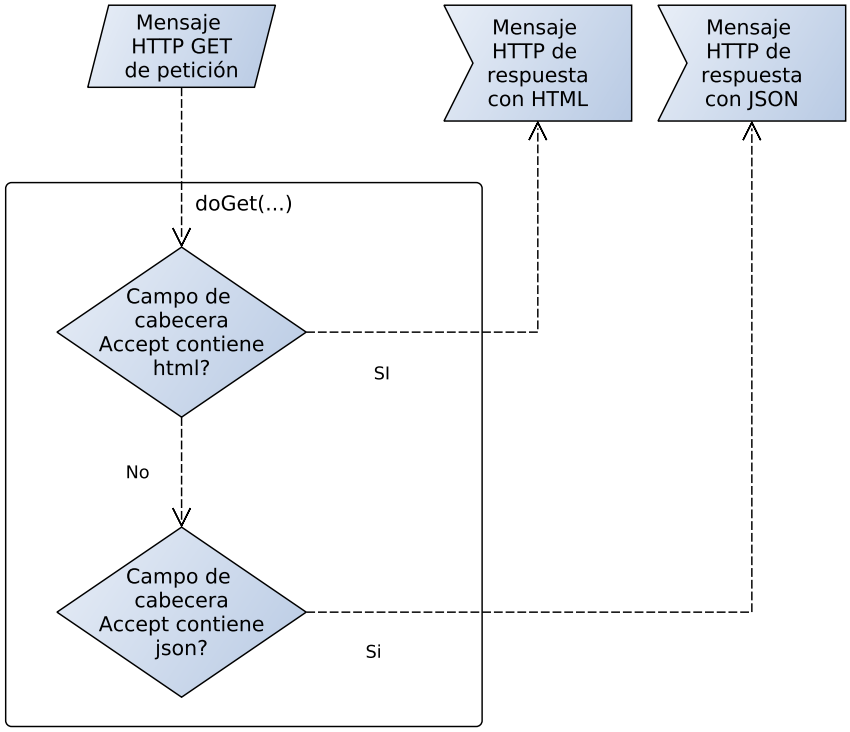
Ejercicio 1

Desarrollar un Servlet que responda a peticiones `GET` y `PUT`. El Servlet debe cargar los datos del fichero `resultados_mesas.txt` [1] y crear objetos del tipo `ResultadoMesa` con cada una de las líneas del fichero. Estos objetos se deben almacenar en un `ArrayList<ResultadoMesa>`. Esta tarea se debe realizar en el método `init()` del `Servlet`.

En las peticiones `GET` se debe devolver el contenido del `ArrayList` pero en dos posibles formatos: en texto y el JSON. El tipo de formato devuelto dependerá del campo de cabecera `Accept` de la petición: si contiene `html` devolveremos HTML pero si contiene `json` entonces devolveremos JSON.



En la petición `PUT` asumimos que en el cuerpo de la petición se pasa un objeto en formato JSON con el resultado de una mesa electoral. Con esta información se crea un objeto del tipo `ResultadoMesa` y se añade al `ArrayList<ResultadoMesa>` de forma que en las siguientes peticiones `GET` ese objeto añadido aparezca listado.

Para realizar la conversión objeto Java a JSON y de JSON a objeto Java se pueden usar las clases que proporciona la librería GSON: http://search.maven.org/remotecontent?filepath=com/google/code/gson/gson/2.3.1/gson-2.3.1.jar. El fichero jar se debe colocar en el proyecto en el directorio `WEB-INF/lib` y se debe configurar el `Build Path` de el proyecto para que las clases que contiene se puedan usar.

El siguiente código muestra cómo convertir un objeto Java a JSON:

```
// Supongo que gson es una referencia a un objeto del tipo Gson
// y que lista es una referencia a un ArrayList<ResultadoMesa>
String json = gson.toJson(lista);
```

El siguiente código muestra cómo convertir de JSON a un objeto Java:

[1] Este fichero ha sido obtenido de
http://datos.gob.es/catalogo/resultados-elecciones-generales-2011-mesa-electoral-11

```
// Supongo que gson es una referencia a un objeto del tipo Gson
// y que r es una referencia a un Reader
ResultadoMesa rm = gson.fromJson(r, ResultadoMesa.class);
```

Para probar las diferentes peticiones se puede usar `curl` que permite realizar peticiones HTTP desde un terminal. A continuación se presentan varios ejemplos de peticiones:

Ejemplo de petición `GET` especificando el campo de cabecera `Accept: text/html`:

```
curl -k -L -s -X GET -H "Accept: text/html" http://localhost:8080/ResultadosMesas/Resultados
```

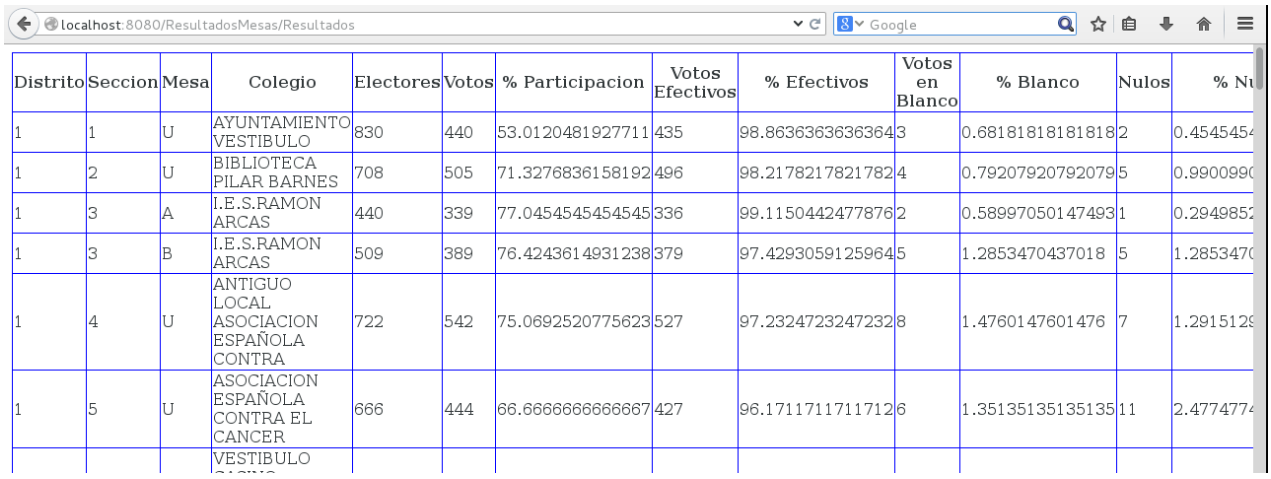
Ejemplo de petición `GET` especificando el campo de cabecera `Accept: application/json`:

```
curl -k -L -s -X GET -H "Accept: application/json" http://localhost:8080/ResultadosMesas/
    Resultados
```

Ejemplo de petición `PUT` especificando el campo de cabecera `Content-Type` y pasando en el cuerpo del mensaje un objeto `JSON`:

```
curl -k -L -s -X PUT -H "Content-Type: application/json" -d '{"votosEfectivos":445,"
    porcParticipacion":72.258064516129,"colegio":"C.P. SAN FRANCISCO","porcNulos
    ":0.22321428571429,"mesa":"U","votosNulos":1,"distrito":2,"electores":620,"porcBlancos
    ":0.44642857142857,"votosBlancos":2,"votosEfectuados":448,"seccion":27,"porcEfectivos
    ":99.3303571428571}' http://localhost:8080/ResultadosMesas/Resultados
```

La siguiente figura muestra un posible ejemplo de salida HTML producida por la aplicación Web:



Distrito	Seccion	Mesa	Colegio	Electores	Votos	% Participacion	Votos Efectivos	% Efectivos	Votos en Blanco	% Blanco	Nulos	% Nu
1	1	U	AYUNTAMIENTO VESTIBULO	830	440	53.0120481927711	435	98.86363636363643	3	0.681818181818182	2	0.4545454
1	2	U	BIBLIOTECA PILAR BARNES	708	505	71.3276836158192	496	98.21782178217824	4	0.792079207920795	5	0.990099
1	3	A	I.E.S.RAMON ARCAS	440	339	77.045454545454	336	99.11504424778762	2	0.589970501474931	1	0.2949852
1	3	B	I.E.S.RAMON ARCAS	509	389	76.4243614931238	379	97.42930591259645	5	1.2853470437018	5	1.285347
1	4	U	ANTIGUO LOCAL ASOCIACION ESPAÑOLA CONTRA	722	542	75.0692520775623	527	97.23247232472328	8	1.4760147601476	7	1.291512
1	5	U	ASOCIACION ESPAÑOLA CONTRA EL CANCER	666	444	66.6666666666674	427	96.17117117117126	6	1.3513513513513511	11	2.4774774
			VESTIBULO									

Para el correcto funcionamiento de la aplicación es posible que se deban implementar bloques `synchronized` para conseguir la exclusión mutua.

Ejercicio 2

Este ejercicio es una ampliación del anterior y es **opcional**. Se pide modificar el código del método `doPut(.,.)` para que el cliente deba enviar un campo de cabecera adicional con un token que servirá para comprobar si puede realizar la acción. El campo de cabecera debe ser `Authorization`. Si el valor enviado en este campo de cabecera no coincide con lo esperado por el servidor el cuerpo del mensaje no se lee.

Para comprobar el correcto funcionamiento se deberá enviar ese campo de cabecera en la petición añadiendo `-H "Authorization: 0b79bab50daca910b000d4f1a2b675d604257e42"` (el valor proporcionado es un posible ejemplo) a la petición del ejercicio anterior. Si el valor enviado no coincide con el valor almacenado en el servidor entonces el cuerpo del mensaje no se lee y no se añade al `ArrayList<ResultadoMesa>`.

Tal y como ya comenté en clase, esto se debería hacer vía HTTPS ya que en caso contrario alguien malintencionado podría obtener ese token y suplantar la identidad.

Se debe subir a Aula Virtual un fichero zip que contenga el proyecto exportado desde Eclipse.

Entrega de la solución: