Documentación proyecto aplicativo “DenunciasApp”

Nefrety Sánchez Campo

Daniela Segovia Perdomo

Ralph Sliger Ordoñez

Yulissa Restrepo Aguilera

Camilo Uribe Navas

Docente

Jairo Serrano Castañeda

Universidad tecnológica de Bolívar

Facultad Ingeniería

Ingeniería de sistemas y computación

Arquitectura de Software - Grupo DevOps D

Cartagena, 2020

Resumen

En el siguiente documento se encontrará la información correspondiente al proyecto aplicativo del curso de Arquitectura de software llamado ***"DenunciasApp"***, el cual consiste en una aplicación que permita a los habitantes de la ciudad de Cartagena realizar denuncias fotográficas acerca de la contaminación y acumulación de residuos sólidos que se da en los caños de barrios y/ localidades. La aplicación ha sido desarrollada en el lenguaje de programación Python, haciendo uso del micro-framework flask y otros recursos y metodologías de desarrollo importantes en el área de la programación de aplicaciones. Con el proyecto se busca, que, a través de las temáticas dadas en clase, se le de solución tecnológica a problemas y/o situaciones de la vida diaria que afectan a una comunidad.

Tabla de contenidos

Introducción

Uno de los principales problemas ambientales que se presentan en la ciudad de Cartagena es la acumulación de residuos sólidos en los caños, ocasionando así diferentes situaciones que pueden afectar a una comunidad. Un ejemplo de estas situaciones se da cuando hay grandes flujos de agua por las fuertes lluvias, esto hace que los caños se desborden, afectando a las comunidades aledañas, sin omitir el hecho de que la acumulación de estos residuos también genera fuerte olores que, además de ser incómodos, pueden llegar a causar enfermedades o afectaciones respiratorias en los seres humanos.

Si bien es cierto que las consecuencias de esta problemática son manejadas por ciertas entidades reguladoras de residuos, existen ciudadanos que no son conscientes del daño ambiental que causan al arrojar basura cerca o dentro de los caños. Sin embargo, hay otros ciudadanos que defienden el medio ambiente y son impulsores del manejo adecuado de los residuos dentro de su comunidad.

Descrita la problemática anterior, el proyecto aplicativo de DenunciasApp consiste en desarrollar una aplicación que, por medio de fotografías, permita a los habitantes de localidades y/o barrios de la ciudad de Cartagena realizar denuncias sobre la acumulación de residuos sólidos en los caños.

1. DenunciasApp

DenunciasApp es un proyecto aplicativo del curso de arquitectura de software de la Universidad tecnológica de Bolívar que consiste en desarrollar una aplicación que permita a los habitantes de barrios y localidades de la ciudad de Cartagena, realizar denuncias fotográficas sobre la acumulación de residuos sólidos en los caños de estas comunidades. La aplicación ha sido desarrollada en Python, implementando el micro-framework Flask y de la metodología Blueprint, la cual permite dividir la complejidad y el desarrollo de la aplicación en módulos, cada uno de ellos con sus funcionalidades y archivos específicos, se ha utilizado MongoDB e implementado Google Cloud Platform para la gestión y almacenamiento de las imágenes de la aplicación.

2. Objetivos

2.1 Generales

Desarrollar una aplicación de denuncias fotográficas sobre la contaminación presente en los caños de las áreas urbanas, haciendo uso de temáticas dadas en el curso de Arquitectura de Software, tales como patrones de arquitectura, patrones de diseño, implementación de frameworks, desarrollo del lado del servidor y utilización de contenedores a través de Docker.

2.2 Específicos

- Analizar e identificar los requerimientos funcionales de la aplicación.
- Identificar las entidades necesarias para la funcionalidad de la aplicación.
- Diseñar el modelado de las comunicaciones presentes en la aplicación.
- Desarrollar una aplicación atendiendo los estándares establecidos en el modelado inicial.

Esta aplicación está enfocada a la comunidad en general y específicamente a los ciudadanos afines con el cuidado del medio ambiente y las empresas reguladoras de desechos y residuos, aquellos que son conscientes del daño que se genera al planeta con la contaminación y mal manejo de las basuras.

3. Requerimientos

3.1 Requerimientos funcionales

3.1.1 Requerimientos del usuario

- El usuario debe registrarse en la aplicación.
- Una vez logueado, podrá observar las denuncias fotográficas más recientes sobre la contaminación en los caños de las áreas urbanas con su respectiva ubicación.
- El usuario registrará las denuncias fotográficas que considere pertinentes.

3.1.2 Requerimientos del sistema

- La aplicación podrá ser utilizada en cualquiera sistema operativo.
- La aplicación deberá ser utilizada en un navegador.
- Los datos de usuario y denuncias serán almacenados en una base de datos.
- La aplicación mostrará las imágenes con su respectiva ubicación en el mapa.

3.2 Requerimientos no funcionales

- Obtener la información de los registros de las denuncias hechas por los usuarios.
- La aplicación será desarrollada para un entorno web.
- La aplicación debe ser responsive, para garantizar la adecuada visualización en múltiples computadoras y dispositivos móviles.

4. Metodología

La aplicación ha sido desarrollada en el lenguaje de programación Python haciendo uso del micro-framework Flask y de la metodología Blueprint, la cual permite dividir la complejidad y el desarrollo de la aplicación en módulos, cada uno de ellos con sus funcionalidades y archivos específicos, por medio de la utilización de esta metodología se puede hacer la instancia de un sólo objeto de Flask que haga uso de los diferentes módulos. En cuanto al procesamiento de los datos y/o información se utiliza MongoDB, que es una base de datos NoSQL, es decir, que no se trabaja con un modelo relacional de entidades, sino que se implementan colecciones como objetos y documentos, además, se ha implementado Google Cloud Platform (GCloud), que actuará como una base de datos completamente enfocada en la gestión y almacenamiento de las imágenes de la aplicación, lo que abarca tanto las imágenes de perfil de los usuarios registrados como las imágenes de las denuncias.

Como se ha mencionado anteriormente, la aplicación estará dividida en módulos, a continuación, se describe cada uno de ellos:

- **Módulo usuario:** Este módulo está orientado a las operaciones CRUD relacionadas con los usuarios de la aplicación, se encarga de administrar datos como nombre, ciudad, email. Además, se ha creado un superusuario con los permisos necesarios para hacer control de usuarios.

- **Módulo Denuncias:** Es el encargado de gestionar las operaciones CRUD relacionadas con las denuncias realizadas y registradas en la aplicación. Administra datos como imagen, datos de localización, descripción de la denuncia y nombre del usuario que la realiza.

5. Patrones de arquitectura

Los patrones de arquitectura se refieren a los esquemas de organización utilizados para estructurar un sistema a nivel general, es decir, se puede interpretar como un bosquejo de dicho sistema, en este caso, en la aplicación se implementará el patrón de arquitectura MVC (Model, View, Controller), el cual se basa en tres componentes fundamentales descritos a continuación:

- **Modelo:** Corresponde a la manera en que se representarán los datos manejados dentro del sistema, es decir, generalmente aquí se realizan las consultas y peticiones a la base de datos.
- **Vista:** Se centra en la parte visual del sistema, es decir, en la interfaz con la que va a interactuar el usuario.
- **Controlador:** Se encarga de establecer una conexión entre el modelo y la vista, de tal manera que se gestione la información de la manera más pertinente para el usuario y se responda a sus peticiones.

En la aplicación DenunciasApp el patrón **MVC** se ve implementado de la siguiente manera:

Tanto en el módulo **User** como en el módulo **Complaint,** se encuentran diferentes archivos, dentro de los cuales podemos mencionar los siguientes: **models.py** en el que se puede visualizar la conexión a la base de datos (**Modelo**), está el archivo **views.py**, en donde se encuentran todas las rutas que retornan las vistas de la aplicación, es decir, la interfaz con la que interactúa el usuario (**Vista**) y por último el archivo **app.py** que es donde se hace el llamado a cada uno de los

módulos que estructuran la funcionalidad de la aplicación, por lo que sirve de intermediario entre las vistas y el controlador, específicamente entre los archivos models.py y views.py.

6. Patrones de diseño

Los patrones de diseño son soluciones para problemas típicos y recurrentes que se pueden presentar al momento de desarrollar una aplicación. En DenunciasApp se implementan los siguientes patrones de diseño:

- **Factory:** Es un patrón de diseño creacional, su objetivo es proporcionar una interfaz para crear objetos en una superclase, pero permite que las subclases alteren el tipo de objetos que se crearán.

  Este patrón ha sido aplicado directamente en el código fuente al considerar a cada módulo como una app independiente e instanciando las mismas en un sólo archivo llamado app.py.

- **Decoradores:** Es un patrón de diseño estructural que permite añadir nuevos comportamientos a los objetos, colocándolos dentro de otros objetos de envoltura especiales que contienen los comportamientos.

  La implementación de este patrón en la aplicación se encuentra en el archivo decorators.py, que hace parte del módulo user. Al envolver el decorador se logra ocultar que la función original ha sido decorada.

7. Modelo de comunicaciones

De acuerdo a los patrones tanto de arquitectura como de diseño descritos previamente, a continuación, se podrá visualizar a través de diagramas UML el modelado de las comunicaciones presentes en la aplicación junto con la descripción de cada uno de sus componentes.

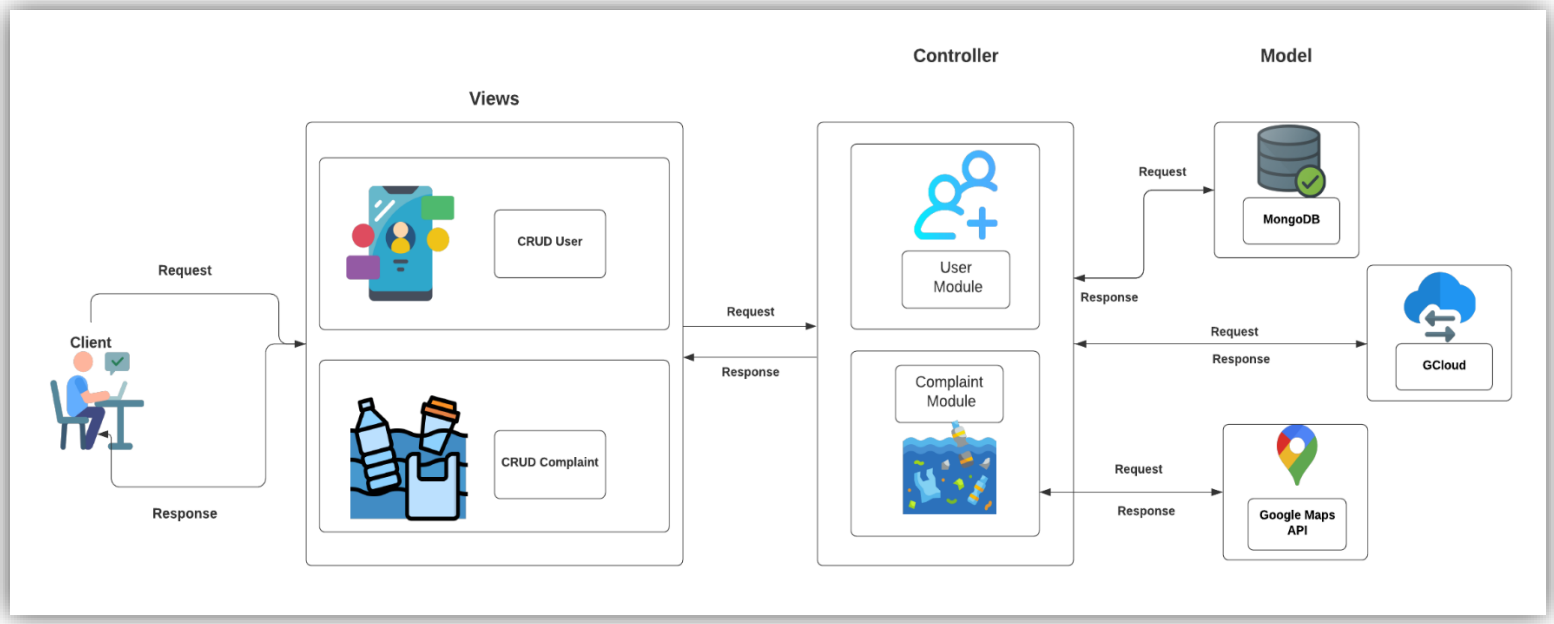

Ilustración 1. Arquitectura general de DenunciasApp

Funcionamiento: Un cliente se conecta a las vistas **(Views)** de la aplicación, estas envían una petición que es recibida por el controlador **(Controller)**, dependiendo del tipo de petición el controlador sabrá qué hacer al momento de conectarse con la base de datos **(Model)**. Por ejemplo: Si el cliente desea realizar un nuevo registro, el controlador recibe una petición POST, luego este le indica a la base de datos que los almacene y le da una respuesta al cliente, observada a través de las vistas.

Descripción de componentes:

- **Client:** Cliente que se conecta a la aplicación y realiza peticiones.
- **Views:** Corresponden a las vistas de la aplicación, incluyendo tanto las de usuario como las de denuncia. Este componente está a su vez conformado por los CRUD de User y Complaints, el primero se encarga de realizar las operaciones CRUD relacionadas con los clientes, mientras que el segundo realiza estas mismas operaciones, pero con los datos de las denuncias. El cliente tendrá una conexión directa con este componente, ya que gestionará la interfaz correspondiente a cada petición.
- **Controller:** Es el controlador de la aplicación, se encuentra conformado por los controladores de los módulos principales, los cuales están encargados de gestionar y responder a las peticiones que hacen los clientes respecto a sus datos o a las denuncias, permitiéndoles visualizar estas acciones a través de la interfaz.
- **Model:** Hace referencia al modelo de la aplicación, se encuentra conformado por la base de datos MongoDB, la implementación de GCloud y la API de Google Maps, la primera es una base de datos no relacional encargada de gestionar y almacenar los datos de la aplicación a nivel general, es decir, los datos relacionados con las denuncias y los usuarios, lo segundo se enfoca en la administración y almacenamiento de las imágenes de la aplicación, refiriéndose con esto a las imágenes de perfil de los usuarios registrados y a las imágenes de las denuncias donde se visualiza la acumulación de residuos sólidos y la tercera actúa como un servicio de Google que está siendo consumido para visualizar ubicaciones en el mapa, es decir, gestiona y almacena las direcciones de las denuncias.

7.1 Google Cloud Platform (GCloud)

Al momento de hablar de GCloud, se debe primero tener en cuenta un concepto muy importante como lo es FaaS, que corresponde a la abreviatura del concepto “Functions as a Service” o “Funciones como un Servicio” y se define como un modelo de ejecución que permite que ciertos elementos de una aplicación sean ejecutados en contenedores sin estado, hacer uso de FaaS simplifica el ciclo de desarrollo de las aplicaciones basadas en microservicios y permite que el costo de la implementación del elemento de código en plataforma sólo se vea reflejado cuando es usado, de lo contrario no se genera valor. Los servicios de alojamiento en la nube se dan a través de ciertos proveedores como AWS, Azure, Google Cloud Platform, etc.

En la aplicación se ha utilizado GCloud que es una plataforma de google en la nube con recursos físicos como servidores y servicios de almacenamiento ubicados en diferentes centros de datos de google, se destacada porque sus servicios van desde alojamiento de sitios web hasta alojamiento de API´S o aplicaciones en desarrollo.

La utilidad de GCloud en DenunciasApp consiste en que la plataforma actuará como una base de datos adicional encargada de gestionar y almacenar todas las imágenes de la aplicación, haciendo referencia con esto a las imágenes de perfil de los usuarios (módulo User) y a las imágenes de las denuncias realizadas (módulo Complaint).

7.2 API de Google Maps

Una Interfaz de programación de aplicaciones o API se define como un conjunto de definiciones y protocolos que se utilizan para integrar el software de aplicaciones, permite que un programador pueda utilizar un sistema sin tener en cuenta de que manera esta implementado.

En la aplicación se ha implementado la API de Google Maps, la cual es un servicio de google que permite generar mapas, crear rutas, insertar marcadores, etc. Con esta API es posible mostrar a los usuarios de una aplicación lugares específicos. En pocas palabras, esto permite que se gestionen datos relacionados con la localización.

En DenunciasApp, la API de Google Maps está siendo consumida con el fin de poder visualizar el mapa con las ubicaciones y direcciones desde las cuales se está realizando la denuncia, por lo que su implementación se observada directamente en el módulo de Complaint.

Tanto en el módulo User como el módulo Complaint, GGloud se está utilizando como la base de datos encargada de almacenar las imágenes. Con respecto al módulo User, almacena las imágenes de perfil de los usuarios que están registrados y en el módulo Complaint, se encarga de almacenar las imágenes de las denuncias realizadas por los usuarios.

Esta implementación se puede observar en el archivo **Storage.py** (DenunciasApp\App\utilities), en el cual existe una función llamada **“upload_image_file”**, la cual recibe como parámetros un archivo, la carpeta de procedencia del mismo y el ID de contenido, para su funcionamiento se hace uso del paquete google.cloud storage para flask.

Lo primero que hace la función es verificar si el archivo existe, de ser así, se procede con la autenticación al bucket de google cloud a través de un apikey.json (generado por google al configurar el bucket) guardada localmente en el proyecto.

Cuando el servicio está autenticado, se instancia el cliente de google cloud y busca el listado de buckets que pertenecen al apikey, se conecta al bucket con el id ‘denuncias-bucket’ y se le pasa como parámetro la ruta de la imagen que el usuario está cargando en la vista, esto se guarda en el bucket y se decodifica a uft-8.

7.3 Módulo User (Módulo usuario) – CRUD User

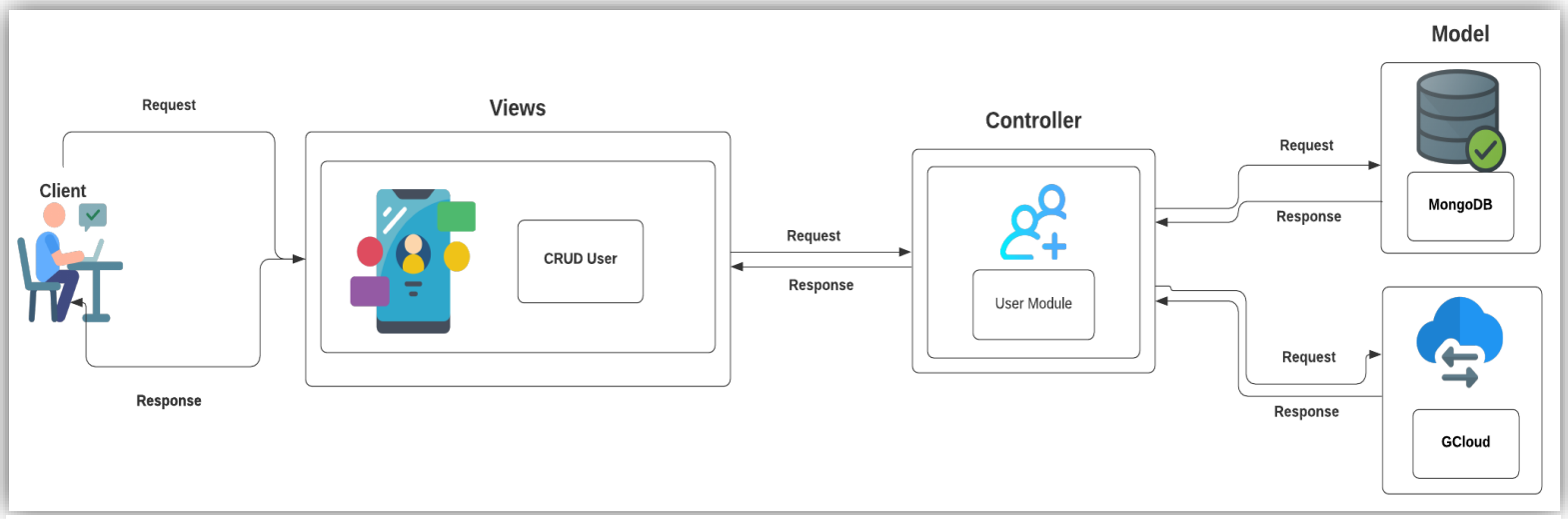

Ilustración 2. Arquitectura módulo User (Módulo usuario) - CRUD User

Descripción: Este módulo realiza operaciones CRUD relacionadas con los clientes, se encarga de permitir la **creación y el acceso de los usuarios**, de tal manera que, realizando peticiones al **controlador**, este pueda gestionar una respuesta a través de la conexión con la base de datos **MongoDB** y **GCloud**.

Los componentes principales de este módulo, como se visualiza en el gráfico, son:

- **Client (Cliente):** Cliente que se conecta a la aplicación y realiza peticiones.

- **CRUD User:** Corresponde a las operaciones CRUD relacionadas con los clientes, éstas permiten la creación, el registro y la modificación de los datos suministrados. Al momento en el que el cliente realice una conexión con este componente, se habilitará la interfaz respectiva a cada una de las peticiones. En el caso de que el usuario desee modificar la imagen de perfil, se hace uso de GCloud, que gestionará y almacenará dicha imagen.
- **User Module (Módulo usuario):** Corresponde al controlador del módulo usuario, es decir, aquí se encuentra la lógica y el funcionamiento general de este módulo. Además, es el encargado de gestionar las peticiones, conexiones y respuestas que recibirá el cliente a través de la interfaz.
- **MongoDB:** Base de datos que podrá gestionar y almacenar toda la información pertinente respecto al cliente, tales como nombre, ciudad, e-mail y contraseña.
- **GCloud:** Funciona como una base de datos adicional encargada de gestionar y almacenar las imágenes de perfil de los usuarios registrados.

7.3.1 Rutas del módulo User

- **/user:** Ruta principal del módulo.
- **/user/newuser [‘GET’,’POST’]:** Ruta para creación de un usuario.
- **/user/login [‘GET’,’POST’]:** Ruta para acceso de un usuario.
- **/user/logout:** Ruta para cerrar sesión
- **/user/edit [‘POST’, ‘GET’]:** Ruta para editar los datos del usuario.
- **/user/password [‘POST’, ‘GET’]:** Ruta para cambiar contraseña del usuario.

7.4 Complaint Module (Módulo denuncia) - CRUD Complaint

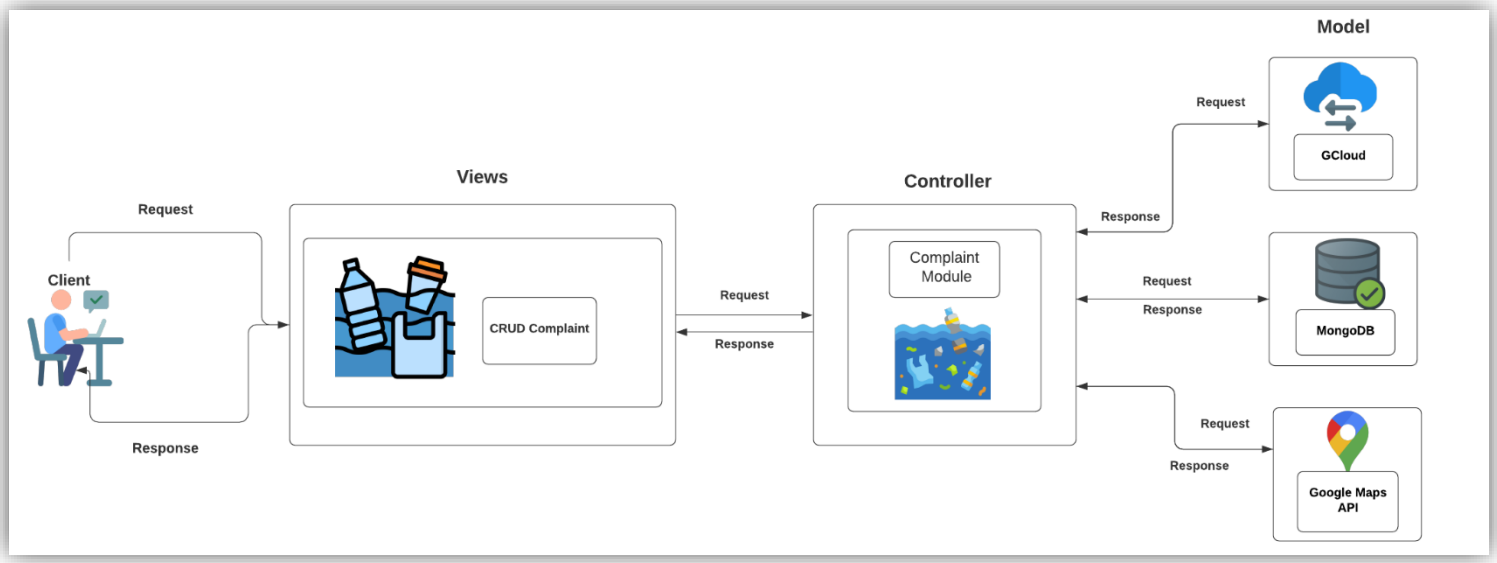

Ilustración 3. Arquitectura módulo Complaint (Módulo denuncia) - CRUD Complaint

Descripción: Será el módulo encargado de permitir al cliente tener **acceso o visualización de las denuncias** hechas por los demás usuarios a través de una conexión con el **controlador**, el cual se conectará a su vez a la base de datos **MongoDB** y a la **API de Google Maps** para gestionar y almacenar datos como imágenes, localización (Latitud, longitud), descripción de la denuncia y cliente al cual pertenece ese registro.

Las componentes principales de este módulo son los siguientes:

- **Client:** Cliente que se conecta a la aplicación y realiza peticiones.
- **CRUD complaints:** Corresponde a las operaciones CRUD relacionadas con las denuncias, estas permiten su creación y modificación.

- **Complaints Module (Módulo denuncias):** Es el controlador del módulo denuncias, es capaz de gestionar las peticiones, conexiones y respuestas que recibirá el cliente acerca de las denuncias realizadas o que desea registrar.
- **GCloud:** Se encarga de gestionar y almacenar las imágenes de las denuncias realizadas por un usuario, es decir, las imágenes donde se evidencia la acumulación de residuos sólidos en los caños.
- **MongoDB:** Base de datos que podrá crear, registrar, eliminar y modificar datos relacionados con la denuncia fotográficas, tales como imagen, localización, descripción de la denuncia y cliente que realiza la denuncia.
- **API Google Maps:** Es el componente encargado de gestionar la visualización de las ubicaciones/direcciones en el mapa, esto lo hace a través de la recepción de parámetros como nombre de la ubicación, lo que permite que se tomen por defecto otros parámetros de importancia como la latitud y la longitud. Es un servicio de Google que está siendo consumido para visualizar ubicaciones.

7.4.1 Rutas del módulo Complaint

- **/complaint:** Ruta principal del módulo denuncia.
- **/complaint/create [‘GET’,’POST’]:** Ruta que permite la creación y registro de las denuncias.
- **/complaint/<id> [‘GET’]:** Ruta para conocer detalles de una denuncia, visualizarla.
- **/complaint/<id>/support [‘GET’]:** Ruta para que un usuario pueda apoyar una denuncia.

- **/complaint/<id>/unsupport [‘GET’]:** Ruta para que un usuario pueda dejar de seguir o apoyar una denuncia.
- **/complaint/<id>/edit [‘GET’,’POST’]:** Ruta que permite modificar los datos registrados de la denuncia, esto sólo se permite si el usuario está registrado en la base de datos.
- **/complaint/<id>/cancel [‘GET’,’POST’]:** Ruta que deshabilita la vista de una denuncia, sólo se permite si existe un ID asociado a la base de datos.
- **/complaint/explore [‘GET’]:** Ruta para la vista principal de denuncias, contiene todas las denuncias registradas.
- **/complaint/manage[‘GET’]:** Ruta para visualizar las denuncias realizadas por un usuario, esto se permite siempre y cuando el usuario haya iniciado sesión previamente.

8 Colecciones de la base de datos principal MongoDB

Al implementar una base de datos NoSQL como lo es MongoDB es propicio destacar que en lugar de entidades aquí se trabaja con colecciones y documentos, lo primero corresponde al conjunto de características de un objeto y lo segundo es la manera de visualizar de manera conjunta esas características o atributos de una misma colección, además, MongoDB provee un ID automático para cada uno de los objetos que se creen, esto facilita las consultas que se quieran realizar en la base de datos. De acuerdo a esto, las colecciones y documentos existentes en la implementación de la aplicación se visualizan a continuación.

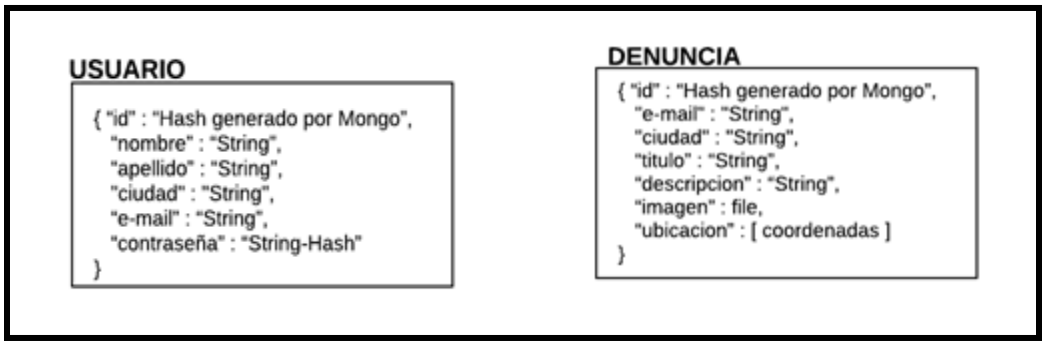

Ilustración 4. Colecciones base de datos MongoDB en DenunciasApp

Link del repositorio: https://github.com/ralphsliger/DenunciasApp

9 Bibliografía

- ¿Qué es el patrón MVC en programación y por qué es útil? Disponible en: https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx

- Design Patterns in Python. Disponible en: https://refactoring.guru/design-patterns/python

- Modular Applications with Blueprints. Disponible en: https://flask.palletsprojects.com/en/1.1.x/blueprints/

- Descripción general de Google Cloud Platform. Disponible en: https://cloud.google.com/docs/overview?hl=es-419