IMPLEMENTACIÓN PRUEBAS SERVIDOR

Luisa Álvarez Valencia

TAPPI: TRIAGE APPLICATION

HISTORIAL DE VERSIONES

Versión	Cambios efectuados	Fecha de actualización
V 0.1	Introducción Estrategias de pruebas Información presente en la base de datos Pruebas unitarias	
V0.2	Realizacion pruebas unitarias Realizacion pruebas funcionales	
V0.3	Pruebas no funcionales	

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

TABLA DE CONTENIDO

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

1 INTRODUCCIÓN

1.1 PROPÓSITO DEL DOCUMENTO PLAN DE PRUEBAS SERVIDOR

El documento TG – Implementación_Pruebas_Servidor brinda la información necesaria que delimita las pruebas realizadas y los resultados de estas en el proyecto. Su público objetivo es el director del proyecto, el equipo de proyecto y equipo de pruebas. Se utilizaron dos estándares de la IEEE para la realización de dicho documento y los estándares son el 829 de 2008 [1] y 29119 de 2013 [2]–[5]. Como tal este documento contiene las pruebas y su numeración correspondiente, para conocer la trazabilidad y a que requerimiento o caso de uso pertenecen referirse a TG - Plan_Pruebas_Servidor.

2 ESTRATEGIA PARA LAS PRUEBAS

2.1 OBJETIVO [6]

El objetivo de la realización de pruebas es verificar que las funcionalidades del servidor de TAppi trabajen acorde a lo especificado.

2.2 SUPOSICIONES[6]

- Los datos requeridos para las pruebas están disponibles en el sistema previo a la realización de pruebas funcionales.
- Los defectos y problemas llevaran consigo una foto de pantalla de prueba.
- El sistema va a ser tratado como una caja negra, es decir si la información es mostrada correctamente en pantallas y en los reportes se asume que la base de datos está funcionando correctamente.

2.3 PRINCIPIOS PARA LAS PRUEBAS

- Las pruebas estarán basadas en los objetivos del proyecto.
- Los procesos para pruebas serán correctamente definidos, aunque flexibles para que estos poseen la habilidad de cambiar en dado caso de que sea necesario.
- Las pruebas son una actividad repetible, cuantificable y medible.
- Las pruebas presentan datos de entrada y salida.

2.4 CARACTERISTICAS

- A lo largo de las pruebas se mencionan diferentes métodos que presentan las clases que utiliza la aplicación:
    - POST: ingresa la información de la clase a la base de datos.
    - DELETE: elimina de la base de datos.
    - PUT: permite editar la información de una clase dado un identificador único.
    - GET: obtiene la información de una determinada clase dado un identificador único.

3 CARACTERISTICAS TECNICAS DE LOS DISPOSITIVOS PARA LAS PRUEBAS

Computador para pruebas:

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

Aspire ES1-711

PC name	idalu
Rename PC	
Organization	WORKGROUP
Connect to work or school	
Edition	Windows 10 Home
Version	1607
OS Build	14393.321
Product ID	00326-10000-00000-AA734
Processor	Intel(R) Pentium(R) CPU N3540 @ 2.16GHz 2.16 GHz
Installed RAM	8.00 GB (7.88 GB usable)
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Ilustración 1 información pc

Servidor:

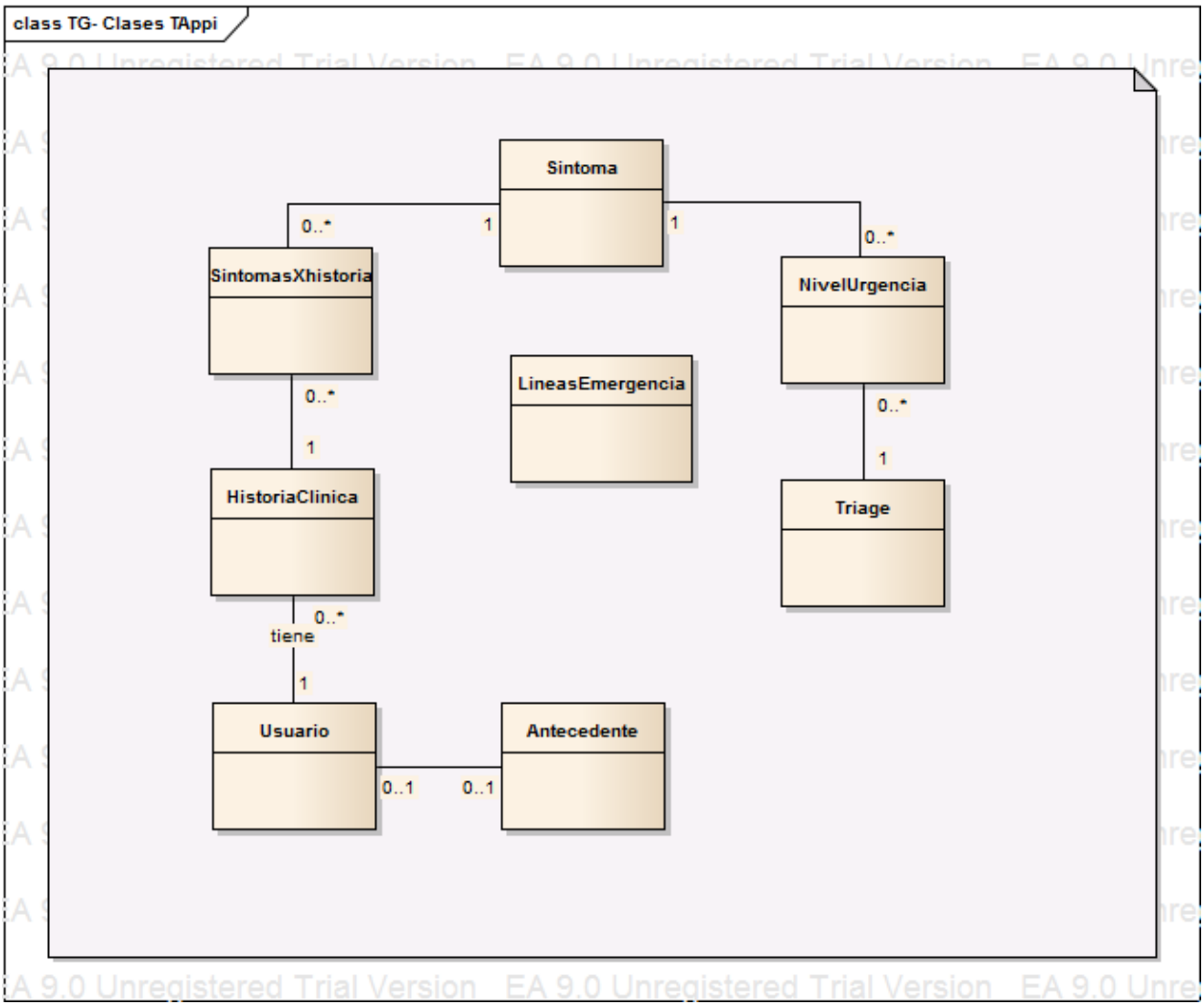
- Servidor alojado en http://donweb.com
- No Cores (CPU):2v
- Espacio en RAM: 2GB
- Espacio en Disco: 20 GB (SSD)
- Sistema operativo: Linux
- (Ubuntu 14.04.4 64 bits)
- Puertos TCP usados: o
- 1527, base de datos
- 8080, puerto para consultas de la aplicación HTTP
- 4848, consola de administración Glassfish
- Ip publica: 200.58.126.15
- Ip privada: 192.168.200.22
- Precio: $70.000 (COP) mensuales

4 INFORMACIÓN PRESENTE EN LA BASE DE DATOS

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

Para la realización de las pruebas es necesario que la base de datos contenga información insertada en sus tablas. A continuación se muestran los inserts en lenguaje SQL realizados para el ingreso de las tuplas, en las determinadas tablas, cuya información se va a utilizar como prueba y dichos datos se van a referenciar a lo largo del documento. Las imágenes de las tablas son extraidas de la implementación de:

http://200.58.126.15:8080/webTappiManager/faces/index.xhtml



4.1 ANTECEDENTE

4.1.1 SQL

```
INSERT INTO APP.ANTECEDENTE (IDANTECEDENTE, DESCRIPCION, TIPO)
  VALUES (1, 'CARDIACOS', 'CONGENITO');
INSERT INTO APP.ANTECEDENTE (IDANTECEDENTE, DESCRIPCION, TIPO)
  VALUES (2, 'TUBO NEURAL', 'CONGENITO');
INSERT INTO APP.ANTECEDENTE (IDANTECEDENTE, DESCRIPCION, TIPO)
  VALUES (3, 'ANANCEFALEA', 'CONGENITO');
```


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

```
INSERT INTO APP.ANTECEDENTE (IDANTECEDENTE, DESCRIPCION, TIPO)
  VALUES (4, 'MENINGOCELE', 'CONGENITO');
INSERT INTO APP.ANTECEDENTE (IDANTECEDENTE, DESCRIPCION, TIPO)
  VALUES (5, 'APENDICECTOMIA', 'QUIRURGICO');
INSERT INTO APP.ANTECEDENTE (IDANTECEDENTE, DESCRIPCION, TIPO)
  VALUES (6, 'COLECISTECTOMIA ABIERTA', 'QUIRURGICO');
INSERT INTO APP.ANTECEDENTE (IDANTECEDENTE, DESCRIPCION, TIPO)
  VALUES (7, 'HERNIA', 'QUIRURGICO');
INSERT INTO APP.ANTECEDENTE (IDANTECEDENTE, DESCRIPCION, TIPO)
  VALUES (8, 'ESQUIZOFRENIA', 'TRASTORNO MENTAL');
INSERT INTO APP.ANTECEDENTE (IDANTECEDENTE, DESCRIPCION, TIPO)
  VALUES (9, 'HOSPITALIZACIONES', 'MORBIDOS');
INSERT INTO APP.ANTECEDENTE (IDANTECEDENTE, DESCRIPCION, TIPO)
  VALUES (10, 'ALCOHOLISMO', 'MORBIDOS');
INSERT INTO APP.ANTECEDENTE (IDANTECEDENTE, DESCRIPCION, TIPO)
  VALUES (11, 'FUMAR', 'HABITOS');
INSERT INTO APP.ANTECEDENTE (IDANTECEDENTE, DESCRIPCION, TIPO)
  VALUES (12, 'PROZAC', 'MEDICAMENTOS');
INSERT INTO APP.ANTECEDENTE (IDANTECEDENTE, DESCRIPCION, TIPO)
  VALUES (13, 'INSULINA', 'MEDICAMENTOS');
INSERT INTO APP.ANTECEDENTE (IDANTECEDENTE, DESCRIPCION, TIPO)
  VALUES (14, 'PENICILINA', 'ALERGIAS');
INSERT INTO APP.ANTECEDENTE (IDANTECEDENTE, DESCRIPCION, TIPO)
  VALUES (15, 'BPH', 'INMUNIZACION');
INSERT INTO APP.ANTECEDENTE (IDANTECEDENTE, DESCRIPCION, TIPO)
  VALUES (16, 'TETANO', 'ALERGIAS');
```


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

4.1.2 Resultado tabla

Idantecedente	Descripcion	Tipo
1	CARDIACOS	CONGENITO
2	TUBO NEURAL	CONGENITO
3	ANANCEFALEA	CONGENITO
4	MENINGOCELE	CONGENITO
5	APENDICECTOMIA	QUIRURGICO
6	COLECISTECTOMIA ABIERTA	QUIRURGICO
7	HERNIA	QUIRURGICO
8	ESQUIZOFRENIA	TRASTORNO MENTAL
9	HOSPITALIZACIONES	MORBIDOS
10	ALCOHOLISMO	MORBIDOS
11	FUMAR	HABITOS

Tabla 1 antecedente

4.2 LINEAS DE EMERGENCIA

4.2.1 SQL

```
INSERT INTO APP.LINEASEMERGENCIA (IDLINEASEMERGENCIA, NUMERO, DESCRIPCION)
  VALUES (1, 136, 'CENTRO TOXOLOGICO');
INSERT INTO APP.LINEASEMERGENCIA (IDLINEASEMERGENCIA, NUMERO, DESCRIPCION)
  VALUES (2, 132, 'CRUZ ROJA COLOMBIA');
INSERT INTO APP.LINEASEMERGENCIA (IDLINEASEMERGENCIA, NUMERO, DESCRIPCION)
  VALUES (3, 125, 'EMERGENCIAS MEDICAS');
INSERT INTO APP.LINEASEMERGENCIA (IDLINEASEMERGENCIA, NUMERO, DESCRIPCION)
  VALUES (4, 155, 'ORIENTACIÓN MUJER VICTIMA VIOLENCIA');
```


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

4.2.2 Resultado tabla

Idlineasemergencia	Numero	Descripcion
1	136	CENTRO TOXOLOGICO
2	132	CRUZ ROJA COLOMBIA
3	125	EMERGENCIAS MEDICAS
4	155	ORIENTACIÓN MUJER VICTIMA VIOLENCIA

Tabla 2 lineas emergencia

4.3 SINTOMAS

4.3.1 EXTRAIDOS

En la parte inferior se encuentra un extracto de la información del ministerio del salud del cual se extraen los signos y sintomas para determinar una prioridad. Cabe resaltar que las reglas para cambiar esta información a un modelo persistente de datos se encuentra en TG – LogicaDelTriage.

Signos y síntomas	Prioridad 1	Prioridad 2	Prioridad 3	Prioridad 4	Prioridad 5
Respiratorios	Obstrucción de la vía aérea por cuerpo extraño, tapón de moco, trauma facial e inmersión.	Disnea con tirajes supraclaviculares, intercostales, aleteo nasal o estridor.	Hemoptisis moderada.	Tos con expectoración y fiebre	Tos seca o productiva hialina, rinorrea, malestar general con o sin fiebre.

Tabla 3 prioridad triage

[7]

4.3.2 SQL

```
INSERT INTO APP.SINTOMA (IDSINTOMA, DESCRIPCIONM, DESCRIPCIONP, COORDENADAX, COORDENADAY)
  VALUES (1, 'OBSTRUCCION VIA AEREA POR CUERPO EXTRANIO', 'OBJETO EN LA GARGANTA', 1, 1);
INSERT INTO APP.SINTOMA (IDSINTOMA, DESCRIPCIONM, DESCRIPCIONP, COORDENADAX, COORDENADAY)
  VALUES (2, 'TAPON DE MOCO', 'NO RESPIRAR POR MOCO', 1, 1);
INSERT INTO APP.SINTOMA (IDSINTOMA, DESCRIPCIONM, DESCRIPCIONP, COORDENADAX, COORDENADAY)
  VALUES (3, 'TRAUMA FACIAL', 'GOLPE EN LA CARA', 1, 1);
```


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

```
INSERT INTO APP.SINTOMA (IDSINTOMA, DESCRIPCIONM, DESCRIPCIONP,
COORDENADAX, COORDENADAY)
  VALUES (4, 'INMERSION', 'RESPIRAR AGUA', 1, 1);

--REGLAS TRIAGE 2
INSERT INTO APP.SINTOMA (IDSINTOMA, DESCRIPCIONM, DESCRIPCIONP,
COORDENADAX, COORDENADAY)
  VALUES (5, 'DISNEA', 'CUESTA RESPIRAR', 1, 1);
INSERT INTO APP.SINTOMA (IDSINTOMA, DESCRIPCIONM, DESCRIPCIONP,
COORDENADAX, COORDENADAY)
  VALUES (6, 'TIRAJES SUPRACLAVICULARES', 'SE VE RESPIRAR EN EL
HOMBRO', 1, 1);
INSERT INTO APP.SINTOMA (IDSINTOMA, DESCRIPCIONM, DESCRIPCIONP,
COORDENADAX, COORDENADAY)
  VALUES (7, 'TIRAJES INTERCOSTALES', 'EL PECHO DE VE DEPRIMIDO', 1, 1);
INSERT INTO APP.SINTOMA (IDSINTOMA, DESCRIPCIONM, DESCRIPCIONP,
COORDENADAX, COORDENADAY)
  VALUES (8, 'ALETEO NASAL', 'AGRANDAN LAS FOSAS NASALES AL
RESPIRAR', 1, 1);
---ESTE ULTIMO ES |
INSERT INTO APP.SINTOMA (IDSINTOMA, DESCRIPCIONM, DESCRIPCIONP,
COORDENADAX, COORDENADAY)
  VALUES (9, 'ESTRIDOR', 'SONIDO AGUDO AL RESPIRAR', 1, 1);

--REGLAS TRIAGE 3
INSERT INTO APP.SINTOMA (IDSINTOMA, DESCRIPCIONM, DESCRIPCIONP,
COORDENADAX, COORDENADAY)
  VALUES (10, 'HEMOPTISIS MODERADA', 'POCA SANGRE CON TOS', 1, 1);

--REGLA TRIAGE 4
INSERT INTO APP.SINTOMA (IDSINTOMA, DESCRIPCIONM, DESCRIPCIONP,
COORDENADAX, COORDENADAY)
  VALUES (11, 'TOS CON ESPECTORACION', 'SONIDO AGUDO AL RESPIRAR', 1,
1);

INSERT INTO APP.SINTOMA (IDSINTOMA, DESCRIPCIONM, DESCRIPCIONP,
COORDENADAX, COORDENADAY)
  VALUES (12, 'FIEBRE', 'FIEBRE', 1, 1);

--REGLA TRIAGE 5
```


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

```
INSERT INTO APP.SINTOMA (IDSINTOMA, DESCRIPCIONM, DESCRIPCIONP, COORDENADAX, COORDENADAY)
  VALUES (13, 'TOS SECA', 'TOS SECA', 1, 1);
---ESTE ES |
INSERT INTO APP.SINTOMA (IDSINTOMA, DESCRIPCIONM, DESCRIPCIONP, COORDENADAX, COORDENADAY)
  VALUES (14, 'TOS PRODUCTIVA HIALINA', 'TOS CON FLEMA AMARILLA', 1, 1);
INSERT INTO APP.SINTOMA (IDSINTOMA, DESCRIPCIONM, DESCRIPCIONP, COORDENADAX, COORDENADAY)
  VALUES (15, 'RINORREA', 'GOTEA MOCO DESDE LA NARIZ', 1, 1);
INSERT INTO APP.SINTOMA (IDSINTOMA, DESCRIPCIONM, DESCRIPCIONP, COORDENADAX, COORDENADAY)
  VALUES (16, 'MALESTAR GENERAL', 'MAL ESTADO', 1, 1);
```

4.3.3 Resultado tablas

Idsintoma	Descripcionm	Descripcionp	Coordenadax	Coordenaday
1	OBSTRUCCION VIA AEREA POR CUERPO EXTRANIO	OBJETO EN LA GARGANTA	1	1
2	TAPON DE MOCO	NO RESPIRAR POR MOCO	1	1
3	TRAUMA FACIAL	GOLPE EN LA CARA	1	1
4	INMERSION	RESPIRAR AGUA	1	1
5	DISNEA	CUESTA RESPIRAR	1	1
6	TIRAJES SUPRACLAVICULARES	SE VE RESPIRAR EN EL HOMBRO	1	1
7	TIRAJES INTERCOSTALES	EL PECHO DE VE DEPRIMIDO	1	1
8	ALETEO NASAL	AGRANDAN LAS FOSAS NASALES AL RESPIRAR	1	1
9	ESTRIDOR	SONIDO AGUDO AL RESPIRAR	1	1
10	HEMOPTISIS MODERADA	POCA SANGRE CON TOS	1	1
11	TOS CON ESPECTORACION	SONIDO AGUDO AL RESPIRAR	1	1
12	FIEBRE	FIEBRE	1	1
13	TOS SECA	TOS SECA	1	1
14	TOS PRODUCTIVA HIALINA	TOS CON FLEMA AMARILLA	1	1
15	RINORREA	GOTEA MOCO DESDE LA NARIZ	1	1
16	MALESTAR GENERAL	MAL ESTADO	1	1

Tabla 4 sintomas

4.4 TRIAGE

4.4.1 SQL

```
INSERT INTO APP.TRIAGE (IDTRIAGE, NIVELTRIAGE, DX)
  VALUES (1, 1, 'PROBLEMAS RESPIRATORIOS');
INSERT INTO APP.TRIAGE (IDTRIAGE, NIVELTRIAGE, DX)
  VALUES (2, 2, 'PROBLEMAS RESPIRATORIOS');
INSERT INTO APP.TRIAGE (IDTRIAGE, NIVELTRIAGE, DX)
  VALUES (3, 2, 'PROBLEMAS RESPIRATORIOS');
```


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

```
INSERT INTO APP.TRIAGE (IDTRIAGE, NIVELTRIAGE, DX)
  VALUES (4, 3, 'PROBLEMAS RESPIRATORIOS');
INSERT INTO APP.TRIAGE (IDTRIAGE, NIVELTRIAGE, DX)
  VALUES (5, 4, 'PROBLEMAS RESPIRATORIOS');
INSERT INTO APP.TRIAGE (IDTRIAGE, NIVELTRIAGE, DX)
  VALUES (6, 5, 'PROBLEMAS RESPIRATORIOS');
INSERT INTO APP.TRIAGE (IDTRIAGE, NIVELTRIAGE, DX)
  VALUES (6, 5, 'PROBLEMAS RESPIRATORIOS');
INSERT INTO APP.TRIAGE (IDTRIAGE, NIVELTRIAGE, DX)
  VALUES (6, 5, 'PROBLEMAS RESPIRATORIOS');
INSERT INTO APP.TRIAGE (IDTRIAGE, NIVELTRIAGE, DX)
  VALUES (6, 5, 'PROBLEMAS RESPIRATORIOS');
```

4.4.2 Resultado tablas

Idtriage	Niveltriage	Dx
1	1	PROBLEMAS RESPIRATORIOS
2	2	PROBLEMAS RESPIRATORIOS
3	2	PROBLEMAS RESPIRATORIOS
4	3	PROBLEMAS RESPIRATORIOS
5	4	PROBLEMAS RESPIRATORIOS
6	5	PROBLEMAS RESPIRATORIOS
7	5	PROBLEMAS RESPIRATORIOS
8	5	PROBLEMAS RESPIRATORIOS
9	5	PROBLEMAS RESPIRATORIOS

Tabla 5 triage

4.5 NIVEL URGENCIA

4.5.1 SQL

```
INSERT INTO APP.NIVELURGENCIA (SINTOMA_IDSINTOMA,
TRIAGE_IDTRIAGE, PRESENCIA)
  VALUES (1, 1, 'SI');
INSERT INTO APP.NIVELURGENCIA (SINTOMA_IDSINTOMA,
TRIAGE_IDTRIAGE, PRESENCIA)
  VALUES (2, 1, 'SI');
```


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

```
INSERT INTO APP.NIVELURGENCIA (SINTOMA_IDSINTOMA, TRIAGE_IDTRIAGE, PRESENCIA)
  VALUES (3, 1, 'SI');
INSERT INTO APP.NIVELURGENCIA (SINTOMA_IDSINTOMA, TRIAGE_IDTRIAGE, PRESENCIA)
  VALUES (4, 1, 'SI');

INSERT INTO APP.NIVELURGENCIA ( TRIAGE_IDTRIAGE, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (2, 5, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (2, 6, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (2, 7, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (2, 8, 'SI');

INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (3, 5, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (3, 6, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (3, 7, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (3, 9, 'SI');

INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (4, 10, 'SI');

INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (5, 11, 'SI');
```


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

```
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE,
SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (5, 12, 'SI');

INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE,
SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (6, 13, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE,
SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (6, 15, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE,
SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (6, 16, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE,
SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (6, 12, 'SI');

INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE,
SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (7, 13, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE,
SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (7, 15, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE,
SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (7, 16, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE,
SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (7, 12, 'NO');


INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE,
SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (8, 14, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE,
SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (8, 15, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE,
SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (8, 16, 'SI');
```


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

```
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (8, 12, 'SI');


INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (9, 14, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (9, 15, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (9, 16, 'SI');
INSERT INTO APP.NIVELURGENCIA (TRIAGE_IDTRIAGE, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES (9, 12, 'NO');
```

4.5.2 Resultado tabla

Presencia	Sintoma	Triage
SI	1	1
SI	2	1
SI	3	1
SI	4	1
SI	5	2
SI	6	2
SI	7	2
SI	8	2
SI	5	3
SI	6	3
SI	7	3

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

SI	9	3
SI	10	4
SI	11	5
SI	12	5
SI	13	6
SI	15	6
SI	16	6
SI	12	6
SI	13	7
SI	15	7
SI	16	7
NO	12	7
SI	14	8
SI	15	8
SI	16	8
SI	12	8
SI	14	9
SI	15	9
SI	16	9
NO	12	9

Tabla 6 nivel urgencia

4.6 USUARIO

4.6.1 SQL

INSERT INTO APP.USUARIO (IDUSUARIO, IDFACEGOO, APELLIDO, CEDULA, CORREO, DIRECCION, FECHANACIMIENTO, GENERO, NOMBRE, ROL, TIPOSANGRE, EPS, ABORTOS, CESAREAS, FECHAPRIMERPERIODO, FECHAULTIMOPERIODO, GESTAS, METODOANTICONCEPTIVO, PARTOS)

VALUES (1, '165465', 'ALVAREZ', 1018445478, 'ALVAREZ.LUISA@JAVERIANA.EDU.CO', 'CR 40 # 92 -4', '1991-10-11', 'F', 'LUISA', 'M', 'O+', 'CAFESALUD', 0, 0, '1999-10-11', '2016-10-11', 0, 'PILDORAS', 0)

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

INSERT INTO APP.USUARIO (IDUSUARIO, IDFACEGOO, APELLIDO, CEDULA, CORREO, DIRECCION, FECHANACIMIENTO, GENERO, NOMBRE, ROL, TIPOSANGRE, EPS, ABORTOS, CESAREAS, FECHAPRIMERPERIODO, FECHAULTIMOPERIODO, GESTAS, METODOANTICONCEPTIVO, PARTOS)

VALUES (2, '455787878', 'VALENCIA', 75555757, 'VALENCIA.@GMAIL.COM', 'CR 40 #9', '1998-10-11', 'M', 'FERNANDO', 'P', 'AB+', 'CAFESALUD', 0, 0, '0001-10-11', '0001-01-18', 0, '0', 0)

4.6.2 Resultado tabla

Idusuario	Idfacegoo	Apellido	Cedula	Correo	Direccion	Fechanacimiento	Genero	Nombre	Rol
1	165465	ALVAREZ	1018445478	ALVAREZ.LUISA@JAVERIANA.EDU.CO	CR 40 # 92 -4	10/11/1991	F	LUISA	M
2	455787878	VALENCIA	75555757	VALENCIA.@GMAIL.COM	CR 40 #9	10/11/1998	M	FERNANDO	P

Tabla 7 usuario

4.7 HISTORIA CLINICA

4.7.1 SQL

INSERT INTO APP.HISTORIACLINICA (IDHISTORIACLINICA, DESCRIPCIONMOTIVOURGENCIA, INICIOMOTIVOURGENCIA, USUARIO_IDUSUARIO, FECHA, NIVELTRIAGE)

VALUES (1, 'NO PUEDO RESPIRAR', '2016-10-11', 1, '2016-10-11', DEFAULT)

INSERT INTO APP.HISTORIACLINICA (IDHISTORIACLINICA, DESCRIPCIONMOTIVOURGENCIA, INICIOMOTIVOURGENCIA, USUARIO_IDUSUARIO, FECHA, NIVELTRIAGE)

VALUES (2, 'ME DUELE LA NARIZ', '2016-10-11', 1, '2016-10-11', DEFAULT)

INSERT INTO APP.HISTORIACLINICA (IDHISTORIACLINICA, DESCRIPCIONMOTIVOURGENCIA, INICIOMOTIVOURGENCIA, USUARIO_IDUSUARIO, FECHA, NIVELTRIAGE)

VALUES (3, 'ME DUELE EL PECHO', '2016-10-11', 2, CURRENT_DATE, DEFAULT)

INSERT INTO APP.HISTORIACLINICA (IDHISTORIACLINICA, DESCRIPCIONMOTIVOURGENCIA, INICIOMOTIVOURGENCIA, USUARIO_IDUSUARIO, FECHA, NIVELTRIAGE)

VALUES (4, 'TENGO COMO GRIPA', '2016-10-11', 1, '2016-10-11', DEFAULT)

INSERT INTO APP.HISTORIACLINICA (IDHISTORIACLINICA, DESCRIPCIONMOTIVOURGENCIA, INICIOMOTIVOURGENCIA, USUARIO_IDUSUARIO, FECHA, NIVELTRIAGE)

VALUES (5, 'NO PUEDO RESPIRAR', '2016-10-11', 1, '2016-10-11', DEFAULT)

INSERT INTO APP.HISTORIACLINICA (IDHISTORIACLINICA, DESCRIPCIONMOTIVOURGENCIA, INICIOMOTIVOURGENCIA, USUARIO_IDUSUARIO, FECHA, NIVELTRIAGE)

VALUES (6, 'MUCHO MOCO', '2016-10-11', 2, CURRENT_DATE, DEFAULT)

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

4.7.2 Resultado tabla

Idhistoriaclinica	Descripcionmotivourgencia	Iniciomotivourgencia	Fecha	Niveltriage	Usuarioldusuario
1	NO PUEDO RESPIRAR	10/11/2016	10/11/2016	5	1
2	ME DUELE LA NARIZ	10/11/2016	10/11/2016	5	1
3	ME DUELE EL PECHO	10/11/2016	10/11/2016	5	2
4	TENGO COMO GRIPA	10/11/2016	10/11/2016	5	1
5	NO PUEDO RESPIRAR	10/11/2016	10/11/2016	5	1
6	MUCHO MOCO	10/11/2016	10/11/2016	5	2

Tabla 8 historia clinica

4.8 SINTOMASXHISTORIA

4.8.1 SQL

---TRIAGE 1

INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
VALUES ('2016-10-11', 1, 1, 'SI');

INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
VALUES ('2016-10-11', 1, 2, 'SI');

INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
VALUES ('2016-10-11', 1, 3, 'SI');

INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
VALUES ('2016-10-11', 1, 4, 'SI');

---TRIAGE 2

INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
VALUES ('2016-10-11', 2, 5, 'SI');

INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
VALUES ('2016-10-11', 2, 6, 'SI');

INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
VALUES ('2016-10-11', 2, 7, 'SI');

INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
VALUES ('2016-10-11', 2, 8, 'SI');

---TRIAGE 3

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

```
INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES ('2016-10-11', 3, 10, 'SI');

---TRIAGE 4
INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES ('2016-10-11', 4, 11, 'SI');
INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES ('2016-10-11', 4, 12, 'SI');

---TRIAGE 5
INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES ('2016-10-11', 5, 14, 'SI');
INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES ('2016-10-11', 5, 15, 'SI');
INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES ('2016-10-11', 5, 16, 'SI');
INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES ('2016-10-11', 5, 12, 'NO');

----TRIAGE 0
INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES ('2016-10-11', 6, 16, 'SI');
INSERT INTO APP.SINTOMASXHISTORIA (FECHA, HISTORIACLINICA_IDHC, SINTOMA_IDSINTOMA, PRESENCIA)
  VALUES ('2016-10-11', 6, 12, 'NO');
```


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

4.8.2 Resultado tablas

Fecha	Presencia	Historiaclinica	Sintoma
10/11/2016	NO	6	12
10/11/2016	SI	1	1
10/11/2016	SI	1	2
10/11/2016	SI	1	3
10/11/2016	SI	1	4
10/11/2016	SI	2	5
10/11/2016	SI	2	6
10/11/2016	SI	2	7
10/11/2016	SI	2	8
10/11/2016	SI	3	10
10/11/2016	SI	4	11
10/11/2016	SI	4	12
10/11/2016	SI	5	14
10/11/2016	SI	5	15
10/11/2016	SI	5	16
10/11/2016	NO	5	12
10/11/2016	SI	6	16

Tabla 9 sintomasxhistoria

4.9 USUARIOXANTECEDENTE

4.9.1 SQL

```
INSERT INTO APP.USUARIOXANTECEDENTE (USUARIO_IDUSUARIO, ANTECEDENTE_IDANTECEDENTE, AUX)
  VALUES (1, 1, DEFAULT);
INSERT INTO APP.USUARIOXANTECEDENTE (USUARIO_IDUSUARIO, ANTECEDENTE_IDANTECEDENTE, AUX)
  VALUES (1, 2, DEFAULT);
INSERT INTO APP.USUARIOXANTECEDENTE (USUARIO_IDUSUARIO, ANTECEDENTE_IDANTECEDENTE, AUX)
  VALUES (1, 5, DEFAULT);
```


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

```
INSERT INTO APP.USUARIOXANTECEDENTE (USUARIO_IDUSUARIO,
ANTECEDENTE_IDANTECEDENTE, AUX)
  VALUES (2, 7, DEFAULT);
INSERT INTO APP.USUARIOXANTECEDENTE (USUARIO_IDUSUARIO,
ANTECEDENTE_IDANTECEDENTE, AUX)
  VALUES (2, 2, DEFAULT);
INSERT INTO APP.USUARIOXANTECEDENTE (USUARIO_IDUSUARIO,
ANTECEDENTE_IDANTECEDENTE, AUX)
  VALUES (2, 4, DEFAULT);
```

4.9.2 Resultado tablas

Antecedente	Usuario
1	1
2	1
5	1
7	2
2	2
4	2

Tabla 10 usuario por antecedente

5 UNIT TESTING Y PRUEBAS DE CONFORMIDAD (CONFORMANCE)

Las pruebas unitarias son aquellas me miden la unidad de prueba contra los requerimientos, Medir significa utilizar datos de ejemplo para probar la unidad y compararlos con el comportamiento real que es requerido y especificado por los requerimientos de esa unidad [8]. Estas por lo tanto se pueden asociar con las pruebas de conformidad ya que estas se obtienen al demostrar que todos los requerimientos para unos determinados procesos han sido satisfechos en el proyecto [3].

5.1 HERRAMIENTA

La herramienta para este tipo de prueba es JUnit, es un framework de código abierto que sirve para escribir y correr pruebas de manera repetible. Es una instancia de la arquitectura xUnit, este incluye aserciones para probar resultados esperados, pruebas fijas para compartir datos de prueba y ejecutores de pruebas [9].

5.2 ELEMENTOS A SER PROBADOS

En la tabla a continuación se muestran los requerimientos más importantes involucrados en las funcionalidades más relevantes las cuales se plantearon se debían cumplir en la propuesta de trabajo de grado y que fueron transformadas a casos de uso. Adicionalmente se tuvo en cuenta la priorización de requerimientos realizada en el SRS para evaluar que pruebas se

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

consideraban más importantes. La persona responsable de llevar a cabo las pruebas de servidor es Luisa Álvarez Valencia

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

Id Prueba	Prueba a ser realizada	Caracteristicas prueba	
		Entradas	Salidas
1	Se debe probar que dado un id de Facebook o Google + el sistema retorne el id de usuario para identificar dicho usuario en la aplicación.	Al ingresar el id de Facebook = 165465	El id del usuario que es Luisa Álvarez idUsuario = 1
2	Se puede identificar un usuario que si existe en el sistema y con ello demostrar que dicho usuario en el login si existe.	Ver Prueba 1	
3	Se debe probar el método de PUT para la clase Usuario en el servidor.	El idUsuario = 1 editar el campo de nombre y cambiarlo a Lorenza	El usuario con identificador 1 su nombre ahora debe ser Lorenza
4	Se debe probar el método de POST para la clase HistoriaClinica y para la de SintomasXHistoria	Crear HistoriaClinica con IDHISTORIACLINICA =7 , DESCRIPCIONMOTIVOURGENCIA = DOLOR NARIZ, INICIOMOTIVOURGENCIA = FECHA_ACTUAL, USUARIO_IDUSUARIO= 2 FECHA = , NIVELTRIAGE = default Se agregan los siguientes sintomas a la HistoriaClinica con idHistoria= 7 Idsintoma 1, presencia = SI Idsintoma 2, presencia = SI Idsintoma 3, presencia = SI Idsintoma 4, presencia = SI	Se agrega una historia clínica con las características descritas y se observa en el modelo de dominio. Se agregan los síntomas a la historia clínica y este persiste en la base de datos
5	El sistema debe verificar que si las variables son de un tipo de dato específico estas cumplan con las características	Intentar ingresar un síntoma donde IDSINTOMA= bla, DESCRIPCIONM = hola, DESCRIPCIONP = hola, COORDENADAX=1, COORDENADAY=1.	No debe permitir su ingreso ya que id síntoma es de tipo numérico.

6	Se debe probar el método de DELETE en Usuario	Crear un usuario con la siguiente información (IDUSUARIO, IDFACEGOO, APELLIDO, CEDULA, CORREO, DIRECCION, FECHANACIMIENTO, GENERO, NOMBRE, ROL, TIPOSANGRE, EPS, ABORTOS, CESAREAS, FECHAPRIMERPERIODO, FECHAULTIMOPERIODO, GESTAS, METODOANTICONCEPTIVO, PARTOS) VALUES (3, '123', 'VALIA', 75555757, 'VALENCIA.@GMAIL.COM', 'CR 40 #9', '1998-10-11', 'M', 'FERNAN', 'P', 'AB+', 'CAFESALUD', 0, 0, '0001-10-11', '0001-01-18', 0, '0', 0) Dado el idUsuario = 3 borrarlo	Ver en la base de datos la creación del usuario, se creó uno nuevo para no borrar uno de los existentes y dañar caso de prueba. Observar en la base de datos que el usuario ingresado ya no existe.
7	El sistema debe probar que si la cuenta no existe no se pueda acceder a ella.	Buscar un usuario de idUsuario =3 Ver prueba 6	Debido a que el id usuario 3 no existe no debe aparecer información.
8	Se debe verificar que dicha cuenta eliminada ya no se encuentre en la base de datos	Ver prueba 7	
9	Se debe verificar que el método de GET en la clase LineasEmergencia funcione.	Obtener todas las lineas de emergencia	Se deben observar todas la líneas de emergencia
10	Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= congénito funcione.	Dado un tipo específico de antecedente se tiene acceso a estos. Tipo = congenito	Se ven los antecedentes de tipo congénito.
11	Se debe verificar que el POST de HistoriaClinica funcione puesto que este es quien posee el atributo.	Ver prueba 4	
12	Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= quirúrgico funcione.	Dado un tipo especifico de antecedente se tiene acceso a estos. Tipo = quirurgico	Se ven los antecedentes de tipo específico.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo). Utiliza una licencia Creative Commons.

13	Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= Trastorno mental funcione.	Dado un tipo específico de antecedente se tiene acceso a estos. Tipo = trastorno mental	Se ven los antecedentes de tipo específico.
14	El método de GET de la clase Síntomas trae los síntomas en términos médicos y en términos coloquiales.	Obtener la lista de sintomas.	Se obtiene una lista de sintomas que tiene el proyecto.
15	Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= mórbidos funcione. Y que estos sean almacenados en AntecedenteXUsuario.	Dado un tipo específico de antecedente se tiene acceso a estos. Tipo = mórbidos Agregar al usuario con idUsuario= 2, el antecedente con idantecedente =9.	Se ven los antecedentes de tipo específico. Se ven en la base de datos que el antecedente fue agregado al usuario correspondiente.
16	Se debe verificar que el Usuario cuando se cree ingrese su información ginecoobstetrica y por lo tanto el POST debe funcionar.	Ver prueba 6	
17	Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= hábitos funcione. Y que estos sean almacenados en AntecedenteXUsuario.	Dado un tipo específico de antecedente se tiene acceso a estos. Tipo= hábitos Agregar al usuario con idUsuario= 2, el antecedente con idantecedente = 11.	Se ven los antecedentes de tipo específico. Se ven en la base de datos que el antecedente fue agregado al usuario correspondiente.
18	Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= medicamentos funcione. Y que estos sean almacenados en AntecedenteXUsuario.	Dado un tipo específico de antecedente se tiene acceso a estos. Tipo= medicamentos Agregar al usuario con idUsuario= 2, el antecedente con idantecedente = 13.	Se ven los antecedentes de tipo específico. Se ven en la base de datos que el antecedente fue agregado al usuario correspondiente.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo). Utiliza una licencia Creative Commons.

19	Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= alergias funcione. Y que estos sean almacenados en AntecedenteXUsuario.	Dado un tipo específico de antecedente se tiene acceso a estos. Tipo = alergias Agregar al usuario con idUsuario= 2, el antecedente con idantecedente = 14.	Se ven los antecedentes de tipo específico. Se ven en la base de datos que el antecedente fue agregado al usuario correspondiente.
20	Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= familiar funcione	Dado un tipo específico de antecedente se tiene acceso a estos. Tipo= familiar	Se ven los antecedentes de tipo específico.
21	Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= inmunizaciones funcione. Y que estos sean almacenados en AntecedenteXUsuario.	Dado un tipo específico de antecedente se tiene acceso a estos. Tipo= inmunizaciones Agregar al usuario con idUsuario= 1, el antecedente con idantecedente = 15.	Se ven los antecedentes de tipo específico. Se ven en la base de datos que el antecedente fue agregado al usuario correspondiente.

Tabla 11: req funcionales

La información contenida se considera pública (a menos que explícitamente se indique de otro modo). Utiliza una licencia Creative Commons.

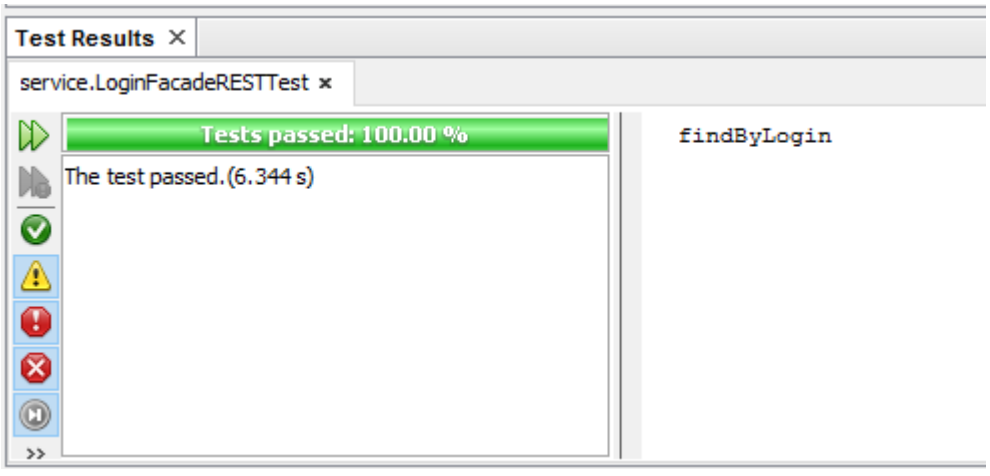
5.3 RESULTADO PRUEBAS

5.3.1 Prueba 1 y 2

La fachada que se utiliza para hacer el login por Facebook o Google, no tiene EJB, por lo tanto la creación de pruebas desde JUnit de manera automática no se permite, es por ello que para esta prueba es necesario crear un cliente local con un proxy que consuma los servicios a manera de XML y con ello se prueba la funcionalidad del código. En la parte inferior se puede observar el código de la prueba en JUnit.

```
/**
 * Test of findByLogin method, of class LoginFacadeREST.
 * Prueba 1: Se debe probar que dado un id de Facebook o Google + el sistema retorne
 * el id de usuario para identificar dicho usuario en la aplicación.
 */
@Test
public void testFindByLogin() throws NamingException {


    System.out.println("findByLogin");
    String fg = "165465";
    loginRestClient l = new loginRestClient();
    Integer expResult = new Integer("1");
    Usuario result = l.findByLogin_XML(Usuario.class, fg);
    assertEquals(expResult, result.getIdusuario());
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

En la parte inferior se puede observar el resultado de la prueba

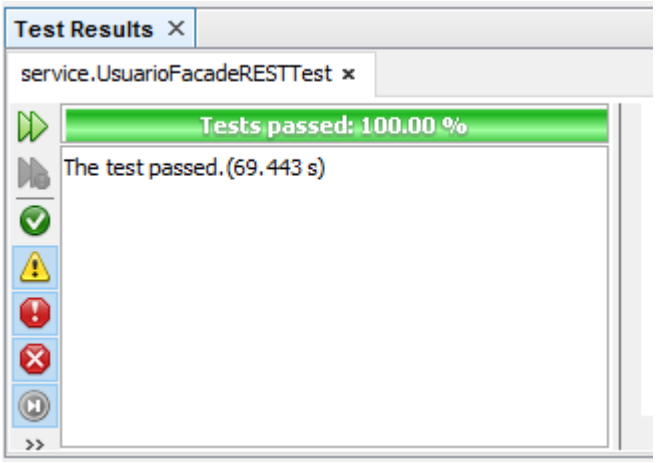


5.3.2 Prueba 3

En la parte inferior se ve el código de JUnit para la prueba

```java
/**
 * Test of edit method, of class UsuarioFacadeREST.
 * Se debe probar el método de PUT para la clase Usuario en el servidor.
 */
@Test
public void testEdit_Integer_Usuario() throws Exception {
    System.out.println("edit");
    Integer id = 1;
    Usuario entity = null;
    EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
    UsuarioFacadeREST instance =
            (UsuarioFacadeREST)container.getContext().lookup("java:global/classes/UsuarioFacadeREST");
    entity = instance.find(id);
    String expResult = "Lorenza";
    entity.setNombre(expResult);
    instance.edit(id, entity);
    entity = instance.find(id);
    assertEquals(expResult, entity.getNombre());
    container.close();
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

En la parte inferior se ve el resultado de la prueba.

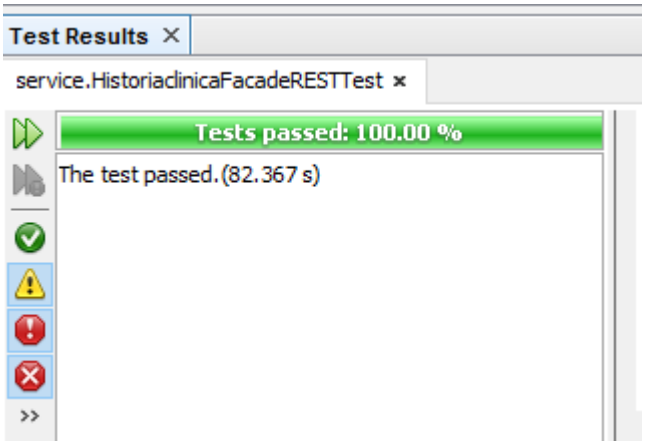


5.3.3 Prueba 4 y 11

En la parte inferior se puede observar el código utilizado para crear la prueba pertinente. Se encuentra primero como crear una historia clínica.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

```
public void testCreate() throws Exception {
    System.out.println("create");
    Historiaclinica entity = new Historiaclinica();
    entity.setIdhistoriaclinica(7);
    entity.setDescripcionmotivourgencia("DOLOR NARIZ");
    entity.setFecha(new Date(2016,10,12));
    entity.setIniciomotivourgencia(new Date(2016,10,12));
    Usuario u = null;

    EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
    UsuarioFacadeREST instance1 =
            (UsuarioFacadeREST)container.getContext().lookup("java:global/classes/UsuarioFacadeREST");
    u = instance1.find(2);
    entity.setUsuarioIdusuario(u);
    HistoriaclinicaFacadeREST instance =
            (HistoriaclinicaFacadeREST)container.getContext().lookup("java:global/classes/HistoriaclinicaFacadeREST");
    instance.create(entity);
    assertEquals(new Integer (7), entity.getIdhistoriaclinica());
    container.close();
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

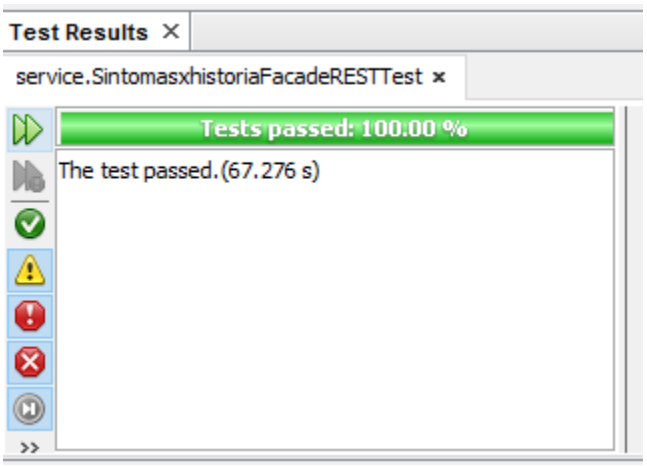


Posteriormente se realizó la creación de los diferentes síntomas asociados a la historia clínica como se muestra a continuación.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

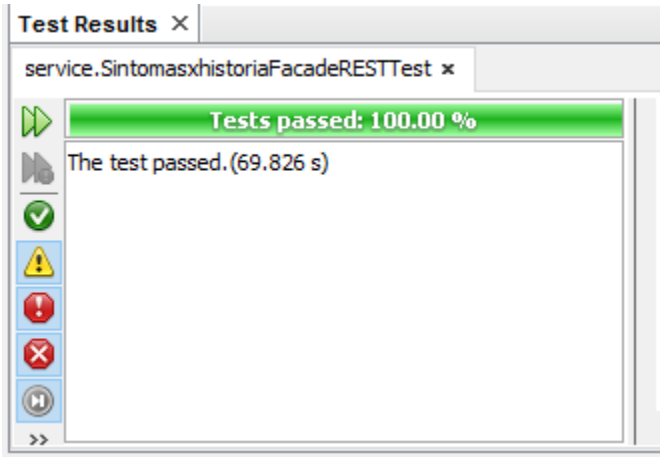
```
@Test
public void testCreate() throws Exception {
    System.out.println("create");
    Sintomasxhistoria entity;
    entity = new Sintomasxhistoria();
    EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
    HistoriaclinicaFacadeREST instance1 =
            (HistoriaclinicaFacadeREST )
            container.getContext().lookup("java:global/classes/HistoriaclinicaFacadeREST");
    Historiaclinica h = null;
    h = instance1.find(7);
    entity.setHistoriaclinica(h);
    entity.setFecha(new Date(2016,10,11));
    entity.setPresencia("SI");

    Sintoma s =null;
    SintomaFacadeREST instance2 =
            (SintomaFacadeREST )
            container.getContext().lookup("java:global/classes/SintomaFacadeREST");
    s = instance2.find(1);
    entity.setSintoma(s);
    SintomasxhistoriaPK sh;
    sh = new SintomasxhistoriaPK();
      sh.setHistoriaclinicaIdhc(7);
      sh.setSintomaIdsintoma(1);

      entity.setSintomasxhistoriaPK(sh);
      SintomasxhistoriaFacadeREST instance =
              (SintomasxhistoriaFacadeREST)
              container.getContext().lookup("java:global/classes/SintomasxhistoriaFacadeREST");
      instance.create(entity);
      container.close();
      assertEquals(new Integer(1), entity.getSintoma().getIdsintoma());
      // TODO review the generated test code and remove the default call to fail.
      //fail("The test case is a prototype.");
  }
```

Prueba primer sintoma

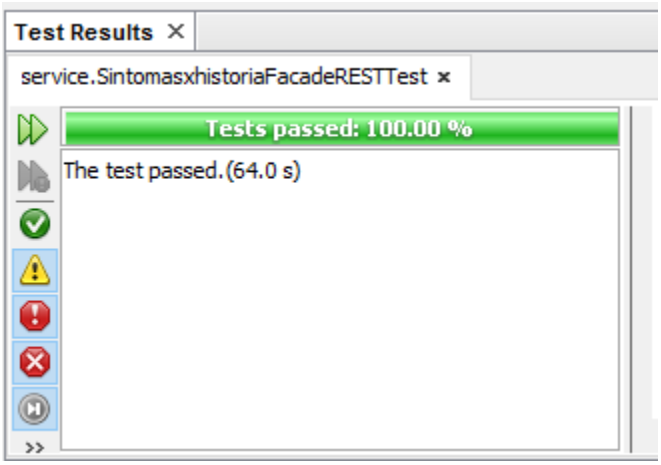


Prueba Segundo sintoma

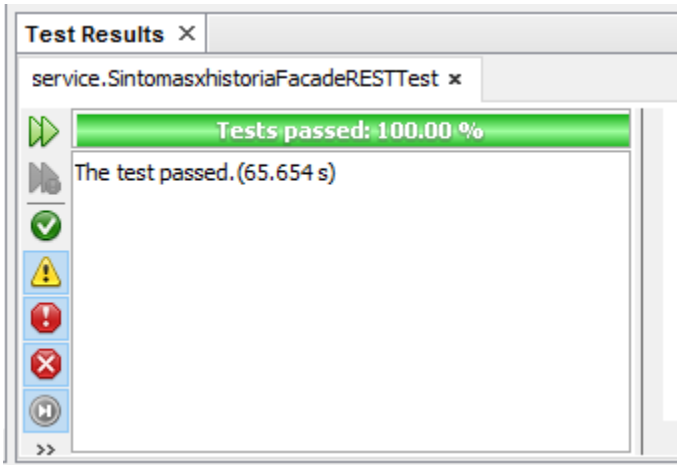
La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.



Prueba tercer sintoma



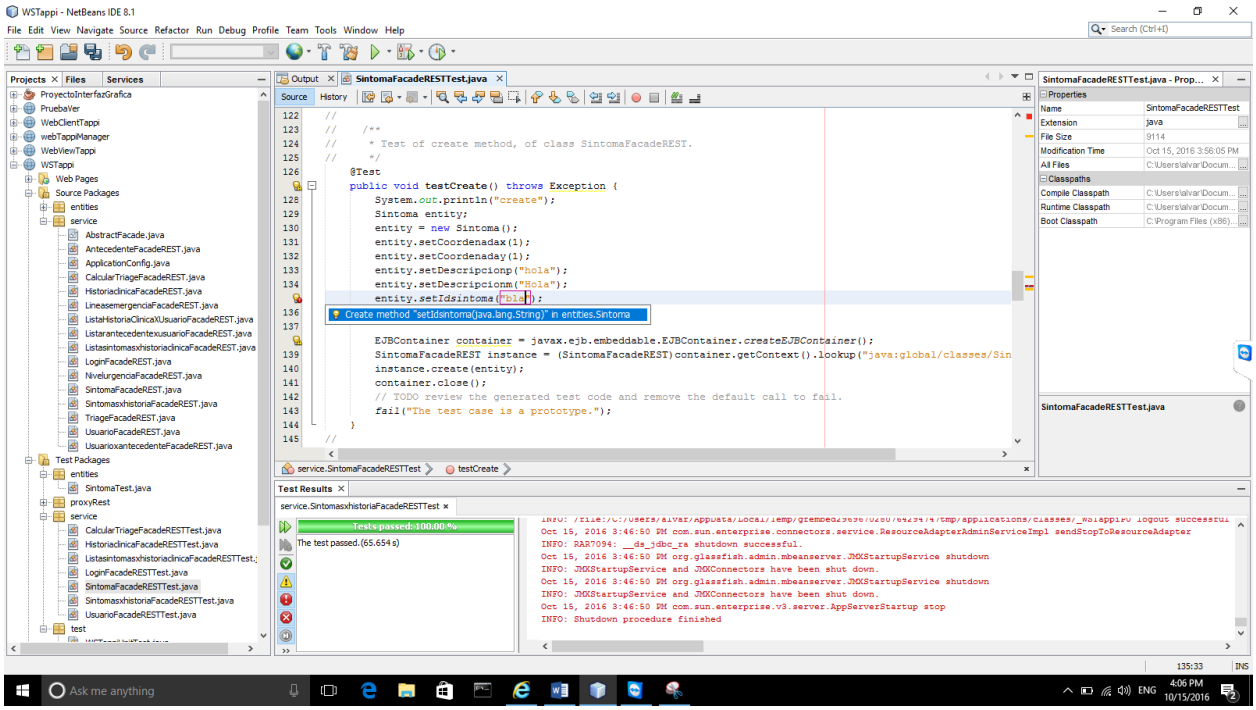
Prueba cuarto sintoma



5.3.4 Prueba 5

En la parte inferior se puede observar la prueba realizada.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.



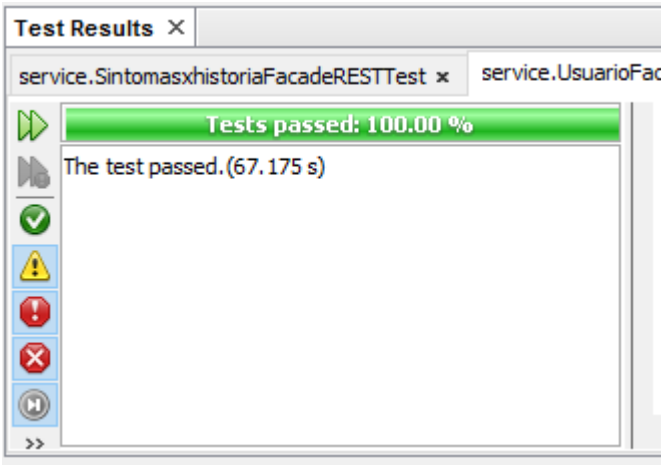
5.3.5 Prueba 6

En la parte inferior se puede observar la creación de un usuario que luego este será borrado.

```
@Test
public void testCreate() throws Exception {
    System.out.println("create");
    Usuario entity = new Usuario ();
    entity.setIdusuario(3);
    entity.setIdfacegoo("123");
    entity.setCedula(755555757);
    entity.setCorreo("VALENCIA.@GMAIL.COM");
    entity.setDireccion("CR 40 #9");
    entity.setFechanacimiento(new Date(1998,10,11));
    entity.setGenero('M');
    entity.setNombre("FERNAN");
    entity.setRol("P");
    entity.setTiposangre("AB+");
    entity.setEps("CAFESALUD");
    entity.setAbortos(0);
    entity.setCesareas(0);
    entity.setFechaprimerperiodo(new Date(0001,10,11));
    entity.setFechaultimoperiodo(new Date (0001,01,18));
    entity.setMetodoanticonceptivo("");
    entity.setGestas(0);
    entity.setPartos(0);
```


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

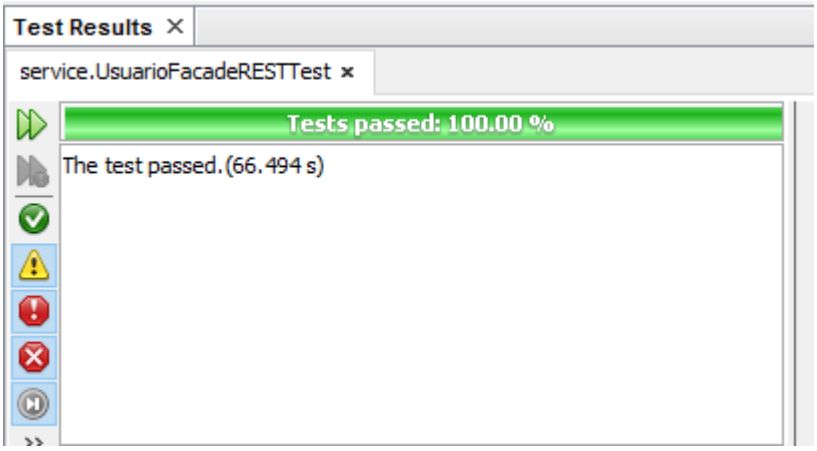
```
EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
UsuarioFacadeREST instance = (UsuarioFacadeREST)
        container.getContext().lookup("java:global/classes/UsuarioFacadeREST");
instance.create(entity);
container.close();
assertEquals(new Integer (3), entity.getIdusuario());
```



Posteriormente se muestra la prueba de eliminación del usuario

```
@Test
public void testRemove_GenericType() throws Exception {
    System.out.println("remove");
    Usuario entity = null;
    EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();

    UsuarioFacadeREST instance =
            (UsuarioFacadeREST)
            container.getContext().lookup("java:global/classes/UsuarioFacadeREST");
    entity= instance.find(3);
    instance.remove(entity);
    Usuario find = null;
    try{
        instance.find(3);
    }catch(Exception ex){
        //
    }
    assertNull(find);
    container.close();
```

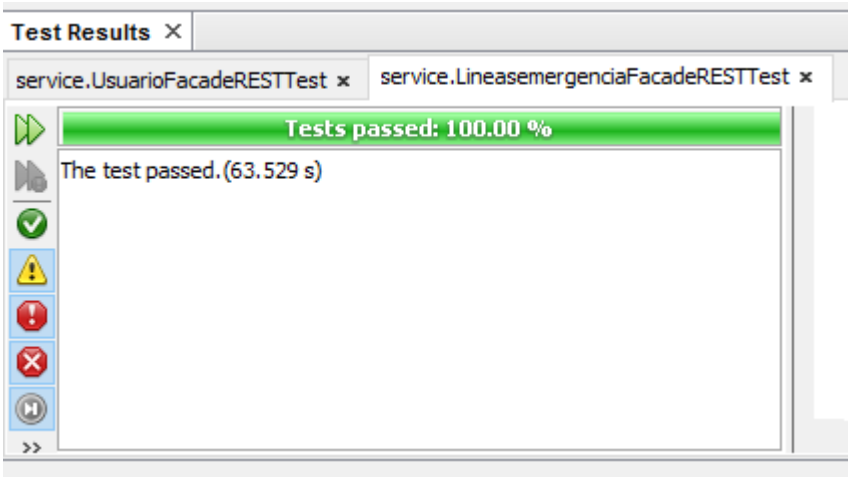

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.



5.3.6 Prueba 9

En la imagen a continuación se puede observar la prueba para encontrar las líneas de emergencia.

```
@Test
public void testFindAll() throws Exception {
    System.out.println("findAll");
    EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
    LineasemergenciaFacadeREST instance = (LineasemergenciaFacadeREST)container.getContext().lookup("java:
    List <Lineasemergencia> expResult;
    expResult = new ArrayList <>() ;
    expResult.add(new Lineasemergencia(1));
    expResult.add(new Lineasemergencia(2));
    expResult.add(new Lineasemergencia(3));
    List<Lineasemergencia> result = instance.findAll();
    for(int i=0; i< expResult.size(); i++)
        assertEquals(expResult.get(i).getIdlineasemergencia(), result.get(i).getIdlineasemergencia());
    container.close();
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```



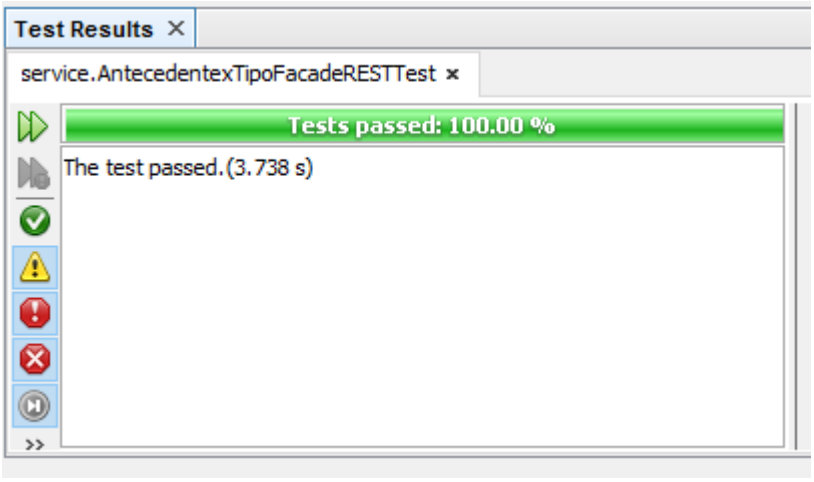
La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

5.3.7 Prueba 10, 12, 13

Las pruebas de antecedentes se encuentran en la parte inferior buscando los tipos determinados.

La prueba de la parte inferior muestra que la cantidad de antecedentes de tipo Congenito debe ser 4.

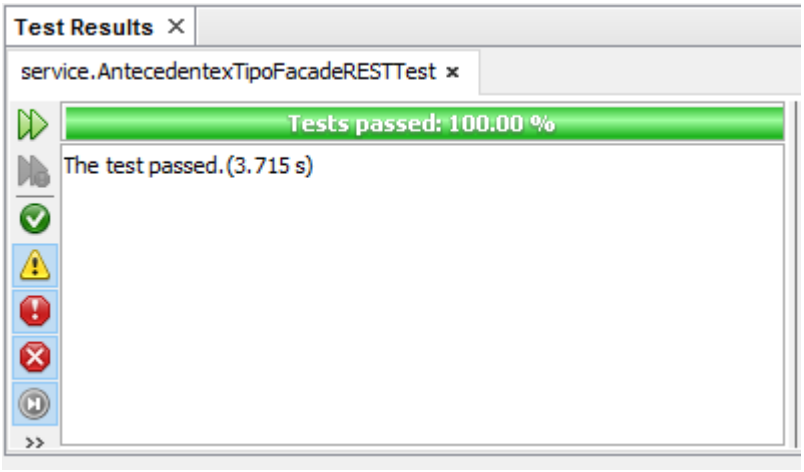
```
@Test
public void testFindByType() {
    System.out.println("findByType");
    String t = "CONGENITO";
    AntecedentexTipoFacadeREST instance = new AntecedentexTipoFacadeREST();
    Integer expResult = 4;
    List<Antecedente> result = instance.findByType(t);

    assertEquals(expResult, new Integer(result.size()));
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```



La prueba de la parte inferior muestra que la cantidad de antecedentes de tipo quirúrgico debe ser 3.

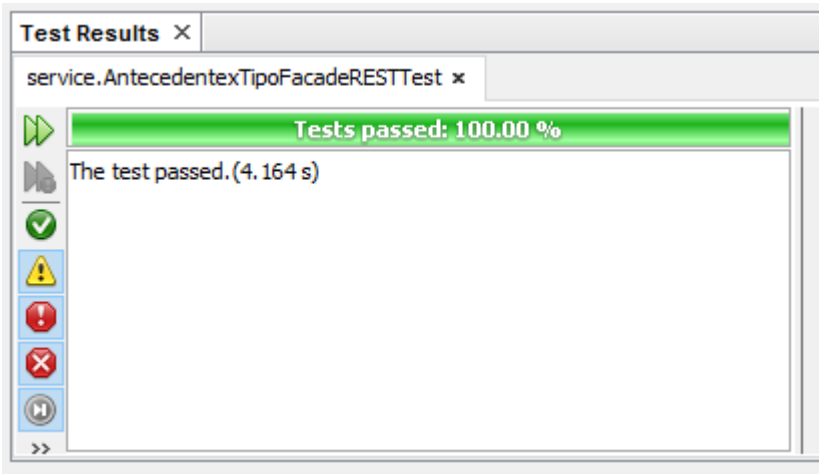
```
@Test
public void testFindByType() throws Exception {
    System.out.println("findByType");
    String t = "QUIRURGICO";
    AntecedentexTipoFacadeREST instance = new AntecedentexTipoFacadeREST();
    Integer expResult = 3;
    tipoAntecedenteRestClient l = new tipoAntecedenteRestClient();
    List<Antecedente> result =l.findByType_List(t);

    assertEquals(expResult, new Integer(result.size()));
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");

}
```


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.



La prueba de la parte inferior muestra que la cantidad de antecedentes de tipo trastorno mental debe ser 1.

```
@Test
public void testFindByType() throws Exception {
    System.out.println("findByType");
    String t = "TRASTORNO MENTAL";
    AntecedentexTipoFacadeREST instance = new AntecedentexTipoFacadeREST();
    Integer expResult = 1;
    tipoAntecedenteRestClient l = new tipoAntecedenteRestClient();
    List<Antecedente> result =l.findByType_List(t);

    assertEquals(expResult, new Integer(result.size()));
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");

}
```

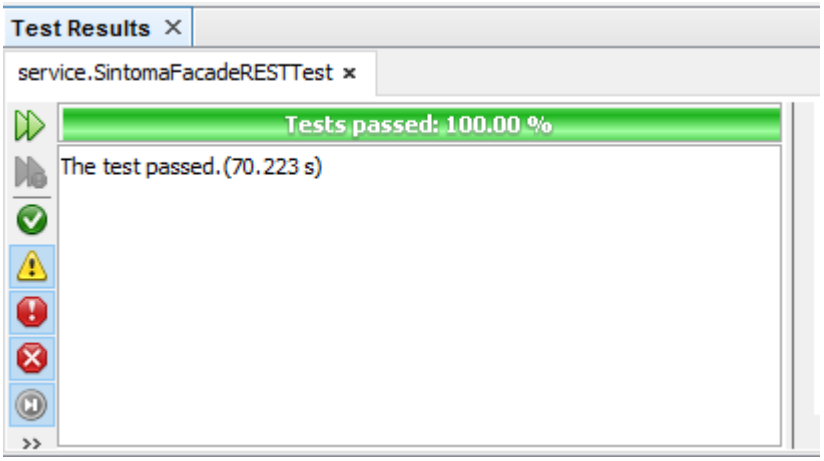


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

5.3.8 Prueba 14

La prueba a continuación obtiene todos los síntomas y los cuenta, con el fin de conocer si se obtuvieron todos.

```
@Test
public void testFindAll() throws Exception {
    System.out.println("findAll");
    EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
    SintomaFacadeREST instance = (SintomaFacadeREST)container.getContext().lookup("java:global/classes/SintomaFacad
    Integer expResult = 16;
    List<Sintoma> result = instance.findAll();
    assertEquals(expResult, new Integer(result.size()));
    container.close();
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

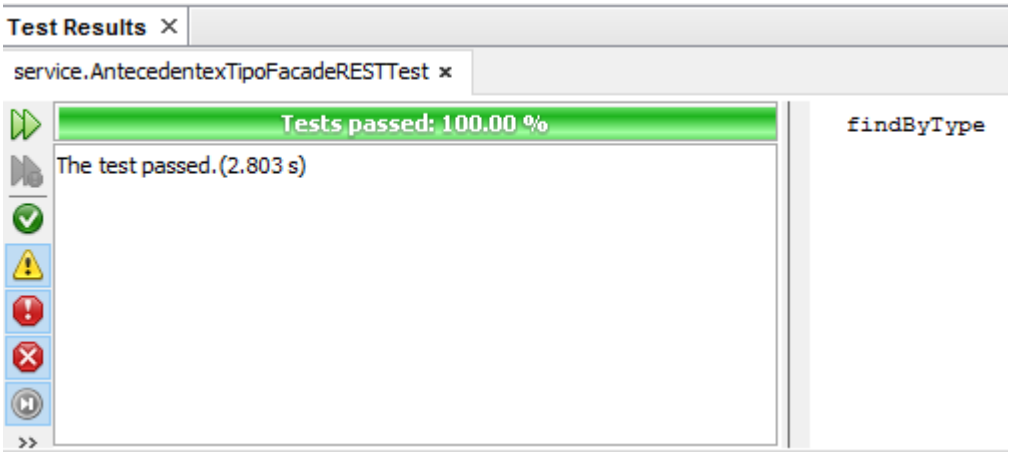


5.3.9 Prueba 15

Esta prueba consta de buscar un tipo de antecedente y luego asociarselo a un usuario en particular.

```
@Test
public void testFindByType() throws Exception {
    System.out.println("findByType");
    String t = "MORBIDOS";
    AntecedentexTipoFacadeREST instance = new AntecedentexTipoFacadeREST();
    Integer expResult = 2;
    tipoAntecedenteRestClient l = new tipoAntecedenteRestClient();
    List<Antecedente> result =l.findByType_List(t);

    assertEquals(expResult, new Integer(result.size()));
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");

}
```

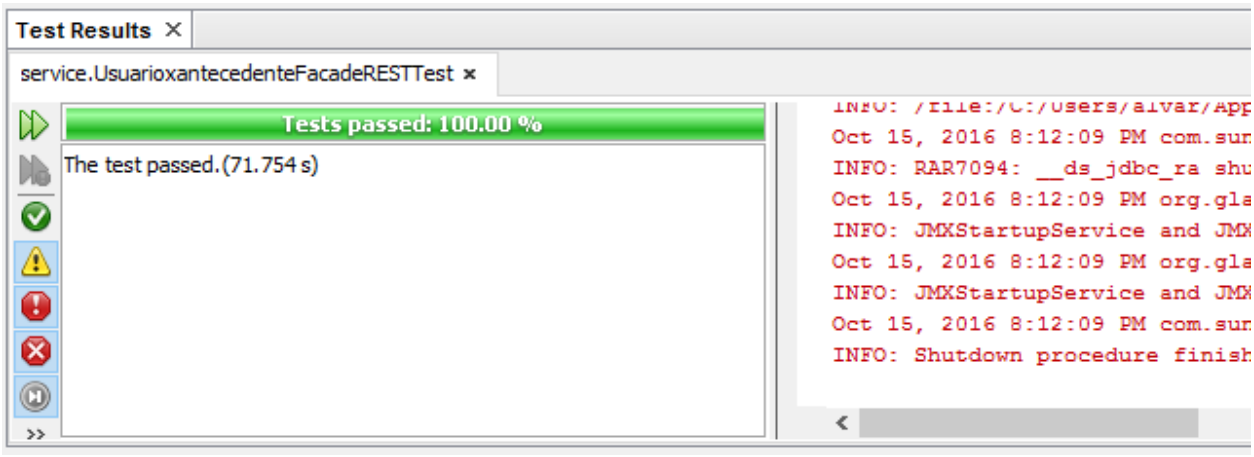

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.



Asociar el al usuario 2 el antecedente 9.

```
@Test
 public void testCreate() throws Exception {
     System.out.println("create");
     Usuarioxantecedente entity = new Usuarioxantecedente();
     UsuarioxantecedentePK upk = new UsuarioxantecedentePK();
     upk.setUsuarioIdusuario(2);
     upk.setAntecedenteIdantecedente(9);
     entity.setUsuarioxantecedentePK(upk);

     EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
     UsuarioFacadeREST instance2 =
             (UsuarioFacadeREST)
             container.getContext().lookup("java:global/classes/UsuarioFacadeREST");
     Usuario u = instance2.find(2);
     entity.setUsuario(u);
     System.out.println(u.toString());
     AntecedenteFacadeREST instance1 =
             (AntecedenteFacadeREST)
             container.getContext().lookup("java:global/classes/AntecedenteFacadeREST");
     Antecedente a =instance1.find(9);
     System.out.println(a.toString());
     entity.setAntecedente(a);
     UsuarioxantecedenteFacadeREST instance =

        (UsuarioxantecedenteFacadeREST)
        container.getContext().lookup("java:global/classes/UsuarioxantecedenteFacadeREST");
instance.create(entity);


container.close();
assertEquals(9, entity.getUsuarioxantecedentePK().getAntecedenteIdantecedente());
```

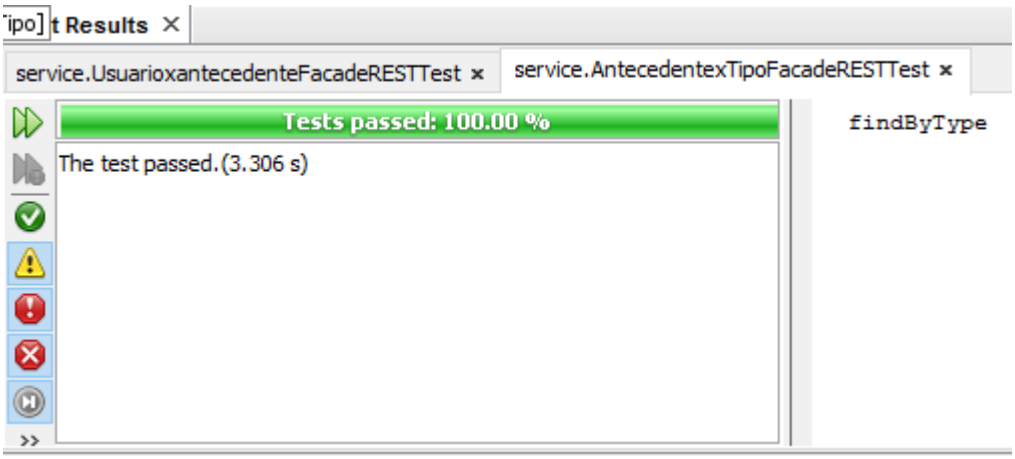

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.



5.3.10 Prueba 17

Buscar antecedente de tipo hábitos y estos debe ser 1.

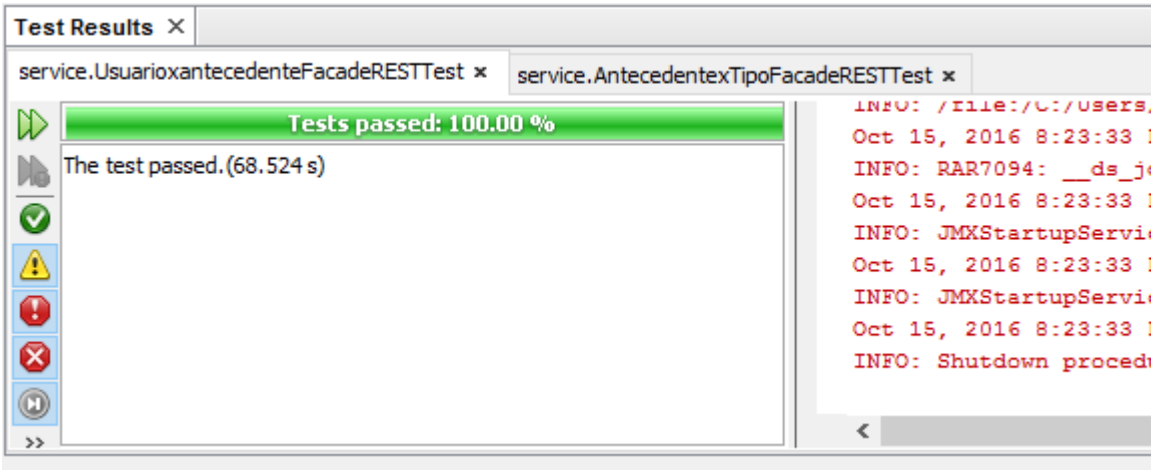
```
@Test
public void testFindByType() throws Exception {
    System.out.println("findByType");
    String t = "HABITOS";
    AntecedentexTipoFacadeREST instance = new AntecedentexTipoFacadeREST();
    Integer expResult = 1;
    tipoAntecedenteRestClient 1 = new tipoAntecedenteRestClient();
    List<Antecedente> result =1.findByType_List(t);

    assertEquals(expResult, new Integer(result.size()));
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");

}
```



Posterior a este asociar el antecedente a un usuario

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

```
@Test
public void testCreate() throws Exception {
    System.out.println("create");
    Usuarioxantecedente entity = new Usuarioxantecedente(2,11);

    EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
    UsuarioFacadeREST instance2 =
            (UsuarioFacadeREST)
            container.getContext().lookup("java:global/classes/UsuarioFacadeREST");
    entity.setUsuario(instance2.find(2));
    AntecedenteFacadeREST instance1 =
            (AntecedenteFacadeREST)
            container.getContext().lookup("java:global/classes/AntecedenteFacadeREST");
    entity.setAntecedente(instance1.find(11));
    UsuarioxantecedenteFacadeREST instance =
            (UsuarioxantecedenteFacadeREST)
            container.getContext().lookup("java:global/classes/UsuarioxantecedenteFacadeREST");
    instance.create(entity);
    container.close();
    assertEquals(11, entity.getUsuarioxantecedentePK().getAntecedenteIdantecedente());
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

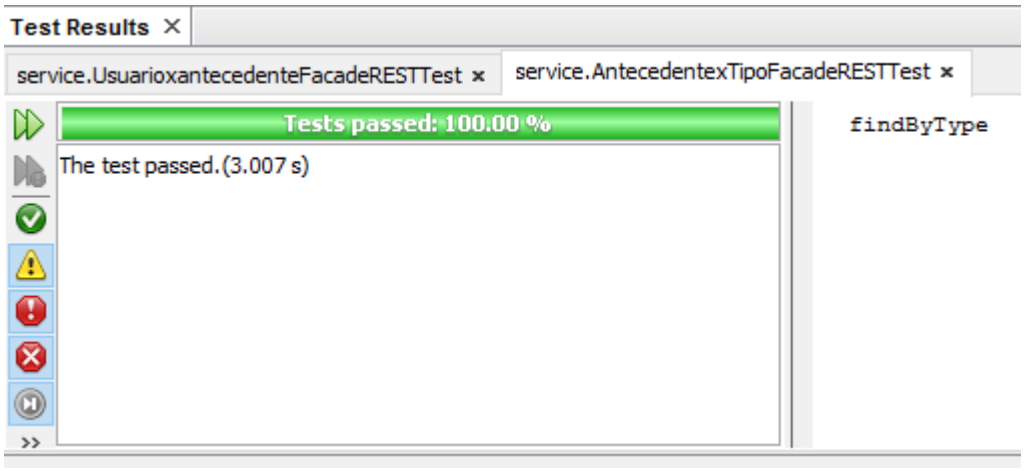


5.3.11 Prueba 18

Busca los antecedentes de tipo medicamentos, de los cuales en el sistema hay 2.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

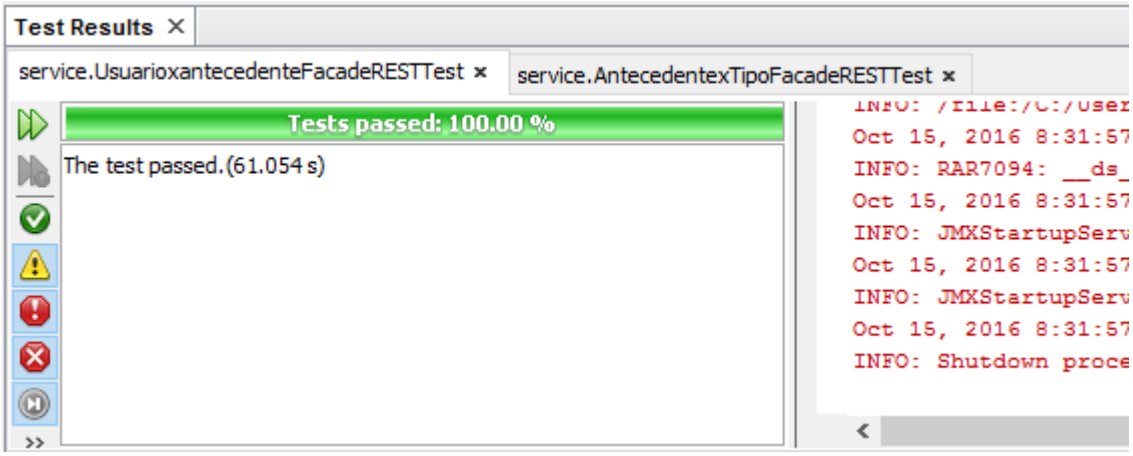
```
@Test
public void testFindByType() throws Exception {
    System.out.println("findByType");
    String t = "MEDICAMENTOS";
    AntecedentexTipoFacadeREST instance = new AntecedentexTipoFacadeREST();
    Integer expResult = 2;
    tipoAntecedenteRestClient 1 = new tipoAntecedenteRestClient();
    List<Antecedente> result =1.findByType_List(t);

    assertEquals(expResult, new Integer(result.size()));
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");

}
```



Posteriormente asocia un antecedente determinado a un usuario.

```
@Test
public void testCreate() throws Exception {
    System.out.println("create");
    Usuarioxantecedente entity = new Usuarioxantecedente(2,13);

    EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
    UsuarioFacadeREST instance2 =
            (UsuarioFacadeREST)
            container.getContext().lookup("java:global/classes/UsuarioFacadeREST");
    entity.setUsuario(instance2.find(2));
    AntecedenteFacadeREST instance1 =
            (AntecedenteFacadeREST)
            container.getContext().lookup("java:global/classes/AntecedenteFacadeREST");
    entity.setAntecedente(instance1.find(13));
    UsuarioxantecedenteFacadeREST instance =
            (UsuarioxantecedenteFacadeREST)
            container.getContext().lookup("java:global/classes/UsuarioxantecedenteFacadeREST");
    instance.create(entity);
    container.close();
    assertEquals(13, entity.getUsuarioxantecedentePK().getAntecedenteIdantecedente());
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

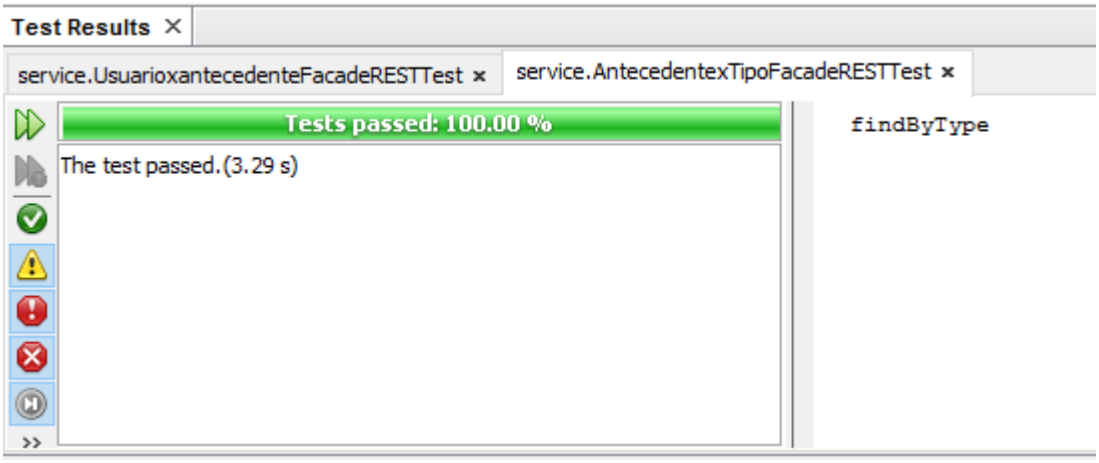

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.



5.3.12 Prueba 19

Se realiza una prueba para saber cuántos antecedentes de tipo Alergia hay, que debe concordar con el número de estos presentes en la base de datos.

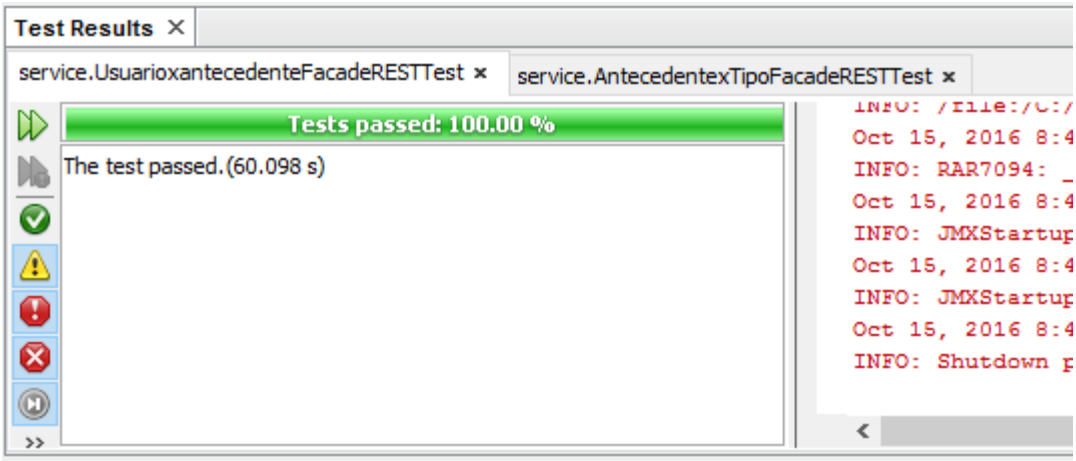
```
@Test
public void testFindByType() throws Exception {
    System.out.println("findByType");
    String t = "ALERGIAS";
    AntecedentexTipoFacadeREST instance = new AntecedentexTipoFacadeREST();
    Integer expResult = 2;
    tipoAntecedenteRestClient 1 = new tipoAntecedenteRestClient();
    List<Antecedente> result =1.findByType_List(t);

    assertEquals(expResult, new Integer(result.size()));
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");

}
```



Asociar un antecedente a un usuario

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

```
@Test
 public void testCreate() throws Exception {
     System.out.println("create");
     Usuarioxantecedente entity = new Usuarioxantecedente(2,14);

     EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
     UsuarioFacadeREST instance2 =
             (UsuarioFacadeREST)
             container.getContext().lookup("java:global/classes/UsuarioFacadeREST");
     entity.setUsuario(instance2.find(2));
     AntecedenteFacadeREST instance1 =
             (AntecedenteFacadeREST)
             container.getContext().lookup("java:global/classes/AntecedenteFacadeREST");
     entity.setAntecedente(instance1.find(14));
     UsuarioxantecedenteFacadeREST instance =
             (UsuarioxantecedenteFacadeREST)
             container.getContext().lookup("java:global/classes/UsuarioxantecedenteFacadeREST");
     instance.create(entity);
     container.close();
     assertEquals(14, entity.getUsuarioxantecedentePK().getAntecedenteIdantecedente());
     // TODO review the generated test code and remove the default call to fail.
     //fail("The test case is a prototype.");
 }
```

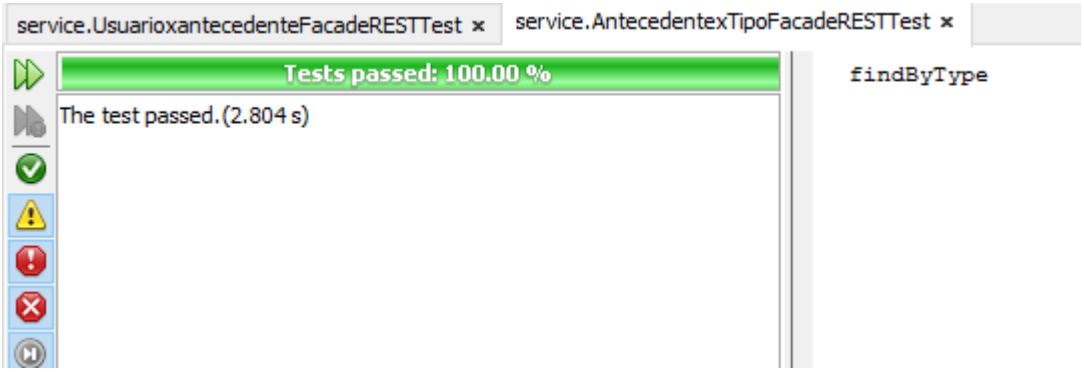


5.3.13 Prueba 20

Obtener antecedentes de tipo familiar, la base de datos no posee de este tipo de antecedentes.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

```
@Test
public void testFindByType() throws Exception {
    System.out.println("findByType");
    String t = "FAMILIAR";
    AntecedentexTipoFacadeREST instance = new AntecedentexTipoFacadeREST();
    Integer expResult = 0;
    tipoAntecedenteRestClient l = new tipoAntecedenteRestClient();
    List<Antecedente> result=null;
    try {
        l.findByType_List(t);
    } catch (Exception e) {
        //
    }
    assertNull(result);
    // TODO review the generated test code and remove the default call to fai
    //fail("The test case is a prototype.");

}
```

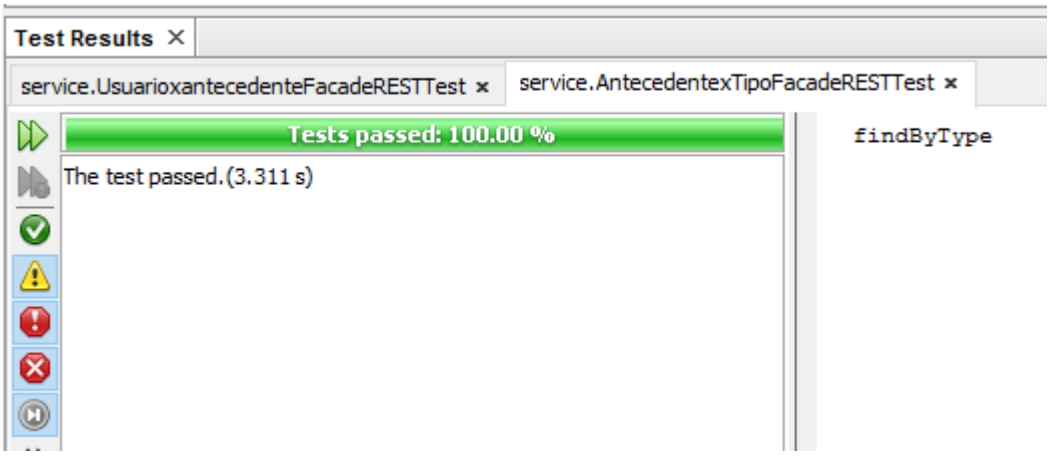


5.3.14 Prueba 21

Esta prueba busca los antecedentes de tipo inmunizaciones y le asocia al usuario 1 un antecedente determinado.

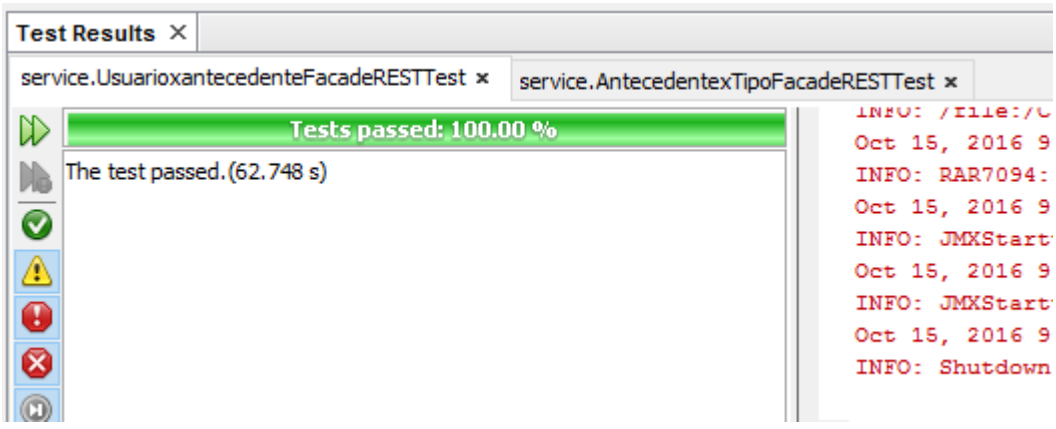
```
@Test
public void testFindByType() throws Exception {
    System.out.println("findByType");
    String t = "INMUNIZACION";
    AntecedentexTipoFacadeREST instance = new AntecedentexTipoFacadeREST();
    Integer expResult = 1;
    tipoAntecedenteRestClient l = new tipoAntecedenteRestClient();
    List<Antecedente> result= l.findByType_List(t);

    assertEquals(expResult, new Integer (result.size()));
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.



Asociarlo a un usuario

```
@Test
public void testCreate() throws Exception {
    System.out.println("create");
    Usuarioxantecedente entity = new Usuarioxantecedente(1,15);

    EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
    UsuarioFacadeREST instance2 =
            (UsuarioFacadeREST)
            container.getContext().lookup("java:global/classes/UsuarioFacadeREST");
    entity.setUsuario(instance2.find(1));
    AntecedenteFacadeREST instance1 =
            (AntecedenteFacadeREST)
            container.getContext().lookup("java:global/classes/AntecedenteFacadeREST");
    entity.setAntecedente(instance1.find(15));
    UsuarioxantecedenteFacadeREST instance =
            (UsuarioxantecedenteFacadeREST)
            container.getContext().lookup("java:global/classes/UsuarioxantecedenteFacadeREST");
    instance.create(entity);
    container.close();
    assertEquals(15, entity.getUsuarioxantecedentePK().getAntecedenteIdantecedente());
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
```



La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

5.4 CONCLUSIONES

- Las pruebas se realizaron de manera exitosa.
- Si la fachada no se relacionaba directamente con un EJB fue requerido un cliente como proxy para acceder a sus servicios y con ello pobras las diferentes funcionalidades.
- Se requirió el desarrollo de una fachada que anteriormente no existía, llamada AntecedentesxTipo, la cual se dedica a buscar los antecedentes dependiendo de un tipo en específico. /WSTappi/webresources/AntecedentexTipo
- Adicionalmente se utilizó un container standalone en el cual los EJB fueron probados, por lo tanto este demuestra cómo crear una instancia embebida de un contenedor en una clase de JUnit y llamar al session bean de la lógica de negocio. De esta manera no es necesario ejecutar las pruebas en el servidor Java EE sino en el contenedor local.

6 PRUEBAS FUNCIONALES (FUNCTIONAL)

Es un tipo de prueba de caja negra, cuyo sinónimo es *pruebas basadas en especificación*, este tipo de pruebas se basan en entradas y salidas basadas en la especificación del software y no directamente en el código o el software ejecutable [4], como son los casos de uso.

6.1 HERRAMIENTA

La herramienta para este tipo de prueba es JUnit, es un framework de código abierto que sirve para escribir y correr pruebas de manera repetible. Es una instancia de la arquitectura xUnit, este incluye aserciones para probar resultados esperados, pruebas fijas para compartir datos de prueba y ejecutores de pruebas [9].

6.2 ELEMENTOS A SER PROBADOS / NO PROBADOS

En la tabla a continuación se muestran los casos de uso involucrados en las funcionalidades del servidor que se debían cumplir. Adicionalmente se tuvo en cuenta la priorización de casos de uso realizada en el SRS para evaluar que pruebas se consideraban más importantes. La persona responsable de llevar a cabo las pruebas de servidor es Luisa Álvarez Valencia.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

Id prueba	Prueba	Funcionalidad – Caso de uso	Característica Prueba	
			Entrada	Salida
1	GET, LoginFacadeREST. A partir de un id que se ingresa por la URL busca al usuario.	Iniciar Sesión	Ver prueba 1, 2 de unit testing	
2	GET, ListaHistoriaClinicaXUsuarioFacadeREST. A partir de un identificador un usuario, esta fachada devuelve las historias clínicas que tiene ese paciente.	Ver historial de historias clínicas	Identificador de idusuario=1	Deben salir una lista de historiasclinicas con idhistoria= 1,2,4,5
3	GET, HistoriaclinicaFacadeREST Dado un id se obtiene la historia clínica en XML.	Seleccionar HC por ID	Ingresar idhistoria =1	Se obtiene la historia clinica con la siguiente información (IDHISTORIACLINICA, DESCRIPCIONMOTIVOURGENCIA, INICIOMOTIVOURGENCIA, USUARIO_IDUSUARIO, FECHA, NIVELTRIAGE) VALUES (1, 'NO PUEDO RESPIRAR', '2016-10-11', 1, '2016-10-11', DEFAULT)
4	POST, HistoriaclinicaFacadeREST. A partir de la URL se realiza un post a esta entidad	Crear Historia clínica	Ver prueba 4 de Unit testing	
5	PUT, HistoriaclinicaFacadeREST. Dado un id de HC se identifica la HC a la cual se le va a realizar el cambio, es necesario para la modificación enviar todos los datos no solo el id.	Modificar historia clínica (HC)	Dado idhistoria =7 editar DESCRIPCIONMOTIVOURGENCIA = DOLOR PECTORAL FUERTE	Los cambios de edición se observan en la base de datos
6	Dado un id de HC se identifican los síntomas que esta tiene.	Listar síntomas presentes en la historia clínica elegida	Dado idhistoria = 1	Lista de sintomas listados con idsintoma = 1,2,3,4
7	PUT, SintomasxhistoriaFacadeREST. Utiliza una llave primaria compuesta y por lo tanto los campos que permiten edición son presencia y fecha.	Elegir síntoma de historia clínica para edición	Dado idhistoria = 7 Editar para cambiar el idsintoma = 1, presencia = 'NO'	Se reflejan los cambios en la base de datos
8	DELETE, SintomasxhistoriaFacadeREST. Utiliza una llave compuesta para lograr borrar los datos de esa tabla	Elegir síntoma de historia clínica para borrar	Dado idhistoria = 7 Borrar idsintoma = 1	Se reflejan los cambios en la base de datos y el síntoma ya no está asociado
9.1	El usuario ingresa los síntomas de su dolencia y la lógica de negocio debe dar un nivel de prioridad tentativo (Triage).	Ingresar Síntomas	Dado ingreso en idhistoria = 1, de idsintomas = 1,2,3,4 todos con presencia = 'SI'	El cálculo de Triage debe ser = 1

9.2			Dado ingreso en idhistoria = 2, de idsintomas = 5,6,7,8 todos con presencia = 'SI'	El cálculo de Triage debe ser = 2
9.3			Dado ingreso en idhistoria = 3, de idsintomas = 10 con presencia = 'SI'	El cálculo de Triage debe ser = 3
9.4			Dado ingreso en idhistoria = 4, de idsintomas = 11,12 todos con presencia = 'SI'	El cálculo de Triage debe ser = 4
9.5			Dado ingreso en idhistoria = 5, de idsintomas = 14,15,16 todos con presencia = 'SI' y idsintoma 12 con presencia = 'NO'	El cálculo de Triage debe ser = 5
9.6			Dado ingreso en idhistoria = 5, de idsintomas = 16 con presencia = 'SI' y idsintoma = 12 con presencia = 'NO'	El cálculo de Triage debe ser = 0 que no existe
10	GET, SintomasFacadeREST, es una fachada de la entidad que contiene todos los síntomas. Trae todos los síntomas que el usuario pueda presentar.		Ver prueba 14 Unit testing	
11	Se ingresan todos los síntomas en el cliente, este envía una lista completa de los síntomas y la lógica de negocio la recibe, en la tabla de SintomasXHistoria. Dichos síntomas se asocian a una determinada historia clínica de modo que el id de la historia sirve como llave primaria.		Ver prueba 4 unit testing	
12	El cálculo del Triage hace parte de la lógica de negocio. Este utiliza los síntomas ingresados por el usuario (encontrados en SintomasxHistoria), se calcula el nivel de triage, lo retorna y lo almacena en la entidad de historia clínica. De ese modo se tiene el nivel de triage tentativo por historia clínica.		Ver prueba 9	
13	PUT, UsuarioFacadeREST	Editar Usuario	Ver prueba 3 Unit Testing	

La información contenida se considera pública (a menos que explícitamente se indique de otro modo). Utiliza una licencia Creative Commons.

	Edita un usuario dando como entrada los datos de un usuario en XML.			
14	POST, UsuarioFacadeREST Crea un usuario dando como entrada los datos de un usuario en XML.	Crear Usuario Ingresar datos ginecológicos	Ve prueba 6 Unit testing que crea un usuario	
15	DELETE, UsuarioFacadeREST Elimina un usuario dado un id como entrada los datos de un usuario en XML.	Eliminar Usuario	Ver prueba 6 Unit testing que crea y elimina un usuario	
16	ListarantecedentexusuarioFacadeREST. Realiza un find por una query con id de usuario.	Listar antecedentes de usuario	Dado idusuario = 1	Lista de antecedentes con idantecedente = 1, 2, 5, 15
17	UsuarioxantecedenteFacadeREST. Este utiliza una llave primaria compuesta de: idantecedente y idusuario.	Eliminar antecedentes de usuario	Dado idusuario = 1 Idantecedente = 15	Borrar el antecedente de dicho usuario
18	UsuarioxantecedenteFacadeREST. Este utiliza una llave primaria compuesta de: idantecedente y idusuario.	Modificar antecedentes de usuario	Modificar la variable auxiliar del antecedente 1.	Permite modificar algún antecedente asociado a un usuario
19	GET, AntecedentesFacadeREST, es una fachada de la entidad que contiene todos los antecedentes. Trae los antecedentes para que el usuario pueda elegir de estos cuales asociar a su perfil.	Ingresar antecedentes familiares	Se obtienen una lista de antecedentes que tiene el software	
20	GET, LineasemergenciaFacadeREST. Permite obtener todas las líneas de emergencia que posea la tabla insertada.	Contactar con líneas de emergencia	Ver prueba 9 de Unit testing	
21	Este utiliza diferentes métodos dictados en la parte superior, como es eliminar un usuario.	Administrar cuentas	Ver prueba 6 Unit testing que crea y elimina un usuario	

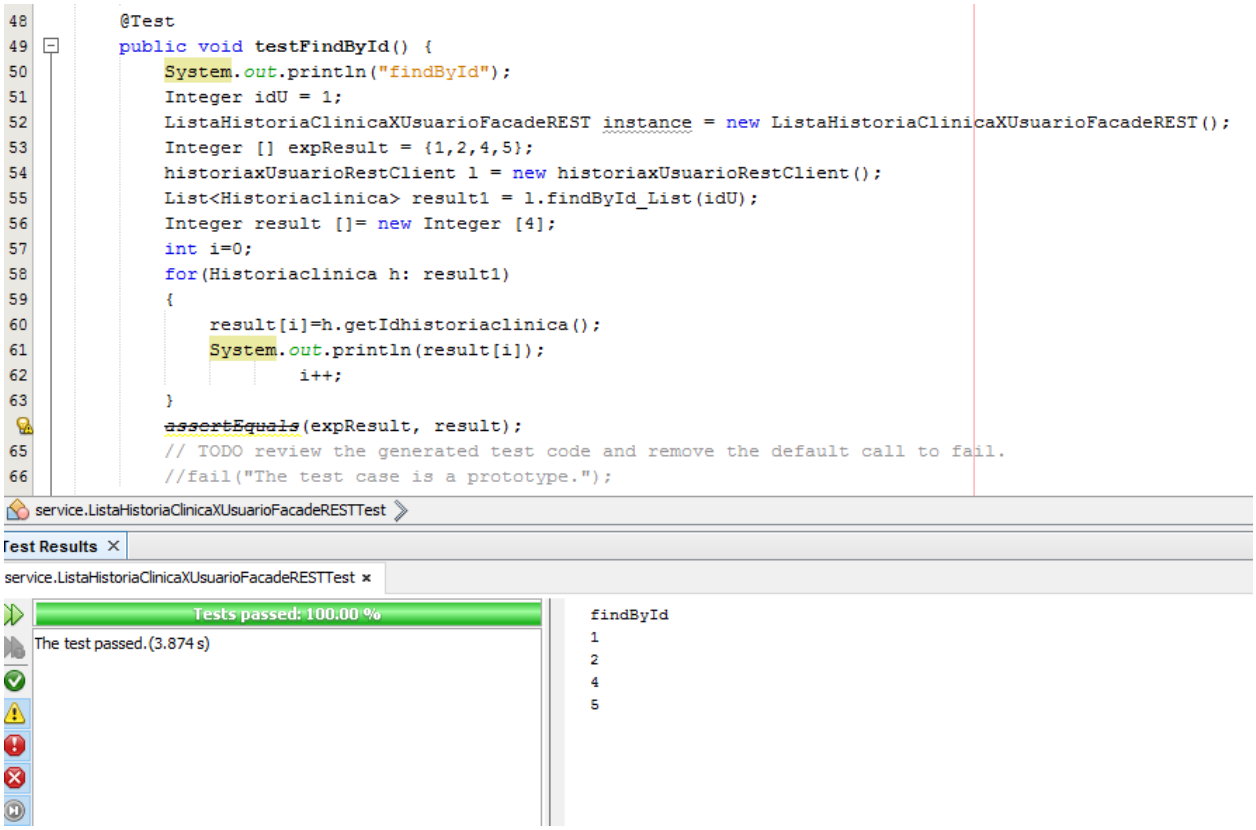
Tabla 12 casos de uso servidor

La información contenida se considera pública (a menos que explícitamente se indique de otro modo). Utiliza una licencia Creative Commons.

6.3 RESULTADO PRUEBAS

6.3.1 Prueba 2

La prueba consta de encontrar las historias clínicas asociadas a un usuario.



```
48      @Test
49      public void testFindById() {
50          System.out.println("findById");
51          Integer idU = 1;
52          ListaHistoriaClinicaXUsuarioFacadeREST instance = new ListaHistoriaClinicaXUsuarioFacadeREST();
53          Integer [] expResult = {1,2,4,5};
54          historiaxUsuarioRestClient l = new historiaxUsuarioRestClient();
55          List<Historiaclinica> result1 = l.findById_List(idU);
56          Integer result []= new Integer [4];
57          int i=0;
58          for(Historiaclinica h: result1)
59          {
60              result[i]=h.getIdhistoriaclinica();
61              System.out.println(result[i]);
62                      i++;
63          }
            assertEquals(expResult, result);
65          // TODO review the generated test code and remove the default call to fail.
66          //fail("The test case is a prototype.");
```

service.ListaHistoriaClinicaXUsuarioFacadeRESTTest

Test Results

service.ListaHistoriaClinicaXUsuarioFacadeRESTTest

Tests passed: 100.00 %

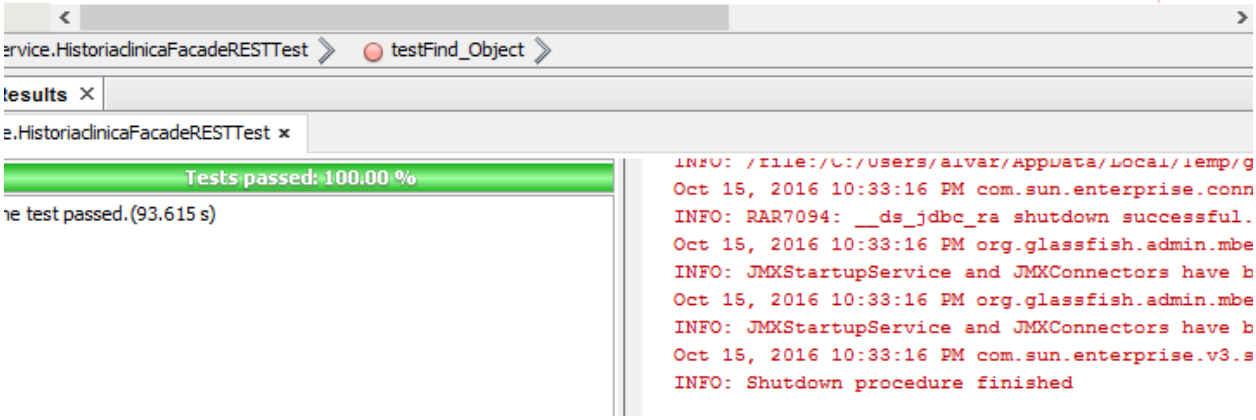
The test passed. (3.874 s)

```
findById
1
2
4
5
```

6.3.2 Prueba 3

Esta prueba consta de encontrar la historia clínica con id =1.

```
    @Test
    public void testFind_Object() throws Exception {
        System.out.println("find");
        Object id = 1;
        EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer()
        HistoriaclinicaFacadeREST instance =
                (HistoriaclinicaFacadeREST)
                container.getContext().lookup("java:global/classes/HistoriaclinicaFacad
        Integer expResult = 1;
        Historiaclinica result = instance.find(id);
        assertEquals(expResult, result.getIdhistoriaclinica());
        container.close();
        // TODO review the generated test code and remove the default call to fail.
        //fail("The test case is a prototype.");
    }

    /**
     * Test of findRange method, of class HistoriaclinicaFacadeREST.
     */
//    @Test
//    public void testFindRange_intArr() throws Exception {
//        System.out.println("findRange");
//        int[] range = null;
```



6.3.3 Prueba 5

Esta prueba consta de editar la historia clínica id =7 para cambiar su motivo de urgencia.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

```
44      /**
45       * Test of edit method, of class HistoriaclinicaFacadeREST.
46       */
47      @Test
        public void testEdit_GenericType() throws Exception {
49          System.out.println("edit");
50          Historiaclinica entity = null;
            EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
52          HistoriaclinicaFacadeREST instance =
53                  (HistoriaclinicaFacadeREST)
54                  container.getContext().lookup("java:global/classes/HistoriaclinicaFacadeREST");
55          entity = instance.find(7);
56          entity.setDescripcionmotivourgencia("DOLOR PECTORAL FUERTE");
57          instance.edit(entity);
58          assertEquals("DOLOR PECTORAL FUERTE", instance.find(7).getDescripcionmotivourgencia());
59          container.close();
60
61          // TODO review the generated test code and remove the default call to fail.
62          // fail("The test case is a prototype.");
63      }
```

service.HistoriaclinicaFacadeRESTTest > testEdit_GenericType >

est Results ×

ervice.HistoriaclinicaFacadeRESTTest ×

Tests passed: 100.00 %

The test passed. (68.347 s)

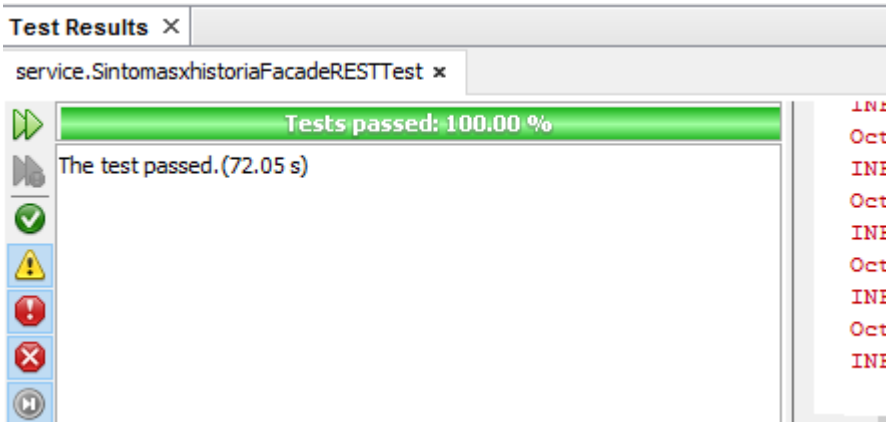
```
INFO: file:/C:/Users/alvar/AppData/Local/Temp/grembed8909
Oct 15, 2016 11:05:13 PM com.sun.enterprise.connectors.serv
INFO: RAR7094: __ds_jdbc_ra shutdown successful.
Oct 15, 2016 11:05:13 PM org.glassfish.admin.mbeanserver.JM
INFO: JMXStartupService and JMXConnectors have been shut do
Oct 15, 2016 11:05:13 PM org.glassfish.admin.mbeanserver.JM
INFO: JMXStartupService and JMXConnectors have been shut do
Oct 15, 2016 11:05:13 PM com.sun.enterprise.v3.server.AppSe
INFO: Shutdown procedure finished
```

6.3.4 Prueba 6

Esta prueba se realiza para concordar la cantidad de síntomas que tiene una historia clínica determinada.

```
@Test
public void testFind_Object() throws Exception {
    System.out.println("find");
    Object id = 1;
    EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
    SintomasxhistoriaFacadeREST instance =
            (SintomasxhistoriaFacadeREST)
            container.getContext().lookup("java:global/classes/SintomasxhistoriaFacadeREST");
    Integer expResult []= {1,2,3,4};
    List <Sintomasxhistoria> result = instance.findAll();
    Integer [] idSin= new Integer [4];
    int i=0;
    for(Sintomasxhistoria s:result)
    {
        if(s.getHistoriaclinica().getIdhistoriaclinica()==1)
        {

            idSin[i]=s.getSintoma().getIdsintoma();
            i++;
        }
    }
```

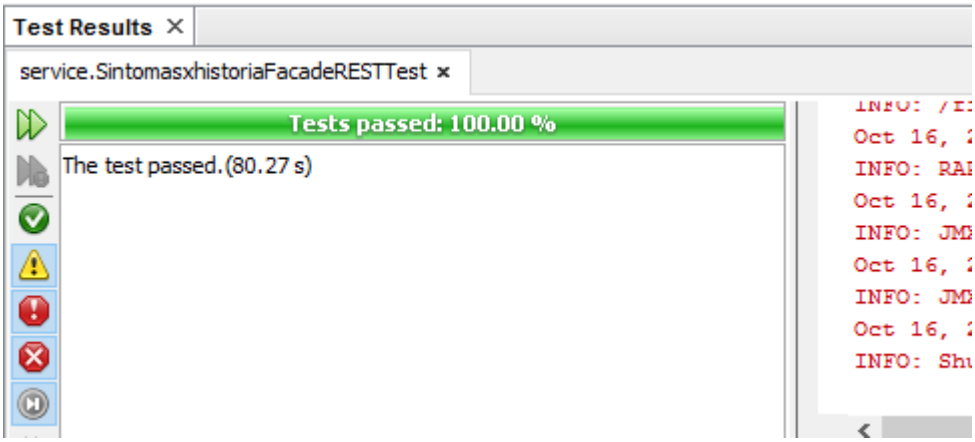

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.



6.3.5 Prueba 7

Esta prueba tiene como fin mostrar la edición de un síntoma que pertenece a una historia clínica.

```
@Test
public void testEdit_GenericType() throws Exception {
    System.out.println("edit");
    Sintomasxhistoria entity = null;
    EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
    SintomasxhistoriaFacadeREST instance =
            (SintomasxhistoriaFacadeREST)
            container.getContext().lookup("java:global/classes/SintomasxhistoriaFacadeREST");
    for(Sintomasxhistoria sh: instance.findAll())
        if(sh.getHistoriaclinica().getIdhistoriaclinica()== 7 && sh.getSintoma().getIdsintoma()== 1)
            entity =sh;
    entity.setPresencia("NO");
    instance.edit(entity);
    container.close();

    assertEquals(new String ("NO"), entity.getPresencia());
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

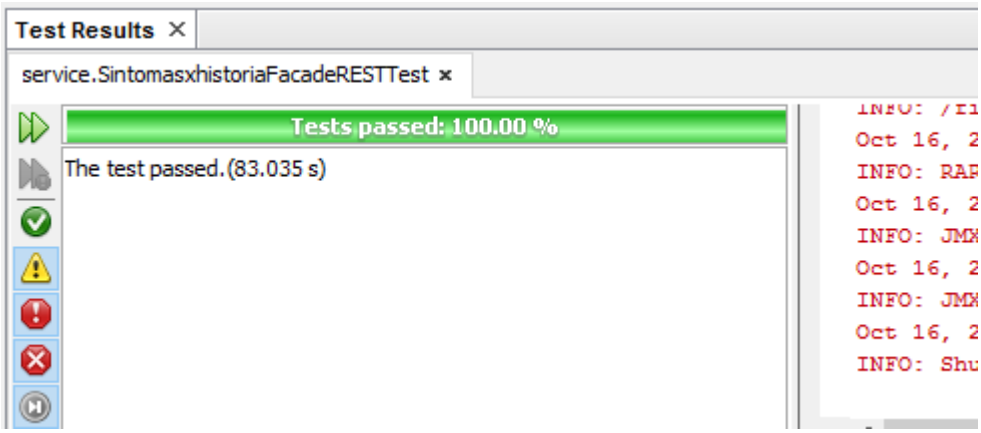


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

6.3.6 Prueba 8

Esta prueba se encarga de mostrar que los síntomas asociados a una historia clínica pueden ser borrados.

```
@Test
public void testRemove_GenericType() throws Exception {
    System.out.println("remove");
    Sintomasxhistoria entity;
    EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
    SintomasxhistoriaFacadeREST instance =
            (SintomasxhistoriaFacadeREST)
            container.getContext().lookup("java:global/classes/SintomasxhistoriaFacadeREST");
    entity = instance.find(new SintomasxhistoriaPK(7,1));
    instance.remove(entity);
    container.close();
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

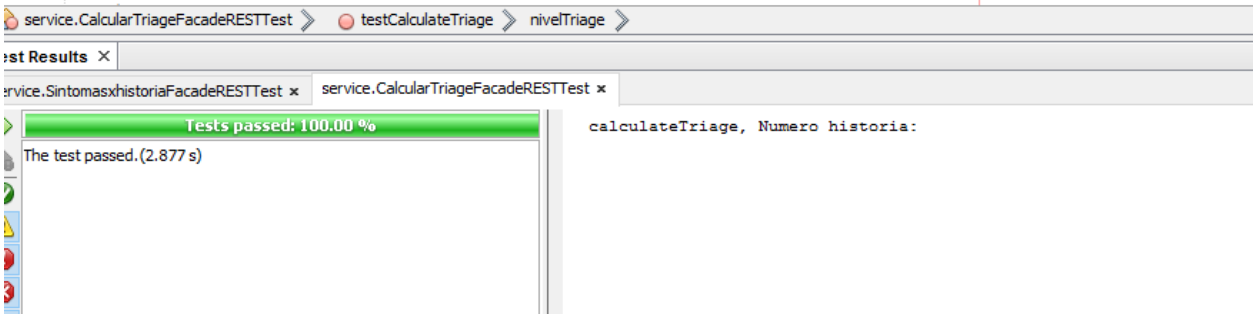


6.3.7 Prueba 9.1

Esta prueba tiene como fin ejemplificar que dado las entradas asignadas en el modelo de persistencia y la lógica de negocio se calcula el Triage correcto.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

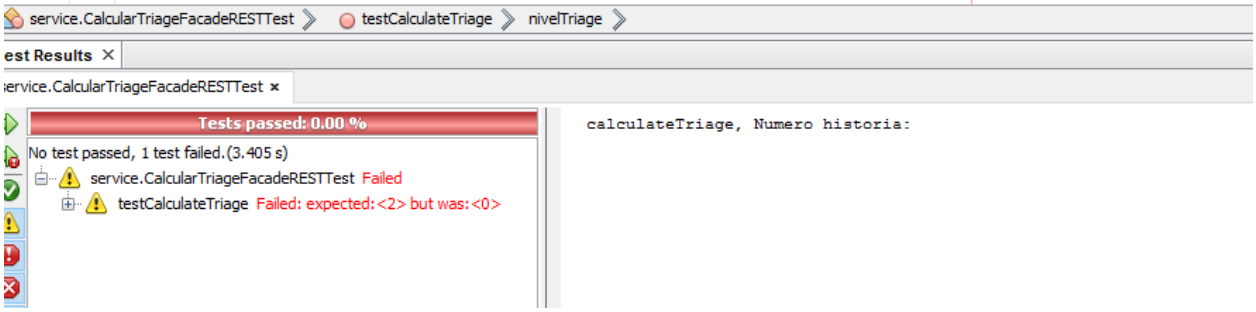
```
@Test
public void testCalculateTriage() {
    System.out.println("calculateTriage, Numero historia: ");
    // poner numero historia clinica
    Integer idhistoriaclinica = 1;
    Integer nivelTriage = 1;
    calcularTriageRestClient instance = new calcularTriageRestClient();
    Historiaclinica result = instance.calculateTriage_XML(Historiaclinica.class,idhistoriaclinica);
    assertEquals(nivelTriage, result.getNiveltriage());
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");

}
```



6.3.8 Prueba 9.2

Se llevó a cabo una prueba que realizaba el cálculo de un Triage 2 a partir de unos determinados síntomas.

```
@Test
public void testCalculateTriage() {
    System.out.println("calculateTriage, Numero historia: ");
    // poner numero historia clinica
    Integer idhistoriaclinica = 2;
    Integer nivelTriage = 2;
    calcularTriageRestClient instance = new calcularTriageRestClient();
    Historiaclinica result = instance.calculateTriage_XML(Historiaclinica.class,idhistoriaclinica);
    assertEquals(nivelTriage, result.getNiveltriage());
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.")
}

/**
```



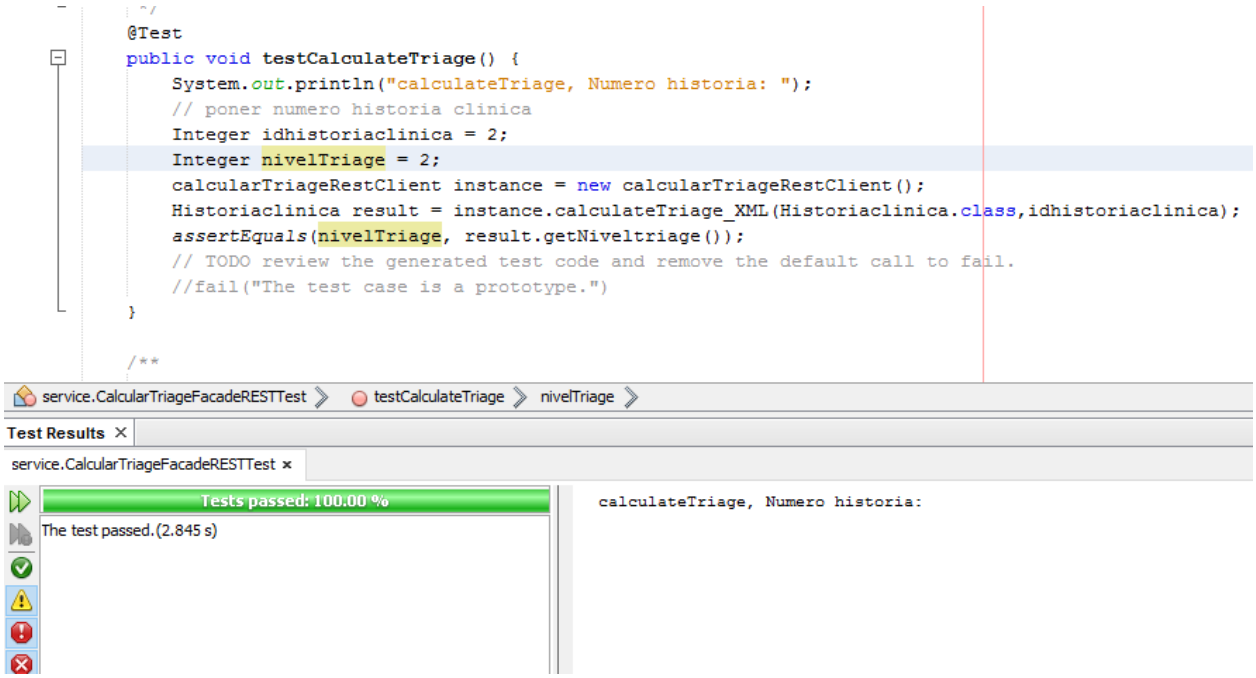
Como se puede observar dicha prueba salió fallida, por lo tanto se propuso revisar la lógica del Triage para corregir el error. Para ello se tuvo en cuenta TG-Diseno_implementacion_triageV0.4. Se verificó en cada una de las vistas relacionadas que su resultado fuera el correcto.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

Se encontró un error y se verificó dicha implementación que se muestra a continuación el cambio:

```
create view Matching_Triage_with_clinic_history_by_presence as
select T.IDHISTORY, T.IDTRIAGE, T.Symptoms_Matching_By_Presence
from
(
select C.IDHISTORY, C.IDTRIAGE, count(*) as Symptoms_Matching_By_Presence
from Matching_Triage_with_clinic_history_by_number_of_symptoms C,
APP.NIVELURGENCIA NU, APP.SINTOMASXHISTORIA SH
where C.IDTRIAGE = NU.TRIAGE_IDTRIAGE
and SH.SINTOMA_IDSINTOMA = NU.SINTOMA_IDSINTOMA
and SH.HISTORIACLINICA_IDHC = C.IDHISTORY
and NU.PRESENCIA = SH.PRESENCIA
group by C.IDHISTORY, C.IDTRIAGE
) T, NUMBER_SYMPTOMS_BY_CLINICHISTORY SBHC
where  SBHC.IDHISTORY = T.IDHISTORY
And SBHC.NUMBER_OF_SYMPTOMS = T.Symptoms_Matching_By_Presence;
```

Y despues de la corrección pertinente la prueba fue exitosa:



6.3.9 Prueba 9.3

Se llevó a cabo una prueba que realizaba el cálculo de un Triage 3 a partir de unos determinados síntomas.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

```
    */
    @Test
    public void testCalculateTriage() {
        System.out.println("calculateTriage, Numero historia: ");
        // poner numero historia clinica
        Integer idhistoriaclinica = 3;
        Integer nivelTriage = 3;
        calcularTriageRestClient instance = new calcularTriageRestClient();
        Historiaclinica result = instance.calculateTriage_XML(Historiaclinica.class,idhistoriaclinica);
        assertEquals(nivelTriage, result.getNiveltriage());
        // TODO review the generated test code and remove the default call to fail.
        //fail("The test case is a prototype.")
    }

    /**
```

service.CalcularTriageFacadeRESTTest > testCalculateTriage

Test Results

service.CalcularTriageFacadeRESTTest

Tests passed: 100.00 %

The test passed. (4.456 s)

```
calculateTriage, Numero historia:
```

6.3.10 Prueba 9.4

Se llevó a cabo una prueba que realizaba el cálculo de un Triage 2 a partir de unos determinados síntomas.

```
    */
    @Test
    public void testCalculateTriage() {
        System.out.println("calculateTriage, Numero historia: ");
        // poner numero historia clinica
        Integer idhistoriaclinica = 4;
        Integer nivelTriage = 4;
        calcularTriageRestClient instance = new calcularTriageRestClient();
        Historiaclinica result = instance.calculateTriage_XML(Historiaclinica.class,idhistoriaclinica);
        assertEquals(nivelTriage, result.getNiveltriage());
        // TODO review the generated test code and remove the default call to fail.
        //fail("The test case is a prototype.")
    }

    /**
```

service.CalcularTriageFacadeRESTTest > testCalculateTriage > nivelTriage

Test Results

service.CalcularTriageFacadeRESTTest

Tests passed: 100.00 %

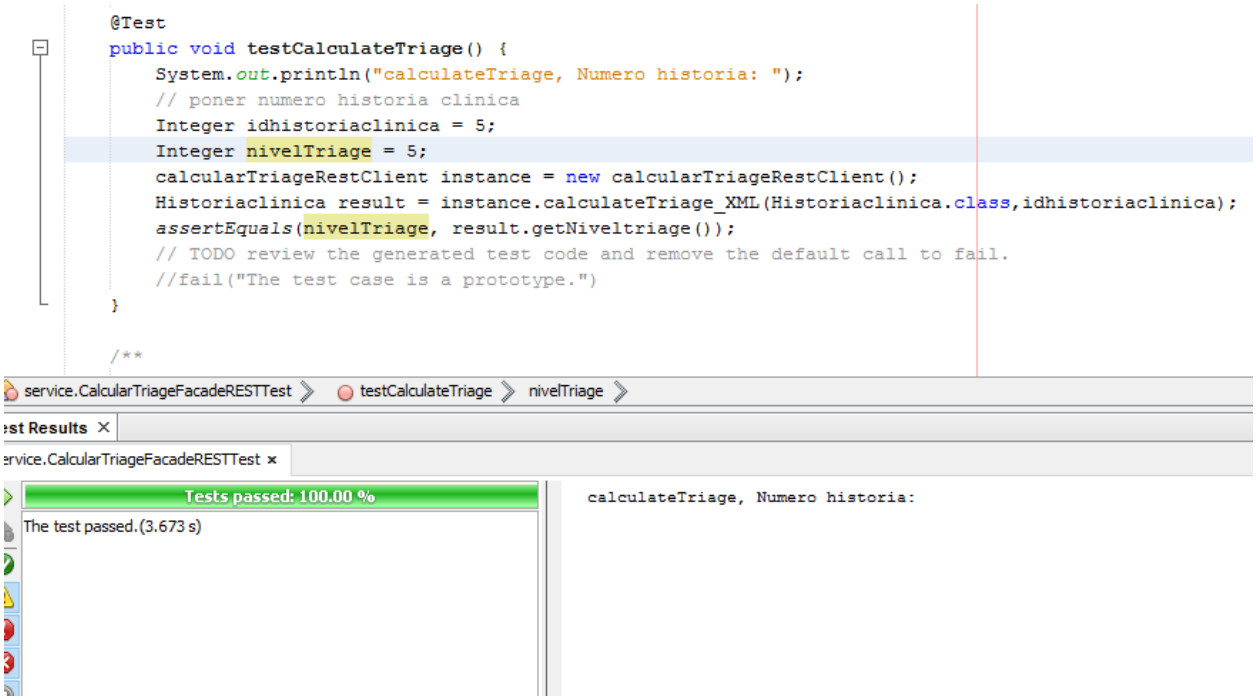
The test passed. (3.142 s)

```
calculateTriage, Numero historia:
```


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

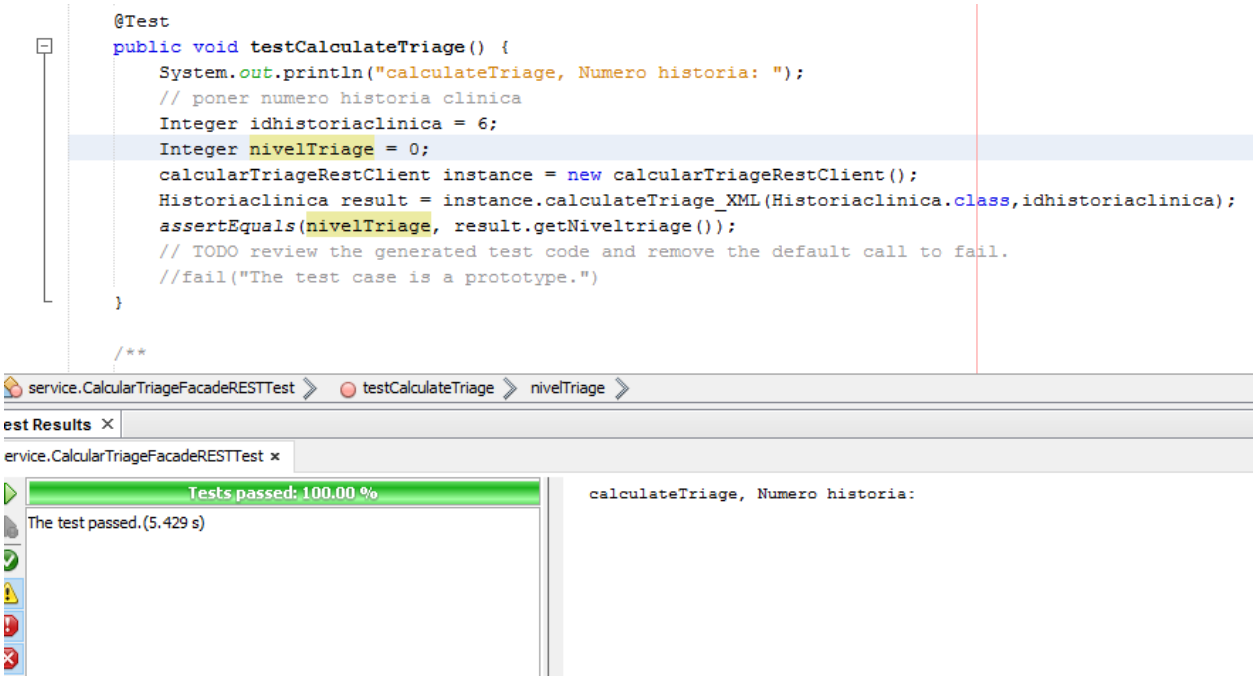
6.3.11 Prueba 9.5

Se llevó a cabo una prueba que realizaba el cálculo de un Triage 2 a partir de unos determinados síntomas.



6.3.12 Prueba 9.6

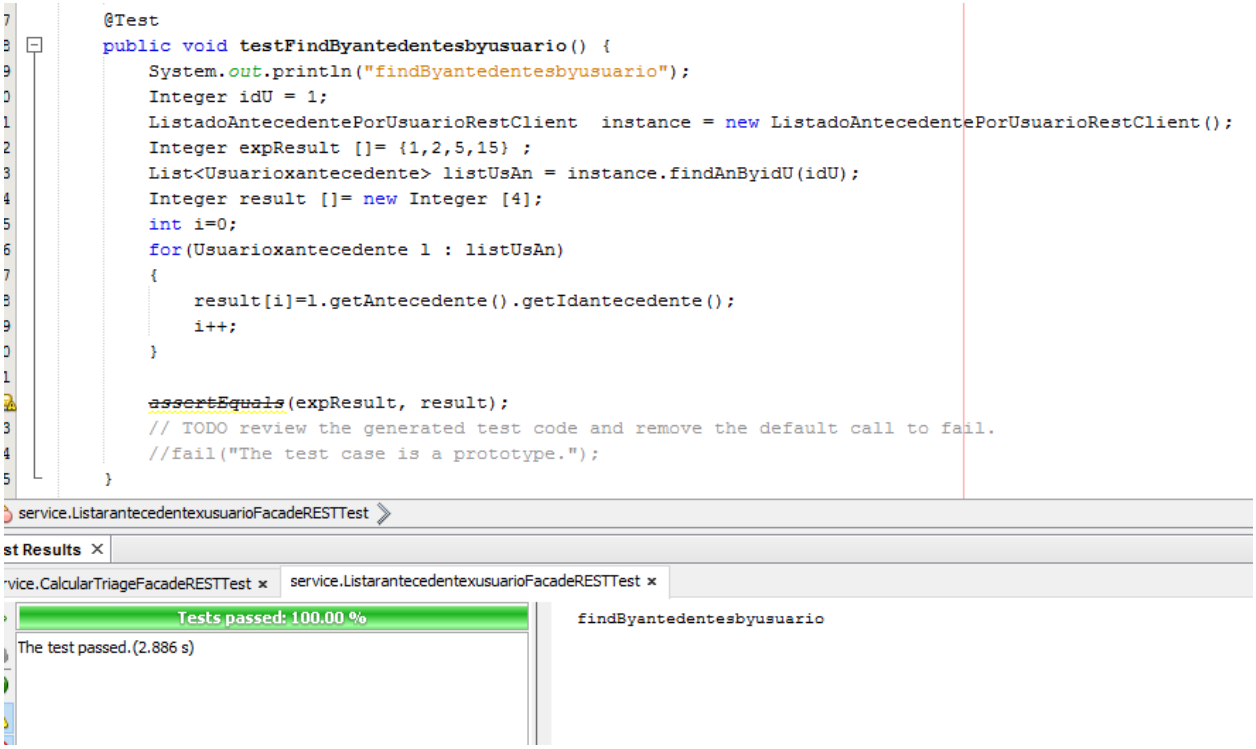
Se llevó a cabo una prueba que realizaba el cálculo de un Triage 0, no existente en la lógica a partir de unos determinados síntomas.



La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

6.3.13 Prueba 16

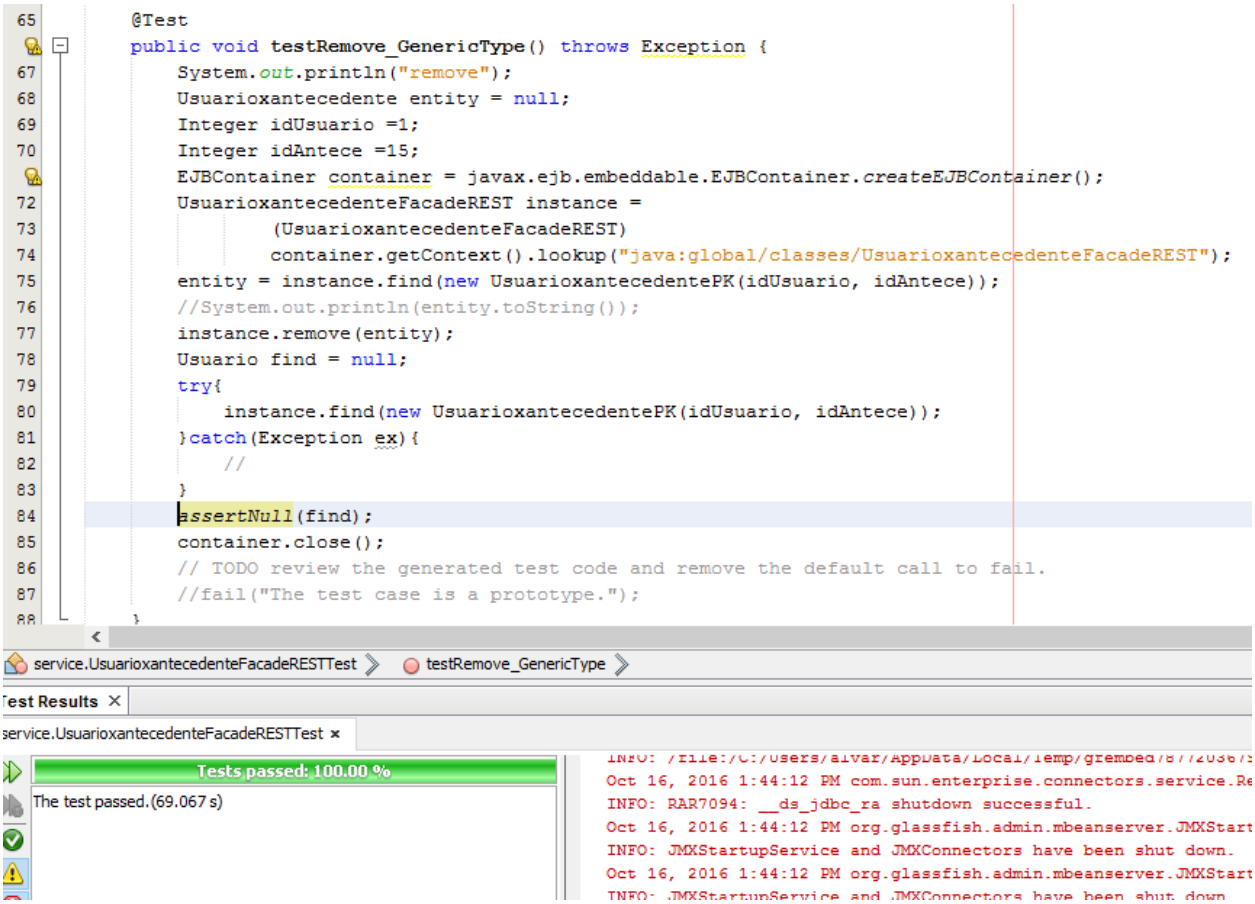
Esta prueba consta de listar antecedentes de un usuario determinado.



6.3.14 Prueba 17

Esta prueba permite ver el borrado correcto de un antecedente.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.



6.3.15 Prueba 18

A continuación se muestra le edición de una tupla de la clase Antecedente y como esta funciona de manera correcta y adecuada.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

```
48      @Test
        public void testEdit_GenericType() throws Exception {
50          System.out.println("edit");
51          UsuarioxantecedentePK id = new UsuarioxantecedentePK(1,1);
52          Usuarioxantecedente entity = null;
            EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
54          UsuarioxantecedenteFacadeREST instance =
55                  (UsuarioxantecedenteFacadeREST)
56                  container.getContext().lookup("java:global/classes/UsuarioxantecedenteFacadeREST");
57          entity = instance.find(id);
58          entity.setAux(true);
59          instance.edit(entity);
60          assertEquals(true, instance.find(id).getAux());
61          container.close();
62          // TODO review the generated test code and remove the default call to fail.
63          //fail("The test case is a prototype.");
64      }
65  //
66  //    /**
67  //     * Test of remove method, of class UsuarioxantecedenteFacadeREST.
```

service.UsuarioxantecedenteFacadeRESTTest › testEdit_GenericType ›

Test Results ×

service.UsuarioxantecedenteFacadeRESTTest ×

Tests passed: 100.00 %

The test passed.(61.887 s)

```
INFO: file:/C:/Users/alvar/AppData/Local/Temp/grembed1829286409
Oct 16, 2016 1:54:07 PM com.sun.enterprise.connectors.service.Re
INFO: RAR7094: __ds_jdbc_ra shutdown successful.
Oct 16, 2016 1:54:07 PM org.glassfish.admin.mbeanserver.JMXStart
INFO: JMXStartupService and JMXConnectors have been shut down.
Oct 16, 2016 1:54:07 PM org.glassfish.admin.mbeanserver.JMXStart
INFO: JMXStartupService and JMXConnectors have been shut down.
```

6.3.16 Prueba 19

Esta prueba permite acceder a todos los antecedentes que presenta el software.

```
189     @Test
        public void testFindAll() throws Exception {
191         System.out.println("findAll");
            EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
193         AntecedenteFacadeREST instance =
                    (AntecedenteFacadeREST)
195                 container.getContext().lookup("java:global/classes/AntecedenteFacadeREST");
196         List<Antecedente> result = instance.findAll();
197
198         assertEquals(16,result.size());
199         container.close();
200         // TODO review the generated test code and remove the default call to fail.
201         //fail("The test case is a prototype.");
```

service.AntecedenteFacadeRESTTest › testFindAll › instance ›

Test Results ×

service.UsuarioxantecedenteFacadeRESTTest × service.AntecedenteFacadeRESTTest ×

Tests passed: 100.00 %

The test passed.(66.288 s)

```
INFO: file:/C:/Users/alvar/AppData/Local/Temp/gre
Oct 16, 2016 2:09:13 PM com.sun.enterprise.connect
INFO: RAR7094: __ds_jdbc_ra shutdown successful.
Oct 16, 2016 2:09:13 PM org.glassfish.admin.mbeans
INFO: JMXStartupService and JMXConnectors have bee
Oct 16, 2016 2:09:14 PM org.glassfish.admin.mbeans
INFO: JMXStartupService and JMXConnectors have bee
```


La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

6.4 CONCLUSIONES

- Las pruebas se realizaron de manera exitosa.
- Si la fachada no se relacionaba directamente con un EJB fue requerido un cliente como proxy para acceder a sus servicios y con ello pobras las diferentes funcionalidades.
- Las pruebas permitieron descubrir un error que se encontraba en el cálculo del Triage, este fue corregido de manera exitosa y se comprobó la efectividad del software de proveer una sistematización tentativa del Triage dadas unas entradas respectivas y una información previamente suministrada en el modelo de persistencia.
- Adicionalmente se utilizó un container standalone en el cual los EJB fueron probados, por lo tanto este demuestra cómo crear una instancia embebida de un contenedor en una clase de JUnit y llamar al session bean de la lógica de negocio. De esta manera no es necesario ejecutar las pruebas en el servidor Java EE sino en el contenedor local.

7 PRUEBAS NO FUNCIONALES

7.1 INFORMACIÓN RELEVANTE

Los servicios y/u operaciones a probar son los casos de uso y las funcionalidades más relevantes del servidor, estos son de tipo GET:

- Login
- Ver historial de historias clínicas de usuario
- Seleccionar historia clínica por id
- Listar síntomas presentes en la historia clínica elegida
- Listar síntomas del sistema
- Calcular el Triage
- Listar antecedentes de usuario
- Listar antecedentes del sistema
- Listar antecedentes familiares
- Contactar con líneas de emergencia

Los días pico de los servicios de urgencias son festivos, quincena y navidad [10], en Colombia hay habilitados 1.612 servicios de urgencias de los cuales 1.110 son públicos. En la Clínica del Country de Bogotá a diario su servicio de urgencias atiende en promedio entre 250 y 300 personas Los días en que más atienden personas son los lunes o martes en donde pueden llegar cerca de 350 pacientes, generalmente los viernes y sábados, la cantidad baja pero la complejidad y la patología varia; las horas con más personas están en la tarde desde las 2, pero la cantidad baja en la madrugada [11]. Por lo anterior se puede llevar a cabo una estimación de número de clientes, si los 1612 servicios de urgencias utilizaran la aplicación y todos estos se encontrarán en días pico de 350 el total de pacientes a atender en un día serian de 1612x350 = 564,200 pacientes/día. Lo que es igual a 23,508 pacientes/hora, 391 pacientes/min y por lo tanto debe atender a 6 pacientes/segundo.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

- Número de peticiones esperadas atendidas por unidad de tiempo: 564,200 pacientes/día. Lo que es igual a 23,508 pacientes/hora, 391 pacientes/min y por lo tanto debe atender a 6 pacientes/segundo.
- Tiempos de respuesta fueron extraídos de un manual de pruebas de servidor [12]
    - Máximos: No deberán superarse tiempos máximos de respuesta en más de 8 segundos.
    - Mínimos:1 seg
    - Promedios esperados: No deberán superarse tiempos medios de respuesta de más de 4 segundos.
- Número de peticiones erróneas esperadas: 0

7.2 PRUEBAS DE CARGA (LOAD)

Las pruebas de carga se realizaron con las siguientes especificaciones:

- Grupos de 12 usuarios (hilos), de acuerdo a la sección anterior se deben probar 6 usuarios, se consideró aumentar la carga para evaluar de mejor manera el rendimiento del servidor y con ello se tuvo en cuenta el Teorema de Shannon, al tener un escenario pesimista de mayores usuarios en la aplicación
- Intervalo de inicio de cada usuario: en 4 seg, este significa que cada 4 segundo se van a disparar los hilos por usuarios.
- Número de veces que se repite la prueba para cada usuario: 10 para cada usuario se van disparar 10 cada una de las pruebas para un total de 120 muestras.
- Porcentaje de peticiones =
  Número de peticiones realmente atendidas por unidad de tiempo / Número de peticiones esperadas atendidas por unidad de tiempo.

Se realizó la prueba con las especificaciones precedentes la cual arrojó una tabla de resultados de JMeter la cual se mostrará posteriormente. Algunos de los datos producidos por la prueba y un análisis producido por los resultados:

- Numero de muestras fue de 120, ya que eran 112 usuarios y las pruebas se repetían en un intervalo de 4 seg 10 veces. Es decir 12 usuarios x 10 veces = 120 pruebas
- Tiempo promedio de muestras en mseg corresponde al tiempo que tomó cada una de las muestras, en promedio las consultas al servidor toman un total de 164mseg, es decir que cumplen con los valores de respuesta que se determinaron para el servidor, ya que los indicadores estipulados de promedio de tiempo de respuesta indicaban 4 segundos y esta prueba resulto en 0.1 segundos.
- La mediana o el valor central de la muestra fue de 132 mseg.
- Las diferentes líneas porcentuales todas cumplen con ser menores a los valores determinados de peticiones que el servidor debía cumplir.
- El tiempo mínimo de todas las pruebas fue de 101 mseg lo que cumple y supera las expectativas del servidor contra el valor estipulado en los indicadores que era de un 1seg.
- El tiempo máximo de todas las pruebas fue de 3275 mseg lo que cumple con la característica de que el tiempo máximo de peticiones debe ser de 8seg.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

- No se presentaron errores, por lo tanto se demuestra que las pruebas unitarias realizadas fueron exitosas ya que los servicios accedidos por URL están correctos.
- Rendimiento o peticiones atendidas por unidad de tiempo cumple con 5.9 seg, siendo el indicador 6 peticiones, por lo tanto 5.9/6 = 98%. Lo que indica que el 98% de las peticiones fueron atendidas.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

La tabla de resultados de JMeter es la siguiente:

Servicio	# Muestra	Promedio ms	Mediana ms	Linea 90%	Linea 95%	Linea 99%	Min ms	Max Ms	% Error	Throughput /seg	KB/seg
login	120	198	137	272	313	699	101	3275	0	5.856515	5.479045
Ver historial de historias clinicas de usuario	120	162	138	220	269	649	109	807	0	5.902897	29.11673
Seleccionar HC por ID	120	144	128	180	258	424	104	534	0	5.908419	7.275895
Listar síntomas presentes en la historia clínica elegida	120	168	136	253	305	680	109	734	0	5.915993	33.54899
Listado de sintomas	120	185	133	284	602	904	105	1045	0	5.997301	20.05348
Cálculo del triage	120	159	131	245	284	528	106	866	0	5.997901	7.386087
Listar antecedentes de usuario	120	160	132	216	271	630	104	644	0	6.009014	18.93074
listar antecedentes	120	153	131	244	290	397	105	403	0	5.997901	13.26684
listar antecedentes familiares	120	153	128	194	262	404	103	1626	0	6.015038	7.477679
Contactar con líneas de emergencia	120	154	126	176	249	466	101	1621	0	6.046254	5.644745
TOTAL	1200	164	132	246	289	677	101	3275	0	55.50673	138.1272

Tabla 13 prueba de carga

A continuación se muestra una gráfica, en el eje x está ubicado el tiempo en minutos que tomó en realizarse la prueba y en el eje y el tiempo en mseg que tomaron las muestras en llevarse a cabo. Como se puede observar todas las muestras se encontraron debajo de 1 segundo. Cabe resaltar que esta grafica muestra las 120 muestras y el tiempo que tomaron en llevarse a cabo que fue en total 22 seg. El login presente un pico de tiempo y esto se debe a la intermitencia de red y la recepción de la respuesta.

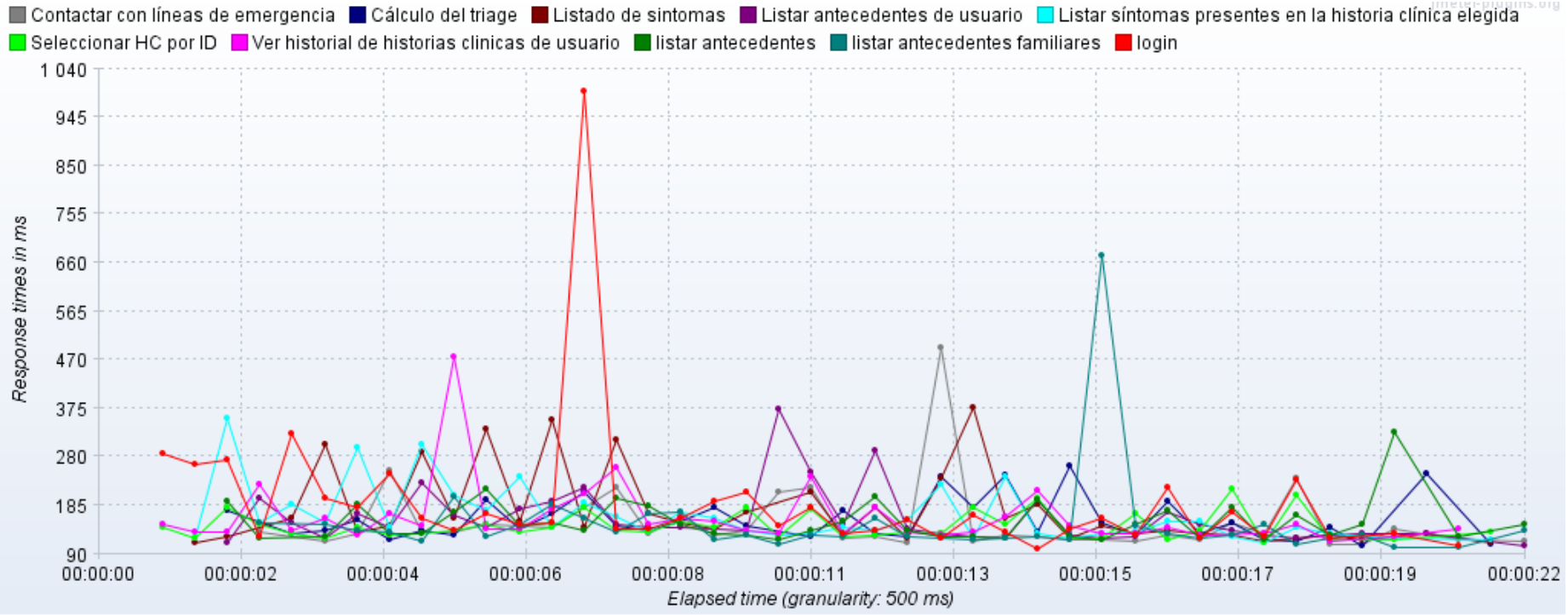


Ilustración 2 pruebas de carga

A continuación se mostrará la gráfica de calcular Triage, esta funcionalidad varia en un tiempo de 100 a 280 mseg, por lo tanto se conoce que dicho cálculo se realiza de manera muy rápida y eficiente.

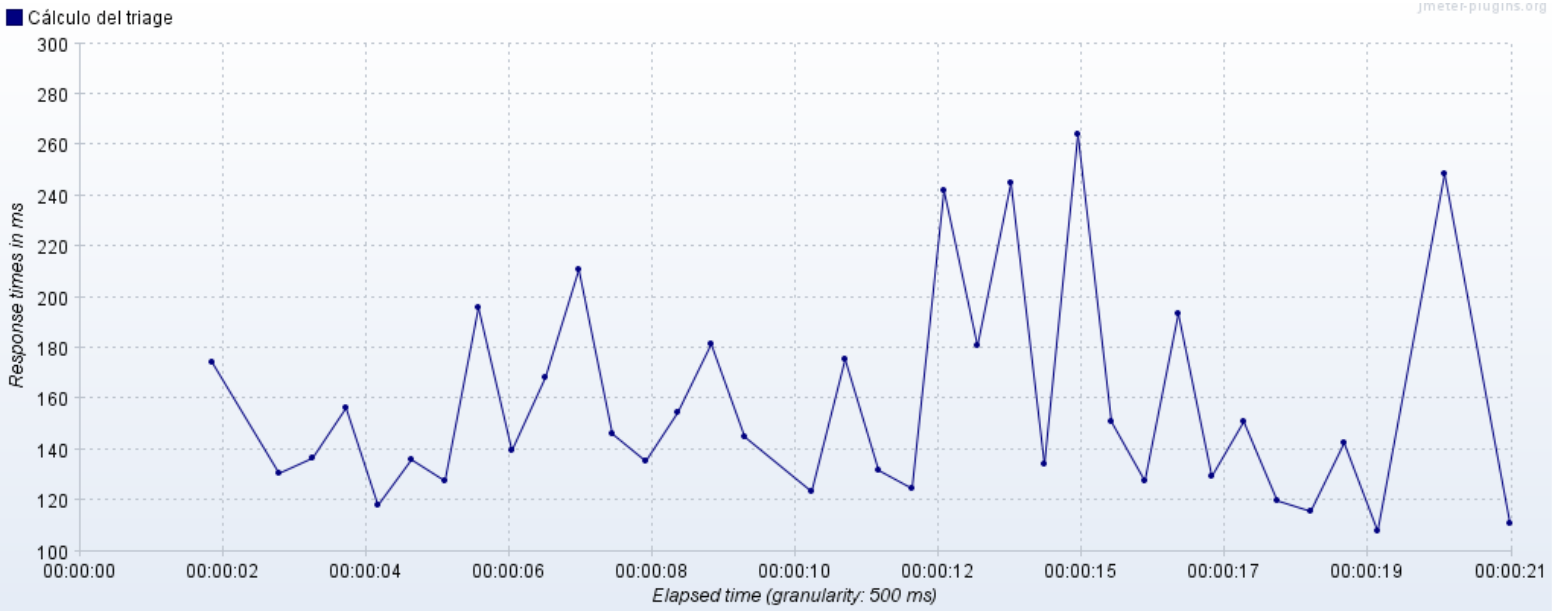


Ilustración 3 calcular triage

La grafica a continuación muestra el uso de la CPU del servidor durante las pruebas en los 22 seg que estas tomaron, se puede ver que en general su uso es bajo, durante las pruebas este subió a un 0.6% en cada respectivo núcleo para un uso total de 1.2%. El eje x representa la hora y el eje y el uso porcentual de la CPU del servidor. Los otros picos que se presentan en la gráfica con pruebas anteriores para evaluar el funcionamiento de JUnit.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo). Utiliza una licencia Creative Commons.

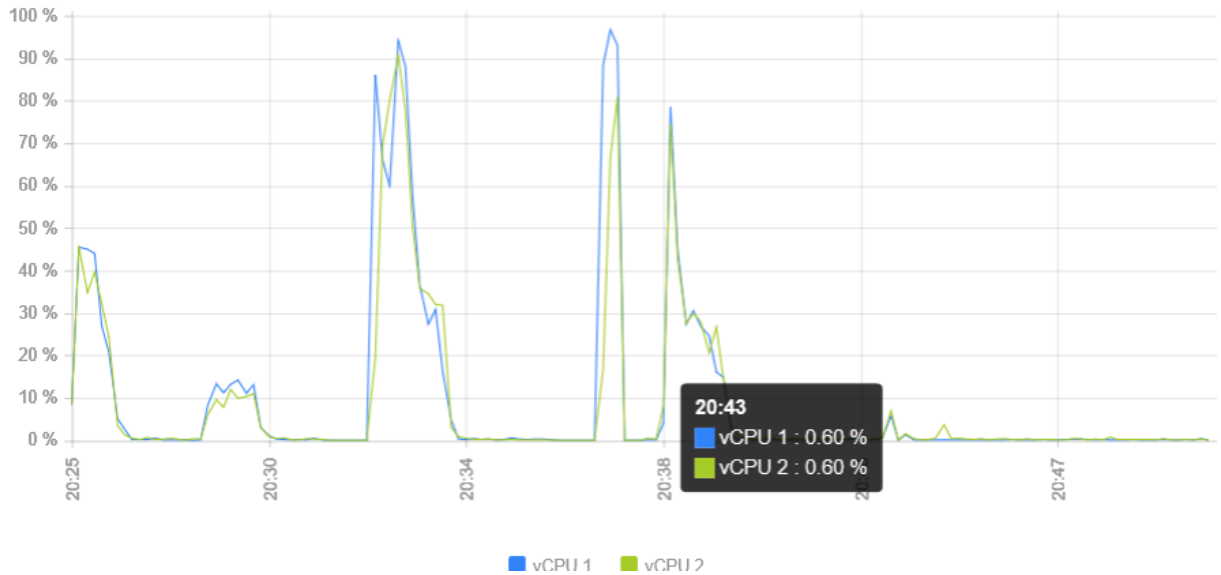


Ilustración 4 cpu prueba de carga

7.3 PRUEBAS DE ESTRÉS (STRESS) Y PRUEBAS DE RENDIMIENTO (PERFORMANCE)

Los siguientes son los pasos de la prueba de carga:

- Establecer grupos de usuarios primero de 10 luego de 50 y aumentar en intervalos de 50.
- Para cada grupo de usuarios ejecutar una prueba de carga.
- Realizar un comparativo de rendimiento y de número de peticiones de cada grupo.
- Establecer una proyección del número de usuarios máximo que puede soportar el aplicativo.

Las pruebas de estrés y rendimiento se realizaron con las siguientes especificaciones para todos los servicios establecidos:

- Grupos de usuarios variables, 10, 50, 100, 150, 200…
- Intervalo de inicio de cada usuario: en 4 seg
- Número de veces que se repite la prueba para cada usuario: 5
- Porcentaje de peticiones =
  Número de peticiones realmente atendidas por unidad de tiempo / Número de peticiones esperadas atendidas por unidad de tiempo.
- Número de funcionalidades 10

En la parte inferior se puede ver una tabla que muestra el tiempo de peticiones y diferentes datos dado todas las funcionalidades determinadas anteriormente del servidor. De esta se puede determinar la siguiente información:

- Los datos de tiempo de respuesta en promedio aumentaron proporcionalmente a las entradas.
- La mediana de los datos aumento proporcionalmente a las entradas.
- Las diferentes líneas también aumentaron proporcionalmente a las entradas.
- El mínimo de los tiempos de los valores se mantuvo constante entre 99 y 104 mseg.

- Se presentó error debido a fallos en conexión o congestion del servidor.
- El rendimiento en todos los casos fue bueno, ya que la tabla completa nos muestra el valor total dadas todas las entradas, por lo tanto la resolución de 12,500 peticiones en 151 segundos.
- La variedad en los valores de los números se da por la velocidad de conexión de internet, la entrada y salida de datos tanto del computador como del servidor.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

Usuarios	# Muestra	Promedio	Mediana	Linea 90%	Linea 95%	Linea 99%	Min	Max	% Error	Throughput 5.6/seg	KB/seg
10	500	166	128	241	289	609	100	4422	0	32.9728304	82.0521156
50	2500	269	243	468	549	1155	103	2103	0	52.6717691	90.1044941
100	5000	505	281	1132	1462	2553	103	9533	0	57.7934439	132.780521
150	7500	732	379	1560	2150	3957	104	26564	0	64.8511882	149.297459
200	11200	863	311	1841	2794	6567	100	54668	0	67.2240662	167.285513
250	12500	1211	513	2997	4684	9800	99	60457	0.1352	151.039149	357.155126
300	14352	1450	544	2544	3959	13209	99	109454	0.001045	128.466317	319.511166

Tabla 14 prueba de estres

En la parte inferior se encuentra un grafica del promedio de las peticiones y la mediana. Se puede observar que a mayor número de usuarios mayor tiempo para solucionar las peticiones debido a que son más, adicionalmente el promedio aumenta bastante mientras la mediana tiende a ser constante, con esto se puede concluir que el dato de la mitad de los datos tiende a ser el mismo mientras que los datos extremos varían bastante y tienden a aumentar.

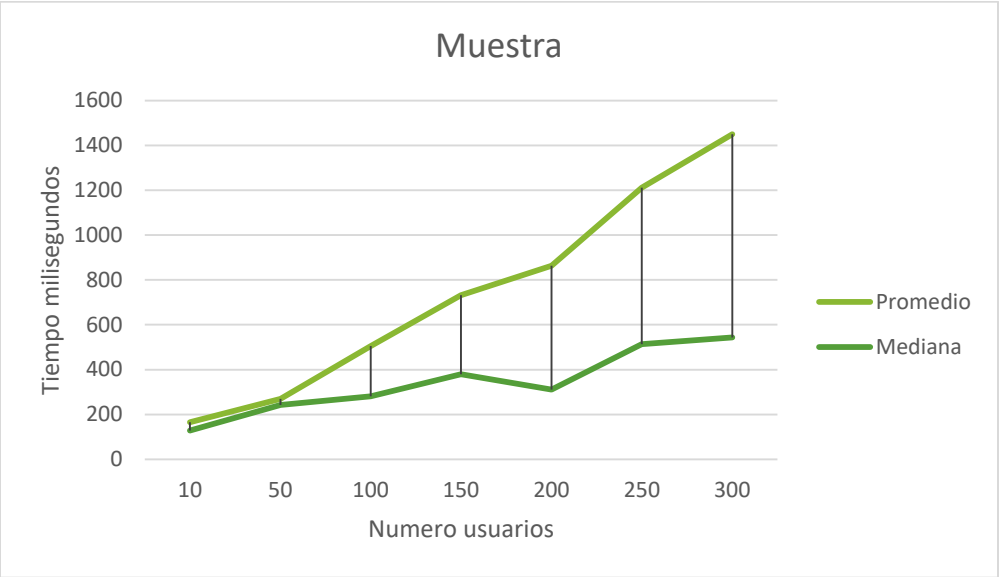


Ilustración 5 grafica prueba estrés

La grafica a continuación muestra el uso de la CPU del servidor durante las pruebas, en flechas rojas se puede observar el uso de la CPU en 250 y en 300 usuarios respectivamente, dicha grafica es extraída de DonWeb el proveedor del servidor. En 250 usuarios para un total de 12000 peticiones se utilizó el 94.2% y el 90.7%, con 300 para 14,352 peticiones el uso fue de 96.7% y 66.5% estas tomaron. El eje x representa la hora y el eje y el uso porcentual de la CPU del servidor. Los otros picos que se presentan en la gráfica con pruebas anteriores para evaluar el funcionamiento de JMeter. La prueba de 300 usuarios fue incapaz de llegar a 15,000 muestras y se detuvo en 14,352 ya que uno de los nucleos de la CPU se encontraba copado.

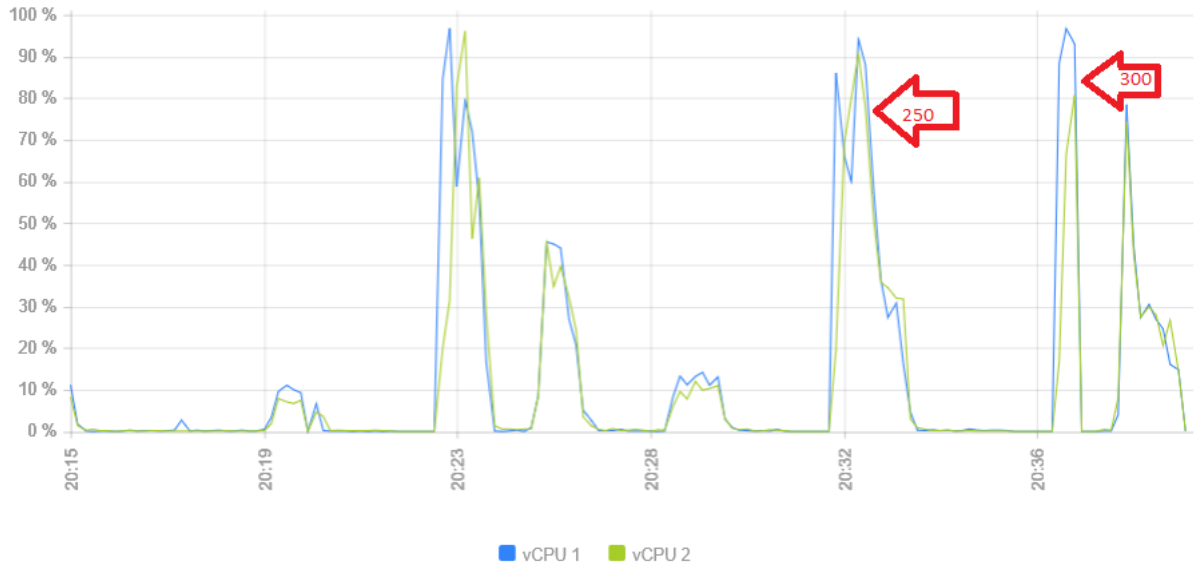


Ilustración 6 uso cpu estrés

7.4 PROYECCIONES

Una de las proyecciones que se puede percibir siendo pesimista teniendo en cuenta que por 12 usuarios que se repite sus peticiones 10 veces y utilizan 10 servicios, para un total de 120 muestras por servicios o 1200 en total, se utiliza el 0.6% de la CPU, por lo tanto, si 12 usuarios es igual a 0.6%, 100 % es igual a (12 x 100)/0.6 = 2000 usuarios teniendo en cuenta que cada uno de sus servicios se pide 10 veces. Si por el contrario se desea calcular es la cantidad de peticiones, entonces 120 es equivalente al 0.6%, por lo tanto (120x100)/0.6 = 20,000 peticiones si se lleva a cabo una proyección de los datos calculados, pero como se puede observar en la práctica no se logró probar más de 15,000.

7.5 CONCLUSIONES

Las conclusiones de las pruebas no funcionales son las siguientes, los datos de tiempo de respuesta, mediana, de las diferentes líneas y del promedio aumentaron proporcionalmente a las entradas. El mínimo de los tiempos de los valores se mantuvo constante entre 99 y 105 mseg. El rendimiento en todos los casos fue bueno, un ejemplo es cuando la muestra completa de todas las funcionalidades nos exhibe que por lo tanto la resolución de 12,500 peticiones se da en 151 segundos. Se presenta alguna variedad en los valores y esto se debe a la velocidad de conexión de internet, la entrada y salida de datos tanto del computador como del servidor y eso también lo reflejan algunos de los errores que se presentaron. Y por último entre mayor es la entrada menor rendimiento de petición/segundo.

8 REFERENCIAS

[1] Institute of Electrical and Electronics Engineers y IEEE Software and Systems Engineering Standards Committee, «IEEE standard for software and system test documentation». IEEE, 2008.

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.

[2] International Organization for Standardization, International Electrotechnical Commission, Institute of Electrical and Electronics Engineers, y IEEE-SA Standards Board, *Software and systems engineering: software testing. Part 3, Test documentation = Ingénierie du logiciel et des systémes : essais du logiciel. Partie 3, Documentation des essais. Part 3, Test documentation = Ingénierie du logiciel et des systémes : essais du logiciel. Partie 3, Documentation des essais.* 2013.

[3] International Organization for Standardization, International Electrotechnical Commission, Institute of Electrical and Electronics Engineers, y IEEE-SA Standards Board, *Software and systems engineering: software testing. Part 2, Part 2*,. 2013.

[4] International Organization for Standardization, International Electrotechnical Commission, Institute of Electrical and Electronics Engineers, y IEEE-SA Standards Board, *Software and systems engineering: software testing. Part 1, Part 1*,. 2013.

[5] *ISO*. ///.

[6] «Live_Project_Test_Plan_SoftwareTestingHelp.pdf». [En línea]. Disponible en: http://www.softwaretestinghelp.com/wp-content/qa/uploads/2014/02/Live_Project_Test_Plan_SoftwareTestingHelp.pdf. [Accedido: 04-oct-2016].

[7] Ministerio de Salud Colombia, «Información para Instituciones Prestadoras de Salud (IPS)», *minSalud*, 02-mar-2016. [En línea]. Disponible en: https://www.minsalud.gov.co/salud/PServicios/Paginas/informacion-de-interes.aspx. [Accedido: 02-mar-2016].

[8] IEEE Computer Society, Software Engineering Technical Committee, American National Standards Institute, IEEE Standards Board, y Institute of Electrical and Electronics Engineers, *IEEE standard for software unit testing*. New York, N.Y.: Institute of Electrical and Electronics Engineers, 1986.

[9] *JUnit - Frequently Asked Questions*. .

[10] «URGENCIAS, EN EMERGENCIA - Archivo Digital de Noticias de Colombia y el Mundo desde 1.990 - eltiempo.com». [En línea]. Disponible en: http://www.eltiempo.com/archivo/documento/MAM-714927. [Accedido: 17-oct-2016].

[11] «Con especialistas buscan cambiar servicio de las salas de urgencias en Colombia», *El Universal Cartagena*, 05-mar-2015. [En línea]. Disponible en: http://www.eluniversal.com.co/salud/con-especialistas-buscan-cambiar-servicio-de-las-salas-de-urgencias-en-colombia-186718. [Accedido: 17-oct-2016].

[12] «ucc_metricas_pruebas_rendimiento_v1.2.pdf». [En línea]. Disponible en: http://www.madrid.org/arquitecturasw/images/documentacion/calidad/actual/generico/ucc_metricas_pruebas_rendimiento_v1.2.pdf. [Accedido: 17-oct-2016].

La información contenida se considera pública (a menos que explícitamente se indique de otro modo).
Utiliza una licencia Creative Commons.