Computación Orientada a Web

Entrega Tema 2 & 3

UNIVERSITAT DE BARCELONA

Sebastian Andrade Zanotti - NIUB17692754

Barcelona, España, 27/03/2022

Apartado 1

Caracteristicas:

Consta de 2 archivos php, el cliente y el servidor.Ambos utilizan la estructura html de la web de la entrega anterior, y lo que se hace en este apartado es cambiar el contenido del centro de la página por un formulario, del lado del cliente, y del lado del servidor se muestra un resumen con la información dada por ese formulario. Siempre con la referencia de la opción del menú de la izquierda “Full Package” marcada como activa, para que se entienda que el usuario está en esta sección.

Funcionalidades:

El formulario consta de 5 campos, 2 de tipo input para introducir el nombre y email, y 3 de tipo select para elegir el hotel, el número de huéspedes, y la fecha. Cuando el usuario presione submit, verá en server.php el resumen de su reserva. Este no interactúa con la base de datos, solo simula.

Comparación con el código ejemplo:

Comparándolo con los fragmentos de código de las diapositivas, una de las cosas que agregamos en el formulario fue la de utilizar max y minlength para el input del nombre, para garantizar a medida de lo posible que del lado del cliente, se introdujera un nombre real. El problema es que al ser un input tipo “text”, no se puede verificar mucho más, y permite utilizar caracteres peligrosos como <powned>, es por esta razón que del lado del servidor se utilizó regex para tratar de evitar este tipo de situaciones.

```
$regexName = "/^[a-zA-z\s]+$/";
$regexMail = "/^[a-zA-z\d\._]+@[a-zA-z\d\._]+.[a-zA-z\d\.]{2,}+$/";
```

Utilizando estas expresiones, como el usuario introduce su nombre y apellido en el mismo campo de input, la expresión regex solo permite caracteres alfabéticos, y \s para permitir espacios (porque es entre el nombre y el apellido), con esto si el usuario introduce <powned> le permitirá hacer el submit, pero del lado del servidor detectará el error, y notificará el problema. También se trato de ser flexible porque al final los nombres no siempre siguen una regla definida.

Para el caso del email pasa algo similar, con la diferencia que utilizamos el tipo “email” en lugar de “mail” para que del lado del cliente ocurra una validación, y evitemos problemas. De todas formas, del lado del servidor también está un regex que verifica que sea un mail de verdad, aquí el problema se presentaba más que nada en que si el usuario quería introducir un email de empresa, como el de la ub u otra compañía, el regexp del servidor era muy restrictivo, entonces se utilizó un formato que permite caracteres alfanuméricos, puntos, y pisos, seguido

por el @, seguido por el mismo primer formato, y por último seguido por “.” y un carácter alfanumérico con un mínimo de 2 caracteres, que puede no tener mucho sentido si es .com, pero si es .uk, .es, o .ub si que tiene sentido.

Por seguridad, no se dice exactamente si está mal el nombre o email, solo se notifica del lado del servidor que uno de los 2 es incorrecto. Del lado del cliente se notificará específicamente si el email es incorrecto, o si el nombre no tiene suficientes caracteres.

También se agregó el parámetro “require” en los 2 inputs y el select para la fecha para que no permita hacer una reservación sin nombre, email, o fecha. Este parámetro no se utilizó para seleccionar el hotel, ni el número de huéspedes porque siempre habrá una opción elegida.

Por ultimo, tambien por seguridad, se utiliza el método POST para que no se vean los parámetros del formulario en la url

Apartado 2

Consta de 3 archivos, client, server y data_base_connect, también está el archivo testing que se utilizó de prueba antes de implementar cosas en la web.

Tienen las mismas características y funcionalidades que el apartado 1, con la diferencia de que se agregaron 2 campos de input de selección. Uno para elegir un país, que está puesto con opciones por defecto, y otro para elegir una ciudad que depende de la base de datos de “world”. Pero igual tiene varias diferencias que explicaré a más detalle. También está el hecho de que ahora aparte de leer de la base de datos, cuando se llega a la página del servidor y se ha confirmado la reserva, se agregara la reserva en una tabla llamada “reservations” que esta en “world.”

Comparación con el código ejemplo:

El input de selección de país, afecta las opciones que mostrará el input de selección de ciudad. Esto se logró utilizando el código de las diapositivas que hace una query y muestra el resultado en una lista, en el archivo “testing.php” tenemos esta implementación para probarlo pero filtrando utilizando el código del país. El problema es que tengo entendido que con php no se puede hacer esta dependencia de que el primer select afecte las opciones del segundo en el mismo formulario sin hacer submit de los datos.

Por tanto, para lograr esto, en la misma página del servidor se utilizó un IF para verificar que la variable de seleccionar ciudad está establecida. En caso que no sea asi, que sera siempre el caso después del primer submit, la acción del form se llamará a sí misma (la pagina) y se modificara el contenido html del centro para que se verifique la información ya introducida (el nombre y el email igual que en el apartado 1) y si todo esta bien, muestra un resumen de la

reserva y pide al usuario seleccionar una ciudad, la cual dependera del pais seleccionado en el cliente.php. Esto es posible gracias al código que se encuentra en testing.php, con la diferencia de que ahora el código de server.php ya no imprime una lista, sino que muestra los resultados de las ciudades como opciones para seleccionar, y como son muchas ciudades se utiliza un LIMIT 5 en la query para que solo muestre las primeras 5 ciudades.

Ahora el usuario podrá elegir la ciudad y confirmar que toda la información está bien, y al hacer submit en “confirm reservation” el IF en server.php será verdad y mostrará el resumen de la reserva, y se agregara exitosamente en la tabla de reservas (que se verá en PHPmyadmin). Como en este segundo submit se pasa información del primer formulario, pasamos todos estos datos como input “hidden,” ya que de no hacer esto al llegar al resumen de la reserva solo tendrá acceso a la ciudad elegida, y no el resto de los datos. También se pasan como hidden porque a estas alturas no queremos dar oportunidad de modificar los datos, ya que se están confirmando, y dejar que los modifique implicaría que pueda introducir algo incorrecto. Si el usuario quiere cambiar algo antes de confirmar, porque cambió de parecer, o porque introdujo algo incorrecto, puede ir atrás y cambiar sus datos. Los datos introducidos deberían seguir ahí, no tendrá que rellenar todo de nuevo.

Por tanto, está el cliente.php con el formulario que luego va a server.php donde el usuario puede elegir la ciudad y confirmar que todo está bien, y luego se muestra el resumen de la reserva y se agrega a la tabla.

Por último, se agregó otro archivo con el nombre “data_basse_connect.php” que está como “require” en ambos ficheros client y server para que se establezca la conexión con la tabla “world” y para crear la tabla “reservations” con un try and catch, para que los profesores no tengan que crearla. Así al ejecutar cliente o servidor, se hará la conexión con “world” luego un try en catch, si la tabla existe no pasara nada, de lo contrario crea la tabla. Al hacer la conexión con “world” se usa el usuario root. En testing.php se utiliza con un usuario y contraseña que cree, pero por facilidad a los profesores se utiliza root en todo el código.

La conexión con las tablas y el try para crear la tabla se hacen en otro archivo tanto por simplificar el código, como por seguridad.

El código de try and catch es similar al de las diapositivas, y el código para crear la tabla se obtuvo de phpmyadmin al crear la tabla, haciendo click en la consola, lo mismo con el insert.

Para gestionar el ID de la tabla “reservations” se utiliza el parámetro extra de AUTO_INCREMENT para que la misma tabla vaya aumentado el número asignado al ID cada vez que se haga una reserva. De la misma manera, cuando se hace el insert a la tabla desde server.php el parámetro de ID no los pasamos porque lo incrementa la misma base de datos.