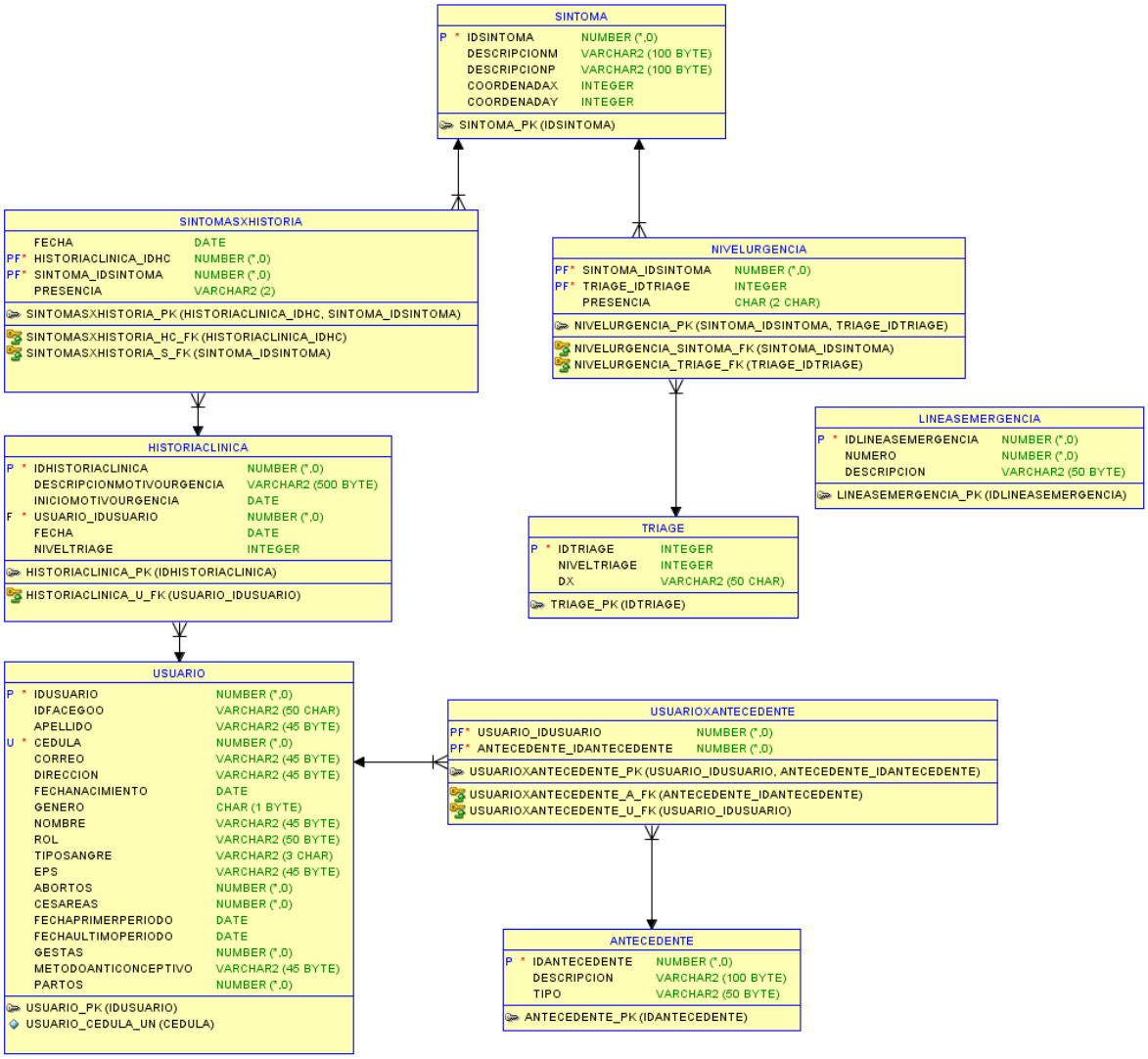
Manual de uso para consumir servicios del servidor (Web Service)	
Elaborado por	Luisa Álvarez Valencia
Objetivo	Este documento tiene como fin listar y explicar los métodos que posee el servidor en su lógica de negocio. De esta manera un cliente (consumidor, proxy) puede hacer uso de dichos servicios.
Información general	IP: 200.58.126.15 Puerto: 4848, accede a Glassfish en modo administrador. Puerto: 8080, accede a los servicios del servidor.
Contenido	

1. Base de datos

A continuación se muestra un diagrama de base de datos de modo que se conozca el modelo de dominio que utiliza la aplicación. El nombre de dichas entidades es utilizado para acceder a los servicios del servidor, es por ello que es relevante.

Adicionalmente la información de la base de datos se lista a continuación:

- serverName: 200.58.126.15
- portNumber: 1527
- databaseName: triageDB
- User: usertappi
- Password: usertappi
- URL: jdbc:derby://200.58.126.15:1527/triageDB
- driverClass: org.apache.derby.jdbc.ClientDriver
- jndi-name: app/dstappiderby
- pool-name: derby_net_triageDB_usertappiPool



2. Métodos y servidor

El acceso a los servicios del servidor se logra por medio de una IP pública y una URL con las siguientes características:

http://200.58.126.15:8080/WSTappi/webresources/entities.*

El * representa cualquiera de las tablas de la base de datos

Método	Acceso	Función
Post		Crea una entidad del tipo que se elija.
Put	En su path recibe un id.	Edita una entidad del tipo que se elija.
Delete	En su path recibe un id.	Borra una entidad del tipo que se elija.
Get	En su path recibe un id. webresources/entities.*/1	Obtiene la entidad que tenga la llave primaria que se escriba. En el ejemplo de la derecha se digito /1, para buscar la entidad con ese identificador.

Get	webresources/entities.*	Obtiene todas las tuplas que la entidad tenga ingresadas.
Get	webresources/entities.*/1/3	Obtiene las tuplas dentro de un rango de identificadores. En ejemplo buscaría las tuplas que estén entre 1 y 3,
Get	webresources/entities.*/count	Cuenta la cantidad de tuplas que tiene la entidad ingresadas

Con ese wadl un cliente puede saber qué hace el servicio rest (equivalente más o menos a la tabla de arriba)

http://200.58.126.15:8080/WSTappi/webresources/application.wadl

3. POST y GET por interfaz grafica

Otro link que también se puede utilizar permite hacer GET y POST por medio de xml, este más que todo se utiliza con el fin de que el desarrollador móvil conozca como los datos deben extraerse de las tablas. La URL es:

http://200.58.126.15:8080/PruebaVer/test-resbeans.html

4. Crear, borrar, editar tuplas a las tablas

Bebido a que la base de datos que se utiliza está alojada en el servidor esta se puede acceder desde la ip pública a través de la siguiente ruta con el fin de que se pueda tener una interfaz gráfica para su actualización y manejo. La siguiente URL permite la administración del contenido de la base de datos:

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

5. Lista de funcionalidades servidor

5.1. Resumen

Tener en cuenta que la URL debe contener la ip y el puerto, seguido de lo que indique la tabla:
http://200.58.126.15:8080

Funcionalidad	Característica	Servicio	Llamado – URL- Metodo
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	GET, LoginFacadeREST. A partir de un id que se ingresa por la URL busca al usuario.	/WSTappi/webresources/login/123 Siendo 123 el id de Facebook o Google.
Ver historial de historias clinicas	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.	GET, ListaHistoriaClinicaXUsuarioFacadeREST. A partir de un identificador un usuario, esta fachada devuelve las historias clínicas que tiene ese paciente.	/WSTappi/webresources/listahistoriaclinicaxusuario/1 Siendo 1 el id de usuario
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,	GET, HistoriaclinicaFacadeREST Dado un id se obtiene la historia clínica en XML.	/WSTappi/webresources/entities.historiaclinica/1 Siendo 1 el id de la historia clinica

	este elige una; se retorna un id y con ese id se busca la HC correspondiente.		
Crear Historia clinica	Se crea la historia clínica a la cual asociar las dolencias del paciente cuando este ingrese sus síntomas.	POST, HistoriaclinicaFacadeREST. A partir de la URL se realiza un post a esta entidad	/WSTappi/webresources/entities.historiaclinica Esta URL da acceso al método Un ejemplo del uso de POST se encuentra en la URL inferior, la lógica para consumir del servidor está en JavaScript (ver archivo fuente github). Este permite crear un usuario a partir de un XML. /PruebaVer/test-resbeans.html
Modificar historia clínica (HC)	Permite modificar la historia clínica en caso de que los datos de ella se hayan ingresado incorrectamente.	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.	/WSTappi/webresources/entities.historiaclinica Esta URL da acceso al método Se envía lo mismo que el POST pero por PUT
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 HC se identifican los síntomas que esta tiene.	/WSTappi/webresources/listasintomasxhistoriaclinica/1 Siendo 1 el id de la historia clínica
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.	PUT, SintomasxhistoriaFacadeREST. Utiliza una llave primaria compuesta y por lo tanto los	/WSTappi/webresources/entities.sintomasxhistoria/somePath;historiaclinicaIdhc=1;sintomaIdsintoma=1

		campos que permiten edición son presencia y fecha. Se debe recalcular el nivel del triage por la edición de los síntomas.	A partir de un id de un usuario y un id sintoma (1) permite editar la información de síntomas de la historia clínica del usuario, usando el método PUT.
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.	DELETE, SintomasxhistoriaFacadeREST. Utiliza una llave compuesta para lograr borrar los datos de esa tabla	/WSTappi/webresources/entities.sintomasxhistoria/somePath;historiaclinicaIdhc=1;sintomaIdsintoma=1 A partir de un id de un usuario y un id síntoma (1) permite eliminar determinado síntomas de la historia clínica del usuario, usando el método DELETE.
Ingresar Síntomas	El usuario ingresa los síntomas de su dolencia y la lógica de negocio debe dar un nivel de prioridad tentativo (Triage).		
	Listado de síntomas	GET, SintomasFacadeREST, es una fachada de la entidad que contiene todos los síntomas. Trae todos los síntomas que el usuario pueda presentar.	/WSTappi/webresources/entities.sintoma Los síntomas se encuentran en lenguaje médico y en lenguaje coloquial.
	Vincular síntoma a usuario solicitando si presenta o no el síntoma	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.	/WSTappi/webresources/entities.sintomasxhistoria/ El ingreso de dolencias por parte del paciente se lleva a cabo a través de PUT, SintomasxhistoriaFacadeREST.
	Cálculo del triage	El cálculo del Triage hace parte de la lógica de negocio.	/WSTappi/webresources/calculartriage/1

		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.	Siendo el 1 el id de historiaclinica del cual se desea calcular el Triage. El servidor da como valor de 0 (cero) el Triage cuando este no encuentra una similitud en las secuencia de reglas.
Editar Usuario	Permite editar la información de un usuario	PUT, UsuarioFacadeREST Edita un usuario dando como entrada los datos de un usuario en XML.	/WSTappi/webresources/entities.usuario Esta URL da acceso al método Se envía lo mismo que el POST pero por PUT
Crear Usuario Ingresar datos ginecológicos	Permite crear un usuario a partir de información suministrada.	POST, UsuarioFacadeREST Crea un usuario dando como entrada los datos de un usuario en XML.	/WSTappi/webresources/entities.usuario Esta URL da acceso al método Un ejemplo del uso de POST se encuentra en la URL inferior, la lógica para consumir del servidor está en JavaScript (ver archivo fuente github). Este permite crear un usuario a partir de un XML. /PruebaVer/test-resbeans.html
Eliminar Usuario	Se elimina un usuario de la aplicación y la información queda almacena por si es	DELETE, UsuarioFacadeREST Elimina un usuario dado un id como entrada los datos de un usuario en XML.	/WSTappi/webresources/entities.usuario

	necesaria en un archivo de persistencia.		
Listar antecedentes de usuario	Lista los antecedentes de un usuario dado su id.	ListarantecedentexusuarioFacade REST. Realiza un find por una query con id de usuario.	/WSTappi/webresources/Listarantecedentexusuario/1 Siendo 1 el id de usuario
Eliminar antecedentes de usuario	Elimina un antecedente de un usuario.	UsuarioxantecedenteFacadeREST. Este utiliza una llave primaria compuesta de: idantecedente y idusuario.	/WSTappi/webresources/entities.usuarioxantecedente/somepath;usuarioIdusuario=1;antecedenteIdantecedente=1 A partir de un id de un usuario y un id antecedente (1) permite eliminar la información del antecedente del usuario, usando el método DELETE
Modificar antecedentes de usuario	Permite modificar la información del usuario	UsuarioxantecedenteFacadeREST. Este utiliza una llave primaria compuesta de: idantecedente y idusuario.	/WSTappi/webresources/entities.usuarioxantecedente/somepath;usuarioIdusuario=1;antecedenteIdantecedente=1 A partir de un id de un usuario y un id antecedente (1) permite modificar la información del antecedente del usuario, usando el método PUT
Ingresar antecedentes familiares	La lógica de negocio permite ingresar antecedentes familiares de un usuario determinado.	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.	WSTappi/webresources/entities.antecedente Permite listar todos los antecedentes /WSTappi/webresources/entities.usuarioxantecedente/somepath;usuarioIdusuario=1;antecedenteIdantecedente=1 A partir de un id de un usuario y un id antecedente (1) permite ingresar la información del antecedente del usuario, usando el método POST. Lo único que se requiere cambiar es el id de antecedente o de usuario.

Contactar con líneas de emergencia	El servidor expone la información de líneas de emergencia	GET, LineasemergenciaFacadeREST. Permite obtener todas las líneas de emergencia que posea la tabla insertada.	/WSTappi/webresources/entities.lineasemergencia Este GET obtiene por XML todas las líneas de emergencia disponibles de la aplicación por medio de REST.
Administrar cuentas	Restringir el uso de las funcionalidades a ciertos usuarios	Este utiliza diferentes métodos dictados en la parte superior, como es eliminar un usuario.	

5.2. Explicación adicional Login

El inicio de sesión de la aplicación se lleva a cabo a través de Google + o Facebook. Al iniciar sesión en cualquiera de estas dos cuentas estas devuelven un id para identificar el usuario, dicho id está almacenado en la tabla *USUARIO.IDFACEGOO*. Con el fin de que se pueda llevar a cabo un login de un usuario que YA EXISTA se posee una fachada (LoginFacadeRest) que coge el idfacegoo y retorna un usuario (sino retorna null), de esa manera se tiene la información completa de este.

http://200.58.126.15:8080/WSTappi/webresources/login/123

Siendo 123 el *USUARIO.IDFACEGOO*

6. Pasos al reiniciar el servidor

- Ingresar por SSH al servidor:
  ssh -p5099 root@200.58.126.15
- Verficar que hostname esté en localhost:
  hostname localhost
- Iniciar Glassfish
  /home/glassfish/bin/asadmin start-domain domain1
- Iniciar Derby
  /home/glassfish/bin/asadmin start-database

7. Archivos fuentes

El archivo fuente se encuentra en http://github.com/alvarezluisa/serv