





**Unai Garcia**

**BG-0140**

**INDICE**

1. **Descripción General 2**

1.1 Entorno de ejecución (requerimientos) 2

1.2 Consideraciones de diseño 2

1.3 Esquema general 4

1.4 Modelo de datos 5

1. **Interfaces Comunes 6**

*2.1 login.html* 6

*2.2 register.html* 7

1. **Interfaces Administrador 9**

*3.1 adminIndex.html* 9

*3.2 Vistas* 11

3.2.1 Alta Usuario (1) 11

3.2.2 Usuarios Registrados (2) 12

3.2.3 Modificar Usuario (3) 16

3.2.4 Solicitudes Alta (4) 18

1. **Interfaces Jugador 21**

*4.1 userIndex.html* 21

*4.2 Vistas* 23

4.2.1 Definir Baraja (1) 23

4.2.2 Seleccionar Baraja (2) 27

4.2.3 Jugar Partida (3) 30

4.2.4 Ver Puntuaciones (4) 33

4.2.5 Cambiar Datos Registro (5) 36

1. **Consideraciones Implementacion 38**

5.1 Principales problemas encontrados 38

* 1. Posibles mejoras/optimizaciones 39

5.3 Conclusiones 40

1. **Anexo I (doc. prac. DHTML)** 4**1**
2. **DESCRIPCION GENERAL**
   1. **Entorno de ejecución:**

Para poder ejecutar la aplicación existen ciertos requisitos a cumplir tanto del lado del servidor como del cliente:

* Lado Servidor:
  + Tener instalado y en ejecución un servicio de gestor de base de datos MySQL con la base de datos de la aplicación definida (i.e. MySQL 5.5.x).
  + Tener instalado el pre-procesador de Hipertexto PHP para la ejecución de scripts en el servidor (i.e. PHP 5.3.x).
  + Tener instalado y en ejecución un servicio de servidor Web, online, con acceso al menos a un directorio local donde residirá la aplicación y que soporte módulos PHP y MySQL (i.e. Apache 2.2.x).
* Lado Cliente:
  + Tener conexión por red al servidor, bien sea red local o Internet.
  + Tener instalado un navegador Web (aplicación verificada en IE8 y FF4).
  + Tener correctamente configurados los parámetros de seguridad en el navegador (i.e. Javascript Habilitado,etc.)
  1. **Consideraciones de diseño:**

Se ha estructurado la aplicación en 3 grupos: *common, user* y  *admin*.

* ***common*** :

Engloba los elementos comunes de la aplicación, esto es, aquellos utilizados independientemente de la naturaleza del usuario (administrador o jugador). Estos son:

- La página de acceso ***login.html*** y el archivo de scripts asociados ***login.js****.*

- La página de registro (solicitudes) ***register.html***, y el archivo de scripts asociados ***register.js.***

- El archivo de scripts del lado del servidor ***commonRequests.php*** asociadas a las dos páginas anteriores.

- El archivo de scripts comunes utilizados en toda la aplicación (lado cliente), ***common.js***.

- La hoja de estilos aplicados tanto a las dos páginas anteriores como a elementos de uso común en toda la aplicación, ***commonStyles.css***.

* ***user*** :

Engloba los elementos de la aplicación involucrados en una sesión de usuario del tipo ‘Jugador’. Estos son:

- La página principal ***userIndex.html***, el archivo de scripts asociados ***userIndex.js****.*y la hoja de estilos asociada a la página y sus distintos tipos de contenido, ***userStyles.css***.

- La página del juego ***juegoCartas.html***, el archivo de scripts asociados ***juegoCartas.js*** y la hoja de estilos asociada al juego ***juegoStyles.css***.

- Los archivos de scripts (además de ***juegoCartas.js***)

para cada tipo de contenido en la página principal: ***definirBaraja.js, elegirBaraja.js,***  ***verPuntuaciones.js y***  ***modificarRegistro.js***.

- El archivo de scripts del lado del servidor ***userRequests.php*** asociadas a las página principal y sus distintos tipos de contenido.

* ***admin*** :

Engloba los elementos de la aplicación involucrados en una sesión de usuario del tipo ‘Administrador. Estos son:

- La página principal ***adminIndex.html***, el archivo de scripts asociados ***adminIndex.js****.*y la hoja de estilos asociada a la página y sus distintos tipos de contenido, ***adminStyles.css***.

- Los archivos de scripts para cada tipo de contenido en la página principal: ***altaUsuarios.js, gestionUsuarios.js,***  ***modificarUsuario.js y***  ***gestionSolicitudes.js***.

- El archivo de scripts del lado del servidor ***adminRequests.php*** asociadas a las página principal y sus distintos tipos de contenido.

Por tanto, el esquema global resultante sería el de la siguiente figura:

* 1. **Esquema General:**

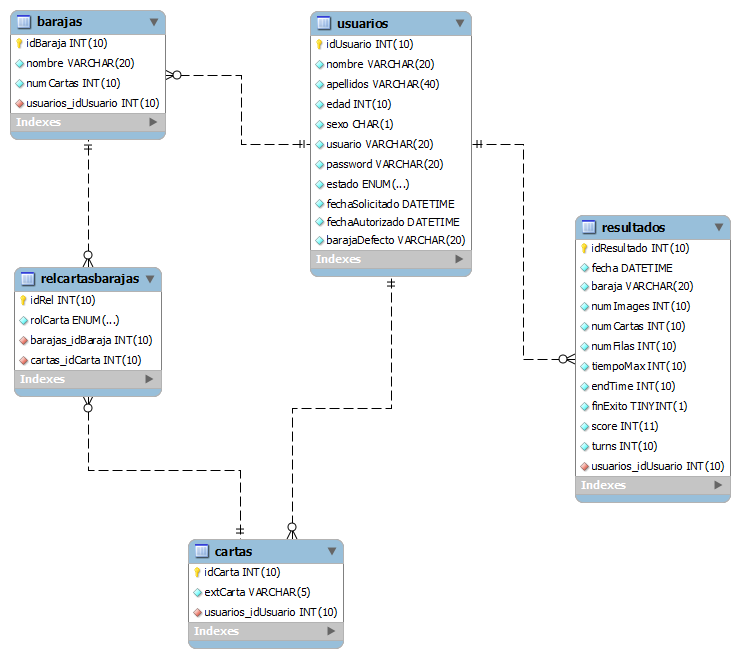


* 1. **Modelo de datos:**

Para el modelo de datos se ha definido una base de datos utilizando el gestor ***MySQL****.*

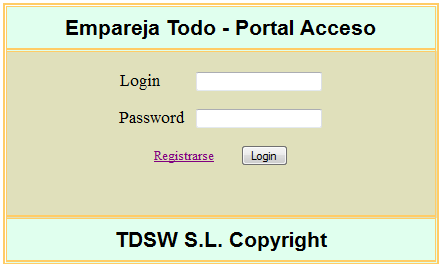
Dicha base de datos está compuesta por distintas tablas relacionadas entre ellas y que dan soporte a los datos relacionados con los usuarios : datos de registro, cartas subidas, barajas definidas y puntuaciones de partidas.

Como resultado se ha definido el modelo de datos de la figura:



1. **INTERFACES COMUNES**
   1. ***login.html*:**

Esta es la página de acceso al portal, donde habrá que autenticarse para obtener acceso. Tiene el siguiente aspecto:



Por tanto las opciones son autenticarse o pinchar el enlace a la página de petición de registro.

El funcionamiento de la autenticación es el siguiente:

1. Se validan los datos introducidos.
2. Se realiza una petición al servidor ( **petición #1** a *‘****commonRequests.php’***) con los datos de autenticación.
3. El servidor comprueba si la autenticación es válida. Si lo es, inicia sesión del usuario en modo administrador o jugador según proceda. Devuelve como respuesta el resultado o fallo en la autenticación.
4. Si ha habido éxito en la autenticación, se reditrecciona a la página que corresponda según el tipo de usuario. Si ha habido algún error, se muestra en la página un mensaje.

Esto obedece al esquema siguiente:



* 1. ***register.html*:**

Esta es la página de petición de registro, donde habrá que rellenar los datos de usuario, enviar y esperar a que un administrador autorice la solicitud, tras lo cual el usuario queda registrado y puede acceder como jugador a la aplicación a través del portal de autenticación. Tiene el siguiente aspecto:



El funcionamiento del registro es el siguiente:

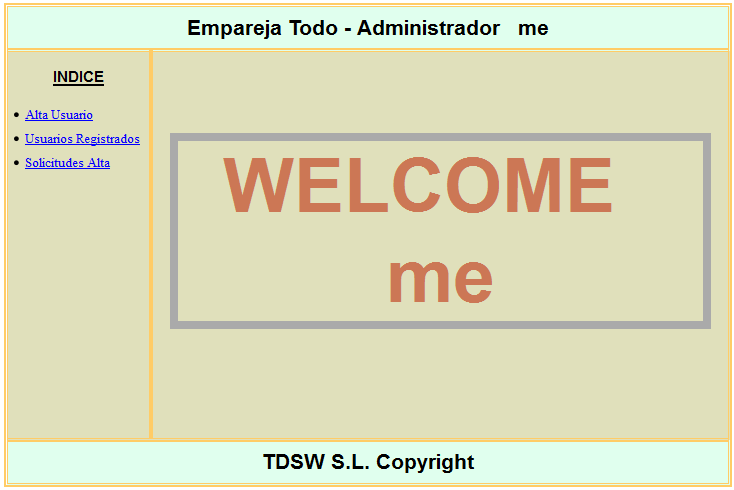
1. Se validan los datos introducidos.
2. Se realiza una petición al servidor ( **petición #2** a *‘****commonRequests.php’***) con los datos de registro.
3. El servidor comprueba primero si el usuario ya existe. Si no existe, crea una nueva entrada en la tabla de usuarios de la base de datos y define su estado como ‘*solicitado*’, de modo que figura como petición sin atender. Devuelve como respuesta el resultado o fallo en la petición de registro.
4. Si ha habido éxito en la petición de registro, se indica al usuario que está pendiente de autorización. Si ha habido algún error, se muestra en la página un mensaje.

Esto obedece al esquema siguiente:



1. **INTERFACES ADMINISTRADOR**
   1. ***adminIndex.html*:**

Esta es la página principal para un usuario de tipo administrador, donde disponemos de un índice con enlaces para escoger el tipo de vista o contenido que se desee. Estas vistas se aplican sobre la celda de contenido ‘*adminContent’*, de manera que el resto de la página permanece estático . Tiene el siguiente aspecto:



El funcionamiento de la página principal es el siguiente:

1. Comprueba si el usuario ha iniciado sesion como administrador mediante petición al servidor (**petición #0**  a *‘****’adminRequests.php’***).
2. Si la respuesta es negativa, se redirecciona a la página de login.
3. Se la respuesta es positiva, se recibe ademá la vista actual de la sesión del usuario (o pantalla de presentación ‘welcome <nombre usuario>’ como en la imagen anterior si no hay vista definida o la anterior sesión expiró).
4. Si la vista está definida, se realiza petición al servidor (**petición #1**  a *‘****’adminRequests.php’***), para que establezca dicha vista en la sesión tal como se hace al pinchar en uno de los enlaces del índice.
5. Se invoca a la función que corresponde con la vista actual (si está definida) tal como se hace al pinchar en uno de los enlaces del índice.

**\*Nota**: La vista 3 (modificar usuario) no es accesible directamente por el índice sino a través de la opción ‘Usuarios Registrados’ al desplegar un usuario de la lista.

Esto obedece al esquema siguiente:



* 1. **Vistas:** 
     1. Alta Usuario:

Esta es la vista donde se puede dar de alta un usuario directamente mediante un formulario de alta similar al de peticiones de registro. La diferencia está en que de este modo no hay petición y el usuario es automáticamente autorizado como jugador.

**\*Nota**: Para dar privilegios de administrador a un usuario, lo debe de realizar un administrador a traves de la vista 3 (*modificar* *usuario*) para un usuario desplegado de la lista de la vista 2 (*usuariosRegistrados*)

Esta vista tiene el siguiente aspecto:



El funcionamiento de la vista es el siguiente:

1. Se muestra el formulario de entrada de datos en la celda de contenido.
2. Se validan los datos introducidos.
3. Se realiza una petición al servidor ( **petición #2** a *‘****adminRequests.php’***) con los datos de registro.
4. El servidor comprueba primero si el usuario ya existe. Si no existe, crea una nueva entrada en la tabla de usuarios de la base de datos y define su estado como ‘*autorizado*, de modo que figura como jugador. Devuelve como respuesta el resultado o fallo en la petición de registro.
5. Se muestra en la página un mensaje con el éxito o error en la operación.

Esto obedece al esquema siguiente:



* + 1. Usuarios Registrados*:*

Esta es la vista donde se muestra un listado de los usuarios registrados (todos los que no están pendientes de autorizar o denegar).

Se definen unos filtros para el listado, así como las cabeceras de los campos son enlaces que permiten cambiar el orden de presentación. Cada vez que se cambie un filtro u orden se lanza una petición al servidor, el cual nos devuelve la lista según el criterio indicado.

Cada entrada de la lista tiene una imagen-enlace que permite desplegar la información completa del usuario en la base de datos, además de la opción de modificar los datos de registro de dicho usuario o eliminarlo.

Además existen casillas de selección (checkBoxes) para poder seleccionar y borrar varios usuarios de una vez con el botón para tal efecto.

**\*Nota**: Uno de los filtros se refiere al estado del usuario (administrador, jugador o con la solicitud denegada). Los respectivos nombres de los filtros son: administrador, usuario y denegado. De este modo se pueden eliminar los usuarios ‘denegados’ si su información es ya irrelevante.

Esta vista tiene el siguiente aspecto:



El funcionamiento de la vista es el siguiente:

1. Se muestra el formulario con el título y filtros de la lista.
2. Se realiza petición al servidor (***peticion#3*** a ‘***adminRequests.php’***) para poblar la lista, según los criterios de filtros y orden especificados (inicialmente no se filtra ninguno y se ordenan descendentemente por fecha en la que fueron autorizados (si no están denegados)).
3. Con los datos recibidos se muestra la lista o el error si lo hubiere.

Y en función de la acción que realice el usuario:

* Cambio en algún filtro:

1. Se actualizan las variables globales (*Javascript*) asociadas a los filtros.
2. Se llama a la función principal de la vista (*usuariosRegistrados()*) para recargar todo el contenido.

* Cambio en algún orden:

1. Se actualizan las variables globales (*Javascript*) asociadas a los órdenes.
2. Se llama a la función principal de la vista (*usuariosRegistrados()*) para recargar todo el contenido.

* Eliminar Usuarios Seleccionados:

1. Con los usuarios seleccionados construye una petición al servidor(***peticion#7*** a ‘***adminRequests.php’***) para eliminarlos de la base de datos.
2. Muestra el resultado de la operación (fallo, éxito) y si es exitosa, tras un tiempo de *delay* , se llama a la función principal de la vista (*usuariosRegistrados()*) para recargar todo el contenido.

* Desplegar Usuario:

1. Activa la fila oculta bajo la entrada desplegada, pasando a ser visible, la cual mostrará la información del usuario.
2. Realiza una petición al servidor(***peticion#4*** a ‘***adminRequests.php’***) para obtener todos los datos del usuario desplegado.
3. Muestra el resultado de la operación (fallo, información). Junto con la información existirán dos botones que permitirán realizar alguna de las acciones siguientes:
   1. – Modificar Usuario : Llama a la **Vista 3** de Administrador (se verá posteriormente) mediante su función principal ***modificarUsuario()***, contenida en ***modificarUsuario.js***.
   2. – Eliminar Usuario: Realiza las mismas acciones que ‘*Borrar usuarios seleccionados’* pero sólo sobre el usuario desplegado.

La siguiente página es un esquema del funcionamiento explicado:



* + 1. Modificar Usuario*:*

Esta es la vista donde se muestra un formulario con los datos de un usuario registrado, pudiendo modificarlos. Tiene el siguiente aspecto:



Si el modo de acceso a esta vista es mediante recarga de página (y comprobación de la vista de la sesión), se llama a otra función distinta de la principal (*modificarUsuario()):*

***modificarUser()*:**

1. Envìa petición al servidor (**petición #8** a *‘****adminRequests.php’***) para así comprobar què usuario se estaba modificando (también en variable de sesión)
2. Una vez comprobado se llama a la función principal como en el caso habitual

El funcionamiento de la vista es el siguiente:

1. Se muestra el formulario vacío y se realiza una petición al servidor (**petición #5** a *‘****adminRequests.php’***) para obtener los datos del usuario.
2. Se muestra o bien el error o los datos .
3. Al pulsar ‘modificar’ se validan los datos modificados.
4. Se realiza una petición al servidor (**petición #6** a *‘****adminRequests.php’***) con los datos de registro modificados.
5. El servidor comprueba primero si el nombre de usuario modificado ya existe. Si no existe, modifica la entrada en la tabla de usuarios de la base de según los parámetros recibidos. Devuelve como respuesta el resultado o fallo en la petición de registro.
6. Se muestra en la página un mensaje con el éxito o error en la operación.

Esto obedece al esquema siguiente:



* + 1. Solicitudes Alta*:*

Esta es la vista donde se muestra un listado de las solicitud registradas (todos los usuarios que están pendientes de autorizar o denegar).

Las cabeceras de los campos son enlaces que permiten cambiar el orden de presentación. Cada vez que se cambie un orden se lanza una petición al servidor, el cual nos devuelve la lista según el criterio indicado.

Cada entrada de la lista tiene una imagen-enlace que permite desplegar la información completa de la solicitud en la base de datos.

Además existen casillas de selección (radioButtons) para poder seleccionar una acción a realizar para cada solicitud (autorizar o denegar). Con el botón de ‘Aplicar Decisiones’ se autorizan o deniegan los usuarios de la lista en función de la acción que tengan escogida (si la tienen).

Esta vista tiene el siguiente aspecto:



El funcionamiento de la vista es el siguiente:

1. Se realiza petición al servidor (***peticion#9*** a ‘***adminRequests.php’***) para poblar la lista, según los criterios de orden especificados (inicialmente se ordenan descendentemente por fecha en la que fueron solicitadas).
2. Con los datos recibidos se muestra la lista o el error si lo hubiere.

Y en función de la acción que realice el usuario:

* Cambio en algún orden:

1. Se actualizan las variables globales (*Javascript*) asociadas a los órdenes.
2. Se llama a la función principal de la vista (*usuariosRegistrados()*) para recargar todo el contenido.

* Aplicar Decisiones:

1. Con las solicitudes que tienen una acción definida, se construye una petición al servidor(***peticion#10*** a ‘***adminRequests.php’***) para actualizarlos en la base de datos.
2. Muestra el resultado de la operación (fallo, éxito) y si es exitosa, tras un tiempo de *delay* , se llama a la función principal de la vista (*solicitudesAlta()*) para recargar todo el contenido.

* Desplegar Petición:

1. Activa la fila oculta bajo la entrada desplegada, pasando a ser visible, la cual mostrará la información del usuario/solicitud.
2. Realiza una petición al servidor(***peticion#4*** a ‘***adminRequests.php’***) para obtener todos los datos de la solicitud desplegada.
3. Muestra el resultado de la operación (fallo, información).

La siguiente página es un esquema del funcionamiento explicado:



1. **INTERFACES JUGADOR**
   1. ***userIndex.html*:**

Esta es la página principal para un usuario de tipo jugador, donde disponemos de un índice con enlaces para escoger el tipo de vista o contenido que se desee. Estas vistas se aplican sobre la celda de contenido ‘*userContent’*, de manera que el resto de la página permanece estático. Tiene el siguiente aspecto:



El funcionamiento de la página principal es el siguiente:

1. Comprueba si el usuario ha iniciado sesion como jugador mediante petición al servidor (**petición #1**  a *‘****’userRequests.php’***).
2. Si la respuesta es negativa, se redirecciona a la página de login.
3. Se la respuesta es positiva, se recibe ademá la vista actual de la sesión del usuario (o pantalla de presentación ‘welcome <nombre usuario>’ como en la imagen anterior si no hay vista definida o la anterior sesión expiró).
4. Si la vista está definida, se realiza petición al servidor (**petición #2**  a *‘****’userRequests.php’***), para que establezca dicha vista en la sesión tal como se hace al pinchar en uno de los enlaces del índice.
5. Se invoca a la función que corresponde con la vista actual (si está definida) tal como se hace al pinchar en uno de los enlaces del índice.

Esto obedece al esquema siguiente:



* 1. **Vistas:** 
     1. Definir Baraja:

Esta es la vista donde se pueden definir y modificar barajas y/o subir imágenes al servidor.

La sección inferior muestra las cartas subidas por el usuario al servidor, mientras que la inmediatamente superior muestra las escogidas (reverso y demás) para la baraja actual (escogida del desplegable o nueva a definir). Las cartas seleccionadas se excluyen de la lista de disponibles automáticamente al seleccionarlas. Para seleccionar/deseleccionar una carta basta con hacer click sobre ella. Para seleccionar el reverso debe estar activada la casilla de verificación en el cuadro de cartas subidas.

La sección superior cuenta con un ‘*select’* con las barajas del usuario para poder seleccionarlas y modificarlas (o bien utilizarlas a modo de plantilla), un campo de texto para introducir el nombre de la baraja nueva a definir y un ‘*input’* de tipo ‘*file’* para subir imágenes al servidor. Además cuenta con los botones para modificar una baraja seleccionada, definir una nueva o subir la(s) imagen(es) seleccionada(s)

Esta vista tiene el siguiente aspecto:



El funcionamiento de la vista es el siguiente:

1. Se muestra el formulario, inicialmente vacío.
2. Se realiza petición al servidor (***peticion#3*** a ‘***userRequests.php’***) para cargar las cartas subidas.
3. Se realiza petición al servidor (***peticion#5*** a ‘***userRequests.php’***) para cargar el ‘*select*’ con las barajas del usuario (llamada a *loadBarajaSelect()*).

Y en función de la acción que realice el usuario:

* Seleccionar una baraja:

1. Se envía petición al servidor(***peticion#6*** a ‘***userRequests.php’***) para obtener las cartas de la baraja seleccionada.
2. Se actualizan las variables globales (*Javascript*) de las cartas seleccionadas.
3. Se llama a las funciones *cargarCartasSubidas() y cargarCartasSeleccionadas()* para recargar el contenido.

* Seleccionar Carta:

1. Las cartas se seleccionan de entre las subidas simplemente haciendo click sobre una. Al seleccionar, dicha carta pasa a estar seleccionada mediante actualización de las var. Globales.
2. Se llama a las funciones *cargarCartasSubidas() y cargarCartasSeleccionadas()* para recargar el contenido.

* Definir Baraja:

1. Con las cartas seleccionadas y el campo para el nombre de baraja nueva, se validan los datos y se construye una petición al servidor (***peticion#4*** a ‘***userRequests.php’)*** para insertar la nueva baraja en la base de datos.
2. Muestra el resultado de la operación (fallo, éxito).
3. Llamada a *loadBarajaSelect()* para recargar el *select* de barajas (ahora con una nueva).

* Modificar Baraja:

1. Con las cartas seleccionadas y el nombre de la baraja elegida, se validan los datos y se construyen dos peticiones al servidor (***peticion#7*** y ***peticion#4*** a ‘***userRequests.php’)*** para borrar e insertar de nuevo (con las nuevas cartas) la baraja en la base de datos.
2. Muestra el resultado de las operaciones (fallos, éxito).
3. Llamada a *loadBarajaSelect()* para recargar el *select* de barajas.

* Subir Carta(s):

1. Las cartas seleccionadas en el *input file* se envian al servidor.
2. El servidor crea entradas para las imágenes/cartas subidas en la base de datos, y con sus índices construye los nombres que utilizará para almacena los archivos. Devuelve el resultado de las operaciones.
3. Muestra el resultado del servidor (errores, éxito) en un **iFrame** definido para así no tener que recargar ninguna página.
4. Llamada a *cargarCartasSubidas()* como consecuencia de la carga del **iFrame**, de este modo se actualizan las cartas subidas cada vez que se sube algún archivo.

**\*Nota**: Se ha definido el *input* file de manera que se puedan seleccionar varios archivos para subir. Esto funciona en FireFox pero en IE no (no se han probado otros exploradores), por lo cual en IE hay que subirlos uno a uno .

La siguiente página es un esquema del funcionamiento explicado:



* + 1. Seleccionar Baraja*:*

Esta es la vista donde se muestra un listado de las barajas definidas por el usuario, donde se podrá borrar barajas y establecer la baraja por defecto.

Se definen unos filtros para el listado, así como las cabeceras de los campos son enlaces que permiten cambiar el orden de presentación. Cada vez que se cambie un filtro u orden se lanza una petición al servidor, el cual nos devuelve la lista según el criterio indicado.

Cada entrada de la lista tiene una imagen-enlace que permite desplegar las cartas de la baraja.

Existen casillas de selección (checkBoxes) para poder seleccionar y borrar varias barajas de una vez con el botón para tal efecto.

También existen casillas de selección (radioButtons) para poder seleccionar la baraja por defecto.

Esta vista tiene el siguiente aspecto:



El funcionamiento de la vista es el siguiente:

1. Se muestra el formulario con el título y filtros de la lista.
2. Se realiza petición al servidor (***peticion#5*** a ‘***userRequests.php’***) para poblar la lista, según los criterios de filtros y orden especificados (inicialmente se filtran entre 3 y MAX cartas y se ordenan descendentemente por el número de cartas que contienen).
3. Con los datos recibidos se muestra la lista o el error si lo hubiere.

Y en función de la acción que realice el usuario:

* Cambio en algún filtro:

1. Se actualizan las variables globales (*Javascript*) asociadas a los filtros.
2. Se llama a la función principal de la vista (*elegirBaraja()*) para recargar todo el contenido.

* Cambio en algún orden:

1. Se actualizan las variables globales (*Javascript*) asociadas a los órdenes.
2. Se llama a la función principal de la vista (*elegirBaraja ()*) para recargar todo el contenido.

* Cambio en la baraja por defecto:

1. Con la baraja seleccionada construye una petición al servidor(***peticion#8***  a ‘***userRequests.php’***) para incluirla como baraja por defecto para el usuario en la base de datos.
2. Muestra el resultado de la operación (fallo, éxito).

* Borrar Barajas Seleccionadas:

1. Con las barajas seleccionados construye una petición al servidor(***peticion#7*** a ‘***userRequests.php’***) para eliminarlas de la base de datos.
2. Muestra el resultado de la operación (fallo, éxito) y si es exitosa, tras un tiempo de *delay* , se llama a la función principal de la vista (*elegirBaraja ()*) para recargar todo el contenido.

* Desplegar Baraja:

1. Activa la fila oculta bajo la entrada desplegada, pasando a ser visible, la cual mostrará las cartas de la baraja.
2. Realiza una petición al servidor(***peticion#6*** a ‘***userRequests.php’***) para obtener todos las cartas de la baraja desplegada.
3. Muestra el resultado de la operación (fallo, cartas).

A continuación un esquema del funcionamiento explicado:



* + 1. Jugar Partida*:*

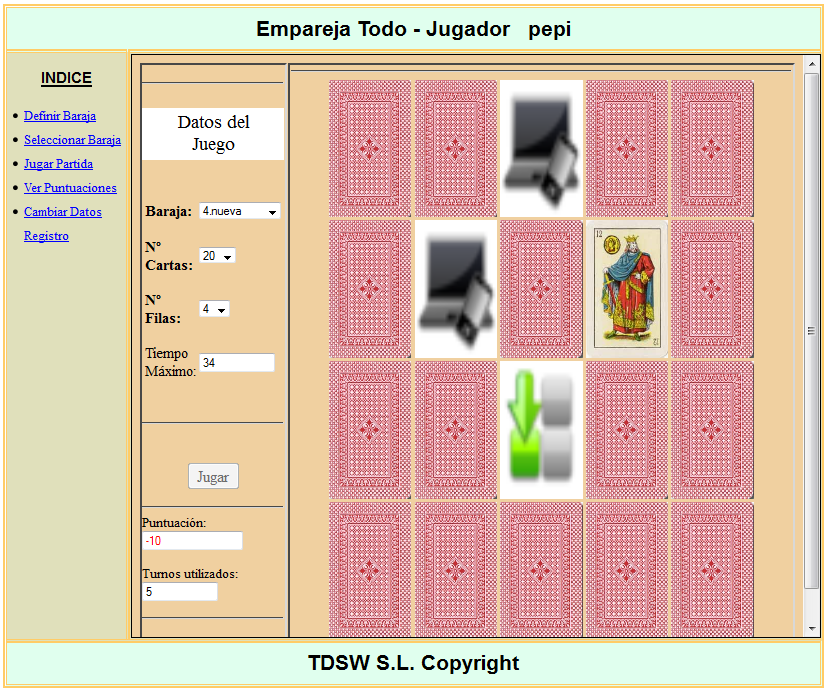
Cuando se selecciona esta vista, se define un ***iFrame*** en el cuadro del contenido, y se carga la página del juego de cartas realizado para la práctica de DHTML (***juegoCartas.html***) en él.

Se han añadido funcionalidades a la página del juego como la posibilidad de poder escoger una baraja para jugar (siempre figurará seleccionada la que esté definida por defecto, también siempre se puede escoger la baraja *–default*-, que es la original del juego).

También se ha añadido comunicación con el servidor para poder guardar las puntuaciones, recibir las cartas de la baraja elegida y recibir las barajas del usuario. Posteriormente se ha dividido el archivo en los correspondientes HTML, CSS y JS para seguir con la estructuración natural de la aplicación.

**\*Nota**: La funcionalidad de la página del juego original (sin los añadidos aquí) está descrita en el **Anexo I**, al final de este documento, donde se ha incluido copia de la documentación que acompañaba en su día a la práctica DHTML. Las funcionalidades añadidas se detallarán a continuación.

La vista tiene el siguiente aspecto:



El funcionamiento de la vista es el siguiente:

1. Se muestra el ***iFrame*** con la página ***juegoCartas.html***.
2. Se realiza petición al servidor (***peticion#5*** a ‘***userRequests.php’***) para obtener las barajas del usuario.
3. Con los datos recibidos se puebla el *select* correspondiente y también los demás elementos del formulario del juego.

Y en función de la acción que realice el usuario:

* Comienzo del juego:

1. Realiza una petición al servidor(***peticion#6*** a ‘***userRequests.php’***) para obtener todas las cartas de la baraja elegida para jugar.
2. Se inicia y transcurre el juego tal y como se definió en la práctica de DHTML (**Anexo I**).

* Fin del juego:

1. Se muestra mensaje de confirmación por pantalla para salvar o no la partida jugada.
2. En caso afirmativo, se realiza petición al servidor (***peticion#9*** a ‘***userRequests.php’***) para guardar los detalles correspondientes en una nueva entrada de la base de datos, asociada al usuario actual.

En la página siguiente se muestra un esquema del funcionamiento explicado:



* + 1. Ver puntuaciones*:*

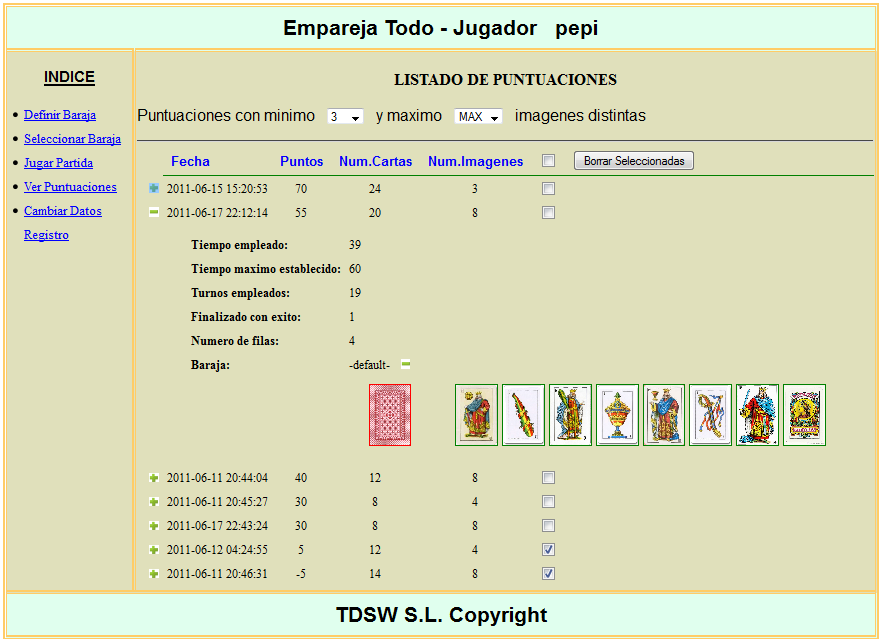
Esta es la vista donde se muestra un listado de las puntuaciones guardadas por el usuario tras las partidas, donde se podrán verlos detalles y/o borrar puntuaciones.

Se definen unos filtros para el listado, así como las cabeceras de los campos son enlaces que permiten cambiar el orden de presentación. Cada vez que se cambie un filtro u orden se lanza una petición al servidor, el cual nos devuelve la lista según el criterio indicado.

Cada entrada de la lista tiene una imagen-enlace que permite desplegar los datos de la partida en cuestión.

Existen casillas de selección (checkBoxes) para poder seleccionar y borrar varias puntuaciones de una vez con el botón para tal efecto.

Esta vista tiene el siguiente aspecto:



El funcionamiento de la vista es el siguiente:

1. Se muestra el formulario con el título y filtros de la lista.
2. Se realiza petición al servidor (***peticion#10***  a ‘***userRequests.php’***) para poblar la lista, según los criterios de filtros y orden especificados (inicialmente se filtran entre 3 y MAX imágenes y se ordenan descendentemente por el número de puntos obtenidos).
3. Con los datos recibidos se muestra la lista o el error si lo hubiere.

Y en función de la acción que realice el usuario:

* Cambio en algún filtro:

1. Se actualizan las variables globales (*Javascript*) asociadas a los filtros.
2. Se llama a la función principal de la vista (*verPuntuaciones()*) para recargar todo el contenido.

* Cambio en algún orden:

1. Se actualizan las variables globales (*Javascript*) asociadas a los órdenes.
2. Se llama a la función principal de la vista (*verPuntuaciones ()*) para recargar todo el contenido.

* Borrar Barajas Seleccionadas:

1. Con las puntuaciones seleccionados construye una petición al servidor(***peticion#12*** a ‘***userRequests.php’***) para eliminarlas de la base de datos.
2. Muestra el resultado de la operación (fallo, éxito) y si es exitosa, tras un tiempo de *delay* , se llama a la función principal de la vista (*verPuntuaciones ()*) para recargar todo el contenido.

* Desplegar Puntuacion:

1. Activa la fila oculta bajo la entrada desplegada, pasando a ser visible, la cual mostrará los detalles de la partida.
2. Realiza una petición al servidor(***peticion#11*** a ‘***userRequests.php’***) para obtener todos los datos de la partida desplegada.
3. Muestra el resultado de la operación (fallo, datos).

* Desplegar Baraja de Partida:

1. Activa la fila oculta bajo la baraja utilizada en el detalle de una partida, pasando a ser visible, la cual mostrará las cartas de la baraja.
2. Realiza una petición al servidor(***peticion#6***  a ‘***userRequests.php’***) para obtener las cartas de la baraja desplegada.
3. Muestra el resultado de la operación (fallo, cartas).

A continuación un esquema del funcionamiento explicado:



* + 1. Modificar Usuario*:*

Esta es la vista donde se muestra un formulario con los datos del usuario de la sesión, pudiendo modificarlos. Tiene el siguiente aspecto:



El funcionamiento de la vista es el siguiente:

1. Se realiza una petición al servidor (**petición #13** a *‘****userRequests.php’***) para obtener los datos del usuario.
2. Se muestra o bien el error o el formulario con los datos .
3. Al pulsar ‘modificar’ se validan los datos modificados.
4. Se realiza una petición al servidor (**petición #14** a *‘****adminRequests.php’***) con los datos de registro modificados.
5. El servidor comprueba primero si el nombre de usuario modificado ya existe. Si no existe, modifica la entrada en la tabla de usuarios de la base de según los parámetros recibidos. Devuelve como respuesta el resultado o fallo en la petición de registro.
6. Se muestra en la página un mensaje con el éxito o error en la operación y si es exitosa, tras un tiempo de *delay* , se llama a la función principal de la vista (*modificarRegistro()*) para recargar todo el contenido.

Esto obedece al esquema siguiente:



1. **CONSIDERACIONES IMPLEMENTACIÓN**
   1. **Principales problemas encontrados:**

Se detallan a continuación las principales dificultades encontradas:

* **Redimensionar contenido**: Se ha tratado de redimensionar los elementos de las páginas al redimensionar la ventana cliente, lo cual no ha sido logrado del todo, sobre todo el ***iFrame*** donde se carga la página del juego (además las incompatibilidades entre navegadores no ayudan (i.e: IE … sin comentarios)).
* **Evitar recargas de página**: Se ha tratado por todos los medios de evitar recarga alguna de la página principal de la aplicación como consecuencia de envío de datos al servidor. Esto en su mayor parte se ha resuelto con peticiones a través del objeto ***XMLHttpRequest***, pero no se ha logrado subir archivos al servidor por este método, por lo cual en ese caso se ha tenido que definir un *iFrame* como objetivo del *script PHP*.
* **Mostrar imágenes dinámicas (esperas)**: Se ha tratado de mostrar una imagen que cambia a intervalos definidos para los tiempos de espera de carga de datos del servidor. Por motivos que no se ha logrado determinar (aunque nunca se llegan a apreciar dado que el servidor responde del orden de *60ms* tras la petición), el sistema de variables globales (*Javascript*) para los intervalos no acaba de funcionar bien, observando cierto acople entre ellos.
* **Subida de varios archivos**: Nada, simplemente que en IE no se ha logrado averiguar cómo lograrlo, por lo cual la subida ha de ser uno a uno.
* **Acceso a controles de formulario**: Sobre todo en el caso de los ***radio Buttons***, no se ha logrado obtener el valor seleccionado en un mismo grupo de controles ‘*radio’*, por lo cual se ha tenido que recurrir a variables globales y eventos ‘*onClick’*. Además, en algún caso no se ha podido acceder correctamente a un elemento de formulario mediante *document.getElementById()* mientras que si se ha podido con *document.<form>.<control>*. (Esto es una de las ‘zonas oscuras’ de la relación *Javascript*🡨🡪*DOM*).
* **Etiquetas y estilos**: Esto más que una ‘zona oscura’ es todavía un ‘abismo’. Se han encontrado múltiples dificultades a la hora de saber qué propiedades específicas **CSS** se aplican a un determinado elemento, su equivalencia con las **HTML**, su acceso mediante **JS**, etc…, llegando a encontrar comportamientos inesperados en algunos componentes. Esto provoca una pérdida de tiempo a veces importante en el *layout* de componentes, por lo cual no es extraño la utilización de software como *DreamWeaver* o *FrontPage* para estas ‘*lindes’*.
  1. **Posibles mejoras / optimizaciones:**

Las posibles optimizaciones son prácticamente infinitas, pero se citan las más inmediatas:

* **Volcado de base de datos:** Tal y como se propuso en la parte opcional, consistiría en un *input file* y sentencias *PHP* para el volcado en el archivo elegido. Habría que utilizar un *iFrame* aquí también para evitar recargas de página.
* **Guardado de datos formularios**: Para el caso en que se recarguen páginas, tener guardados los datos introducidos en el formulario en variables globales (quizá ante eventos *onBlur*) para así volver a mostrarlos después.
* **Redisposición de cartas en el Juego:**  Se pensó en la posibilidad de, una vez dispuestas las cartas del juego, re-disponerlas si se redimensiona la ventana del cliente, guardando previamente en variables su estado actual. Esto podría solucionar los problemas de redimensión del *iFrame* (la altura es el problema).
* **Puntuaciones generales:**  Se podrían mostrar todas las puntuaciones de todos los usuarios a modo de ‘*High Score’* como en los videojuegos, para así promover la competitividad entre jugadores.
* **Ampliación datos entidades:**  Siempre se pueden ampliar los datos de los usuarios registrados (fotografía, email, …) o de otras entidades de la base de datos y mostrarlos / modificarlos en formularios.
* **Seguridad SSL**: Aunque esto es fácilmente implementable en *Apache* a través de su módulo correspondiente (implica obtener y publicar certificado del servidor y requerir certificado cliente), habría que investigar cómo utilizar el certificado del cliente para autenticarlo frente a la base de datos o bien qué funcionalidades ofrecen *JS / PHP* para ello.
* **Cifrado de claves** : Como continuación de lo anterior, sería interesante que el intercambio de claves en la comunicación y su almacenamiento en la base de datos se realizara de forma cifrada para evitar suplantaciones por escuchas en la red o acceso malicioso a la base de datos.
  1. **Conclusiones:**

Se han sacado las siguientes conclusiones tras la realización de la práctica:

* La correspondencia entre etiquetas HTML, sus propiedades CSS, sus atributos y métodos Javascript y su representación en el DOM es, a mi entender, el gran problema actual del desarrollo de aplicaciones web sin acudir a software de terceros que nos abstraigan de ello (como dreamWeaver). A veces te puedes volver ‘loco’ buscando una correspondencia para un elemento y luego no estar seguro de si no funciona porque no se ha invocado bien , porque el navegador no la soporta o porque simplemente su comportamiento no es el esperado o no es la correspondencia adecuada. Al fina todo repercute en la pérdida de un tiempo valioso.
* Las incompatibilidades entre exploradores son otro problema que provoca tener que escribir más código (en el mejor de los casos) o simplemente cambiar su planteamiento. Principalmente hablamos de IE que parece no ajustarse a un estándar al que los demás lo van haciendo (como esto siga así tendremos que recodificar una aplicación completa en VB y .NET sólo para el caso que el explorador del cliente sea IE).
* La funcionalidad del objeto ***XMLHttpRequest*** ha permitido que casi la totalidad de comunicaciones con el servidor sean de forma asíncrona. De este modo se ha tratado de minimizar los datos intercambiados (se podía haber minimizado más aún) de manera que en lugar de obtener una respuesta ya en código *HTML*, se obtienen sólo los datos necesarios para construir dicho código con *Javascript* y así cargar los procesos cliente en lugar de los procesos del servidor.

Por último, y a nivel personal, me he llevado una grata sorpresa al comprobar la capacidad adquirida para hacer sencillas (de momento) aplicaciones con apenas tres meses de docencia. Por tanto no me queda otro remedio que felicitar a los artífices de esto (Javier, Jorge, Santiago) como al departamento OEI por ello. Es sin duda la asignatura en la que más satisfecho he quedado con los conocimientos adquiridos a la vez que es en la que mayor aplicación práctica ‘*real’* se vislumbra a tenor de lo que demandan estos tiempos en lo que a aplicaciones informáticas se refiere.

1. **ANEXO I**

DOCUMENTACIÓN PRÁCTICA DHTML (JUEGO CARTAS)

Los aspectos más destacados del documento html son los siguientes:

Diseño:

El cuerpo del documento se estructura como una tabla de una fila y

dos columnas, presentando en la celda izquierda el formulario con

los correspondientes controles (selects, input text, button, etc.)

y en la derecha el espacio de las cartas del juego. La parte izquierda

tiene asignado un 15% de anchura del total del documeto y el lado

derecho, por tanto, el resto (85%).

Se asignan ids y clases a varias etiquetas para su asociación a estilos

CSS y código Javascript.

Se intenta mediante estilos CSS dejar el aspecto visual lo más parecido

a lo mostrado en el enunciado.

Funcionalidad:

Cuando acaba de cargar el cuerpo del html, se llama a una función

Javascript ***'loadInputs'*** que carga las opciones de los select del

formulario. Se ha definido algunas variables globales para fijar estos

rangos.

Una vez cargados los formularios, con click en el botón 'Jugar' se

invocan las funciones Javascript ***'mostrarCartas'*** y ***'startGame'*** que

muestran las cartas e inician el juego:

***'mostrarCartas'*** --> llamada secuencial a funciones ***'asignarCartas'*** y

***'colocarCartas'***

***'asignarCartas'*** --> Se recorren aleatoriamente las imagenes disponibles

para las cartas y se asignan siempre a una pareja de cartas. Se repite

este proceso hasta que no quedan cartas por asignar. Para ello se declaran

un array de imágenes (local) y uno de cartas (global - conteniendo el

numero de imagen asignada a cada carta). También se declara una variable

global con el número de imágenes.

***'colocarCartas'*** --> Se obtienen las dimensiones de la ventana y los datos

proporcionados en el formulario. En función de ello se calcula todo lo

demás: numero de cartas por fila, anchura y altura de cada carta, etc..

Se ha tratado de aprovechar todo el espacio posible e intercalar filas

de distinto numero de cartas. Se va construyendo una cadena de html con

las imagenes de las cartas para al final asignarselo a la propiedad

'innerHtml' del elemento contenedor.

***'startGame'*** --> Se inhabilitan/inicializan controles varios y se cambia el

valor de la variable global 'finalizado' a 0, indicando así que el juego

ha comenzado. Si se ha introducido un número válido en la casilla de tiempo,

se creará un intervalo de llamadas a ***'countDown'*** cada segundo.

***'countDown'*** --> Decrementa el tiempo en una unidad y comprueba si ha

finalizado. En tal caso, finaliza el intervalo, avisa al jugador y llama

a la funcion ***'finalizar'***

***'finalizar'*** --> Si el intervalo está activo, lo finaliza. Además, activa

los controles desactivados al inicio del juego y pone la variable global

'finalizado' a 1, indicando así que el juego ha finalizado.

Llegados a este punto, el juego ha comenzado, con lo cual al hacer click

sobre una carta se dispara el evento y se llama a la función Javascript

***'voltearCarta'***:

***'voltearCarta'*** --> Primero comprueba que:

- El juego no haya finalizado

- Se hayan tapado las dos cartas (distintas) de la jugada anterior si

la hubiere (esto se controla con la variable global 'esperando')

- La carta no esté ya volteada, claro (mediante un array global).

Si se cumplen estas condiciones, se voltea la carta.

Si la carta en cuestión es:

- La primera de la jugada: Se asigna su número a la variable global

'primeraVolteada' y se prosigue con el juego.

- La segunda de la jugada: Se comprueba si es igual a la

'primeraVolteada':

- Si lo es, marcará 'primeraVolteada' como -1 (la siguiente

será la primera), y llama a la función ***'variarPuntuacion'***

(con parámetro +10) para sumar puntos. Tras esto se comprueba

si están todas volteadas mediante la función ***'todasVolteadas'***,

avisando en su caso al usuario y finalizando el juego mediante

llamada a ***'finalizar'***.

- Si no lo es, se marca 'esperando' a 1 y se establece un

segundo de delay para la llamada a la función ***'taparCartas'***

(con parámetros los números de las dos cartas).

***'variarPuntuacion'*** --> Simplemente varía el valor del control añadiéndole la

cantidad recibida como parámetro. Después comprueba su valor para darle el

color correspondiente.

***'todasVolteadas'*** --> Simplemente recorre el array global para ver si todas

están o no volteadas y devuelve un booleano.

***'taparCartas'*** --> tapa (desvoltea) las dos cartas cuyos números recibe como

parámetro, marca 'primeraVolteada' a -1 y esperando a '0' (para permitir

voltear la siguiente primera carta) y disminuye la puntuación mediante llamada

a la función ***'variarPuntuaion'*** (con parámetro -5).

-------------------------- FIN -----------------------------