Camboya Digital Media

Backend

Technical Specification

Kinder - Speedy Kings

Santiago Rodriguez - santi@camboya.ws
Versión 1.4

Indice

Historial de Versiones

- 1.0 - Lunes 20 de julio de 2009
  - Versión Inicial
- 1.1 - Lunes 21 de julio de 2009
  - Se agregan register y login
- 1.2 - Miercoles 22 de julio de 2009
  - Se corrigen tipos de datos
  - Se corrigen valores requeridos
  - Se corrigen consultas SQL
- 1.3 - Viernes 24 de julio de 2009
  - Se quita de Recommend el parámetro ID
  - Se agrega código de error general: alreadyDone
  - Se corrije formato de la fecha
- 1.4 - Jueves 30 de julio de 2009
  - Se agrega Forgot Password
  - Se eliminan campos en Register

General

Concepto general de este documento

Este documento define lo más precisamente posible las acciones que el backend debe realizar, como así también los parámetros que recibe, las consultas que realiza en la base de datos y la respuesta que genera. Estas especificaciones no son taxativas, sino que pueden ser modificadas si surgieran mejores alternativas para la implementación (por ejemplo, cambiar el nombre a un tag de xml, cambiar el nombre de un parámetro, etc).

Códigos de errores generales

status	statusdescription	Comment
-2	InvalidRequestData	Datos inválidos en el request. Se agregan atributos **field** y **validationmessage**
-1	InternalError	Error general de la aplicación. Se agrega el atributo **message** para mas información
0	OK	Acción exitosa.
1	NotFound	No se encuentra el registro.
2	AlreadyDone	La acción ya fue hecha con anterioridadreg

Esquema

```
<Response status="-1" statusdescription="InternalError" message=”Error de conexión a la base de datos: el usuario no es válido” />
```

Cache

Ninguna respuesta puede ser cacheada salvo que esté explícitamente indicado en el apartado respectivo.

Validación

Para evitar D.O.S. (Deny of Service) se implementará una validación de cada REQUEST.
La misma se generará de la sigueinte manera:

1. Concatenar las variables enviadas en orden alfabético (por nombre de parámetro) utilizando como separador el caracter pipe ("|")
2. Adjuntar como último campo una clave compartida por el Flash y el PHP
3. Aplicar una funcion MD5

El resultante será el campo a enviar como validación.

Ej:
Las variables y sus respectivos valores pueden ser:
name=Santi
points=15
level=4

La clave compartida es: CAMBOYA
Entonces:
str_aux=4|Santi|15|CAMBOYA

Aplicamos MD5:
validation=MD5(str_Aux)

El resultante será enviado en el request y validado en el PHP.

Definición de Datos

games

Campo	Tipo	Null	Default	Extra	Comentario
id (PK)	int	no		auto_increment	
username	varchar(255)	si			
points	int	si			
startdate	datetime	no			Por defecto now()
enddate	datetime	si			Por defecto now()
level	smallint	si	0		
ts	double	no		unsigned	

Backend

Top

Archivo: /php/top.php

Método: GET

Respuesta: XML

Parámetros

Nombre	Tipo	Validación	Default	Requerido
top	smallint	> 0	10	no
validation	string			si

Funcionamiento

Validar los datos recibidos.

Devolver la cantidad de juegos solicitados con el parámetro top ordenados por puntos descendiente.

Consulta de referencia

```
//trae los datos solicitados
SELECT g.username, g.points, DATE_FORMAT(g.enddate, '%d-%m-%Y') enddate, g.level
FROM games g
ORDER BY g.points DESC, g.enddate DESC
LIMIT :top;
```

Esquema

```
<top status="0" statusdescription="0">
       <game username="Guillermo" points="1000" date="16-07-2009" level="4" />
       <game username="Guillermo" points="1000" date="16-07-2009" level="4" />
       <game username="Guillermo" points="1000" date="16-07-2009" level="4" />
       ...
</top>
```

Start

Archivo: /php/start.php

Método: POST

Respuesta: XML

Parámetros

Nombre	Tipo	Validación	Default	Requerido
username	string	< 255 caracteres y existente	-	no
validation	string			si

Funcionamiento

Validar los datos recibidos. Devuelve el id del juego inicializado con fecha/hora actual. Si el username es enviado también lo guarda.

Consulta de referencia

```
//Inicializa el registro
INSERT INTO games (username, startdate, ts) VALUES (:username, now(), :ts);

//Saco el último id
SELECT id from games WHERE ts = :ts;
```

Nota: ts es un timestamp para sacar el último id correspondiente.

Esquema

```
<start status="0" statusdescription="0" id="45" />
```

Save

Archivo: /php/save.php

Método: POST

Respuesta: XML

Parámetros

Nombre	Tipo	Validación	Default	Requerido
username	string	< 255 caracteres y existente	-	si
points	int	>= 0	-	si
level	int	>= 0	-	si
id	int	> 0 y existente	-	si
validation	string			si

Funcionamiento

Validar los datos recibidos. Existiendo el registro con el *id* enviado actualiza los datos.

Consulta de referencia

```
//Actualiza el registro correspondiente
UPDATE games set username = :username, points = :points, level = :level, enddate = now()
WHERE id = :id
```

Esquema

```
<save status="0" statusdescription="0" />
```

Recommend

Archivo: /php/recommend.php

Método: POST

Respuesta: XML

Parámetros

Nombre	Tipo	Validación	Default	Requerido
fromname	string	< 255 caracteres	-	si
fromemail	string	< 255 caracteres - email	-	si
toname	string	< 255 caracteres	-	si
toemail	string	< 255 caracteres - email	-	si
validation	string			si

Funcionamiento

Validar los datos recibidos. Realiza el envío.

Esquema

```
<recommend status="0" statusdescription="0" />
```

Registro

Archivo: /php/register.php

Método: POST

Respuesta: XML

Parámetros

Nombre	Tipo	Validación	Default	Requerido
fname	string	< 255 caracteres	-	no
lname	string	< 255 caracteres	-	no
bdate	string	DD-MM-YYYY	-	no
address	string	< 255 caracteres	-	no
phone	string	< 255 caracteres	-	no
email	string	< 255 caracteres - email		si
username	string	< 255 caracteres		si
password	string	< 255 caracteres		si
validation	string			si

Funcionamiento

Validar los datos recibidos. Valida que no exista un usuario con ese email ni el username. Pasando las validaciones inserta el registro.

Consulta de referencia

```
//Actualiza el registro correspondiente si no existe antes
INSERT INTO usuarios
       SELECT NULL,
              :usuario,
              :apellido,
              :nombre,
              :email,
              :password,
              :edad,
              :fecha_nac,
              :direccion,
              :telefono,
       FROM usuarios
       WHERE usuario = :usuario OR email = :email
       HAVING COUNT(*) = 0
```

Esquema

```
<register status="0" statusdescription="0" />
```

Login

Archivo: /php/login.php

Método: POST

Respuesta: XML

Parámetros

Nombre	Tipo	Validación	Default	Requerido
username	string	< 255 caracteres	-	si
pass	string	< 255 caracteres	-	si
validation	string			si

Funcionamiento

Validar los datos recibidos. Verifica la existencia de usuario y contraseña.

Consulta de referencia

```
//Verifica existencia
SELECT usuario FROM usuarios
WHERE usuario = :username AND password = :pass
```

Esquema

```
<login status="0" statusdescription="0" />
```

Forgot Password

Archivo: /php/forgot.php

Método: POST

Respuesta: XML

Parámetros

Nombre	Tipo	Validación	Default	Requerido
email	string	< 255 caracteres	-	si
validation	string			si

Funcionamiento

Validar los datos recibidos. Verifica la existencia de usuario y envía email.

Consulta de referencia

```
//Verifica existencia
SELECT usuario, password FROM usuarios
WHERE email = :email;
```

Esquema

```
<forgot status="0" statusdescription="0" />
```