

# Plan de pruebas de software

*Andres Felipe Olmos Rojas*

## Plan de pruebas de software

***C.R.U.D JAVA WEB***  
***Fecha: 05/04/2021***

# Plan de pruebas de software

Andres Felipe Olmos Rojas

## Tabla de contenido

### Tabla de contenido

Historial de versiones .....	3
Información del proyecto .....	3
Aprobaciones .....	3
Resumen ejecutivo.....	4
Alcance de las pruebas .....	4
Elementos de pruebas .....	4
Pruebas de regresión.....	5
Funcionalidades a no probar.....	5
Enfoque de pruebas (estrategia).....	6
jmeter .....	6
Pruebas unitarias .....	7
Criterios de aceptación o rechazo .....	9
Criterios de aceptación o rechazo.....	9
Criterios de suspensión.....	9
Criterios de reanudación .....	10
Recursos .....	11
Requerimientos de entornos – Hardware.....	11
Requerimientos de entornos – Software .....	11
Herramientas de pruebas requeridas.....	12
Personal.....	12
Entrenamiento.....	12

# Plan de pruebas de software

Andres Felipe Olmos Rojas

## Historial de versiones

Fecha	Versión	Autor	Organización	Descripción
	1.0	Andres Olmos	Sena	Se realiza el crud funcional sin realizar pruebas
	1.1	Andres Olmos	Sena	Se le aplican las pruebas y se realiza la documentacion

## Información del proyecto

Empresa / Organización	S.E.N.A
Proyecto	C.R.U.D java web
Fecha de preparación	
Cliente	S,E,N.A
Patrocinador principal	S.E.N.A
Gerente / Líder de proyecto	Andres Olmos
Gerente / Líder de pruebas de software	Andres Olmos

## Aprobaciones

Nombre y Apellido	Cargo	Departamento u organización	Fecha	Firma
Oscar iban Benavides	docente	Cundinamarca, Facatativá, S.E.N.A		

# Plan de pruebas de software

Andres Felipe Olmos Rojas

## Resumen ejecutivo

Para este plan se pruebas se necesitan realizar pruebas de dos tipos de estrés con jmeter y pruebas unitarias con junit se realizaran de estos dos distintos tipos para probar de forma completa que los procesos funcionen correctamente además de probar cuantos usuarios resiste la pagina

## Alcance de las pruebas

### Elementos de pruebas


Con las pruebas de jmeter se busca visualizar que tanto puede soportar la página según el numero de usuarios que accedan al mismo tiempo

Se van a probar con las pruebas unitarias

- ✚ **Búsqueda SQL:** se realizará el proceso de búsqueda de todos los apartados de la lista con una sentencia SQL se testeará que retorne la lista correctamente
- ✚ **Búsqueda SQL por nombre:** se realizará el proceso de búsqueda de todos los apartados de la lista con una sentencia SQL que coincidan con el nombre a buscar se testeará que retorne la lista correctamente
- ✚ **Inserción SQL:** se realizará el proceso de inserción de todos los apartados de la lista con una sentencia SQL se testeará que realice el proceso correctamente
- ✚ **actualizado SQL:** se realizará el proceso de actualización de los ítems de la lista con una sentencia SQL que coincida con el id a actualizar se testeará que realice el proceso correctamente

# Plan de pruebas de software

*Andres Felipe Olmos Rojas*

 **borrado SQL:** se realizará el proceso de borrado de los ítems de la lista con una sentencia SQL que coincida con el id a borrar se testeará que realice el proceso correctamente.

## Pruebas de regresión

Se realizaran las pruebas unitarias empleando la función de NetBeans acompañada con junit para verificar el estado de los procesos que debe realizar el proyecto tal como lo es la inserción ,la actualización y el borrado

## Funcionalidades a no probar

Las funcionalidades que no se probaron en el proyecto son los controladores los cuales tienen la función de redirigir a diferentes paginas y llamar los métodos que di han sido probados con junit no se probaron las variables porque no hay nada que probar

# Plan de pruebas de software

Andres Felipe Olmos Rojas

## Enfoque de pruebas (estrategia)

### jmeter

Se realizarán dos tipos de pruebas en el proyecto

- prueba de estrés estas se realizaran por medio de jmeter
- se realizo la prueba de estrés con 100 usuarios en 1 segundo y no se pudo observar ninguna falla



# Plan de pruebas de software

Andres Felipe Olmos Rojas

81	16:10:50.378	Grupo de Hil...	Petición HTTP	18311	✓	185533	121	3850	3174
82	16:10:50.478	Grupo de Hil...	Petición HTTP	18211	✓	185462	121	3550	3065
83	16:10:50.357	Grupo de Hil...	Petición HTTP	18346	✓	185554	121	4328	3200
84	16:10:50.495	Grupo de Hil...	Petición HTTP	18214	✓	185527	121	4095	3042
85	16:10:50.373	Grupo de Hil...	Petición HTTP	18365	✓	185474	121	3744	3169
86	16:10:50.592	Grupo de Hil...	Petición HTTP	18147	✓	185544	121	3677	2974
87	16:10:50.593	Grupo de Hil...	Petición HTTP	18186	✓	185531	121	3453	2962
88	16:10:50.439	Grupo de Hil...	Petición HTTP	18701	✓	185457	121	4113	3125
89	16:10:50.591	Grupo de Hil...	Petición HTTP	18756	✓	185541	121	3669	2945
90	16:10:50.493	Grupo de Hil...	Petición HTTP	18930	✓	185538	121	4161	3074
91	16:10:50.578	Grupo de Hil...	Petición HTTP	18972	✓	185518	121	3432	2955
92	16:10:50.565	Grupo de Hil...	Petición HTTP	19107	✓	185526	121	4643	3216
93	16:10:50.300	Grupo de Hil...	Petición HTTP	19374	✓	185537	121	4200	3250
94	16:10:50.390	Grupo de Hil...	Petición HTTP	19541	✓	185534	121	3584	3161
95	16:10:50.592	Grupo de Hil...	Petición HTTP	19376	✓	185532	121	3397	2942
96	16:10:50.584	Grupo de Hil...	Petición HTTP	19648	✓	185538	121	4676	2967
97	16:10:50.587	Grupo de Hil...	Petición HTTP	19782	✓	185519	121	3552	2958
98	16:10:50.353	Grupo de Hil...	Petición HTTP	20203	✓	185523	121	3931	3203
99	16:10:50.590	Grupo de Hil...	Petición HTTP	20222	✓	185558	121	3559	2968
100	16:10:50.292	Grupo de Hil...	Petición HTTP	22369	✓	185493	121	4684	3265

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Máx	% Error	Rendimie...	Kb/sec	Sent KB/s...
Petición ...	100	16490	16542	18930	19541	20222	11873	22369	0,00%	4,5/sec	809,39	0,53
Total	100	16490	16542	18930	19541	20222	11873	22369	0,00%	4,5/sec	809,39	0,53

## Pruebas unitarias

Con las pruebas unitarias se busca comprobar el estado de los procesos que se definen en los elementos de las pruebas

### 🚦 Búsqueda SQL:

```
@Test
public void testTestcliente() {
    System.out.println("testcliente");
    clidao instance = new clidao();
    List expResult = instance.testcliente();
    List result = instance.testcliente();
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

### 🚦 Búsqueda SQL por nombre:

```
public void testConsultarcliente() {
    System.out.println("consultarcliente");
    String nonbre = "";
    clidao instance = new clidao();
    List expResult = instance.consultarcliente(nonbre);
    List result = instance.consultarcliente(nonbre);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

# Plan de pruebas de software

Andres Felipe Olmos Rojas

## + Inserción SQL:

```
@Test
public void testActcliente() {
    System.out.println("actcliente");
    String nonbre = "";
    String apellido = "";
    int edad = 0;
    String correo = "";
    clidao instance = new clidao();
    instance.actcliente(nonbre, apellido, edad, correo);
    // TODO review the generated test code and remove the default call to fail
    //fail("The test case is a prototype.");
}
```

## + actualizado SQL:

```
@Test
public void testModcliente() {
    System.out.println("modcliente");
    String nonbre = "";
    String apellido = "";
    int edad = 0;
    String correo = "";
    int id = 0;
    clidao instance = new clidao();
    instance.modcliente(nonbre, apellido, edad, correo, id);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

## + borrado SQL:

```
@Test
public void testBorcliente() {
    System.out.println("borcliente");
    int id = 0;
    clidao instance = new clidao();
    instance.borcliente(id);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

## + las pruebas se adjuntaron de la siguiente manera:

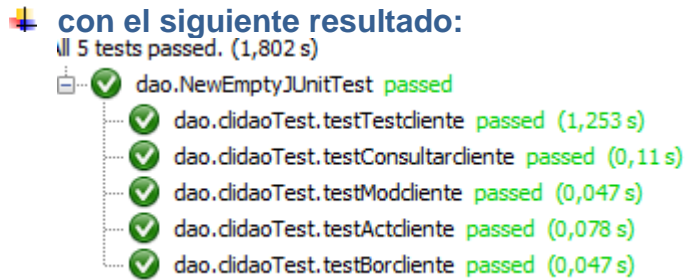
```
@RunWith(Suite.class)
@Suite.SuiteClasses({dao.clidaoTest.class})
public class NewEmptyJUnitTest {

}
```



# Plan de pruebas de software

Andres Felipe Olmos Rojas



## Criterios de aceptación o rechazo

### Criterios de aceptación o rechazo

Para que el proyecto sea aprobado se necesitara cubrir el 75% de las pruebas unitarias en estado aprobado no se tendrá que ver fallas en el código y se necesitaran realizar pruebas de estrés con al menos 100 usuarios en 10 segundos con errores minimos

### Criterios de suspensión

Se detendrá la realización de pruebas en caso de que mas del 30% de las pruebas unitarias sean fallidas o en el caso de que las pruebas de estrés no soporten mas de 10 usuarios en un segundo

# Plan de pruebas de software

*Andres Felipe Olmos Rojas*

## **Criterios de reanudación**

Se reanudará si las pruebas regresan a ser lo suficiente para considerarse aceptadas

# Plan de pruebas de software

*Andres Felipe Olmos Rojas*

## Recursos

### Requerimientos de entornos – Hardware

Se necesitará un equipo (portátil o pc de escritorio) que cuente con los siguientes requerimientos

<b>Sistema Operativo :</b>	Windows 7, Windows XP, Windows Vista (Windows XP Professional SP3/Vista SP1/Windows 7 Professional)
<b>Procesador :</b>	Intel Core i5 (Intel Core i5 or equivalent)
<b>RAM :</b>	2 GB, 4 GB (2 GB (32-bit), 4 GB (64-bit))
<b>Disco duro :</b>	1.5 GB HDD
<b>Tarjeta grafica :</b>	-
<b>Resolución de la pantalla :</b>	1024 x 728

### Requerimientos de entornos – Software

Se necesitará tener correctamente instalado NetBeans 8

# Plan de pruebas de software

*Andres Felipe Olmos Rojas*

## Herramientas de pruebas requeridas

Se necesitara tener correctamente instalado NetBeans 8 , jmeter para realizar las pruebas de estrés , y junit para las pruebas unitarias

## Personal

Se necesitara un técnico, tecnólogo o ingeniero en sistemas

## Entrenamiento

Se utilizara junit y jmeter aparte de ello no se necesitara nada mas

# Plan de pruebas de software

*Andres Felipe Olmos Rojas*

