



IES GASPAR MELCHOR DE
Jovellanos

PROGRAMACIÓN WEB EN ENTORNO SERVIDOR

PROYECTO CORDOVA FIREBASE

AUTORES

CICLO FORMATIVO DE GRADO SUPERIOR DE DESARROLLO DE
APLICACIONES WEB

Juan Chira, Mhd Nour Sendyan y Foad Serroukh

Tabla de contenido:

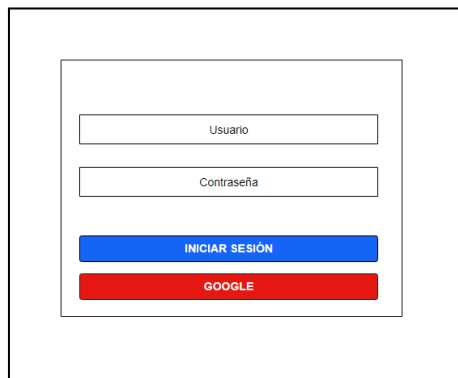
Mockup:	3
Inicio de sesión:	3
Registro:	3
Antes de iniciar sesión:	4
Después de iniciar sesión:	4
Google Authentication:	5
Inicio de sesión con GitHub:	6
Registrarse con correo y contraseña:	7
Inicio de sesión con correo y contraseña:	9
Imprimir la tabla desde la base de datos realtime:	10
Cerrar sesión:	11
Casos de uso:	13
Página principal sin inicio de sesión:	13
Registrarse:	13
Correo inválido:	14
Contraseña débil:	14
Inicio de sesión con correo y contraseña:	15
Correo no encontrado:	15
Contraseña incorrecta:	16
Inicio de sesión con Google:	16
Inicio de sesión con GitHub:	17
Resultado final:	17

La Liga Santander

Mockup:

Este punto contiene los Mockups realizados previamente a la creación del proyecto, con un diseño acordado entre los tres miembros del equipo

Inicio de sesión:



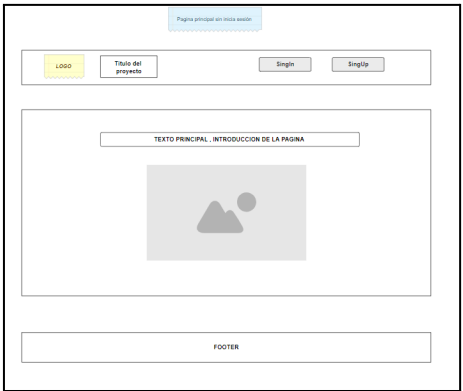
A login form mockup enclosed in a black rectangular border. Inside, there is a white rectangular container. Within this container, there are two input fields: the top one is labeled 'Usuario' and the bottom one is labeled 'Contraseña'. Below these fields are two buttons: a blue button labeled 'INICIAR SESIÓN' and a red button labeled 'GOOGLE'.

Registro:

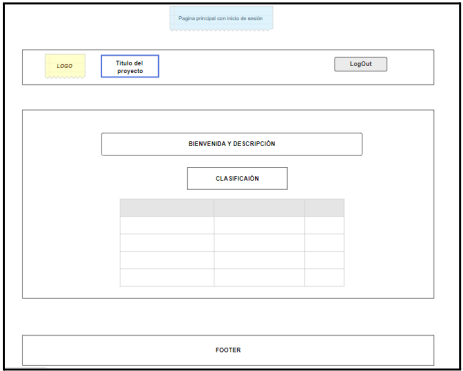


A registration form mockup enclosed in a black rectangular border. Inside, there is a white rectangular container with a blue border. In the top right corner of this container is a small 'x' icon. The container contains two input fields: the top one is labeled 'Correo electronico' and the bottom one is labeled 'Contraseña'. Below these fields is a blue button labeled 'REGISTRARSE'.

Antes de iniciar sesión:



Después de iniciar sesión:



Este proyecto usa un script que utiliza **Firestore** para la **autenticación** de usuarios y almacenamiento de información en una base de datos en tiempo real generada por el propio **Firestore**.

La primera parte del código importa las diferentes funciones y herramientas de **Firestore** necesarias para su ejecución, como **Firestore Authentication** y **Firestore Database**.

Luego se define la configuración de **Firestore** en `firebaseConfig`, que contiene información como la clave de API, el dominio de **autenticación** y la URL de la base de datos.

A continuación, se inicializan los servicios de **Firestore** a través de las variables `app`, `auth` y `db`.

Google Authentication:

Se define la función `onDeviceReady()`, que se ejecuta cuando el dispositivo está listo para su uso. Esta función agrega un `eventListener` al botón de inicio de sesión con `Google` y llama a la función `getRed()` para obtener el resultado de la **autenticación**.

La función `getRed()` maneja el resultado de la **autenticación** con Google, mediante el uso de la función `getRedirectResult()` de **Firestore Authentication**. Si la **autenticación** es exitosa, el código guarda la información del usuario en la base de datos y muestra la página principal de la aplicación, de lo contrario, muestra un **error**. También cambia la página de inicio de sesión a la página principal de la aplicación mediante el uso de las clases CSS `d-none` y `d-block`. También agrega el nombre del usuario autenticado a la página principal con una bienvenida y llama a la función `getClasificacionEquipos()` para obtener información de la base de datos y mostrarla en la aplicación.

Además de la **autenticación** y el almacenamiento de datos, también muestra cómo interactuar con la interfaz de usuario de la aplicación.

Además, la función `getClasificacionEquipos()` utiliza la función `onChildAdded()` de **Firestore Database** para mostrar dinámicamente los datos de la clasificación de equipos en la aplicación a medida que se actualizan en la base de datos.

En general, es una aplicación básica que utiliza **Firestore** para autenticar usuarios, almacenar datos y actualizar dinámicamente la interfaz de usuario de la aplicación.

Inicio de sesión con GitHub:

Este bloque se encarga de configurar el inicio de sesión mediante GitHub y de manejar el flujo de autenticación.

Primero, se selecciona el botón que se utilizará para iniciar sesión con GitHub utilizando el método `querySelector()`. Luego, se le añade un evento de click que se escucha cuando pulsamos en el botón.

En el evento, se crea una instancia de autenticación de GitHub utilizando `GithubAuthProvider()`. Luego, se llama al método `signInWithPopup()` y se le pasa la instancia de la autenticación. Este método inicia un flujo de autenticación con GitHub que abre una ventana emergente donde el usuario debe proporcionar sus credenciales de GitHub, tanto usuario como contraseña.

Si el usuario se autentica con éxito, el método `signInWithPopup()` devuelve las credenciales del usuario como un objeto `UserCredential`. Este objeto se utiliza para obtener información sobre el usuario y para establecer su sesión.

En el evento, si la autenticación tiene éxito, se cierra el modal de inicio de sesión y se actualiza la interfaz de usuario con los datos del usuario autenticado. También se llama a la función `getClasificacionEquipos()` para obtener y mostrar la clasificación de los equipos en la aplicación.

Si la autenticación falla, se vuelve a mostrar un mensaje de error utilizando la función `lanzarModal()`.

Registrarse con correo y contraseña:

Este bloque se encarga de registrar a un usuario mediante **correo y contraseña**. Primero, se selecciona el formulario de registro (con el id **signup-form**) y se le agrega un evento de escucha para el evento submit. Luego, cuando el formulario se envía, se evita que se recargue la página con **e.preventDefault()**.

A continuación, se obtiene el **correo** y la **contraseña** ingresados en el formulario, a través de **signUpForm["signup-email"].value** y **signUpForm["signup-password"].value**, respectivamente.

Después, se utiliza la función **createUserWithEmailAndPassword** de **Firestore Auth** para crear el usuario en **Firestore**, pasando como argumentos el **correo** y la **contraseña**. Si la creación de usuario es exitosa, se ejecuta el bloque de código dentro de la función "then".

Si se crea correctamente, se muestra una alerta de bienvenida y este bloque de código tiene la función de cerrar el modal de registro (si es que se está usando uno), resetear el formulario, actualizar la interfaz de usuario para mostrar que el usuario ha iniciado sesión exitosamente y llamar a una función para obtener y mostrar la clasificación de equipos desde una base de datos en tiempo real (**Firestore Realtime Database**).

Si se produce un error al crear el usuario, para mostrar un mensaje de error correspondiente al código de error en una alerta, se ejecuta el bloque de código dentro de la función **"catch"**. En este bloque, se comprueba el código del error y se lanza un mensaje de error correspondiente utilizando una función **"lanzarModal()"**. Además, si el registro es exitoso, se obtiene el nombre de usuario a partir de la dirección de correo electrónico y se muestra en la página de bienvenida.

Finalmente, se obtiene la referencia a la base de datos de **Firestore** y se llama a la función **getClasificacionEquipos** para obtener la clasificación de equipos y mostrarla en la página. Lo explicado anteriormente maneja la funcionalidad para que un usuario registrado pueda iniciar sesión con su correo electrónico y contraseña.

Primero, se obtiene una referencia al formulario de inicio de sesión con el correo electrónico y contraseña mediante el método **document.querySelector()**. Luego, se agrega un **EventListener** al formulario para que cuando se envíe el formulario, se ejecute la función **signInWithEmail()**. Dentro de **signInWithEmail()**, se evita que el formulario se envíe por defecto y se obtienen los valores de correo electrónico y contraseña del usuario. Luego, se llama a **signInWithEmailAndPassword()** para autenticar al usuario con **Firestore**.

Si la autenticación es exitosa, se muestra un mensaje de bienvenida y se oculta el modal de inicio de sesión. Si ocurre un error, se muestran diferentes mensajes de alerta según el tipo de error.

En resumen, este bloque utiliza una aplicación de **Firebase** para autenticar usuarios y almacenar información en una base de datos en tiempo real o más conocida como **Real-Time**.

Inicio de sesión con correo y contraseña:

Este bloque de código maneja el inicio de sesión de usuarios con **correo y contraseña** en una aplicación web.

Primero se selecciona el formulario de inicio de sesión y se agrega un evento de escucha para el evento **"submit"**. Al prevenir el comportamiento por defecto del formulario, se evita que se envíe a un servidor.

A continuación, se obtienen los valores de correo electrónico y contraseña del formulario y se utiliza el método **signInWithEmailAndPassword** de la API de autenticación de **Firebase** para autenticar al usuario.

Si la autenticación es exitosa, se ejecuta una función con el objeto **userCredential** que se devuelve. En este caso, se cierra el modal de inicio de sesión, se restablece el formulario, se oculta la información de inicio de sesión y se muestra el contenido principal de la página. También se recupera la clasificación de los equipos de la base de datos y se muestra en la página.

Si la autenticación no es exitosa, se maneja el error y se muestra un mensaje de error adecuado en un modal utilizando el método **lanzarModal()**.

En general, este bloque proporciona un manejo básico del **inicio de sesión** de usuarios con correo y contraseña en una aplicación web utilizando Firebase.

Imprimir la tabla desde la base de datos realtime:

La función **getClasificacionEquipos(clasificacionEquipos)** se encarga de obtener los datos de la tabla de clasificación de equipos desde la base de datos en tiempo real, y agregarlos a la tabla #myTable en el HTML.

La función **pintarCelda()** se encarga de dar formato visual a las celdas de la tabla, dependiendo de la posición del equipo en la clasificación. Primero se obtienen todas las filas de la tabla, se convierten en un array y se iteran por cada una de ellas.

Si la fila está en las primeras cuatro posiciones, se seleccionan las primeras dos celdas de la fila y se les agrega la clase "**blue**".

Si la fila está en la quinta posición, se seleccionan las primeras dos celdas de la fila y se les agrega la clase "**yellow**".

Si la fila está en la sexta posición, se seleccionan las primeras dos celdas de la fila y se les agrega la clase "**green**".

Si la fila está en las últimas tres posiciones, se seleccionan las primeras dos celdas de la fila y se les agrega la clase "**red**".

Finalmente, se remueve la clase "**red**" de las demás celdas.

Esto permite dar un formato visual a la tabla de clasificación de equipos, de manera que sea más fácil identificar la posición de cada equipo.

Cerrar sesión:

Este es el último bloque, en el que maneja el evento de click en el botón de logout y cierra la sesión del usuario actual. A continuación, una explicación línea por línea:

const logout = document.querySelector("#logout"); - Selecciona el botón de logout del documento HTML.

logout.addEventListener("click", async (e) => { - Agrega un event listener para el evento de clic en el botón de logout.

e.preventDefault(); - Previene el comportamiento predeterminado del evento de clic (que es seguir un enlace o enviar un formulario).

try { await signOut(auth); - Cierra la sesión actual del usuario utilizando la función **signOut** proporcionada por Firebase Authentication. Esto devuelve una promesa que se resuelve cuando la sesión se ha cerrado.

const pageLogin = document.querySelectorAll("#page-login li"); - Selecciona todos los elementos li en el elemento con ID page-login.

pageLogin[0].classList.remove("d-none"); - Elimina la clase d-none del primer elemento li, que contiene el botón de login.

pageLogin[1].classList.remove("d-none"); - Elimina la clase d-none del segundo elemento li, que contiene el botón de registro.

pageLogin[2].classList.add("d-none"); - Agrega la clase d-none al tercer elemento li, que contiene el botón de logout.

const pageWelcome = document.getElementById("page-main-info"); - Selecciona el elemento con ID page-main-info, que contiene un mensaje de bienvenida.

pageWelcome.classList.remove("d-none"); - Elimina la clase d-none del mensaje de bienvenida para mostrarlo en la página.

pageWelcome.classList.add("d-block"); - Agrega la clase d-block al mensaje de bienvenida para asegurarse de que se muestre como un bloque en la página.

const pageMain = document.getElementById("page-main"); - Selecciona el elemento con ID page-main, que contiene la tabla de clasificación de equipos.

pageMain.classList.remove("d-block"); - Elimina la clase d-block del elemento de la tabla de clasificación para ocultarlo en la página.

pageMain.classList.add("d-none"); - Agrega la clase d-none al elemento de la tabla de clasificación para asegurarse de que no se muestre en la página.

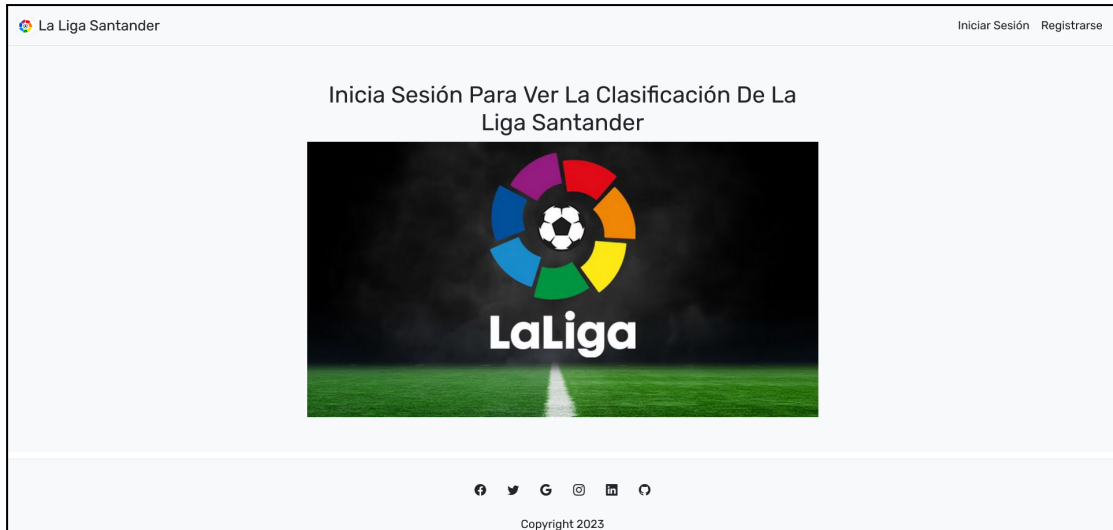
const footer = document.getElementById("footer"); - Selecciona el elemento con ID footer, que es el pie de página.

footer.classList.add("fixed-bottom"); - Agrega la clase fixed-bottom al pie de página para asegurarse de que permanezca en la parte inferior de la página.

catch (error) { console.log(error); } - Si ocurre un error al cerrar la sesión, se muestra en la consola del navegador.

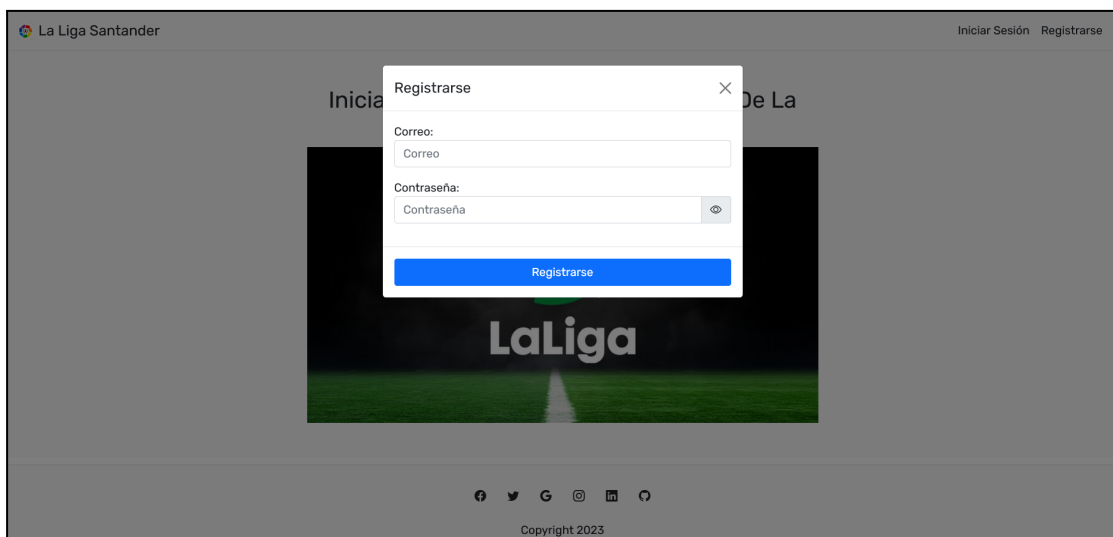
Casos de uso:

Página principal sin inicio de sesión:



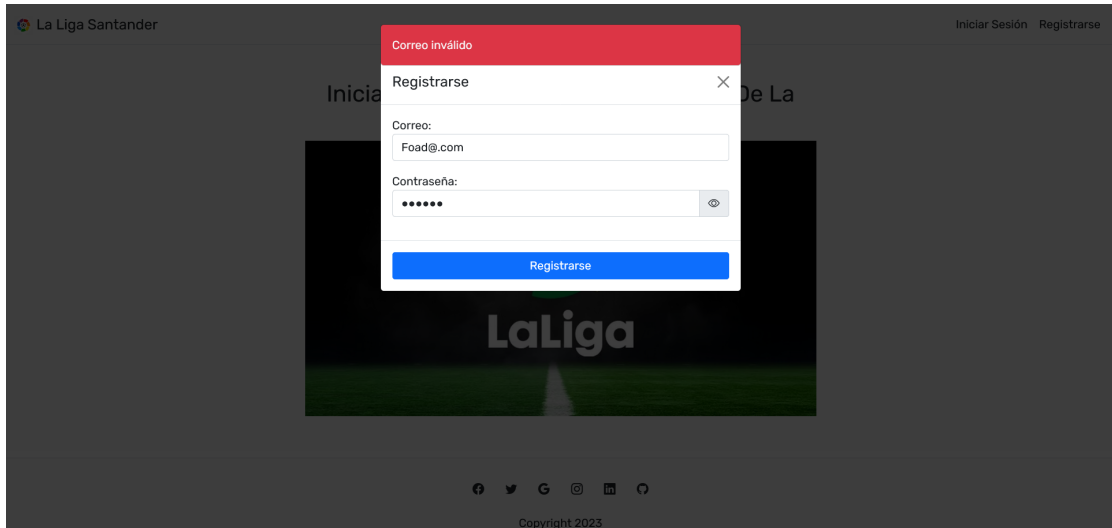
Página principal sin iniciar sesión

Registrarse:



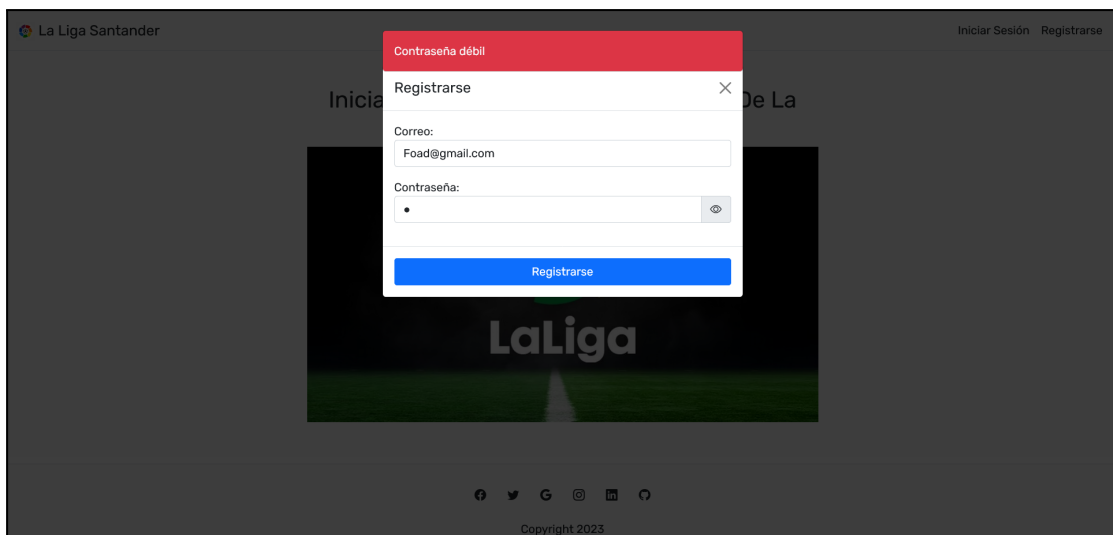
Registro

Correo inválido:



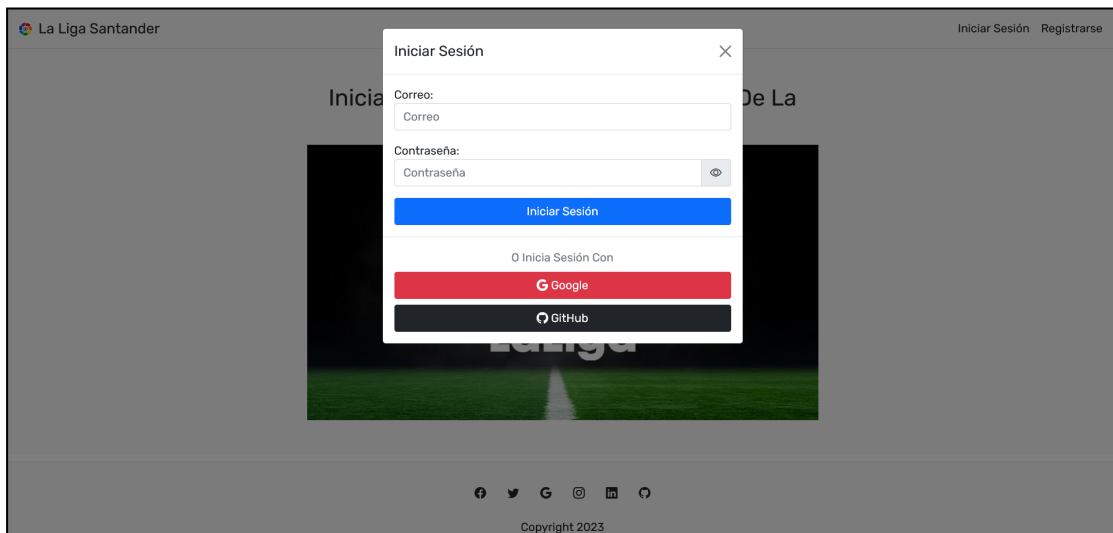
(Error) Correo inválido

Contraseña débil:



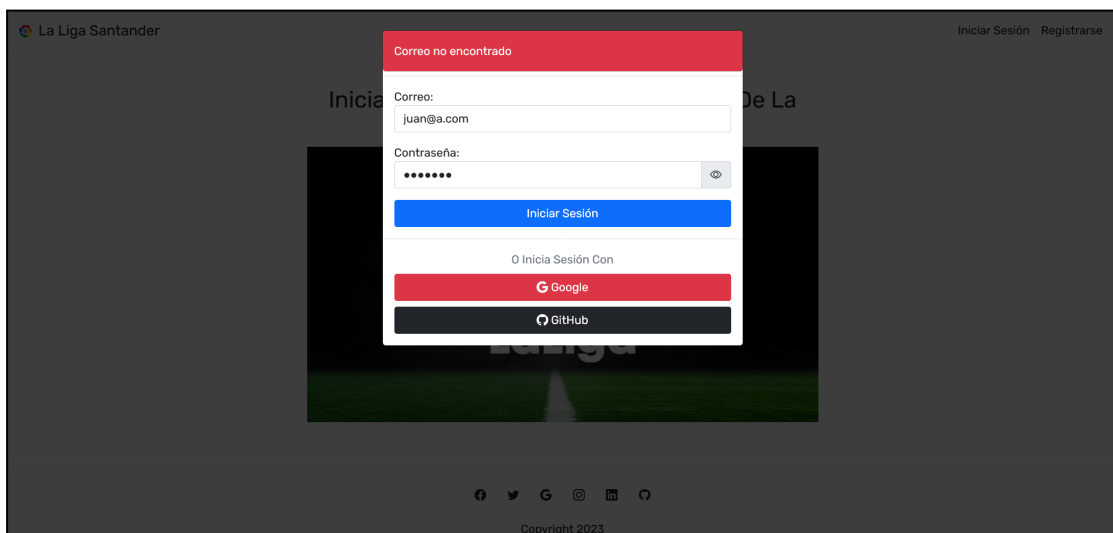
(Error) Contraseña débil

Inicio de sesión con correo y contraseña:



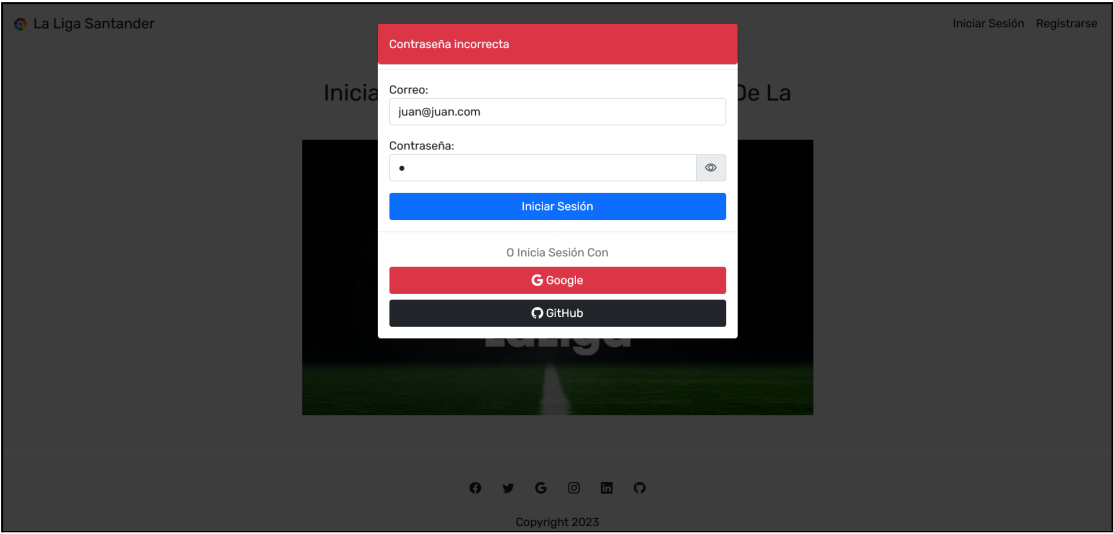
Inicio de sesión con correo y contraseña

Correo no encontrado:



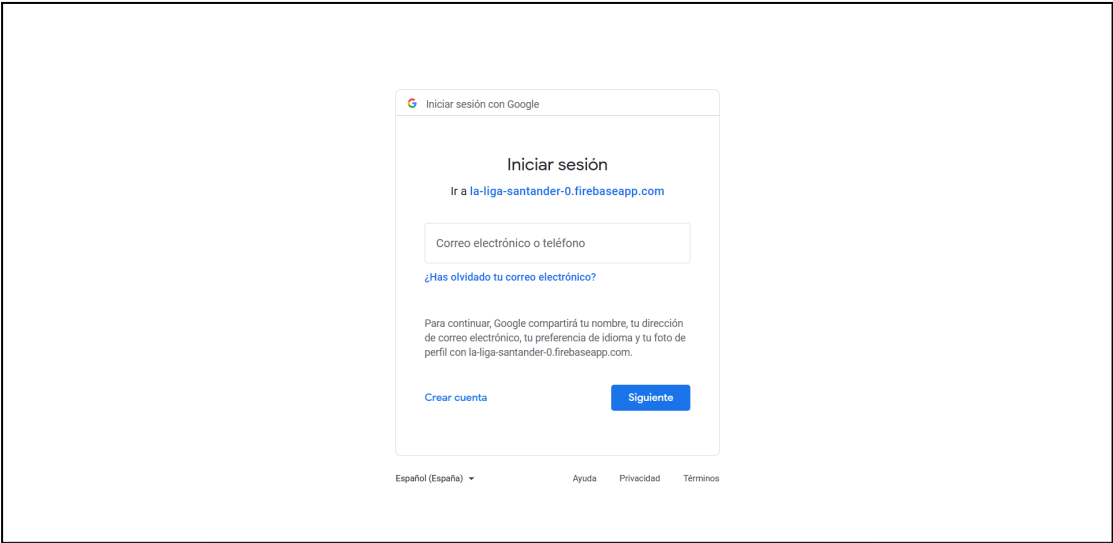
(Error) Correo no encontrado

Contraseña incorrecta:



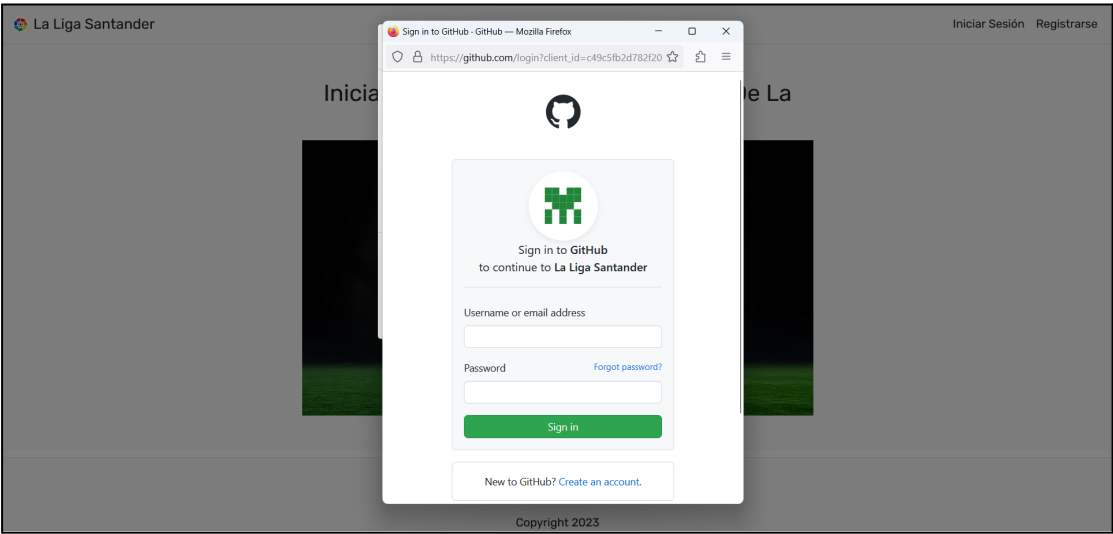
(Error) Contraseña incorrecta

Inicio de sesión con Google:



Inicio de sesión mediante Google

Inicio de sesión con GitHub:



Inicio de sesión mediante GitHub

Resultado final:

La Liga Santander							Cerrar sesión
Bienvenido Nour._07							
Clasificación De La Liga Santander							
#	Nombre del equipo	Puntos	Partidos jugados	Partidos ganados	Partidos empatados	Partidos perdidos	
1	Barcelona	58	21	18	2	1	
2	Real Madrid	48	21	15	3	3	
3	Real Sociedad	42	21	13	3	5	
4	Atletico de Madrid	38	21	11	5	5	
5	Real Betis	34	21	10	4	7	
6	Rayo Vallecano	33	21	9	6	6	
7	Athletic Bilbao	32	21	9	5	7	
8	Villareal	31	21	9	4	8	
9	Osasuna	30	21	8	6	7	
10	Mallorca	28	21	8	4	9	
11	Girona	24	21	6	6	9	
12	Sevilla	24	21	6	6	9	
13	Real Valladolid	24	21	7	3	11	
14	Celta de Vigo	23	21	6	5	10	
15	Almeria	22	21	6	4	11	
16	Cadiz	22	21	5	7	9	
17	Espanyol	21	21	4	9	8	
18	Valencia	20	21	5	5	11	
19	Getafe	19	21	4	7	10	
20	Eliche	9	21	1	6	14	
Champions League Europa League UEFA Conference League Descenso							
Facebook Twitter Google+ Instagram LinkedIn YouTube							
Copyright 2023							

Resultado final