Manual Técnico

<—-Sección hecha con el lenguaje PHP-—>

Para la página principal se realizó lo siguiente:

- En la sección `head`, se definen algunos metadatos y se enlazan las hojas de estilo y los archivos JavaScript necesarios para la apariencia y funcionalidad de la página. Se utiliza Bootstrap, una biblioteca de CSS y JavaScript, para aplicar estilos y diseño responsivo al formulario.
- El contenido principal de la página se encuentra dentro de las etiquetas `<body>`. Hay un encabezado (`<header>`) que muestra el logotipo y el título de la página.
- El formulario de inicio de sesión se encuentra dentro de un contenedor (`<div class="container">`). Consiste en dos campos de entrada (`<input>`) para el nombre de usuario y la contraseña, seguidos de un botón de inicio de sesión (`<button>`). También hay un elemento `<p>` para mostrar mensajes de error relacionados con el inicio de sesión.
- El formulario de registro es similar al de inicio de sesión, pero también incluye un campo adicional para confirmar la contraseña.
- En la parte inferior de la página, hay un pie de página (`<footer>`) que muestra el año y el nombre de la aplicación.
- Después del contenido principal, se incluye código PHP dentro de las etiquetas `<?php ?>`. Este código verifica si se han enviado datos de inicio de sesión o registro a través del formulario y realiza acciones en consecuencia. Dependiendo del valor del botón de envío (`$_POST['submit']`), se incluyen otros archivos PHP para procesar el inicio de sesión o el registro del usuario.
- Finalmente, se enlazan los archivos JavaScript necesarios para la funcionalidad adicional de la página, como la interacción con los elementos del formulario.

Para la sección del Dashboard se realizó lo siguiente:

- En la primera sección, se llama a la función `session_start()` para iniciar la sesión. Luego, se verifica si el usuario ha iniciado sesión o no, verificando el valor de `$_SESSION['isLoggedIn']`. Si el usuario no ha iniciado sesión, se redirige al usuario a la página "index.php" utilizando la función `header()`. Esto significa que el acceso a esta página está restringido solo a usuarios que hayan iniciado sesión correctamente.

- Después de la etiqueta `<?php ?>`, se encuentra la estructura básica de una página HTML. En la sección `<head>`, se definen metadatos y se enlazan las hojas de estilo y los archivos JavaScript necesarios para la apariencia y funcionalidad de la página. Se utiliza Bootstrap y se incluyen algunos archivos JavaScript personalizados (`dashboard.js`). También se establece un favicon para la página.
- El contenido principal de la página se encuentra dentro de las etiquetas `<body>`. Hay un encabezado (`<header>`) que muestra el logotipo y el título de la página, y un menú desplegable para cerrar sesión. El menú desplegable tiene un enlace que apunta a "processes/logout.php" para cerrar la sesión del usuario.
- Después del encabezado, dentro de la etiqueta `<main>`, hay código PHP incrustado que recupera información de la sesión actual. Se asignan los valores de `$_SESSION['username']` y `$_SESSION['avatars']` a las variables `$username` y `$avatars`, respectivamente. Luego, se muestran estos valores en la página utilizando etiquetas HTML y se ejecuta una función JavaScript (`SetNumAvatares(3)`).
- Dentro de la etiqueta `<div class="container">`, hay un contenedor que contiene una fila (`<div class="row">`). El contenido de esta fila puede variar dependiendo del código PHP o JavaScript adicional. En este caso, el contenido se carga dinámicamente utilizando la función JavaScript `LoadAvatarCards()`.
- Al final del documento, en la etiqueta `<footer>`, se muestra el año y el nombre de la aplicación.
- Finalmente, se incluye más código PHP dentro de las etiquetas `<?php ?>`. Aquí, se ejecuta una función JavaScript (`LoadAvatarCards()`) y se incluye otro archivo PHP (`loadavatardashboard.php`) para cargar los avatares en el tablero.
-

Con respecto a la creación del avatar decidimos hacerlo de esta manera:

1. En la primera línea, se llama a la función `session_start()` para iniciar la sesión. Esto permite que se utilicen variables de sesión en el código.
2. A continuación, hay una condición `if` que verifica si el usuario ha iniciado sesión (`$_SESSION['isLoggedIn']`). Si el usuario no ha iniciado sesión, se redirige al usuario a la página "index.php". Esto se hace usando la función `header()` con la ubicación de redirección.
3. Luego sigue el código HTML que representa la estructura de la página web de personalización de avatares. Contiene un encabezado (`header`), una sección principal

con opciones de personalización y contenedores de imágenes, y un pie de página (`footer`).

4. En el encabezado, se muestra un logotipo y un título.
5. En la sección principal, hay un formulario con diferentes campos de selección (`select`) que permiten al usuario personalizar el avatar. Los campos incluyen opciones para seleccionar el color de la piel, la vestimenta, el estilo de pelo y los accesorios del avatar.
6. Después del formulario, hay tres botones: "Aleatorio", "Guardar y Salir" y "Descargar". Estos botones llaman a funciones JavaScript definidas en un archivo llamado "avatar.js" que se carga previamente. Estas funciones se encargan de realizar acciones como generar un avatar aleatorio, guardar el avatar y descargarlo.
7. Luego viene el pie de página, que muestra el año actual y el nombre de la aplicación de creación de avatares.
8. Al final del archivo, se incluyen dos archivos PHP adicionales: "loadcosmetics.php" y "loadAvatar.php". Estos archivos contienen procesos de carga de cosméticos (como colores de piel, vestimenta, etc.) y la carga del avatar previamente guardado, respectivamente.

Ahora se explicará la sección de procesos:

Connect:

- Comienza con un bloque de comentarios que muestra los detalles de la conexión a MySQL.
- Luego, se define una conexión a una base de datos Oracle. Se especifican el host, el puerto, el SID (identificador del sistema) y el tablespace deseado.
- Se crea una cadena de conexión utilizando los valores anteriores.
- A continuación, se realiza la conexión a la base de datos Oracle utilizando la función `oci_connect()`. Se pasan como argumentos el nombre de usuario, la contraseña y la cadena de conexión.
- Si la conexión no se establece correctamente, se muestra un mensaje de error y se detiene la ejecución del script.
- A continuación, se ejecuta una consulta para establecer el esquema o tablespace predeterminado utilizando la sentencia `ALTER SESSION SET CURRENT_SCHEMA`. La consulta se prepara con `oci_parse()`, se ejecuta con `oci_execute()` y se establece el tablespace especificado.

Load Avatar (Cargar Avatar):

- La línea `<script src="../js/avatar.js"></script>` incluye un archivo JavaScript llamado "avatar.js", que probablemente contiene funciones relacionadas con la manipulación y visualización de avatares en la página.
- A continuación, se inicia una sección de código PHP. En primer lugar, se obtiene un valor `avatar` a través del método GET, que se asigna a la variable `$nAvatar`.
- Se declaran varias variables para almacenar IDs y nombres relacionados con el avatar.
- Se crea un nombre de variable de sesión utilizando el valor `$nAvatar` y se obtiene el ID del avatar almacenado en esa variable de sesión.
- Se muestra una etiqueta de script que llama a una función JavaScript `setAvatarID($avatarID)` para establecer el ID del avatar en el código JavaScript.
- Luego, se ejecutan una serie de consultas SQL utilizando el objeto de conexión `$conn` para obtener diferentes IDs y nombres relacionados con el avatar.
- Después de cada consulta, se comprueba si se obtiene un resultado válido y se asigna el valor correspondiente a la variable correspondiente (`$ID_color`, `$ID_head`, `$ID_torso`, `$ID_accessory`, `$color`, `$head`, `$torso`, `$accessory`).
- Finalmente, se muestra otra etiqueta de script que llama a una función JavaScript `loadAvatar('$color', '$torso', '$head', '$accessory')` para cargar el avatar con los nombres obtenidos en la página.

Load Avatar (Dashboard):

- La línea `<script src="../js/dashboard.js"></script>` incluye un archivo JavaScript llamado "dashboard.js", que probablemente contiene funciones relacionadas con la manipulación y visualización de avatares en el panel de control.
- A continuación, se incluye el archivo "connect.php" utilizando `require_once`, lo que sugiere que este archivo contiene la configuración de conexión a la base de datos.
- Se declaran varias variables para almacenar IDs y nombres relacionados con los avatares.
- Luego, se inicia un bucle `for` que se ejecutará tres veces, ya que el bucle recorre los valores de `1` a `3`. Esto indica que se esperan tres avatares en el panel de control.

- En cada iteración del bucle, se construye un nombre de variable de sesión utilizando el valor `$i` y se obtiene el ID del avatar almacenado en esa variable de sesión.
- Se ejecutan una serie de consultas SQL utilizando el objeto de conexión `$conn` para obtener diferentes IDs y nombres relacionados con el avatar actual.
- Después de cada consulta, se comprueba si se obtiene un resultado válido y se asigna el valor correspondiente a la variable correspondiente (`$ID_color`, `$ID_head`, `$ID_torso`, `$ID_accessory`, `$color`, `$head`, `$torso`, `$accessory`).
- Finalmente, se muestra una etiqueta de script que llama a una función JavaScript `loadAvatarsImages($i, '$color', '$torso', '$head', '$accessory')` para cargar las imágenes de los avatares en el panel de control, pasando los valores correspondientes a la función.

Load Cosmetics (Cargar Cosméticos):

- La línea `<script src="../js/avatar.js"></script>` incluye un archivo JavaScript llamado "avatar.js", que probablemente contiene funciones relacionadas con la manipulación y visualización de avatares.
- A continuación, se incluye el archivo "connect.php" utilizando `require_once`, lo que sugiere que este archivo contiene la configuración de conexión a la base de datos.
- El código realiza consultas SQL a la base de datos para obtener diferentes nombres de elementos relacionados con la personalización del avatar, como cabello, colores, torsos y accesorios.
- Para cada tipo de elemento, se ejecuta una consulta SQL para obtener los nombres correspondientes de la base de datos.
- Después de ejecutar la consulta, se crea un arreglo llamado `$names` y se recorren los resultados obtenidos de la consulta utilizando un bucle `while`. En cada iteración, se agrega el nombre obtenido al arreglo `$names`.
- Luego, se recorre el arreglo `$names` utilizando un bucle `foreach`. En cada iteración, se muestra una etiqueta de script que llama a una función JavaScript específica para cargar el elemento correspondiente en la página web. El nombre del elemento se pasa como argumento a la función.
- Esto se repite para cada tipo de elemento (cabello, colores, torsos y accesorios) obteniendo los nombres correspondientes de la base de datos y llamando a las funciones JavaScript adecuadas para cargar los elementos en la página.

Login:

- La línea `<script src="../js/index.js"></script>` incluye un archivo JavaScript llamado "index.js", que probablemente contiene funciones relacionadas con la interacción del formulario de inicio de sesión.
- A continuación, se obtienen los datos de inicio de sesión del formulario utilizando `$_POST`. Se obtienen el nombre de usuario y la contraseña ingresados por el usuario.
- La contraseña se convierte en mayúsculas y se le aplica una función de hash (en este caso, MD5) utilizando `strtoupper(md5($password))`. Esto sugiere que la contraseña almacenada en la base de datos también está en mayúsculas y se ha aplicado el mismo hash.
- Se realiza una consulta SQL a la base de datos para buscar un usuario que coincida con el nombre de usuario y la contraseña proporcionados. La consulta compara el nombre de usuario y la contraseña con los valores en la tabla "USERS".
- Si se encuentra un usuario coincidente en la base de datos, se inicia una sesión utilizando `session_start()`. Se guardan varias variables de sesión, como el nombre de usuario, el estado de inicio de sesión (`isLoggedIn`) y los avatares asociados al usuario.
- Luego, se realiza otra consulta para obtener los identificadores de los avatares asociados al usuario. Los identificadores se almacenan en un arreglo llamado `$pkList`.
- A continuación, cada identificador de avatar se almacena en variables de sesión separadas (`$_SESSION['AVATAR_1']`, `$_SESSION['AVATAR_2']`, `$_SESSION['AVATAR_3']`). Si no hay suficientes identificadores, se asigna `null` a las variables de sesión correspondientes.
- Finalmente, si se autentica con éxito, se redirige al usuario a la página "dashboard.php" utilizando `header("Location: dashboard.php")`. De lo contrario, si no se encuentra un usuario coincidente en la base de datos, se muestra un mensaje de error utilizando una función JavaScript llamada `login_error()`.

Log out:

1. `session_start();` inicia la sesión actual o reanuda una sesión existente.
2. `session_destroy();` destruye todos los datos de la sesión actual. Esto incluye eliminar todas las variables de sesión almacenadas y finalizar la sesión actual. Después de esta línea, se considera que la sesión ha sido cerrada.

3. `header("Location: ../index.php");` redirige al usuario a la página "index.php" ubicada en el directorio anterior (`../`). Esta línea utiliza la función `header()` para enviar una nueva cabecera HTTP al navegador y redirigirlo a la página especificada.

En resumen, este código PHP finaliza la sesión actual, eliminando todos los datos de la sesión, y luego redirige al usuario de regreso a la página de inicio "index.php".

Save Avatar (Guardar Avatar):

1. `<script src="../js/avatar.js"></script>` carga el archivo JavaScript "avatar.js" ubicado en el directorio "../js". Esto proporciona funcionalidad adicional al código.
2. `session_start();` inicia la sesión actual o reanuda una sesión existente. Esto permite el uso de variables de sesión para almacenar y acceder a datos durante la sesión.
3. `require_once "connect.php";` importa el archivo "connect.php", que contiene la lógica de conexión a la base de datos.
4. `$avatarID = $_POST['AvatarID'];` recupera el valor del parámetro "AvatarID" enviado a través de una solicitud POST y lo almacena en la variable `$avatarID`.
5. Se definen las variables `$ID_color`, `$ID_head`, `$ID_torso` y `$ID_accessory` como nulas.
6. Las variables `$color`, `$head`, `$torso` y `$accessory` se inicializan con los valores enviados a través de la solicitud POST en los parámetros correspondientes (colorSelected, hairSelected, clothesSelected, accessorySelected).
7. Se utiliza la función `str_replace()` para reemplazar los espacios en blanco en las variables `$color`, `$head`, `$torso` y `$accessory` por guiones ("-"). Esto se hace para formatear correctamente los nombres de archivo.
8. Se agrega la extensión ".png" a las variables `$color`, `$head`, `$torso` y `$accessory` para formar los nombres de archivo completos.
9. Se ejecutan consultas SQL para obtener los ID correspondientes a los nombres de color, torso, cabeza y accesorio especificados.
10. Si las consultas tienen resultados, los ID se asignan a las variables `$ID_color`, `$ID_torso`, `$ID_head` y `$ID_accessory`, respectivamente.
11. Se construye una consulta SQL que utiliza los valores de `$avatarID`, `$ID_color`, `$ID_head`, `$ID_torso` y `$ID_accessory` para actualizar el avatar en la base de datos.
12. Se ejecuta la consulta SQL utilizando `oci_execute()`.

Save Cosmetics (Guardar Cosméticos):

1. `require_once "connect.php";` importa el archivo "connect.php", que contiene la lógica de conexión a la base de datos.
2. El código se encarga de insertar nuevos registros en diferentes tablas de la base de datos para los elementos de color, cabello, torso y accesorios de un avatar.
3. Para cada tipo de elemento (colors, hair, torsos, accessories), se especifica una carpeta donde se encuentran las imágenes correspondientes en el servidor.
4. Se utiliza la función `scandir()` para obtener una lista de archivos en la carpeta especificada.
5. Se prepara una consulta SQL utilizando una sentencia PL/SQL para insertar el nombre de un archivo en la tabla correspondiente. Por ejemplo: `'BEGIN AVATARCREATOR_DBA.INSERT_INTO_COLORS(:name); END;'`.
6. Se utiliza `oci_parse()` para preparar la consulta SQL en una declaración.
7. Se recorre la lista de archivos y se realiza lo siguiente:
    - Se verifica que el archivo no sea "." o ".." para evitar procesar los directorios actuales y superiores.
    - Se ejecuta una consulta SQL para verificar si el nombre de archivo ya existe en la tabla correspondiente.
    - Si el resultado de la consulta es 0, significa que el archivo no está en la tabla y se procede a realizar la inserción.
    - Se asigna el nombre de archivo a la variable `:name` en la consulta preparada utilizando `oci_bind_by_name()`.
    - Se ejecuta la consulta utilizando `oci_execute()`.

Sign Up:

1. Importa el archivo JavaScript "index.js" utilizando la etiqueta `<script>`.
2. Recibe los datos del formulario de registro a través de `$_POST`.
3. Verifica las siguientes condiciones para validar los datos ingresados:
    - Utiliza una expresión regular para verificar que el nombre de usuario contenga al menos 4 letras y no contenga caracteres especiales.

- Compara la contraseña y la confirmación de contraseña para asegurarse de que coincidan.
- Verifica que la longitud de la contraseña sea de al menos 8 caracteres.

4. Si alguna de las condiciones no se cumple, se muestra un mensaje de error mediante la función JavaScript `signup_error()` y se detiene la ejecución del código utilizando `exit()`.
5. Si se cumplen todas las condiciones, se realiza una consulta SQL para verificar si el nombre de usuario ya existe en la base de datos.
6. Si el nombre de usuario ya existe, se muestra un mensaje de error.
7. Si el nombre de usuario no existe, se encripta la contraseña utilizando `md5()` y se realiza una consulta SQL para insertar el nuevo usuario en la base de datos.
8. Si la inserción es exitosa, se incluye el archivo "login.php" para redirigir al usuario a la página de inicio de sesión.
9. Si ocurre algún error durante el proceso, se muestra un mensaje de error.

Update Avatars (Actualizar Avatares):

1. Inicia una sesión utilizando `session_start()` para mantener la sesión actual.
2. Importa el archivo "connect.php" que contiene la conexión a la base de datos.
3. Obtiene el nombre de usuario y el número de avatares seleccionados del arreglo `$_SESSION` y `$_POST` respectivamente.
4. Realiza una consulta SQL para actualizar el número de avatares del usuario en la base de datos utilizando el procedimiento almacenado "UPDATE_USERS_AVATARS".
5. Ejecuta la consulta SQL utilizando `oci_execute()` y guarda el resultado en la variable `$result`.
6. Si la ejecución de la consulta es exitosa, se actualiza la variable de sesión `$_SESSION['avatars']` con el nuevo número de avatares.
7. Si ocurre algún error durante la ejecución de la consulta, se muestra un mensaje de error en la consola del navegador a través de `console.log()`.

En resumen, este código actualiza el número de avatares seleccionados por un usuario en la base de datos y actualiza la variable de sesión correspondiente. Si hay algún error, se muestra un mensaje de error en la consola.

<—-Sección hecha con el lenguaje JavaScript-—>

Avatar:

- Las primeras líneas definen variables globales, como `AvatarID` y las funciones `changeColor`, `changeHair`, `changeClothes` y `changeAccessory`.
- A continuación, se definen varias funciones, como `LoadColors`, `LoadClothes`, `LoadHair` y `LoadAccessories`, que se utilizan para cargar opciones en los elementos `<select>` del formulario. Estas funciones toman un nombre como parámetro, lo manipulan y crean un elemento `<option>` con el nombre ajustado y lo agregan al elemento `<select>` correspondiente.
- Luego, hay funciones como `ChangeColor`, `ChangeHair`, `ChangeTorso`, `ChangeAccessory` y `RandomAvatar` que se llaman en respuesta a ciertos eventos en la página. Estas funciones actualizan las imágenes del avatar en función de las opciones seleccionadas por el usuario.
- La función `DownloadAvatar` se utiliza para generar una imagen descargable del avatar actual. Crea un lienzo `<canvas>` y dibuja las imágenes del avatar en él. Luego, genera una URL de datos de la imagen resultante y crea un enlace `<a>` con la URL de datos para que el usuario pueda descargar la imagen.
- La función `loadAvatar` se utiliza para cargar un avatar previamente guardado. Toma los parámetros de color, ropa, cabello y accesorios y los utiliza para seleccionar las opciones correspondientes en los elementos `<select>` del formulario y actualizar las imágenes del avatar.
- La función `SaveAvatar` se utiliza para guardar el avatar personalizado en el servidor. Realiza una solicitud de POST a un archivo PHP y envía los parámetros seleccionados del avatar y el ID del avatar. Después de manejar la respuesta del archivo PHP, redirige al usuario a otra página.

Dashboard:

1. La variable `nAvatares` se inicializa con el valor 0 y se utiliza para realizar un seguimiento del número de avatares en la página.
2. La función `SetNumAvatares` se utiliza para establecer el valor de `nAvatares` con un número específico.
3. La función `addCard` se ejecuta cuando se quiere agregar una nueva tarjeta de avatar. Crea un nuevo elemento `div` con una clase y un ID únicos, y asigna valores a las variables `newDiv` y `altValue`. Luego, establece el contenido HTML del nuevo div con una

plantilla de cadena de texto que incluye imágenes y botones. Finalmente, agrega el nuevo div al contenedor existente en la página.

4. La función `editCard` se utiliza para redirigir al usuario a la página "avatar.php" junto con el parámetro "avatar" en la URL. Este parámetro representa el avatar que se va a editar.
5. La función `LoadAvatarCards` se utiliza para cargar todas las tarjetas de avatar en la página. Almacena el valor actual de `nAvatares` en la variable `n`, establece `nAvatares` en 0 y luego ejecuta la función `addCard` `n` veces para crear las tarjetas correspondientes.
6. La función `loadAvatarsImages` se utiliza para cargar las imágenes de los avatares en las tarjetas correspondientes. Recibe el número de avatar (`n`) y los nombres de los estilos de color, ropa, cabello y accesorios. Luego, actualiza las imágenes de cada tarjeta de avatar utilizando el ID y la ruta de la imagen.

Index:

Este código contiene dos funciones relacionadas con la manipulación y visualización de mensajes de error en un formulario de inicio de sesión y registro en una página web. Aquí está una breve explicación de cada función:

1. La función `login_error` se utiliza para mostrar un mensaje de error en el formulario de inicio de sesión. Primero, obtiene el elemento `paragraph` con el ID "login_error". Luego, actualiza el contenido HTML del elemento `paragraph` con el mensaje de error "(!) Los Credenciales son incorrectos". Además, obtiene el elemento `paragraph` con el ID "signup_error" y lo establece en un valor vacío para borrar cualquier mensaje de error previo relacionado con el formulario de registro.
2. La función `signup_error` se utiliza para mostrar un mensaje de error en el formulario de registro. Recibe un parámetro `message` que representa el mensaje de error específico. Primero, obtiene el elemento `paragraph` con el ID "login_error" y lo establece en un valor vacío para borrar cualquier mensaje de error previo relacionado con el formulario de inicio de sesión. Luego, obtiene el elemento `paragraph` con el ID "signup_error" y actualiza su contenido HTML con el valor del parámetro `message` proporcionado.