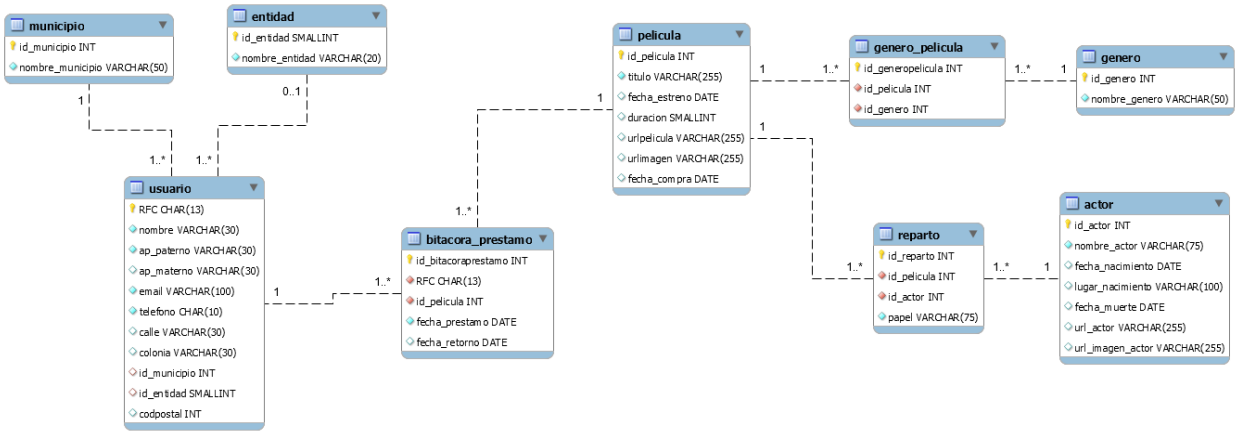
DESARROLLO DE APLICACIONES EN INTERNET

PROGRAMA 8

El programa consiste en realizar la primera parte de un mini sistema de control de préstamos de películas usando servicios PHP para acceder a bases de datos en MySQL a través del modelo PDO (Portable Data Objects). La parte a realizar en este programa es el relacionado con los usuarios a los cuales se les podrá prestar películas. La base de datos a utilizar la crearon para el programa 5 (cuya finalidad era instalar PHP y probar que estuviera listo para acceder a MySQL).



En la figura los campos que tienen una llavecita amarilla a la izquierda son la llave primaria de la tabla, los campos con un rombo azul o rojo son obligatorios (los rojos indican de hecho que son llaves foráneas, es decir, que el valor contenido en estos campos son el valor de una llave primaria en la tabla relacionada, y esta relación se indica con las líneas punteadas que unen diferentes tablas).

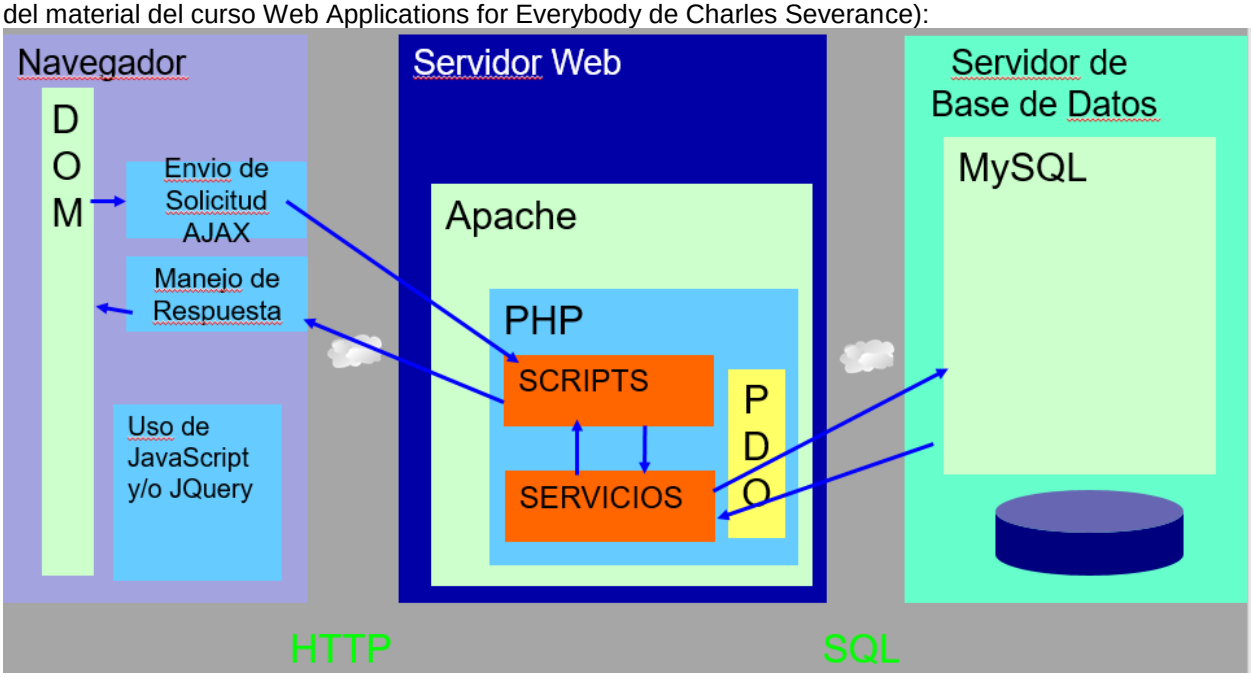
El sistema en su totalidad deberá consistir de un conjunto de servicios PHP y de páginas HTML dinámicas programadas usando HTML, CSS, Javascript y JQuery.

La interface gráfica deberá quedar de acuerdo a las indicaciones dadas para el Programa 7.

Con el propósito de avanzar rápido, se proporciona un código base que será completado en las sesiones del 10 y 17 de junio (debiendo subir la primera parte funcionando via Github al repositorio privado) a más tardar el viernes 12 de junio a las 23:59.

Se hace notar que algunas validaciones no serán realizadas en la parte de Usuarios que desarrollaremos en este programa, pero que deben agregarse en el Programa Final, a saber, verificar que los datos obligatorios estén presentes, que todos los datos tengan un formato y longitud adecuada (tanto del lado del navegador como del lado del servidor), por ejemplo, que el código postal sea de 5 digitos (si es que se proporciona) y el teléfono sea de 10 dígitos obligatoriamente. En general deberían aceptarse en los campos de texto solo hasta el máximo de caracteres de acuerdo a como esté definida la columna correspondiente en la base de datos y si son numéricos solo debe aceptar dígitos

El modelo a usarse para el desarrollo del sistema se muestra en la siguiente figura (basada en una figura del material del curso Web Applications for Everybody de Charles Severance):



En el navegador estará corriendo código en Javascript y/o JQuery que estará tomando datos del DOM para hacer solicitudes asíncronas a ciertos scripts del servidor que verificarán que haya una sesión activa y que de ser así se ayudarán de los servicios, los cuales usando PDO accederán a la base de datos en MySQL. Los scripts del servidor pudieran darle otro formato a la información recibida por parte de los servicios (si fuera de utilidad para el cliente) y se la regresarán al navegador, en donde al recibirse la respuesta habrá una función que se ejecutará para manejar la respuesta, modificando en la mayoría de los casos el DOM de la página para reflejar la información recibida.

La mayor parte del código del sistema en Javascript/Jquery estará contenido en el archivo misfunciones.js (en la subcarpeta js). Además de incluir librerías tales com JQuery, Boostrap o Popper.

Los scripts del servidor (para la primera parte a desarrollar) serán los siguientes:

Nombre del Script	Propósito
config.php	Contener las variables de configuración para el acceso a la base de datos y textos a usar en algunos títulos. DEBE MODIFICAR LO QUE CORRESPONDA EN ESTE SCRIPT
MiPDO.php	Clase personalizada que contendrá un objeto PDO a usarse por los servicios con métodos de uso práctico como run para ejecutar una sentencia SQL y lastInsertId para obtener el ID de un campo autoincrementable al agregar un registro
headdoc.php	Script con la parte a usarse como inicio de todas las páginas del sistema
footer.php	Script con la parte a usar como final de todas las páginas del sistema
barranav.php	Script con el contenido de la barra de navegación a aparecer en todas las páginas del sistema
login.php	Script con el código HTML de una forma de ingreso al sistema

verificalogin.php	Script que verifica que los datos dados en la forma de ingreso al sistema sean válidos (se usan datos estáticos como ejemplo, usuario DesApp, clave UAZej2020). De ser validados los datos se crea una sesión
logout.php	Cierra la sesión activa y redirige a login.php
modalcancelar.php	Script con el código HTML de una forma modal que pide la confirmación de cancelación, antes de incluir este archivo es necesario asignarle a la variable $nomfuncioncancelar el nombre de la función de Javascript a ejecutarse si se confirma la cancelación
modaleliminar.php	Script con el código HTML de una forma modal que pide la confirmación de eliminación, antes de incluir este archivo es necesario asignarle a la variable $nomfuncioneliminar el nombre de la función de Javascript a ejecutarse si se confirma la eliminación
obtenmuns.php	Script que habiendo validado que haya una sesión activa y que se haya dado como parámetro un ID de entidad, obtiene con ayuda del servicio ServicioMunicipio los municipios de una entidad y los regresa al cliente como una serie de etiquetas option a colocarse dentro de un select
obteninfousuario.php	Script que habiendo validado que haya una sesión activa y que se haya dado como parámetro el RFC de un usuario, obtiene la información de ese usuario con ayuda del servicio ServicioUsuario, y se la regresa al cliente como un objeto JSON donde las llaves son los nombres de las columnas de la tabla usuario
eliminausuario.php	Script que habiendo validado que haya una sesión activa y que se haya dado como parámetro el RFC de un usuario, con ayuda del servicio ServicioUsuario, solicita la eliminación de tal usuario en la tabla regresando true o false dependiendo de si se pudo o no
agregausuario.php	Script que habiendo validado que haya una sesión activa y que se haya dado como parámetro un objeto JSON con la información del usuario a agregar, solicita al servicio ServicioUsuario, la creación de tal usuario en la tabla regresando true o false dependiendo de si se pudo o no
actualizausuario.php	Script que habiendo validado que haya una sesión activa y que se haya dado como parámetro un objeto JSON con la información del usuario a actualizar, solicita al servicio ServicioUsuario, la actualización de tal usuario en la tabla regresando true o false dependiendo de si se pudo o no
servicioentidad.php	Contiene la clase ServicioEntidad que deberá implementar los métodos necesarios para acceder a la tabla entidad
serviciomunicipio.php	Contiene la clase ServicioMunicipio que deberá implementar los métodos necesarios para acceder a la tabla municipio

servicioentidad.php	Contiene la clase ServicioUsuario que deberá implementar los métodos necesarios para acceder a la tabla usuario

Los métodos que debería tener los servicios PHP necesarios para la parte de Usuarios y que se encargarán de hacer la conexión a la base de datos serán los siguientes (los nombres pueden variar un poco en su implementación, pero el trabajo que realizan debe ser el indicado:

ServicioUsuario:

- **obtenUsuarios()** Este método devolverá una arreglo con los registros de la tabla usuario, conteniendo cada elemento del arreglo solo los campos de RFC y el nombre completo (como nomcompleto), que es la concatenación de los campos nombre, ap_paterno y ap_materno

- **agregaUsuario(info)** Este método agrega el usuario cuya información está contenida en el objeto JSON info recibido a la tabla usuario (note que puede ser que no se pueda agregar el objeto por lo que regresa true si se pudo o false si no). En el objeto JSON los elementos deben llamarse tal como se llaman las columnas en la tabla usuario (cuidar mayúsculas y minúsculas)
- **eliminaUsuario(RFC)** Este método elimina al usuario de la tabla usuario cuyo valor en el campo llave sea igual al dado como argumento (RFC) (note que puede ser que no se pueda eliminar el objeto por lo que regresa true si se pudo o false si no). Este método debe devolver false en caso de que haya registros en la tabla bitacora_prestamo que hagan referencia al usuario a eliminar y por tanto no eliminarlo.
- **actualizaUsuario(info)** Este método modifica los datos del usuario cuya información está contenida en el el objeto JSON info recibido, los cambios deben quedar reflejados en la tabla usuario (note que puede ser que no se pueda modificar el objeto por lo que regresa true si se pudo o false si no)

ServicioEntidad:

- **obtenEntidades()** Este método devolverá un arreglo con los registros de la tabla entidad de la base de datos (conteniendo cada elemento del arreglo las dos columnas de las que consta el registro)

ServicioMunicipio:

- **obtenMunicipios(idEntidad)** Este método devolverá un arreglo con los registros de la tabla municipio de la base de datos, siempre y cuando pertenezcan a la entidad cuyo ID es recibido como argumento (conteniendo cada elemento del arreglo las dos columnas de las que consta el registro)

ES IMPORTANTE ACLARAR CUALQUIER DUDA SOBRE LAS SESIONES PUES PARA EL PROGRAMA FINAL, A ENTREGARSE EN LA FECHA DE ORDINARIO, SE HARA ALGO SIMILAR PARA **ALGUNAS** DE LAS OTRAS PESTAÑAS DEL SISTEMA DE CONTROL DE PRESTAMOS, CUYA FUNCIONALIDAD SERÁ MUY SIMILAR.

Hagan los commits necesarios para su programa 8 y hagan push del mismo periódicamente. El programa 8 como ya se mencionó arriba será completado durante las sesiones del 10 y 17 de junio y tendrán un par de días extra para corregir problemas que se les pudieran presentar durante las sesiones. El push final debe ser realizado a más tardar a las 23:59 del viernes 12 de junio.