PLAN DE PRUEBAS SERVIDOR

Luisa Álvarez Valencia

TAPPI: TRIAGE APPLICATION

HISTORIAL DE VERSIONES

Versión	Cambios efectuados	Fecha de actualización
V 0.1	Realización de la plantilla con los cambios relevantes. Pruebas unitarias Introducción Estrategias para las pruebas	1 Oct 2016
V0.2	Unit testing y pruebas de conformidad	3 Oct 2016
	Pruebas funcionales	10 Oct 2016
V0.3	Pruebas no funcionales	16 oct 2016

TABLA DE CONTENIDO

1. INTRODUCCIÓN

1.1 PROPÓSITO DEL DOCUMENTO PLAN DE PRUEBAS SERVIDOR

El documento TG – Plan_Pruebas_Servidor brinda las pistas y la información necesaria para definir de manera efectiva el enfoque a utilizar para las pruebas de producto del proyecto. El documento se crea durante la fase de planificación del proyecto. Su público objetivo es el director del proyecto, el equipo de proyecto y equipo de pruebas. Algunas partes de este documento pueden en ocasiones ser compartidos con el cliente / usuario y cualquier otro participante, cuya entrada / aprobación en el proceso de prueba que se necesite. Se utilizaron dos estándares de la IEEE para la realización de dicho documento y una plantilla extraída de CDC (Center for Disease Control and Prevention) [1] y los estándares son el 829 de 2008 [2] y 29119 de 2013 [3]–[6].

ESTRATEGIA PARA LAS PRUEBAS

1.2 OBJETIVO [7]

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

1.3 SUPOSICIONES[7]

- 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.

1.4 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.

1.5 CARACTERISTICAS

- Riesgos de las pruebas / dificultades: describe los riesgos asociados a las pruebas de productos. También delinea las estrategias apropiadas de mitigación y planes de contingencia.
- Herramienta: describe la herramienta de software que es necesaria para la realización de las pruebas
- Elementos a ser probados: describe los elementos / características / funciones a ensayar que están dentro del alcance de este plan de pruebas. Incluye una descripción de la forma en que se pondrá a prueba, cuándo, por quién, y los estándares de calidad que este maneja.

- Enfoque de la(s) prueba(s): describe el enfoque general de la prueba que se utiliza para probar el producto del proyecto. Proporcionar un resumen de las pruebas previstas.
- Criterios de aceptación / fallo de las pruebas: describe los criterios utilizados para determinar si un elemento de prueba aprueba o no la prueba.
- Criterios de entrada / salida de pruebas: describe los criterios de entrada y salida utilizados para iniciar la prueba y determinar cuándo parar la prueba.
- Entregables de las pruebas: corresponden a documentos que avalen que dichas pruebas se llevaron a cabo de manera completa sean los resultados positivos o negativos.
- 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.

1.6 CICLO DE LAS PRUEBAS

El ciclo de las pruebas consta de los diferentes pasos realizados para que los ensayos se llevaran a cabo de manera exitosa. A continuación se numeran los pasos para estas

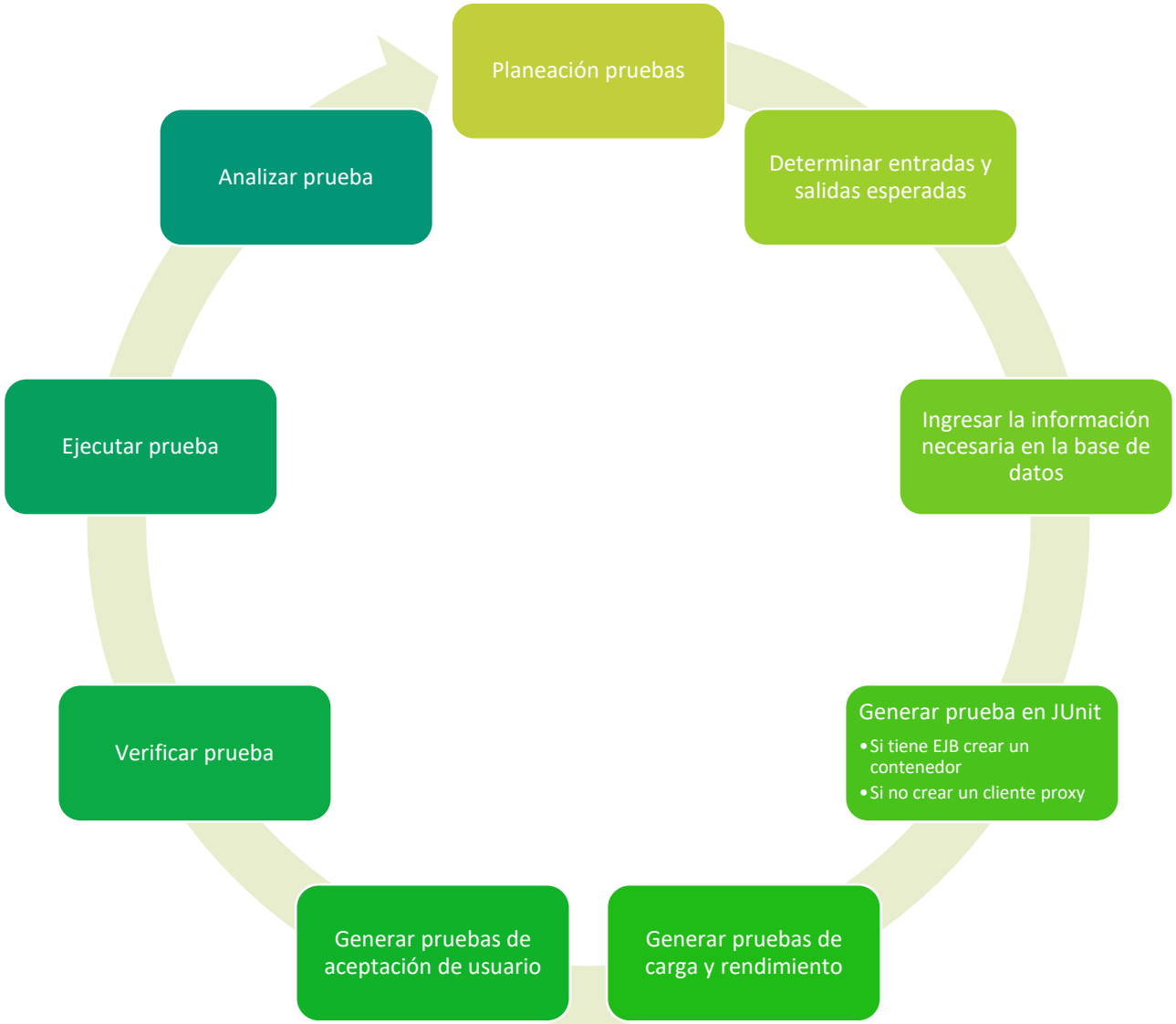


Ilustración 1ciclo vida pruebas

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 [4].

1.7 RIESGOS DE LAS PRUEBAS / DIFICULTADES

Los riesgos de estas pruebas se presentan cuando la especificación de requerimientos no está muy clara y por lo tanto no se pueden llevar a cabo las pruebas de una manera trazable, en la cual se vea la relación de la unidad de prueba con el requerimiento que corresponde. Adicionalmente otro riesgo posible es que las pruebas no se lleven a cabo de manera exitosa, por lo tanto el error se debe reportar, con ello se tienen 3 días hábiles para generar un plan para corregir el error, corregirlo y volverlo a probar.

1.8 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].

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

Id Prueba	Prueba a ser realizada	Id Req	Requerimiento	Fecha
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.	1	El sistema debe dar la opción de iniciar sesión con cuentas de Facebook o google+ (la validación está implícita usando este sistema de login)	10/11/16
2	Se puede identificar un usuario que si existe en el sistema y con ello demostrar que dicho usuario en el login si existe.	4	El sistema debe notificarle al usuario si ha realizado su registro correctamente.	10/11/16
3	Se debe probar el método de PUT para la clase Usuario en el servidor.	5	El sistema debe dar la opción al usuario de editar los datos de su cuenta	10/11/16
4	Se debe probar el método de POST para la clase HistoriaClinica y para la de SintomasXHistoria	6	El sistema debe dar la opción al usuario de ingresar sus síntomas	10/11/16
5	El sistema debe verificar que si las variables son de un tipo de dato	7	El sistema debe verificar que los datos que se van a cambiar sean correctos	10/11/16

	específico estas cumplan con las características			
6	Se debe probar el método de DELETE en Usuario	11	El sistema debe dar la opción de borrar una cuenta creada	10/11/16
7	El sistema debe probar que si la cuenta no existe no se pueda acceder a ella.	13	El sistema debe impedir acceder a una cuenta si esta ha sido borrada	10/11/16
8	Se debe verificar que dicha cuenta eliminada ya no se encuentre en la base de datos	14	El sistema debe eliminar la cuenta de la base de datos cuando esto sea solicitado	10/11/16
9	Se debe verificar que el método de GET en la clase LineasEmergencia funcione.	18	El sistema debe mostrar una lista con las líneas de emergencia	10/11/16
10	Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= congénito funcione.	19	El sistema debe mostrar una lista de posibles enfermedades congénitas.	10/11/16
11	Se debe verificar que el POST de HistoriaClinica funcione puesto que este es quien posee el atributo.	20	El sistema debe almacenar el motivo de la consulta.	10/11/16
12	Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= quirúrgico funcione.	22	El sistema debe almacenar los tratamientos quirúrgicos que el paciente pudo ser sometido.	10/11/16
13	Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= Trastorno mental funcione.	23	El sistema presenta una lista de antecedentes de trastornos mentales posibles (depresión mayor, trastorno bipolar, trastorno psicóticos, ansiedad, trastorno alimentario, trastorno personalidad, impulsividad y agresión)	10/11/16
14	El método de GET de la clase Síntomas trae los síntomas en términos médicos y en términos coloquiales.	24	El sistema debe mostrar una lista de opciones para la anamnesis remota en un vocabulario que comprenda los diferentes usuarios del sistema.	10/11/16
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.	25	Anamnesis remota: El sistema debe almacenar los antecedentes mórbidos (HTA, dislipidemia, asma, diabetes historia de cáncer.)	10/11/16
16	Se debe verificar que el Usuario cuando se cree ingrese su información ginecoobstetrica y por lo tanto el POST debe funcionar.	26	Anamnesis remota: El sistema debe almacenar los antecedentes ginecoobstétricos (Menarquia, menopausia)	10/11/16
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.	27	Anamnesis remota: El sistema debe almacenar los hábitos del usuario (Tabaco, Alcohol, Otras).	10/11/16
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.	29	Anamnesis remota: El sistema debe almacenar los medicamentos que la persona consume.	10/11/16
19	Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= alergias funcione. Y que estos	30	Anamnesis remota: El sistema debe almacenar las alergias que el paciente presenta.	10/11/16

	sean almacenados en AntecedenteXUsuario.			
20	Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= familiar funcione. Y que estos sean almacenados en AntecedenteXUsuario.	31	Anamnesis remota: El sistema almacena los Antecedentes familiares.	10/11/16
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.	32	Anamnesis remota: El sistema almacena las inmunizaciones de usuario.	10/11/16

Tabla 1: req funcionales

1.10 ENFOQUE DE LA(S) PRUEBA(S)

Las pruebas se realizarán con la ayuda de la herramienta JUnit para probar cada uno de los métodos de las diferentes clases y su interacción con el sistema con el fin de demostrar que dichos requerimientos funcionan.

1.11 CRITERIOS DE ACEPTACIÓN / FALLO DE LAS PRUEBAS

Describe cada uno de los criterios para que las pruebas sean exitosas o fallida. A continuación se lista el Id de la prueba y cada uno de sus criterios. Cabe resaltar que algunas pruebas se unieron ya que se relacionaban con el mismo elemento a ser ensayado.

1.11.1 Prueba 1 y 2

Descripción 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.

Descripción 2: Se puede identificar un usuario que si existe en el sistema y con ello demostrar que dicho usuario en el login si existe.

Aceptación: que el id de Facebook o Google corresponde al de un usuario que existe en la aplicación.

Fallo: Dicho usuario no existe y por lo tanto no se encuentra registrado en la aplicación.

1.11.2 Prueba 3

Descripción: Se debe probar el método de PUT para la clase Usuario en el servidor.

Aceptación: el usuario existe y por lo tanto sus datos se pueden editar.

Fallo: el usuario no existe o los datos de este no se permiten editar.

1.11.3 Prueba 4 y 11

Descripción 4: Se debe probar el método de POST para la clase HistoriaClinica y para la de SintomasXHistoria

Descripción 11: Se debe verificar que el POST de HistoriaClinica funcione puesto que este es quien posee el atributo motivo de consulta.

Aceptación: se crea una HistoriaClinica y en esta se permite el ingreso de sus síntomas.

Fallo: no se permite la creación de la HistoriaClinica o permite crear la HistoriaClinica pero no ingresar los síntomas.

1.11.4 Prueba 5

Descripción: El sistema debe verificar que si las variables son de un tipo de dato específico estas cumplan con las características.

Aceptación: las variables son de determinado tipo de dato y gracias a esta se pueden llevar a cabo las diferentes acciones del sistema.

Fallo: así las variables no concuerden el sistema permite realizar los cambios que se le envíen.

1.11.5 Prueba 6

Descripción: Se debe probar el método de DELETE en Usuario.

Aceptación: se eliminó un usuario de la aplicación y toda su información asociada.

Fallo: no se permite eliminar un usuario.

1.11.6 Prueba 7, 8

Descripción 7: El sistema debe probar que si la cuenta no existe no se pueda acceder a ella.

Descripción 7: Se debe verificar que dicha cuenta eliminada ya no se encuentre en la base de datos

Aceptación: una vez sea eliminada una cuenta de usuario no se permita ver sus datos ni acceder a ella nuevamente.

Fallo: la cuenta se encuentra en la base de datos y se puede acceder a ella.

1.11.7 Prueba 9

Descripción: Se debe verificar que el método de GET en la clase LineasEmergencia funcione.

Aceptación: se pueden observar todas las líneas de emergencia disponibles.

Fallo: no se muestran las líneas de emergencia

1.11.8 Prueba 10, 12, 13

Descripción 10: Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= congénito funcione.

Descripción 12: Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= quirúrgico funcione.

Descripción 13: Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= Trastorno mental funcione.

Aceptación: se obtienen los diferentes Antecedentes de acuerdo a su tipo.

Fallo: no se obtienen los Antecedentes de acuerdo a su tipo.

1.11.9 Prueba 14

Descripción: El método de GET de la clase Síntomas trae los síntomas en términos médicos y en términos coloquiales.

Aceptación: se pueden observar todos los síntomas que tiene un paciente de una determinada dolencia.

Fallo: no se tiene acceso a los síntomas.

1.11.10Prueba 15, 17, 18, 19, 20, 21

Descripción 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.

Descripción 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.

Descripción 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.

Descripción 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.

Descripción 20: Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= familiar funcione. Y que estos sean almacenados en AntecedenteXUsuario.

Descripción 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.

Aceptación: se tiene acceso a los antecedentes de acuerdo a su tipo y se pueden asociar a determinado usuario ya que existen.

Fallo: no se tiene acceso a los antecedentes, o se tiene acceso pero no se permite asociarlos a un usuario.

1.11.11Prueba 16

Descripción: Se debe verificar que el Usuario cuando se cree ingrese su información ginecoobstetrica y por lo tanto el POST debe funcionar.

Aceptación: se permite la creación correcta de un usuario.

Fallo: no se permite crear un usuario.

1.12 CRITERIOS DE ENTRADA / SALIDA DE PRUEBAS

Describe cada uno de los criterios entrada y de salida de cada una de las pruebas. A continuación se lista el Id de la prueba y cada uno de sus criterios. Cabe resaltar que algunas pruebas se unieron ya que se relacionaban con el mismo elemento a ser ensayado.

1.12.1 Prueba 1 y 2

Descripción 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.

Descripción 2: Se puede identificar un usuario que si existe en el sistema y con ello demostrar que dicho usuario en el login si existe.

Entrada: id de Facebook o Google

Salida: el usuario con su identificador

1.12.2 Prueba 3

Descripción: Se debe probar el método de PUT para la clase Usuario en el servidor.

Entrada: un usuario con todos sus campos diligenciados.

Salida: observar la persistencia de dicho usuario en la base de datos por medio de un GET.

1.12.3 Prueba 4 y 11

Descripción 4: Se debe probar el método de POST para la clase HistoriaClinica y para la de SintomasXHistoria

Entrada: una historia clínica con todos sus campos diligenciados y síntomasxHistoria asociarlo a una historiaClinica

Salida: ver la persistencia de los datos en la base de datos de la clase HistoriaClinica y de SintomaXHistoria

1.12.4 Prueba 5

Descripción: El sistema debe verificar que si las variables son de un tipo de dato específico estas cumplan con las características.

Entrada: ingresar las variables del tipo que corresponde en alguna de las clases e ingresar las variables de tipo que no corresponden de ese modo probar ambos casos.

Salida: permite el ingreso de los datos cuando estos son del tipo que corresponde y no permitir el ingreso cuando no.

1.12.5 Prueba 6

Descripción: Se debe probar el método de DELETE en Usuario.

Entrada: el identificador único del usuario que se desea eliminar.

Salida: el usuario ya no persiste en la base de datos.

1.12.6 Prueba 7, 8

Descripción 7: El sistema debe probar que si la cuenta no existe no se pueda acceder a ella.

Descripción 7: Se debe verificar que dicha cuenta eliminada ya no se encuentre en la base de datos

Entrada: después de eliminar un usuario verificar que este ya no exista en la base de datos.

Salida: la cuenta se eliminó correctamente y ya no persiste.

1.12.7 Prueba 9

Descripción: Se debe verificar que el método de GET en la clase LineasEmergencia funcione.

Entrada: un identificador único de líneas de emergencia o acceder a todas las LineasEmergencia.

Salida: se obtienen todas las LineasEmergencia

1.12.8 Prueba 10, 12, 13

Descripción 10: Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= congénito funcione.

Descripción 12: Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= quirúrgico funcione.

Descripción 13: Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= Trastorno mental funcione.

Entrada: el tipo de atributo de Antecedente al cual se desea tener acceso.

Salida: los antecedentes de ese tipo.

1.12.9 Prueba 14

Descripción: El método de GET de la clase Síntomas trae los síntomas en términos médicos y en términos coloquiales.

Entrada: obtener todos los síntomas que almacena el sistema.

Salida: lista de síntomas.

1.12.10Prueba 15, 17, 18, 19, 20, 21

Descripción 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.

Descripción 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.

Descripción 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.

Descripción 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.

Descripción 20: Se debe verificar que el método de GET en la clase Antecedentes con atributo Tipo= familiar funcione. Y que estos sean almacenados en AntecedenteXUsuario.

Descripción 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.

Entrada: un tipo de Antecedente

Salida: los antecedentes que corresponden al tipo y estos asociados al usuario que corresponde.

1.12.11Prueba 16

Descripción: Se debe verificar que el Usuario cuando se cree ingrese su información ginecoobstetrica y por lo tanto el POST debe funcionar.

Entrada: un usuario con todos los datos de este diligenciados.

Salida: la persistencia del usuario en la aplicación.

1.13 ENTREGRABLES DE LAS PRUEBAS

Un documento con fotos de pantalla de las pruebas llevadas a cabo por JUnit con las salidas esperadas y la implementación.

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 [5], como son los casos de uso.

1.14 RIESGOS DE LAS PRUEBAS / DIFICULTADES

Los riesgos de estas pruebas se presentan cuando la especificación de los casos de uso no está muy clara y por lo tanto no se pueden llevar a cabo las pruebas de una manera trazable, en la cual se vea la relación de la unidad de prueba con el caso de uso que corresponde. Adicionalmente otro riesgo posible es que las pruebas no se lleven a cabo de manera exitosa, por lo tanto el error se debe reportar, con ello se tienen 3 días hábiles para generar un plan para corregir el error, corregirlo y volverlo a probar.

1.15 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.

Id prueba	Prueba	Funcionalidad – Caso de uso	Característica	Prueba	Fecha
1	GET, LoginFacadeREST. A partir de un id que se ingresa por la URL busca al usuario.	Iniciar Sesión	La lógica de negocio cuenta con login, el cual a partir de un id de Facebook o google permite acceder a la información del usuario	No, ya que corresponde a la Prueba 1,2 de Unit Testing	10/11/16
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	Permite al usuario ver el historial de historias clínicas y de acuerdo a esto elegir una. (Se muestra la fecha para saber cuál es más reciente). Cabe resalta que las historias clínicas cesan a ser editables a partir de una semana después de su creación.	Dado un identificador de un usuario se obtienen las historias clínicas	10/11/16
3	GET, HistoriaclinicaFacadeREST Dado un id se obtiene la historia clínica en XML.	Seleccionar HC por ID	Permite al usuario seleccionar alguna de sus HC si no desea crear una nueva. Dada una lista de HC, que se le proporciono al cliente, este elige una; se retorna un id y con ese id se busca la HC correspondiente.	Dado un id de historia clínica se obtiene la historia clínica correspondiente	10/11/16
4	POST, HistoriaclinicaFacadeREST. A partir de la URL se realiza un post a esta entidad	Crear Historia clínica	Se crea la historia clínica a la cual asociar las dolencias del paciente cuando este ingrese sus síntomas.	Se puede crear una historia clínica	10/11/16
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)	Permite modificar la historia clínica en caso de que los datos de ella se hayan ingresado incorrectamente.	Dado un id de una historia clínica esta puede editarse	10/11/16
6	Dado un id de HC se identifican los síntomas que esta tiene.	Listar síntomas presentes en la historia clínica elegida	Permite listar los síntomas que ya se tienen diligenciados en la historia clínica que se eligió.	Dado un id de una historia clínica se conocen los síntomas a los que está asociada.	10/11/16
7	PUT, SintomasxhistoriaFacadeREST. Utiliza una llave primaria compuesta y por lo tanto los campos que permiten edición son presencia y fecha. Se debe recalcular el nivel del Triage por la edición de los síntomas.	Elegir síntoma de historia clínica para edición	Permite a partir de la lista de síntomas que se envió que el usuario elija un id de un síntoma dada una historia clínica y lo edite.	Permite ingresar síntomas que pertenecen a una historia clínica.	10/11/16

8	DELETE, SintomasxhistoriaFacadeREST. Utiliza una llave compuesta para lograr borrar los datos de esa tabla	Elegir síntoma de historia clínica para borrar	Permite a partir de la lista de síntomas que se envió que el usuario elija un id de un síntoma dada una historia clínica y lo borre.	Permite borrar síntomas de una historia clínica	10/11/16
9	El usuario ingresa los síntomas de su dolencia y la lógica de negocio debe dar un nivel de prioridad tentativo (Triage).	Ingresar Síntomas		Calculo del Triage dado unos síntomas ingresados	
10	GET, SintomasFacadeREST, es una fachada de la entidad que contiene todos los síntomas. Trae todos los síntomas que el usuario pueda presentar.		Listado de síntomas	Permite ver todos los sintomas, o dado un id de un síntoma ver el sintoma	10/11/16
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.		Vincular síntoma a usuario solicitando si presenta o no el síntoma	Se permite hacer los ingresos a sintomaxhistoria	10/11/16
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. Este metodo debe llamarse al final de borrar o editar síntomas para que vuelva a ejecutarse el cálculo.		Cálculo del triage	Calculo del triage dado unos sintomas ingresados	10/11/16
13	PUT, UsuarioFacadeREST Edita un usuario dando como entrada los datos de un usuario en XML.	Editar Usuario	Permite editar la información de un usuario	Permite editar un usuario	10/11/16
14	POST, UsuarioFacadeREST Crea un usuario dando como entrada los datos de un usuario en XML.	Crear Usuario Ingresar datos ginecológicos	Permite crear un usuario a partir de información suministrada.	Permite crear un nuevo usuario	10/11/16
15	DELETE, UsuarioFacadeREST Elimina un usuario dado un id como entrada los datos de un usuario en XML.	Eliminar Usuario	Se elimina un usuario de la aplicación y la información queda almacena por si es necesaria en un archivo de persistencia.	Permite borrar un usuario	10/11/16

16	ListarantecedentexusuarioFacadeREST. Realiza un find por una query con id de usuario.	Listar antecedentes de usuario	Lista los antecedentes de un usuario dado su id.	Busca los antecedentes asociados a un usuario	10/11/16
17	UsuarioxantecedenteFacadeREST. Este utiliza una llave primaria compuesta de: idantecedente y idusuario.	Eliminar antecedentes de usuario	Elimina un antecedente de un usuario.	Dado un id de un usuario elimina los antecedentes asociados a este	10/11/16
18	UsuarioxantecedenteFacadeREST. Este utiliza una llave primaria compuesta de: idantecedente y idusuario.	Modificar antecedentes de usuario	Permite modificar la información del usuario	Permite modificar algún antecedente asociado a un usuario	10/11/16
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. Es un método que permite recibir el id del antecedente asociado al id del usuario, utiliza UsuarioxantecedenteFacadeREST.	Ingresar antecedentes familiares	La lógica de negocio permite ingresar antecedentes familiares de un usuario determinado.	Se obtienen todos los antecedentes.	10/11/16
20	GET, LineasemergenciaFacadeREST. Permite obtener todas las líneas de emergencia que posea la tabla insertada.	Contactar con líneas de emergencia	El servidor expone la información de líneas de emergencia	Se obtiene toda la información de todas las lineas de emergencia	10/11/16
21	Este utiliza diferentes métodos dictados en la parte superior, como es eliminar un usuario.	Administrar cuentas	Restringir el uso de las funcionalidades a ciertos usuarios	Permite la eliminación de un usuario dado un id	10/11/16

Tabla 2 casos de uso servidor

1.16 ENFOQUE DE LA(S) PRUEBA(S)

Las pruebas se realizarán con la ayuda de la herramienta JUnit para probar cada uno de los métodos de las diferentes clases y su interacción con el sistema con el fin de demostrar que dichos casos de uso funcionan.

1.17 CRITERIOS DE ACEPTACIÓN / FALLO DE LAS PRUEBAS

Describe cada uno de los criterios para que las pruebas sean exitosas o fallida. A continuación se lista el Id de la prueba y cada uno de sus criterios.

Id prueba	Prueba	Aceptación	Fallo
1	GET, LoginFacadeREST. A partir de un id que se ingresa por la URL busca al usuario.	NA	NA
2	GET, ListaHistoriaClinicaXUsuarioFacadeREST. A partir de un identificador un usuario, esta fachada devuelve las historias clínicas que tiene ese paciente.	Se encuentran las historias clínicas de un usuario determinado	No encuentra la historia clínica o encuentra la de un usuario que no le corresponde.
3	GET, HistoriaclinicaFacadeREST Dado un id se obtiene la historia clínica en XML.	Se obtiene la historia clínica dado un id	No se obtiene la historia clínica o se obtiene otra
4	POST, HistoriaclinicaFacadeREST. A partir de la URL se realiza un post a esta entidad	Se crea una historia clínica	No se crea una historia clínica y no persiste en la base de datos.
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.	Se edita una historia clínica dado un id	No se permite la edición de la historia clinica o no se encuentra el id.
6	Dado un id de HC se identifican los síntomas que esta tiene.	Dado un id de una historia clínica se conocen los síntomas asociados.	No se listan los síntomas
7	PUT, SintomasxhistoriaFacadeREST. Utiliza una llave primaria compuesta y por lo tanto los campos que permiten edición son presencia y fecha.	Se permite la edición de unos sintomas de una historia clinica determinada.	No se permite la edición de sintomas.

	Se debe recalcular el nivel del triage por la edición de los síntomas.		
8	DELETE, SintomasxhistoriaFacadeREST. Utiliza una llave compuesta para lograr borrar los datos de esa tabla	Dado un determinado id de una historia clinica y un síntoma este se puede eliminar	No se encuentra el id del síntoma o de la historia clinica y por lo tanto no se puede borrar.
9	El usuario ingresa los síntomas de su dolencia y la lógica de negocio debe dar un nivel de prioridad tentativo (Triage).	Permite dado unas entradas de sintomas, calcular el Triage	No calcula el Triage correcto
10	GET, SintomasFacadeREST, es una fachada de la entidad que contiene todos los síntomas. Trae todos los síntomas que el usuario pueda presentar.	Permite obtener todos los síntomas que tiene la aplicación.	No obtiene los sintomas
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.	Se permite el ingreso de síntomas a una determinada historia clínica	No se pueden asociar unos sintomas a una historia clinica
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. Este metodo debe llamarse al final de borrar o editar síntomas para que vuelva a ejecutarse el cálculo.	Permite dado unas entradas de sintomas, calcular el Triage	No calcula el Triage correcto
13	PUT, UsuarioFacadeREST Edita un usuario dando como entrada los datos de un usuario en XML.	Permite la edición de un usuario	No permite la edición de un usuario
14	POST, UsuarioFacadeREST Crea un usuario dando como entrada los datos de un usuario en XML.	Permite la creación de un nuevo usuario	No permite crear un nuevo usuario y este no persiste en la base de datos.
15	DELETE, UsuarioFacadeREST Elimina un usuario dado un id como entrada los datos de un usuario en XML.	Elimina un usuario dado un determinado id	No elimina el usuario o no encuentra el id
16	ListarantecedentexusuarioFacadeREST. Realiza un find por una query con id de usuario.	Permite buscar los antecedentes	No permite ver los

		de un usuario dado su id	antecedentes de un usuario asociado.
17	UsuarioxantecedenteFacadeREST. Este utiliza una llave primaria compuesta de: idantecedente y idusuario. Eliminar	Permite eliminar un antecedente perteneciente a un usuario	Este no se ve reflejado en la base de datos
18	UsuarioxantecedenteFacadeREST. Este utiliza una llave primaria compuesta de: idantecedente y idusuario. Editar	Permite editar alguno de los antecedentes asociados a un usuario	Se ve en la base de datos los cambios
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. Es un método que permite recibir el id del antecedente asociado al id del usuario, utiliza UsuarioxantecedenteFacadeREST.	Permite ver todos los antecedentes posibles de un usuario	No permite ver los antecedentes
20	GET, LineasemergenciaFacadeREST. Permite obtener todas las líneas de emergencia que posea la tabla insertada.	Permite el acceso a todas las líneas de emergencia	No se pueden ver las líneas de emergencia
21	Este utiliza diferentes métodos dictados en la parte superior, como es eliminar un usuario.	Se elimina un usuario de la aplicación dado un id	No se elimina un usuario

Tabla 3 prueba cu aceptación-fallo

1.18 CRITERIOS DE ENTRADA / SALIDA DE PRUEBAS

Describe cada uno de los criterios entrada y de salida de cada una de las pruebas. A continuación se lista el Id de la prueba y cada uno de sus criterios.

Id prueba	Prueba	Entrada	Salida
1	GET, LoginFacadeREST. A partir de un id que se ingresa por la URL busca al usuario.	NA	NA
2	GET, ListaHistoriaClinicaXUsuarioFacadeREST. A partir de un identificador un usuario, esta fachada devuelve las historias clínicas que tiene ese paciente.	Identificador de usuario	Historias clinicas asociadas a este
3	GET, HistoriaclinicaFacadeREST Dado un id se obtiene la historia clínica en XML.	Id de una historia clinica	La historia clinica que tiene ese id
4	POST, HistoriaclinicaFacadeREST. A partir de la URL se realiza un post a esta entidad	Una historia clinica	Se agregra una historia clinica a la base de datos

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.	Dado un id y una historia clinica esta se edita	Historia clínica editada y encontrada en el modelo de persistencia
6	Dado un id de HC se identifican los síntomas que esta tiene.	Id de historia clinica	Síntomas que esta historia tiene asociados
7	PUT, SintomasxhistoriaFacadeREST. Utiliza una llave primaria compuesta y por lo tanto los campos que permiten edición son presencia y fecha. Se debe recalcular el nivel del triage por la edición de los síntomas.	Id de síntomas x historia que permite editar la lista de síntomas que tiene asociado.	Los cambios se muestran en la persistencia.
8	DELETE, SintomasxhistoriaFacadeREST. Utiliza una llave compuesta para lograr borrar los datos de esa tabla	Id de sintomasxhistoria	La información fue borrada y no se muestra en la base de datos.
9	El usuario ingresa los síntomas de su dolencia y la lógica de negocio debe dar un nivel de prioridad tentativo (Triage).	Ingreso de los síntomas del usuario.	Valor del Triage tentativo.
10	GET, SintomasFacadeREST, es una fachada de la entidad que contiene todos los síntomas. Trae todos los síntomas que el usuario pueda presentar.	Obtener todos los sintomas	Todos los síntomas
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.	Entrada los sintomasxhistoria de un cliente.	Se pueden ver los síntomas asociados a una determinada historia.
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. Este metodo debe llamarse al final de borrar o editar síntomas para que vuelva a ejecutarse el cálculo.	Síntomas que ingresa el paciente como su dolencia	Numero de Triage tentativo
13	PUT, UsuarioFacadeREST Edita un usuario dando como entrada los datos de un usuario en XML.	Edita un usuario	Los cambios de Usuario se ven reflejados en la base de datos.

14	POST, UsuarioFacadeREST Crea un usuario dando como entrada los datos de un usuario en XML.	Crea un usuario dando como entrada un usuario.	El usuario es agregado al modelo de persistencia.
15	DELETE, UsuarioFacadeREST Elimina un usuario dado un id como entrada los datos de un usuario en XML.	El id de un usuario para identificarlo.	El usuario es eliminado del modelo de persistencia
16	ListarantecedentexusuarioFacadeREST. Realiza un find por una query con id de usuario.	Dado un id de un usuario.	Se listan los antecedentes de este.
17	UsuarioxantecedenteFacadeREST. Este utiliza una llave primaria compuesta de: idantecedente y idusuario. Eliminar	Llave de usuarios por antecedentes	Se elimina el antecedente de la base
18	UsuarioxantecedenteFacadeREST. Este utiliza una llave primaria compuesta de: idantecedente y idusuario. Editar	Llave de usuarios por antecedentes.	Se ven los cambios en la base de datos.
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. Es un método que permite recibir el id del antecedente asociado al id del usuario, utiliza UsuarioxantecedenteFacadeREST.		Se obtienen todos los antecedentes posibles
20	GET, LineasemergenciaFacadeREST. Permite obtener todas las líneas de emergencia que posea la tabla insertada.		Se obtienen todas las líneas de emergencia
21	Este utiliza diferentes métodos dictados en la parte superior, como es eliminar un usuario.	Dado un id de un usuario	Este es eliminado de la base de datos.

Tabla 4 criterios cu entrada-salida

1.19 ENTREGRABLES DE LAS PRUEBAS

Un documento con fotos de pantalla de las pruebas llevadas a cabo por JUnit con las salidas esperadas.

PRUEBAS NO FUNCIONALES (NON FUNCTIONAL)

Las pruebas no funcionales se basan en los requerimientos de software de TAppi asociadas a medidas o tácticas para que estos se cumplan. Una táctica de software ayuda en vincular atributos de calidad contra decisiones arquitecturales y requerimientos.

Table 1 req no funcionales

Id Prueba	Prueba a ser realizada / información	Id Req	Requerimiento	Fecha
1	Táctica: la inyección de SQL se previene ya que se utilizan consultas parametrizadas y no concatenación de cadenas.	33	El sistema debe evitar ataques de Inyección SQL.	19/10/2016
2	Táctica: las especificaciones mínimas que debe manejar el celular se encuentran en el SRS.	34	El sistema debe poder ser accedido desde cualquier Smartphone Android	19/10/2016
3	Se realizaron pruebas finales con un usuario y se obtuvo retroalimentación.	35	El sistema debe ser fácil de entender para el usuario.	19/10/2016
4	Esta prueba se lleva a cabo como una prueba de estres.	37	El sistema debe responder a las acciones solicitadas en menos de un minuto	19/10/2016
6	Táctica: se maneja versionamiento y la documentación pertinente	41	Cuando se instala una nueva versión del sistema todos los ajustes y configuraciones que hayan sido definidos previamente persistirán.	19/10/2016
7	Táctica: podría estar programado en esa herramienta o en otra que proporcione funcionalidades que permitan su programación para celular.	43	El sistema debe ser programado en Android Studio.	19/10/2016
8	Táctica: la programación se realizó en Java EE.	45	La aplicación deberá utilizar un paradigma orientado a objetos.	19/10/2016
9	Táctica: el servidor utilizado para la aplicación tiene un tope de almacenamiento de 20GB	46	El sistema debe almacenar datos correspondientes a 2 GB de espacio.	19/10/2016
10	Táctica: la aplicación TAppi utiliza una base de datos en su servidor la cual almacena la información de sus usuarios.	47	Los usuarios deben autenticarse ante el servicio de directorio centralizado de usuarios de Tappi.	19/10/2016
11	Táctica: en la herramienta en la cual se encuentra alojado el servidor se cuanta con recuperación de información y Snapshots para volver a la última versiona almacenada.	48	El sistema debe recuperar su estado frente a un fallo.	19/10/2016
12	En la base de datos los síntomas tienen dos entradas una para lenguaje médico y otra para una persona coloquial. Se debe validar por rol en la pantalla.	50	El sistema debe utilizar un vocabulario simple para el usuario paciente.	19/10/2016
13	En la base de datos los síntomas tienen dos entradas una para lenguaje médico y otra para una persona coloquial. Se debe validar por rol en la pantalla.	51	El sistema debe utilizar un vocabulario técnico para el paciente médico.	19/10/2016
14	En la memoria y otros archivos se explica que la lógica se almacena en clases y en una secuencia de reglas y no en un árbol.	52	La persistencia del árbol de toma de decisión se encontrará en un archivo XML	19/10/2016

1.20 PASOS PARA LAS PRUEBAS

Esta sección tiene como propósito proporcionar los pasos que se tomaron para la realización de las pruebas no funcionales

1.20.1 Planeación de la prueba

Para el caso de prueba, se definieron los objetivos de la prueba de carga, esto es, se definieron las operaciones y/o servicios web va a probar. Se seleccionaron los servicios y/u operaciones de dichos servicios web que se consideraron tenían la mayor carga de consultas de los usuarios y/o aplicativos.

Se hizo una estimación del número de clientes del servicio web. Se debe tener en cuenta también las horas picos del servicio y cuántos clientes estarán activos durante dichas horas pico.

Se establecieron unas referencias para determinar si las pruebas eran o no satisfactorias, a saber:

- Número de peticiones esperadas atendidas por unidad de tiempo: este número al comienzo se puede obtener basados en el número de usuarios.
- Tiempos de respuesta máximos, mínimos y promedios esperados.
- Número de peticiones erróneas esperadas: por ejemplo 0 implicaría que no se esperan errores.

NOTA: En este punto se puede notar que se podrían generar peticiones erróneas lo que implicaría que también se estaría soportando el proceso de pruebas funcionales en sí mismo.

Establezcer indicadores:

Porcentaje de peticiones =

Número de peticiones realmente atendidas por unidad de tiempo / Número de peticiones esperadas atendidas por unidad de tiempo

1.20.2 Creación de la prueba

La estrategia contempla la utilización de la herramienta JMeter. La herramienta provee todas las funcionalidades requeridas para realizar las pruebas de carga sobre los servicios web de la aplicación.

La creación de la prueba se realiza de forma manual. Vea la sección de creación y ejecución del plan de pruebas para servicios web más adelante en este documento.

1.20.3 Ejecución de la prueba

Una vez se ha creado la prueba, se definen el número de hilos (usuarios) con los cuales se realizará las pruebas de carga. El número de usuarios debe definirlo acorde al número de usuarios esperado para el servicio web. Se deben ejecutar un número grande de instancias de las pruebas para evitar los efectos de cachés y otras estrategias que podrían ensuciar los resultados. Se pueden almacenar los resultados de las diferentes instancias de las pruebas para posterior análisis.

1.20.4 Análisis de resultados

Compare los indicadores esperados establecidos en el paso 1 contra el resultado de las pruebas y establezca un porcentaje entre dichos indicadores. Basados en los anteriores valores planeados y obtenidos de la prueba decida si dicha prueba es exitosa o no exitosa. Por ejemplo si esperaba que se atendieran 100 peticiones por segundo y sólo se obtuvieron 80 entonces su porcentaje de cumplimiento sería del 80%.

El análisis debería extenderse en cuanto sea posible para:

- Identificar los errores de programación (de acuerdo al número de peticiones erróneas)
- Mejorar la arquitectura: de acuerdo a los tiempos de respuesta muy elevados se podrían sugerir modificaciones a la arquitectura del aplicativo que soporta el servicio web.

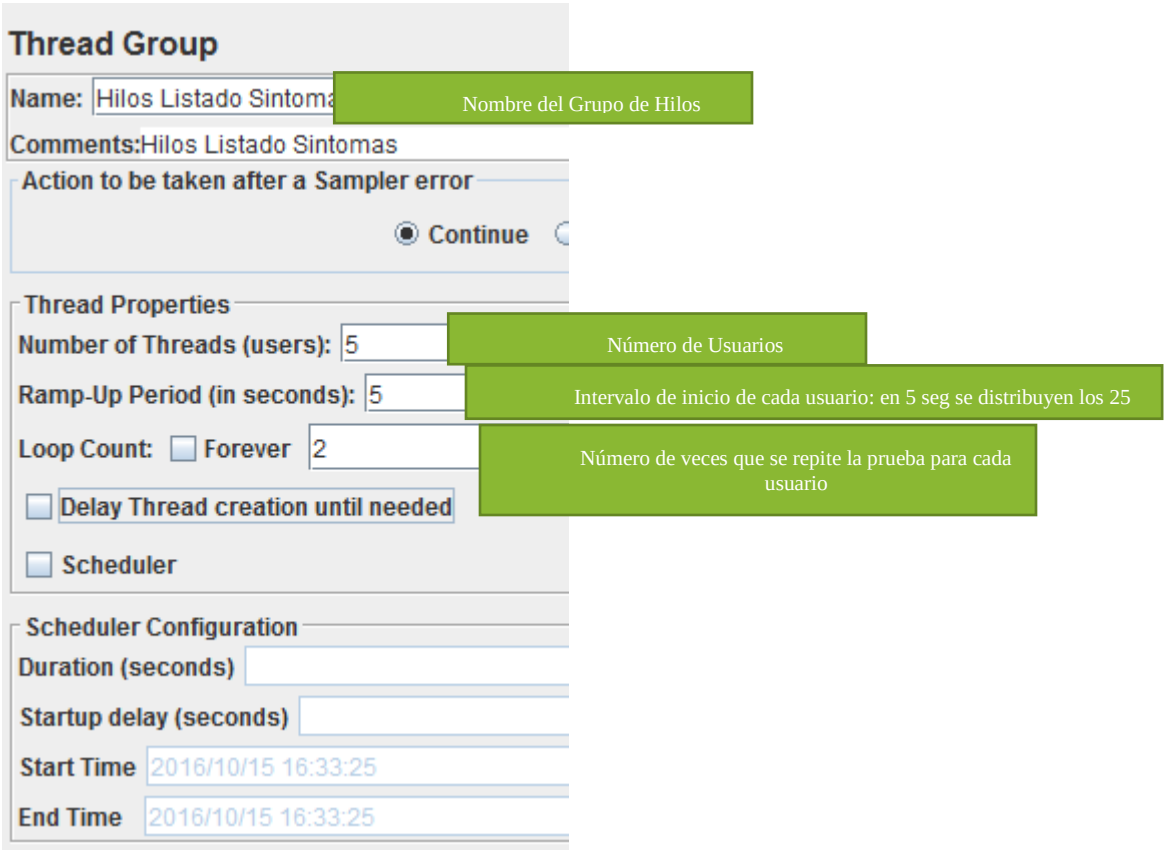
1.21 CREACIÓN Y EJECUCIÓN DEL PLAN DE PRUEBAS PARA SERVICIOS WEB

Para la creación y ejecución del Plan [10], [11]:

1.21.1 Agregar un Thread Group element.

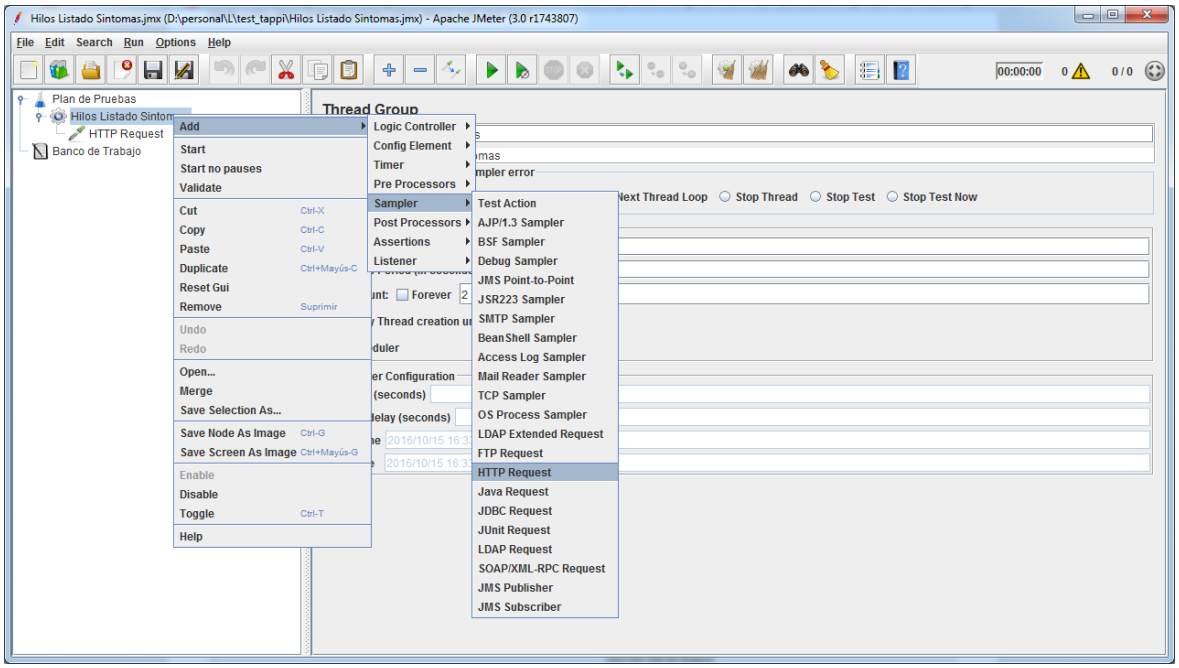
1.21.2 Asignar las propiedades que se ven en la siguiente figura:

Nota: Un grupo de Hilos por cada operación de cada servicio a probar



NOTA: para las pruebas de estrés se irán cambiando paulatinamente los anteriores valores hasta alcanzar la saturación del servidor.

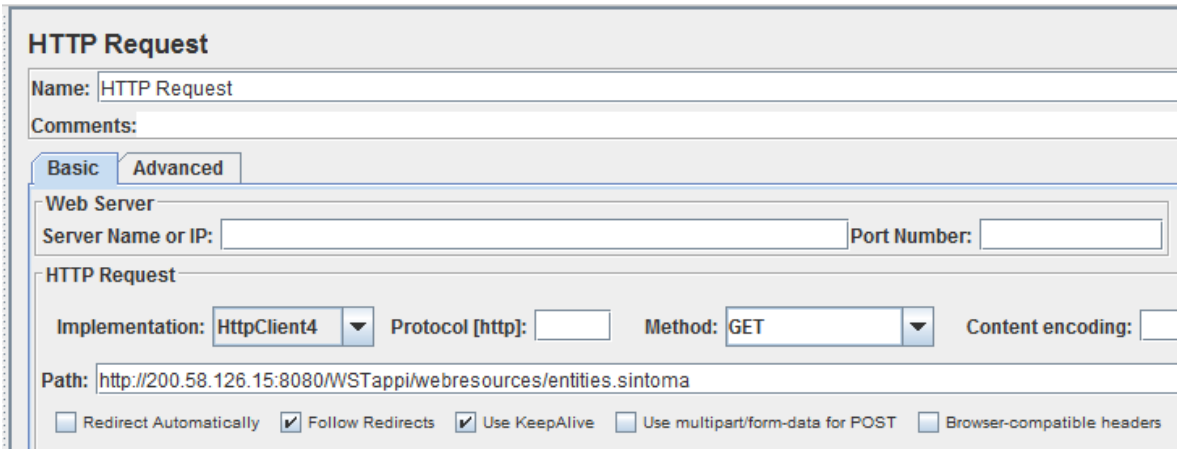
1.21.3 Agregar un muestreador de tipo HTTP:



Configure:

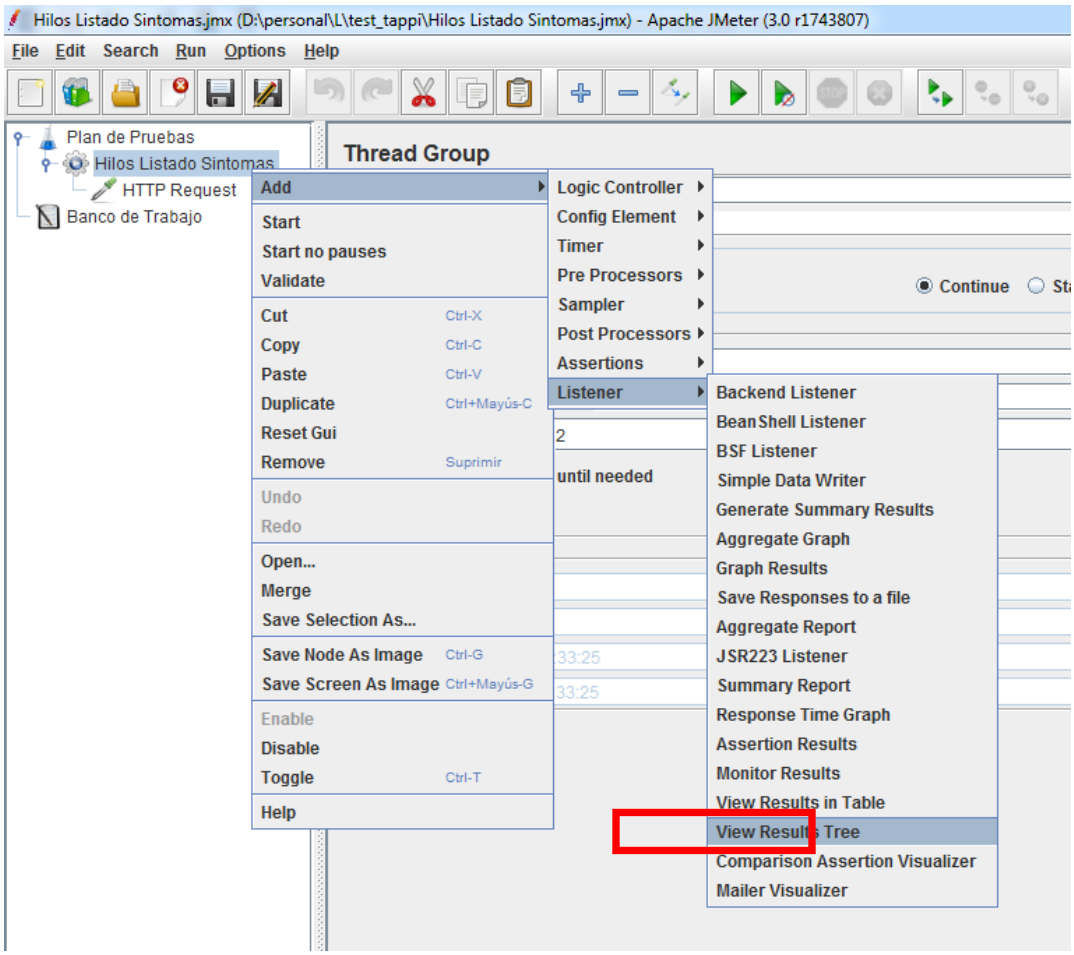
- IP del servidor: coloque la IP del servidor y el nombre del contexto de los recursos
- El atributo path: con el path a probar
- Implementation: HttpClient4
- Method: el método a usar del HTTP, en este caso GET

Verifique que las opciones “Follow Redirects” y “Use KeepAlive” están activas

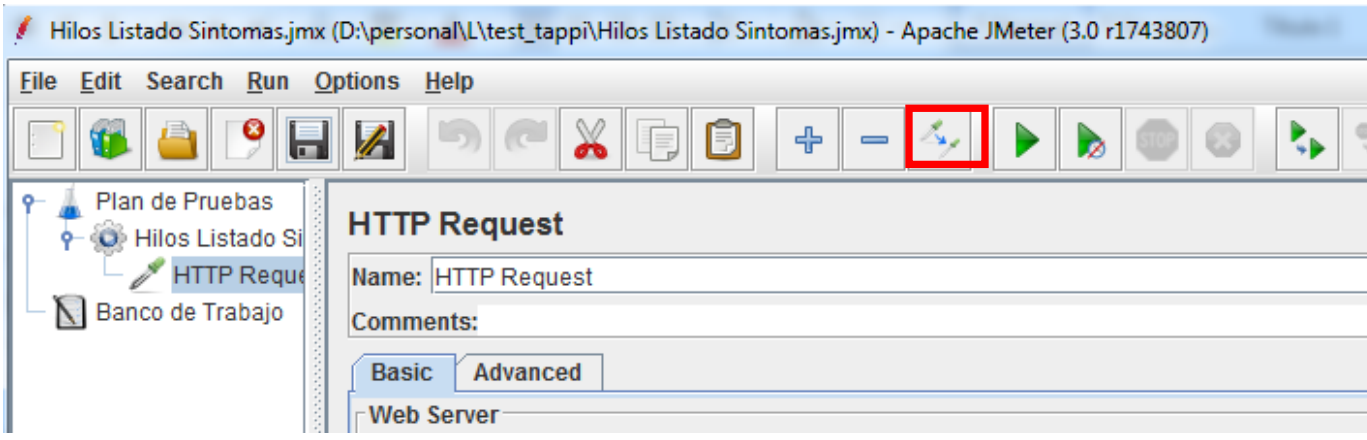


1.21.4 Ejecute el testt

Agregue un receptor de “Ver Árbol de Resultados” que muestra el detalle de cada petición:



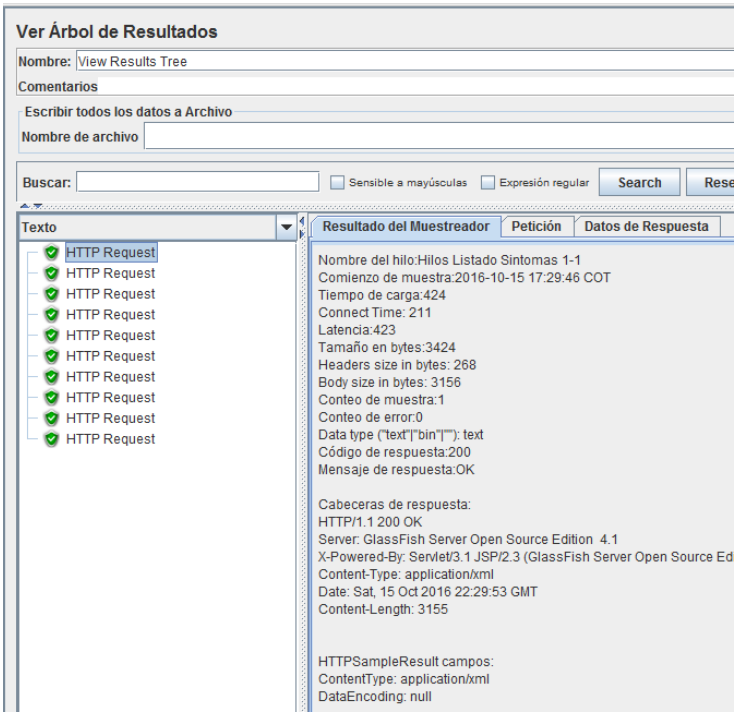
Ejecute el test:



NOTA: En el momento de realizar la ejecución de pruebas debería removerse el receptor de "Ver Árbol de Resultados".

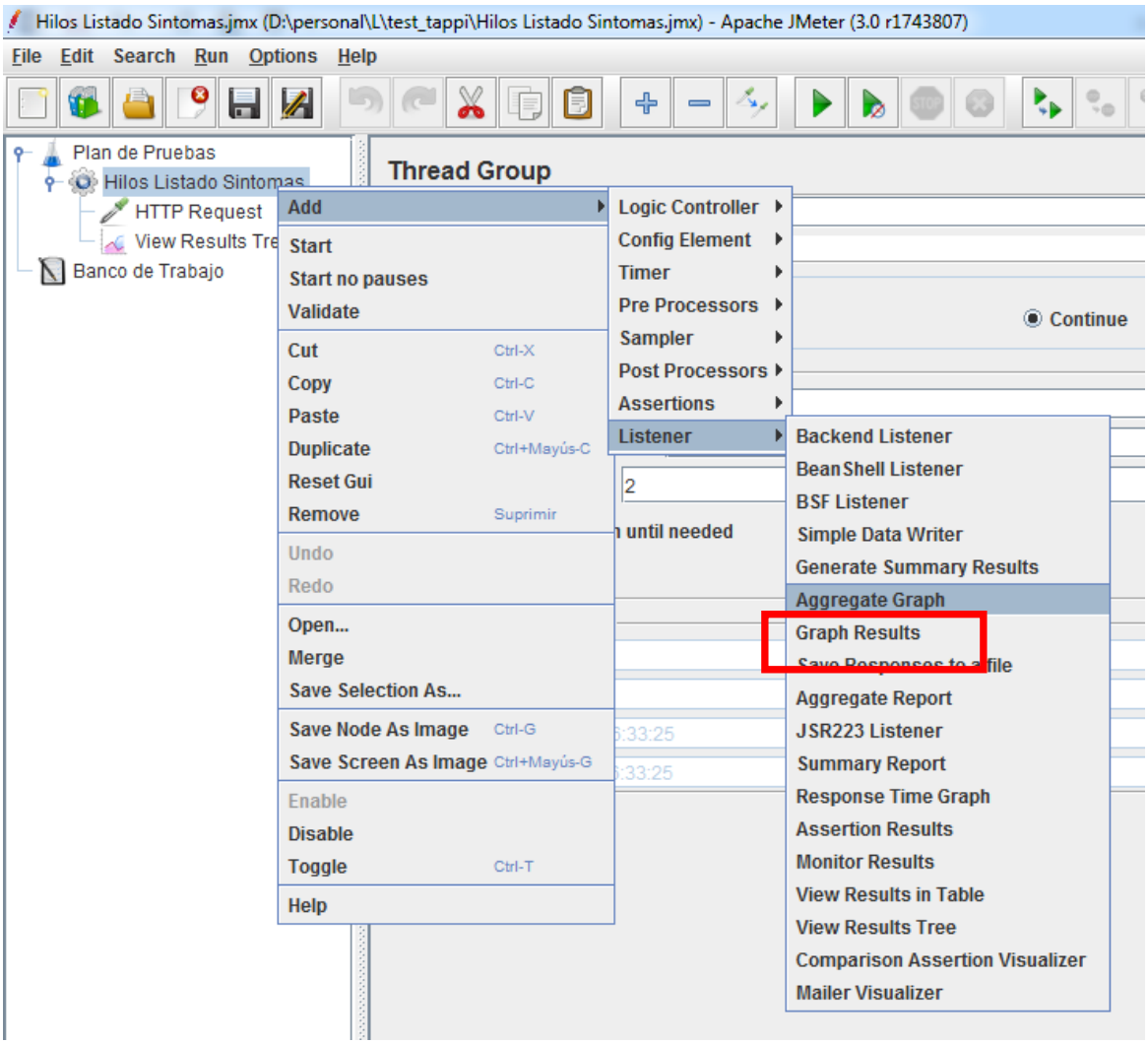
NOTA: Si Al ejecutar aparece en la parte superior derecha de la pantalla un icono de warning, haga clic sobre dicho icono y verifique y corrija los errores que se muestran.

Se revisa el árbol de peticiones para verificar que no hay errores:

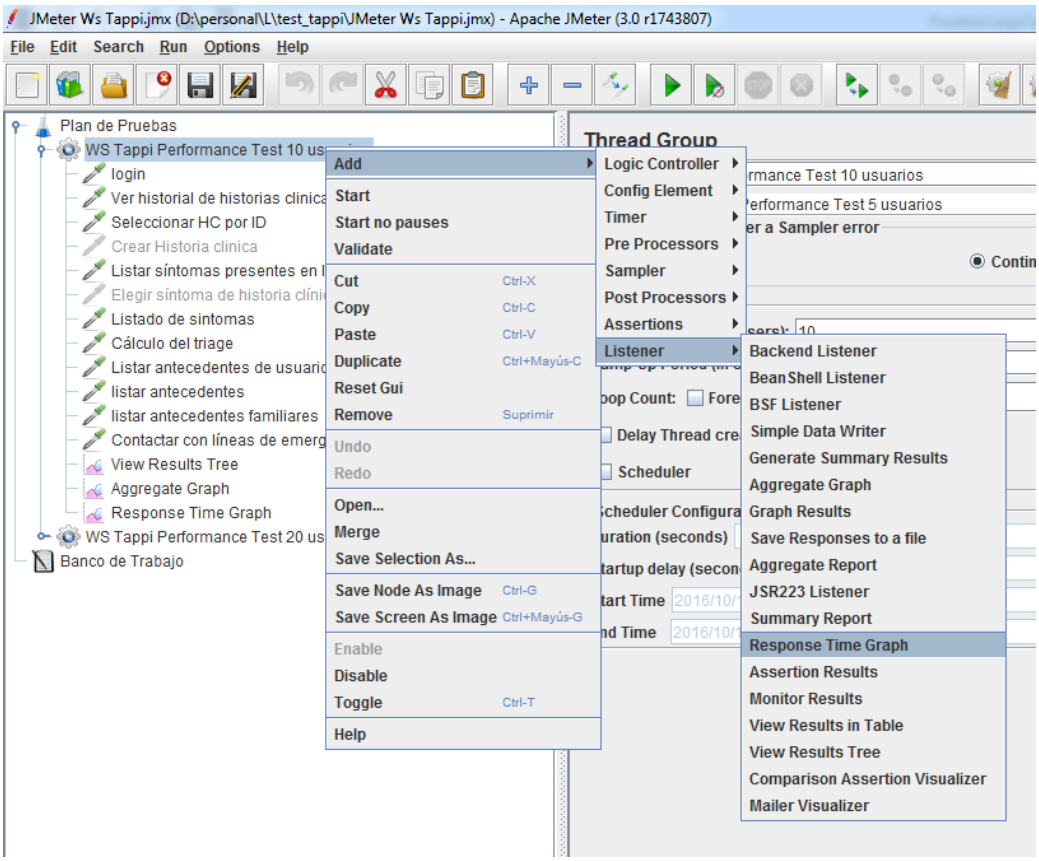


Y los datos de la respuesta:

Agregamos un “Aggregate Graph” para recoger resultados:

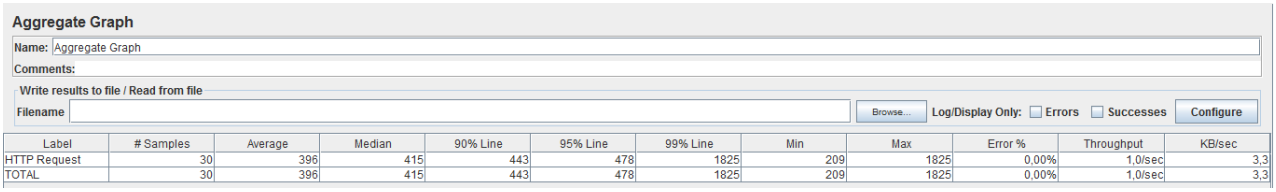


Agregamos un Listener de “response Time Graph” para representar el tiempo de respuesta de las peticiones:

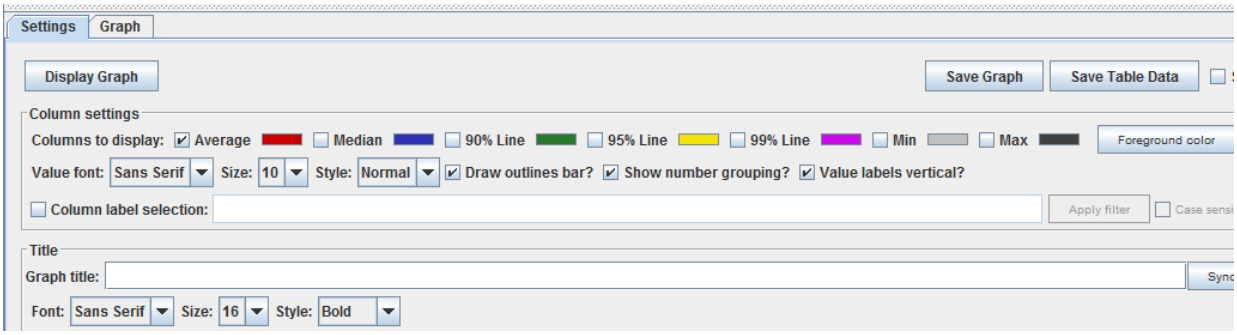


1.21.5 Análisis de resultados

Revise el gráfico agregado de resultados:



Puede guardar los datos en csv, haga clic en el botón “Save Table Data”.



La siguiente es la descripción de los valores arrojados:

- Etiqueta: contiene los valores “HTTP Request” y “Total”
- # Muestras: corresponde al número de peticiones
- Media: tiempo promedio en milisegundos de todas las muestras
- Mediana: tiempo de la mediana en milisegundos de todas las muestras
- 90% Line: es el valor por debajo del cual 90% de las muestras fallan.
- 95% Line: es el valor por debajo del cual 95% de las muestras fallan.
- 99% Line: es el valor por debajo del cual 99% de las muestras fallan.
- Mín: tiempo mínimo en milisegundos de todas las muestras
- Máx: tiempo máximo en milisegundos de todas las muestras
- % Error: número porcentual de peticiones fallidas
- Rendimiento: número de peticiones atendidas por unidad de tiempo.

NOTA: este valor nos soportará las pruebas de rendimiento

Kb/sec: tamaño en kb por segundo de las peticiones

Se pueden almacenar los resultados en archivos xml o csv para posterior análisis detallado.

1.21.5.1 Cálculo del percentile 90%

El percentile 90 es el valor para el tiempo de respuesta para el cual el 90% de los puntos de datos son más pequeños y 10% son más grandes.

Para calcu lar el percentile 90:

1. ordene las peticiones por valor del tiempo de respuesta desde el mejor tiempo al peor
2. quite el 10% de las instancias con tiempos mayores
3. el valor más grande es el 90 percentil

Ejemplo:

Diez tiempos de respuesta:

10Sec,5Sec,3Sec,9Sec,4Sec,3Sec,1Sec,7Sec,2Sec

Ordenados de menor a mayor

1 sec,2 sec,3 sec,4 sec,5 sec,6 sec,7 sec,8 sec,9 sec,10 sec

Removemos el 10 segundos (tiempos mayores)

1 sec,2 sec,3 sec,4 sec,5 sec,6 sec,7 sec,8 sec,9 sec

“9 sec” es el valor 90th percentile.

1.22 PRUEBAS DE CARGA (LOAD)

Las pruebas de carga son un tipo de prueba de medición de eficiencia conducidas para evaluar el comportamiento de los determinados casos de prueba. Se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. Esta prueba puede mostrar los tiempos de respuesta de todas las transacciones importantes de la aplicación [5].

1.23 PRUEBAS DE ESTRES (STRESS) Y PRUEBAS DE RENDIMIENTO (PERFORMANCE)

Es un tipo de prueba que evalúa el comportamiento de un determinado elemento cuando la carga se pone superior o inferior a lo anticipado o a lo asumido por los requerimientos [5].

- Pruebas de estrés: Esta prueba se utiliza normalmente para romper la aplicación. Se va doblando el número de usuarios que se agregan a la aplicación y se ejecuta una prueba de carga hasta que se rompe. Este tipo de prueba se realiza para determinar la solidez de la aplicación en los momentos de carga extrema y ayuda a los administradores para determinar si la aplicación rendirá lo suficiente en caso de que la carga real supere a la carga esperada.
- De rendimiento: estas pruebas se realizan para medir la respuesta de la aplicación a distintos volúmenes de carga esperados (cantidad de usuarios y/o peticiones). Ejemplo: velocidad de respuesta al procesar el ingreso de 10, 100 y 1000 usuarios en forma simultánea.

REFERENCIAS

[1] CDC Unified Process, CDC UP, CDCUP, «Test Plan Template», 01-oct-2010. [En línea]. Disponible en: www2a.cdc.gov/cdcup/library/templates/CDC_UP_Test_Plan_Template.doc. [Accedido: 04-oct-2016].

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

[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 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.

[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 2, Part 2,*. 2013.

[5] 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.

[6] *ISO*. ///.

[7] «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].

[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] «Apache JMeter - Apache JMeter™». [En línea]. Disponible en: http://jmeter.apache.org/index.html. [Accedido: 17-oct-2016].

[11] «Apache JMeter - User's Manual: Building a SOAP WebService Test Plan». [En línea]. Disponible en: http://jmeter.apache.org/usermanual/build-ws-test-plan.html. [Accedido: 17-oct-2016].