

Tecnológico de Costa Rica

Ingeniería en Computación

IC6831 -Aseguramiento de la Calidad del Software

Informe del Sprint

Profesor: Saúl Calderón Ramírez

Estudiante:

José Miguel Mora Rodríguez

Dylan Rodríguez Barboza

Karina Zeledón Pinell

Octubre, 2017

Contents

1	Verificación del estándar de codificación	3
1.1	Instalación y verificación	3
1.1.1	Instalación	3
1.1.2	Verificación	5
2	Manual de Administración de la Configuración del Software	5
2.1	Acceso a las secciones del repositorio	5
2.1.1	Acceso al repositorio	5
2.1.2	Acceso a los requerimientos del sistema	7
2.1.3	Acceso al diseño	8
2.1.4	Acceso a la aplicación web	10
2.1.5	Descarga de todo el repositorio	12
3	Métricas Implementadas	13
3.1	Atributo de calidad: Comportamiento temporal	13
3.1.1	Métrica: tiempo de respuesta	13
3.1.2	Implementación de la métrica	14
3.2	Atributo de calidad: Instalabilidad	15
3.2.1	Métrica: facilidad de instalación	15
3.2.2	Implementación de la métrica	15
4	Especificación de pruebas unitarias	15

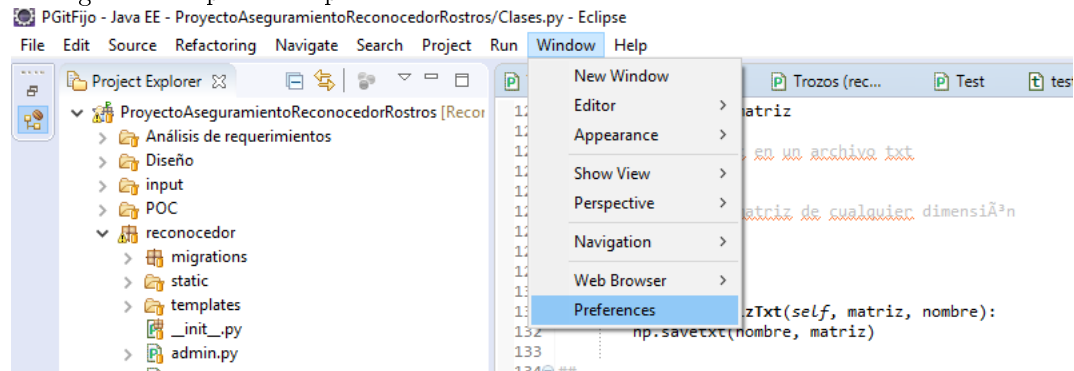
1 Verificación del estándar de codificación

La herramienta seleccionada para la verificación del estándar de codificación seleccionado es la que tiene integrada eclipse, la cual se especializa en la verificación del estándar PEP8.

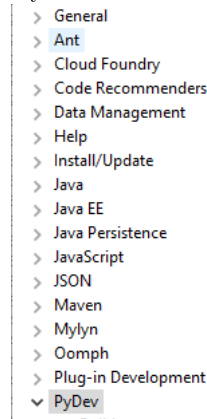
1.1 Instalación y verificación

1.1.1 Instalación

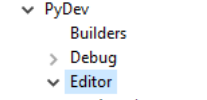
Para habilitar la opción del estándar de codificación PEP8 en eclipse, se deben seleccionar las siguientes opciones: Window >> Preferences >> PyDev >> Editor >> Code Analysis >> pycodestyle.py(pep8) >> Warning. Se ilustra con las siguientes capturas de pantalla: Window >> Preferences



PyDev



Editor



Code Analysis

- ▼ Editor
 - Auto Imports
 - > Code Analysis

Una vez seleccionado Code Analysis, en la parte derecha de la pantalla deben aparecer opciones como las siguientes:

Code Analysis

PyDev Analysis
NOTE: Any file with the comment below will not be analyzed.
#@PydevCodeAnalysisIgnore

☒ Do code analysis?

Unused | Undefined | Imports | Others | pycodestyle.py (pep8)

Unused import
☐ Error ☒ Warning ☐ Info ☐ Ignore

Unused wild import
☐ Error ☒ Warning ☐ Info ☐ Ignore

Don't report unused imports in modules named: (comma separated)
__init__, *QT

Unused parameter
☐ Error ☐ Warning ☒ Info ☐ Ignore

Unused variable
☐ Error ☒ Warning ☐ Info ☐ Ignore

Don't report unused variable if name starts with: (comma separated)
dummy, _ unused

Save to ... Show from ... Open location ...

Showing from: Workspace

Seleccionar pycodestyle.py (pep8)

Code Analysis

PyDev Analysis
NOTE: Any file with the comment below will not be analyzed.
#@PydevCodeAnalysisIgnore

☒ Do code analysis?

Unused | Undefined | Imports | Others | pycodestyle.py (pep8)

Pep8
☐ Error ☒ Warning ☐ Info ☐ Don't run

Save to ... Show from ... Open location ...

Showing from: Workspace

Seleccionar Warning, luego, se deben aplicar los cambios:


Save to ... Show from ... Open location ...

Showing from: Workspace

Restore Defaults Apply

1.1.2 Verificación

En el ambiente de eclipse, una vez que se hayan seguido los pasos anteriormente dados, se darán ciertas advertencias cada vez que se incumpla con el estándar de codificación PEP8: Ejemplo de una advertencia que puede ser notificada por el ambiente eclipse:



```
7
8 centroides = [] #atributo de la clase de entrenamiento, en el init deberá-a intentar cargar el a
9 #atributo de la clase de entrenamiento, en el init deberá-a intentar cargar el ar
10 #atributo de la clase de entrenamiento, en el init deberá-a intentar cargar el ar
11 #atributo de la clase de entrenamiento, en el init deberá-a intentar cargar el ar
12
13 ##
```

The screenshot shows three yellow warning icons on the left margin. A tooltip box is open over the third icon, displaying the following messages:

- E262 inline comment should start with '#'
- E501 line too long (114 > 79 characters)
- E261 at least two spaces before inline comment

El objetivo que se tiene con esta implementación en eclipse es reducir al máximo el número de advertencias, entre menos de estas se tenga, más se cumplirá con el estándar de codificación.

2 Manual de Administración de la Configuración del Software

2.1 Acceso a las secciones del repositorio

2.1.1 Acceso al repositorio

Para acceder a las secciones que ofrece el repositorio, se debe acceder al siguiente enlace, el cual redirecciona al repositorio de nombre ReconocedorRostros: <https://github.com/dylanrodbar/ReconocedorRostros>. Una vez seleccionado, debe aparecer una pestaña en el navegador con el siguiente contenido:

dylanrodbar / ReconocedorRostros

Watch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

Proyecto semestral del curso de Aseguramiento de la Calidad del Software en el Instituto Tecnológico de Costa Rica

Add topics

29 commits 1 branch 0 releases 2 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

File	Commit Message	Time
dylanrodbar Update README.md	Latest commit 9e38354 4 minutes ago	
Diseño	Diseño del sistema agregado al repositorio	a day ago
POC	Version final	a month ago
ReconocedorRostrosA	Llamadas a funciones	a day ago
__pycache__	Llamadas a funciones	a day ago
input	Unit test corregido	2 months ago
reconocedor	Llamadas a funciones	a day ago
.project	Llamadas a funciones	a day ago
.pydevproject	Llamadas a funciones	a day ago
README.md	Update README.md	4 minutes ago
Test.py	Unit test Javadoc	a month ago
Trozos.py	Arreglo a reconocedor	3 hours ago
autovectores.txt	Llamadas a funciones	a day ago
centroides.txt	Llamadas a funciones	a day ago
db.sqlite3	Trozos+Front-End	2 months ago
manage.py	Trozos+Front-End	2 months ago

Y la siguiente introducción:

README.md

ReconocedorRostros

Proyecto semestral del curso de Aseguramiento de la Calidad del Software en el Instituto Tecnológico de Costa Rica, II semestre, 2017

Desarrollado por los estudiantes:

Jose Mora Rodríguez

Dylan Rodríguez Barboza

Karina Zeledón Pinnel

Seguido de información acerca del sistema:

Acerca de la aplicación

El sistema de reconocimiento de rostros implementado busca satisfacer estas necesidades en Costa Rica, un país en el cual no se cuenta con gran abundancia de este tipo de medidas, en donde se han encontrado necesidades tales como el reconocimiento de costarricenses mediante sus rostros, de manera eficiente.

Y un resumen de las secciones que ofrece:

Elementos del repositorio

En este repositorio se guardarán las siguientes secciones:

Diseño

Aplicación web

Configuraciones de la aplicación web

El acceso a las diferentes secciones del repositorio se explicarán a continuación



2.1.2 Acceso a los requerimientos del sistema

Debajo de la introducción, se puede apreciar una sección dedicada a los requerimientos del sistema, como se muestra a continuación:

Requerimientos

En esta sección se podrá encontrar el documento asociado a los atributos de calidad, además del SyRS, asociados al sistema de reconocimiento de rostros que se implementa en esta aplicación, se puede acceder mediante [este enlace](#)

Una vez seleccionado el enlace, se debe visualizar lo siguiente:

dylanrodbar Documentos añadidos		Latest commit 37ab795 3 hours ago
..		
 Atributos de calidad.pdf	Documentos añadidos	3 hours ago
 SyRS.pdf	Documentos añadidos	3 hours ago

Para acceder a los atributos de calidad, se debe seleccionar “Atributos de calidad.pdf”, donde una vez hecho, se mostrará lo siguiente:



En donde basta con seleccionar el botón “Download” para descargar el archivo. Deben seguirse los mismos pasos para descargar el SyRS:



Una alternativa a los pasos mencionados anteriormente, es seguir los pasos dados en el readme.md de la sección correspondiente al repositorio.

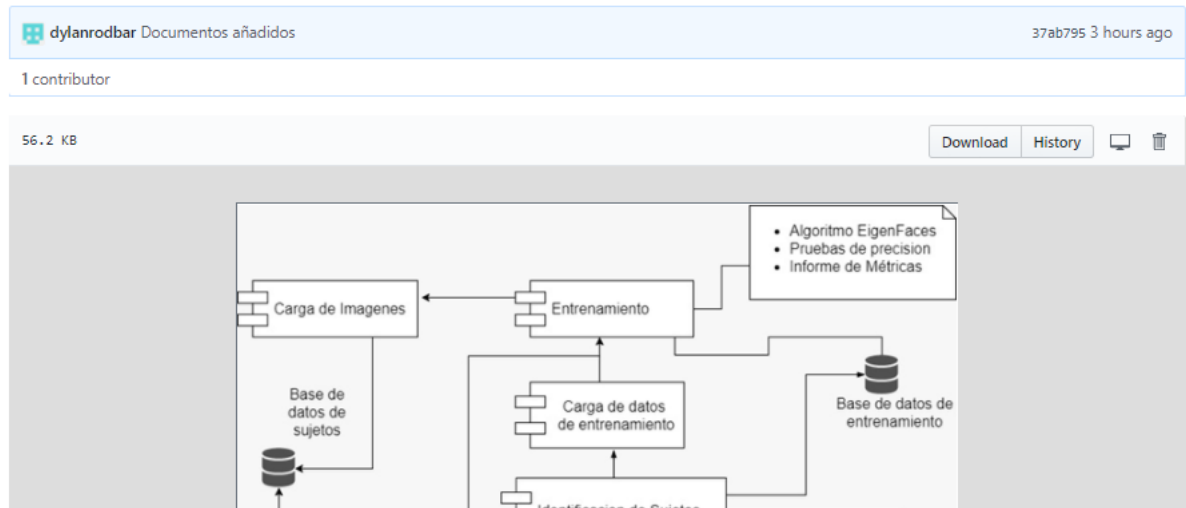
2.1.3 Acceso al diseño

Debajo de la sección de los requerimientos del sistema (mencionado en la sección anterior), se puede apreciar una sección dedicada al diseño, como se muestra a continuación:

Diseño

En esta sección se podrá encontrar el diagrama de clases asociado al sistema de reconocimiento de rostros que se implementa en esta aplicación, se puede acceder mediante [este enlace](#)

Una vez seleccionado el enlace mostrado, se deberá mostrar un contenido como el siguiente:



Una alternativa a los pasos mencionados anteriormente, es seguir los pasos dados en el `readme.md` de la sección correspondiente al repositorio.

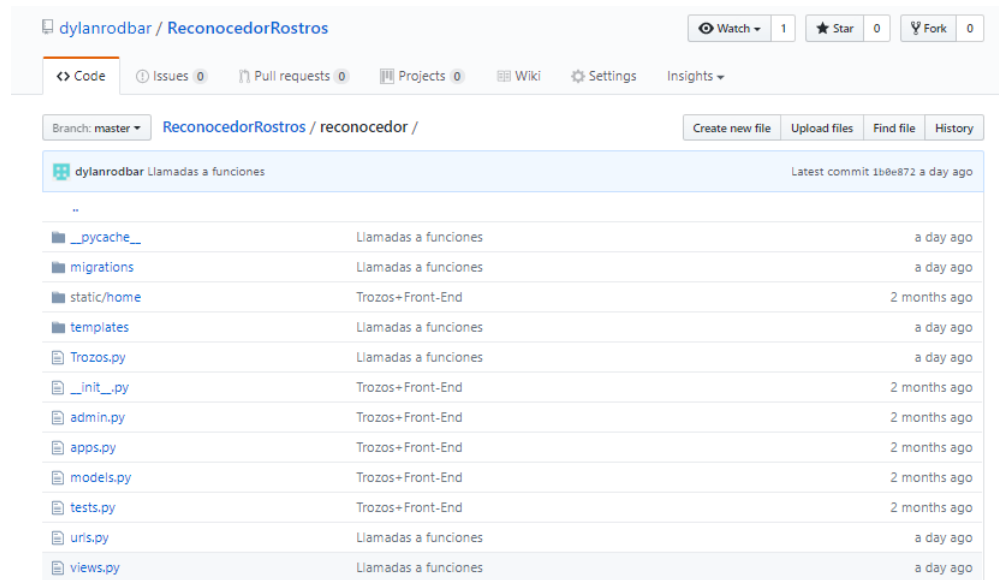
2.1.4 Acceso a la aplicación web

Debajo de la sección de diseño presentada anteriormente, se muestra la sección para el acceso a la aplicación web:

Aplicación web

En esta sección se podrán encontrar los scripts referentes a la aplicación web que implementa el reconocedor de rostros, puede ser visualizada mediante [este enlace](#)

Donde una vez que se seleccione el enlace, se debe mostrar un contenido como el siguiente:



Para el acceso al código de cada script en esta sección, es preciso seleccionar el script que se desee, seguidamente se le mostrará el contenido de este, debido a que la aplicación web está desarrollada mediante el framework django, el código que se verá, está implementado en python, html y css, para mostrar de ejemplo, se seleccionará el script “Trozos.py”, del cual se desplegará un contenido como el siguiente:

Branch: master ReconocedorRostros / reconocedor / Trozos.py

1 contributor

262 lines (215 sloc) 7.37 KB

```

1 import cv2
2 import os
3 import numpy as np
4 from numpy import matrix
5
6 centroides = [] #atributo de la clase de entrenamiento, en el init debería intentar cargar el archivo centroides.txt
7 diffPrima = [] #atributo de la clase de entrenamiento, en el init debería intentar cargar el archivo autovectores.txt
8
9
10 ##
11 # Se cargan n imágenes .png de dimensiones l * a = p, donde l son la
12 # cantidad de pixeles de largo y a la cantidad de pixeles de ancho,
13 # de una carpeta específica. Posteriormente se convierte cada una
14 # en vectores de forma que se guarden en una matriz M de dimensiones
15 # p * n. Por último calcula la matriz de covarianza de la matriz M
16 # y la guarda en un archivo .txt de nombre MatrizCovarianza.txt en
17 # la dirección 'root'.
18 #
19 # <p>
20 # Esta funcion es el metodo principal de la aplicación por lo que
21 # se debe de llamar para iniciar la aplicación.

```

2.1.5 Descarga de todo el repositorio

Se debe acceder al siguiente enlace: <https://github.com/dylanrodbar/ReconocedorRostros>
Una vez hecho esto, seleccionar la opción “clone and download”:

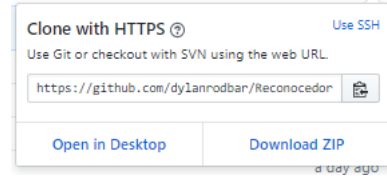
Proyecto semestral del curso de Aseguramiento de la Calidad del Software en el Instituto Tecnológico de Costa Rica

29 commits 1 branch 0 releases 2 contributors

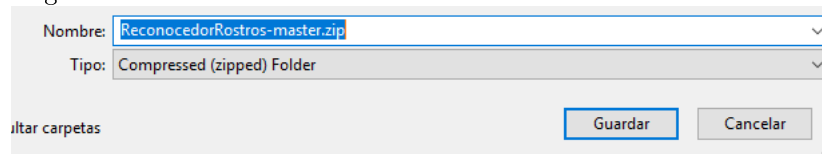
Branch: master New pull request Create new file Upload files Find file Clone or download

File/Folder	Description	Time
Diseño	Diseño del sistema agregado al repositorio	a day ago
POC	Version final	a month ago
ReconocedorRostrosA	Llamadas a funciones	a day ago
__pycache__	Llamadas a funciones	a day ago
input	Unit test corregido	2 months ago
reconocedor	Llamadas a funciones	a day ago
.project	Llamadas a funciones	a day ago
.pydevproject	Llamadas a funciones	a day ago
README.md	Update README.md	21 minutes ago
Test.py	Unit test Javadoc	a month ago

Seguidamente se desplegará contenido como el siguiente:



Seleccionar “Download ZIP” y seleccionar la carpeta en la que se quiera descargar:



Seguidamente se debe descomprimir el archivo. Una vez hecho esto, se tendrá acceso a todos los archivos correspondientes al repositorio.

3 Métricas Implementadas

3.1 Atributo de calidad: Comportamiento temporal

3.1.1 Métrica: tiempo de respuesta

- Requerimientos funcionales asociados con esta métrica: RF-03, encontrar coincidencias entre los sujetos y sus rostros. RF-04, capacidad de entrenar al sistema. La relación principal que tiene con estos requerimientos es que se debe medir el tiempo que tarda el sistema en satisfacer los mismos, para esto se pretende utilizar una herramienta implementada en el lenguaje de programación python, que se encarga de medir el tiempo de ejecución de una función deseada.
- Umbral: El tiempo de respuesta fijado como métrica para la clasificación de cada sujeto fue de entre 5 y 10s. El tiempo de entrenamiento fijado como métrica fue de menos de 2 minutos por sujeto.
- Herramienta seleccionada para medir: time de python, es una librería encargada de medir el tiempo de ejecución de las funciones que se deseen, para esto, se usa una función llamada time(), que debe colocarse antes y después del código encargado de ejecutar una función deseada. Por ejemplo, en la imagen que ilustra a continuación, se notarán las líneas de código encargadas de calcular el tiempo de ejecución (delimitadas por ##).

```

from time import time
def holaMundo():
    print("Hola mundo")
    print("La suma de 3 y 4 es: ")
    print(3+4)
    print("Adiós")

def calcularTiempo():
    tiempoInicio = time() ##
    retorno = holaMundo() ##
    tiempoGenerado = time()-tiempoInicio ##
    print("Tiempo recorrido en función holaMundo: ")
    print(tiempoGenerado)
    return retorno

calcularTiempo()

```

Generará el siguiente resultado:

```

Hola mundo
La suma de 3 y 4 es:
7
Adiós
Tiempo recorrido en función holaMundo:
0.035549163818359375

```

3.1.2 Implementación de la métrica

Para un 80% de muestras como entradas para el entrenamiento(328 imágenes) y el 20% restante como muestras a clasificar y calculando el 100% de los autovectores se obtuvieron los siguientes resultados:

```

Tiempo de entrenamiento: 8.12598204612732
Tiempo promedio de reconocimiento: 0.0242898551429

```

El tiempo de entrenamiento se refiere al tiempo llevado a cabo recibiendo todas las muestras como entradas, ordenándose y realizando todos los cálculos necesarios para obtener los centroides de los sujetos en el nuevo espacio de autovectores. El tiempo promedio de reconocimiento se refiere al tiempo en promedio que tardó el sistema en la clasificación de los sujetos, tomando en cuenta que del 100% de las muestras del sujeto se clasificaron 20%, en este caso, de 10 muestras se clasificaron 2 (las muestras clasificadas no se encontraban dentro de las de entrenamiento.) Se realizaron las medidas de tiempo utilizando un 10% de los autovectores lo cual dió los siguientes resultados:

```

Tiempo de entrenamiento: 2.5515644550323486
Tiempo promedio de reconocimiento: 0.0176765918732

```

Lo cual refleja que a menor porcentaje de autovectores, mejor tiempo promedio de ejecución.

En conclusión en ambos casos se cumplieron los umbrales establecidos de forma eficaz y eficiente.

3.2 Atributo de calidad: Instalabilidad

3.2.1 Métrica: facilidad de instalación

- Requerimientos funcionales asociados con esta métrica: RF-01, almacenamiento de muestras de diferentes personas. RF-02, almacenamiento de datos personales de diferentes personas. RF-03, encontrar coincidencias entre sujetos y sus rostros. RF-04, capacidad de entrenar al sistema. RF-05, interfaz web. RF-06, manejo de diferentes usuarios que puedan ingresar al sistema. RF-07, reconocedor complementario de iris ocular. RF-08, reconocedor complementario de huella dactilar. La relación principal que tiene con estos requerimientos es que todos deben ser capaz de satisfacerse en diferentes entornos.

3.2.2 Implementación de la métrica

Utilización de un lenguaje de programación multiplataforma y la utilización de un framework implementado en el mismo lenguaje que facilite la ejecución de la misma en cualquier tipo de entorno. Es decir, al utilizar un lenguaje de programación como python, y el uso de un framework como django, el cual está implementado en python, se hace viable ejecutar el sistema en cualquier ambiente de trabajo, incluyendo diferentes sistemas operativos.

4 Especificación de pruebas unitarias

En el presente sprint se implementaron seis pruebas unitarias tipo “pytest”, las cuales son:

1. test_convertirMatrizAVector
2. test_crearMatrizDeVectores
3. test_cara_prom
4. test_auto_vectores
5. test_matriz_diferencias
6. test_normalizar

Las primeras dos fueron implementadas en el POC por lo que no se especificarán en este documento, sin embargo el resto presentan el siguiente comportamiento:

Prueba a la función cara_prom

Entradas esperadas: Se espera que se reciba un conjunto de vectores sobre los cuales se sacará su promedio, ejemplo:

- Vectores = [[11,10,4],[100,200,20]]

- Salida esperada = $[25/3, 320/3]$

Caso Especial: Pruebas con vectores de magnitud 0, la salida debería ser un vector de la misma dimensión de los vectores con magnitud 0.

Prueba para la función `auto_vectores`

Entradas esperadas: Se espera recibir una matriz con vectores del tipo `numpy.array`. A raíz de esta matriz se calculan los autovectores de la matriz la cual la entrada corresponde a su matriz de diferencia. Esta función depende ampliamente del comportamiento de la librería de `numpy` por lo que su salida depende de esta librería.

Prueba para la función `matriz_diferencias`

Entradas esperadas: Se espera recibir una matriz con vectores del tipo `numpy.array` y un vector del tipo `numpy.array`. Esta función depende ampliamente del comportamiento de la librería de `numpy` por lo que su salida depende de esta librería. Ejemplo:

- Vectores = $[[1, 2, 3, 4], [1, 2, 3, 4]]$
- Vector = $[1, 2, 3, 4]$
- Salida esperada = $[[0, 0, 0, 0], [0, 0, 0, 0]]$

Prueba para la función `normalizar`

Entradas esperadas: Se espera recibir un vector. La prueba unitaria de esta función consiste en comprobar si la forma en que se normaliza el vector da como resultado un vector de magnitud igual a uno. Esta función depende del comportamiento de la librería de `numpy` por lo que su salida depende de esta librería.

Ejemplo:

- Entrada = `np.linalg.norm(entrenamiento.normalizar([123.0032, 123.4, 36.0043, 3, 0, 345.19]))`
== 1
- Salida: `True`