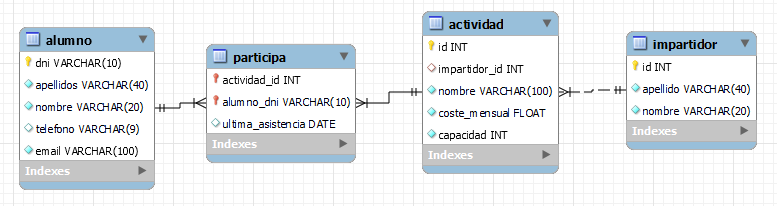
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **2º Trimestre– DEWS - JavaEE** | | | | | **2022-23** |
| Ciclo: **DESARROLLO DE APLICACIONES WEB** | | | **Curso: 2º** | | |
| **Módulo: DESARROLLO DE APL WEB ENTORNO SERVIDOR** | | | **Cod. Módulo: DEWS** | | |
| Cod. Prof: PEI3 | **Profesor: Amaia De la cruz** | | **Fecha: 24-01-2023** | | |
| **DATOS ALUMNO** | | | | | | Calificación | |
| **Nº:** | **Apellidos:** | | | **Nombre:** | |

# **Ejercicio Web recuperación PHP - Base de datos de Actividad**

**PROYECTO Web Java**

Se basa en la base de datos MySQL **bdactividad**, conla siguiente estructura:



**Tablas de BD - Proyecto PHP**

|  |  |
| --- | --- |
| **alumno** | **impartidor** |
| **actividad** | **participa** |

|  |  |
| --- | --- |
| **Ejercicio Web examen JavaEE - Aplicación a desarrollar de Actividad**  **Consideraciones generales del desarrollo en el examen**   * **Contenido de la estructura inicial a descargar:**  Fichero .RAR con **una carpeta de nombre 2eval\_ctrl\_actividad**.   + **Carpeta Elementos ayuda login**:     - **Ficheros java y jsp:** Ficheros para poder crear el login. Para ello es necesario colocar los ficheros dados en la estructura adecuada en el proyecto Web y crear los DAOs que permitirán la obtención de los datos necesarios de base de datos para el correcto funcionamiento.   *Index.jsp, ServletLogin.java, Alumno.java, Impartidor.java, Actividad.java*   * + **Carpeta bd**     - Fichero *bd\_actividad\_2ev\_2223.sql* para la importación de la ***bdactividad*** en phpMyadmin. * **Contenido de la entrega**. La entrega respetará la estructura dada que contendrá la carpeta de nombre **2eval\_ctrl\_actividad:**   + **Fichero ZIP o RAR** con el siguiente nombre: **nombre\_primerApellido\_2eval\_ctrl**   + Dentro del ZIP/RAR la carpeta **2eval\_ctrl\_actividad** con:     - **Proyecto Java**   + **Carpeta bd**     - fichero *bd\_actividad\_2ev\_2223.sql*     - fichero *bdactividad\_export.sql* exportación de bd tras el examen * **Puntuación general***:*   + Se **valorará que el aplicativo** **funcione correctamente** al hacer las pruebas mínimas de funcionamiento del **Happy Path**: Accedo a la página de login con información de un alumno. Se carga la pantalla de actividades del alumno y de posibles actividades a las que puede inscribirse. Se pulsa el enlace inscribirse, deberá de aparecer un enlace de guardado con el número de enlaces de inscripción pulsado. Por último, pulsar el enlace de guardado y guardar la inscripción en BD y mostrar la actualización en pantalla. (0,5 puntos) * **Penalizaciones en el examen:**   + Es **obligatorio seguir la convención de nombres de programación** (nombres significativos, formas de escribir variables, métodos, ficheros y clases). (-0,25 puntos)   + Se tendrán en cuenta la **legibilidad del código**. Para ello se revisará el formato y el aspecto estructurado del código. (-0,25 puntos)   + Es **obligatorio que el código este comentado** para que sea posible su comprensión por el profesor. Por lo tanto. (-0,25 puntos)   + Las clases Java deberán estar **organizadas en paquetes adecuados al MVC** (-0,25 puntos)   + La **configuración de los Servlets** será obligatoria crearla sobre el fichero **web.xml** (-0,25 puntos)   + Es **obligatorio** **utilizar los objetos bean para** movilizar la información de la parte modelo a la parte de la vista y viceversa. (-0,25 puntos)   + Es **obligatorio** **crear ficheros DAO para el manejo de las consultas con la BD**. (-0,25 puntos)   + Es **obligatorio** **crear un Pool de conexiones**. (-0,25 puntos)   + Es **obligatorio añadir las librerías necesarias** para el correcto funcionamiento del proyecto en la ruta adecuada - /WEB-INF/lib (-0,25 puntos)   + A la hora de realizar consultas con **información dinámica o precompilada, deberemos utilizar un preparedStatement** para añadir los parámetros. (-0,25 puntos) |  |

**Estructura para el examen 1 pto**

**Paquete de elementos Beans 0.25ptos**

* Bean **Alumno**: con la misma información que la tabla correspondiente de la BD
* Bean **Impartidor**: con la misma información que la tabla correspondiente de la BD
* Bean **Actividad**: que almacena id de la actividad, nombre, coste mensual e Impartidor que la imparte (objeto).

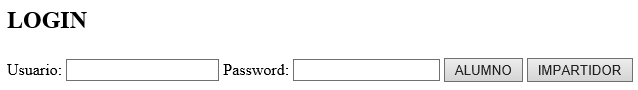
**Paquete de clases Dao 0.25ptos**

* Daos relacionados con el proyecto
  + **AlumnosDao**
  + **ActividadesDao**
  + **ImpartidoresDao**
  + **ParticiparDao**

**Paquete de clases conex 0.5ptos**

* Crea una clase de nombre ConexPoolBD para realizar la conexión con el pool de conexiones.
* Crea un Pool de conexiones: Pon a disposición de los métodos una fuente de datos asociada a un **pool de conexiones de nombre jdbc/poolActividadesDB**. Creación mediante configuración ficheros de Tomcat.

**Parte 1: index.jsp 1 pto**



**Vista 0 pto**

**index.jsp** es el punto de inicio del proyecto. Ambos submit envían el formulario al ServletLogin, quien redirige:

* A alumno.jsp, en caso de login correcto de un alumno
* A impartidor.jsp, en caso de login correcto de un impartidor
* A index.jsp, en caso de error de login

**Control/Servlet: 0 pto**

Creación de **ServletInscripcion.** Este servlet buscará la información a cargar en las dos tablas accediendo al modelo para cargar siguientes datos:

* Datos del alumno
* Datos de la lista de actividades de dicho alumno (Nombre actividad, Precio, Nombre Impartidor)
* Datos de la lista de actividades a las que no está apuntado el alumno (Nombre de Actividad, Nombre de impartidor)

**Modelo: 1 pto**

**AlumnosDao**

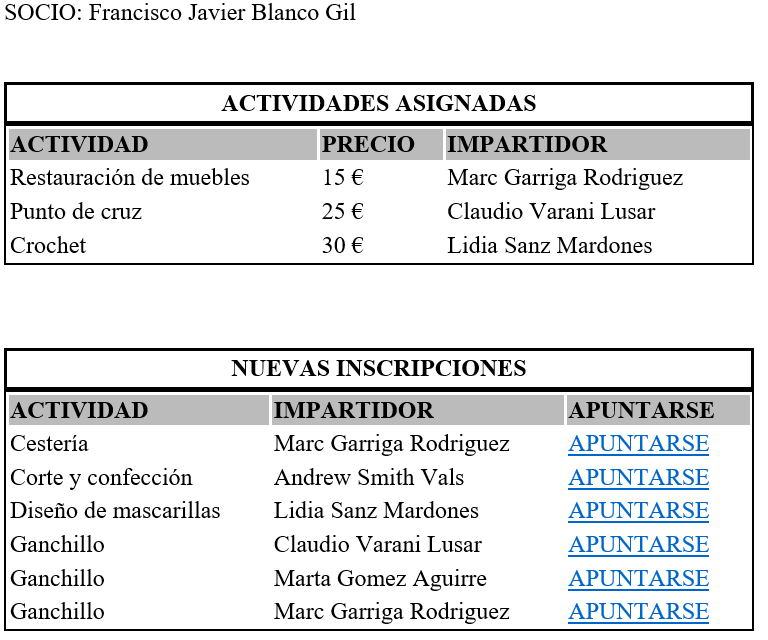
* Método Alumno **getAlumno**(String dni): Método que comprueba si existe el dni introducido por el usuario buscando en la tabla de Alumnos por el dni.

**ImpartidoresDao**

* Método Impartidor **getImpartidor**(String id): Método que comprueba si existe el impartidor en la tabla de Impartidores en base al id que introduce el usuario por pantalla.

**Parte 2: ALUMNO para inscribirse en nuevas actividades Paso 1: 2 ptos**

**Vista 0.5 pto**

**Estado inicial - alumno.jsp**

La página **alumno.jsp** muestra:

* El nombre del alumno logueado
* 1 tabla con sus actividades actuales de dicho alumno
* Una tabla con las actividades en las que puede matricularse: puede matricularse en actividades con plazas libres y cuyo nombre no coincida con ninguna actividad en la que ya esté inscrito. La última columna de esta tabla será un link para apuntarse.

\* Ambas tablas deben utilizar sintaxis JSTL

**Control/Servlet: 0.5pto**

Creación de **ServletInscripcion.** Este servlet buscará la información a cargar en las dos tablas accediendo al modelo para cargar siguientes datos:

* Datos del alumno
* Datos de la lista de actividades de dicho alumno (Nombre actividad, Precio, Nombre Impartidor)
* Datos de la lista de actividades a las que no está apuntado el alumno ( Nombre de Actividad, Nombre de impartidor)

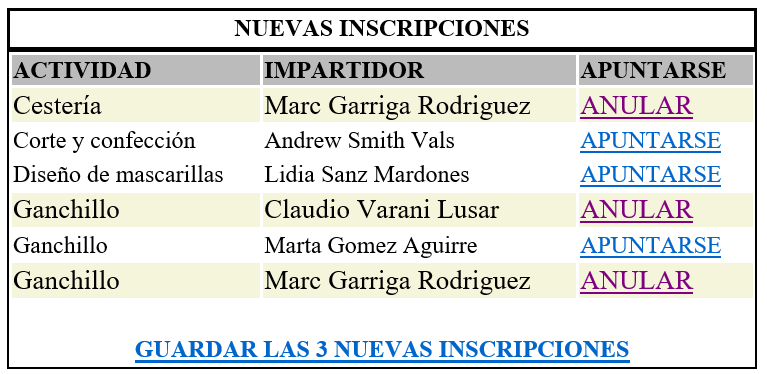
**Modelo: 1 pto**

**ActividadesDao**

* Método **obtenerActividadesParticipa**: recibe un alumno y devuelve una lista con las actividades en las que está inscrito. (0.5 ptos)
* Método **obtenerActividadesLibresNoParticipa**: recibe un alumno y devuelve una lista con las actividades en las que él puede matricularse. Un alumno puede matricularse en aquellas actividades en las que NO está inscrito, cuyo nombre no coincide con ninguna de las actividades en las que sí está inscrito, y con alguna plaza libre. (0.5 ptos)

**Parte 2: ALUMNO para inscribirse en nuevas actividades Paso 2: 1 ptos**

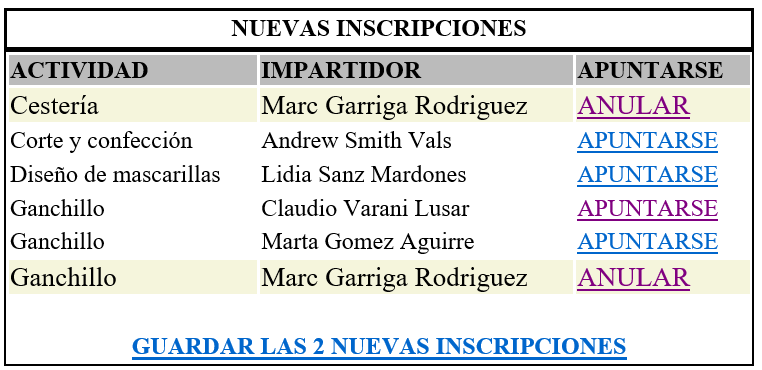
**Vista 0.5 pto**



**Interacciones con la pantalla - alumno.jsp**

Ejemplo: Pinchamos en el 1º, 4º y 6º enlace ****

* + Una colección va guardando las nuevas actividades en las que el alumno planea inscribirse (aún no se guardan en la BD, se guardan en sesión)
* Las actividades pinchadas se marcan con un fondo y el enlace se cambia al texto ANULAR
* La tabla añade un enlace final para materializar las inscripciones en la BD, indicando el número de inscripciones.



Ejemplo: Pinchamos en el 4º enlace 

* Se borra la correspondiente actividad del conjunto de posibles inscripciones en sesion y modifica el enlace de guardar indicando el numero de inscripciones actualizadas.

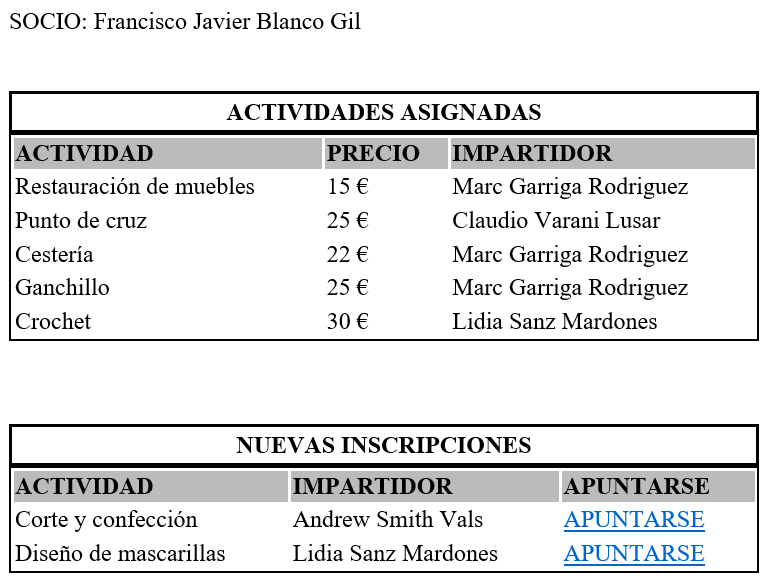
**Control/Servlet: 0.5 pto**

El **ServletInscripcion** deberá de gestionar **mediante distintos métodos los enlaces de Apuntarse y Anular,** volviendo a cargar la informaciónen pantalla con los cambios necesarios.

* Apuntarse: En el momento en el que pulsamos el botón de Apuntarse, se incluirá esa actividad en una colección en sesión, se creará o modificará el enlace de guardado con el número total de inscripciones, se modificará el enlace para mostrar el enlace Anular. (0,25 puntos)
* Anular: Al pulsar el enlace Anular, se eliminará la actividad de la colección de sesión y se modificará o eliminará el enlace de guardado y se modificará el enlace de Anular a Apuntarse. (0,25 puntos)

**Parte 2: ALUMNO para inscribirse en nuevas actividades Paso 2: 1.25ptos**

**Vista 0.25 ptos**



**Interacciones con la pantalla - alumno.jsp**

Ejemplo: Tras pinchar , se muestra:

* Mostrará de nuevo las tablas de actividades asignadas y de nuevas inscripciones y eliminará el enlace de guardado.
* El servlet incluye las nuevas inscripciones en la BD y vuelve a **alumno.jsp**, reflejando el nuevo estado.

**Control/Servlet: 0.5 pto**

**El ServletInscripcion** obtendrá la información de sesión, guardará las inscripciones mediante el método de llamada al modelo. De ahí volverá a cargar la pantalla, reflejado el nuevo estado.

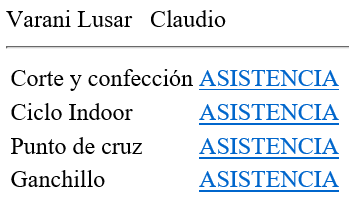
**Modelo: 0.5 pto**

**ParticipaDao**

* Método **inscribir**: Método que inscribe las actividades seleccionadas de la lista de nuevas inscripciones, con el dni del alumno, id de actividad y fecha de inscripción en la tabla. Este método recibirá el vean de Alumno y de Actividad y se hará la insert en BD.

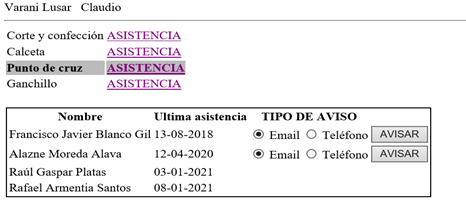
**Parte 3: IMPARTIDOR Paso 1: 2.5 ptos**

**Vista - NO usa JSTL 1 pto**



**impartidor.jsp** se encarga de mostrar el nombre del impartidor logueado en la aplicación y un listado de las actividades impartidas por ese profesor, además de un enlace Asistencia a ese curso. (0.5ptos)

Al pinchar una actividad (enlace), ésta se destaca y se añade una tabla con los alumnos de la actividad. Cada fila muestra: (0.5ptos)

* Datos del alumno
* Fecha formateada de su última asistencia a la actividad elegida
* Hay 2 radios excluyentes para elegir si les avisamos por email o teléfono
* Hay un submit para “anotar” el aviso al alumno elegido

**Control/Servlet: 0.5 pto**

El **ServletAvisos** será el encargado de gestionar las interacciones con la pantalla **impartidor.jsp.** Métodos del servlet:

* Método que carga de lista de las actividades del profesor logueado
* Método que cargar asistencia de la actividad que interesa, se muestra un listado con la información de los alumnos de esa actividad, la fecha de la última asistencia y un radioButon con las opciones Email/Teléfono

**Modelo: 1 ptos**

**ActividadesDao:**

* Método **actividadesImpartidor**, que devuelve una lista con todas las actividades impartidas por un impartidor. (0.5 ptos)
* Método **mapaAsistenciaActividad**, que recibe un identificativo de actividad y devuelve un mapa ordenado que tiene como claves los Alumnos que participan en ella, y como valor la fecha de su última asistencia a la actividad. (0.5 ptos)

**Parte 3: IMPARTIDOR Paso 2: 0.75 ptos**

**Control/Servlet:**

Al pinchar 

* El servlet **ServletAvisos** anota el aviso, añadiendo una línea a un **fichero de texto**, con el formato: Dni\_alumno, fecha de aviso
* El nombre del fichero es parámetro de inicio del del servlet, deberas añadirlo en el web.xml

Tras anotar el aviso, **impartidor.jsp** muestra un mensaje de retroalimentación en la pantalla indicando que el aviso ha sido realizado correctamente.

Siguiendo con el ejemplo, si el profesor decide  por email a “Alazne Moreda”, el fichero contendrá:

78886971F - 26/03/2021 - Teléfono

……………….. - ……………….. - …………..

31123456V - 12/02/2022 - Email