

# Aseguramiento de la Calidad del Software

## Avance 3, Proyecto Semestral

M. Sc. Saúl Calderón Ramírez  
Tecnológico de Costa Rica  
Escuela de Ingeniería en Computación  
Ignacio Cantillo, Daniel Delgado, Kevin Montoya

11 de noviembre de 2018

## 1. Introducción

El aseguramiento de la calidad del software es un tema de mucha importancia en la actualidad, aunque muchas empresas no le dan la importancia que este tema merece. En el software y en muchos otros ámbitos podemos relacionar la calidad con el valor que obtiene un proyecto u objeto al asegurarnos que cuenta con ciertos atributos de calidad, por eso para que un sistema sea de gran valor debería ser de gran calidad.

El laboratorio de quimiosensibilidad tumoral en la Universidad de Costa Rica, lo integran diversos doctores que suelen realizar experimentos, en los cuales la función principal es el procesamiento de imágenes. Para automatizar estos experimentos se realizará un sistema que permita optimizar esos análisis, pero debe ser un sistema de calidad que asegure que los resultados son confiables y precisos. Para ello se deben definir ciertos atributos y métricas que permitan evaluar la calidad que ofrecerá el sistema y sus resultados.

## 2. Especificación de requerimientos de software (SyRS)

### 1. Propósito del sistema

Los doctores en microbiología Steve Quirós y Rodrigo Mora, quienes dirigen el laboratorio de quimiosensibilidad tumoral de la Universidad de Costa Rica han contactado a la empresa para desarrollar un sistema que les permita analizar automáticamente imágenes de actividad celular. Por lo que es necesario detectar de manera automática la posición de las células y contarlas, de manera que sea posible generar un informe con la cantidad de células y la posición de cada una, esto con el objetivo de desarrollar tratamientos quimioterapéuticos personalizados a cada paciente y además basados en la información obtenida poder dar un diagnóstico preciso de la reacción de cada paciente con ciertos tratamientos, para poder elegir el de mayor compatibilidad y efectividad en cada uno.

### 2. Alcance del sistema

El alcance del sistema será la automatización de la detección, rastreo y determinación de la línea hereditaria de cada célula en una imagen dada. Permitirá a los funcionarios de la Universidad de Costa Rica cargar miles de imágenes en forma secuencial. El desarrollo de la aplicación será web, para facilitar el uso de la misma; se aprovechará el poder de procesamiento del servidor del laboratorio, donde se realizarán los cálculos de detección.

### 3. Resumen del sistema

#### a) Contexto del sistema

Los dos componentes vitales del sistema serán la aplicación web y el servidor dedicado al procesamiento. La interacción entre los componentes será a través de métodos HTTP. Las imágenes serán cargadas en la aplicación web y transferidas al servidor para su oportuno procesamiento. La transferencia y computación de las imágenes será transparente para el usuario.

b) Función del sistema

La función principal a realizar del sistema, es el análisis automático de imágenes de actividad celular. Es un sistema que permite realizar segmentación de células para así, reconocer la posición de cada célula presente en el video y contar la cantidad de células reconocidas por el sistema.

c) Características de usuario

Este sistema va dirigido al laboratorio de quimiosensibilidad tumoral de la Universidad de Costa Rica, por lo tanto, los usuarios que utilizarán este sistema son doctores en microbiología que pertenecen a este laboratorio. Son usuarios con un conocimiento medio en lo que respecta al uso de aplicaciones en computadora.

#### 4. Requerimientos funcionales

ID	Requerimiento	Prioridad
1	El sistema permitirá la carga de múltiples imágenes en formato PNG, las cuales serán seleccionadas desde la aplicación cliente (los archivos deberán encontrarse en el equipo donde se está ejecutando la aplicación web)	Alta
2	El usuario podrá descargar un archivo en formato CSV con los datos generados como resultado de la respuesta del procesamiento de las imágenes	Alta
3	Todas las funcionalidades del sistema podrán ser utilizadas desde la aplicación web del cliente	Alta
4	La aplicación web, mencionada en el requerimiento 4.3, podrá ser utilizada desde cualquiera de los principales navegadores de Internet (Google Chrome, Firefox, Microsoft Edge, Safari)	Alta
5	El sistema deberá poder utilizarse sin necesidad de instalar ningún tipo de software adicional (a excepción de un navegador web)	Alta
6	Los cálculos para la segmentación de las imágenes se realizarán utilizando los recursos del servidor (nunca los del cliente)	Alta
7	Los resultados del requerimiento 4.6 serán almacenados en archivos	Alta
8	El sistema validará si el formato de las imágenes es válido para su debido procesamiento, aceptando sólo las extensiones de archivos de tipo PNG	Media
9	La aplicación web debe ejecutarse con un delay mínimo de un segundo, siempre y cuando exista una conexión a Internet estable	Media
10	El usuario podrá escoger el nombre del archivo CSV al que se refiere el requerimiento 4.2	Baja
11	Se podrán agregar notas o comentarios, previo a la descarga del archivo CSV al que se refiere el requerimiento 4.2	Baja

#### Justificación de prioridades

1. La carga de imágenes es una de las tareas más importantes del sistema, definido así por el cliente.
2. Obtener los resultados de las predicciones es trascendental para los usuarios del sistema.
3. Entre las funcionalidades que el cliente solicitó dar prioridad se encontraba utilizar el sistema desde un explorador web.
4. Debido al requerimiento anterior, la aplicación web debe poderse ejecutar en los exploradores más utilizados, para asegurar la facilidad de uso y evitar posibles fallos del software.

5. Con el fin de asegurar la facilidad de uso, poder ejecutar las funciones del sistema, sin necesidad de realizar instalaciones extra es vital.
6. Debido a que los dispositivos utilizados para accesar a la aplicación web pueden ser computadoras personales o celulares, el procesamiento de las imágenes debe realizarse en servidores apropiados para la tarea.
7. Debe ser posible, para los especialistas, comprobar la correctitud en la segmentación de imágenes realizada por el sistema.
8. Para evitar fallas en el sistema, solamente se deben aceptar los tipos de archivos conocidos.
9. Con el objetivo de asegurar la atención de los usuarios, la respuesta de la aplicación web debe ser lo más rápida posible.
10. Este requerimiento no afecta en ninguna medida la correctitud, eficiencia, precisión, etc. del sistema, por lo que no es considerado con una prioridad mayor.
11. Igual que el requerimiento anterior, la realización o no de este, no implicará una diferencia notable para el usuario.

## 5. Requerimientos no funcionales

ID	Requerimiento	Prioridad
1	Los resultados de la identificación de células deben ser transferidos del servidor al cliente al finalizar el procesamiento	Alta
2	La aplicación web debe poseer un diseño <i>responsive</i> a fin de garantizar la adecuada visualización en múltiples computadoras personales, dispositivos tablets y teléfonos inteligentes	Alta
3	El nuevo sistema y sus procedimientos de mantenimiento de datos deben cumplir con las leyes y reglamentos de protección de datos médicos	Alta
4	La aplicación, para el cálculo de la segmentación de las imágenes, ejecutada en el servidor, deberá utilizar los recursos del servidor sin generar una sobrecarga	Media
5	El sistema debe asegurar que los datos estén protegidos de accesos no autorizados	Media
6	El sistema deberá contar con manuales de usuario estructurados adecuadamente para el rápido aprendizaje del software	Media
7	El sistema deberá proporcionar mensajes de error informativos y orientados al usuario final	Media
8	El sistema debe poseer interfaces gráficas bien formadas y claras, las cuales no generen confusión	Media
9	Las pruebas de software serán ejecutadas utilizando software especializado para la automatización de pruebas	Media
10	El tiempo de aprendizaje promedio, para un usuario, deberá ser menor o igual a 60 minutos, haciendo uso de los requerimientos 5.6, 5.7, 5.8	Baja
11	La tasa de errores cometidos por el usuario deberá ser menor del 1 % de las transacciones totales	Baja
12	La documentación técnica del software será realizada con herramientas de automatización	Baja

### Justificación de prioridades

1. Para que los usuarios puedan visualizar la segmentación de células en las imágenes, es necesario transferir los archivos de las predicciones a la aplicación del cliente.
2. Debido a la gran variedad de dispositivos que existen actualmente, para una página web es importante ser *responsive* y asegurar la correcta visualización en pantallas de distintos tamaños.
3. Las imágenes utilizadas por el sistema son información sensible, pues pertenecen a pacientes reales.
4. Si se enviara una cantidad de solicitudes que el servidor no podrá procesar, este ocasionará problemas (reinicios y mayores tiempos de espera para el usuario).
5. Debido a la sensibilidad de la información, ningún tercero debería poder acceder sin la autorización necesaria, solamente los especialistas quienes necesiten generar datos con esta.
6. Debido a que los usuarios finales no son profesionales en áreas afines a las ciencias de la computación, los manuales de uso deberán ser lo más claros posibles.
7. Mantener al usuario informado es vital para la correcta utilización de un software, mensajes apropiados ayudan a esto.
8. Por la misma razón del punto 6, a este requerimiento debe prestársele atención.

9. Para asegurar la correcta ejecución y creación de pruebas, además de reducir el esfuerzo para la realización de pruebas, la automatización es vital.
10. La aplicación web debe ser altamente intuitiva, debido al tipo de usuario final.
11. Al cumplir con los requerimientos 5.6, 5.7, 5.8 y 5.10 la tasa de errores debería ser mínima, sin embargo, se debe reducir a lo mínimo posible la cantidad de errores de parte de los usuarios.
12. Para asegurar la facilidad de creación de la documentación técnica, deberá considerarse la utilización de herramientas apropiadas, *Doxxygen*, por ejemplo.

## 6. Requerimientos de rendimiento

ID	Requerimiento	Prioridad
1	El sistema debe ser capaz de operar óptimamente con la carga máxima de imágenes esperada (menor a 150 archivos)	Alta
2	Se deberá contar con una conexión estable y de banda ancha a Internet	Alta
3	En caso de existir un <i>delay</i> en el sistema, este no superará los dos segundos	Media
4	El sistema deberá soportar al menos cinco usuarios enviando solicitudes de procesamiento al servidor	Media
5	El tiempo de respuesta del procesamiento de las imágenes está basado en N imágenes, por lo tanto, depende totalmente de la cantidad de imágenes a procesar. Con un tiempo promedio de 0,3 segundos por imagen	Media
6	La tasa de tiempos de falla del sistema no podrá ser mayor al 0,5 % del tiempo de operación total	Media
7	La aplicación del cliente no ocupará espacio de almacenamiento en la máquina del cliente (a menos de que este desee descargar los resultados)	Baja
8	El tiempo para iniciar o reiniciar el sistema no podrá ser mayor a 90 segundos	Baja

### Justificación de prioridades

1. Este requerimiento se encuentra relacionado con el 4.1, por lo que es fundamental para la ejecución del software.
2. El sistema será accedido solamente a través de Internet, por lo que contar con banda ancha estable de Internet es imperativo.
3. Para mantener la atención del usuario, los tiempos de respuesta deben ser mínimos.
4. Debido a que el sistema es accedido mediante una aplicación web, múltiples sesiones podrían encontrarse abiertas al mismo tiempo, el sistema debe poder responder ante esta situación apropiadamente.
5. A pesar de que el tamaño y complejidad de las imágenes puede variar, la velocidad de procesamiento debe ser la mayor posible, con el fin de generar los datos necesarios rápidamente.
6. La tasa de fallos debe minimizarse lo más posible, asegurando la máxima disponibilidad del sistema.
7. Como consecuencia del requerimiento 4.3, el sistema no ocupará espacio en el dispositivo del cliente, sin embargo tampoco se deberá hacer uso del almacenamiento si no es necesario, y en caso de utilizarlo, liberar el espacio al finalizar.
8. Con el objetivo de asegurar la mejor utilización del tiempo de los profesionales involucrados, el tiempo debe considerarse, sin embargo el estado normal del sistema será encendido, por lo que la prioridad de asegurar este requerimiento es bajo.

## 7. Interfaces de sistema

Las interfaces para la implementación del sistema serán para la comunicación entre módulos propios, específicamente la aplicación web y el servidor, ubicado físicamente en el laboratorio. No será necesario el desarrollo de interfaces para entidades externas, pues no se incluirán componentes de terceros.

## 8. Operaciones de sistema

### a) Requerimientos de integración humano-sistema

El manejo de las imágenes en el sistema es vital, debe cumplirse con la alta cantidad de las mismas, mientras al mismo tiempo se aseguran estándares de seguridad lo suficientemente fuertes como para evitar el extravío de datos. El sistema solo debe permitir al usuario autorizado cargar las imágenes para el experimento y ningún otro usuario podrá descargar o ver información de las mismas, pues en el caso de que las imágenes no se encuentren anonimizadas completamente, podría verse afectada la privacidad del paciente.

### b) Mantenibilidad

Debido a que por la gran cantidad de cálculos necesarios para el procesamiento de las imágenes no se esperará recibir una respuesta inmediata, el aspecto de la estabilidad no es considerada como de alta prioridad. Por lo anterior se está definiendo un uptime para el sistema en un 99.9 %, el cual consideraría como aceptable un downtime diario, semanal, mensual y anual de 1m 26s, 10m 5s, 43m 50s, 8h 46m, respectivamente. Gracias a que se crearán las interfaces necesarias para una interacción limpia entre la aplicación web y el servidor, el número de personas especializadas necesarias para solucionar algún problema que ocurriese será bajo, una o dos personas como máximo para esta función.

### c) Fiabilidad

Para el sistema la fiabilidad tiene un peso medio-bajo. Lo anterior porque la madurez, la tolerancia a fallos y la capacidad de recuperación no son considerados como aspectos vitales para el funcionamiento de las aplicaciones, pues el uso final que se le pretende dar al sistema será más bien intermitente, ya que los experimentos de los médicos no serán ejecutados constantemente.

### d) Estados y modos de sistema

El sistema a desarrollar solamente cuenta con un modo de funcionamiento, al cual se le puede llamar como un modo de usuario. Este modo es en el cual los usuarios finales harán su interacción con el sistema y podrán utilizar todas las funcionalidades implementadas en el sistema.

### e) Características físicas

#### 1) Requerimientos físicos

El sistema requiere que el equipo donde se ejecutará, cuente con acceso a internet y que además, esta conexión a la red sea estable para que el sistema tenga un correcto funcionamiento y que este sea lo más fluido posible.

#### 2) Requerimientos de adaptabilidad

Solamente es necesario que el equipo donde correrá el sistema cuente con alguno de los navegadores compatibles que fueron antes mencionados.

### f) Condiciones ambientales

El laboratorio donde se encontrarán los servidores, contará con una temperatura regulada mediante aire acondicionado, con un rango entre 21 y 23 grados centígrados, lo cual asegura una temperatura deseable para el correcto funcionamiento de las computadoras.

La humedad del ambiente debe mantenerse entre 45 % y 50 % para asegurar la operación segura de los servidores. Si la humedad se encuentra muy alta, esto puede generar problemas de corrosión a nivel de hardware.

Debido a que la actividad humana es la que genera mayor cantidad de contaminantes, es recomendado limitar el acceso al sitio solamente a aquellas personas vitales para el correcto funcionamiento del sistema, y solamente en casos donde esto sea necesario.

*g) Seguridad del sistema*

Para el acceso al sistema será necesario contar con un nombre de usuario y una contraseña, esto será utilizado para limitar el acceso a la información y a las funciones a solamente los usuarios con los permisos adecuados. La protección de los datos es fundamental, pues debe asegurarse el anonimato de los pacientes, para lograrlo los datos serán transmitidos de forma segura entre el cliente y el servidor. Una vez procesadas las imágenes estas serán eliminadas del sistema. Se implementarán logs, estos contendrán información de todas las actividades realizadas por los usuarios.

*h) Manejo de la información*

Las imágenes recibidas serán transmitidas mediante una conexión segura al servidor, el cual realizará los cálculos empleando las fotografías suministradas por el usuario, las cuales serán eliminadas del sistema luego de finalizado el trabajo.

El reporte generado será transmitido nuevamente al cliente utilizando la misma conexión. Para el log, solo se guardarán datos de usuario, hora, actividad solicitada y tiempo de ocupación del servidor.

*i) Políticas y regulaciones*

Para esta primera etapa del proyecto se desarrollará el sistema para científicos costarricenses, por lo que el idioma será el español, con la posibilidad de aumentar el soporte de idiomas en el futuro.

Es necesario conocer el usuario que solicita una tarea, por lo que esta información debe ser recopilada, sin embargo no será compartida o expuesta de ninguna manera, solamente se encontrará a disposición de los encargados del sistema.

*j) Ciclo de vida del sistema*

Para asegurar que el sistema tenga un ciclo de vida bastante amplio y un uso correcto durante este ciclo de vida, el equipo de desarrollo se compromete a realizar un entrenamiento previo al lanzamiento del sistema, sobre el uso básico de las funcionalidades.

Se brindarán todo tipo de datos para poder contactar al equipo en caso de necesitar cierto tipo de ayuda o soporte respecto al sistema desarrollado.

*k) Embalaje, manipulación, envío y transporte*

Por el tipo de sistema y en el ambiente que se va a realizar, estos tipos de requerimientos no aplican.

*l) Verificación*

Para cada funcionalidad desarrollada y presente en el sistema, se le aplicarán pruebas de correctitud, esto con el fin de revisar que los resultados de todas las operaciones que se podrán realizar, sean 100 % correctos.

Se realizarán también pruebas de unidad para verificar el correcto funcionamiento de cada parte del sistema y luego se aplicarán pruebas de integridad para revisar los resultados y el correcto funcionamiento del sistema por completo.

*m) Suposiciones y dependencias*

Se supone que en todos los equipos o todas las máquinas en las cuales se va a utilizar este sistema, existe al menos un navegador compatible instalado y en funcionamiento, en el cual se pueda correr el sistema sin inconvenientes. También se supone que los equipos se pueden conectar a alguna red de internet.

Se supone que el laboratorio donde estará el equipo en el cual se ejecutará el sistema, existe una conexión a internet confiable, estable y con la capacidad o velocidad suficiente para correrlo y realizar todas las solicitudes necesarias para contribuir de la mejor manera al trabajo de quienes lo utilicen.

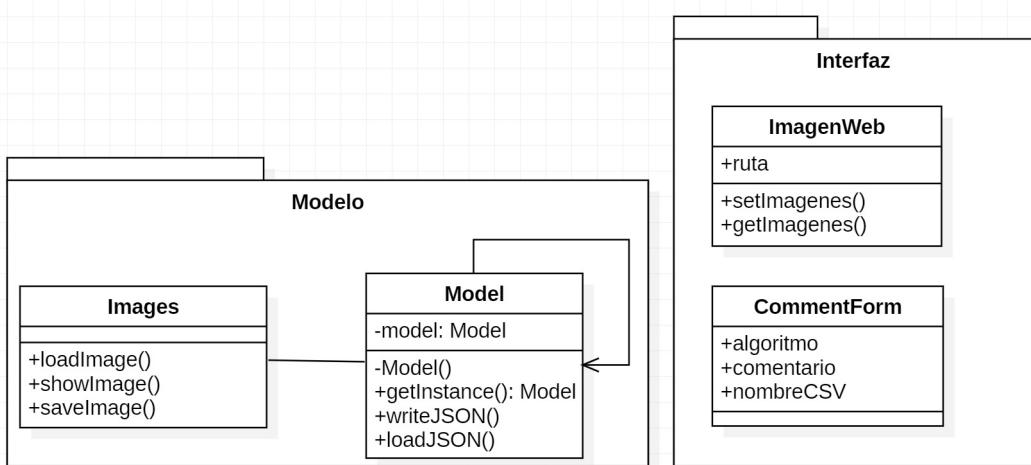
### **3. Estándar de codificación y documentación**

Para el desarrollo del proyecto, se tomó la decisión de usar PEP8 como estándar de codificación y PEP257 como estándar de documentación. Estos estándares se utilizarán en todo el código presente para el sistema. Se decidió por estos estándares porque poseen la suficiente documentación que permite al equipo conocer todos las indicaciones para aplicarlo de la mejor manera posible.

Se escogieron además porque son la mejor opción o los mejores estándares a seguir cuando se está usando Python. Ambos PEP's tienen sus propias librerías que se pueden descargar e instalar, para usarlas como herramientas que ayudarán en la revisión y en el control del cumplimiento de ambos estándares, ya que, dado un archivo con código, estas librerías lo revisan en su totalidad y en caso de que se haya dado algún error en codificación o documentación donde de incumpla con las reglas de los mismos, se mostrará el tipo de error y la sección de código donde se dio.

## 4. Diagramas

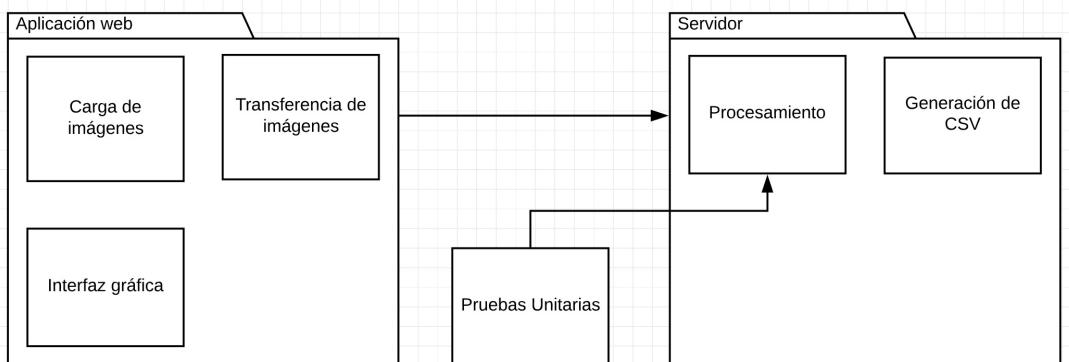
### 4.1. Diagrama UML



Para este diseño de clases presentado para la primera iteración del proyecto, se utilizó el patrón de diseño llamado Singleton. Este patrón se utiliza en cualquier sistema cuando en él, solamente se necesita una instancia de una clase en específica, así que al utilizar este patrón, se asegura que en todo momento se utilizará la misma instancia de esta clase seleccionada. Para este caso, de la clase llamada “Model” solamente se utilizará una instancia de ella, la cual representa el modelo cargado con la librería de Keras que permitirá el procesamiento de las imágenes brindadas por el usuario y el funcionamiento del sistema.

Los patrones de diseño consultados se pueden encontrar en: <https://devexperto.com/patrones-de-diseno-software/>

### 4.2. Diagrama de Componentes



## 5. Actividades

Actividad	Responsable	Iteración
Corrección y documentación de las no conformidades	Ignacio Cantillo	Sprint 1, 2, 3
Definición de las condiciones de aceptación del producto	Daniel Delgado	PoC
Definición de las necesidades de soporte del usuario	Daniel Delgado	Sprint 3
Evaluación de la satisfacción del usuario con el producto final	Daniel Delgado	Fin del proyecto
Construcción y revisión de los tests unitarios	Kevin Montoya	Sprint 1, 2, 3
Construcción y revisión de los tests de integración	Kevin Montoya	Sprint 2, 3
Evaluación de las actividades del proyecto respecto a los requerimientos	Ignacio Cantillo	Sprint 1, 2, 3
Evaluación de las métricas realizadas, respecto a las planificadas	Ignacio Cantillo	Sprint 2, 3

## 6. Pruebas Unitarias

### 1. Carga de imágenes en Keras

- Verificación de carga correcta de imagen con todos los píxeles en negro, utilizando los arrays de Keras para comprobar el valor total de todos los píxeles.
- Verificación de carga correcta de imagen con todos los píxeles en blanco, utilizando los arrays de Keras para comprobar el valor total de todos los píxeles.

Prueba:

```
def test_BlackPixels(self):  
    image = Images()  
    self.assertEqual(image.loadImage("BlackPixels.jpg"), 0)  
    print("----_end_test_----")  
  
def test_WhitePixels(self):  
    image = Images()  
    self.assertEqual(image.loadImage("WhitePixels.jpg"),  
    2550000)  
    print("----_end_test_----")
```

Validación:

```
def loadImage(self, fileName):  
    self.im = kImage.load_img(fileName,  
                            color_mode="grayscale",  
                            target_size=None)  
    self.imageArray = kImage.img_to_array(self.im)  
  
    # For the tests  
    total = 0  
    for i in range(self.imageArray.shape[0]):  
        for j in range(self.imageArray.shape[1]):  
            total += int(self.imageArray[i][j])  
  
    return total
```

## 2. Aplicación Web

- a) Utilización de algoritmo usado en carga de imágenes con un valor de ejemplo, verificando que este brinde un resultado el cual no sea un valor nulo.

Prueba:

```
def test_listdT(self):
    print("test_listdT")
    result = run.listdT(6)
    self.assertIsNotNone(result, "None")
    print("----end test----")
```

Validación:

```
def listaT(tam):
    cont = 1
    l = []
    if (tam == 0):
        raise ValueError('Lista_de_Imagenes_Vacia')
    while cont < tam:
        l = l + [cont]
        cont += 1
    return l
```

- b) Utilización de algoritmo para la aplicación web con la función de carga de imágenes donde se implementa el algoritmo con un valor dado y se verifica que el resultado sea exactamente el esperado y no un resultado erróneo.

Prueba:

```
def test_generacionLista(self):
    print("test_generacionLista")
    result = run.listaT(10)
    self.assertListEqual(result, [1, 2, 3, 4, 5, 6, 7, 8, 9])
    print("----end test----")
```

Validación:

```
def listaT(tam):
    cont = 1;
    l = []
    if (tam == 0):
        raise ValueError('Lista_de_Imagenes_Vacia')
    while cont < tam:
        l = l + [cont]
        cont += 1
    return l
```

- c) Verificación del caso donde el usuario no haya cargado ninguna imagen, pero presione el botón de cargar en la página, dando como resultado la utilización de un algoritmo con una lista vacía de imágenes, es por esto que se valida que el sistema tome esto como un error.

Prueba:

```
def test_listavacia(self):
    print("test_listavacia")
    self.assertRaises(ValueError, run.listaT, 0)
    print("----end test----")
```

Validación:

```
def listaT(tam):
    cont = 1;
    l = []
```

```

if (tam == 0):
    raise ValueError('Lista_de_Imagenes_Vacia')
    while cont < tam:
        l = l + [cont]
        cont += 1
    return l

```

- d) Validación en el método para la creación del archivo de resultados (en formato CSV), el cual es producto del procesamiento de las imágenes, esto para verificar que se atrape el error específico donde el usuario olvida llenar alguno de los campos para la generación de dicho archivo.

Prueba:

```

def test_createCSV(self):
    print("test_createCSV")
    self.assertRaises(ValueError, run.escribirCSV,
                      "", "", "")
    print("----end test----")

```

Validación:

```

def escribirCSV(nombre, noAlgoritmo, comentario):
    if (nombre=="" or noAlgoritmo=="" or comentario==""):
        raise ValueError('Todos los campos'
                         + 'deben estar llenos')
    diccionario = {'Algorithm_Number': [numAlg],
                   'Obj_quantity': [24],
                   'Precision': [42],
                   'Notes': [comentario]}
    listaColumnas = ['Algorithm_Number',
                     'Obj_quantity',
                     'Precision',
                     'Notes']
    df = pd.DataFrame(diccionario,
                      columns=listaColumnas)
    df.to_csv('ruta' + nombre + '.csv')
    print("Creado con Exito")

```

### 3. Modelo de Segmentación de Imágenes

- a) Verificación en la función principal del modelo para atrapar un error en caso de que se envíe como parámetro una lista de imágenes vacía, esto en el caso que haya un error en la carga de las mismas

Prueba:

```

def test_predict(self):
    print("test_predict")
    self.assertRaises(ValueError, run.predict, [])
    print("----end test----")

```

Validación:

```

...
if len(ruta) == 0:
    raise ValueError("Error al cargar las imágenes")
...

```

- b) Verificación en la función predict para verificar que la carga de los pesos para el uso del modelo se ejecute correctamente

Prueba:

```

def test_predictWeight(self):
    print("test_predictWeight")
    self.assertRaises(ValueError, run.predict, [])
    print("----end----")

```

Validación:

```

...
    if len(weights_path) == 0:
        raise ValueError("Error en la carga de los pesos")
...

```

- c) Se verifica en la función de la carga de los datos que se reciba una obtención certera de las imágenes

Prueba:

```

def test_load_test_data(self):
    print("test_load_test_data")
    self.assertRaises(ValueError, run.load_test_data, [])
    print("----end----")

```

Validación:

```

def load_test_data(image_path):
    raw = []
    image_filename = dict()
    count = 0

    if len(image_path) == 0:
        raise ValueError ("Error al obtener las imagenes")
...

```

- d) Se verifica en la función de pre procesamiento que las imágenes cargadas anteriormente se envíen de forma correcta

Prueba:

```

def test_preprocess(self):
    print("test_preprocess")
    self.assertRaises(ValueError, run.preprocess, [])
    print("----end----")

```

Validación:

```

def preprocess(imgs):
    if len(imgs) == 0:
        raise ValueError ("Error en las imagenes")
    imgs_p = np.ndarray((len(imgs), img_rows, img_cols),
                        dtype=np.float32)
    for i in range(len(imgs)):
        imgs_p[i] = imgs[i].reshape((img_rows,
                                    img_cols))/255.

    imgs_p = imgs_p[... , np.newaxis]
...

```

## 7. Estándar de Codificación

El estándar de codificación utilizado en el proyecto de Segmentación de Células es el estándar de Python llamado pep8. Este es una guía o estilo para codificación en el proyecto, es uno de los estándares más utilizados para este lenguaje de programación, ya que trae muchos beneficios al código, entre ellos se pueden mencionar:

- Orden al Código
- Facilidad en la Documentación
- Mayor Comprensibilidad
- Facilidad de Lectura
- Consistencia en el Código

Su uso es muy simple y se puede utilizar siguiendo los siguientes pasos:

1. Instalación: Se debe ejecutar el siguiente comando

```
pip install pep8
```

2. Uso: Luego de instalarlo se pueden utilizar las siguientes opciones

- Descripción de errores

```
pep8 --first ejemplo.py
```

- Ejemplos de como dar solución a los errores

```
pep8 --show-source --show-pep8 ejemplo.py
```

Algunas veces la salida del pep8 solo muestra una ocurrencia por cada tipo de error y a medida que se vayan solucionando se van mostrando los demás, hasta que se hayan solucionado todos los errores y la salida no contenga ningún error.

## 8. Validación de Diseño

Unidad de Diseño	Requerimiento
Images	La segmentación de imágenes se realizará en el servidor y los resultados almacenarse en archivos
Model	Debe optimizarse la utilización de los recursos al máximo
ImagenWeb	El sistema permitirá la carga de imágenes dada o seleccionada una carpeta ubicada en el equipo donde se está utilizando tal sistema
CommentForm	Se podrán agregar notas o comentarios al reporte, previo a la descarga del archivo. Este campo asignado para este fin, acepta caracteres alfabéticos, numéricos y especiales

## 9. Validación de la Implementación

Ítem de Implementación	Requerimiento
Carga de Imágenes	El sistema debe ser capaz de operar adecuadamente con la carga esperada de imágenes
Procesamiento de Imágenes	La aplicación del servidor deberá consumir los recursos del servidor eficientemente
Carga y escritura de archivos CSV	Se le permitirá al usuario escoger el nombre del archivo en el campo asignado para ello, el cual acepta caracteres alfabéticos, numéricos y especiales
Carga de Modelo	El sistema debe ser capaz de procesar N transacciones por segundo

## 10. Evaluación de las Actividades del Proyecto respecto a los requerimientos

A lo largo del proyecto se han recaudado diversos requerimientos que se han cumplido de manera exitosa, satisfaciendo las necesidades que prioriza el cliente. En la primer iteración o avance se solicitó como necesidad prioritaria la carga de imágenes, la cuál funciona completamente y de manera eficiente en la versión actual del sistema.

Luego de esto se continua trabajando con los siguientes requerimientos expuestos por el usuario, por lo que se trabaja en el desarrollo de segmentación de las imágenes en fondo y no fondo; además de esto se calcula la métrica de precisión de dicha segmentación.

## 11. Métricas

Para efecto de este avance, se escogió cuantificar métricas de los siguientes requerimientos:

1. Se le permitirá al usuario escoger el nombre del archivo en el campo asignado para ello, el cual acepta caracteres alfabéticos, numéricos y especiales
2. El sistema permitirá la carga de imágenes dada o seleccionada una carpeta ubicada en el equipo donde se está utilizando tal sistema

Para ambos requerimientos se decidió cuantificar el tiempo que duran en ejecutar las funciones o los módulos encargados de realizar dichas acciones.

En este caso, como herramienta se decidió usar un simple timer en la función encargada de crear el archivo CSV una vez que se le dan los parámetros como el nombre del archivo que se le desea dar al archivo que se va a descargar.

Utilizar esta herramienta es muy sencillo, solamente se debe de importar la librería time con el siguiente código: "import time". Una vez importada la librería, se debe ubicar la función de la cual se quiere medir su tiempo. Cuando se tiene el código fuente de la función buscada, en la primer línea de la función se debe poner algo como lo siguiente: "start\_time = time.time()"; con esto se almacena en la variable *start\_time* el tiempo exacto en el cual se empezó a ejecutar la función. Y por último, para saber el tiempo total de ejecución, se debe usar lo siguiente como última línea de código de la función: "total = time.time() - start\_time". En esta variable llamada *total*, se almacena el tiempo que duró la ejecución de la función.

Los resultados al realizar las métricas, fueron los siguientes:

- Para el primer requerimiento escogido, el sistema mostró un comportamiento muy bueno, se dieron tiempos entre 0,05 y 0,10 segundos.
- Para el segundo, dado una cantidad considerable de imágenes a cargar, la ejecución tardó alrededor de 0,30 y 0,50 segundos.

En relación a los límites:

- Para el primer requerimiento, el sistema debería de tener un tiempo mínimo de ejecución como el mostrado en las métricas realizadas, o sea, debería tener un tiempo mínimo de 0,05 segundos; esto es porque para esta prueba, se utilizaron pocos datos a introducir en el archivo. Como tiempo máximo y claramente dependiendo de la cantidad de datos que se vayan a almacenar en el archivo, la ejecución no debería tardar más de 1,5 segundos, ya que se considera un tiempo suficiente para que se haga la creación del archivo con cualquier cantidad de datos y porque se considera un tiempo con el cual el usuario pueda seguir satisfecho con la fluidez del sistema.
- Para el requerimiento #2, el tiempo de ejecución depende mucho de la cantidad de imágenes que vayan a ser cargadas, ya que, puede que se carguen 10 imágenes o igualmente, en otro momento puede que se carguen 100, 1000 ó más imágenes, dependiendo de la necesidad del momento. Partiendo de este dato, el tiempo mínimo esperado está alrededor de 0,20 segundos, esto puede que se dé si se cargan pocas imágenes. En caso contrario, de que sea mucha cantidad de imágenes, se esperaría que la ejecución de la función no tarde más de 5 segundos. Igualmente que la anterior, este tiempo máximo se considera suficiente para la ejecución y además, un tiempo que satisface al usuario con respecto a la fluidez del sistema.

## 12. Utilización de Jenkins

Jenkins es un servidor para la automatización del software, provee cientos de plugins para el desarrollo, despliegue y automatización de un proyecto.

Para la utilización de esta herramienta en un proyecto de software lo primero que se debe hacer es descargar e instalar. Una vez este se encuentra en la máquina, se puede hacer uso por primera vez accediendo desde un explorador a la dirección `localhost:8080`. La cual pedirá crear el primer usuario. Al completar el registro se podrá crear un nuevo item, estos items se refieren a una tarea de automatización.

En el caso del presente proyecto, se creó un item para comprobar que las pruebas escritas fueran exitosas. Se cuenta con gráficos para mostrar la historia de los *builds*, sin embargo, no fue posible conectarlo a GitHub para recibir las notificaciones de cuando se realizaba un push, pues para crear un OAuth para una aplicación como Jenkins se necesitaba a la herramienta siendo ejecutada en un servidor.

A pesar de lo anterior, es claro que Jenkins presenta funcionalidades importantes para desarrolladores y facilita muchísimo el trabajo gracias a la automatización de un producto open source.

## 13. Utilización de Git Hooks

Git proporciona la maravillosa funcionalidad de hooks, los cuales son similares a los triggers en las bases de datos. Haciendo uso de esto, se logró ejecutar un hook antes de cada commit de usuario, para asegurar que las pruebas unitarias pasaran antes de permitir realizar el commit y/o push.

Para crear un hook se debe acceder a la carpeta `.git` (usualmente se encuentra oculta), la cual existe dentro de cualquier repositorio Git. En el path `.git/hooks` se debe agregar un archivo que defina la tarea que se ejecutará, en el caso del presente proyecto es un pre-commit, o sea, se ejecuta antes de cada commit.

Dentro de este archivo se ejecuta el código de `test_run.py`. Si todas las pruebas pasan se permitirá hacer un commit, de lo contrario se mostrará un mensaje de error al usuario con la información para solucionar el problema, además después de cada commit se crea un reporte de pruebas en formato XML, el cual puede ser utilizado por Jenkins para la generación de estadísticas.

## 14. Precisión

Para evaluar la precisión de la segmentación se utilizó el coeficiente de dice o de Sørensen, el cual se puede usar para evaluar la similitud estructural entre dos imágenes binarias.

$$S = \frac{2|U \cap V|}{|U| + |V|}$$

$|U \cap V|$ : las barras denotan la cantidad de pixeles a los que se les aplicó la operación AND y dio como resultado 1, la operación se realiza con el pixel  $i, j$  de ambas matrices a la vez.

$|U| + |V|$ : el denominador se refiere a la suma de los pixeles con valor 1 de cada imagen.

Las imágenes utilizadas para comprobar la precisión fueron las siguientes:

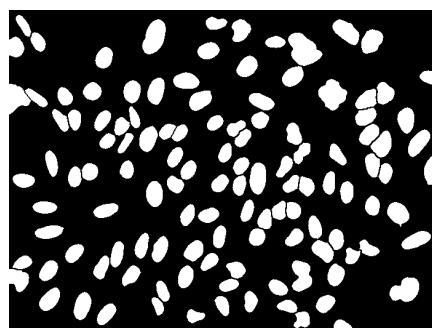


Figura 1: Imagen *groundtruth* suministrada por el profesor

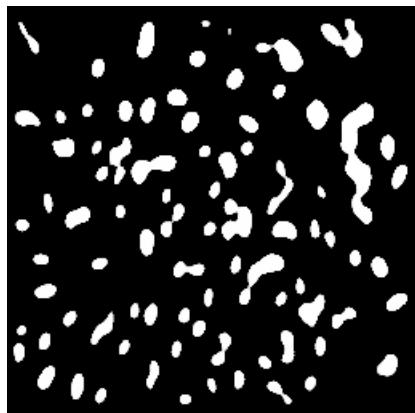


Figura 2: Imagen segmentada por el algoritmo

El resultado del coeficiente fue un promedio del 83 % para la precisión en el segmentado de las imágenes. Los archivos con los cuales se calculó este valor se encuentran en el repositorio de GitHub, en las carpetas *Predictions* y *GTResized*.

## 15. Pruebas para Avance 3

### 15.1. Clases de Equivalencia

ID CE	Módulo	Clase de Equivalencia	Válida
1	Segmentación	Directorio con imágenes válidas	Válida
2	Segmentación	Directorio vacío	No Válida
3	Etiquetado de Células	Lista con objetos de las células segmentadas y lista de centroides	Válida
4	Coloreado de Células	Lista con objetos de las células segmentadas	Válida
5	Generación CSV	Lista de resultados vacía	No Válida
6	Conteo de Células	Imágenes raw	Válida
7	Centroide	Array de célula vacío	No Válida
8	Tiempo de Ejecución	Lista de imágenes	Válida

## 15.2. Diseño de Pruebas

<b>Id. de la prueba</b>	<b>Tipo de prueba</b>	<b>Descripción</b>	<b>Precondiciones</b>	<b>Resultados Esperados</b>
1	Unitaria	Verificar la existencia de la célula recortada previo a la búsqueda de su centro	Imágenes correctamente cargadas y segmentadas	Retorno del centro de la célula
2	Unitaria	Revisar que se dé la lista con los datos de las células para su conteo	Células correctamente segmentadas	Retorno de la cantidad de células en cada imagen
3	Unitaria	Verificación de la existencia de un centro válido de una célula para luego etiquetarla	Células correctamente segmentadas y reconocimiento correcto de los centros	Células etiquetadas correctamente
4	Unitaria	Revisión de la validez de la ruta previo al cálculo del tiempo de ejecución	Inicio correcto de la segmentación de células	Retorno de un tiempo aproximado de ejecución
5	Integración	Verificar que se muestren las imágenes luego de la segmentación de las células	El directorio de imágenes debe ser una ruta existente en el equipo. Imágenes correctas y correctamente cargadas en la página web	Imágenes de las células segmentadas
6	Integración	Probar el coloreado de las células	Células segmentadas correctamente	Imágenes de las células coloreadas
7	Integración	Probar el etiquetado de las células	Células segmentadas correctamente	Imágenes de las células etiquetadas
8	Integración	Verificar la generación del archivo CSV para la muestra de datos al usuario	Digitación correcta de los datos brindados por el usuario	El archivo CSV que incluya los datos brindados por el usuario
9	Sistema	Segmentación, coloreado y etiquetado de células	Existencia de ruta dada por el usuario. Correctitud en el formato de las imágenes de células a segmentar.	El archivo CSV que incluya los datos brindados por el usuario
10	Sistema	Generación del archivo CSV	Digitación correcta de los datos brindados por el usuario	El archivo CSV con el informe de los datos resultantes de la segmentación

### 15.3. Asociación de Pruebas con Clases de Equivalencia

Clase de Equivalencia	Id. de la prueba
CE 7	1
CE 6	2
CE 3	3
CE 8	4
CE 1	5
CE 4	6
CE 3	7
CE 5	8
CE 2	9
CE 5	10

### 15.4. Pruebas

#### 1. Pruebas Unitarias

##### Prueba 1:

Prueba:

```
def test_getCentroide(self):
    print("test_getCentroide")
    self.assertRaises(ValueError, cellPostProcess.getCellCenter, [])
    print("----end----")
```

Validación:

```
def getCellCenter(immat, X, Y):
    if immat == None:
        raise ValueError("Error al buscar la imagen")
    m = np.zeros((X, Y))
    for x in range(X):
        for y in range(Y):
            m[x, y] = immat[(x, y)] != 0
    m = m / np.sum(np.sum(m))
    ...
    ...
```

##### Prueba 2:

Prueba:

```
def test_countCells(self):
    print("test_countCells")
    self.assertRaises(ValueError, cellPostProcess.countCells, [])
    print("----end----")
```

Validación:

```
def countCells(clusteredArray, X, Y):
    if len(clusteredArray) == 0:
        raise ValueError("Error. La lista esta vacia")
    numCells = 0
    for x in range(X):
        for y in range(Y):
            if clusteredArray[x, y] > numCells:
                numCells = clusteredArray[x, y]
    return numCells
```

### Prueba 3:

Prueba:

```
def test_paintLabel(self):
    print("test_paintLabel")
    self.assertRaises(ValueError, cellPostProcess.paintLabel, [])
    print("----end test----")
```

Validación:

```
def paintLabel(immat, center, cellNumber):
    if center == None:
        raise ValueError("Error_El centro dado no es valido .")
    cellNumberLen = len(str(abs(cellNumber)))
    # To center the label in cell
    center = (center[0] - (cellNumberLen - 1) * 2, center[1])
    for i in range(0, cellNumberLen):
        printingNum = int(str(cellNumber)[i])
        if printingNum == 1:
            for y in range(-4, 5):
                immat = paintLabelAux(immat, center[0] + (i * 4), center[1] + y)
    ...
```

### Prueba 4:

Prueba:

```
def test_getAproxExecTime(self):
    print("test_getAproxExecTime")
    self.assertRaises(ValueError, cellPostProcess.getAproxExecTime, [])
    print("----end test----")
```

Validación:

```
def getAproxExecTime(path_):
    if len(path_) == 0:
        raise ValueError("Error_La ruta no es valida")
    approxExecTime = 0
    if path_.isdir(path_):
        for i in listdir(path_):
            img = Image.open(path_ + '\\' + i)
            if img.size[0] < 400 or img.size[1] < 400:
                approxExecTime += img.size[0] * 0.003 + img.size[1] * 0.003
            else:
                approxExecTime += img.size[0] * 0.02 + img.size[1] * 0.02
    else:
        img = Image.open(path_)
        approxExecTime = img.size[0] * 0.003 + img.size[1] * 0.003

    return approxExecTime
```

## 2. Pruebas de Integración

### Prueba 5:

Prueba:

```
def test_cellPostProcessImage(self):
    print("test_cellPostProcessImage")
    self.assertRaises(ValueError, cellPostProcess.cellPostProcess,
                      "failPath.png", "failName.png", False)
    print("----_end_test_----")
```

Validación:

```
def cellPostProcess(image_path, imageName, labelCells = False):
    try:
        img = Image.open(image_path)
    except:
        raise ValueError("Error")
    rgbimg = Image.new("RGBA", img.size)
    rgbimg.paste(img)
    immat = rgbimg.load()
    ...
```

### Prueba 6:

Prueba:

```
def test_cellPostProcess(self):
    print("test_cellPostProcessImage")
    result, result2 = cellPostProcess.cellPostProcess("C:/Users/Kevin_MM/
-----eclipse-workspace/SegmentacionCelulas/preds/1_pred.png",
                                                    "1_pred.png")
    self.assertEqual(result2, [(0, 0), (9.989672977624785, 56.07401032702241),
                             (25.17663043478261, 139.02989130434779), (36.36567164179104, 35.5),
                             (54.22727272727273, 1.3181818181818183), (68.45454545454545,
                             241.43734643734643), (95.24411134903642, 174.19914346895084),
                             (90.3141592920354, 136.24557522123897), (107.66666666666666,
                             254.33333333333331), (135.1878453038674, 127.83057090239413),
                             (131.00704225352115, 251.5140845070423), (151.60869565217394,
                             175.0517598343685), (154.2447325769854, 73.20907617504052),
                             (159.4927536231884, 208.39420289855082), (182.0939947780679,
                             184.45430809399474), (189.18971061093248, 211.62700964630227),
                             (192.0176470588235, 249.61764705882356), (200.09546925566343,
                             127.80097087378648), (203.63448275862066, 171.08965517241379),
                             (225.69747899159665, 204.97058823529403), (236.5296912114014,
                             39.011084718923215), (238.8961218836565, 127.13850415512469),
                             (251.2214285714286, 195.45000000000002), (255.0, 30.0)])
    print("----_end_test_----")
```

Validación:

```
def cellPostProcess(image_path, imageName, labelCells = False):
    try:
        img = Image.open(image_path)
    except:
        raise ValueError("Error")
    rgbimg = Image.new("RGBA", img.size)
    rgbimg.paste(img)
    immat = rgbimg.load()
```

```

(X, Y) = rgbimg.size
m = np.zeros((X, Y))
slicedIm = np.zeros((X, Y))

for x in range(X):
    for y in range(Y):
        # To find the clusters
        if immat[(x, y)] != (0, 0, 0, 255):
            m[x, y] = 1

# Making clusters
lw, num = measurements.label(m)
np.set_printoptions(threshold=np.nan)

# Getting number of cells
cellsCount = countCells(lw, X, Y)

# This is an array containing the area of each cell, in order
# The first element represents the image black area
cellsArea = measurements.sum(m, lw, index=arange(lw.max() + 1))
cellsCenter = [(0, 0)]

for cell in range(1, cellsCount+1):
    cellColor = getRandomColor()
    for y in range(Y):
        for x in range(X):
            if lw[x, y] == cell:
                slicedIm[x, y] = 1
                if labelCells: immat[(x, y)] = cellColor
    # Paints a red pixel the center of the cell
    cellCenter = getCellCenter(slicedIm, X, Y)
    cellsCenter.append(cellCenter)
    immat = paintLabel(immat, cellCenter, cell)
    slicedIm = np.zeros((X, Y))

rgbimg.save('postProcess\\' + 'post_' + imageName)
rgbimg.save('static\\img\\' + 'post_' + imageName)

return cellsArea, cellsCenter

```

### Prueba 7:

Prueba:

```

def test_paintLabelIndex(self):
    print("test_paintLabelIndex")
    self.assertRaises(IndexError, cellPostProcess.paintLabel,
[0,0,0], [100], 100)
    print("----end test----")

```

Validación:

```

def paintLabel(immat, center, cellNumber):
    if center == None:
        raise ValueError ("Error. El centro dado no es valido.")
    cellNumberLen = len(str(abs(cellNumber)))
    # To center the label in cell
    center = (center[0]-(cellNumberLen-1)*2, center[1])
    for i in range(0, cellNumberLen):

```

```

printingNum = int(str(cellNumber)[ i ])
if printingNum == 1:
    for y in range(-4, 5):
        immat = paintLableAux(immat, center[0]+(i*4), center[1]+y)
elif printingNum == 2:
    # Horizontal lines
    for x in range(-1, 2):
        immat = paintLableAux(immat, center[0]+x+(i*4), center[1]-4)
        immat = paintLableAux(immat, center[0]+x+(i*4), center[1])
        immat = paintLableAux(immat, center[0]+x+(i*4), center[1]+4)
    # Vertical lines
    for y in range(-3, 0):
        immat = paintLableAux(immat, center[0]+1+(i*4), center[1]+y)
        immat = paintLableAux(immat, center[0]-1+(i*4), center[1]-y)
elif printingNum == 3:
    # Horizontal lines
    for x in range(-1, 2):
        immat = paintLableAux(immat, center[0]+x+(i*4), center[1]-4)
        immat = paintLableAux(immat, center[0]+x+(i*4), center[1])
        immat = paintLableAux(immat, center[0]+x+(i*4), center[1]+4)
...

```

### Prueba 8:

Prueba:

```

def test_escribirCSVDatosCapturados(self):
    print("test_escribirCSVDatosCapturados")
    self.assertRaises(ValueError, run.escribirCSV, None, None,
[], [], "failPath.png")
    print("-----end-----")

```

Validación:

```

def escribirCSV(nombre, noProcedimiento, cellsArea, cellsCenter, imageName):
    if (nombre == "" or noProcedimiento == ""):
        raise ValueError('Todos los campos deben estar llenos')
    elif (cellsArea == [] or cellsCenter == []):
        raise ValueError('Datos no validos')
...

```

### 3. Pruebas de Sistema

Prueba 9 (Selenium):

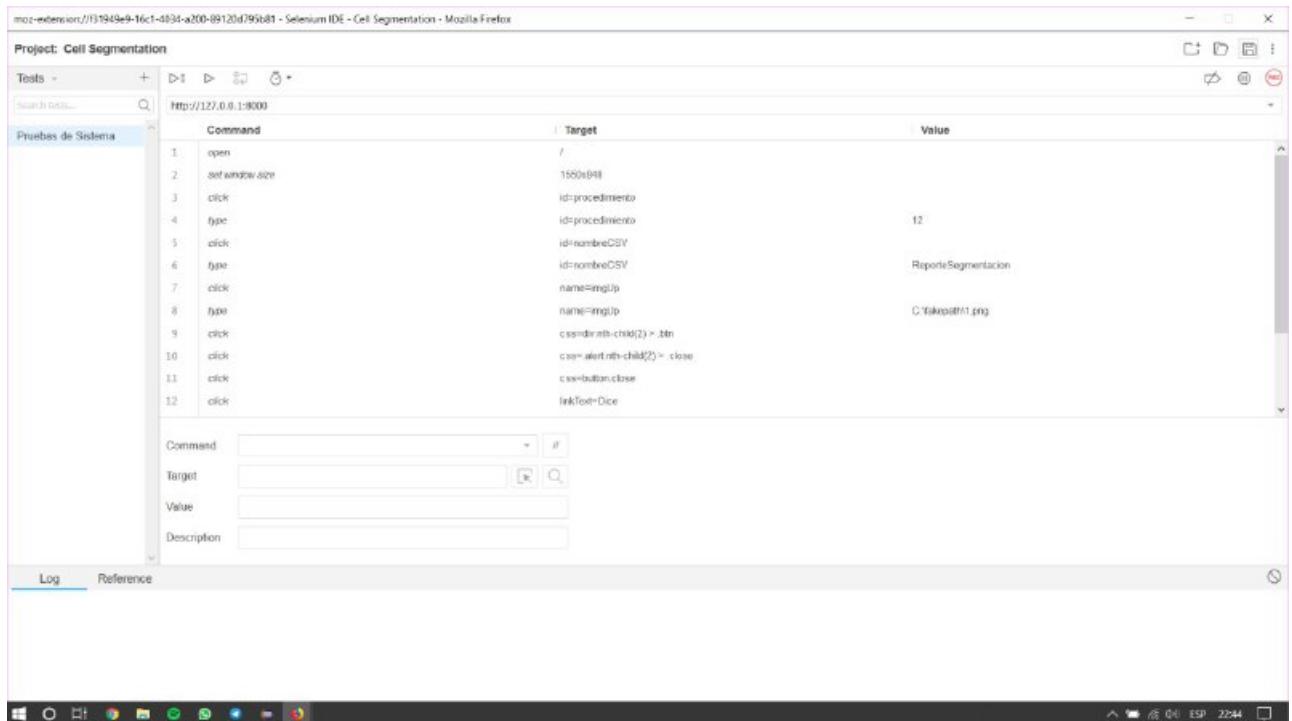


Figura 3: Utilización Selenium

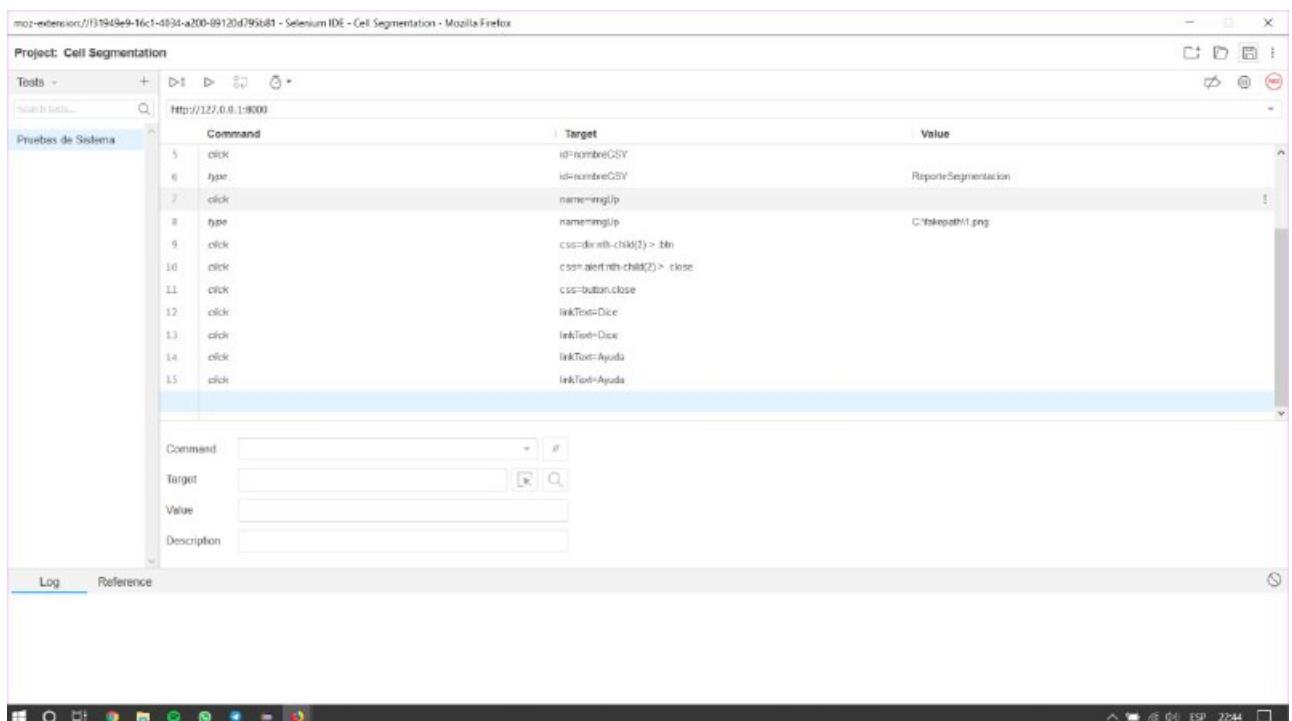


Figura 4: Utilización Selenium

## Prueba 10:

Prueba:

```
def test_escribirCSVDatos(self):
    print("test_escribirCSVDatos")
    self.assertRaises(ValueError, run.escribirCSV, "failPath.png", 1, [], [])
    print("----end test----")
```

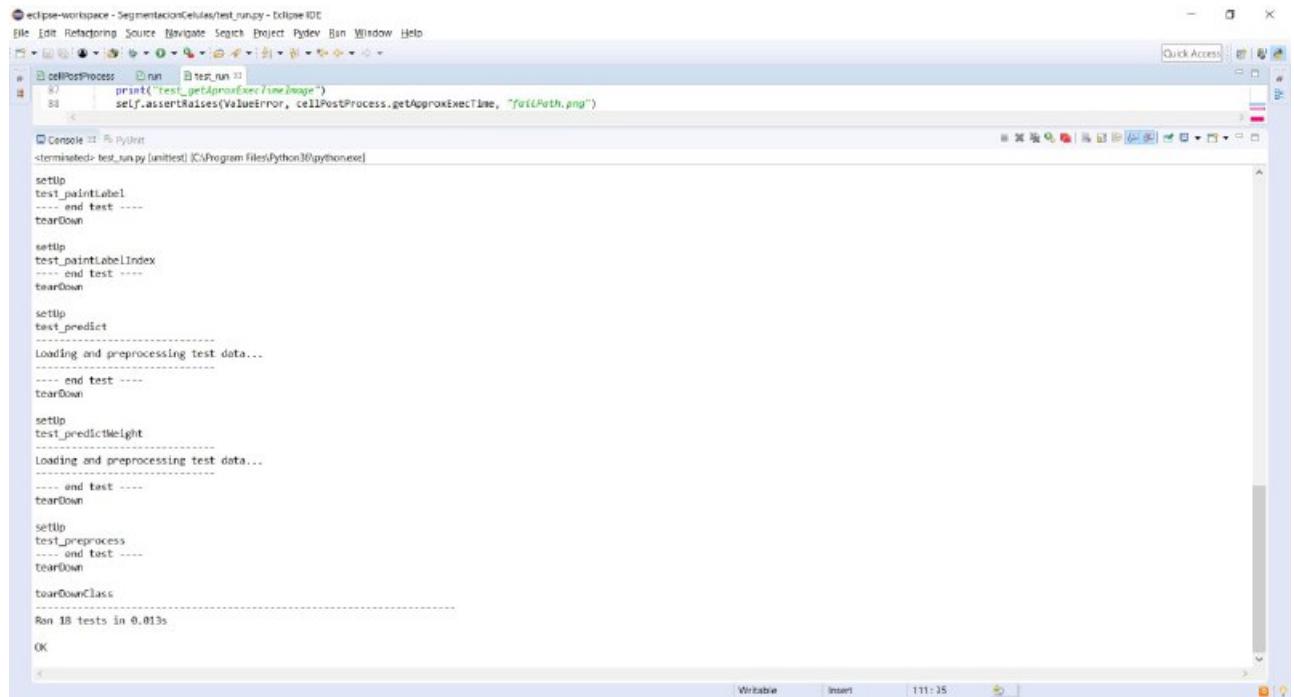
Validación:

```
def escribirCSV(nombre, noProcedimiento, cellsArea, cellsCenter, imageName):
    """ Datos Default para prueba POC """
    """ Diccionario de Datos ficticios que se agregarán al CSV """
    if (nombre == "" or noProcedimiento == "" or nombre == None or noProcedimiento == None):
        raise ValueError('Todos los campos deben estar llenos')
    elif (cellsArea == [] or cellsCenter == []):
        raise ValueError('Datos no validos')
    diccionario = {'Cell_Number': list(range(1, len(cellsArea))), 'Area': cellsArea[1:], 'Center': cellsCenter[1:]}

    """ Columnas de la Estructura del CSV """
    listaColumnas = ['Cell_Number', 'Area', 'Center']

    """ Creando CSV """
    df = pd.DataFrame(diccionario, columns=listaColumnas)
    df.to_csv(STATIC_FOLDER[:-4] + '/csv/' + nombre + "(" + imageName + ")" + "(" + noProcedimiento + ")")
    # Especificar ruta
    print("Creado con Exito")
```

## 15.5. Evidencia de Pruebas



The screenshot shows the Eclipse IDE interface with a Python test run. The code being run is:

```
cellPostProcess
    print("test_getApproxExecTimeImage")
    self.assertRaises(ValueError, cellPostProcess.getApproxExecTime, "fullPath.png")
```

The console output shows the execution of various test methods:

```
Console 21: Python
terminated: test_unittest [C:\Program Files\Python36\python.exe]
setUp
test_paintLabel
---- end test ----
tearDown

setUp
test_paintLabelIndex
---- end test ----
tearDown

setUp
test_predict
-----
Loading and preprocessing test data...
---- end test ----
tearDown

setUp
test_predictWeight
-----
Loading and preprocessing test data...
---- end test ----
tearDown

setUp
test_preprocess
---- end test ----
tearDown

tearDownClass
-----
Run 18 tests in 0.013s
OK
```

Figura 5: Pruebas

## 16. Repositorio en GitHub

El enlace al repositorio es: <https://github.com/ddelgadoJS/Cell-Segmentation-System>

## 17. Conclusiones

Para los doctores del laboratorio de quimiosensibilidad tumoral de la Universidad de Costa Rica es importante avanzar con los experimentos, los cuales es necesario procesar miles de imágenes en una infraestructura computacional modesta.

El interés en el proyecto es considerable, pues analizar las imágenes "a mano" tardaría muchísimo tiempo y requeriría muchísimos recursos de esta institución pública.

Gracias a la definición minuciosa de los atributos de calidad y las métricas, con las cuales se cuantificará el nivel alcanzado en distintos ámbitos de la calidad del software, será posible, para los involucrados en el proyecto, definir posibles mejoras y/o cambios, previo al inicio del desarrollo, cuando aún es posible realizar correcciones sin generar grandes inconvenientes.

## 18. Pasos para obtener versión actual

### 18.1. Pre-requisitos

Para ejecutar o probar el proyecto actual, necesita Python 3.x y las siguientes bibliotecas en su computadora.

- Theano

```
$ pip install --upgrade --no-deps  
git+git://github.com/Theano/Theano.git
```

- TensorFlow

```
$ pip install tensorflow
```

- Keras

```
$ pip install --upgrade keras
```

- Flask

```
$ pip install Flask
```

### 18.2. Instalación

1. Descargar o clonar el repositorio

```
$ git init  
$ git clone https://github.com/delgadoJS/Cell-Segmentation-System
```

2. Ejecutar el archivo run.py

```
$ python run.py
```

```
(base) C:\Users\Daniel\Documents\Git\Cell-Segmentation-System>python run.py  
C:\Users\Daniel\Anaconda3\lib\site-packages\h5py\_init_.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.  
  from _conv import register_converters as _register_converters  
Using TensorFlow backend.  
* Serving Flask app "run" (lazy loading)  
* Environment: production  
  WARNING: Do not use the development server in a production environment.  
  Use a production WSGI server instead.  
* Debug mode: on  
* Restarting with stat  
C:\Users\Daniel\Anaconda3\lib\site-packages\h5py\_init_.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.  
  from _conv import register_converters as _register_converters  
Using TensorFlow backend.  
* Debugger is active!  
* Debugger PIN: 271-581-308  
* Running on http://0.0.0.0:8000/ (Press CTRL+C to quit)
```

Figura 6: Ejecución aplicación web en servidor

3. Acceder a la dirección *127.0.0.1:8000* en un explorador.

Una vez que se haya completado, tendrá acceso a todas las funcionalidades del proyecto.



Figura 7: Aplicación web

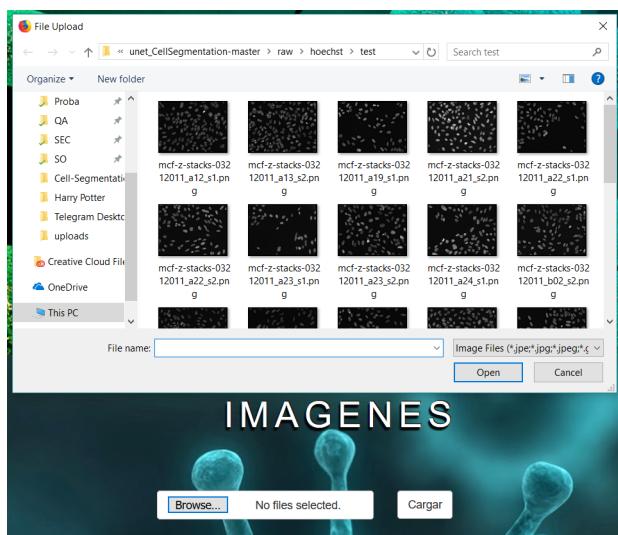


Figura 8: Carga de imágenes

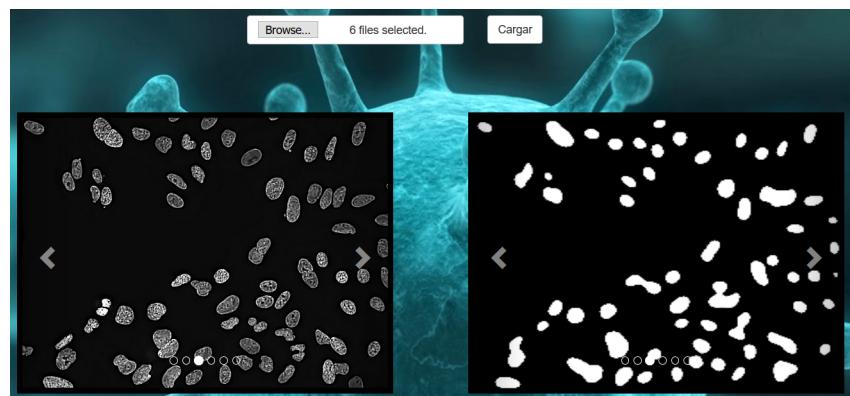


Figura 9: Visualización de predicciones

### 18.3. Ejecutar las pruebas

Para ejecutar las pruebas debe ejecutar el archivo *test\_run.py*. Incluso puede agregar pruebas extra para el sistema en este mismo archivo.

```
$ python test_run.py
```

```
(base) C:\Users\Daniel\Documents\Git\Cell-Segmentation-System>python test_run.py
C:\Users\Daniel\Anaconda3\lib\site-packages\h5py\_init_.py:36: FutureWarning: Conversion of the second argument of issubdtype from 'float' to 'np.floating' is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type` .

    from ._conv import register_converters as _register_converters
Using TensorFlow backend.
setUpClass
setUp
test_crearCSV
---- end test ----
tearDown

.setUp
test_generacionLista
---- end test ----
tearDown

.setUp
test_listaN
---- end test ----
tearDown

.setUp
test_listavacia
---- end test ----
tearDown

.tearDownClass

-----
Ran 4 tests in 0.012s
OK
```

Figura 10: Ejecución de pruebas del sistema

## **19. Referencias**

ISO/IEC 9126-1:2001 - Software engineering – Product quality – Part 1: Quality model. (s.f.). Recuperado 1 de septiembre de 2018, de <https://www.iso.org/standard/22749.html>