Trabajo

Fundamentos de Aplicaciones y Servicios Telemáticos

2º Curso Grado en Ingeniería de Tecnologías de Telecomunicación

Departamento de Ingeniería Telemática (DIT)

Universidad de Sevilla

Contenido

1 Introducción 1

2 Creación y configuración de la aplicación y del servidor 1

2.1 Crear aplicación web (AppTrabajo) 1

2.2 Añadir aplicación al servidor (Tomcat) 3

2.3 Modificar web.xml de la aplicación (AppTrabajo) 4

2.4 Modificar server.xml del servidor (Tomcat) 5

2.5 Copiar postgresql-42.0.0.jar 5

2.6 Descargar y extraer los ficheros iniciales 6

2.7 Ficheros y estructura de directorios 6

3 Base de datos 8

3.1 Tablas 8

3.2 Sistema gestor de base de datos 8

3.3 Creación de las tablas 8

4 Ficheros 11

4.1 Ficheros correspondientes al bloque I 11

4.1.1 Ficheros de admin 11

4.1.2 Fichero de estilos 12

4.1.3 Fichero de datos JSON 12

4.2 Resto de ficheros 12

4.2.1 Ficheros jsp y relacionados 12

4.2.2 Ficheros java 12

4.2.3 Ficheros CGI 12

5 Listener 13

5.1 AppListener.java 13

6 Filtros 13

6.1 FiltroMenu.java 13

6.2 FiltroAdmin.java 14

6.3 FiltroClientes.java 14

7 JavaBean 14
7.1 Usuario.java 14
8 Páginas jsp 14
8.1 Páginas en WebContent 15
8.1.1 Página de cabecera: cabecera.jsp 15
8.1.2 Página de pie: pie.jsp 15
8.1.3 Página de inicio: index.jsp 16
8.2 Páginas en WebContent/clientes 18
8.2.1 Página de “Menú clientes”: menu.jsp 18
8.2.2 Página de “Ver equipos” de clientes: verEquipos.jsp 18
8.2.3 Página de “Ver interfaces sin asignar” de clientes: verIntSinAsignar.jsp 19
8.2.4 Página de “Ver/crear interfaces” de clientes: verCrearInterfaces.jsp 19
8.2.5 Página de “Error”: error.jsp 20
8.3 Páginas en WebContent/admin 20
8.3.1 Página de “Menú administrador”: menu.jsp 20
8.3.2 Página de “Ver usuarios”: verUsuarios.jsp 20
8.3.3 Página de “Crear/modificar usuarios”: usuForm.jsp 20
8.3.4 Página de “Ver/crear interfaces (de todos los usuarios)” de administrador: verCrearInterfaces.jsp 21
8.3.5 Página de “Ver equipos (de todos los usuarios)” de administrador: verEquipos.jsp 21
8.3.6 Página de “Asignar direcciones IP”: asigIpIntSinAsignar.jsp 21
8.3.7 Página ipForm.jsp 21
9 Servlets 21
9.1 ServletCerrar.java 21
9.2 ServletIntSinAsignar.java 22
9.3 ServletInterfaces.java 22
9.4 ServletEquipos.java 22
9.5 ServletUsuarios.java 23
9.6 ServletNuevaInterfaz.java 23
9.7 ServletModificarUsuario.java 23
9.8 ServletModificarIp.java 24
9.9 ServletMostrarRedes.java 24

10 Ficheros java adicionales 25
10.1 OperacionesIP 25
11 Entrega 25
12 Anexo 25
12.1 Includes 25
12.2 Filtros 25
12.3 Reenvío y redireciones 25
12.4 URLs en navegador 26
12.5 Depuración 26

1 Introducción

El trabajo consiste en la realización de una aplicación web que ayude a la gestión de las direcciones IP de una serie de interfaces de red. Estas interfaces de red están dentro de un equipo, cada equipo pertenece a un usuario, y en la aplicación hay múltiples usuarios.

Cada interfaz estará identificada por un identificador numérico (0, 1, 2, etc.) dentro de un equipo, y será de un tipo.

Cada equipo estará identificado por su nombre (cadena de caracteres), y tiene asociado un usuario (dueño del equipo).

Cada interfaz puede tener varias direcciones IP, pero una dirección IP sólo puede estar asociada a una interfaz.

Los usuarios necesitan introducir su usuario y contraseña para poder acceder, y hay varios tipos de usuarios:

- clientes: pueden ver el listado de sus equipos y direcciones. Pueden crear nuevas interfaces para sus equipos
- administrador: además ver el listado de todos los equipos y direcciones, puede modificar la configuración de todas las direcciones IP de las interfaces y crear/modificar clientes (crear clientes nuevos y asignarle contraseña, o modificar la contraseña de los clientes existentes).

Los usuarios se identifican por su nombre de usuario. Cada usuario (puede ser administrador o no) tiene asociada una contraseña.

2 Creación y configuración de la aplicación y del servidor

Se debe crear una aplicación web, y para ello se deben seguir los siguientes pasos:

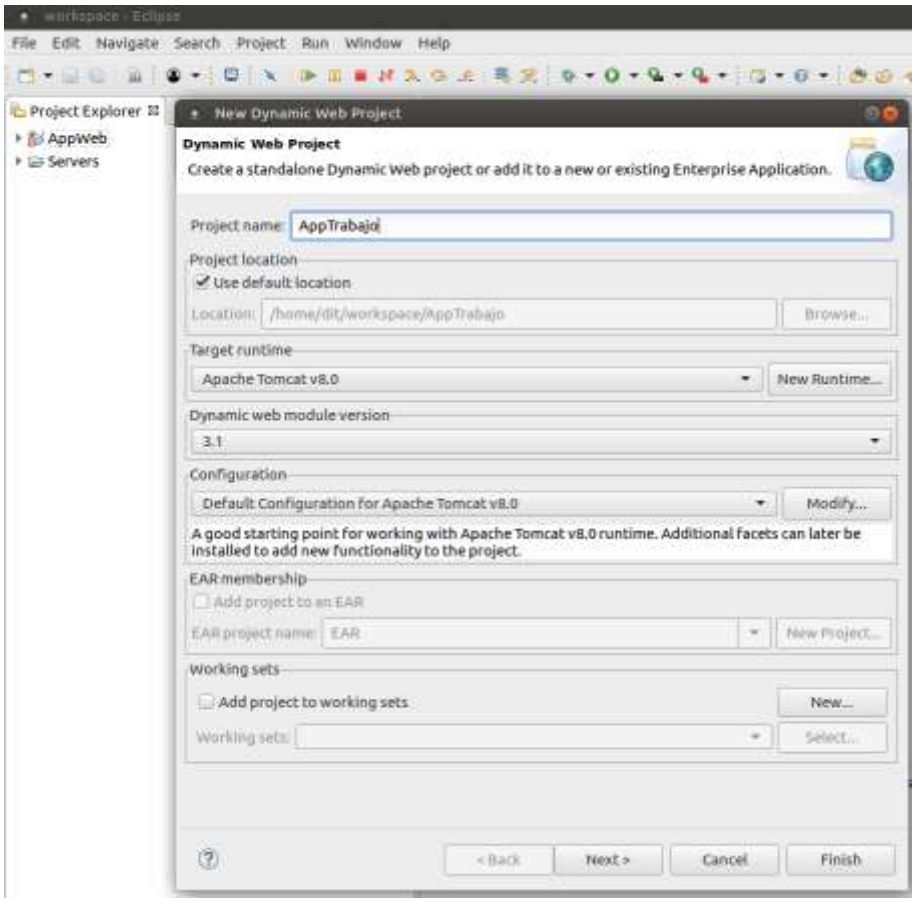
- Crear aplicación web (AppTrabajo)
- Añadir aplicación al servidor (Tomcat)
- Modificar web.xml de la aplicación (AppTrabajo)
- Modificar server.xml del servidor (Tomcat)
- Copiar postgresql-42.0.0.jar
- Descargar y extraer los ficheros iniciales

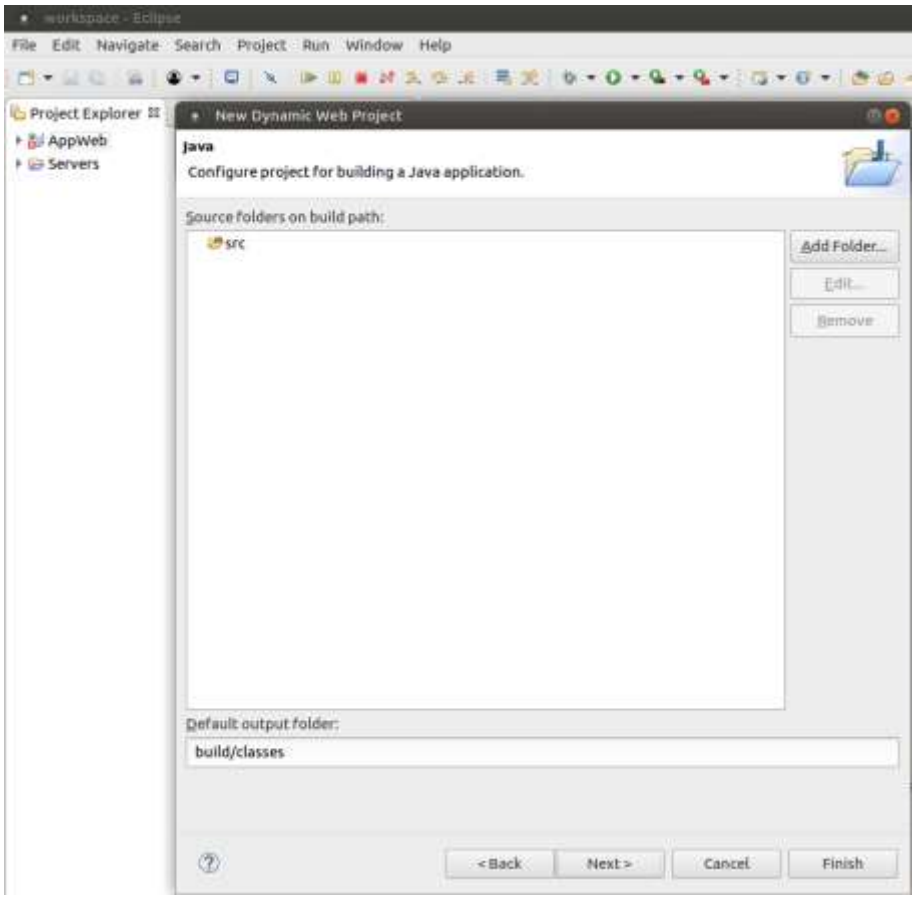
2.1 Crear aplicación web (AppTrabajo)

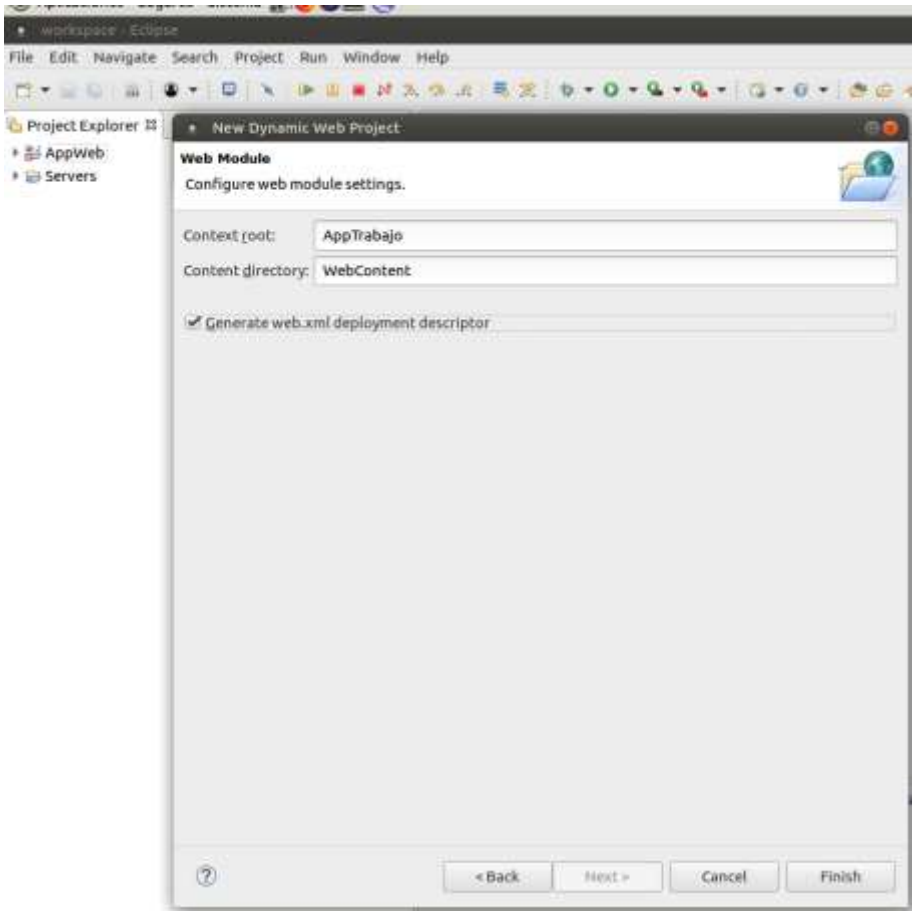
Dentro del workspace de Eclipse (`/home/dit/workspace`, tal como se explicó en la práctica P01), se debe crear una aplicación web equivalente a AppWeb y de nombre AppTrabajo. Para ello:

File > New > Dynamic Web Project

Observe las capturas a continuación:


workspace - Eclipse
File Edit Navigate Search Project Run Window Help
Project Explorer
AppWeb
Servers
New Dynamic Web Project
Dynamic Web Project
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.
Project name: AppTrabajo
Project location
Use default location
Location: /home/dit/workspace/AppTrabajo
Browse...
Target runtime
Apache Tomcat v8.0
New Runtime...
Dynamic web module version
3.1
Configuration
Default Configuration for Apache Tomcat v8.0
Modify...
A good starting point for working with Apache Tomcat v8.0 runtime. Additional facets can later be installed to add new functionality to the project.
EAR membership
Add project to an EAR
EAR project name: EAR
New Project...
Working sets
Add project to working sets
New...
Working sets:
Select...
< Back
Next >
Cancel
Finish

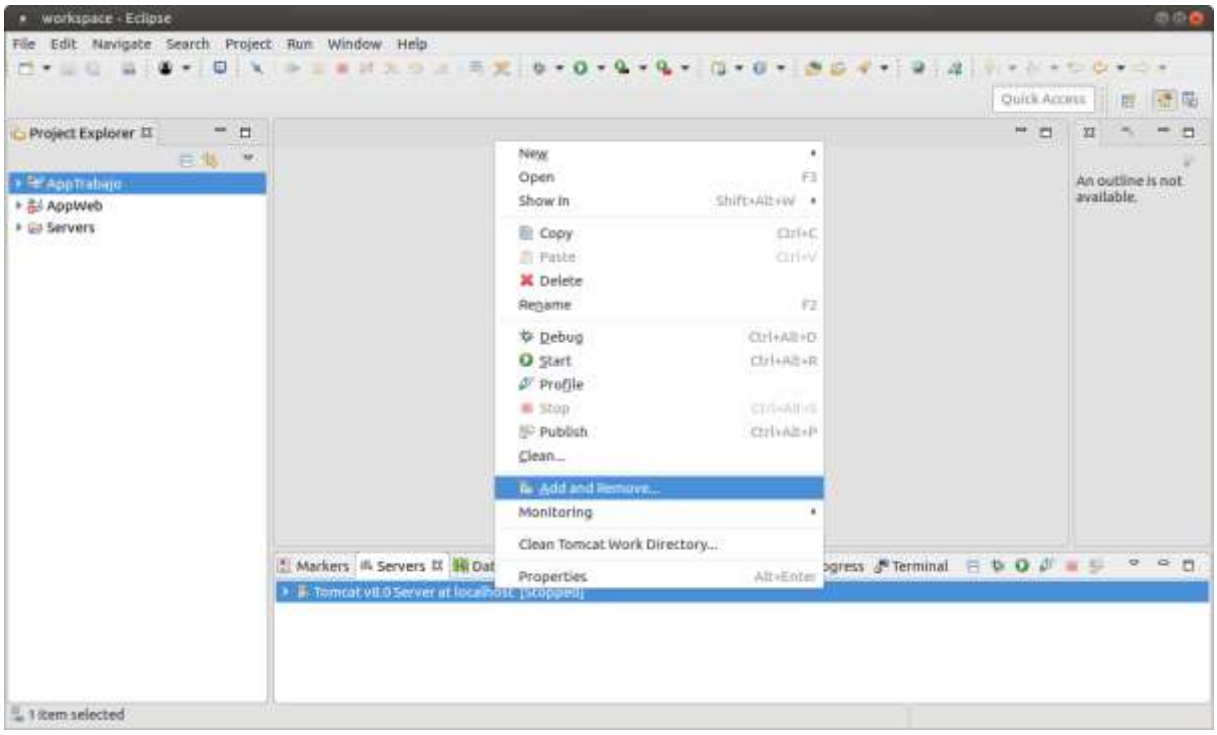

workspace - Eclipse
File Edit Navigate Search Project Run Window Help
Project Explorer
AppWeb
Servers
New Dynamic Web Project
Java
Configure project for building a Java application.
Source folders on build path:
src
Add Folder...
Edit...
Remove
Default output folder:
build/classes
< Back
Next >
Cancel
Finish

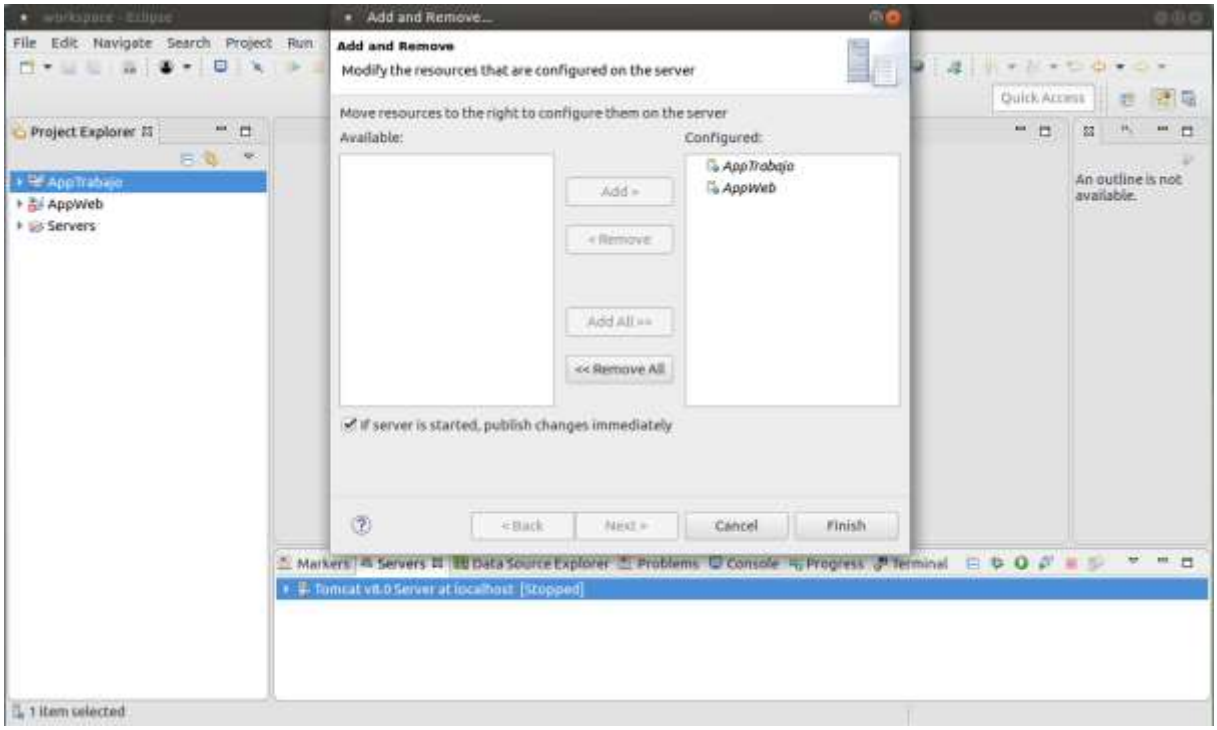


2.2 Añadir aplicación al servidor (Tomcat)

Una vez creada, se debe añadir al servidor:

Pestaña Server > Tomcat > Botón derecho > Add and Remove …

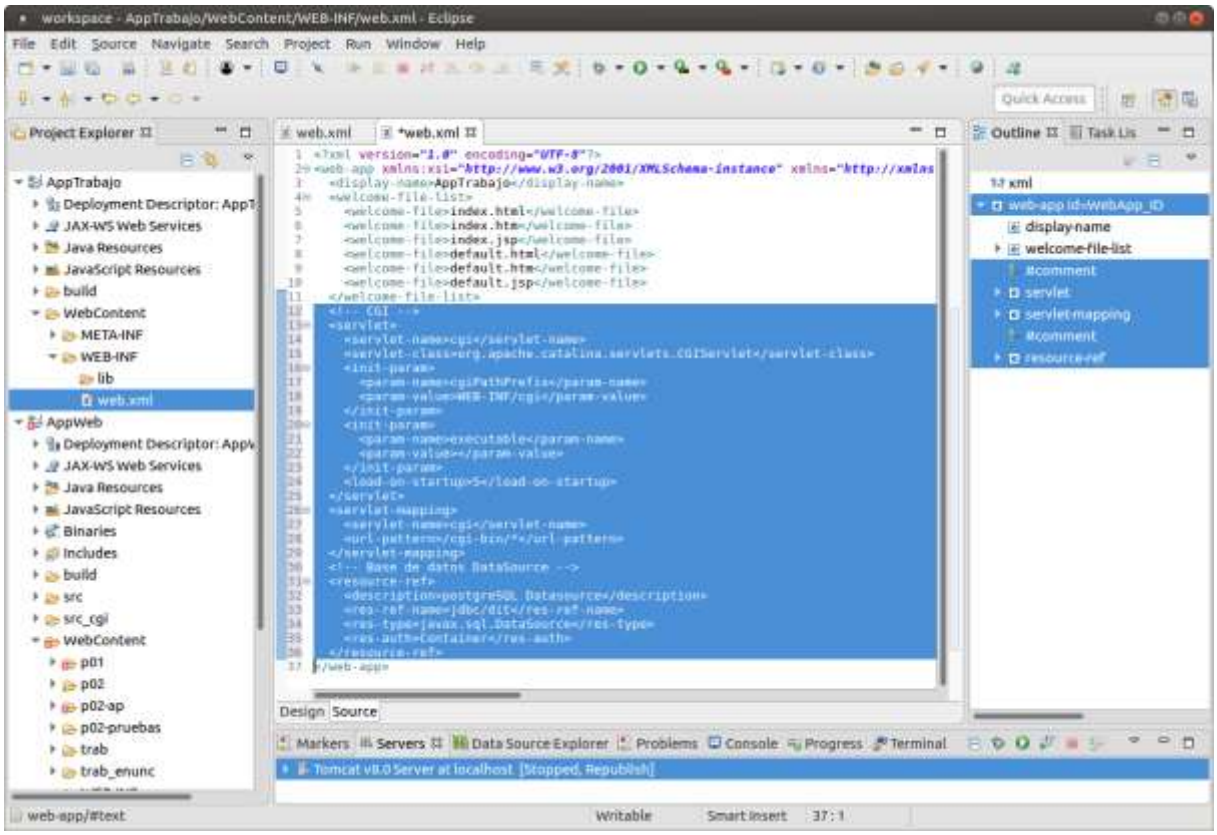




2.3 Modificar web.xml de la aplicación (AppTrabajo)

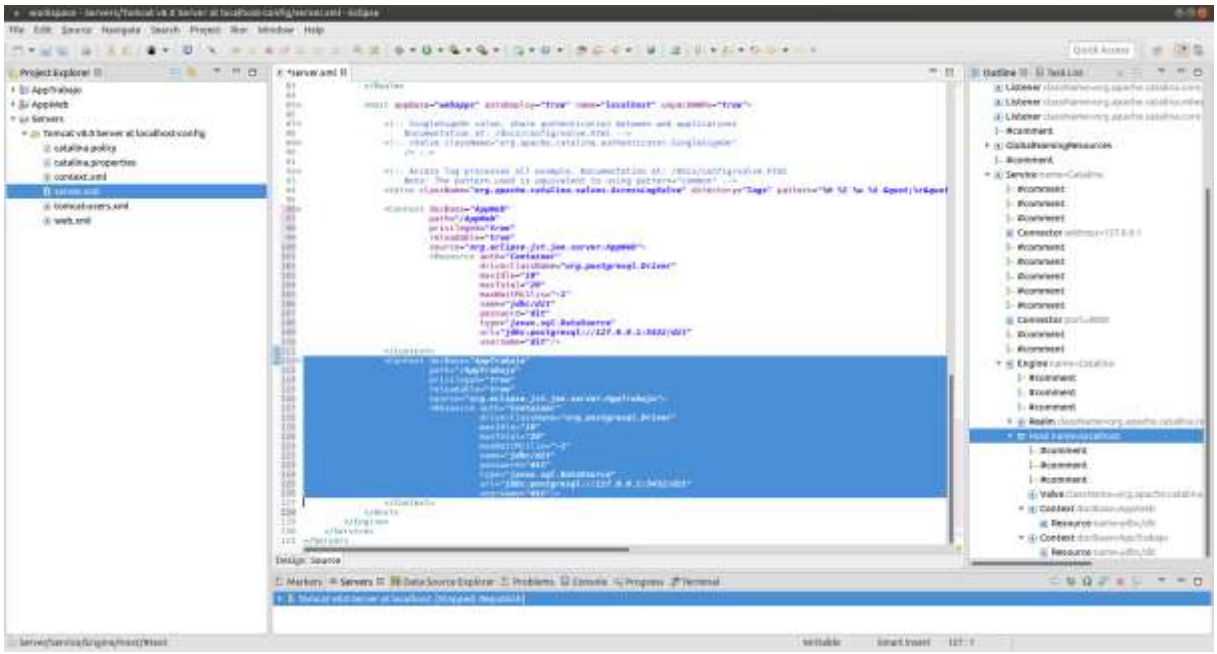
Modificar el fichero web.xml de la aplicación para que contenga la parte correspondiente a la parte de CGI y Bases de datos igual que el de AppWeb:

```
<!-- CGI -->
<servlet>
   <servlet-name>cgi</servlet-name>
…
</servlet>
<servlet-mapping>
   <servlet-name>cgi</servlet-name>
   …
</servlet-mapping>
<!-- Base de datos DataSource -->
<resource-ref>
…
</resource-ref>
```



2.4 Modificar server.xml del servidor (Tomcat)

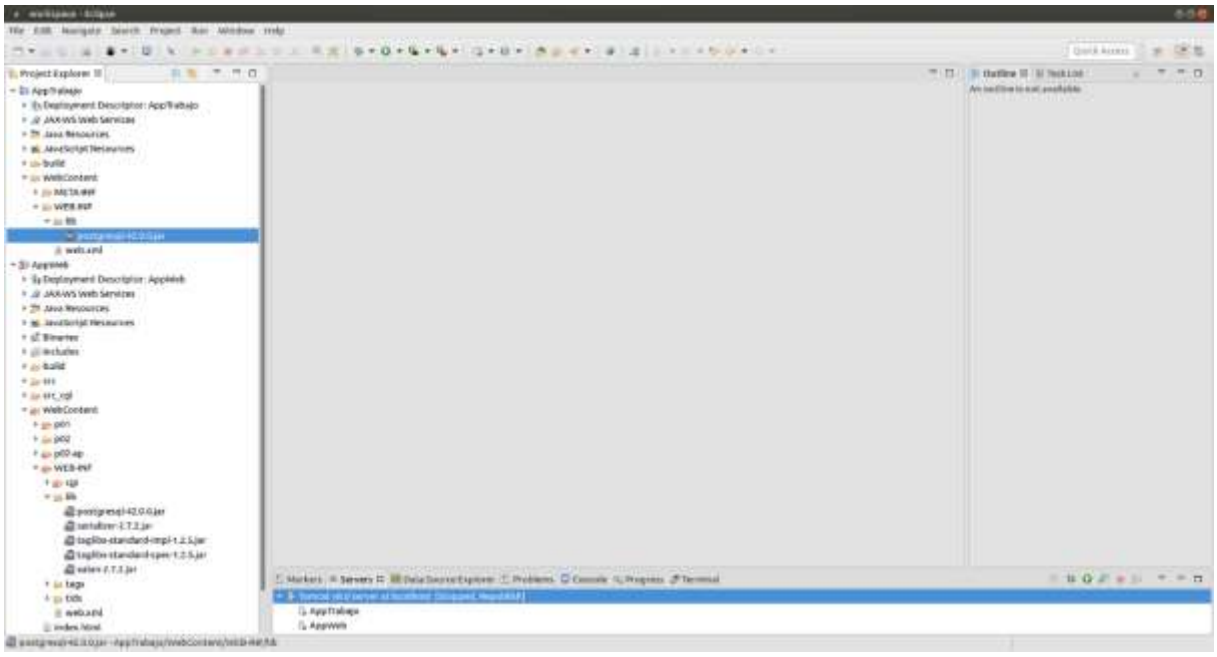
Modificar el fichero server.xml del servidor Tomcat para que el elemento <Context docBase="AppTrabajo" sea igual al elemento <Context docBase="AppWeb" (pero cambiando el nombre de la aplicación):



2.5 Copiar postgresql-42.0.0.jar

Copiar el fichero postgresql-42.0.0.jar de la biblioteca de AppWeb a la biblioteca de AppTrabajo

AppWeb/WebContent/WEB-INF/lib/postgresql-42.0.0.jar



2.6 Descargar y extraer los ficheros iniciales

Una vez creada la aplicación descargue el fichero

`inicial.tar.gz`

en /home/dit y extráigalo con tar (debe haber creado la aplicación y haber cerrado Eclipse).

```
dit@localhost:~$ tar xvfz FAST_ inicial.tar.gz
```

El fichero tar.gz contiene ficheros INICIALES para el directorio (y subdirectorios):

- workspace/AppTrabajo/

Con estos ficheros, la aplicación NO FUNCIONA, ya que algunos de los ficheros suministrados están incompletos o vacíos, y deben ser completados para que cumplan la funcionalidad especificada.

También se suministra un fichero con capturas de las apariencias de las páginas:

`capturasAppTrabajo.tar.gz`

La aplicación debe contener TODOS los ficheros que se especifican a continuación. Podría contener ficheros adicionales (siempre dentro de **`workspace/AppTrabajo`**)

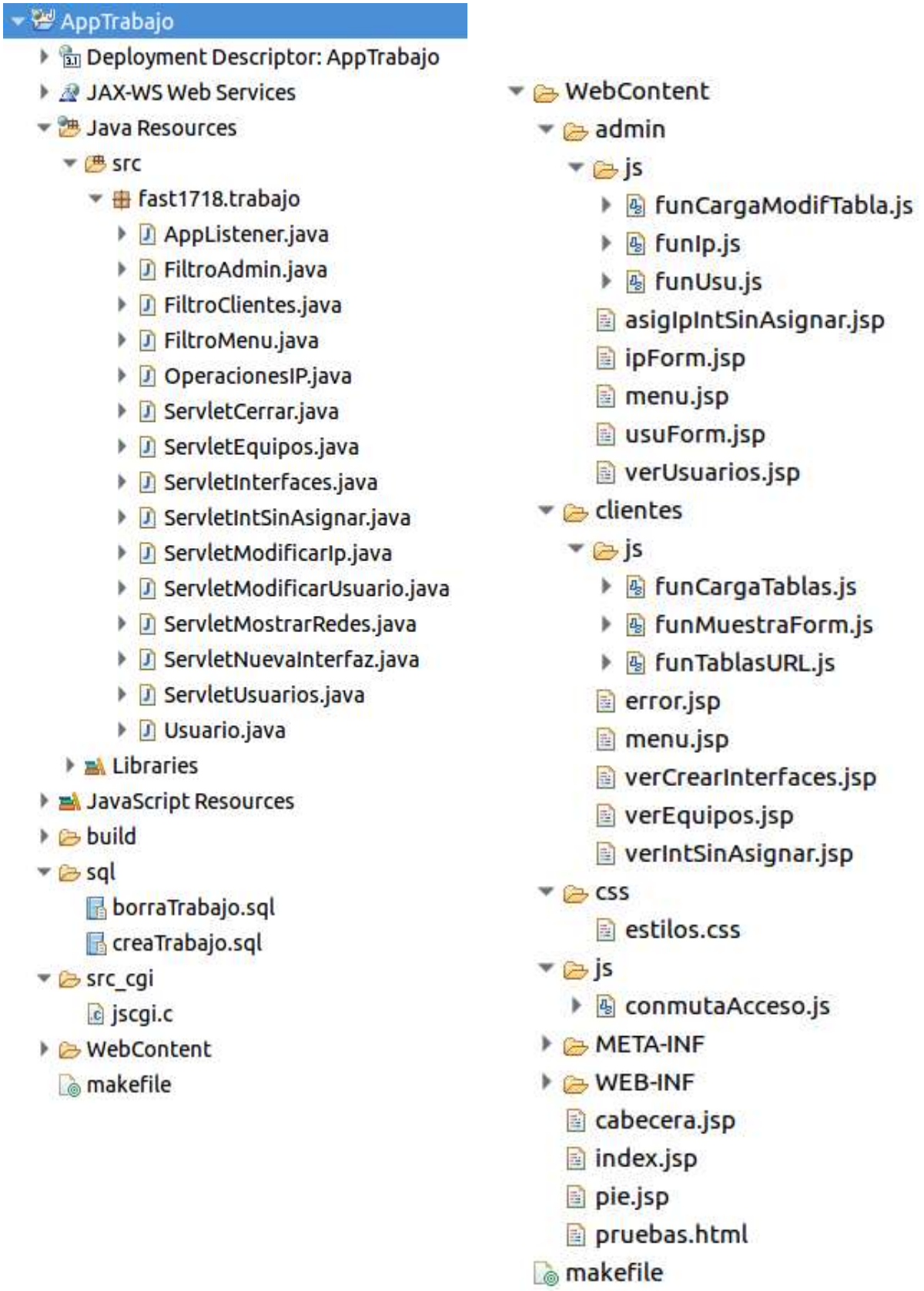
Debe completar el código suministrado, al menos donde aparecen comentarios con la palabra COMPLETAR, para que cumpla lo especificado.

En algunos casos, donde aparecen comentarios con la palabra MODIFICAR, debe modificar la línea o líneas siguientes, para que cumpla lo especificado.

Puede que COMPLETAR y MODIFICAR aparezcan precedidos de la palabra TODO (significa “por hacer”, “to do” en inglés).

2.7 Ficheros y estructura de directorios

Los ficheros y subdirectorios que debe haber en la aplicación son:


AppTrabajo
Deployment Descriptor: AppTrabajo
JAX-WS Web Services
Java Resources
src
fast1718.trabajo
AppListener.java
FiltroAdmin.java
FiltroClientes.java
FiltroMenu.java
OperacionesIP.java
ServletCerrar.java
ServletEquipos.java
ServletInterfaces.java
ServletIntSinAsignar.java
ServletModificarIp.java
ServletModificarUsuario.java
ServletMostrarRedes.java
ServletNuevaInterfaz.java
ServletUsuarios.java
Usuario.java
Libraries
JavaScript Resources
build
sql
borraTrabajo.sql
creaTrabajo.sql
src_cgi
jscgi.c
WebContent
makefile
WebContent
admin
js
funCargaModifTabla.js
funIp.js
funUsu.js
asigIpIntSinAsignar.jsp
ipForm.jsp
menu.jsp
usuForm.jsp
verUsuarios.jsp
clientes
js
funCargaTablas.js
funMuestraForm.js
funTablasURL.js
error.jsp
menu.jsp
verCrearInterfaces.jsp
verEquipos.jsp
verIntSinAsignar.jsp
css
estilos.css
js
conmutaAcceso.js
META-INF
WEB-INF
cabecera.jsp
index.jsp
pie.jsp
pruebas.html
makefile

3 Base de datos

3.1 Tablas

Se van a usar 5 tablas:

- usuarios: con los campos
  - id_usuario (clave primaria)
  - password
  - tipo_usu (valor numérico que si vale 0 indica que es administrador y si vale 1 cliente)
- equipos: con los campos
  - id_eq (clave primaria)
  - id_usuario (referencia a usuarios, indica su dueño)
- tipos_in: contiene los tipos de interfaz, con los campos
  - id_ti (clave primaria)
  - des (descripción del tipo de interfaz)
- interfaces: con los campos
  - id_eq (referencia a equipos, indica el equipo al que pertenece)
  - num_in (número de interfaz)
  - id_ti (referencia a tipos_interfaces, indica tipo de interfaz)
    Nota: (id_eq y num_in conjuntamente forman la clave primaria)
- direcciones_IP: con los campos
  - ip (clave primaria)
  - masc (máscara)
  - ip_gw (dirección IP de pasarela)
  - id_eq
  - num_in
    Nota: (id_eq y num_in referencian conjuntamente a interfaces, indicando la interfaz del equipo al que pertenece)

3.2 Sistema gestor de base de datos

Se va a utilizar como SGBD (sistema gestor de base de datos) postgresql, con las órdenes start-postgresql y stop-postgresql (con ps aux | grep postgresql puede ver si está ejecutándose):

```
dit@localhost:~$ stop-postgresql
dit@localhost:~$ ps aux | grep postgresql
dit        8860  0.0  0.0  15476   944 pts/0    S+   00:13   0:00 grep --color=auto postgresql
dit@localhost:~$ start-postgresql
dit@localhost:~$ ps aux | grep postgresql
postgres       8897    0.0    0.6  296128  24552  ?                S         00:14       0:00
/usr/lib/postgresql/9.5/bin/postgres         -D         /var/lib/postgresql/9.5/main         -c
config_file=/etc/postgresql/9.5/main/postgresql.conf
dit        8917  0.0  0.0  15476   940 pts/0    S+   00:14   0:00 grep --color=auto postgresql
```

3.3 Creación de las tablas

Una vez que se está ejecutando el SGBD, para crear las tablas e introducir algunos datos, o borrar las tablas use los siguientes ficheros (ubicados por comodidad en **~/workspace/AppTrabajo/sql**):

- borraTrabajo.sql
- creaTrabajo.sql

```
dit@localhost:~/workspace/AppTrabajo/sql$ ls *.sql
borraTrabajo.sql  creaTrabajo.sql
dit@localhost:~/workspace/AppTrabajo/sql$ cat creaTrabajo.sql
```

```
DROP TABLE IF EXISTS usuarios CASCADE;
DROP TABLE IF EXISTS equipos CASCADE;
DROP TABLE IF EXISTS tipos_in CASCADE;
DROP TABLE IF EXISTS interfaces CASCADE;
DROP TABLE IF EXISTS direcciones_IP CASCADE;

CREATE TABLE usuarios (
  id_usuario varchar(10) NOT NULL,
  password varchar(15) NOT NULL,
  tipo_usu integer NOT NULL,
  PRIMARY KEY(id_usuario)
);

CREATE TABLE equipos (
  id_eq varchar(30) NOT NULL,
  id_usuario varchar(10),
  PRIMARY KEY(id_eq),
  FOREIGN KEY(id_usuario)
    REFERENCES usuarios(id_usuario)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE tipos_in (
  id_ti integer NOT NULL,
  des varchar(15) NOT NULL,
  PRIMARY KEY(id_ti)
);

CREATE TABLE interfaces (
  id_eq varchar(30) NOT NULL,
  num_in integer NOT NULL,
  id_ti integer NOT NULL,
  PRIMARY KEY(id_eq,num_in),
  FOREIGN KEY(id_eq)
    REFERENCES equipos(id_eq)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(id_ti)
    REFERENCES tipos_in(id_ti)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE direcciones_IP (
  ip varchar(15) NOT NULL,
  masc integer NOT NULL,
  ip_gw varchar(15) NOT NULL,
  id_eq varchar(30) NOT NULL,
  num_in integer NOT NULL,
  PRIMARY KEY(ip),
  FOREIGN KEY(id_eq,num_in)
    REFERENCES interfaces(id_eq,num_in)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

COPY usuarios (id_usuario, password, tipo_usu) FROM stdin;
admin   1234aaAA        0
c1      1234aaAA        1
c2      1234aaAA        1
c3      1234aaAA        1
\.

COPY equipos (id_eq, id_usuario) FROM stdin;
e1      c1
e2      c1
e3      c2
e4      c2
e5      c3
```

```
e6      c3
e91     admin
\.

COPY tipos_in (id_ti, des) FROM stdin;
1       ethernet
2       wifi
\.

COPY interfaces (id_eq,num_in,id_ti) FROM stdin;
e1      0       1
e1      1       1
e2      0       1
e2      1       2
e2      3       2
e3      0       1
e3      1       2
e4      0       2
e4      1       2
e5      0       1
e5      1       2
e6      0       2
e6      1       2
e91     0       2
e91     1       2
\.

COPY direcciones_IP (ip, masc, ip_gw, id_eq, num_in) FROM stdin;
10.1.1.10       8       10.1.1.1        e1      0
10.1.1.11       8       10.1.1.1        e1      1
10.1.2.13       8       10.1.1.1        e2      3
10.1.3.10       8       10.1.3.1        e3      0
10.1.3.11       8       10.1.3.1        e3      1
\.

select * from usuarios;
select * from equipos;
select * from tipos_in;
select * from interfaces;
select * from direcciones_IP;
```

Para crear las tablas e introducir algunos datos:

```
dit@localhost:~/workspace/AppTrabajo/sql$ psql < creaTrabajo.sql
```

Puede comprobar el contenido de las tablas:

```
dit@localhost:~$ psql
psql (9.5.12)
Type "help" for help.

dit=> \d
             List of relations
 Schema |      Name      | Type  | Owner
--------+----------------+-------+-------
 public | direcciones_ip | table | dit
 public | equipos        | table | dit
 public | interfaces     | table | dit
 public | tipos_in       | table | dit
 public | usuarios       | table | dit
(5 rows)

dit=> select * from usuarios;
 id_usuario | password | tipo_usu
------------+----------+----------
 admin      | 1234aaAA |        0
 c1         | 1234aaAA |        1
 c2         | 1234aaAA |        1
 c3         | 1234aaAA |        1
```

```
(4 rows)

dit=> \q
dit@localhost:~$
```

Puede borrar las tablas (por si quiere empezar de nuevo):

```
dit@localhost:~/workspace/AppTrabajo/sql$ cat borraTrabajo.sql
DROP TABLE IF EXISTS usuarios CASCADE;
DROP TABLE IF EXISTS equipos CASCADE;
DROP TABLE IF EXISTS tipos_in CASCADE;
DROP TABLE IF EXISTS interfaces CASCADE;
DROP TABLE IF EXISTS direcciones_IP CASCADE;


dit@localhost:~/workspace/AppTrabajo/sql$ psql < borraTrabajo.sql
NOTICE:  drop cascades to constraint equipos_id_usuario_fkey on table equipos
DROP TABLE
NOTICE:  drop cascades to constraint interfaces_id_eq_fkey on table interfaces
DROP TABLE
NOTICE:  drop cascades to constraint interfaces_id_ti_fkey on table interfaces
DROP TABLE
NOTICE:  drop cascades to constraint direcciones_ip_id_eq_fkey on table direcciones_ip
DROP TABLE
DROP TABLE
```

4 Ficheros

4.1 Ficheros correspondientes al bloque I

4.1.1 Ficheros de admin

Deben estar en el subdirectorio admin

- ipForm.html convertido a ipForm.jsp
- usuForm.html convertido a usuForm.jsp
- js/funIp.js
- js/funUsu.js

A los ficheros jsp hay que hacerle las siguientes modificaciones:

- añadir la directiva @page (como en los demás ficheros jsp)
- referenciar al fichero estilos.css en su nueva ubicación (ver más adelante)
- añadir el fichero cabecera.jsp (ver más adelante) al comienzo de body
- eliminar el elemento footer y poner en su lugar el fichero pie.jsp (ver más adelante) al final de body
- añadir en la cabecera html: < script src="../cgi-bin/jscgi "></script>

Al fichero ipForm.jsp hay que hacerle las siguientes modificaciones:

- tenga en cuenta que esta página va a ser accedida cuando se quiere asignar una dirección ip a una interfaz de un equipo y recibe como parámetros el identificador del equipo y el número de interfaz. Por ejemplo: ipForm.jsp?id_eq=e1&num_in=0
- añadir al formulario 2 campos, para enviar también el identificador del equipo y el número de interfaz. El nombre de estos campos debe ser equipos e interfaz. El valor de estos campos debe ser igual al de los parámetros que recibe este jsp y expresado mediante EL (Expression Language). El campo de nombre equipo debe tener el valor del parámetro id_eq. El campo de

nombre interfaz debe tener el valor del parámetro num_in. Estos campos no se deben poder modificar por el usuario.
- el formulario se debe enviar a modificarIp (en vez de a muestraIp.jsp y sin extensión jsp), para que se ejecute el servlet ServletModificarIp.

Al fichero usuForm.jsp hay que hacerle las siguientes modificaciones:

- la cabecera h1 debe contener: Creación/modificación de usuario
- el formulario se debe enviar a modificarUsuario (en vez de a muestraUsu.jsp y sin extensión jsp), para que se ejecute el servlet ServletModificarUsuario.

Al fichero funIp.js hay que hacerle las siguientes modificaciones:

- respecto a la petición AJAX, hacer que:
    - sólo se envíe la petición si el valor del campo "Dirección IP" es correcto (la función procesaIp devuelve una cadena de caracteres)
    - al pulsar "Mostrar Redes" la petición sea a "mostraRedes" en vez de a "data/direcciones.json", para que se ejecute el servlet ServletMostrarRedes
    - sólo se envíe el parámetro ip
- la función pedirRedesURL() no debe cambiar el texto del botón a "Ocultar Redes" (eso se hace ahora en la función mostrarRedes()
- la función mostrarRedes() debe cambiar el texto del botón a "Ocultar Redes" si ha rellenado la tabla con al menos una fila. Debe cambiar el texto del botón a "Mostrar Redes" si no ha podido crear ninguna fila (además de cuando lo hacía anteriormente cuando se eliminaba la última fila)

4.1.2 Fichero de estilos

Debe estar en el subdirectorio css (al mismo nivel que admin y clientes):

- estilos.css

Se suministra un nuevo fichero que es igual al suministrado en la especificación del trabajo del bloque I, pero con una parte añadida al final que no se debe modificar. Por lo tanto, lo que se debe hacer es usar lo que haya hecho para la solución del bloque I y añadirla la parte central del fichero suministrado (lo que está después del comentario /***Empieza el trabajo de curso ***/ y antes del comentario /*** trabajo completo ***/)

4.1.3 Fichero de datos JSON

Ya no es necesario, ya que se usa el valor devuelto por el servlet ServletMostrarRedes.

4.2 Resto de ficheros

4.2.1 Ficheros jsp y relacionados

Los ficheros jsp se especifican en 8, junto con sus ficheros js.

4.2.2 Ficheros java

Se especifican más adelante, en distintos apartados (9 y 10), según su funcionalidad.

4.2.3 Ficheros CGI

El fichero workspace/AppTrabajo/src_cgi/jscgi.c debe tener su ejecutable asociado a la URL:

```
/AppTrabajo/cgi-bin/jscgi
```

Este programa no recibe datos. Genera código ECMAScript para cargarse en las páginas jsp. El código ECMAScript debe hacer que se añada al elemento footer de la página un párrafo con el valor de un contador y otro párrafo con el valor de las cookies. El contador lo mantiene en un fichero de nombre contador.txt. Las cookies las obtiene de la petición.

Todos los ficheros jsp que tengan cabecera html (elemento head) deben incluir en ella:

```
<script src="cgi-bin/jscgi"></script>
```

aunque la URL puede ser diferente según la página que lo incluye.

El fragmento de código C que se debe modificar en el fichero suministrado debe generar el código ECMAScript que le falta a la función sin nombre para que dentro del elemento footer de la página (se puede suponer que sólo hay uno) añada dos párrafos, uno con el contador y otro con las cookies. Por ejemplo:

```
<p><strong>Contador=</strong>1
</p>
<p><strong>Cookies:</strong><br/>
usuario=c1; JSESSIONID=E00AA4EA5639AEC0FCF5C6B3ADEA2536
</p>
```

Se suministra también un fichero makefile que facilita la compilación, copia y modificación de permisos.

5 Listener

5.1 AppListener.java

Al iniciarse la aplicación AppTrabajo, realiza las siguientes operaciones:

Crea un atributo de aplicación de nombre autor y cuyo valor es una cadena que debe contener "Apellidos, Nombre - UVUS" correspondiente al alumno.

Accede a la base de datos y leyendo de la tabla de tipos de interfaz, rellena un mapa de nombre tipos_in (para cada elemento, la clave es el identificador de tipo y el valor es la descripción).

Crea un atributo de aplicación de nombre tipos_in y cuyo valor es el mapa anterior (este atributo se usa para no tener que acceder continuamente a la base de datos, ya que esta tabla no cambia durante la ejecución de la aplicación).

Crea un atributo de aplicación de nombre ds y cuyo valor es el objeto de tipo DataSource usado para establecer la conexión (este atributo se usa para no tener que hacer la llamada a lookup en otros accesos a la base de datos).

6 Filtros

6.1 FiltroMenu.java

Captura la URL /menu, y realiza las siguientes operaciones:

Si la petición contiene usuario y contraseña:

- accede a la base de datos, comprueba que sean correctos y en ese caso averigua si es administrador.
- si ha comprobado que son correctos:
    - crea un atributo de sesión de nombre usuario y cuyo valor es un objeto de la clase Usuario con los valores correctos (esta clase debe hacerla previamente como se explica más adelante)

- crea una cookie de nombre usuario y valor el identificador de usuario y la añade a la respuesta

Después de lo anterior (independientemente del resultado), busca un atributo de sesión de nombre usuario (que se ha podido crear en el paso anterior o en otra petición anterior del mismo usuario que ya puede estar autenticado):

- si lo encuentra:
  - si el usuario es administrador: reenvía la petición al menú de administrador
  - si el usuario no es administrador: reenvía la petición al menú de clientes
- si no lo encuentra: reenvía la petición al inicio de la aplicación (es un usuario que está intentando acceder al menú y no está autenticado)

6.2 FiltroAdmin.java

Su objetivo es que sólo pueda acceder a las páginas de administrador (están el directorio admin) los usuarios autenticados y que además sean de tipo administrador. De esta forma en esas páginas no hace falta comprobar nada. Si no se usaran los filtros, habría que incluir en cada página la comprobación correspondiente.

Captura las peticiones con URL que empiecen con /admin/ y realiza las siguientes operaciones

Busca un atributo de sesión de nombre usuario:

- Si existe (eso significa que se ha autenticado) y es administrador deja seguir la petición
- En caso contrario ejecuta lo siguiente:

```
HttpServletResponse res = (HttpServletResponse) response;
res.sendError(HttpServletResponse.SC_FORBIDDEN, "Acceso prohibido");
```

6.3 FiltroClientes.java

Su objetivo es que sólo pueda acceder a las páginas de clientes (están el directorio clientes) los usuarios autenticados. Similar al anterior, pero con las URL que empiecen con /clientes/. En este caso no hay que comprobar si es administrador (pero sí debe existir el atributo de sesión).

7 JavaBean

7.1 Usuario.java

Es una clase con 3 propiedades:

```
private String usu;
private String contra;
private int tipo_usu;
```

y los “setters” y “getters” correspondientes.

Tiene también dos constantes:

```
public static final int ADMINISTRADOR = 0;
public static final int CLIENTE = 1;
```

Estas dos constantes se usan para comparar con el campo tipo_usu.

8 Páginas jsp

Todas las páginas jsp de la aplicación deben:

- Tener al principio (excepto si van a ser incluidas con la directiva <%@include):
    - la directiva @page
- Tener en la cabecera html:
    - referencia a la página de estilos estilos.css (con el camino adecuado)
    - incluido el código ECMAScript generado por cgi-bin/jscgi, con el camino adecuado (ver ejemplo en 4.2.3).
- Tener al principio del cuerpo html (excepto index.jsp):
    - incluido cabecera.jsp, con la forma y el camino adecuado (ver 8.1.1)
- Tener al final del cuerpo html:
    - incluido pie.jsp, con la forma y el camino adecuado (ver 8.1.2)

8.1 Páginas en WebContent

8.1.1 Página de cabecera: cabecera.jsp

Debe ser incluida por el resto de páginas jsp (a excepción de index.jsp) con la etiqueta de acción <jsp:include.

Debe incluir la directiva <%@page y un elemento con identificador de valor "cabecera".

Dentro del div debe haber dos elementos <a>, un elemento y otro .

Primer elemento <a>:

- referencia: /AppTrabajo/menu
- clase: acceso
- identificador: volver
- contenido: Volver al menú

Segundo elemento <a>:

- referencia: /AppTrabajo/cerrar
- clase: acceso
- contenido: Cerrar sesión

Elemento :

- clase: acceso
- identificador: nombreusuario
- contenido: el identificador del usuario que está accediendo

Elemento (sin contenido):

- identificador: sepCabecera

Muy importante: este fichero NO debe contener el texto AppTrabajo, para obtener la referencia se debe usar una EL (Expression Language), y no se deben usar scriplets. La ventaja de usar EL para obtener /AppTrabajo es que es un código reutilizable en otras aplicaciones. Para obtener el identificador del usuario, se debe usar una EL con el atributo de sesión de nombre usuario que se establece cuando un usuario se autentica.

Para las referencias tenga en cuenta también lo explicado en 12.1 y siguientes.

8.1.2 Página de pie: pie.jsp

Debe ser incluida por el resto de páginas jsp con la directiva <%@include.

Tenga en cuenta que la inclusión es distinta a cabecera.jsp y por lo tanto este fichero no incluye la directiva <@page.

Debe contener un elemento :

- identificador: pie
- contenido: un elemento footer que contiene un elemento cabecera de nivel 2 cuyo contenido es:

  2017-18 Fast: Apellidos, Nombre – UVUS

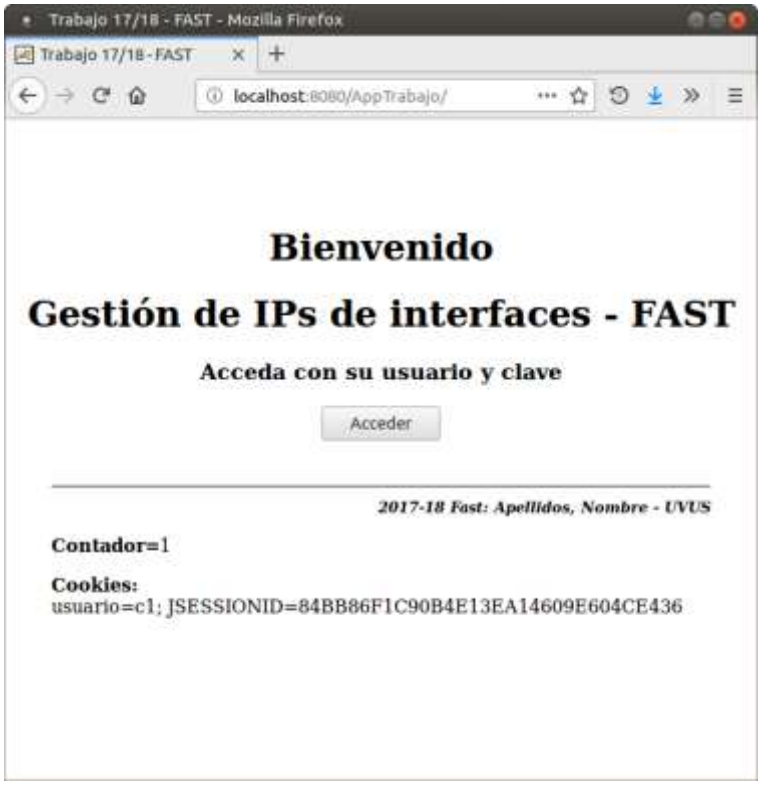
  del autor, obtenido mediante una EL (Expression Language) a partir del atributo de aplicación que se establece cuando la aplicación se inicia.

8.1.3 Página de inicio: index.jsp

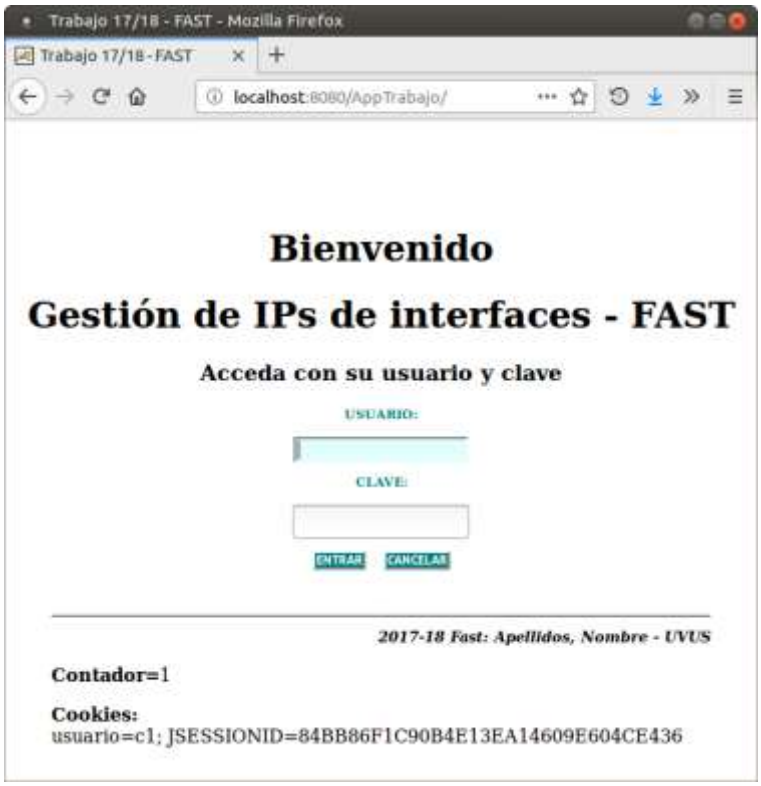
La primera vez que se accede a

http://localhost:8080/AppTrabajo/

debe mostrar la página de bienvenida con el botón para acceder (en el pie deben aparecer los datos del alumno, que se habrán inicializado en el Listener):



Cuando se pulse el botón acceder, se debe mostrar el formulario de acceso y se podrán introducir los datos de usuario:



Si se pulsa CANCELAR se debe ocultar el formulario de acceso.

Si se pulsa ENTRAR se debe acceder a la URL menu (que será capturada por el filtro correspondiente), y si los datos son incorrectos se vuelve a la página de inicio.

El fichero suministrado cumple las especificaciones, pero debe completar el fichero conmutaAcceso.js para que funcione adecuadamente (ver siguiente apartado).

8.1.3.1 Ficheros asociados

8.1.3.1.1 js/conmutaAcceso.js

Debe contener las funciones:

- muestraFormAcceso(): muestra el elemento con id igual a “formacceso” (estilo de display “block”, ya que inicialmente, por su estilo, estaba oculto), pone el foco en el elemento con id igual a “usu” y oculta el elemento con id igual a “botAcceso” (estilo de display “none”).
- ocultaFormAcceso(): restaura el elemento con id igual a “botAcceso” (estilo de display “”) y oculta el elemento con id igual a “formacceso”.
- Función sin nombre que se ejecuta al terminar de cargar la página: Hace que se ejecute la función muestraFormAcceso cuando se hace click en el elemento con id igual a “botAcceso” y que se ejecute la función ocultaFormAcceso cuando se hace click en el elemento con id igual a “cancelar”

8.1.3.1.2 css/estilo.css

Este es el fichero de estilo usado por todas las páginas.

El contenido de este fichero se ha especificado en el apartado 4.1.2.

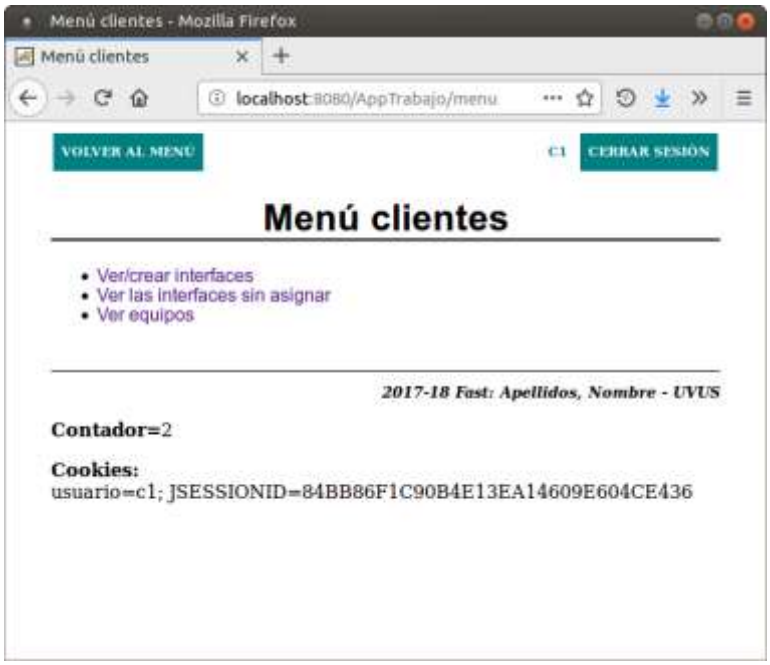
8.2 Páginas en WebContent/clientes

Las páginas dentro de este directorio están protegidas por FiltroClientes, y por lo tanto sólo podrán ser accedidas por usuarios autenticados.

8.2.1 Página de "Menú clientes": menu.jsp

Esta página es accedida a través del filtro menu. Se llega después de accecer a index.jsp o desde el enlace que hay en cabecera.jsp.

Debe completar el código suministrado para que cumpla lo especificado.



8.2.2 Página de "Ver equipos" de clientes: verEquipos.jsp

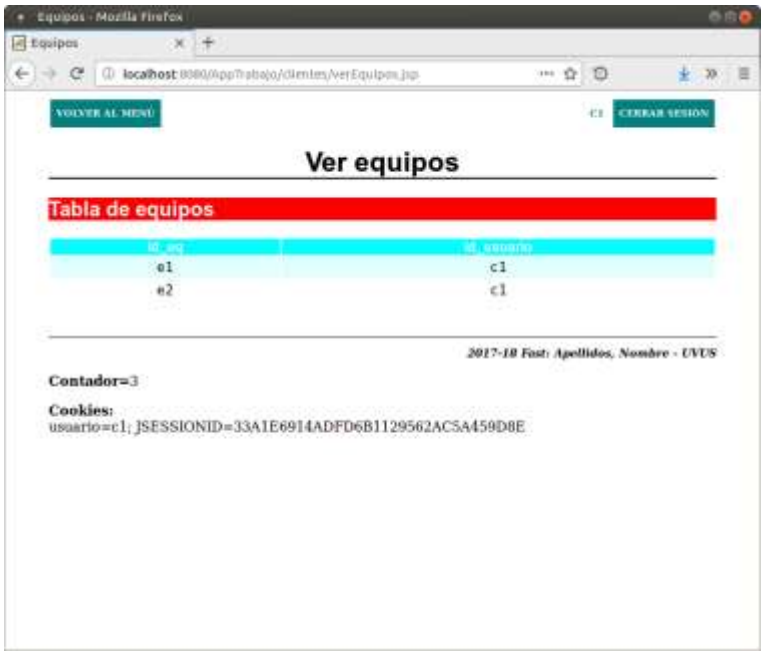
Esta página es accedida a través del enlace que hay en menu.jsp.

Debe mostrar los equipos que pertenecen al usuario, haciendo uso del servlet ServletEquipos.

Debe modificar la etiqueta de inicio de table para que ésta se rellene con los datos devueltos por el servlet. Para ello debe analizar y comprender el funcionamiento de los ficheros ECMAScript suministrados (que debe modificar para que funcionen correctamente):

- clientes/js/funCargaTablas.js
- clientes/js/funTablasURL.js

En funTablasURL.js están todas las funciones relacionadas con tablas, aunque en verEquipos.jsp no se usen todas.



Estas capturas y las correspondientes a las siguientes páginas las puede consultar en capturas.tar.gz

8.2.3 Página de "Ver interfaces sin asignar" de clientes: verIntSinAsignar.jsp

Esta página es accedida a través del enlace que hay en menu.jsp.

Debe mostrar las interfaces que no tiene asignadas IP y que pertenecen a equipos del usuario, haciendo uso del servlet ServletIntSinAsignar.

Este fichero debe ser similar verEquipos.jsp

8.2.4 Página de "Ver/crear interfaces" de clientes: verCrearInterfaces.jsp

Esta página es accedida a través del enlace que hay en menu.jsp.

Debe mostrar todas las interfaces que pertenecen a equipos del usuario, haciendo uso del servlet ServletIntSinAsignar.

Debe modificar la etiqueta de inicio de table para que ésta se rellene con los datos devueltos por el servlet. Para ello debe analizar y comprender el funcionamiento de los ficheros ECMAScript suministrados (los mismos que en el apartado 8.2.2).

Cuando se pulse el botón de "Añadir Interfaz" debe mostrarse un formulario inicialmente oculto con 3 campos: equipo, tipo e interfaz.

El campo tipo es una lista desplegable que debe contener el identificador y la descripción de los tipos de interfaces posibles, obtenidos del atributo de aplicación tipos_in (recuerde que es un mapa que se inicializa en AppListener). El campo select se rellena en el servidor mediante un scriplet que se debe modificar para que funcione correctamente.

El campo equipo es una lista desplegable que debe contener los identificadores de los equipos que aparecen en la tabla mostrada (sin repetir). Esta lista se genera en el cliente (el navegador) mediante el código ECMAScript del fichero:

- clientes/js/funMuestraForm.js

Este fichero se suministra con comentarios, y se debe modificar para que funcione correctamente.

Los datos del formulario (equipo, tipo e interfaz) se deben enviar al servlet ServletNuevaInterfaz (ver 9.6).

8.2.5 Página de “Error”: error.jsp

A esta página se llega cuando se produce un error al intentar hacer alguna modificación mediante un servlet (ver los servlets para entender cómo se llega a esta página).

Esta página usa dos atributos de petición a los que previamente el servlet correspondiente les habrá asignado el valor adecuado. Debe modificar el fichero suministrado para que el contenido de la cabecera de nivel 1 sea el contendido del atributo de nombre mensaje. Debe usar una EL para ello. El contendido del otro atributo (de nombre volverURL) ya está asociado con el atributo href del enlace y debe servir para volver a la página previa al servlet que tuvo el error (el contenido del atributo lo establecerá el servlet correspondiente).

Esta página es única (no hay otra en el directorio admin).

8.3 Páginas en WebContent/admin

Las páginas dentro de este directorio están protegidas por FiltroAdmin, y por lo tanto sólo podrán ser accedidas por usuarios autenticados de tipo administrador.

8.3.1 Página de “Menú administrador”: menu.jsp

Esta página es accedida a través del filtro FiltroMenu (y por lo tanto sólo se llega si el usuario autenticado es de tipo administrador. Se llega después de acceder a index.jsp o desde el enlace que hay en cabecera.jsp.

Es similar al menú de clientes con algunos cambios:

- Cambiar título y cabecera h1 a: Menú administrador
- Cambiar enlaces a:
    - Ver usuarios
    - Crear/modificar usuarios
    - Ver/crear interfaces (de todos los usuarios)
    - Ver todos los equipos (de todos los usuarios)
    - Asignar direcciones IP

8.3.2 Página de “Ver usuarios”: verUsuarios.jsp

Esta página permite ver todos los usuarios de la base de datos haciendo uso del servlet ServletVerUsuarios. Es similar a la página de clientes verEquipos.jsp, pero cambiando el título, la cabecera h1 y el identificador de la tabla. Igual que en verEquipos.jsp debe cargar los ficheros:

- clientes/js/funCargaTablas.js
- clientes/js/funTablasURL.js

para que se rellene la tabla con los datos devueltos por el servlet.

Tenga en cuenta esta página se encuentra en el directorio admin y no en clientes.

8.3.3 Página de “Crear/modificar usuarios”: usuForm.jsp

Esta página ya está especificada en 4.1.1.

8.3.4 Entrada de menú de “Ver/crear interfaces (de todos los usuarios)” de administrador

Esta entrada del menú de administrador debe llevar a la misma página de “Ver/crear interfaces” de clientes, especificada en 8.2.4. Aunque sea la misma página, el servlet usado permitirá en este caso ver/crear interfaces en todos los equipos de todos los usuarios (gracias a que el usuario autenticado es de tipo administrador).

8.3.5 Entrada de menú de “Ver equipos (de todos los usuarios)” de administrador

Esta entrada del menú de administrador debe llevar a la misma página de “Ver equipos” de clientes, especificada en 8.2.2. Aunque sea la misma página, el servlet usado permitirá en este caso ver los equipos de todos los usuarios (gracias a que el usuario autenticado es de tipo administrador).

8.3.6 Página de “Asignar direcciones IP”: asigIpIntSinAsignar.jsp

Esta página permite ver todas las interfaces de todos los equipos de todos los usuarios, que no tienen dirección IP asignada, haciendo uso del servlet ServletIntSinAsignar. Es similar a la página de clientes verEquipos.jsp, pero cambiando el título, la cabecera h1 y el identificador de la tabla. En este caso debe cargar los ficheros:

- clientes/js/funTablasURL.js
- admin/js/funCargaModifTabla.js

para que se rellene la tabla con los datos devueltos por el servlet, y además se eliminan las 3 últimas columnas de la tabla (que en este caso son la IP, la máscara y la IP de pasarela, que valen null), y se añade una columna con un elemento <a>. El atributo href del elemento <a> sirve para acceder a la página ipForm.jsp, enviándole dos parámetros correspondientes a las dos primeras columnas (que son el identificador del equipo y la interfaz). Los nombres de los parámetros se toman de las columnas de la cabecera.

Por ejemplo, si una fila es:

```
<tr><td>e2</td><td>1</td><td>wifi</td><td>null</td><td>null</td><td>null</td></tr>
```

se convierte en:

```
<tr><td>e2</td><td>1</td><td>wifi</td><td><a href="ipForm.jsp?id_eq=e2&num_in=1">asigna IP</a>
</td></tr>
```

Cuando se hiciera click en la columna añadida de esa fila, se accedería a ipForm.jsp con dos parámetros: el identificador del equipo (que valdría e2) y el número de interfaz (que valdría 1).

8.3.7 Página ipForm.jsp

Esta página ya está especificada en 4.1.1. Es accedida desde la página de “Asignar direcciones IP”.

9 Servlets

9.1 ServletCerrar.java

Debe estar asociado a la URL "/cerrar" y debe cerrar la sesión y reenviar la petición al inicio de la aplicación. También debe borrar la cookie creada por la aplicación (para comprobar el borrado también se puede observar la cabecera apropiada en el mensaje HTTP).

9.2 ServletIntSinAsignar.java

Debe estar asociado a varias URLs: una URL permitida para administradores y a una URL permitida para clientes (ver filtros). Las URLs deben terminar en verIntSinAsignar.

Debe devolver en un Array de objetos JSON todas las interfaces de todos los equipos de todos los usuarios, que no tengan ninguna dirección IP asignada, si el usuario autenticado es administrador. Si no es administrador, sólo debe devolver las de los equipos del usuario autenticado.

Se debe modificar el fichero suministrado para que cumpla lo especificado.

Un ejemplo sería:

```
[
  {"id_eq": "e1","num_in": "2","des": "ethernet","ip": "null","masc": "null","ip_gw": "null"},
  {"id_eq": "e2","num_in": "0","des": "ethernet","ip": "null","masc": "null","ip_gw": "null"},
  {"id_eq": "e2","num_in": "1","des": "wifi","ip": "null","masc": "null","ip_gw": "null"},
  {"id_eq": "e4","num_in": "0","des": "wifi","ip": "null","masc": "null","ip_gw": "null"},
  {"id_eq": "e5","num_in": "0","des": "ethernet","ip": "null","masc": "null","ip_gw": "null"},
  {"id_eq": "e5","num_in": "1","des": "wifi","ip": "null","masc": "null","ip_gw": "null"},
  {"id_eq": "e6","num_in": "1","des": "wifi","ip": "null","masc": "null","ip_gw": "null"},
  {"id_eq": "e91","num_in": "1","des": "wifi","ip": "null","masc": "null","ip_gw": "null"}
]
```

9.3 ServletInterfaces.java

Debe estar asociado a una URL permitida tanto para administradores como para clientes (ver filtros). La URL debe terminar en verInterfaces.

Debe devolver en un Array de objetos JSON todas las direcciones IP de todas las interfaces de todos los equipos de todos los usuarios, si el usuario autenticado es administrador. Para cada IP asignada, el objeto JSON debe tener el identificador de equipo, número de interfaz, descripción de la interfaz, la IP, la máscara y IP de pasarela.

Debe ser similar al servlet ServletIntsinAsignar. Si el usuario autenticado no es administrador, sólo debe devolver las de los equipos del usuario del usuario autenticado. Las sentencias SQL son similares a las del servlet anterior, eliminando la condición de que la dirección IP sea nula.

Un ejemplo sería:

```
[
  {"id_eq":  "e1","num_in":  "1","des":  "ethernet","ip":  "10.1.1.11","masc":  "8","ip_gw":
"10.1.1.1"},
  {"id_eq": "e1","num_in": "2","des": "ethernet","ip": "null","masc": "null","ip_gw": "null"},
  {"id_eq": "e2","num_in": "3","des": "wifi","ip": "10.1.2.13","masc": "8","ip_gw": "10.1.1.1"}
]
```

9.4 ServletEquipos.java

Debe ser similar al servlet ServletInterfaces. La URL debe terminar en verEquipos. Si el usuario autenticado no es administrador, sólo debe devolver los equipos del usuario autenticado. Las sentencias SQL son mucho más simples, ya que sólo accede a una tabla.

Un ejemplo sería:

```
[
  {"id_eq": "e1","id_usuario": "c1"},
  {"id_eq": "e2","id_usuario": "c1"},
  {"id_eq": "e3","id_usuario": "c2"},
  {"id_eq": "e4","id_usuario": "c2"},
  {"id_eq": "e5","id_usuario": "c3"},
  {"id_eq": "e6","id_usuario": "c3"},
  {"id_eq": "e91","id_usuario": "admin"}
]
```

9.5 ServletUsuarios.java

Debe estar asociado a una URL permitida sólo para administradores (ver filtros). La URL debe terminar en verEquipos. Debe devolver todos los usuarios existentes en la base de datos (sólo el identificador).

Un ejemplo sería:

```
[
  {"id_usuario": "admin"},
  {"id_usuario": "c1"},
  {"id_usuario": "c2"},
  {"id_usuario": "c3"}
]
```

9.6 ServletNuevaInterfaz.java

Debe estar asociado a una URL permitida tanto para administradores como para clientes (ver filtros). La URL debe terminar en nuevaInterfaz.

Este servlet es accedido a través de la página verCrearInterfaces.jsp, de la que recibe los datos del formulario, con los nombres: equipo, tipo e interfaz.

Debe crear una nueva interfaz en el equipo recibido como parámetro, que puede ser cualquiera si el usuario autenticado es administrador. Si no es administrador, sólo debe crearla si el equipo pertenece al usuario autenticado.

Si todo es correcto y se crea la interfaz correctamente, debe redirigir el navegador a la página de "Ver/crear Interfaces".

Si al intentar crear la nueva interfaz se produce algún error, debe reenviar la petición a la página de error (error.jsp), estableciendo previamente dos atributos de petición de nombres "mensaje" y "volverURL".

El valor de "mensaje" debe ser "Error creando la nueva interfaz.".

El valor de "volverURL" debe ser el necesario para que desde la página de error se pueda volver a la página accedió a este servlet.

Se debe modificar el fichero suministrado para que cumpla lo especificado.

9.7 ServletModificarUsuario.java

Debe estar asociado a una URL permitida sólo para administradores (ver filtros). La URL debe terminar en modificarUsuario.

Debe ejecutarse cuando el administrador quiere modificar la contraseña de un usuario existente o crear uno nuevo.

Recibe los parámetros del formulario de usuForm.jsp, y se debe comprobar que cumplen las mismas condiciones que en el formulario (ahora en el servidor las comprobaciones se hacen en java).

Si todo es correcto y la modificación/creación se efectúa correctamente, debe redirigir el navegador a la página de "Ver usuarios".

Si hay algún error debe reenviar la petición a la página de error (error.jsp), estableciendo previamente dos atributos de petición de nombres "mensaje" y "volverURL".

El valor de "mensaje" debe ser el de la variable mensaje (generado durante la ejecución del servlet).

El valor de "volverURL" debe ser el necesario para que desde la página de error se pueda volver a la página accedió a este servlet.

Se debe modificar el fichero suministrado para que cumpla lo especificado.

9.8 ServletModificarIp.java

Debe estar asociado a una URL permitida sólo para administradores (ver filtros). La URL debe terminar en modificarIp.

Debe ejecutarse cuando el administrador quiere modificar la dirección IP de una interfaz (y la máscara y la IP de pasarela).

Recibe los parámetros del formulario de ipForm.jsp, y se debe comprobar que cumplen las mismas condiciones que en el formulario (ahora en el servidor las comprobaciones se hacen en java).

Si todo es correcto y la asignación se efectúa correctamente, debe redirigir el navegador a la página de "Asignar direcciones IP" (donde se ven las interfaces sin asignación de IP).

Si hay algún error debe reenviar la petición a la página de error (error.jsp), estableciendo previamente dos atributos de petición de nombres "mensaje" y "volverURL".

El valor de "mensaje" debe ser "Error asignando la nueva IP.".

El valor de "volverURL" debe ser el necesario para que desde la página de error se pueda volver a la página accedió a este servlet, pero con los parámetros correspondientes a identificador de equipo y número de interfaz. Estos parámetros deben tomar sus valores de los parámetros correspondientes recibidos por el servlet. Por ejemplo, la URL (tenga en cuenta el camino de la página de error) podría contener:

```
ipForm.jsp?id_eq=e1&num_in=0
```

si el servlet recibió entre sus parámetros el identificador de equipo con el valor "e1" y número de interfaz con el valor "0".

Se debe modificar el fichero suministrado para que cumpla lo especificado.

9.9 ServletMostrarRedes.java

Debe estar asociado a una URL permitida sólo para administradores (ver filtros). La URL debe terminar en mostrarRedes.

Debe ejecutarse cuando el administrador quiere comprobar si la dirección IP que está intentando asignar está dentro de alguna subred de las direcciones IP ya asignadas. Este servlet es invocado por la petición AJAX que se envía al pulsar el botón "Mostrar Redes" de la página ipForm.jsp (ver 8.3.7).

Recibe como parámetro una dirección IP y debe devolver en un Array de objetos JSON todas las direcciones IP ya asignadas (de todas las interfaces de todos los equipos de todos los usuarios) en las que la IP recibida esté en la misma subred que la IP asignada.

Para cada IP devuelta, el objeto JSON debe tener la IP, la máscara y IP de pasarela.

Un ejemplo sería (si se recibe `10.2.3.4`):

```
[
  {"ip": "10.1.1.10","masc": "8","ip_gw": "10.1.1.1"},
  {"ip": "10.1.1.11","masc": "8","ip_gw": "10.1.1.1"},
  {"ip": "10.1.2.13","masc": "8","ip_gw": "10.1.1.1"},
  {"ip": "10.1.3.10","masc": "8","ip_gw": "10.1.3.1"},
  {"ip": "10.1.3.11","masc": "8","ip_gw": "10.1.3.1"}
]
```

10 Ficheros java adicionales

10.1 OperacionesIP

Contiene código que facilita la comprobación y comparación de direcciones IP.

11 Entrega

Se debe entregar un fichero **trabajo.tar.gz** que contenga el directorio **workspace/AppTrabajo** (de **/home/dit**):

```
dit@localhost:~$ tar cvfz trabajo.tar.gz workspace/AppTrabajo
```

12 Anexo

Algunas notas:

12.1 Tipos de "include"

Recuerde las diferencias entre los dos tipos de "include" en una página JSP:

Etiqueta de acción:

```
<jsp:include page="URL" />
```

Directiva:

```
<%@include file="FICHERO"%>
```

Además, tenga en cuenta que el valor de page es una URL (dentro de la aplicación) pero el de file es un fichero (con camino absoluto o relativo a la página que cargó el navegador). Ver más información en 12.3.

12.2 Filtros

Los filtros no capturan por defecto los reenvíos (forward), pero si se hace una redirección (sendRedirect) se obliga al navegador a hacer una nueva petición, y en ese caso el filtro se vuelve a ejecutar en la nueva petición.

Eso es lo que ocurre en una redirección, como por ejemplo en index.jsp:

```
response.sendRedirect("menu");
```

Si se hiciera un reenvío (forward), entonces el filtro por defecto no lo captura.

12.3 Reenvío y redirecciones

Tenga en cuenta que un reenvío (forward) ocurre dentro del servidor y, en cambio, una redirección (sendRedirect) hace que el cliente (navegador) realice una nueva petición. Por lo tanto, el camino en la URL de ambos se debe construir teniendo en cuenta dónde se va a evaluar (servidor o navegador)

12.4 URLs en navegador

Cuando en el navegador se hace click en un enlace relativo, el navegador utiliza como referencia la URL actual, que puede ser servida por un servlet que esté en otro directorio.

Por ejemplo, cuando se hace un reenvío (forward), el navegador ve la URL original pero el servlet ejecutado es otro diferente.

Ejemplo: Cuando el navegador accede a un recurso (por ej. /AppTrabajo/index.jsp) y luego es redirigido a un segundo recurso (por ej. `menu`) que luego es reenviado en el servidor a un tercer recurso (por ej. "/admin/menu.jsp"), el navegador no conoce la ubicación de ese tercer recurso y por lo tanto los caminos relativos a otros recursos en el navegador serán respecto a la ubicación del segundo (es decir /AppTrabajo/) y no del tercero (es decir /AppTrabajo/admin/)

12.5 Depuración

Modo depuración en Eclipse: el servidor se puede iniciar en modo depuración (Ctl+Alt+D o pulsando el icono del “bug” en vez del “play”). Funciona de forma similar al depurador del navegador (permite fijar puntos de interrupción, inspeccionar variables, etc).