Programación Web

Informe de Prácticas

Práctica 3. DESARROLLO DE UNA APLICACIÓN WEB

Daniel Ortega León

Damián Martínez Ávila

Introducción

En esta práctica haremos uso de todos los conocimientos aprendidos, en las sesiones tanto de teoría como de práctica, en referencia al desarrollo de back-end y front-end en web y buenas prácticas.

Pila de tecnología utilizada para el desarrollo: Eclipse IDE , Apache Tomcat 8.5, Java 1.7, HTML, CSS y Javascript.

Decisiones de diseño e implementación

Partiendo de la práctica anterior, debemos añadir la gestión de anuncios. Creamos una nueva página para que cada usuario registrado pueda acceder a ella y gestionar sus propios anuncios.

Dentro de esta página el usuario podrá administrar sus anuncios ya creados (ya sea cambiando el estado del mismo o modificando su contenido) o crear nuevos anuncios de varios tipos.

Index mostrará el tablón de anuncios de cada usuario refrescándose cada pocos segundos gracias a una consulta asíncrona (sin necesidad de recargar la página) que obtendrá los anuncios requeridos del Servlet GetAnunciosController. Lo hemos hecho mediante la consulta asíncrona (XMLHttpRequest) ya que al hacerlo con HttpServletRequest pasando

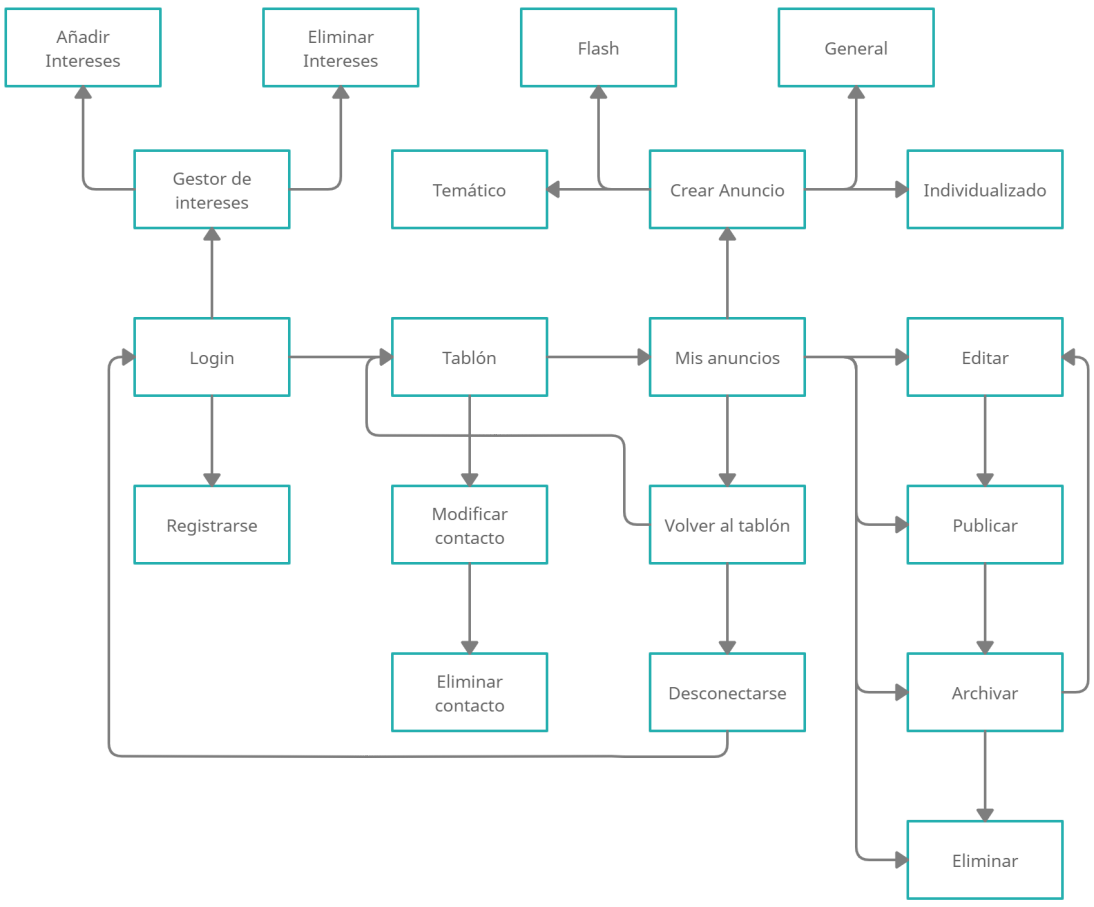
los anuncios con request.getSession.setAttribute() cada vez que se actualizan los datos se reinicia la página y vuelve al inicio, lo que nos parecía que era muy molesto.

En esta página también se podrá filtrar los anuncios del tablón gracias a javascript y buscar en toda la base de datos según un criterio aunque el usuario no sea destinatario de dicho anuncio (también mediante una consulta asíncrona).

Las vistas de nuestra aplicación consistirán en formularios para crear anuncios, registrar usuarios, etc... Utilizaremos Javascript para el control de errores y que muestre distintos elementos según el tipo de anuncio que se quiera crear, por ejemplo, en las vistas que consistirán en formularios HTML Por último, antes de que el usuario se registre en la aplicación Web, podrá gestionar los intereses de la aplicación.

En cuanto a la conexión a la base de datos. Los DAO reciben la conexión a la base de datos de la clase DAO que es una especie de factoría.

Mapa de la aplicación



Login: Index.jsp -> loginController.jsp -> loginView.jsp -> Index.jsp

Registrarse: loginView.jsp -> registerController.jsp -> registerView.jsp -> registerController.jsp -> Index.jsp

Tablon: Index.jsp -> getAnunciosController -> Index.jsp

Modificar Datos: index.jsp -> registerController.jsp -> registerView.jsp -> registerController.jsp -> index.jsp

Eliminar Usuario: registerView.jsp -> registerController.jsp -> index.jsp

Mis anuncios: anuncios.jsp -> getAnunciosController-> anuncios.jsp

Crear Anuncio: anuncios.jsp -> anunciosController -> anuncioView.jsp -> anunciosController -> anuncios.jsp

Editar/Publicar/Archivar/Eliminar/Recuperar: anuncios.jsp -> estadosController -> anuncios.jsp

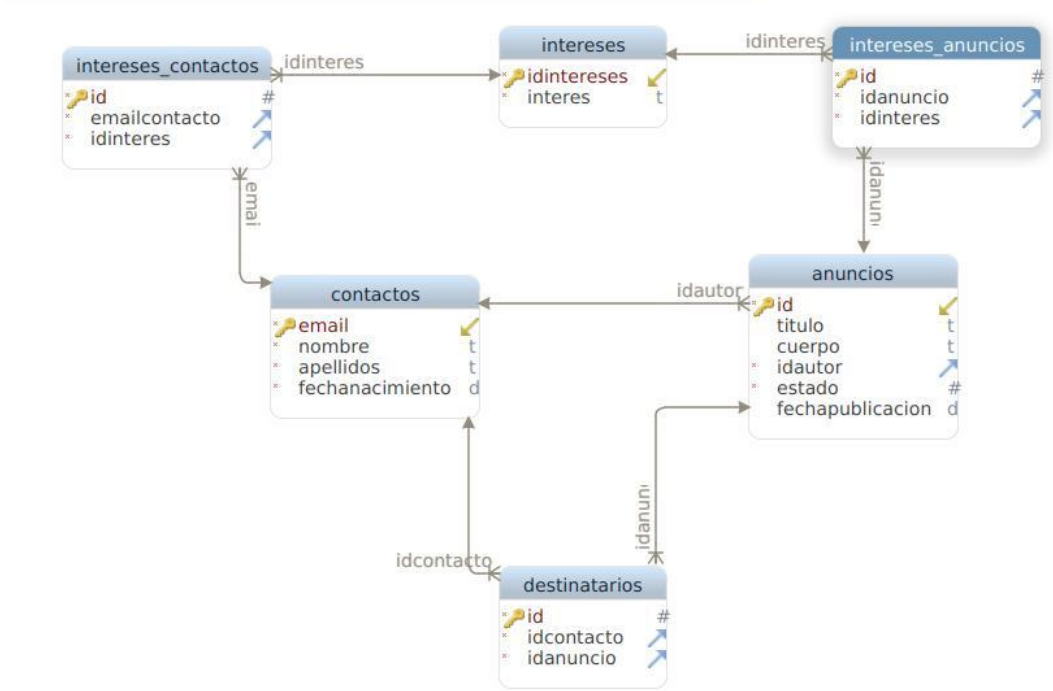
Desconectarse: Index.jsp -> loginController.jsp -> Index.jsp -> loginView.jsp

Gestor de Intereses: loginView.jsp -> intereses.jsp

Crear intereses: intereses.jsp -> interesesController -> interesesView.jsp -> InteresesController -> intereses.jsp

Eliminar Intereses:intereses.jsp -> estadosController -> index.jsp ->loginView

Esquema relacional de la Base de Datos



DAO y sus metodos

ContactoDAO

- getInstance()
- crearContacto()
- borrarContacto(Contacto)
- actualizarContactoNombre()
- actualizarContactoApellido(Contacto, String)
- actualizarContactoFecha(Contacto, String)
- actualizarContactoInteres(Contacto, String[])
- buscarContactoEmail(String)
- getContactos()
- getContactosInteres(int)

InteresDAO

- getInstance()
- crearInteres(String)
- borrarInteres(int)
- getIntereses()
- getInteresesContacto(String)
- getInteresesAnuncio(int)

AnunciosDAO

- getInstance(Connection)
- getAnuncios()
- getMisAnuncios(String)
- getAnunciosTematicos()
- getAnunciosInteres(int)
- insertarDestinatariosTematico(Anuncio)
- insertarTodosDestinatarios(Anuncio)
- insertarDestinatariosIndividualizado()
- guardarAnuncio(Anuncio, String[])
- publicarAnuncio(Anuncio)
- modificarTitulo(Anuncio,String)
- modificarCuerpo(Anuncio, String)
- modificarInteres(Anuncio, String[])
- almacenarAnuncios(ResultSet)
- modificarDestinatarios(Anuncio, String[])
- modificarFechaInicio(Anuncio, String)
- modificarFechaFin(Anuncio, String)
- archivarAnuncio(int)
- borrarAnuncio(int)
- restaurarAnuncio(int)
- getAnunciosContacto(String)
- getAnuncioById(int)
- getDestinatarios(int)
- ordenarPropietario(ArrayList)
- buscarFecha(String)
- buscarPropietario(String)
- buscarTitulo(String)

Para ver una descripción más detallada acceder a la documentación del proyecto en doc.

Instrucciones de acceso

Está hecho para funcionar en Java 7 y Apache Tomcat 8.5, la versión que está instalada en la UCO.

Abra la terminal y ejecute si no lo has hecho nunca para crear el directorio .tomcat:

/opt/apache-tomcat-8.5.24/bin/startup.sh

/opt/apache-tomcat-8.5.24/bin/shutdown.sh

En /home/tuloginuco/.tomcat/webapps/ -> pegas el war y lo extraes y eliminas el war.

Aclaración: para entrar en .tomcat debe pulsar control+h o habilitar la opción mostrar archivos ocultos.

Sustituya el fichero que está en /home/tuloginuco/.tomcat/conf/web.xml por el web.xml que se encuentra en el zip.

Por último abra el terminal y ejecute: /opt/apache-tomcat-8.5.24/bin/startup.sh

Acceda a localhost:8080/Ucopolis (o el nombre que le de al war)

Usuarios

Usuario: damian@gmail.com Contraseña: 1234

Usuario: diortega@gmail.com Contraseña: 1234

Fuentes Consultadas

Hemos recurrido a las diapositivas disponibles en Moodle, la documentación de Java y Javascript para las funcionalidades de ciertas funciones y también ciertos foros como StackOverflow.

https://es.stackoverflow.com/questions/95805/actualizar-%C3%BAnicamente-div-sin-actualizar-p%C3%A1gina (XMLHttpRequest)

https://www.w3schools.com (Diseño de la página Web)

https://regex101.com (Comprobacion email al crear la cuenta)

Análisis de Dificultades Encontradas

Hemos tenido algunas dificultades a la hora de implementar algunas de las funcionalidades de nuestros archivos .css para que hiciesen lo que realmente queríamos con el diseño de la página.

Otro de los problemas que se podrían destacar era que a la hora de crear el filtro de intereses en el index.jsp teníamos que darle la posibilidad de que pudiera detectar los anuncios con más de un interés además de que estaba la opción de que se podían crear más intereses por lo que el filtro también tenía que detectar dichos intereses.

Luego también hablando de los filtros hemos tenido otro contratiempo dado que no funcionan simultáneamente por lo que solamente se pueden filtrar los anuncios por un filtro.

Para la funcionalidad de que se recuerden los filtros si el usuario accede otra vez, no la hemos implementado. Lo único que se nos ocurre es guardándolo en la base de datos, porque un javaBean si entra otro usuario después no sería válido.

Hemos tenido varios problemas a la hora de la codificación de caracteres. Hemos optado por elegir UTF-8 para la codificación de la base de datos, pero hay casos en los que no mostraba los caracteres (ñ y tildes) correctamente y hemos mezclado UTF-8 con ISO-8859-1 en la aplicación web. Aún puede haber ciertos caracteres que no se muestren correctamente en alguna de las páginas aunque no ocurre con frecuencia.

En alguna ocasión se puede llegar a desconectarse de la base de datos al estar un tiempo considerable sin utilizar la aplicación web, por lo que hemos decidido tomarlo como una funcionalidad (la aplicación tiene un tiempo máximo de inactividad y una vez superado este tiempo se desconecta al usuario y se le redirecciona al login).