Aseguramiento de la Calidad del Software
Documento del PoC

André Arroyo Piedra 2015073657
Bryan Vargas 2015011562
Juan Villacis 2016201681

23 de Agosto del 2018

1. Repositorio

Es posible encontrar el repositorio donde se almacena el proyecto en la siguiente direccion
https://github.com/4a75616e/segmentator

2. Diagrama de componentes

En la figura 1 es posible observar el diagrama de componentes del sistema

3. Analisis de trozos implementados

3.1. Backend

El backend fue implementado en el lenguaje de programacion Python usando la libreria para desarrollo de aplicaciones web Flask. Esta herramienta permite desarrollar rapida y facilmente aplicaciones y se puede unir sencillamente con otras herramientas necesarias para este *PoC* como Keras, Pandas y Numpy.

3.2. Frontend

La interfaz fue implementada mediante la union de HTML5, AngularJS y CSS. El HTML5 se encarga de darle a la interfaz la estructura basica. Esta herramienta fue utilizada por su simplicidad y ya que es necesaria cuando se trabaja en entornos web. El CSS se usa para mejorar la apariencia de la pagina, al modificar el color, el tipo de letra, el tamaño y la ubicacion de los elementos agregados con HTML. Por ultimo se uso la libreria AngularJS para poder implementar funcionalidades adicionales en la interfaz y poder comunicarse con el backend. En la figura 2 es posible observar la pagina de inicio de la aplicacion web.

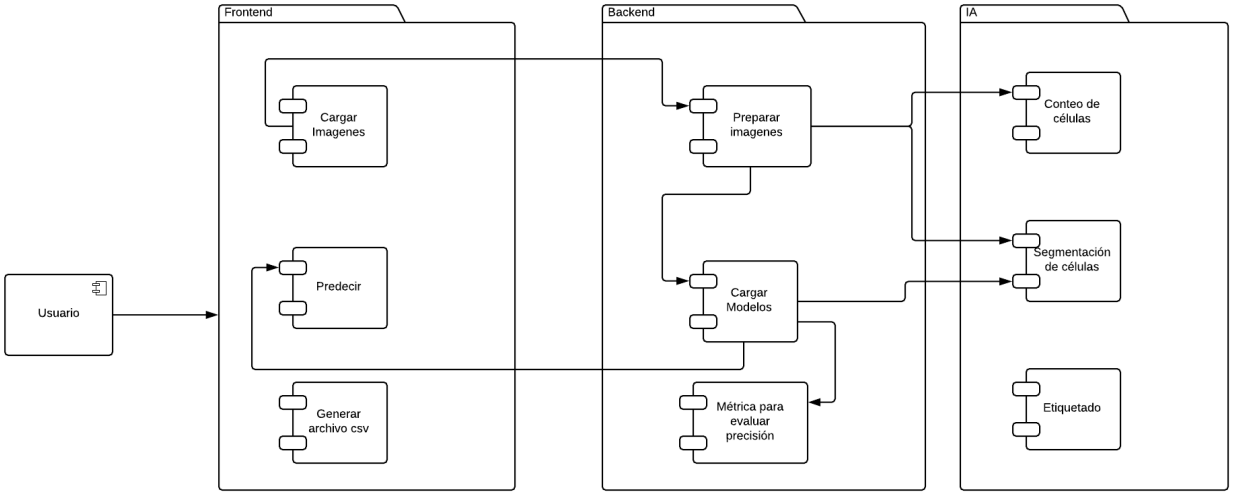

Figura 1: Diagrama de componentes del sistema de segmentacion de celulas

Figura 2: Pagina de inicio de la aplicacion web

3.3. Carga de imagenes

El proceso de carga de imagenes fue realizado mediante la union de distintas herramientas en el backend y en el frontend. En el frontend se uso un *form* de HTML5 en el que el usuario carga la imagen. Despues de esto, mediante una funcion asincronica de AngularJS se envia la imagen al servidor con una peticion

POST. En el servidor se obtiene la imagen y se procede a almacenarla en un directorio destinado para tal fin

3.4. Algoritmos con keras

El algoritmo implementado con Keras corresponde a la clasificacion de una imagen usando la red *VGG_16*. Para enviar la imagen a la red se usa la metodologia descrita en la seccion 3.3. Una vez que el servidor tiene la imagen se le hace un preprocesamiento simple usando los algoritmos *preprocess_input*, *img_to_array* y *np.expand_dims*. Despues de esto se pasa por el modelo (que se carga una sola vez cuando se inicia la aplicacion) y se retorna la clase que este predice como parte de los parametros de respuesta de la peticio *POST*

3.5. Generacion de archivos *.csv*

Para generar estos archivos se usa la libreria *Pandas*. El usuario envia una peticion *GET* al servidor, y este le responde con un archivo *.csv* que acaba de generar. En el servidor se construye el archivo usando los mecanismos que provee Pandas para tal fin. Cuando ha generado el archivo envia

4. Detalles de problemas encontrados

A continuacion se detallan los problemas encontrados mas significativos

1. **Envio de imagen al servidor mediante AngularJS:** Al inicio se enviaba la imagen mediante un *form* de HTML; sin embargo esto no permitia mantener al usuario en la misma pagina, para que luego pudiera obtener una respuesta en tal sitio o pudiera cargar otra imagen. Por esto se desarrollo una funcion de AngularJS que enviara la imagen y mostrara una respuesta al usuario. El principal desafio de esto fue la falta de documentacion de como realizarlo, ya que habia varios aspectos relacionados al envio de los datos en la peticion *POST* que debian tener configuraciones especificas. Un ejemplo de esto es que los datos enviados debian tener en el encabezado que eran de tipo *multipartformdata* . Para poder resolver este problema se recurrio a una gran cantidad de sitios en el internet en los que habia ejemplos y a la documentacion de AngularJS y Flask. Despues de hacer todo esto se logro solucionar el problema

2. **Obtencion de imagenes enviadas al servidor:** Otro problema relacionado al punto anterior fue el de como obtener los datos enviador al servidor con la peticion *POST* hecha en AngularJS. La documentacion sobre como realizar esto en Flask, al igual que en el punto anterior, es escasa. Por lo que se requirio la realizacion de pruebas con las distintas configuraciones posibles asi como la busqueda de ejemplos en el internet para poder solucionar este problema

3. **Implementacion de la herramienta Doxygen:** Cuando se comenzo a utilizar esta herramienta en eclipse se tuvo problemas al intentar integrarlo. Despues de varias busquedas en internet se encontro que la pagina de donde se estaba intentando descargar el plugin de eclox estaba obsoleta y que había cambiado, con esto ya fue posible la integracion de la herramienta a eclipse.