

Instituto Tecnológico de Costa Rica

Ingeniería en Computación

IC6831-Aseguramiento de la Calidad
Avance 3: Pruebas unitarias, de integración y de sistema.
Profesor: Saúl Calderón Ramírez

Estudiantes:
Luis Alonzo Cascante Franco 2014159896
Olman Castillo Picado 2015148651
María Laura Pizarro Moreno 2015120626
Gabriel Venegas Castro 2013115967

Segundo Semestre 2017

Índice

1	Requerimientos para el sprint	2
1.1	Pruebas	2
1.1.1	Clases de equivalencia:	2
1.1.2	Diseño de las pruebas:	3
1.2	Evidencias	4
1.2.1	Pruebas de unidad:	4
1.2.2	Pruebas de Integración:	5
1.2.3	Pruebas de Sistema:	6

1 Requerimientos para el sprint

1.1 Pruebas

1.1.1 Clases de equivalencia:

Clase de equivalencia	Válida	No valida	Id. de la prueba
Carga de datos	Dirección de carga real	Direccion de carga inexistente	1,2,5,9
Almacenamiento de datos	Los datos existen y es algún tipo de dato de la librería numpy	El conjunto de datos es vacío o no es de la librería numpy	8,9
Manipulación de datos	Se ejecutan los pasos anteriores. Ejemplo: el cálculo de los autovalores y autovectores que requiere de la covarianza	El omitir un paso anterior	3,4,6,9,10
Ejecución de operaciones	Uso de datos adecuados: Ejemplo en el caso 3,4,6,7 se usan datos de la librería numpy	Datos de no compatibles	3,4,6,7,9,10

1.1.2 Diseño de las pruebas:

Id. de la prueba	Tipo de prueba	Descripción	Precondiciones	Resultados esperados
1	Unitaria	Se encarga de cargar una imagen	Debe darse la dirección de la imagen	La carga de la imagen
2	Unitaria	Se carga la bd muestral y se selecciona aleatoriamente 50%	Debe existir la ruta en que se ubican las imágenes	La carga de los sujetos con sus respectivas imágenes
3	Unitaria	Genera la matriz de covarianza mediante un stub	Debe existir un stub con una matriz de tipo np.matriz	El cálculo de la matriz de covarianza
4	Unitaria	Calcular los autovectores y autovalores de una matriz de covarianza	Se debe calcular la matriz de covarianza mediante np.cov(), en un stub para la matriz de tipo np.matriz	Los autovalores y autovectores
5	Integración	Carga de datos y generación de la matriz de covarianza	Debe existir la ruta en que se ubican las imágenes	El integrar la carga de imágenes con la generación de muestra
6	Integración	Calcular las proyecciones a partir de la autovalores de la muestra	Se debe haber cargado la muestra y calculado la matriz de covarianza	Obtener el las proyecciones de la muestra
7	Integración	Identificar sujeto mediante las proyecciones	Se debe tener la imagen a identificar y el cálculo de las proyecciones	La identificación de un sujeto
8	Integración	Almacenar la proyección mediante el Dao_DBPCA	Se deben haber calculado las proyecciones	Guardar las proyecciones en un archivo .csv
9	Sistema	Entrenamiento	Tener la muestra	Entrenamiento del sistema
10	Sistema	Identificación de sujeto	Haber realizado el entrenamiento	El nombre de sujeto identificado

1.2 Evidencias

1.2.1 Pruebas de unidad:

```
Console  Remote Systems  PyUnit
<terminated> UnitTest.py [unittest] [C:\Python34\python.exe]
Finding files... done.
Importing test modules ... done.
Creating test database for alias 'default'...
System check identified no issues (0 silenced).

Test1: Cargar imagen
Test2: Cargar BD con un 50% para muestras
Test3: Generar matriz covarianza
Test4: generar autovectores y autovalores
-----
Ran 4 tests in 0.154s

OK
Destroying test database for alias 'default'...
```

Prueba 1:

```
def test_1(self):
    print("Test1: Cargar imagen")
    img = Imagen(None)
    self.assertTrue(img.leer_imagen(Configuracion.RUTA_2+"otros/1.pgm"), "Error al leer la imagen")
```

Prueba 2:

```
def test_2(self):
    print("Test2: Cargar BD con un 50% para muestras")
    gestor=GestorMuestra()
    gestor.cargar(50)
    print("Tamaño de muestra")
    print(np.shape(gestor.muestra))
    self.assertTrue(gestor.muestra.sujetos, "Fallo en la carga, los sujetos no fueron cargados")
```

Prueba 3:

```
def test_3(self):
    print("Test3: Generar matriz covarianza")
    muestra=Muestra()
    sm=stub_muestra()
    muestra.matriz=sm.matriz
    muestra.generar_matriz_covarianza()
    self.assertTrue(muestra.generar_matriz_covarianza(), "Error en calcular la matriz de covarianza")
```

Prueba 4:

```
def test_4(self):
    print("Test4: generar autovectores y autovalores")
    muestra = Muestra()
    sm=stub_muestra()
    muestra.matriz_covarianza=sm.matriz_covarianza
    muestra.calcular_autovalores_autovectores()
    self.assertTrue(muestra.calcular_autovalores_autovectores(),"Error en calcular los autovalores y autovectores")
```

1.2.2 Pruebas de Integración:

```
Console  Remote Systems  PyUnit
<terminated> IntegrationTest.py [unittest] [C:\Python34\python.exe]
Finding files... done.
Importing test modules ... done.
Creating test database for alias 'default'...
System check identified no issues (0 silenced).

Test1: pueba de gestor de carga y muestra
Test2: muestra y proyeccion
Test3: proyeccion y clasificacion
Test4: proyeccion y dao db pca
-----
Ran 4 tests in 10.144s

OK
Destroying test database for alias 'default'...
```

Prueba 1:

```
def test_1(self):
    print("Test1: pueba de gestor de carga y muestra")
    IntegrationTest.gestor_muestra.cargar(0)
    IntegrationTest.muestra=IntegrationTest.gestor_muestra.muestra
    IntegrationTest.muestra.generar_matriz()
    IntegrationTest.muestra.generar_matriz_covarianza()
```

Prueba 2:

```
def test_2(self):  
    print("Test2: muestra y proyeccion")  
    IntegrationTest.muestra.calcular_autovalores_autovectores()  
    IntegrationTest.proyeccion.proyectar(IntegrationTest.muestra,10)
```

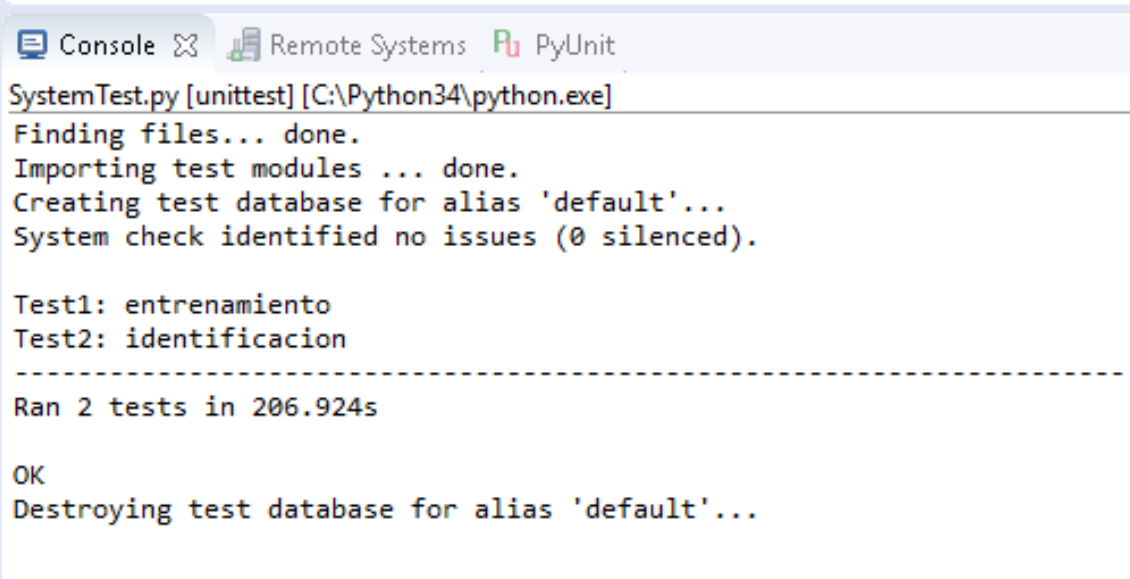
Prueba 3:

```
def test_3(self):  
    print("Test3: proyeccion y clasificacion")  
    img = Imagen(None)  
    img.leer_imagen(Configuracion.RUTA_2+"otros/1.pgm")  
    IntegrationTest.clasificacion=DistanciaCentroide(IntegrationTest.muestra)  
    IntegrationTest.clasificacion.clasificar(img,self.proyeccion.autocaras,self.proyeccion.proyecciones)
```

Prueba 4:

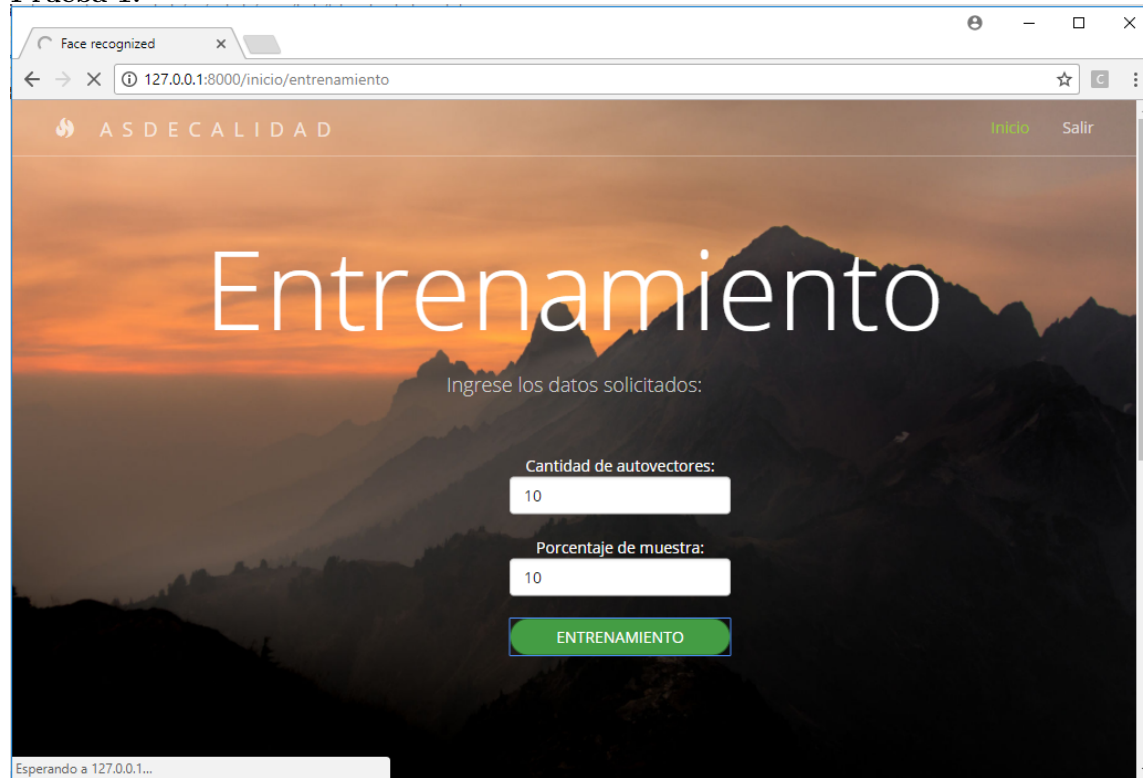
```
def test_4(self):  
    print("Test4: proyeccion y dao db pca")  
    IntegrationTest.dao_db_pca.guardar(IntegrationTest.proyeccion)
```

1.2.3 Pruebas de Sistema:



```
Console  Remote Systems  PyUnit  
SystemTest.py [unittest] [C:\Python34\python.exe]  
Finding files... done.  
Importing test modules ... done.  
Creating test database for alias 'default'...  
System check identified no issues (0 silenced).  
  
Test1: entrenamiento  
Test2: identificacion  
-----  
Ran 2 tests in 206.924s  
  
OK  
Destroying test database for alias 'default'...
```

Prueba 1:



```
def test_1(self):
    print("Test1: entrenamiento")
    SystemTest.driver.get("http://127.0.0.1:8000/inicio/principal")
    SystemTest.driver.find_element_by_css_selector("input.btn.btn-success").click()
    SystemTest.driver.find_element_by_name("cantidad_autovectores").clear()
    SystemTest.driver.find_element_by_name("cantidad_autovectores").send_keys("10")
    SystemTest.driver.find_element_by_name("porcentaje_muestra").clear()
    SystemTest.driver.find_element_by_name("porcentaje_muestra").send_keys("10")
    SystemTest.driver.find_element_by_css_selector("input.btn.btn-success").click()
```


Prueba 2:



```
def test_2(self):  
    print("Test2: identificacion")  
    SystemTest.driver.get("http://127.0.0.1:8000/inicio/principal")  
    SystemTest.driver.find_element_by_xpath("//input[@value='Identificar']").click()  
    SystemTest.driver.find_element_by_id("input-1a").clear()  
    SystemTest.driver.find_element_by_id("input-1a").send_keys("C:\\Users\\olman\\Documents\\GitHub\\Asdecalidad-1\\website\\src\\website\\inicio\\back_end\\")  
    SystemTest.driver.find_element_by_xpath("//button[@type='submit']").click()
```