. Estas nuevas APIs (como Web Storage, Canvas, y otras) son interfaces para librerías incluidas en navegadores.

La idea es hacer disponible poderosas funciones a través de técnicas de programación sencillas y estándares, expandiendo el alcance del lenguaje y facilitando la creación de programas útiles para la web.

2.4. Inteligencia Artificial

La inteligencia artificial (IA) es un conjunto de algoritmos diseñados para crear maquinas o dispositivos que imiten las capacidades del ser humano. Existen diferentes tipos de inteligencia artificial, entre las que se encuentran: los sistemas que piensan como humanos, como las redes neuronales artificiales que automatizan ciertas actividades como el aprendizaje y la toma de decisiones; los sistemas que actúan como humanos, como los robots, que son máquinas que realizan actividades de una manera similar a como las haría un ser humano; sistemas que piensan racionalmente, como los sistemas expertos, que tratan de que las computadoras piensen, razonen y actúen por sí mismas; sistemas que actúan racionalmente, como los agentes inteligentes, que intentan imitar el comportamiento del ser humano.

Actualmente se pueden encontrar dispositivos y máquinas que utilizan la inteligencia artificial en múltiples áreas de la industria. Por ejemplo, la creación de robots y máquinas que ayuden en la producción en masa de productos. Así mismo, la inteligencia artificial se encuentra en el sector del transporte aéreo, al controlar el

tráfico de un aeropuerto, rutas y sistemas de los aviones, así como en el transporte terrestre, en coches autónomos. También se puede apreciar la inteligencia artificial en múltiples videojuegos al crear entornos virtuales.

2.5. Reconocimiento de Imágenes

El reconocimiento de imágenes es la capacidad de las computadoras de reconocer objetos, personas, lugares, texto a partir de imágenes. Estas computadoras utilizan una cámara para tomar la imagen y un software con inteligencia artificial para realizar el reconocimiento.

El reconocimiento de objetos es una tarea sencilla para la gran mayoría de los seres humanos, sin embargo, para las computadoras, realizar esta actividad requiere una gran cantidad de recursos.

Generalmente, las computadoras requieren 4 pasos para el reconocimiento correcto de las imágenes.

- El primero paso es obtener las características de cada pixel de la imagen. Cada pixel está formado por un conjunto de números y el rango de estos números representan la profundidad del color, es decir, el número máximo de colores que puede ser usado en la imagen.
- El segundo paso es etiquetar imágenes. Una vez extraídas las características de las imágenes, estas se etiquetan como una categoría según el objeto que presentan. Entre más imágenes sean utilizadas en el conjunto de entrenamiento mejor será el modelo predictivo.

- El tercer paso es entrenar el modelo para que categorice imágenes. Al modelo se le ingresan las imágenes previamente etiquetadas y pasan por un filtro (red neuronal) para que entren en la categoría deseada.
- El cuarto y último paso es reconocer o predecir una nueva imagen dentro de las categorías. Una vez que el modelo sea entrenado puede ser usado para reconocer o predecir dentro de qué categoría entra una nueva imagen. Introducir una nueva imagen representa pasar por todo el proceso de nuevo, desde la extracción de características de cada pixel a su clasificación mediante la red neuronal.

En este proyecto se hace uso de Google Cloud Vision, el cual, sigue un proceso similar al expresado anteriormente para la identificación de imágenes y se explica más a detalle a continuación.

2.6. Google Cloud Vision

Google Cloud Vision es una herramienta desarrollada por Google que permite hacer un análisis y procesamiento detallado de imágenes. Esta herramienta utiliza millones de imágenes ya indexadas que crece día con día. Estas imágenes se pueden considerar como el conjunto de entrenamiento que ayuda a clasificar las nuevas imágenes, es decir, que cuando una nueva imagen necesita ser analizada, Vision la clasifica y obtiene los metadatos de imágenes similares a la nueva. Por ejemplo, si se quiere analizar la imagen de un auto, Vision intentará encontrar imágenes similares a un auto, y después de ese conjunto de imágenes encontrará algunas con los mismos colores, formas y tamaños y presentará los metadatos de las imágenes encontradas como si fueran de la nueva imagen ingresada.

Esta herramienta puede ser accedida mediante su API, la cual permite que los desarrolladores puedan hacer uso de toda la capacidad de procesamiento sin necesidad de gastar muchos recursos.

2.7. Node Js

Este *framework* de desarrollo web fue creado sobre el motor de JavaScript de Google Chrome (V8 *Engine*). Es un ambiente de desarrollo multiplataforma del lado del servidor y sus aplicaciones son creadas en JavaScript, las cuales pueden ser ejecutadas en Mac Os, Windows y Linux. Así mismo, también ofrece una gran variedad de módulos de JavaScript, lo que hace que el desarrollo sea más simple.

Node Js proporciona una E/S asíncrona controlada por eventos en los que se pueden escribir múltiples tipos de aplicaciones, tales como: aplicaciones web, de línea de comandos, chat en tiempo real, servidor API REST, entre otras.

La utilización de JavaScript del lado del servidor permite una ejecución más ágil, lo cual en este tipo de proyectos es importante, debido a la cantidad de información y recursos que se requieren. De la misma manera, la forma de utilizar Node.js en este proyecto será como un servidor API REST, el cual es consumido desde una aplicación móvil.

2.8. Express

Express es el framework más popular de Node Js que sirve para simplificar sus APIs y añadir nuevas características, así mismo hace más fácil organizar la funcionalidad de aplicaciones con la implementación de Middlewares y rutas. Express está basado en un middleware de Node Js llamado Connect, el cual utiliza

el módulo http. Una de las ventajas de utilizar Express es que es fácil de configurar, permite configurar un middleware para el manejo de errores, es sencillo de conectar con bases de datos como MySQL, Mongo DB, Redis, además, permite crear un servidor API REST, por lo tanto, es ideal para la utilización en este proyecto.

2.9. Flutter

En la actualidad, la creación de aplicaciones móviles de forma rápida es de gran importancia, debido a la cantidad de aplicaciones ya existentes y las que se generan todos los días.

Flutter, que es un framework de desarrollo de aplicaciones móviles de código abierto de Google, permite la creación de aplicaciones nativas de alta calidad de múltiples plataformas y con un excelente rendimiento. Flutter cuenta con una función llamada *Hot Reload*, la cual permite visualizar los cambios en tiempo real, lo que hace que el tiempo de desarrollo baje considerablemente.

Además, esta herramienta ofrece algunas otras ventajas, entre las que se encuentran: la creación de diseños fáciles y adaptables a las diferentes dimensiones de los dispositivos con un tiempo de renderización muy rápido, rendimiento nativo muy superior al que ofrecen herramientas para crear aplicaciones hibridas.

3. Aplicaciones Similares

El estado del arte es una modalidad de la investigación documental que permite el estudio del conocimiento acumulado (escrito en textos) dentro de un área específica. Sus orígenes se remontan a los años ochenta, época en la que se utilizaba como herramienta para compilar y sistematizar información especialmente el área de ciencias sociales, sin embargo, en la medida en que estos estudios se realizaron con el fin de hacer balances sobre las tendencias de investigación y como punto de partida para la toma de decisiones, el estado del arte se posicionó como una modalidad de investigación.

3.1. AiPoly

Esta es una aplicación de reconocimiento de objetos y colores que ayuda a las personas ciegas, con discapacidad visual o daltónicas [14]. Esta aplicación utiliza cámara incorporada al teléfono y solo apuntando al objeto de interés y seleccionando uno de los botones de la parte inferior activa la inteligencia artificial. A continuación, se presentan algunas de las características más importantes de Aipoly:

- Identifica más de 2000 especies de plantas y animales.
- Lee texto en múltiples lenguajes.
- Reconoce más de mil platillos de comida.
- Se puede acceder a función de voz, en la que se puede escuchar todo lo que se reconoce.
- Puede funcionar sin conexión a internet.

- Activa automáticamente el flash de la cámara, en caso de detectar oscuridad.

3.2. CalorieMama

Esta es una aplicación de reconocimiento de comida, en la que, con solo tomar una foto de un platillo, se puede obtener información nutrimental de dicho plato [15]. Se basa en las últimas innovaciones en aprendizaje profundo y tecnología de clasificación de imágenes para identificar de forma rápida y precisa los alimentos.

Identifica miles de categoría de alimentos y mejora constantemente a medida que nuevas imágenes son agregadas.

La aplicación funciona de la siguiente forma:

- Al descargar la aplicación, pregunta el peso y la altura actual del usuario, así como su género y su meta de pérdida de peso.
- La aplicación determina la cantidad de calorías que el usuario debe consumir para alcanzar dicha meta de peso.
- El usuario puede seleccionar agregar elementos de desayuno, comida o cena, mientras observa la cantidad de calorías recomendadas para cada comida.
- También registra agua, ejercicio y pasos realizados por el usuario.

3.3. ScreenShop

ScreenShop es una aplicación para el reconocimiento y búsqueda de ropa, en la cual, el usuario toma una foto de un conjunto y busca entre millones de productos similares a cada prenda en él. Se pueden utilizar filtros de colores para ver artículos

en el color de tu elección [16]. El usuario puede mezclar el producto elegido a una nueva categoría, así como buscar artículos similares a los que le hayan gustado. Por último, puede explorar y comprar a miles de marcas y tiendas mundialmente reconocidas.

3.4. LeafSnap

LeafSnap es una guía electrónica desarrollada en conjunto por investigadores de las universidades de Columbia, Maryland y el Instituto Smithsoniano [17]. La aplicación web, utiliza la detección de imágenes con inteligencia artificial para identificar especies de árboles, a partir de fotografías de sus hojas. También contiene imágenes en alta resolución de hojas, frutas, flores, semillas y cortezas de árboles.

Actualmente cuenta con dos versiones: la original, la cual incluye árboles que se encuentran en el noreste de Estados Unidos y Canadá, y la versión del Reino Unido, que además de incluir árboles de esta nación, cuenta con información e imágenes proporcionadas por el Museo de Historia Natural de Londres.

3.5. Flow Powered

Flow Powered es una aplicación de realidad aumentada creada por Amazon, que permite obtener información de las cosas que se encuentran a nuestro alrededor [18]. Puede identificar millones de artículos entre los que se encuentran:

- Portadas de libros
- Videojuegos
- DVD y CD

- Cajas de cereal

También puede decodificar códigos QR, números de teléfono, URL, direcciones de correo electrónico, tarjetas de presentación, entre otros.

3.6. CamFind

Esta es una aplicación que permite a los usuarios buscar en el mundo físico, ofreciendo un motor de búsqueda visual [19]. Se toma una foto de un objeto y la aplicación genera resultados prácticos para el usuario, tales como imágenes, videos y opciones de compra.

Los usuarios tienen la posibilidad de guardar sus búsquedas y resultados en sus perfiles y poder compartirlos con cualquier persona de manera fácil y eficaz. Así mismo se pueden seguir a otros usuarios para descubrir nuevos productos.

3.7. TapTapSee

TapTapSee es una aplicación móvil diseñada específicamente para personas ciegas o con discapacidad visual que funciona con la API de reconocimiento de imágenes “CloudSight” [20]. La aplicación utiliza la cámara del dispositivo y funciones de voz para tomar una foto y un video y reconocer los objetos en voz alta. Para tomar una foto se debe tocar dos veces en el lado derecho de la pantalla y para tomar un video, tocar dos veces en el lado izquierdo de la pantalla. TapTapSee puede analizar de forma precisa objetos en dos o tres dimensiones y en cualquier ángulo en cuestión de segundos. La función de voz, dice lo identificado en voz alta.

3.8. Vivino

Vivino es una comunidad online cuyo objetivo es la venta y recomendación de vinos [21]. Cuenta con un catálogo de más de 3 millones de vinos disponibles y los usuarios pueden valorar, comentar y recomendar vinos de forma sencilla. Además, ofrece un sistema de reconocimiento de vinos, mediante una foto de la etiqueta de la botella o de la carta del restaurante, la aplicación brinda una recomendación del mejor vino para cada ocasión.

3.9. Conclusiones

Las aplicaciones analizadas en los puntos anteriores comparten muchas similitudes, entre las que se encuentran, la detección de objetos que satisfacen una necesidad del usuario, agregando ciertas funciones extras para dar una mejor experiencia de uso tal como la posibilidad de guardar los resultados obtenidos y funciones de voz que dicen en voz alta el objeto detectado.

Sin embargo, ninguna de estas aplicaciones explora la detección y reconocimiento de obras de arte, que es el principal trabajo de investigación expuesto en esta tesis.

4. Desarrollo

El desarrollo del proyecto estuvo basado en la metodología SCRUM [22], la cual es una metodología de desarrollo ágil de aplicaciones en la que se consideran varias iteraciones para lograr un producto final. El propósito de cada iteración es que se complete una nueva versión del producto, con una nueva funcionalidad, característica o mejora. Para la culminación de este proyecto, se consideraron tres iteraciones.

4.1. Primera iteración

En la primera iteración del proyecto se cimientan las bases para el desarrollo del proyecto. Se investigaron diferentes aplicaciones que pudieran ser de ayuda para el uso de las herramientas.

Se hicieron pruebas de detección de objetos en la herramienta Google Cloud Vision, en la cual, se carga la imagen de manera directa a la página de demostración y la herramienta procesa la imagen y proporciona los resultados. Los resultados que nos proporciona incluyen: la detección de marcas, expresiones faciales, objetos, propiedades del color, etiquetas y resultados web.

En la primera prueba, se cargó a la herramienta Google Cloud Vision una imagen genérica de un sendero y los resultados que se obtuvieron se pueden observar a continuación:

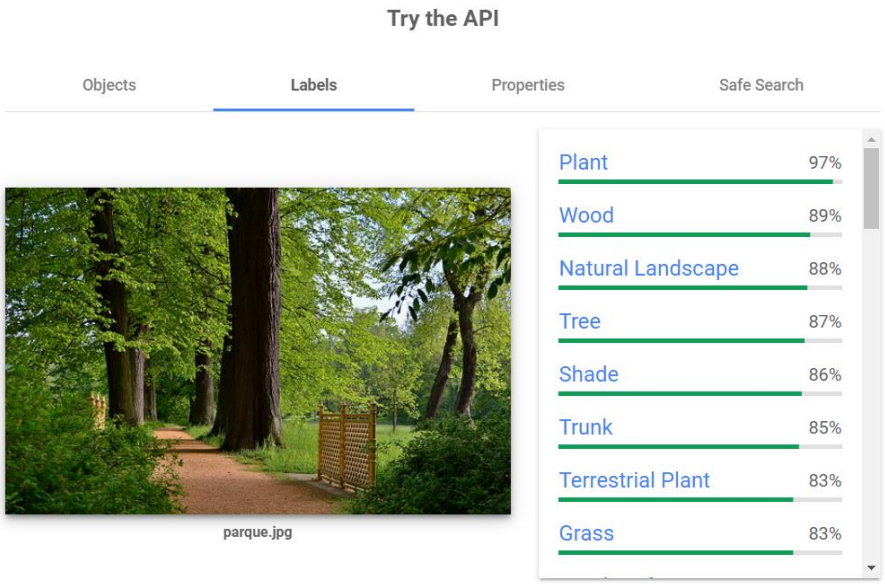

Figura 1. Prueba de detección de etiquetas con Google Cloud Vision.

En la figura 1 se muestra la pestaña de etiquetas, en la cual se puede observar que los resultados obtenidos al utilizar una imagen de un sendero de un bosque fueron: plantas y bosques como los resultados más probables, así como árboles, sombra y pasto como resultados altamente posibles.

Los resultados web son los más importantes para el desarrollo de este proyecto, ya que serán los que determinen las obras de arte.

Se configuró el entorno de desarrollo:

- Descarga e instalación de Node.js
- Descarga e instalación del SDK de Google
- Descarga de módulos de JavaScript: Express, que es una infraestructura para el desarrollo web; nodemon, para visualizar

cambios sin reiniciar el servidor; body-parser, para controlar los parámetros de las peticiones.

- Descarga e instalación de las librerías cliente de Google Cloud para Node.js.

Una vez que se configuró el entorno de desarrollo en Node.js se estableció la conexión de nuestro proyecto con la API de Google Cloud Vision. Para esto se hace uso de una cuenta de Google para activar el uso de la herramienta Vision. Se siguen los siguientes pasos para la configuración de la conexión de la API con el proyecto:

1. Se crea un proyecto dentro de Google Cloud Platform. En dicho proyecto se organizan todos los recursos de Google Cloud y consta de colaboradores, API's habilitadas, herramientas de supervisión datos de facturación y controles de autenticación y acceso.
2. Se habilita la facturación. Esto es para pagar por los recursos utilizados. Dependiendo de las peticiones realizadas a la API, serán los recursos utilizados y por lo tanto el cobro efectuado.
3. Se habilita la API de Google Cloud Vision. Google Cloud Platform cuenta con múltiples servicios disponibles a través de diferentes API's, las cuales están desactivadas por defecto, por lo tanto, se deben de activar aquellas que se utilizarán, en este caso Google Cloud Vision.
4. Se configura la autenticación. Para utilizar cualquier API se requiere estar autenticado, para esto se utiliza una cuenta de servicio, ya que

proporciona las credenciales para aplicaciones, y no para usuarios finales.

5. Se usa la cuenta de servicio dentro del entorno de Node.js, esto es creando una nueva variable de entorno para Google Cloud Platform.

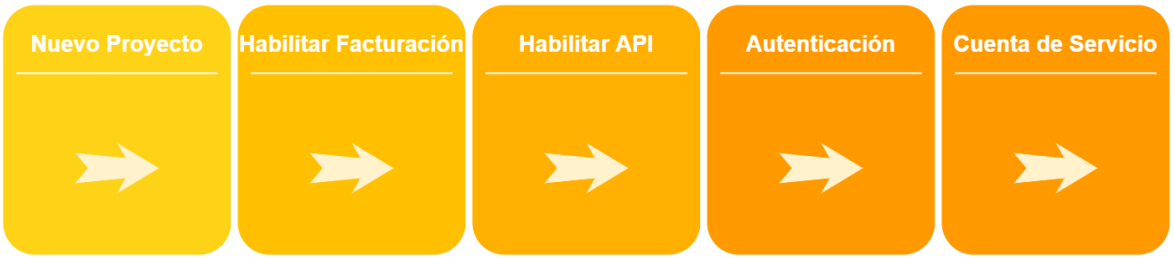

Figura 2. Proceso de configuración de Google Cloud Vision.

4.2. Segunda iteración

Para cargar la imagen al servidor, es necesario crear una ruta, la cual será la ruta del *endpoint* para realizar la petición desde la aplicación móvil. Dentro de esta ruta, se realiza la carga de la imagen, así como su almacenamiento dentro de un directorio de nuestro proyecto. Dicho directorio es llamado *"uploads"*.

En caso de que la carga de la imagen se haya dado de forma exitosa, proseguirá con el siguiente proceso de reconocimiento de la imagen, en caso contrario, regresará un mensaje de error, el cual describe el problema ocurrido.

Una vez cargada la imagen correctamente, el siguiente paso es hacer la petición a la herramienta Google Cloud Vision para su reconocimiento. Para esto se crea una función la cual es mandada a llamar después de la carga correcta de la imagen.

Dicha función utilizará una promesa (*Promise*), la cual representa la terminación o el fracaso de una operación asíncrona.

La petición a la herramienta tiene como parámetro la imagen guardada en el directorio "*Uploads"* y se utiliza la función "*WebDetection"*, la cual, obtendrá los resultados web relacionados con la imagen ingresada y los regresará al servidor Node.js. En caso de que la herramienta no encuentre ningún resultado, el servidor mandará un mensaje al cliente (la aplicación móvil) con el motivo del error.

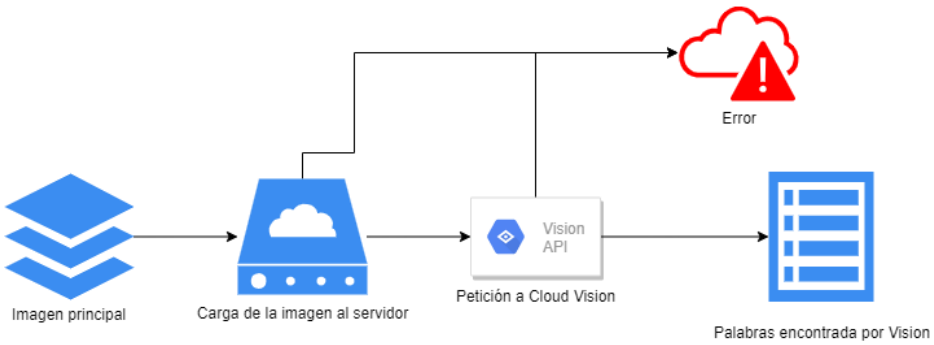

Figura 3. Proceso de la segunda iteración.

4.3. Tercera iteración

Los resultados web obtenidos en la iteración anterior son filtrados de acuerdo a su puntuación. Esta puntuación es otorgada por Google Cloud Vision y determina el nivel de confianza del resultado web en la imagen.

Google Cloud Vision encuentra múltiples resultados de cada imagen, sin embargo, no todos los resultados son relevantes para este proyecto. Por lo tanto, se analiza el puntaje de cada entidad web y se seleccionan solo aquellas que tengan

un *Score* mayor a 0.70. Las entidades web que cumplan con el requisito anterior serán guardas en un arreglo llamado “Palabras”.

Por ejemplo, al hacer las pruebas con una imagen de un automóvil Volkswagen, los resultados web obtenidos fueron Volkswagen Beetle, Volkswagen, Volkswagen Type 2, Van, Car con su respectivo score. Tomando en cuenta su score, los términos que serían agregados al arreglo son todos excepto por la palabra “Car”.

Figura 4. Imagen utilizada como ejemplo.

```
5 Pages with matching images retrieved
Url   : http://www.photos-public-domain.com/2011/01/07/old-volkswagen-bug-and-van/
Url   : http://pix-hd.com/old+volkswagen+van+for+sale
...

2 Full Matches found:
Url   : http://www.photos-public-domain.com/wp-content/uploads/2011/01/old-vw-bug-and-van.jpg
Url   : http://www.wbwagen.com/media/old-volkswagen-bug-and-van-picture-free-photograph-photos-public_s_6

4 Partial Matches found:
Url   : http://www.photos-public-domain.com/wp-content/uploads/2011/01/old-vw-bug-and-van.jpg
Url   : http://www.wbwagen.com/media/old-vw-bug-and-vanjpg_s_ac343d7f041b5f8d.jpg
...

5 Web entities found:
Score      : 5.35028934479
Description: Volkswagen Beetle
Score      : 1.43998003006
Description: Volkswagen
Score      : 0.828279972076
Description: Volkswagen Type 2
Score      : 0.75271999836
Description: Van
Score      : 0.690039992332
Description: Car
```

Figura 5. Resultados web obtenidos a partir del análisis de la figura 4.

Este arreglo será devuelto a la aplicación móvil y serán desplegadas las palabras en forma de lista para que el usuario obtenga los conceptos relacionados con la imagen que capturó.

5. Resultados

El resultado, principal, es una aplicación móvil para el análisis, reconocimiento y recuperación de Información de una obra de arte en un museo. A continuación, se muestra el funcionamiento final de la aplicación móvil:

- Al abrir la aplicación se muestra una pantalla con una figura y dos botones, para cargar una imagen (véase la figura 6).

Figura 6. Pantalla de inicio de la aplicación móvil.

- Al hacer clic en el botón principal, se abre la cámara del dispositivo para capturar una fotografía (véase la figura 7).

Figura 7. Cámara del dispositivo para tomar una fotografía.

- En el otro caso, al hacer clic en el botón secundario se abre el administrador de archivos del dispositivo para elegir la imagen que se cargará en el servidor (véase la figura 8).
- Se da clic en el botón con la flecha para subir la imagen al servidor.
- En caso de querer cambiar la imagen o fotografía, con dar clic en el botón con X se eliminan los elementos capturados y se puede volver a realizar el proceso (véase la figura 9).

Figura 8. Selecciona imagen del dispositivo.

Figura 9. Subir imagen seleccionada.

- La imagen se sube al servidor Node Js y a su vez es enviada como parámetro en la petición a la API de Google Cloud Vision, que analiza dicha imagen, regresando los resultados encontrados (véase la figura 10).
- El usuario elige uno de los resultados obtenidos y la aplicación realiza una petición al servidor que se encargara de consumir la API de Wikipedia para obtener información relevante. Esta información es filtrada de manera que se pueda deducir sí se trata de una obra de arte. En caso de obtener más de un resultado por la API de Wikipedia se le retornaran dichas opciones al usuario, para que pueda elegir entre esas opciones (véase la figura 11). En caso de que solo se obtenga un resultado de la API de Wikipedia, al usuario se le mostrara la información más importante de dicha obra de arte tales como: autor, fecha, descripción, etcétera (véase la figura 12). En el caso de que lo buscado no sea una obra de arte se le mostrara al usuario un mensaje de error (véase la figura 13).

Figura 10. Resultados obtenidos junto con su puntaje.

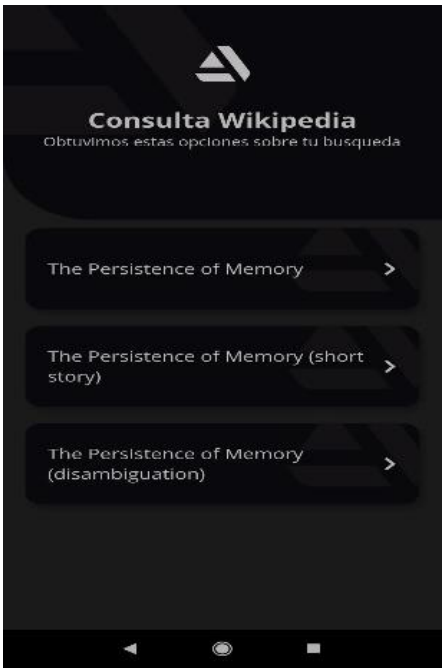

Figura 11. Opciones obtenidas de Wikipedia.

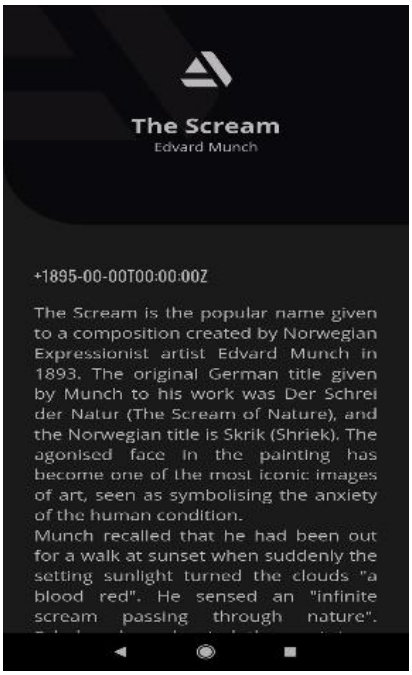

Figura 12. Información relevante de la obra de arte.

Figura 13. Mensaje indicando que no es una obra de arte.

6. Conclusiones

En este trabajo de tesis se realizó el análisis y reconocimiento de imágenes. Obteniendo, principalmente, tres excelentes resultados: 1) Una aplicación móvil que analiza y reconoce una fotografía tomada con el móvil de una obra de arte en un museo; obteniendo los datos del nombre del autor, año de creación y descripción de la obra. 2) La integración de diferentes plataformas en una única aplicación móvil, que simplifica y agiliza el proceso de análisis y reconocimiento de la imagen. Las plataformas integradas en nuestra aplicación móvil fueron: Google Cloud Vision, que compara la imagen ingresada con cientos de miles que tiene ya indexadas y categorizadas; Wikipedia, que proporciona una gran enciclopedia digital para recuperar información de nuestro interés y de una manera fácil a través de una API; y Google Cloud Platform, que es un conjunto de servicios en la nube ejecutados en la misma infraestructura que Google usa internamente para sus productos de usuario final. 3) La publicación “Aplicación Movil para el Análisis, Reconocimiento y Recuperación de Información de Obras de Arte de un Museo” en la revista indexada Pistas Educativas, vol. 42 (137), pp. 1045-1061, 2020, que muestra la calidad del trabajo realizado. El trabajo futuro se centrará en mejorar la optimización en el proceso de análisis y reconocimiento de la imagen capturada con el dispositivo móvil.

7. Bibliografía

[1] Vision Google Cloud. https://cloud.google.com/vision/?hl=es, accedido el 10 de enero de 2020.
[2] Node Js. https://www.ibm.com/developerworks/ssa/opensource/library/os-nodejs/index.html, accedido el 10 de enero de 2020.
[3] JavaScript. https://developer.mozilla.org/es/docs/Web/JavaScript, accedido el 10 de enero de 2020.
[4] Flutter. https://flutter-es.io/, accedido el 10 de enero de 2020.
[5] R. Gonzalez and R. Woods, *Digital image processing*, 4th ed. 1992.
[6] D. Matich, *Redes Neuronales: Conceptos Básicos y Aplicaciones*, pp. 4-5, 2001
[7] B. Schmarzo, *Big Data*.1st ed. 2013.
[8] "¿Tratando de entender PaaS?", *Oracle.com*. [Online]. https://www.oracle.com/mx/artificial-intelligence/what-is-machine-learning.html, accedido el 12 de enero de 2020.
[9] R. López de Mántaras Badia and P. Meseguer González, *Inteligencia artificial*. 2017.
[10] "Primeros pasos en la Web", *Documentación web de MDN*. [Online]. https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web, accedido el 12 de enero de 2020.
[11] "Interfaz de usuario - EcuRed", *Ecured.cu*. [Online]. https://www.ecured.cu/Interfaz_de_usuario, accedido el 12 de enero de 2020.
[12] D. Norman, *Emotion and design: Attractive things work better,* pp. 36-42. 2007.
[13] J. Mora, "Mosaic | Diseño de Interfaces Multimedia: Una web responsiva", *Mosaic.uoc.edu*, 2014. [Online]. https://mosaic.uoc.edu/2014/09/30/diseno-de-interfaces-multimedia-una-web-responsiva/, accedido el 13 de enero de 2020.
[14] "V7 Aipoly", *Aipoly.com*. [Online]. https://www.aipoly.com/, accedido el 17 de enero de 2020.
[15] "Calorie Mama Food AI - Food Image Recognition and Calorie Counter using Deep Learning", *Caloriemama.ai*. [Online]. https://www.caloriemama.ai/, accedido el 17 de enero de 2020.
[16] "Screenshop", *Screenshopit.com*. [Online]. https://www.screenshopit.com/, accedido el 17 de enero de 2020.
[17] "Home | Leafsnap: An Electronic Field Guide", *Leafsnap.com*. [Online]. http://leafsnap.com/, accedido el 17 de enero de 2020.
[18] *Amazon.com*. [Online]. https://www.amazon.com/-/es/dp/B008G318PE, accedido el 17 de enero de 2020.
[19] "CamFind - Search the Physical World™", *Camfindapp.com*. [Online]. Disponible en: https://camfindapp.com/, accedido el 23 de febrero de 2020.
[20] "TapTapSee - Blind and Visually Impaired Assistive Technology - powered by CloudSight.ai Image Recognition API", *Taptapseeapp.com*. [Online]. Disponible en: https://taptapseeapp.com/, accedido el 23 de febrero de 2020.
[21] A. Garcia, "Vivino, la aplicación que te ayuda a elegir el mejor vino para cada momento", *La Vanguardia*, 2016. [Online]. Disponible en: https://www.lavanguardia.com/tecnologia/20160312/40382591445/vivino-app-elegir-vino.html, accedido el 23 de febrero de 2020.
[22] Scrum. https://www.scrum.org/resources/blog/que-es-scrum, accedido el 15 de enero de 2020.

8. Anexo 1: Manual de Usuario

8.1. Instalación

Para instalar la aplicación en un dispositivo móvil, se deben de seguir los siguientes pasos.

- Se debe habilitar la opción de instalación de aplicaciones desde fuentes desconocidas. Esto se realiza de la siguiente manera:
    - En el apartado de configuración del equipo se debe de dar click en las siguientes opciones:

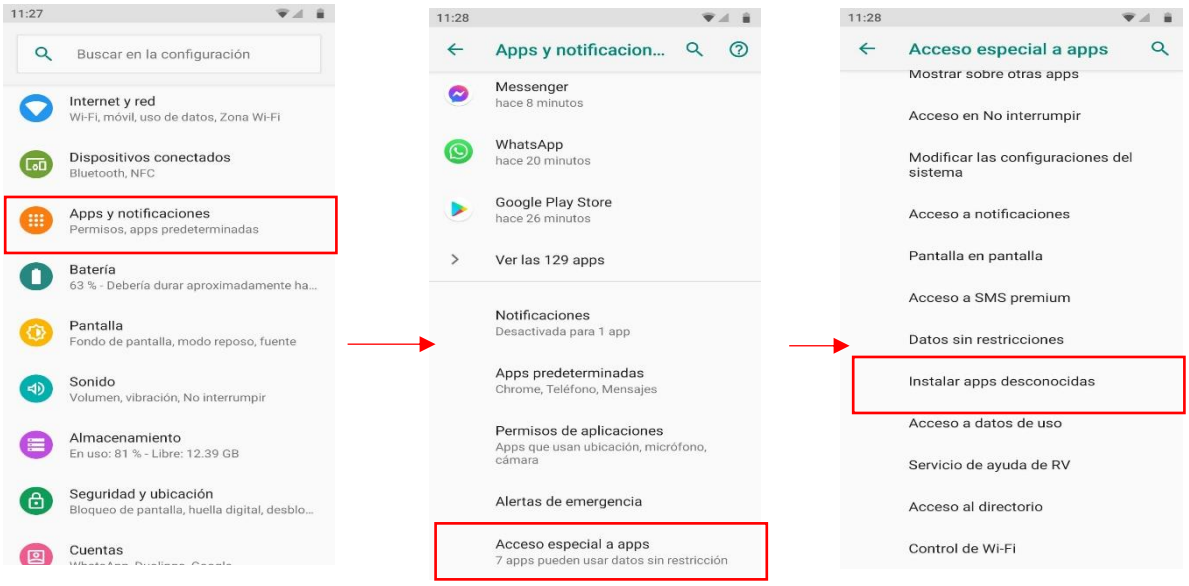

Figura 14, 15 y 16. Ruta para acceder a la opción Instalar apps desconocidas.

- Se habilita la opción “Confiar en esta fuente”.

Figura 17. Habilita la opción que permite instalar aplicaciones desde fuentes desconocidas.

- Se debe descargar el APK de la aplicación, que es un paquete el cual contiene dicha aplicación, además de todo lo necesario para su instalación.
- Al término de la descarga, se abre el archivo y se da clic en la opción instalar.

8.2. Uso de la aplicación

Al abrir la aplicación, aparecen dos botones. El principal que se encuentra en el centro de la pantalla, es para abrir la aplicación de la cámara del dispositivo y poder una foto. El segundo botón, que se encuentra en la parte inferior derecha de la pantalla, es para seleccionar una imagen previamente almacenada en el dispositivo.

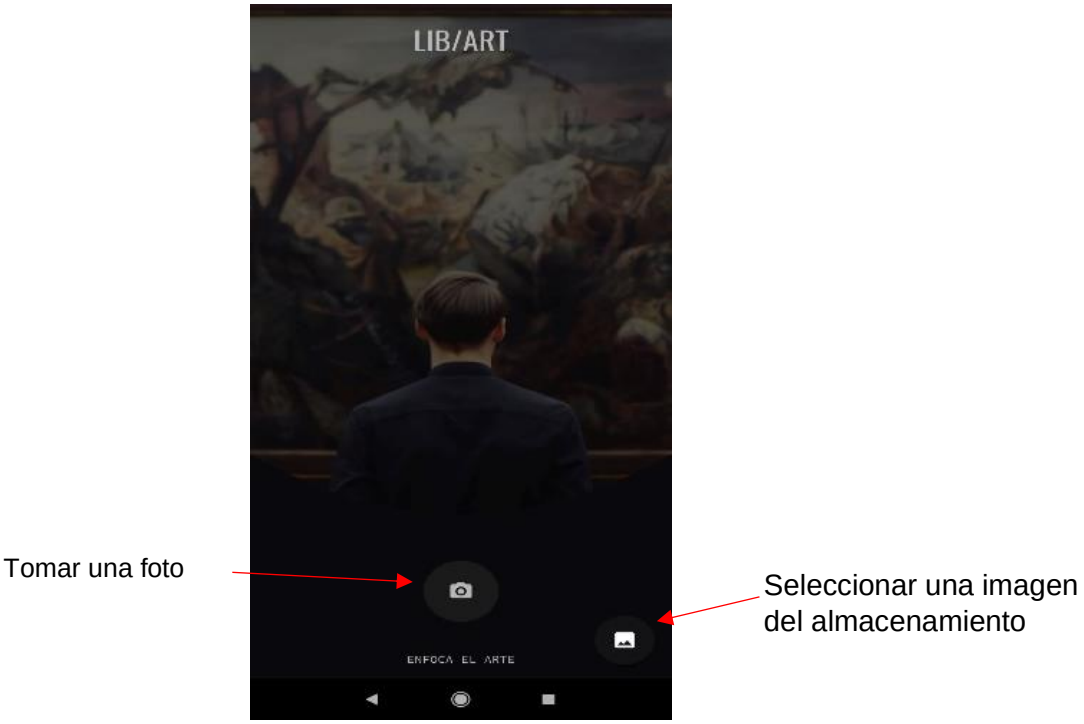

Figura 18. Pantalla principal de la aplicación.

En la primera opción, al tomar una foto aparecen otra vez dos botones para confirmar o para cancelar la foto. En caso de dar clic en confirmar, se iniciará el proceso de reconocimiento de la imagen. En caso contrario, se abre nuevamente la cámara para tomar una nueva foto.

Figura 19. Confirmar fotografía capturada.

En la segunda opción, se abre el gestor de archivos del dispositivo y selecciona una imagen para que sea reconocida por la aplicación.

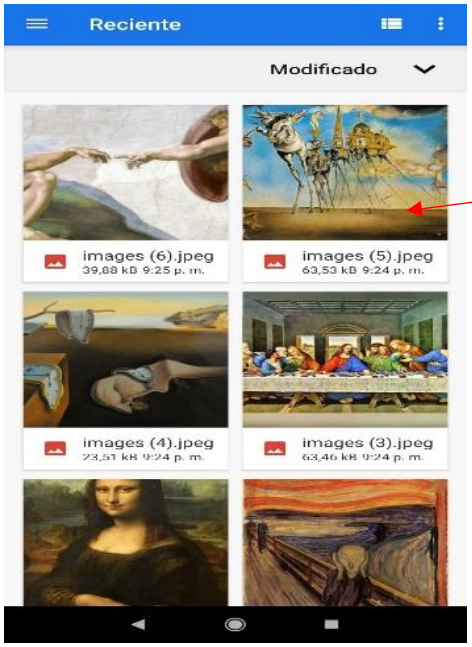

Figura 20. Pantalla para seleccionar una imagen desde el almacenamiento del dispositivo.

Se muestran los resultados obtenidos por el reconocimiento de la imagen y se puede dar clic en cualquiera de ellos, lo cual inicia la búsqueda en Wikipedia.

Figura 21. Resultados en cadenas de texto.

En caso de que se determine que la imagen es de una obra de arte, se muestra la información más importante de esta. En caso contrario, se muestra un mensaje de error. Con la flecha de navegación del dispositivo, se puede ir a la página anterior para ver otro resultado o ingresar una nueva imagen.

Figura 22. Búsqueda en Wikipedia.

Figura 23. Resultado mostrando la información encontrada de la pintura.

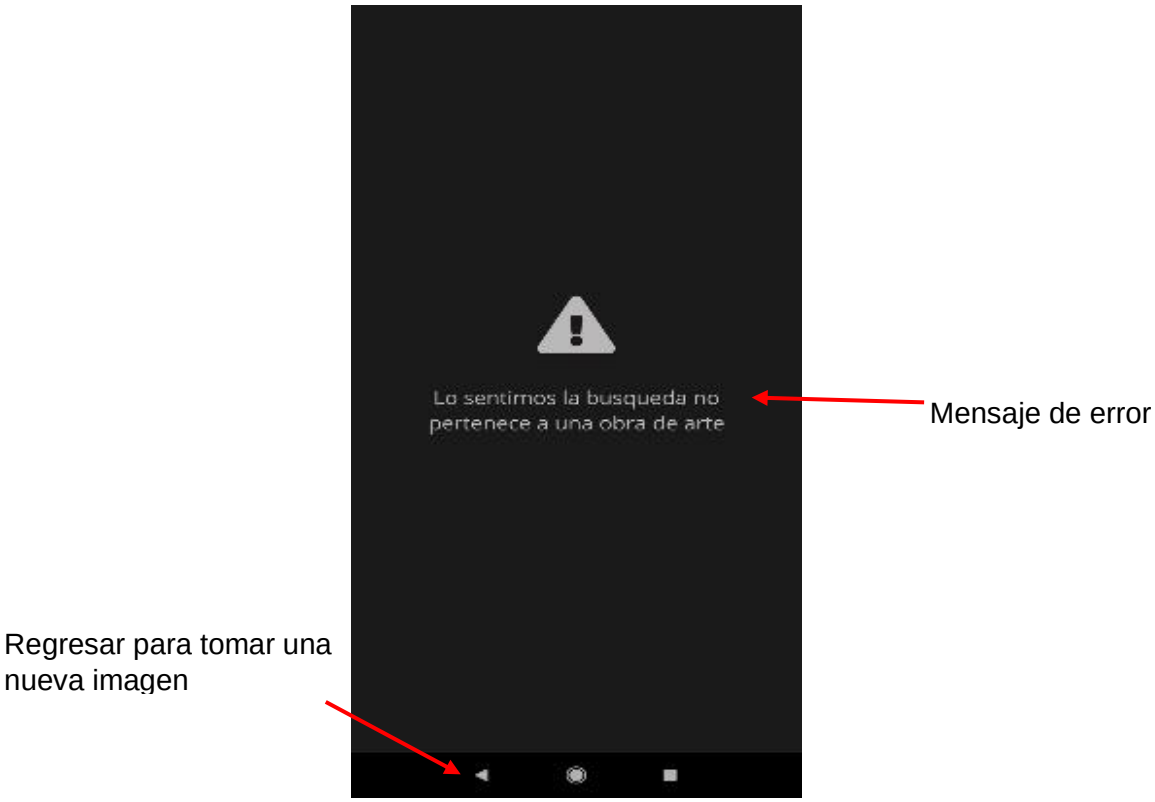

Figura 24. Mensaje indicando que no es una obra de arte.

9. Anexo 2: Manual Técnico

La aplicación esta creada con Flutter para la parte del Front-end, y con Node js para la parte del Back-End, por lo tanto, para el funcionamiento de la aplicación, se debe contar con el servidor de Node en funcionamiento y tener la aplicación de Flutter instalada en el dispositivo.

A continuación, el código con el que fue programado la aplicación:

Server.js

```
require('./config/config');
const express = require('express');
const bodyParser = require('body-parser');
const app = express();

app.use(bodyParser.urlencoded({ extended: false }));

app.use(bodyParser.json());
```

```
app.use(require('./routes/vision2'));

app.listen(process.env.PORT,()=>{
    console.log('Servidor corriendo');
});
```

Vision2.js

```
const express = require('express');
const app = express();
const multer = require('multer');
var path = require('path');
const { getConsulta } = require('../services/peticion');
const colors = require('colors');


var storage = multer.diskStorage({
    destination: function (req, file, cb) {
        cb(null, 'server/uploads');
    },
    filename: function (req, file, cb) {
        cb(null, file.fieldname + '-
' + Date.now()+path.extname(file.originalname));
    }
});

var upload = multer({ storage: storage });

/**
 * Recuperacion de información a la API de Wikipedia
 */


app.post('/libart',upload.single('image'),(req, res)=>{

    /**
     * Control de carga de imagen
     */
    const file = req.file;
    if (!file) {
        const error = {
            nivel: 1,
            message : 'Error al cargar la imagen',
            statusCode : 400
```

```
        }
        res.send(error);
        console.log(`\n
        Error al cargar la imagen
        \n`.red);
    }else{
        const fileName = req.file.filename;
        console.log(
        `\n
        Nivel 1: Carga de imagen
        \n`.green);
        getConsulta(fileName).then(
            mensaje => res.send(mensaje)
        ).catch(
            err => res.send(err)
        )
    }
});


module.exports = app;
```

peiticion.js

```
const { getWebEntities } = require('./vision');
const { getEsPintura } = require('./wikipedia');
const { getInfoPintura } = require('./wikipedia');

getConsulta = async (fileName)=>{

    let palabras = await getWebEntities(fileName);
    let value = await getEsPintura(palabras);
    let info = await getInfoPintura(value);
    return info;

}

module.exports = {
    getConsulta
};
```

Vision.js

```
const vision = require('@google-cloud/vision');
```

```
const client = new vision.ImageAnnotatorClient({
  keyFilename: 'server/key/api_key.json'
});

getWebEntities = (fileName) =>{

    return new Promise((resolve, reject)=>{

        client.webDetection(`server/uploads/${fileName}`).then((results)=>{
            let palabras = [];
            const webs = results[0].webDetection;
            if(webs.webEntities.length){
                webs.webEntities.forEach(webEntity =>{
                    if(webEntity.score > 0.70){
                        palabras.push(webEntity.description);
                    }
                });
            }

            /**
             * Se definen si las Web Entities son mayores a 75
             */

            if(palabras.length===0) reject({
                nivel: 2,
                message:'Las web entities son menores a 75',
                statusCode : 400
            })
            else resolve(palabras)
            console.log(
            `\n
        Nivel 2: Definición de WebEntities \n
        Se obtuvieron las siguientes WebEntities ${palabras}
            \n`.green);
        })

    });

}

module.exports = {
    getWebEntities
};
```

Wikipedia.js

```
const request = require('request');

getEsPintura = (palabras) =>{

    return new Promise((resolve,reject)=>{
        var noPinturas = [];
        var valuePintura = [];
        for(i =0; i<palabras.length; i++){
            let palabra = palabras[i];
            let url = 'https://en.wikipedia.org/w/api.php?action=query&forma
tversion=2&gpslimit=5&generator=prefixsearch&gpssearch='+palabra+'&prop=extr
acts|pageimages&exintro=1&explaintext=1&redirects=1&format=json&piprop=origi
nal';
            request(url,(err, res, body)=>{
                if(err) reject({
                    nivel: 3,
                    message: 'Cannot connect to the server',
                    statusCode : 400
                })
                else{
                    var wiki = JSON.parse(body);
                    for (let key in wiki.query.pages) {
                        let value = wiki.query.pages[key];
                        if(value.extract.search(/Painting|fresco/i) != -1){
                            valuePintura = value;
                        }else{
                            noPinturas.push(value);
                            console.log(`no es pintura ${value.title}`);
                        }
                    }
                }
                if(valuePintura.length != 0){
                    resolve(valuePintura)
                }else{
                    reject({
                        nivel: 4,
                        message: 'Lo sentimos no es una pintura',
                        statusCode : 400,
                        results: noPinturas
                    })
                }
            })
        }
        console.log(
        `\n
```

```
        Nivel 3: Definir si es pintura
        \n`.green);
    });
}

getInfoPintura = (value) =>{

    return new Promise((resolve, reject)=>{

        const url = "https://www.wikidata.org/w/api.php?action=wbgetentities
&sites=enwiki&props=claims&titles="+value.title+"&format=json";
        request(url, function(error, response, body){
            if(error) reject(error)
            else{
                var wiki2 = JSON.parse(body);
                for (let key in wiki2.entities) {
                    let value2 = wiki2.entities[key];
                    if(value2.claims.hasOwnProperty('P571')){
                        fecha = value2.claims.P571[0].mainsnak.datavalue.val
ue.time;
                    }else{
                        fecha = "Desconocida";
                    }
                    if(value2.claims.hasOwnProperty('P170')){
                        autor = value2.claims.P170[0].mainsnak.datavalue.val
ue.id;
                    }else{
                        autor = "Desconocido";
                    }
                    const url2 = "https://www.wikidata.org/w/api.php?action=
wbgetentities&sites=enwiki&props=claims|sitelinks&ids="+autor+"&format=json"
;
                    request(url2, function(error, response, body2){
                        var wiki3 = JSON.parse(body2);
                        for(let key3 in wiki3.entities){
                            let value3 = wiki3.entities[key3];
                            autores = value3.sitelinks.eswiki.title;

                            let info = {
                                title: value.title,
                                description: value.extract,
                                pageId: value.pageId,
                                image: value.original.source,
                                date: fecha,
                                autor: autores
```

```
                            };
                            resolve(info)
                        }
                    });
                }
            }
        });
        console.log(
        `\n
        Nivel 5: Retorno de la información de la pintura
        \n`.green);
    });

}

module.exports = {
    getEsPintura,
    getInfoPintura
};
```