Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Curso de Aseguramiento de la Calidad del Software
Avance No. 2.2 de Proyecto:
Documento de Extensión del SyRS

Francisco Monge Zúñiga (2013029434)
Andrés Gutiérrez Salas (201223823)
Joseph Salazar Acuña (2015100516)

October 18, 2018

1 Introducción

Los dispositivos de captura y procesamiento de imágenes biomédicas, como resonancias magnéticas, radiografías y tomografías computarizadas, cobran cada vez mayor protagonismo en los procesos de toma de decisiones para los profesionales de la salud. Más recientemente, la industria biomédica ha desarrollado interesantes sistemas de software que realizan diagnósticos automáticos a partir de imágenes y señales biomédicas.

En nuestro caso específico, se abordará un problema de gran interés: la segmentación de imágenes digitales tomadas de un tejido de glioblastoma, para la detección de células cancerígenas. Proceso que, de realizarse de manera manual, toma demasiado tiempo por parte de los microbiólogos encargados.

En el presente Documento de Especificación de Requerimientos (SyRS) encontrará una serie de funcionalidades, atributos y condiciones que el sistema de software a desarrollar debe cumplir, de cara a resolver satisfactoriamente las necesidades de quienes lo van a utilizar.

2 Prioridad de Requerimientos funcionales

Para este avance se especificaron una serie de necesidades prioritarias, a saber:

1. Cargar una secuencia de imágenes almacenada en la dirección provista por el usuario.
2. Segmentar las imágenes en las regiones de fondo y no fondo. a) Los resultados deben ser almacenados en archivos.
3. Todas las funcionalidades deben ser accesibles desde la interfaz web del sistema.

Dadas estas condiciones fue necesario ajustar las prioridades de todos los requerimientos del sistema. Dichos cambios pueden ser consultados en el Documento de Especificación de Requerimientos (SyRS), que se adjunta junto con este documento.

Prioridad de Requerimientos funcionales

3 Diagrama de Componentes

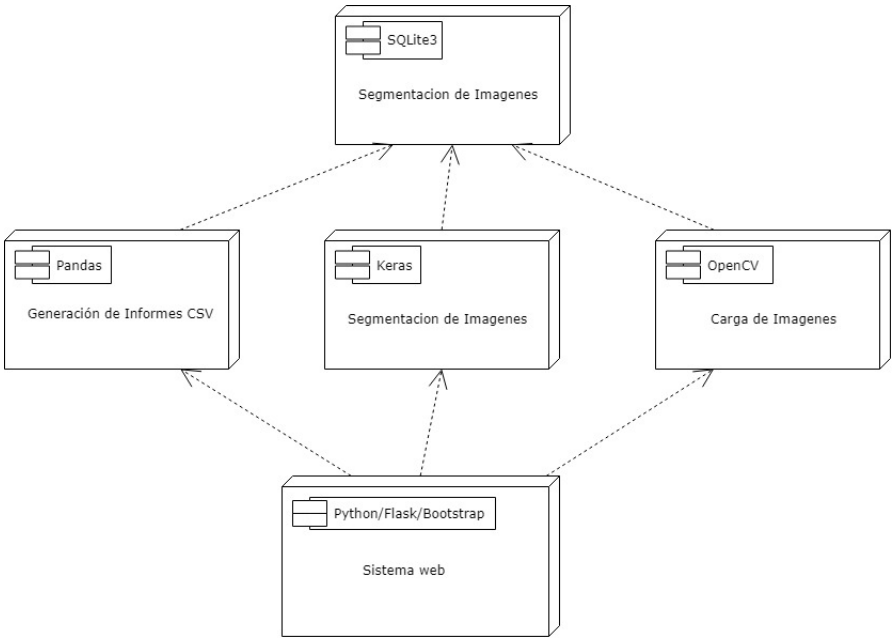

Figure 1: Diagrama de componentes de alto nivel del sistema implementado.

4 Diagrama de Clases

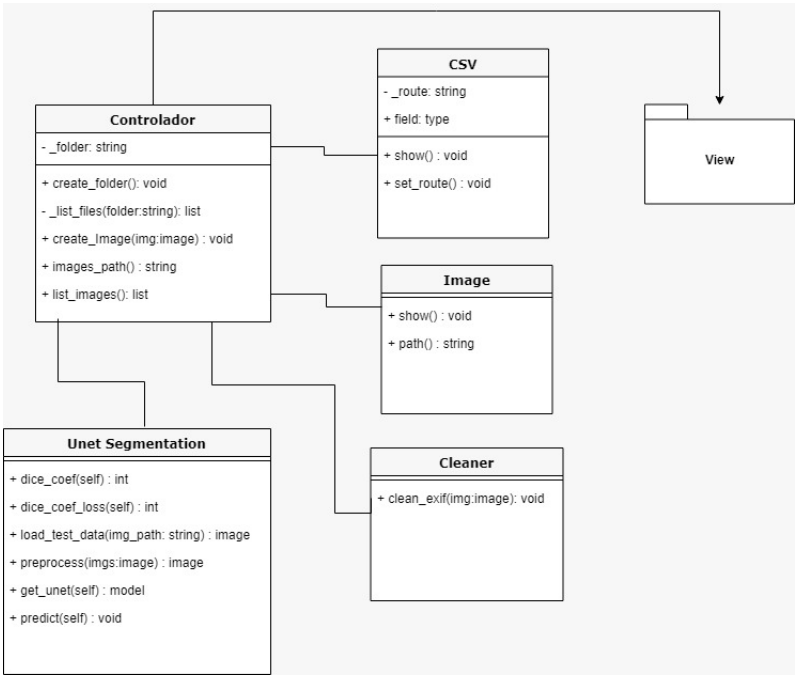

Figure 2: Diagrama de clases del sistema implementado.

Bibliografía consultada: Larman, C. (s.f.) "Appliyng UML and Patterns - An introduction to Object-Oriented Analysis and Design and the Unified Process".

Validación del diseño para requerimientos	
Artefacto Diseño	Requerimiento
Cleaner	REQ-1
Image	REQ-1
Unet Segmentation	REQ-2
CSV	REQ-4
View	REQ-6

Validación de implementación para diseño	
Ítem Implementado	Ítem Diseño
Cleaner	Cleaner
LoadImages	Image
USegmentation	Unet Segmentation
LoadCSV	CSV
Controlador	Controlador

5 Validación del Diseño

En la siguiente tabla se muestra la validación de los artefactos de diseño respecto a los requerimientos del SyRS. Es decir, los requerimientos del sistema que justifican la existencia de los diferentes artefactos incorporados en el diseño.

6 Validación de la Implementación

En la siguiente tabla se muestra la validación de la implementación del sistema respecto a los artefactos de diseño previamente establecidos. Es decir, que la implementación realizada realmente cumpla con los elementos tal cual fueron establecidos en el diseño.

7 Estándares de Codificación

El estándar de codificación que se implementará en el lenguaje Python es conocido como Python Enhancement Proposals (PEP). Su última versión es la número ocho (PEP8), siendo esta es la que se utilizará como referencia. Se implementará dicho estándar de codificación dado que es el oficial y el que soporta Python y toda su comunidad.

Una de las herramientas que se está usando para validar el estándar de codificación es pycodestyle. Esta herramienta permite conocer cuáles son todos los incumplimientos de pep8 para un archivo dado. El método más sencillo para correrlo es por medio de la consola.

El comando más básico solamente dice el error y la línea en que se encuentra:

pycodestyle –first nombreArchivo.py

El siguiente comando genera la linea donde hay incumplimiento, el codigo que incumple y las reglas de pep8 que incumple:

pycodestyle –show-source –show-pep8 nombreArchivo.py

Es importante tomar en cuenta que nombreArchivo debe incluir toda la ruta del archivo que se desea analizar. Se adjunta una imagen de ejemplo:

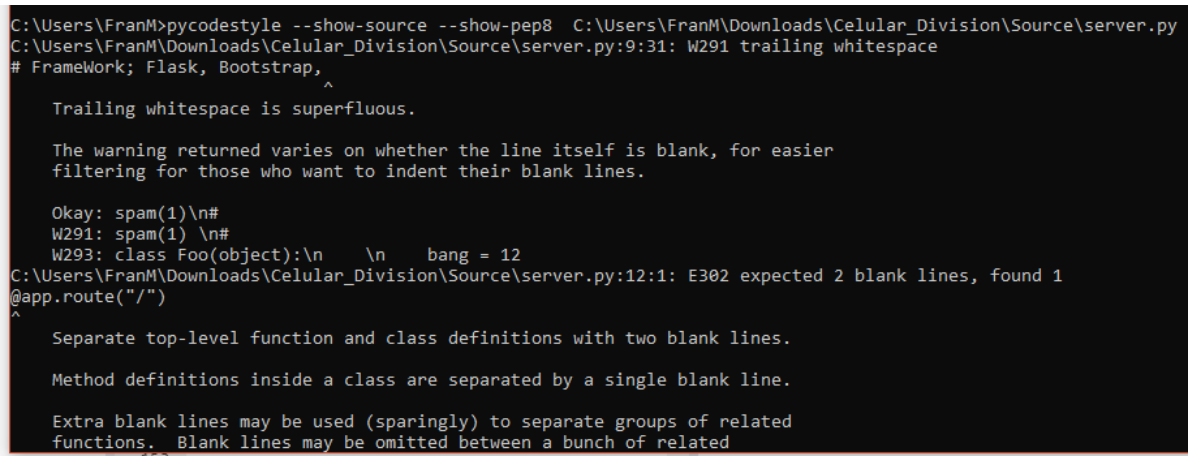

Figure 3: Corrida de ejemplo de la herramienta pycodestyle.

Por otra parte, se está utilizando PyLint, que en algunas cosas es similar a pycodestyle, pero valida más cosas que la anterior, por decirlo de alguna manera. Además de ver cuáles son los incumplimientos del estándar pep8 presentes en el código, permite ver el número de ocurrencias. Además, genera un índice que va del 1 al 10, siendo uno lo más bajo y 5 lo mas alto, que indica qué porcentaje de cumplimiento se ha obtenido. Por ejemplo, un archivo con índice 10 no incumple ninguna de las normas del pep8.

Se muestra un ejemplo a continuación:

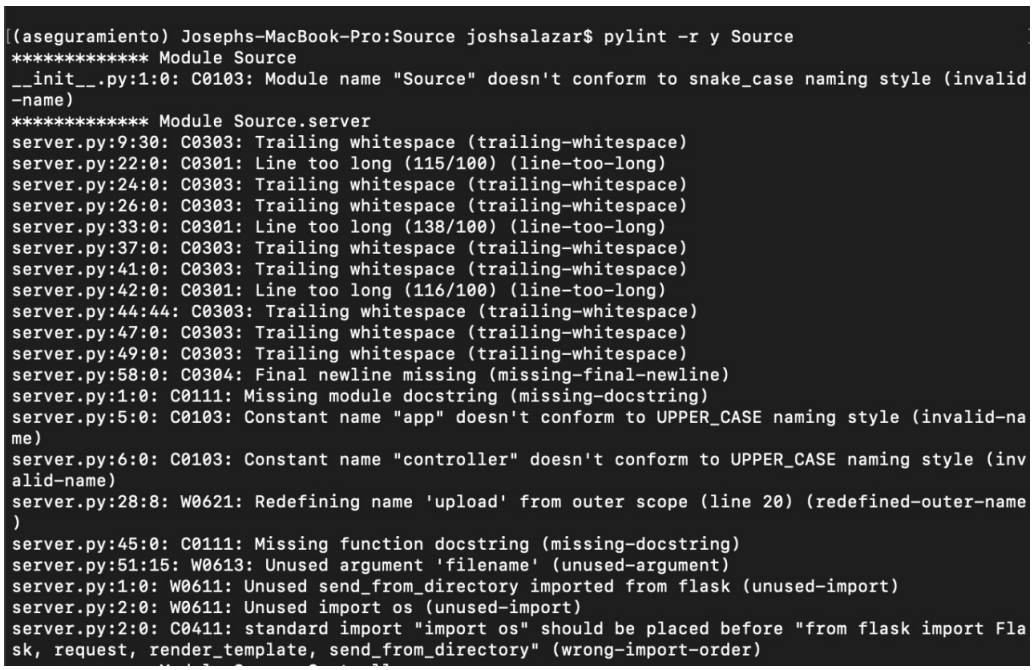

Figure 4: Corrida de ejemplo de la herramienta PyLint.

Ambas herramientas se complementan muy bien y permiten llevar a cabo un buen análisis del código escrito bajo el estándar pep8.

8 Configuración del software

Se añadió al repositorio de Github un mecanismo para poder obtener la última versión del sistema con las configuraciones necesarias para que sea ejecutado correctamente. Además, ya se encuentran allí todos ítems de configuración del sistema hasta el momento.

9 Métricas

La primera métrica que se estableció es la siguiente: Todo archivo con código python incluido en el proyecto debe tener un índice (generado por PyLint) mayor o igual a 9, esto en busca de garantizar un alto nivel de mantenibilidad por parte del equipo.

Para mostrar un ejemplo, la primera vez que se ejecutó la verificación en el archivo llamado "cleaner.py", el índice generado fue de 7. Tras realizar las correcciones y de nuevo la verificación, se obtuvo un índice de 9.

La segunda métrica que se estableció es la siguiente: Todo archivo con código python incluido en el proyecto tendrá un valor de importancia asignado, de manera tal que complementando dicho valor con el número de líneas de código totales en el proyecto, se pueda generar una media del tamaño recomendado de cada archivo (clase), que permita garantizar una entendibilidad alta por parte del equipo.

10 Actividad Aseguramiento de Calidad

La actividad de aseguramiento de calidad realizada para este avance es la actividad número 4, correspondiente al cumplimiento de los estándares de codificación. Con esto se logró corregir los incumplientos del estándar pep8 en el proyecto.