ANEXOS

ANEXO I

Tabla 1. Historia de Usuario GU001-1

	Historia de Usuario
Identificador (ID): GU001-1	**Usuario:** Nuevo Usuario, admin
Nombre de historia: Registrar usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 5	**Sprint:** 1
Descripción: Como nuevo usuario quiero registrarme como usuario del sistema web y aplicación móvil para acceder a la información sobre las ligas, torneos, equipos y jugadores.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un nombre válido y Cuando el cliente envié un correo electrónico válido y Cuando el cliente envié una contraseña válida y Cuando el cliente envié un tipo válido Entonces el sistema retornará un json con el campo success en true Y el sistema insertará al usuario en la BD. • **Criterio de Aceptación 2: Envió de datos vacíos** Cuando el cliente envié un nombre vacío o Cuando el cliente envié un correo electrónico vacío o Cuando el cliente envié una contraseña vacía o Cuando el cliente envié un tipo vacío Entonces el sistema retornará un json con el campo noHayDatos en estado true Y el sistema no insertará al usuario en la BD. • **Criterio de Aceptación 3: Envió de correo electrónico registrado** Cuando el cliente envié un correo electrónico registrado	

Entonces el sistema retornará un json con el campo success en estado false Y el sistema no registrará al usuario en la BD. • **Criterio de Aceptación 4: Error en inserción** Cuando exista un error en la inserción en la BD Entonces el sistema retornará un json con el campo noInserto en estado true Y el sistema no insertará al usuario en la BD.

Tabla 2. Historia de Usuario GU001-2

Historia de Usuario	
Identificador (ID): GU001-2	**Usuario:** Usuario Registrado
Nombre de historia: Iniciar de sesión	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 3	**Sprint:** 1
Descripción: Como usuario registrado quiero iniciar sesión en el sistema web o aplicación móvil para acceder a la información sobre las ligas, torneos, equipos y jugadores.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un correo electrónico válido y Cuando el cliente envié una contraseña válida Entonces el sistema retornará un json con un token, el correo del usuario y el tipo de usuario • **Criterio de Aceptación 2: Envió de datos vacíos** Cuando el cliente envié un correo electrónico vacío o Cuando el cliente envié una contraseña vacía Entonces el sistema retornará un json con el campo noHayDatos en estado true • **Criterio de Aceptación 3: Envió de correo electrónico no registrado** Cuando el cliente envié un correo electrónico no registrado Entonces el sistema retornará un json con el campo correo en estado false • **Criterio de Aceptación 4: Contraseña incorrecta** Cuando el cliente envié un correo electrónico registrado y Cuando el cliente envié una contraseña incorrecta Entonces el sistema retornará un json con el campo success en estado false	

Tabla 3. Historia de Usuario GL001-1

Historia de Usuario	
Identificador (ID): GL001-1	**Usuario:** Presidente
Nombre de historia: Registrar liga	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 5	**Sprint:** 1
Descripción: Como usuario presidente quiero registrar liga para tener las estadísticas de cada liga y hacer un seguimiento de esta	

Criterios de aceptación:

- **Criterio de Aceptación 1: Envió de datos correctos**

Cuando el cliente envié un nombre válido y
Cuando el cliente envié una fecha de fundación válida y
Cuando el cliente envié una dirección válida y
Cuando el cliente envié un correo electrónico válido
Entonces el sistema retornará un json con el id de la liga insertada y un success en true
Y el sistema insertará la liga en la BD.

- **Criterio de Aceptación 2: Envió de datos vacíos**

Cuando el cliente envié un nombre vacío o
Cuando el cliente envié una fecha de fundación vacío o
Cuando el cliente envié una dirección vacío o
Cuando el cliente envié un correo electrónico vacío
Entonces el sistema retornará un json con el campo noHayDatos en estado true
Y el sistema no insertará la liga en la BD.

- **Criterio de Aceptación 3: Envió de correo electrónico no registrado**

Cuando el cliente envié un correo electrónico no registrado
Entonces el sistema retornará un json con el campo no_existe_usuario en estado true
Y el sistema no insertará la liga en la BD.

- **Criterio de Aceptación 4: Envió de correo electrónico que no es presidente**

Cuando el cliente envié un correo electrónico que no pertenece a un presidente
Entonces el sistema retornará un json con el campo no_es_presidente en estado true
Y el sistema no insertará la liga en la BD.

- **Criterio de Aceptación 5: Envió de correo electrónico que tiene liga registrada**

Cuando el cliente envié un correo electrónico que tiene una liga registrada
Entonces el sistema retornará un json con el campo existe_registro en estado true
Y el sistema no insertará la liga en la BD.

- **Criterio de Aceptación 6: Envió de nombre de liga existente**

Cuando el cliente envié un correo electrónico válido y
Cuando el cliente envié un nombre de liga existente
Entonces el sistema retornará un json con el campo existe_nombre en estado true
Y el sistema no insertará la liga en la BD.

- **Criterio de Aceptación 7: Error en inserción**

Cuando exista un error en la inserción en la BD
Entonces el sistema retornará un json con el campo noInserto en estado true
Y el sistema no insertará la liga en la BD.

Tabla 4. Historia de Usuario GL001-2

Historia de Usuario	
Identificador (ID): GL001-2	**Usuario:** Presidente

Nombre de historia: Registrar categoría	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 5	**Sprint:** 1
Descripción: Como usuario presidente quiero registrar categoría organizar adecuadamente las distintas categorías y tener una gestión eficiente de los torneos.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un nombre válido y Cuando el cliente envié un número de equipos válido y Cuando el cliente envié un id de liga válido Entonces el sistema retornará un json con el campo success en true Y el sistema insertará la categoría en la BD. • **Criterio de Aceptación 2: Envió de un id de liga incorrecto** Cuando el cliente envié un id de liga incorrecto Entonces el sistema retornará un json con el campo no_existe_liga en true Y el sistema no insertará la categoría en la BD. • **Criterio de Aceptación 3: Envió de datos vacíos** Cuando el cliente envié un id de liga válido y Cuando el cliente envié un nombre vacío o Cuando el cliente envié un número de equipos vacío Entonces el sistema retornará un json con el campo noHayDatos en estado true Y el sistema no insertará la categoría en la BD. • **Criterio de Aceptación 4: Envió de nombre de categoría existente en la liga** Cuando el cliente envié un id de liga válido y Cuando el cliente envié un número de equipos válido y Cuando el cliente envié un nombre de categoría existente en la liga Entonces el sistema retornará un json con el campo existe_nombre en estado true Y el sistema no insertará la categoría en la BD. • **Criterio de Aceptación 5: Error en inserción** Cuando exista un error en la inserción en la BD Entonces el sistema retornará un json con el campo noInserto en estado true Y el sistema no insertará la categoría en la BD.	

Tabla 5. Historia de Usuario GL001-3

	Historia de Usuario
Identificador (ID): GL001-3	**Usuario:** Presidente
Nombre de historia: Mostrar liga	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Sprint:** 1
Descripción: Como usuario presidente quiero ver la liga para ver información detallada sobre una liga específica dentro del sistema	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un correo electrónico válido	

Entonces el sistema retornará un json con el campo datos en el cual estará toda la información de la liga • **Criterio de Aceptación 2: Envió de correo electrónico no registrado** Cuando el cliente envié un correo electrónico no registrado Entonces el sistema retornará un json con el campo no_existe_usuario en estado true • **Criterio de Aceptación 3: Envió de correo electrónico que no es presidente** Cuando el cliente envié un correo electrónico que no pertenece a un presidente Entonces el sistema retornará un json con el campo no_es_presidente en estado true • **Criterio de Aceptación 4: Error en la consulta** Cuando exista un error en la consulta en la BD Entonces el sistema retornará un json con el campo datos en vacío

Tabla 6. Historia de Usuario GL001-4

Historia de Usuario	
Identificador (ID): GL001-4	**Usuario:** Presidente
Nombre de historia: Mostrar categorías	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Sprint:** 1
Descripción: Como usuario presidente quiero ver categorías para ver los equipos disponibles en cada categoría en el sistema	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de liga válido Entonces el sistema retornará un json con el campo datos en el cual estará toda la información de la o las categorías • **Criterio de Aceptación 2: Envió de un id de liga incorrecto** Cuando el cliente envié un id de liga incorrecto Entonces el sistema retornará un json con el campo no_existe_liga en true • **Criterio de Aceptación 3: Error en la consulta** Cuando exista un error en la consulta en la BD Entonces el sistema retornará un json con el campo datos en vacío	

Tabla 7. Historia de Usuario GE001-2

Historia de Usuario	
Identificador (ID): GE001-2	**Usuario:** Público general
Nombre de historia: Validar número de equipos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 3	**Sprint:** 1
Descripción: Como admin quiero validar el número de equipos en una categoría para saber el número de equipos inscritos en una categoría específica dentro del sistema, asegurando que el número de equipos cumpla con los requisitos establecidos para esa categoría	

Criterios de aceptación:

- **Criterio de Aceptación 1: Envió de datos correctos y limite alcanzado**

Cuando el cliente envié un id de categoría válido y
Se alcanzo el límite de equipos en esa categoría
Entonces el sistema retornará un json con el campo limite_equipos en true

- **Criterio de Aceptación 2: Envió de datos correctos y limite no alcanzado**

Cuando el cliente envié un id de categoría válido y
No se alcanzó el límite de equipos en esa categoría
Entonces el sistema retornará un json con el campo limite_equipos en false

- **Criterio de Aceptación 3: Envió de un id de categoría incorrecto**

Cuando el cliente envié un id de categoría incorrecto
Entonces el sistema retornará un json con el campo no_existe_categoria en true

- **Criterio de Aceptación 4: Error en la consulta**

Cuando exista un error en la consulta en la BD
Entonces el sistema retornará un json con el campo limite_equipos en false

Tabla 8. Historia de Usuario GE001-3

Historia de Usuario	
Identificador (ID): GE001-3	**Usuario:** Presidente
Nombre de historia: Registrar equipo	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 8	**Sprint:** 1
Descripción: Como usuario presidente quiero registrar equipo para facilitar la participación de equipos y un seguimiento adecuado de los equipos participantes.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un nombre válido y Cuando el cliente envié una fecha de fundación válida y Cuando el cliente envié un presidente válido y Cuando el cliente envié un color válido y Cuando el cliente envié un escudo válido y Cuando el cliente envié un id de categoría válido y Cuando el cliente envié un id de liga válido Entonces el sistema retornará un json con el campo success en true Y el sistema insertará el equipo en la BD. • **Criterio de Aceptación 2: Envió de un id de liga incorrecto** Cuando el cliente envié un id de liga incorrecto Entonces el sistema retornará un json con el campo no_existe_liga en true Y el sistema no insertará el equipo en la BD. • **Criterio de Aceptación 3: Envió de un id de categoría incorrecto** Cuando el cliente envié un id de liga válido y	

Cuando el cliente envié un id de categoría incorrecto
Entonces el sistema retornará un json con el campo no_existe_categoria en true
Y el sistema no insertará el equipo en la BD.

- **Criterio de Aceptación 4: Envió de datos vacíos**

Cuando el cliente envié un id de liga válido y
Cuando el cliente envié un id de categoría válido y
Cuando el cliente envié un nombre vacío o
Cuando el cliente envié una fecha de fundación vacía o
Cuando el cliente envié un presidente vacío o
Cuando el cliente envié un color vacío o
Cuando el cliente envié un escudo vacío o
Entonces el sistema retornará un json con el campo noHayDatos en estado true
Y el sistema no insertará el equipo en la BD.

- **Criterio de Aceptación 5: Envió de nombre de equipo existente en la liga**

Cuando el cliente envié un id de liga válido y
Cuando el cliente envié un id de categoría válido
Cuando el cliente envié una fecha de fundación válida y
Cuando el cliente envié un presidente válido y
Cuando el cliente envié un color válido y
Cuando el cliente envié un escudo válido
Cuando el cliente envié un nombre de equipo existen en la liga
Entonces el sistema retornará un json con el campo existe_nombre en estado true
Y el sistema no insertará el equipo en la BD.

- **Criterio de Aceptación 6: Error en inserción**

Cuando exista un error en la inserción en la BD
Entonces el sistema retornará un json con el campo noInserto en estado true
Y el sistema no insertará el equipo en la BD.

Tabla 9. Historia de Usuario GE001-1

Historia de Usuario	
Identificador (ID): GE001-1	**Usuario:** Presidente
Nombre de historia: Mostrar equipos de liga	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Sprint:** 1
Descripción: Como usuario presidente quiero ver los equipos de una liga para ver los equipos participantes en una liga específica dentro del sistema	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de liga válido Entonces el sistema retornará un json con el campo datos en el cual estará toda la información de los equipos de esa liga • **Criterio de Aceptación 2: Envió de un id de liga incorrecto** Cuando el cliente envié un id de liga incorrecto Entonces el sistema retornará un json con el campo no_existe_liga en true	

• **Criterio de Aceptación 3: Error en la consulta** Cuando exista un error en la consulta en la BD Entonces el sistema retornará un json con el campo datos en vacío

Tabla 10. Historia de Usuario PI001-5

Historia de Usuario	
Identificador (ID): PI001-5	**Usuario:** Público general
Nombre de historia: Mostrar equipo	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Sprint:** 1
Descripción: Como público general quiero ver la información de un equipo para ver la información detallada de un equipo específico en el sistema.	
Criterios de aceptación: **Criterios de aceptación:** • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de equipo válido Entonces el sistema retornará un json con el campo datos en el cual estará toda la información del equipo • **Criterio de Aceptación 2: Envió de un id de liga incorrecto** Cuando el cliente envié un id de equipo incorrecto Entonces el sistema retornará un json con el campo no_existe_equipo en true • **Criterio de Aceptación 3: Error en la consulta** Cuando exista un error en la consulta en la BD Entonces el sistema retornará un json con el campo datos en vacío	

Tabla 11. Historia de Usuario GE001-4

Historia de Usuario	
Identificador (ID): GE001-4	**Usuario:** Presidente
Nombre de historia: Registrar jugador	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 8	**Sprint:** 2
Descripción: Como usuario presidente quiero registrar jugador para para registrar los jugadores y tener las estadísticas de cada uno.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié una cédula válida y Cuando el cliente envié un nombre válido y Cuando el cliente envié una posición válida y Cuando el cliente envié una fecha de fundación válida y Cuando el cliente envié una foto válida y Cuando el cliente envié una estatura válida y Cuando el cliente envié un número de camiseta válida y Cuando el cliente envié un id de equipo válido Entonces el sistema retornará un json con el campo success en true Y el sistema insertará el jugador en la BD. • **Criterio de Aceptación 2: Envió de un id de equipo incorrecto** Cuando el cliente envié un id de equipo incorrecto	

Entonces el sistema retornará un json con el campo no_existe_equipo en true
Y el sistema no insertará el jugador en la BD.

- **Criterio de Aceptación 3: Envió de cédula existente**

Cuando el cliente envié un id de equipo válido y
Cuando el cliente envié una cédula existente
Entonces el sistema retornará un json con el campo existe_ jugador en true
Y el sistema no insertará el jugador en la BD.

- **Criterio de Aceptación 4: Envió de datos vacíos**

Cuando el cliente envié un id de equipo válido y
Cuando el cliente envié una cédula válida y
Cuando el cliente envié un nombre vacío o
Cuando el cliente envié una posición vacía o
Cuando el cliente envié una fecha de fundación vacía o
Cuando el cliente envié una foto vacía o
Cuando el cliente envié una estatura vacía o
Cuando el cliente envié un número de camiseta vacío
Entonces el sistema retornará un json con el campo noHayDatos en estado true
Y el sistema no insertará el jugador en la BD.

- **Criterio de Aceptación 5: Envió de número de camiseta existen en equipo**

Cuando el cliente envié un id de equipo válido y
Cuando el cliente envié una cédula válida y
Cuando el cliente envié un nombre válido y
Cuando el cliente envié una posición válida y
Cuando el cliente envié una fecha de fundación válida y
Cuando el cliente envié una foto válida y
Cuando el cliente envié una estatura válida y
Cuando el cliente envié un número de camiseta existente en equipo
Entonces el sistema retornará un json con el campo exite_numero_camiseta en true
Y el sistema no insertará el jugador en la BD.

- **Criterio de Aceptación 6: Error en inserción**

Cuando exista un error en la inserción en la BD
Entonces el sistema retornará un json con el campo noInserto en estado true
Y el sistema no insertará la categoría en la BD.

Tabla 12. Historia de Usuario GT001-1

Historia de Usuario	
Identificador (ID): GT001-1	**Usuario:** Presidente
Nombre de historia: Registrar torneos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 8	**Sprint:** 2
Descripción: Como usuario presidente quiero registrar torneos para establecer un registro completo y estructurado de los torneos lo que garantiza una competición organizada.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos sin grupos**	

Cuando el cliente envié un id de categoría válido y
Cuando el cliente envié una etapa válida y
Cuando el cliente envié una fecha de inicio válida y
Cuando el cliente envié una fecha de fin válida y
Cuando el cliente envié canchas válidas y
Cuando el cliente envié grupos vacío y
Cuando el cliente envié el número de clasificados válido
Entonces el sistema retornará un json con el campo success en true
Y el sistema insertará el torneo en la BD.

- **Criterio de Aceptación 2: Envió de datos correctos con grupos**

Cuando el cliente envié un id de categoría válido y
Cuando el cliente envié una etapa válida y
Cuando el cliente envié una fecha de inicio válida y
Cuando el cliente envié una fecha de fin válida y
Cuando el cliente envié canchas válidas y
Cuando el cliente envié grupos válido y
Cuando el cliente envié el número de clasificados válido
Entonces el sistema retornará un json con el campo success en true y los id de los torneos insertados.
Y el sistema insertará los torneos en la BD.

- **Criterio de Aceptación 3: Envió de un id de categoría incorrecto**

Cuando el cliente envié un id de categoría incorrecto
Entonces el sistema retornará un json con el campo no_existe_ categoría en true
Y el sistema no insertará el o los torneos en la BD.

- **Criterio de Aceptación 4: Envió de datos vacíos**

Cuando el cliente envié un id de categoría válido y
Cuando el cliente envié una etapa vacía o
Cuando el cliente envié una fecha de inicio vacía o
Cuando el cliente envié una fecha de fin vacía o
Cuando el cliente envié canchas vacías o
Cuando el cliente envié el número de clasificados vacío
Entonces el sistema retornará un json con el campo noHayDatos en estado true
Y el sistema no insertará el o los torneos en la BD.

- **Criterio de Aceptación 5: Error en inserción**

Cuando exista un error en la inserción en la BD
Entonces el sistema retornará un json con el campo noInserto en estado true
Y el sistema no insertará el o los torneos en la BD.

Tabla 13. Historia de Usuario GT001-2

Historia de Usuario	
Identificador (ID): GT001-2	**Usuario:** Presidente
Nombre de historia: Generar partidos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 8	**Sprint:** 2
Descripción: Como usuario presidente quiero generar partidos para obtener partidos completos y equitativos para los torneos asegurando una competición justa para todos los involucrados.	

Criterios de aceptación:

- **Criterio de Aceptación 1: Envió de datos correctos**

Cuando el cliente envié un id de torneo válido y
Cuando el cliente envié equipos válidos
Entonces el sistema retornará un json con el campo success en estado true
Y el sistema registrará los equipos en la tabla estadísticas de equipo en la BD
Y el sistema generará el calendario de juego
Y el sistema registrará cada partido en la BD

- **Criterio de Aceptación 2: Envió de un id de torneo incorrecto**

Cuando el cliente envié un id de torneo incorrecto
Entonces el sistema retornará un json con el campo no_existe_torneo en true.
Y el sistema no registrará los equipos en la tabla estadísticas de equipo en la BD
Y el sistema no generará el calendario de juego
Y el sistema no registrará cada partido en la BD

- **Criterio de Aceptación 3: Envió de un id de torneo que tiene partidos registrados**

Cuando el cliente envié un id de torneo con partidos registrados
Entonces el sistema retornará un json con el campo existen_partidos en true.
Y el sistema no registrará los equipos en la tabla estadísticas de equipo en la BD
Y el sistema no generará el calendario de juego
Y el sistema no registrará cada partido en la BD

- **Criterio de Aceptación 4: Envió de datos vacíos**

Cuando el cliente envié un id de torneo válido y
Cuando el cliente envié equipos vacíos
Entonces el sistema retornará un json con el campo noHayDatos en estado true
Y el sistema no registrará los equipos en la tabla estadísticas de equipo en la BD
Y el sistema no generará el calendario de juego
Y el sistema no registrará cada partido en la BD

- **Criterio de Aceptación 5: Envió de id de equipo incorrecto**

Cuando el cliente envié un id de equipo incorrecto
Entonces el sistema retornará un json con el campo no_existen_equipo en true.
Y el sistema no registrará los equipos en la tabla estadísticas de equipo en la BD
Y el sistema no generará el calendario de juego
Y el sistema no registrará cada partido en la BD

- **Criterio de Aceptación 6: Error en la consulta**

Cuando exista un error en la consulta en la BD
Entonces el sistema retornará un json con el campo no_existen_equipo en true.
Y el sistema no registrará los equipos en la tabla estadísticas de equipo en la BD
Y el sistema no generará el calendario de juego
Y el sistema no registrará cada partido en la BD

- **Criterio de Aceptación 7: Error en inserción**

Cuando exista un error en la inserción en la BD

Entonces el sistema retornará un json con el campo noInserto en estado true Y el sistema no registrará partido en la BD.

Tabla 14. Historia de Usuario GT001-3

Historia de Usuario	
Identificador (ID): GT001-3	**Usuario:** Presidente
Nombre de historia: Mostrar torneos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Sprint:** 2
Descripción: Como usuario presidente quiero ver los torneos para ver información sobre los torneos disponibles dentro del sistema, proporcionando detalles como fechas de inicio y fin, equipos participantes, formato del torneo y cualquier otra información relevante.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de categoría válido Entonces el sistema retornará un json con el campo datos en el cual estará toda la información de la o los torneos • **Criterio de Aceptación 2: Envió de un id de categoría incorrecto** Cuando el cliente envié un id de categoría incorrecto Entonces el sistema retornará un json con el campo no_existe_categoria en true • **Criterio de Aceptación 3: Error en la consulta** Cuando exista un error en la consulta en la BD Entonces el sistema retornará un json con el campo datos en vacío	

Tabla 15. Historia de Usuario GP001-1

Historia de Usuario	
Identificador (ID): GP001-1	**Usuario:** Presidente
Nombre de historia: Programar partidos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 5	**Sprint:** 2
Descripción: Como usuario presidente quiero programar los partidos para asignar fechas, horarios y canchas para los encuentros entre los equipos participantes en el sistema.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de partido válido y Cuando el cliente envié partido válido Entonces el sistema retornará un json con el campo success en true Y el sistema actualizará el partido en la BD. • **Criterio de Aceptación 2: Envió de un id de partido incorrecto** Cuando el cliente envié un id de torneo válido y Cuando el cliente envié un id de partido incorrecto Entonces el sistema retornará un json con el campo no_existe_partido en true Y el sistema no actualizará el partido en la BD.	

• **Criterio de Aceptación 3: Envió de un id de partido en un estado que no es programar o pospuesto** Cuando el cliente envié un id de partido en un estado que no es programar o pospuesto Entonces el sistema retornará un json con el campo partido_ y el estado del partido en true Y el sistema no actualizará el partido en la BD. • **Criterio de Aceptación 4: Envió de datos vacíos** Cuando el cliente envié un id de partido válido y Cuando el cliente envié partido vacío Entonces el sistema retornará un json con el campo noHayDatos en estado true Y el sistema no actualizará el partido en la BD. • **Criterio de Aceptación 5: Error en la actualización de estadísticas partido** Cuando exista un error en la actualización de partido en la BD Entonces el sistema retornará un json con el campo noInserto en estado true Y el sistema no actualizará el partido en la BD.

Tabla 16. Historia de Usuario PI001-1

Historia de Usuario	
Identificador (ID): PI001-1	**Usuario:** Público general
Nombre de historia: Mostrar todas las ligas inscritas	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Sprint:** 2
Descripción: Como público general quiero ver las ligas para ver una lista de todas las ligas disponibles en el sistema, lo que facilita la participación y el seguimiento.	
Criterios de aceptación: • **Criterio de Aceptación 1: Consulta correcta** Cuando la consulta en la BD se correcta Entonces el sistema retornará un json con el campo datos en el cual estará toda la información de las ligas. • **Criterio de Aceptación 2: Error en la consulta** Cuando exista un error en la consulta en la BD Entonces el sistema retornará un json con el campo datos en vacío	

Tabla 17. Historia de Usuario PI001-2

Historia de Usuario	
Identificador (ID): PI001-2	**Usuario:** Público general
Nombre de historia: Mostrar equipos de categoría	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Sprint:** 2
Descripción: Como público general quiero ver los equipos de una categoría para ver una lista de equipos que pertenecen a una categoría específica en el sistema, lo que ayuda a conocer mejor la composición y la competencia en cada división.	

Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de categoría válido Entonces el sistema retornará un json con el campo datos en el cual estará toda la información de los equipos de esa categoría. • **Criterio de Aceptación 2: Envió de un id de categoría incorrecto** Cuando el cliente envié un id de categoría incorrecto Entonces el sistema retornará un json con el campo no_existe_categoria en true • **Criterio de Aceptación 3: Error en la consulta** Cuando exista un error en la consulta en la BD Entonces el sistema retornará un json con el campo datos en vacío

Tabla 18. Historia de Usuario GE001-5

Historia de Usuario	
Identificador (ID): GE001-5	**Usuario:** Presidente
Nombre de historia: Mostrar jugadores	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Sprint:** 2
Descripción: Como usuario presidente general quiero ver los jugadores para ver los jugadores de un equipo específico dentro del sistema, proporcionando una lista completa y actualizada de los jugadores que forman parte de ese equipo.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de equipo válido Entonces el sistema retornará un json con el campo datos en el cual estará toda la información de los jugadores de ese equipo. • **Criterio de Aceptación 2: Envió de un id de equipo incorrecto** Cuando el cliente envié un id de equipo incorrecto Entonces el sistema retornará un json con el campo no_existe_equipo en true. • **Criterio de Aceptación 3: Error en la consulta** Cuando exista un error en la consulta en la BD Entonces el sistema retornará un json con el campo datos en vacío	

Tabla 19. Historia de Usuario PI001-3

Historia de Usuario	
Identificador (ID): PI001-3	**Usuario:** Público general
Nombre de historia: Mostrar posiciones	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 5	**Sprint:** 2
Descripción: Como público general quiero ver posiciones para seguir la clasificación de los equipos en el torneo.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de torneo válido	

Entonces el sistema retornará un json con el campo tabla en el cual estará las posiciones de ese torneo junto con la información del equipo.

- **Criterio de Aceptación 2: Envió de un id de torneo incorrecto**

Cuando el cliente envié un id de torneo incorrecto
Entonces el sistema retornará un json con el campo no_existe_torneo en true.

- **Criterio de Aceptación 3: Error en la consulta**

Cuando exista un error en la consulta en la BD
Entonces el sistema retornará un json con el campo tabla en vacío

Tabla 20. Historia de Usuario GP001-2

Historia de Usuario	
Identificador (ID): GP001-2	**Usuario:** Vocal
Nombre de historia: Registrar resultados	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 5	**Sprint:** 3
Descripción: Como usuario vocal quiero registrar los resultados de los partidos para mantener un registro preciso y oficial de los resultados del torneo.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de torneo válido y Cuando el cliente envié un id de partido válido y Cuando el cliente envié partido válido Entonces el sistema retornará un json con el campo success en true Y el sistema modificará el partido en la BD Y el sistema actualizará las estadísticas del equipo en la BD. • **Criterio de Aceptación 2: Envió de un id de torneo incorrecto** Cuando el cliente envié un id de torneo incorrecto Entonces el sistema retornará un json con el campo no_existe_torneo en true Y el sistema no modificará el partido en la BD Y el sistema no actualizará las estadísticas del equipo en la BD. • **Criterio de Aceptación 3: Envió de un id de torneo sin partidos** Cuando el cliente envié un id de torneo sin partidos Entonces el sistema retornará un json con el campo no_existen_partidos en true Y el sistema no modificará el partido en la BD Y el sistema no actualizará las estadísticas del equipo en la BD. • **Criterio de Aceptación 4: Envió de un id de partido incorrecto** Cuando el cliente envié un id de torneo válido y Cuando el cliente envié un id de partido incorrecto Entonces el sistema retornará un json con el campo no_existe_partido en true Y el sistema no modificará el partido en la BD Y el sistema no actualizará las estadísticas del equipo en la BD. • **Criterio de Aceptación 5: Envió de un id de partido en un estado que no es por jugar** Cuando el cliente envié un id de torneo válido y Cuando el cliente envié un id de partido en un estado que no es por jugar	

Entonces el sistema retornará un json con el campo partido_ y el estado del partido en true
Y el sistema no modificará el partido en la BD
Y el sistema no actualizará las estadísticas del equipo en la BD.

- **Criterio de Aceptación 6: Envió de datos vacíos**

Cuando el cliente envié un id de torneo válido y
Cuando el cliente envié un id de partido válido y
Cuando el cliente envié partido vacío
Entonces el sistema retornará un json con el campo noHayDatos en estado true
Y el sistema no modificará el partido en la BD
Y el sistema no actualizará las estadísticas del equipo en la BD.

- **Criterio de Aceptación 7: Error en la actualización de partido**

Cuando exista un error en la actualización de partido en la BD
Entonces el sistema retornará un json con el campo no_modifico_partido en estado true
Y el sistema no modificará el partido en la BD
Y el sistema no actualizará las estadísticas del equipo en la BD.

- **Criterio de Aceptación 8: Error en la actualización de estadística equipo**

Cuando exista un error en la actualización de estadística equipo en la BD
Entonces el sistema retornará un json con el campo no_inserto_estadistica en estado true
Y el sistema no actualizará las estadísticas del equipo en la BD.

Tabla 21. Historia de Usuario GP001-3

Historia de Usuario	
Identificador (ID): GP001-3	**Usuario:** Admin
Nombre de historia: Comprobar registro de estadísticas de jugadores	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 3	**Sprint:** 3
Descripción: Como admin del sistema, quiero implementar una verificación de registros previos de estadísticas de jugadores, para evitar la inserción de estadísticas de jugadores ya registradas.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de torneo válido y Cuando el cliente envié un id de partido válido Entonces el sistema retornará un json con el campo existen_registros en estado true. • **Criterio de Aceptación 2: Envió de un id de torneo incorrecto** Cuando el cliente envié un id de torneo incorrecto Entonces el sistema retornará un json con el campo no_existe_torneo en true. • **Criterio de Aceptación 3: Envió de un id de partido incorrecto** Cuando el cliente envié un id de torneo válido y Cuando el cliente envié un id de partido incorrecto Entonces el sistema retornará un json con el campo no_existe_partido en true. • **Criterio de Aceptación 4: Envió de un id de partido en un estado que no es jugado**	

Cuando el cliente envié un id de torneo válido y
Cuando el cliente envié un id de partido que no en esta en estado jugado
Entonces el sistema retornará un json con el campo partido_ y el estado del partido en true.

- **Criterio de Aceptación 5: Error en la consulta**

Cuando exista un error en la consulta en la BD
Entonces el sistema retornará un json con el campo no_existen_registros en estado true.

Tabla 22. Historia de Usuario GP001-4

Historia de Usuario	
Identificador (ID): GP001-4	**Usuario:** Vocal
Nombre de historia: Registrar estadísticas jugadores: rojas, amarillas, goles, goles recibidos en el caso de porteros, autogoles y partidos jugados.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 8	**Sprint:** 3
Descripción: Como usuario vocal quiero registrar estadísticas jugadores para mantener un registro detallado y actualizado de las estadísticas disciplinarias de los jugadores en los partidos.	

Criterios de aceptación:

- **Criterio de Aceptación 1: Envió de datos correctos**

Cuando el cliente envié un id de torneo válido y
Cuando el cliente envié un id de partido válido y
Cuando el cliente envié jugadores válidos
Entonces el sistema retornará un json con el campo success en true
Y el sistema actualizará las estadísticas de los jugadores en la BD.

- **Criterio de Aceptación 2: Envió de un id de torneo incorrecto**

Cuando el cliente envié un id de torneo incorrecto
Entonces el sistema retornará un json con el campo no_existe_torneo en true
Y el sistema no actualizará las estadísticas de los jugadores.

- **Criterio de Aceptación 3: Envió de un id de torneo sin partidos**

Cuando el cliente envié un id de torneo sin partidos
Entonces el sistema retornará un json con el campo no_existen_partidos en true
Y el sistema no actualizará las estadísticas de los jugadores en la BD.

- **Criterio de Aceptación 4: Envió de un id de partido incorrecto**

Cuando el cliente envié un id de torneo válido y
Cuando el cliente envié un id de partido incorrecto
Entonces el sistema retornará un json con el campo no_existe_partido en true
Y el sistema no actualizará las estadísticas de los jugadores en la BD.

- **Criterio de Aceptación 5: Envió de un id de partido en un estado que no es jugado**

Cuando el cliente envié un id de torneo válido y
Cuando el cliente envié un id de partido en un estado que no es jugado
Entonces el sistema retornará un json con el campo partido_ y el estado del partido en true
Y el sistema no actualizará las estadísticas de los jugadores en la BD.

• **Criterio de Aceptación 6: Envió de datos vacíos** Cuando el cliente envié un id de torneo válido y Cuando el cliente envié un id de partido válido y Cuando el cliente envié jugadores vacíos Entonces el sistema retornará un json con el campo noHayDatos en estado true Y el sistema no actualizará las estadísticas de los jugadores en la BD. • **Criterio de Aceptación 7: Error en la actualización de estadísticas de jugador** Cuando exista un error en la actualización de estadísticas de jugadores en la BD Entonces el sistema retornará un json con el campo noInserto en estado true Y el sistema no actualizará las estadísticas de los jugadores en la BD.

Tabla 23. Historia de Usuario PI001-4

Historia de Usuario	
Identificador (ID): PI001-4	**Usuario:** Público general
Nombre de historia: Mostrar partidos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 5	**Sprint:** 3
Descripción: Como público general quiero ver partidos para conocer las fechas y horarios de los partidos.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de torneo válido Entonces el sistema retornará un json con el campo partidos en el cual estará todos los partidos de ese torneo. • **Criterio de Aceptación 2: Envió de un id de torneo incorrecto** Cuando el cliente envié un id de torneo incorrecto Entonces el sistema retornará un json con el campo no_existe_torneo en true. • **Criterio de Aceptación 3: Error en la consulta** Cuando exista un error en la consulta en la BD Entonces el sistema retornará un json con el campo partidos en vacío	

Tabla 24. Historia de Usuario PI001-6

Historia de Usuario	
Identificador (ID): PI001-6	**Usuario:** Público general
Nombre de historia: Mostrar partidos de cada equipo	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 5	**Sprint:** 3
Descripción: Como público general quiero ver los partidos de un equipo para ver la lista de partidos en los que un equipo específico ha participado o participará.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de torneo válido y Cuando el cliente envié un id de equipo válido	

Entonces el sistema retornará un json con el campo partidos en el cual estará todos los partidos de ese equipo en un torneo dado. • **Criterio de Aceptación 2: Envió de un id de torneo incorrecto** Cuando el cliente envié un id de torneo incorrecto Entonces el sistema retornará un json con el campo no_existe_torneo en true. • **Criterio de Aceptación 3: Envió de un id de equipo incorrecto** Cuando el cliente envié un id de torneo válido y Cuando el cliente envié un id de equipo incorrecto Entonces el sistema retornará un json con el campo no_existe_equipo en true. • **Criterio de Aceptación 4: Error en la consulta** Cuando exista un error en la consulta en la BD Entonces el sistema retornará un json con el campo partidos en vacío

Tabla 25. Historia de Usuario PI001-7

Historia de Usuario	
Identificador (ID): PI001-7	**Usuario:** Público general
Nombre de historia: Mostrar estadísticas de jugadores de equipo	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 5	**Sprint:** 3
Descripción: Como público general quiero ver las estadísticas de los jugadores para ver las estadísticas individuales de esos jugadores en un equipo específico.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de equipo válido Entonces el sistema retornará un json con el campo datos en el cual estará todas las estadísticas de los jugadores de un equipo dado. • **Criterio de Aceptación 2: Envió de un id de equipo incorrecto** Cuando el cliente envié un id de equipo incorrecto Entonces el sistema retornará un json con el campo no_existe_equipo en true. • **Criterio de Aceptación 3: Error en la consulta** Cuando exista un error en la consulta en la BD Entonces el sistema retornará un json con el campo datos en vacío.	

Tabla 26. Historia de Usuario PI001-8

Historia de Usuario	
Identificador (ID): PI001-8	**Usuario:** Público general
Nombre de historia: Mostrar información de jugador	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Sprint:** 3
Descripción: Como público general quiero ver un jugador para ver la información detallada de ese jugador específico en el sistema.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié una cédula válida	

Entonces el sistema retornará un json con el campo datos en el cual estará toda la información de un jugador.

- **Criterio de Aceptación 2: Envió de una cédula incorrecta**

Cuando el cliente envié una cédula incorrecta
Entonces el sistema retornará un json con el campo no_existe_CI en true.

- **Criterio de Aceptación 3: Error en la consulta**

Cuando exista un error en la consulta en la BD
Entonces el sistema retornará un json con el campo datos en vacío

Tabla 27. Historia de Usuario GP001-5

Historia de Usuario	
Identificador (ID): GP001-5	**Usuario:** Vocal
Nombre de historia: Registrar alineación	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alta
Puntos estimados: 5	**Sprint:** 3
Descripción: Como usuario vocal quiero registrar alineación para registrar la alineación de los equipos antes de los partidos en el sistema	

Criterios de aceptación:

- **Criterio de Aceptación 1: Envió de datos correctos**

Cuando el cliente envié un id de partido válido y
Cuando el cliente envié una alineación válida
Entonces el sistema retornará un json con el campo success en true
Y el sistema insertará la alineación en la BD.

- **Criterio de Aceptación 2: Envió de un id de partido incorrecto**

Cuando el cliente envié un id de partido incorrecto
Entonces el sistema retornará un json con el campo no_existe_partido en true
Y el sistema no insertará la alineación en la BD.

- **Criterio de Aceptación 3: Envió de un id de partido en un estado que no es por jugar**

Cuando el cliente envié un id de partido en un estado que no es por jugar
Entonces el sistema retornará un json con el campo partido_ y el estado del partido en true
Y el sistema no insertará la alineación en la BD.

- **Criterio de Aceptación 4: Envió de datos vacíos**

Cuando el cliente envié un id de partido válido y
Cuando el cliente envié una alineación vacía
Entonces el sistema retornará un json con el campo noHayDatos en estado true
Y el sistema no insertará la alineación en la BD.

- **Criterio de Aceptación 5: Envió de un id de partido que ya tiene registrado alineación**

Cuando el cliente envié un id de partido que ya tiene registrada una alienación
Entonces el sistema retornará un json con el existe_alienacion en true
Y el sistema no insertará la alineación en la BD.

- **Criterio de Aceptación 6: Error en inserción**

Cuando exista un error en la inserción en la BD
Entonces el sistema retornará un json con el campo noInserto en estado true

Y el sistema no insertará la alineación en la BD.

Tabla 28. Historia de Usuario PI001-9

Historia de Usuario	
Identificador (ID): PI001-9	**Usuario:** Público general
Nombre de historia: Mostrar alineación	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 3	**Sprint:** 3
Descripción: Como público general quiero ver las estadísticas de los jugadores para ver las estadísticas individuales de esos jugadores en un equipo específico.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de partido válido Entonces el sistema retornará un json con el campo datos en el cual estará la alineación de ese partido. • **Criterio de Aceptación 2: Envió de un id de partido incorrecto** Cuando el cliente envié un id de partido incorrecto Entonces el sistema retornará un json con el campo no_existe_partido en true. • **Criterio de Aceptación 3: Envió de un id de partido en un estado que no es jugado** Cuando el cliente envié un id de partido en un estado que no es jugado Entonces el sistema retornará un json con el campo partido_ y el estado del partido en true • **Criterio de Aceptación 4: Error en la consulta** Cuando exista un error en la consulta en la BD Entonces el sistema retornará un json con el campo datos en vacío.	

Tabla 29. Historia de Usuario GP001-6

Historia de Usuario	
Identificador (ID): GP001-6	**Usuario:** Vocal
Nombre de historia: Registrar informe de sanciones	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Sprint:** 3
Descripción: Como usuario vocal quiero registrar informe de sanciones para registrar informes de sanciones disciplinarias en el sistema junto con el árbitro del partido.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de partido válido y Cuando el cliente envié un informe del local válido y Cuando el cliente envié un informe del visitante válido y Cuando el cliente envié un árbitro válido Entonces el sistema retornará un json con el campo success en true Y el sistema insertará el informe en la BD. • **Criterio de Aceptación 2: Envió de un id de partido incorrecto** Cuando el cliente envié un id de partido incorrecto	

Entonces el sistema retornará un json con el campo no_existe_partido en true
Y el sistema no insertará el informe en la BD.

- **Criterio de Aceptación 3: Envió de un id de partido en un estado que no es jugado**

Cuando el cliente envié un id de partido en un estado que no es por jugar
Entonces el sistema retornará un json con el campo partido_no_jugado en estado true
Y el sistema no insertará el informe en la BD.

- **Criterio de Aceptación 4: Envió de un id de partido que ya tiene registrado una sanción**

Cuando el cliente envié un id de partido que ya tiene registrada una sanción
Entonces el sistema retornará un json con el existe_sancion en true
Y el sistema no insertará el informe en la BD.

- **Criterio de Aceptación 5: Envió de datos vacíos**

Cuando el cliente envié un id de partido válido y
Cuando el cliente envié un árbitro vacío
Entonces el sistema retornará un json con el campo noHayDatos en estado true
Y el sistema no insertará el informe en la BD.

- **Criterio de Aceptación 6: Error en inserción**

Cuando exista un error en la inserción en la BD
Entonces el sistema retornará un json con el campo noInserto en estado true
Y el sistema no insertará el informe en la BD.

Tabla 30. Historia de Usuario GP001-7

Historia de Usuario	
Identificador (ID): GP001-7	**Usuario:** Presidente
Nombre de historia: Registrar tribunal de sanciones	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Sprint:** 4
Descripción: Como usuario presidente quiero registrar el tribunal de sanciones para registrar información sobre el tribunal de sanciones en el sistema junto con sus responsable.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de partido válido y Cuando el cliente envié un tribunal del local válido y Cuando el cliente envié un tribunal del visitante válido y Cuando el cliente envié responsables válidos Entonces el sistema retornará un json con el campo success en true Y el sistema actualizará la sanción en la BD. • **Criterio de Aceptación 2: Envió de un id de partido incorrecto** Cuando el cliente envié un id de partido incorrecto Entonces el sistema retornará un json con el campo no_existe_partido en true Y el sistema no actualizará la sanción en la BD. • **Criterio de Aceptación 3: Envió de un id de partido en un estado que no es jugado**	

Cuando el cliente envié un id de partido en un estado que no es por jugar
Entonces el sistema retornará un json con el campo partido_no_jugado en estado true
Y el sistema no actualizará la sanción en la BD.

- **Criterio de Aceptación 4: Envió de un id de partido que no tiene registrado una sanción**

Cuando el cliente envié un id de partido que no tiene registrada una sanción
Entonces el sistema retornará un json con el no_existe_sancion en true
Y el sistema no modificará la sanción en la BD.

- **Criterio de Aceptación 5: Envió de datos vacíos**

Cuando el cliente envié un id de partido válido y
Cuando el cliente envié los responsables vacíos
Entonces el sistema retornará un json con el campo noHayDatos en estado true
Y el sistema no actualizará la sanción en la BD.

- **Criterio de Aceptación 6: Error en inserción**

Cuando exista un error en la inserción en la BD
Entonces el sistema retornará un json con el campo noInserto en estado true
Y el sistema no actualizará la sanción en la BD.

Tabla 31. Historia de Usuario GP001-8

Historia de Usuario	
Identificador (ID): GP001-8	**Usuario:** Presidente
Nombre de historia: Mostrar acta de juego	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alta
Puntos estimados: 3	**Sprint:** 3
Descripción: Como usuario presidente quiero ver el acta de juego para observar en el sistema información a los detalles completos del partido, como alineaciones, resultados, sanciones y estadísticas.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de torneo válido y Cuando el cliente envié un id de partido válido Entonces el sistema retornará un json con los campos datos_partido con la información del partio, estadisticas con las estaditicas de los jugadores en ese partido y sanciones con el informe de sanción y el arbitro. • **Criterio de Aceptación 2: Envió de un id de torneo incorrecto** Cuando el cliente envié un id de torneo incorrecto Entonces el sistema retornará un json con el campo no_existe_torneo en true. • **Criterio de Aceptación 3: Envió de un id de partido incorrecto** Cuando el cliente envié un id de torneo válido y Cuando el cliente envié un id de partido incorrecto Entonces el sistema retornará un json con el campo no_existe_partido en true. • **Criterio de Aceptación 4: Envió de un id de partido en un estado que no es jugado** Cuando el cliente envié un id de partido en un estado que no es por jugar	

Entonces el sistema retornará un json con el campo partido_ y el estado del partido en true. • **Criterio de Aceptación 5: Error en la consulta** Cuando exista un error en la consulta en la BD Entonces el sistema retornará un json con el campo datos_partido o estadisticas o sanciones en vacío.

Tabla 32. Historia de Usuario GU001-3

Historia de Usuario	
Identificador (ID): GU001-3	**Usuario:** Usuario Registrado
Nombre de historia: Recuperar contraseña	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alta
Puntos estimados: 8	**Sprint:** 4
Descripción: Como usuario registrado quiero recuperar mi contraseña para que en caso de olvido pueda volver a acceder al sistema.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** • **Criterio de Aceptación 5: Envió de datos vacíos** • **Criterio de Aceptación 6: Error en inserción**	

Tabla 33. Historia de Usuario GU001-4

Historia de Usuario	
Identificador (ID): GU001-4	**Usuario:** Usuario Registrado
Nombre de historia: Actualizar usuario	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alta
Puntos estimados: 3	**Sprint:** 4
Descripción: Como usuario registrado quiero actualizar mis datos para actualizar y modificar la información personal en el sistema.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un correo electrónico válido y Cuando el cliente envié un nombre válido y Cuando el cliente envié un tipo válido Entonces el sistema retornará un json con el campo success en true Y el sistema actualizará al usuario en la BD. • **Criterio de Aceptación 2: Envió de datos vacíos** Cuando el cliente envié un nombre vacío o Cuando el cliente envié un correo electrónico vacío o Cuando el cliente envié un tipo vacío Entonces el sistema retornará un json con el campo noHayDatos en estado true Y el sistema no actualizará al usuario en la BD. • **Criterio de Aceptación 3: Envió de correo electrónico no registrado** Cuando el cliente envié un correo electrónico no registrado Entonces el sistema retornará un json con el campo success en estado false Y el sistema no actualizará al usuario en la BD.	

Tabla 34. Historia de Usuario GL001-5

Historia de Usuario	
Identificador (ID): GL001-5	**Usuario:** Presidente
Nombre de historia: Actualizar categoría	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alta
Puntos estimados: 3	**Sprint:** 4
Descripción: Como usuario presidente quiero actualizar categoría para ajustar las categorías dentro del sistema.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un nombre válido y Cuando el cliente envié un número de equipos válido y Cuando el cliente envié un id de categoría válido Entonces el sistema retornará un json con el campo success en true Y el sistema actualizará la categoría en la BD. • **Criterio de Aceptación 2: Envió de un id de categoría incorrecto** Cuando el cliente envié un id de categoría incorrecto Entonces el sistema retornará un json con el campo no_existe_categoria en true Y el sistema no actualizará la categoría en la BD. • **Criterio de Aceptación 3: Envió de datos vacíos** Cuando el cliente envié un id de categoría válido Cuando el cliente envié un nombre vacío o Cuando el cliente envié un número de equipos vacío Entonces el sistema retornará un json con el campo noHayDatos en estado true Y el sistema no actualizará la categoría en la BD. • **Criterio de Aceptación 4: Error en actualización** Cuando exista un error en la actualización en la BD Entonces el sistema retornará un json con el campo noInserto en estado true.	

Tabla 35. Historia de Usuario GE001-6

Historia de Usuario	
Identificador (ID): GE001-6	**Usuario:** Presidente
Nombre de historia: Actualizar equipo	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 3	**Sprint:** 4
Descripción: Como usuario presidente quiero actualizar equipo para actualizar la información de los equipos participantes en los torneos.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un nombre válido y Cuando el cliente envié un presidente válido y Cuando el cliente envié un escudo válido y Cuando el cliente envié un id de equipo válido Entonces el sistema retornará un json con el campo success en true	

Y el sistema actualizará el equipo en la BD.

- **Criterio de Aceptación 2: Envió de un id de categoría incorrecto**

Cuando el cliente envié un id de equipo incorrecto
Entonces el sistema retornará un json con el campo no_existe_equipo en true
Y el sistema no actualizará el equipo en la BD.

- **Criterio de Aceptación 3: Envió de datos vacíos**

Cuando el cliente envié un id de equipo válido
Cuando el cliente envié un nombre vacío o
Cuando el cliente envié un presidente vacío o
Cuando el cliente envié un escudo vacío
Entonces el sistema retornará un json con el campo noHayDatos en estado true
Y el sistema no actualizará el equipo en la BD.

- **Criterio de Aceptación 4: Error en actualización**

Cuando exista un error en la actualización en la BD
Entonces el sistema retornará un json con el campo noInserto en estado true.

Tabla 36. Historia de Usuario GE001-7

Historia de Usuario	
Identificador (ID): GE001-7	**Usuario:** Presidente
Nombre de historia: Actualizar jugador	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alta
Puntos estimados: 3	**Sprint:** 4
Descripción: Como usuario presidente quiero actualizar jugador para realizar ajustes en la información de los jugadores registrados en los equipos participantes en los torneos.	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié una cédula válida y Cuando el cliente envié una posición válida y Cuando el cliente envié una foto válida y Cuando el cliente envié una estatura válida y Cuando el cliente envié un número de camiseta válida y Cuando el cliente envié un id de equipo válido Entonces el sistema retornará un json con el campo success en true Y el sistema actualizará el jugador en la BD. • **Criterio de Aceptación 2: Envió de un id de equipo incorrecto** Cuando el cliente envié un id de equipo incorrecto Entonces el sistema retornará un json con el campo no_existe_equipo en true Y el sistema no actualizará el jugador en la BD. • **Criterio de Aceptación 3: Envió de cédula existente** Cuando el cliente envié un id de equipo válido y Cuando el cliente envié una cédula existente Entonces el sistema retornará un json con el campo existe_ jugador en true Y el sistema no actualizará el jugador en la BD. • **Criterio de Aceptación 4: Envió de datos vacíos**	

Cuando el cliente envié un id de equipo válido y
Cuando el cliente envié una cédula válida y
Cuando el cliente envié una posición vacía o
Cuando el cliente envié una foto vacía o
Cuando el cliente envié una estatura vacía o
Cuando el cliente envié un número de camiseta vacío
Entonces el sistema retornará un json con el campo noHayDatos en estado true
Y el sistema no actualizará el jugador en la BD.

- **Criterio de Aceptación 5: Envió de número de camiseta existen en equipo**

Cuando el cliente envié un id de equipo válido y
Cuando el cliente envié una cédula válida y
Cuando el cliente envié una posición válida y
Cuando el cliente envié una foto válida y
Cuando el cliente envié una estatura válida y
Cuando el cliente envié un número de camiseta existente en equipo
Entonces el sistema retornará un json con el campo exite_numero_camiseta en true
Y el sistema no actualizará el jugador en la BD.

Tabla 37. Historia de Usuario GU001-5

Historia de Usuario	
Identificador (ID): GU001-5	**Usuario:** Usuario Registrado
Nombre de historia: Eliminar usuario	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Sprint:** 4
Descripción: Como usuario registrado quiero darme de baja para darse de baja del sistema	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un correo electrónico válido Entonces el sistema retornará un json con el campo success en true Y el sistema actualizará al usuario en estado desactivado en la BD. • **Criterio de Aceptación 2: Envió de datos vacíos** Cuando el cliente envié un correo electrónico vacío Entonces el sistema retornará un json con el campo noHayDatos en estado true Y el sistema no actualizará al usuario en estado desactivado en la BD. • **Criterio de Aceptación 3: Envió de correo electrónico no registrado** Cuando el cliente envié un correo electrónico no registrado Entonces el sistema retornará un json con el campo success en estado false Y el sistema no actualizará al usuario en estado desactivado en la BD. • **Criterio de Aceptación 4: Envió de correo electrónico que es presidente** Cuando el cliente envié un correo electrónico de presidente Entonces el sistema retornará un json con el campo es_presidente en estado true Y el sistema no actualizará al usuario en estado desactivado en la BD.	

• **Criterio de Aceptación 5: Error en la actualización** Cuando exista un error en la actualización en la BD Entonces el sistema retornará un json con el campo noInserto en estado true.

Tabla 38. Historia de Usuario GL001-6

Historia de Usuario	
Identificador (ID): GL001-6	**Usuario:** Presidente
Nombre de historia: Eliminar liga	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Sprint:** 4
Descripción: Como usuario presidente quiero eliminar liga para retirar de manera efectiva una liga específica del sistema	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de liga válido y Entonces el sistema retornará un json con el campo success en true Y el sistema desactivará la liga en la BD. • **Criterio de Aceptación 2: Envió de un id de liga incorrecto** Cuando el cliente envié un id de liga incorrecto Entonces el sistema retornará un json con el campo no_existe_liga en true Y el sistema no desactivará la liga en la BD.	

Tabla 39. Historia de Usuario GL001-7

Historia de Usuario	
Identificador (ID): GL001-7	**Usuario:** Presidente
Nombre de historia: Eliminar categoría	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Sprint:** 4
Descripción: Como usuario presidente quiero eliminar categoría para retirar de manera efectiva una categoría específica del sistema	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de categoría válido y Entonces el sistema retornará un json con el campo success en true Y el sistema desactivará la categoría en la BD. • **Criterio de Aceptación 2: Envió de un id de categoría incorrecto** Cuando el cliente envié un id de categoría incorrecto Entonces el sistema retornará un json con el campo no_existe_categoría en true • Y el sistema no desactivará la categoría en la BD.	

Tabla 40. Historia de Usuario GE001-8

Historia de Usuario	
Identificador (ID): GE001-8	**Usuario:** Presidente

Nombre de historia: Eliminar equipo	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Sprint:** 4
Descripción: Como usuario presidente quiero eliminar equipo para retirar de manera efectiva un equipo específico del sistema	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un id de equipo válido y Entonces el sistema retornará un json con el campo success en true Y el sistema desactivará el equipo en la BD. • **Criterio de Aceptación 2: Envió de un id de equipo incorrecto** Cuando el cliente envié un id de equipo incorrecto Entonces el sistema retornará un json con el campo no_existe_equipo en true • Y el sistema no desactivará el equipo en la BD.	

Tabla 41. Historia de Usuario GE001-9

	Historia de Usuario
Identificador (ID): GE001-9	**Usuario:** Presidente
Nombre de historia: Eliminar jugador	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Sprint:** 4
Descripción: Como usuario presidente quiero eliminar jugador para retirar de manera efectiva un jugador específico del sistema	
Criterios de aceptación: • **Criterio de Aceptación 1: Envió de datos correctos** Cuando el cliente envié un CI válido y Entonces el sistema retornará un json con el campo success en true Y el sistema desactivará el jugador en la BD. • **Criterio de Aceptación 2: Envió de un CI incorrecto** Cuando el cliente envié un CI incorrecto Entonces el sistema retornará un json con el campo no_existe_jugador en true • Y el sistema no desactivará el jugador en la BD.	

ANEXO II

Código	Historia de Usuario	Tareas
GU001-1	Registrar usuario	Método verificar usuario
		Método insertar usuario
		Prueba del método insertar usuario
		Prueba del método verificar usuario
GU001-2	Iniciar de sesión	Método verificar usuario
		Prueba del método verificar usuario
GL001-1	Registrar liga	Método verificar usuario
		Método mostrar liga presidente
		Método mostrar todas las ligas inscritas
		Método insertar liga
		Pruebas del método verificar usuario
		Pruebas del método mostrar liga
		Pruebas del método mostrar todas las ligas inscritas
		Pruebas del método insertar liga
GL001-2	Registrar categoría	Método verificar ID
		Método verificar nombre categoría
		Método insertar categoría
		Pruebas del método verificar ID
		Pruebas del método verificar nombre categoría
		Pruebas del método insertar categoría
GL001-3	Mostrar liga	Método verificar usuario
		Método mostrar liga presidente
		Prueba del método verificar usuario
		Pruebas del método mostrar liga presidente
GL001-4	Mostrar categorías	Método verificar ID
		Método mostrar categorías
		Pruebas del método verificar ID
		Pruebas del método mostrar categorías
GE001-1	Mostrar equipos de liga	Método verificar ID
		Método mostrar equipos liga
		Pruebas del método verificar ID
		Pruebas del método mostrar equipos liga
GE001-2	Validar número de equipos	Método verificar ID
		Método validar número de equipos
		Pruebas del método verificar ID

		Pruebas del método validar número de equipos
GE001-3	Registrar equipo	Método verificar ID
		Método mostrar categorías
		Método verificar nombre categoría
		Método insertar equipo
		Pruebas del método verificar ID
		Pruebas del método mostrar categorías
		Pruebas del método verificar nombre categoría
		Prueba del método insertar equipo
PI001-5	Mostrar equipo	Método verificar ID
		Método mostrar equipo
		Pruebas del método verificar ID
		Pruebas del método mostrar equipo
GE001-4	Registrar jugador	Método verificar ID
		Método verificar CI
		Método verificar camiseta
		Método insertar jugador
		Pruebas del método verificar ID
		Pruebas del método verificar CI
		Pruebas del método verificar camiseta
		Prueba del método insertar jugador
GT001-1	Registrar torneos	Método verificar ID
		Método insertar torneos
		Pruebas del método verificar ID
		Prueba del método insertar torneos
GT001-2	Generar partidos	Método verificar ID
		Método generar partidos
		Método insertar partidos
		Método insertar estadísticas equipo
		Pruebas del método verificar ID
		Prueba del método generar partidos
		Prueba del método insertar partidos
		Prueba del método insertar estadísticas equipo
GT001-3	Mostrar torneos	Método verificar ID
		Método mostrar torneos
		Pruebas del método verificar ID
		Pruebas del método mostrar torneos
GP001-1	Programar partidos	Método verificar ID
		Método modificar partido
		Pruebas del método verificar ID

		Pruebas del método modificar partido
PI001-1	Mostrar todas las ligas inscritas	Método mostrar ligas
		Pruebas del método mostrar ligas
PI001-2	Mostrar equipos de categoría	Método verificar ID
		Método mostrar equipos categoría
		Pruebas del método verificar ID
		Pruebas del método equipos categoría
GE001-5	Mostrar jugadores	Método verificar ID
		Método mostrar jugadores
		Pruebas del método verificar ID
		Pruebas del método mostrar jugadores
PI001-3	Mostrar posiciones	Método verificar ID
		Método mostrar posiciones
		Pruebas del método verificar ID
		Pruebas del método mostrar posiciones
GP001-2	Registrar resultados	Método verificar ID
		Método mostrar partidos
		Método modificar partidos
		Método actualizar estadísticas de equipo
		Método modificar partido
		Prueba del método verificar ID
		Prueba del método mostrar partidos
		Prueba del método modificar partidos
		Prueba del método actualizar estadísticas de equipo
		Prueba del método modificar partido
GP001-3	Comprobar registro de estadísticas de jugadores	Método verificar ID
		Método extraer partidos jugados
		Prueba del método verificar ID
		Pruebas del método extraer partidos jugados
GP001-4	Registrar estadísticas jugadores: rojas, amarillas, goles, goles recibidos en el caso de porteros, autogoles y partidos jugados.	Método verificar ID
		Método mostrar partidos
		Método extraer partidos jugados
		Método insertar actualizar estadísticas jugadores
		Prueba del método verificar ID
		Prueba del método mostrar partidos
		Pruebas del método extraer partidos jugados
		Prueba del método insertar actualizar estadísticas jugadores

PI001-4	Mostrar partidos	Método verificar ID
		Método mostrar partidos
		Prueba del método verificar ID
		Pruebas del método mostrar partidos
PI001-6	Mostrar partidos de cada equipo	Método verificar ID
		Método mostrar partidos equipo
		Prueba del método verificar ID
		Pruebas del método mostrar partidos equipo
PI001-7	Mostrar estadísticas de jugadores de equipo	Método verificar ID
		Método mostrar estadísticas jugador equipo
		Prueba del método verificar ID
		Pruebas del método mostrar estadísticas jugador equipo
PI001-8	Mostrar información de jugador	Método verificar CI
		Método mostrar información jugador
		Prueba del método verificar CI
		Pruebas del método mostrar información jugador
GP001-5	Registrar alineación	Método verificar ID
		Método verificar existencia alineación
		Método insertar alineación
		Prueba del método verificar ID
		Pruebas del método verificar existencia alineación
		Prueba del método insertar alineación
PI001-9	Mostrar alineación	Método verificar ID
		Método verificar existencia alineación
		Prueba del método verificar ID
		Pruebas del método verificar existencia alineación
GP001-6	Registrar informe de sanciones	Método verificar ID
		Método verificar sanción
		Método insertar informe sanción
		Prueba del método verificar ID
		Prueba del método verificar sanción
		Prueba del método insertar informe sanción
GP001-7	Registrar tribunal de sanciones	Método verificar ID
		Método verificar sanción
		Método insertar tribunal
		Prueba del método verificar ID
		Prueba del método verificar sanción
		Prueba del método insertar tribunal
GP001-8	Mostrar acta de juego	Método mostrar partido

		Método mostrar estadísticas jugador equipo partido
		Método mostrar sanción
		Pruebas del método mostrar partido
		Pruebas del método estadísticas jugador equipo partido
		Pruebas del método mostrar sanción
GU001-3	Recuperar contraseña	Método recuperar contraseña
		Prueba del método recuperar contraseña
GU001-4	Actualizar usuario	Método verificar usuario
		Método actualizar usuario
		Prueba del método verificar usuario
		Prueba del método actualizar usuario
GL001-5	Actualizar categoría	Método verificar ID
		Método verificar CI
		Método verificar camiseta
		Método actualizar jugador
		Pruebas del método verificar ID
		Pruebas del método verificar CI
		Pruebas del método verificar camiseta
		Prueba del método actualizar jugador
GE001-6	Actualizar equipo	Método verificar ID
		Método actualizar equipo
		Prueba del método verificar ID
		Prueba del método actualizar equipo
GE001-7	Actualizar jugador	Método actualizar jugador
		Prueba del método actualizar jugador
GU001-5	Eliminar usuario	Método verificar usuario
		Método eliminar usuario hincha
		Prueba del método verificar usuario
		Prueba del método eliminar usuario hincha
GL001-6	Eliminar liga	Método eliminar liga
		Prueba del método eliminar liga
GL001-7	Eliminar categoría	Método eliminar categoría
		Prueba del método eliminar categoría
GE001-8	Eliminar equipo	Método eliminar equipo
		Prueba del método eliminar equipo
GE001-9	Eliminar jugador	Método eliminar jugador
		Prueba del método eliminar jugador

ANEXO III

https://github.com/CarlosVelasquezZ/SGLigas_backend/tree/main/Modelo

ANEXO IV

Sprint 1		
Código	**Historia de Usuario**	**Tareas**
GU001-1	Registrar usuario	Método verificar usuario
		Método insertar usuario
		Prueba del método insertar usuario
		Prueba del método verificar usuario
GU001-2	Iniciar de sesión	Método verificar usuario
		Prueba del método verificar usuario
GL001-1	Registrar liga	Método verificar usuario
		Método mostrar liga presidente
		Método mostrar todas las ligas inscritas
		Método insertar liga
		Pruebas del método verificar usuario
		Pruebas del método mostrar liga
		Pruebas del método mostrar todas las ligas inscritas
		Pruebas del método insertar liga
GL001-2	Registrar categoría	Método verificar ID
		Método verificar nombre categoría
		Método insertar categoría
		Pruebas del método verificar ID
		Pruebas del método verificar nombre categoría
		Pruebas del método insertar categoría
GL001-3	Mostrar liga	Método verificar usuario
		Método mostrar liga presidente
		Prueba del método verificar usuario
		Pruebas del método mostrar liga presidente
GL001-4	Mostrar categorías	Método verificar ID
		Método mostrar categorías
		Pruebas del método verificar ID
		Pruebas del método mostrar categorías
GE001-1	Mostrar equipos de liga	Método verificar ID
		Método mostrar equipos liga
		Pruebas del método verificar ID
		Pruebas del método mostrar equipos liga
GE001-2	Validar número de equipos	Método verificar ID
		Método validar número de equipos
		Pruebas del método verificar ID

		Pruebas del método validar número de equipos
GE001-3	Registrar equipo	Método verificar ID
		Método mostrar categorías
		Método verificar nombre categoría
		Método insertar equipo
		Pruebas del método verificar ID
		Pruebas del método mostrar categorías
		Pruebas del método verificar nombre categoría
		Prueba del método insertar equipo
PI001-5	Mostrar equipo	Método verificar ID
		Método mostrar equipo
		Pruebas del método verificar ID
		Pruebas del método mostrar equipo
Sprint 2		
Código	**Historia de Usuario**	**Tareas**
GE001-4	Registrar jugador	Método verificar ID
		Método verificar CI
		Método verificar camiseta
		Método insertar jugador
		Pruebas del método verificar ID
		Pruebas del método verificar CI
		Pruebas del método verificar camiseta
		Prueba del método insertar jugador
GT001-1	Registrar torneos	Método verificar ID
		Método insertar torneos
		Pruebas del método verificar ID
		Prueba del método insertar torneos
GT001-2	Generar partidos	Método verificar ID
		Método generar partidos
		Método insertar partidos
		Método insertar estadísticas equipo
		Pruebas del método verificar ID
		Prueba del método generar partidos
		Prueba del método insertar partidos
		Prueba del método insertar estadísticas equipo
GT001-3	Mostrar torneos	Método verificar ID
		Método mostrar torneos
		Pruebas del método verificar ID
		Pruebas del método mostrar torneos
GP001-1	Programar partidos	Método verificar ID

		Método modificar partido
		Pruebas del método verificar ID
		Pruebas del método modificar partido
PI001-1	Mostrar todas las ligas inscritas	Método mostrar ligas
		Pruebas del método mostrar ligas
PI001-2	Mostrar equipos de categoría	Método verificar ID
		Método mostrar equipos categoría
		Pruebas del método verificar ID
		Pruebas del método equipos categoría
GE001-5	Mostrar jugadores	Método verificar ID
		Método mostrar jugadores
		Pruebas del método verificar ID
		Pruebas del método mostrar jugadores
PI001-3	Mostrar posiciones	Método verificar ID
		Método mostrar posiciones
		Pruebas del método verificar ID
		Pruebas del método mostrar posiciones
Sprint 3		
Código	**Historia de Usuario**	**Tareas**
GP001-2	Registrar resultados	Método verificar ID
		Método mostrar partidos
		Método modificar partidos
		Método actualizar estadísticas de equipo
		Método modificar partido
		Prueba del método verificar ID
		Prueba del método mostrar partidos
		Prueba del método modificar partidos
		Prueba del método actualizar estadísticas de equipo
		Prueba del método modificar partido
GP001-3	Comprobar registro de estadísticas de jugadores	Método verificar ID
		Método extraer partidos jugados
		Prueba del método verificar ID
		Pruebas del método extraer partidos jugados
GP001-4	Registrar estadísticas jugadores: rojas, amarillas, goles, goles recibidos en el caso de porteros, autogoles y partidos jugados.	Método verificar ID
		Método mostrar partidos
		Método extraer partidos jugados
		Método insertar actualizar estadísticas jugadores
		Prueba del método verificar ID

		Prueba del método mostrar partidos
		Pruebas del método extraer partidos jugados
		Prueba del método insertar actualizar estadísticas jugadores
PI001-4	Mostrar partidos	Método verificar ID
		Método mostrar partidos
		Prueba del método verificar ID
		Pruebas del método mostrar partidos
PI001-6	Mostrar partidos de cada equipo	Método verificar ID
		Método mostrar partidos equipo
		Prueba del método verificar ID
		Pruebas del método mostrar partidos equipo
PI001-7	Mostrar estadísticas de jugadores de equipo	Método verificar ID
		Método mostrar estadísticas jugador equipo
		Prueba del método verificar ID
		Pruebas del método mostrar estadísticas jugador equipo
PI001-8	Mostrar información de jugador	Método verificar CI
		Método mostrar información jugador
		Prueba del método verificar CI
		Pruebas del método mostrar información jugador
GP001-5	Registrar alineación	Método verificar ID
		Método verificar existencia alineación
		Método insertar alineación
		Prueba del método verificar ID
		Pruebas del método verificar existencia alineación
		Prueba del método insertar alineación
PI001-9	Mostrar alineación	Método verificar ID
		Método verificar existencia alineación
		Prueba del método verificar ID
		Pruebas del método verificar existencia alineación
GP001-6	Registrar informe de sanciones	Método verificar ID
		Método verificar sanción
		Método insertar informe sanción
		Prueba del método verificar ID
		Prueba del método verificar sanción
		Prueba del método insertar informe sanción
Sprint 4		
Código	**Historia de Usuario**	**Tareas**

GP001-7	Registrar tribunal de sanciones	Método verificar ID
		Método verificar sanción
		Método insertar tribunal
		Prueba del método verificar ID
		Prueba del método verificar sanción
		Prueba del método insertar tribunal
GP001-8	Mostrar acta de juego	Método mostrar partido
		Método mostrar estadísticas jugador equipo partido
		Método mostrar sanción
		Pruebas del método mostrar partido
		Pruebas del método estadísticas jugador equipo partido
		Pruebas del método mostrar sanción
GU001-3	Recuperar contraseña	Método recuperar contraseña
		Prueba del método recuperar contraseña
GU001-4	Actualizar usuario	Método verificar usuario
		Método actualizar usuario
		Método actualizar contraseña
		Prueba del método verificar usuario
		Prueba del método actualizar contraseña
		Prueba del método actualizar usuario
GL001-5	Actualizar categoría	Método verificar ID
		Método actualizar categoría
		Prueba del método verificar ID
		Prueba del método actualizar categoría
GE001-6	Actualizar equipo	Método verificar ID
		Método actualizar equipo
		Prueba del método verificar ID
		Prueba del método actualizar equipo
GE001-7	Actualizar jugador	Método actualizar jugador
		Prueba del método actualizar jugador
GU001-5	Eliminar usuario	Método verificar usuario
		Método eliminar usuario hincha
		Prueba del método verificar usuario
		Prueba del método eliminar usuario hincha
GL001-6	Eliminar liga	Método eliminar liga
		Prueba del método eliminar liga
GL001-7	Eliminar categoría	Método eliminar categoría
		Prueba del método eliminar categoría
GE001-8	Eliminar equipo	Método eliminar equipo
		Prueba del método eliminar equipo

GE001-9	Eliminar jugador	Método eliminar jugador
		Prueba del método eliminar jugador

ANEXO V

Código “GU001-1 Registrar Usuario”

```php
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la inserción de usuarios en una base de datos.
 */

/**
 * @mainpage Documentación de Inserción de Usuarios en una Base de Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la inserción de usuarios en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_usuario.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Inserta un nuevo usuario en la base de datos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param string $correo_usuario El correo del usuario.
    * @param string $nombre El nombre del usuario.
    * @param string $contraseña La contraseña del usuario.
    * @param string $tipo El tipo de usuario (por ejemplo, 'presidente' o 'hincha').
```

```
    */
    $correo_usuario = isset($_POST['correo']) ? trim($_POST['correo']) : "";
    $nombre = isset($_POST['nombre']) ? trim($_POST['nombre']) : "";
    $contrasena = isset($_POST['password']) ? trim($_POST['password']) : "";
    $tipo = isset($_POST['tipo']) ? trim($_POST['tipo']) : "";

    // Crea una instancia de la clase usuario.
    $objeto_usuario = new usuario();

    // Verifica que los datos no estén vacíos.
    if (!empty($correo_usuario) && !empty($nombre) && !empty($contrasena) &&
!empty($tipo)) {

        // Consulta para verificar que ese usuario no existe.
        $existe_usuario = $objeto_usuario-
>verificar_usuario($correo_usuario, $conexion);
        if (empty($existe_usuario)) {

            // Consulta para insertar usuario.
            $resultado = $objeto_usuario->insertar_usuario($correo_usuario,
$nombre, $contrasena, $tipo, $conexion);

            // Verifica que se insertó el usuario.
            if ($resultado) {
                echo json_encode(array('success' => true));
            } else {
                echo json_encode(array('noInserto' => true));
            }
        } else {
            // El usuario ya existe y no se puede insertar.
            echo json_encode(array('success' => false));
        }
    } else {
        // Al menos una de las variables está vacía.
        echo json_encode(array('noHayDatos' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “GU001-2 Iniciar de sesión”

```
<?php
/**
```

```php
 * @file
 * Este archivo contiene un script PHP para manejar la autenticación de usuarios en el sistema basado
 * en los datos ingresados previamente en una base de datos.
 */

/**
 * @mainpage Documentación de Autenticación de Usuarios
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la autenticación de usuarios en un sistema basado en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de datos, así como la librería firebase/php-jwt para generar y manejar tokens JWT.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_usuario.php');
include("conexion.php");
require 'vendor/autoload.php'; // La librería firebase/php-jwt
use \Firebase\JWT\JWT; // Clase JWT

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Realiza la autenticación del usuario y maneja el inicio de sesión.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param string $correo_usuario El correo del usuario.
    * @param string $contraseña La contraseña del usuario.
    */
    $correo_usuario = isset($_POST['correo']) ? trim($_POST['correo']) : "";
    $contrasena = isset($_POST['password']) ? trim($_POST['password']) : "";

    // Crea una instancia de la clase usuario.
```

```php
    $objeto_usuario = new usuario();

    // Verifica que los datos no estén vacíos.
    if (!empty($correo_usuario) && !empty($contrasena)) {

        // Consulta para verificar si el usuario existe.
        $existe_usuario = $objeto_usuario-
>verificar_usuario($correo_usuario, $conexion);
        if (!empty($existe_usuario)) {

            // Obtiene la contraseña registrada en la BD.
            $contrasena_encriptada = $existe_usuario['contraseña'];

            // Comprueba si la contraseña es correcta.
            if (password_verify($contrasena, $contrasena_encriptada)) {

                    // Inicio de sesión exitoso y verificar si el usuario es
presidente.
                    $userType = $existe_usuario['tipo_usuario'];
                    $correoUsuario = $existe_usuario['correo'];
                    // Generar el token JWT.
                    $key = 'admin_web1019';
                    $payload = array(
                        "correo" => $correoUsuario,
                        "tipo_usuario" => $userType
                    );
                    $token = JWT::encode($payload, $key, 'HS256');
                    $response = array(
                        'token' => $token,
                        'userData' => array(
                            'correo' => $correoUsuario,
                            'tipo_usuario' => $userType
                        )
                    );
                    echo json_encode($response);
            } else {
                // Contraseña incorrecta.
                echo json_encode(array('success' => false));
            }
        } else {
            // Usuario no encontrado.
            echo json_encode(array('correo' => false));
        }
    } else {
        // Al menos una de las variables está vacía.
        echo json_encode(array('noHayDatos' => true));
    }
```

```
    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “GL001-1 Registrar liga”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para insertar una nueva liga en una
base de datos.
 */

/**
 * @mainpage Documentación de Inserción de Liga
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la inserción de una nueva liga en
una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_liga.php');
include('clases/clase_usuario.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Inserta una nueva liga en la base de datos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
```

```
    *
    * @param string $nombre_liga El nombre de la liga a insertar.
    * @param string $fecha_fundacion La fecha de fundación de la liga en
formato 'YYYY-MM-DD'.
    * @param string $direccion La dirección de la liga.
    * @param string $correo_admin El correo del administrador de la liga.
    */
    $nombre_liga = isset($_POST['nombre_liga']) ?
trim($_POST['nombre_liga']) : "";
    $fecha_fundacion = isset($_POST['fecha_fundacion']) ?
trim($_POST['fecha_fundacion']) : "";
    $direccion = isset($_POST['direccion']) ? trim($_POST['direccion']) :
"";
    $correo_admin = isset($_POST['correo_admin']) ?
trim($_POST['correo_admin']) : "";
    $estado="activo";

    // Crea instancias de las clases a ser usadas.
    $objeto_usuario = new usuario();
    $objeto_liga = new liga();

    // Verifica que los datos no estén vacíos.
    if (!empty($nombre_liga) && !empty($fecha_fundacion) &&
!empty($direccion) && !empty($correo_admin)) {

        // Verifica si existe ese correo
        $verificar = $objeto_usuario->verificar_usuario($correo_admin,
$conexion);
        if (!empty($verificar)) {

            // Verifica si el usuario es presidente.
            if ($verificar['tipo_usuario'] == "presidente") {

                // Verifica si el correo no tiene una liga registrada.
                $verificar = $objeto_liga-
>mostrar_liga_presidente($correo_admin, $conexion);
                if (empty($verificar)) {

                    // Verifica si existe una liga con el nombre ingresado.
                    $verificar = $objeto_liga-
>mostrar_todas_ligas($nombre_liga, $conexion);
                    if (empty($verificar)) {

                        // Consulta para insertar liga.
                        $resultado = $objeto_liga-
>insertar_liga($nombre_liga, $fecha_fundacion, $direccion, $correo_admin,
$estado, $conexion);
```

```
                        // Verifica que se insertó la liga.
                        if ($resultado) {

                            // Obtiene los datos de la liga insertada.
                            $liga_insertada = $objeto_liga-
>mostrar_liga_presidente($correo_admin, $conexion);

                            // Enviar datos al cliente
                            echo json_encode(array('success' => true,
'datos' => $liga_insertada));
                        } else {
                            echo json_encode(array('noInserto' => true));
                        }
                    } else {
                        // Existe una liga registrada con ese nombre.
                        echo json_encode(array('existe_nombre' => true));
                    }
                } else {
                    // Existe una liga registrada con el correo ingresado.
                    echo json_encode(array('existe_registro' => true));
                }
            } else {
                // Si el usuario no es presidente.
                echo json_encode(array('no_es_presidente' => true));
            }
        } else {
            // Si el usuario no existe.
            echo json_encode(array('no_existe_usuario' => true));
        }
    } else {
        // Al menos una de las variables está vacía.
        echo json_encode(array('noHayDatos' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “GL001-2 Registrar categoría”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la inserción de
categorías en una base de datos.
```

```php
 */

/**
 * @mainpage Documentación de Inserción de Categorías en una Base de Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la inserción de categorías en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_categoria.php');
include('clases/clase_liga.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Realiza la inserción de la categoría en la base de datos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param string $nombre El nombre de la categoría a insertar.
    * @param int $num_equipos El número de equipos en la categoría.
    * @param int $id_liga El ID de la liga a la que se asociará la categoría.
    */
    $nombre = isset($_POST['categoria']) ? trim($_POST['categoria']) : "";
    $num_equipos = isset($_POST['num_equipos']) ? trim($_POST['num_equipos']) : "";
    $id_liga = isset($_POST['id_liga']) ? intval(trim($_POST['id_liga'])) : 0;
    $estado="activo";

    // Crea instancias de las clases a ser usadas.
```

```php
    $objeto_liga = new liga();
    $objeto_categoria = new categoria();

    // Verifica si existe ese ID de liga.
    $id_liga = $objeto_liga->verificar_ID($id_liga, $conexion);
    if ($id_liga > 0) {

        // Verifica que los datos no estén vacíos.
        if (!empty($nombre) && !empty($num_equipos)) {

            // Verifica que no exista una categoría con el mismo nombre en
la liga.
            $verificar = $objeto_categoria-
>verificar_nombre_categoria($nombre, $id_liga, $conexion);
            if (empty($verificar)) {

                // Inserta la nueva categoría en la base de datos.
                $resultado = $objeto_categoria->insertar_categoria($nombre,
$num_equipos, $estado, $id_liga, $conexion);

                // Verifica que se insertó la categoría.
                if ($resultado) {
                    echo json_encode(array('success' => true));
                } else {
                    echo json_encode(array('noInserto' => true));
                }
            } else {
                // Si existe una categoría con el mismo nombre.
                echo json_encode(array('existe_nombre' => true));
            }
        } else {
            // Al menos una de las variables está vacía.
            echo json_encode(array('noHayDatos' => true));
        }
    } else {
        // Si la liga no existe.
        echo json_encode(array('no_existe_liga' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “GL001-3 Mostrar liga”

```php
<?php
/**
 * @file
 * Este archivo contiene un script PHP para mostrar la liga de un presidente
en la base de datos.
 */

/**
 * @mainpage Documentación de Mostrar Liga de Presidente
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la consulta de liga de un
presidente en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_liga.php');
include('clases/clase_usuario.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Muestra las liga de un presidente en la base de datos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param string $correo_admin El correo del administrador de la liga.
    */
    $correo_admin = isset($_POST['correo_admin']) ?
trim($_POST['correo_admin']) : "";

    // Crea instancias de las clases a ser usadas.
    $objeto_usuario = new usuario();
    $objeto_liga = new liga();
```

```php
    // Verifica si existe ese correo.
    $verificar = $objeto_usuario->verificar_usuario($correo_admin,
$conexion);
    if (!empty($verificar)){

        // Verifica si el correo pertenece a un presidente.
        if ($verificar['tipo_usuario'] == "presidente") {

            // Obtiene los datos de la liga del correo ingresado.
            $resultado = $objeto_liga-
>mostrar_liga_presidente($correo_admin, $conexion);

            //Devuelve la información detallada de la liga en formato JSON.
            echo json_encode(array('datos' => $resultado));
        } else {
            // Si el usuario no es presidente.
            echo json_encode(array('no_es_presidente' => true));
        }
    } else {
        // Si el usuario no existe.
        echo json_encode(array('no_existe_usuario' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “GL001-4 Mostrar categorías”

```php
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la obtención de
categorías de una liga en una base de datos.
 */

/**
 * @mainpage Documentación de Obtención de Categorías de una Liga en una
Base de Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la obtención de categorías de una
liga en una base de datos.
```

```php
 * Utiliza clases y archivos necesarios para conectar y manipular la base de datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_categoria.php');
include('clases/clase_liga.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Obtiene las categorías de una liga y las devuelve en formato JSON.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_liga El ID de la liga a la que se asociará la categoría.
    */
    $id_liga = isset($_POST['id_liga']) ? intval(trim($_POST['id_liga'])) : 0;

    // Crea instancias de las clases a ser usadas.
    $objeto_liga = new liga();
    $objeto_categoria = new categoria();

    // Verifica si existe el ID de liga ingresado.
    $id_liga = $objeto_liga->verificar_ID($id_liga, $conexion);
    if ($id_liga > 0) {

        // Obtiene las categorías de una liga.
        $resultado = $objeto_categoria->mostrar_categorias($id_liga, $conexion);

        //Devuelve la información la categoria en formato JSON, si no hay datos retorna array vacio.
        echo json_encode(array('datos' => $resultado));
```

```
    } else {
        // Si la liga no existe.
        echo json_encode(array('no_existe_liga' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “GE001-2 Validar número de equipos”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para verificar el límite de equipos
en una categoría en una base de datos.
 */

/**
 * @mainpage Documentación de Verificar Límite de Equipos en una Categoría
en una Base de Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para verificar si un torneo ha alcanzado su
límite de equipos en una categoría en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_equipo.php');
include('clases/clase_categoria.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Obtiene la información del numero de equipos en una categoría.
 */
```

```php
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_categoria El ID de la categoría para validar el número
de equipos.
    */
    $id_categoria=isset($_POST['id_categoria']) ?
intval(trim($_POST['id_categoria'])) : 0;

    // Crea instancias de las clases a ser usadas.
    $objeto_categoria = new categoria();
    $objeto_equipo = new equipo();

    // Verifica si existe ese ID de categoria.
    $verificar=$objeto_categoria->verificar_ID($id_categoria,$conexion);
    if ($verificar>0) {

        // Consultar el numero de equipos en esa categoria.
        $verificar=$objeto_equipo-
>verificar_num_equipos($id_categoria,$conexion);

        // Verificar que se obtuvo de la consulta.
        if($verificar){
            echo json_encode(array('limite_equipos' => true));
        } else {
            echo json_encode(array('limite_equipos' => false));
        }
    }
    else {
        // Si la categoría no existe.
        echo json_encode(array('no_existe_categoria' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

Código “GE001-3 Registrar equipo”

```php
<?php
```

```php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la inserción de un equipo en una base de datos.
 */

/**
 * @mainpage Documentación de Inserción de un Equipo en una Base de Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la inserción de un equipo en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de datos.
 */

 // Incluye las clases y archivos necesarios.
include('clases/clase_equipo.php');
include('clases/clase_categoria.php');
include('clases/clase_liga.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param string $nombre_equipo El nombre del equipo a insertar.
    * @param string $fecha_fundacion La fecha de fundación del equipo en formato 'YYYY-MM-DD'.
    * @param string $presidente El nombre del presidente del equipo.
    * @param string $colores Los colores del equipo en formato  RGB (Red, Green, Blue) hexadecimal ejemplo (#00ff04).

    * @param string $escudo La URL del escudo del equipo.
    * @param int $id_categoria El ID de la categoría a la que pertenece el equipo.
    * @param int $id_liga El ID de la liga a la que pertenece el equipo.
    */
```

```php
    $nombre_equipo=isset($_POST['nombre_equipo']) ?
trim($_POST['nombre_equipo']) : "";
    $fecha_fundacion=isset($_POST['fecha_fundacion']) ?
trim($_POST['fecha_fundacion']) : "";
    $presidente=isset($_POST['presidente']) ? trim($_POST['presidente']) :
"";
    $colores=isset($_POST['color']) ? trim($_POST['color']) : "";
    $escudo=isset($_POST['escudo']) ? trim($_POST['escudo']) : "";
    $id_categoria=isset($_POST['id_categoria']) ?
intval(trim($_POST['id_categoria'])) : 0;
    $id_liga = isset($_POST['id_liga']) ? intval(trim($_POST['id_liga'])) :
0;
    $estado="activo";

    // Crea instancias de las clases a ser usadas.
    $objeto_categoria = new categoria();
    $objeto_liga = new liga();
    $objeto_equipo = new equipo();

    // Verifica si existe ese ID de liga.
    $verificar=$objeto_liga->verificar_ID($id_liga,$conexion);
    if($verificar){

        //Obtiene las categorias de asociadas al ID de liga ingresado.
        $categoria=$objeto_categoria-
>mostrar_categorias($id_liga,$conexion);

        //Verifica que exista el ID de categoria ingresado en la liga.
        $verificar=false;
        foreach ($categoria as $elemento) {
            if ($elemento["id_categoria"] === $id_categoria) {
                $verificar = true;
                break; // Terminar el bucle cuando se encuentra el
resultado.
            }
        }
        if($verificar){
            // Verificar que los datos no esten vacios.
            if (!empty($nombre_equipo) && !empty($fecha_fundacion) &&
!empty($presidente) && !empty($colores) && !empty($escudo)) {

                // Verificar si existe el nombre de equipo.
                $verificar=$objeto_equipo-
>verificar_nombre_equipo($nombre_equipo,$id_liga,$conexion);
                if(!$verificar){

                    // Consulta para insertar equipo.
```

```
                    $resultado=$objeto_equipo-
>insertar_equipo($nombre_equipo,$fecha_fundacion,$presidente,$colores,$escud
o,$estado,$id_categoria,$conexion);

                    // Verificar que se inserto el equipo.
                    if ($resultado) {
                        echo json_encode(array('success' => true));
                    } else {
                        echo json_encode(array('noInserto' => true));
                    }
                } else {
                    echo json_encode(array('existe_nombre' => true));
                }
            }
            else {
                // Al menos una de las variables está vacía.
                echo json_encode(array('noHayDatos' => true));
            }
        } else {
            // Si la categoría no existe.
            echo json_encode(array('no_existe_categoria' => true));
        }
    } else {
        // Si la liga no existe.
        echo json_encode(array('no_existe_liga' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “GE001-1 Mostrar equipos de liga”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para mostrar equipos de una liga en
una base de datos.
 */

/**
 * @mainpage Documentación de Mostrar Equipos de una Liga en una Base de
Datos
 *
 * @section intro_sec Introducción
```

```php
 * Este script PHP se utiliza para manejar la obtención de equipos de una
liga en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_equipo.php');
include('clases/clase_liga.php');
include('clases/clase_categoria.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Obtiene la información de los equipos de una liga.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_liga El ID de la liga para mostrar los equipos.
    */
    $id_liga = isset($_POST['id_liga']) ? intval(trim($_POST['id_liga'])) :
0;

    // Crea instancias de las clases a ser usadas.
    $objeto_liga = new liga();
    $objeto_equipo = new equipo();

    // Verifica si existe ese ID de liga.
    $id_liga = $objeto_liga->verificar_ID($id_liga, $conexion);
    if ($id_liga > 0) {

        // Obtiene los resultados.
        $resultado = $objeto_equipo->mostrar_equipos_ligas($id_liga,
$conexion);
```

```
        // Devuelve los equipos de una liga en formato JSON, si no hay datos retorna array vacio.
        echo json_encode(array('datos' => $resultado));
    } else {
        // Si la liga no existe
        echo json_encode(array('no_existe_liga' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

Código “PI001-5 Mostrar equipo”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para mostrar información detallada de un equipo en una base de datos.
 */

/**
 * @mainpage Documentación de Mostrar Información Detallada de un Equipo en una Base de Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la obtención de información detallada de un equipo en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_equipo.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}
```

```php
/**
 * Obtiene información detallada de un equipo.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_equipo El ID del equipo para mostrar.
    */
    $id_equipo = isset($_POST['id_equipo']) ?
intval(trim($_POST['id_equipo'])) : 0;

    // Crea una instancia de la clase equipo.
    $objeto_equipo = new equipo();

    // Verifica si existe ese ID de equipo.
    $verificar=$objeto_equipo->verificar_ID($id_equipo,$conexion);
    if ($id_equipo > 0) {

        // Obtiene los resultados para mostrar los equipos.
        $resultado = $objeto_equipo->mostrar_equipo($id_equipo, $conexion);

        // Devuelve la informacion detallada del equipo en formato JSON, si
no hay datos retorna array vacio.
        echo json_encode(array('datos' => $resultado));
    } else {
        // Si el equipo no existe.
        echo json_encode(array('no_existe_equipo' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

ANEXO VI

Código “GE001-4 Registrar jugador”

```php
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la inserción de
jugadores en una base de datos.
 */

/**
 * @mainpage Documentación de Manejar Inserción de Jugadores en una Base de
Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la inserción de jugadores en una
base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_equipo.php');
include('clases/clase_jugador.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Ejecuta la función principal para insertar un nuevo jugador.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $CI El número de cédula del jugador.
    * @param string $nombre El nombre del jugador.
    * @param string $posicion La posición del jugador.
```

```
    * @param string $fecha_nacimiento La fecha de nacimiento del jugador en
formato 'YYYY-MM-DD'.
    * @param string $foto La URL de la foto del jugador.
    * @param int $estatura La estatura del jugador en cm ejemplo (160).
    * @param int $num_camiseta El numero de camiseta del jugador.
    * @param int $id_equipo El ID del equipo al que pertenece el jugador.
    */
    $CI = isset($_POST['CI']) ? intval(trim($_POST['CI'])) : 0;
    $nombre = isset($_POST['nombre']) ? trim($_POST['nombre']) : "";
    $posicion = isset($_POST['posicion']) ? trim($_POST['posicion']) : "";
    $fecha_nacimiento = isset($_POST['fecha_nacimiento']) ?
trim($_POST['fecha_nacimiento']) : "";
    $foto = isset($_POST['foto']) ? trim($_POST['foto']) : "";
    $estatura = isset($_POST['estatura']) ? intval(trim($_POST['estatura']))
: 0;
    $num_camiseta = isset($_POST['num_camiseta']) ?
intval(trim($_POST['num_camiseta'])) : 0;
    $id_equipo = isset($_POST['id_equipo']) ?
intval(trim($_POST['id_equipo'])) : 0;
    $estado = "activo";

    // Crea instancias de las clases a ser usadas.
    $objeto_equipo = new equipo();
    $objeto_jugador = new jugador();

    // Verifica si existe ese ID de equipo.
    $id_equipo = $objeto_equipo->verificar_ID($id_equipo, $conexion);
    if ($id_equipo > 0) {

        // Verificar si la CI de ese jugador no existe.
        $verificar=$objeto_jugador->verificar_CI($CI,$conexion);
        if(empty($verificar)){

            // Verifica que los datos no estén vacíos.
            if (!empty($nombre) && !empty($posicion) &&
!empty($fecha_nacimiento) && !empty($foto) && !empty($estatura) &&
!empty($num_camiseta)) {

                // Consulta para verificar el numero de camiseta.
                $verificar=$objeto_jugador->verificar_camiseta($id_equipo,
$num_camiseta, $conexion);
                if($verificar == 0){

                    // Inserta el nuevo jugador en la base de datos.
                    $resultado = $objeto_jugador->insertar_jugador($CI,
$nombre, $posicion, $foto, $fecha_nacimiento, $estatura, $id_equipo,
$estado, $num_camiseta, $conexion);
```

```
                    // Verificar que se inserto el jugador.
                    if ($resultado) {
                        echo json_encode(array('success' => true));
                    } else {
                        echo json_encode(array('noInserto' => true));
                    }
                } else {
                    // Existe el numero de camiseta.
                    echo json_encode(array('exite_numero_camiseta' =>
true));
                }
            } else {
                // Al menos una de las variables está vacía.
                echo json_encode(array('noHayDatos' => true));
            }
        } else {
            // Si existe el jugador.
            echo json_encode(array('existe_jugador' => true));
        }
    } else {
        // Si el equipo no existe.
        echo json_encode(array('no_existe_equipo' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

Código “GT001-1 Registrar torneos”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la inserción de torneos
en una base de datos.
 */

/**
 * @mainpage Documentación de Manejar Inserción de Torneos en una Base de
Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la inserción de torneos en una
base de datos.
```

```php
 * Utiliza clases y archivos necesarios para conectar y manipular la base de datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_torneo.php');
include('clases/clase_categoria.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Maneja la inserción de torneos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param string $etapa El nombre de la etapa del torneo.
    * @param string $fecha_inicio La fecha de inicio del torneo en formato 'YYYY-MM-DD'.
    * @param string $fecha_fin La fecha de fin del torneo en formato 'YYYY-MM-DD'.
    * @param string $canchas Los nombres de las canchas separadas por como ejemplo (cancha1,cancha2).
    * @param int $grupo El numero de cuantos grupos habrán en caso que se etapa de grupos.

    * @param int $num_clasificados El número de equipos que clasifican en cada etapa, si es etapa de grupos se debe separr por coma ejemplo (2,2).
    * @param int $id_categoria El ID de la categoría a la que pertenece el torneo.
    */
    $etapa = isset($_POST['etapa']) ? trim($_POST['etapa']) : "";
    $fecha_inicio = isset($_POST['fecha_inicio']) ? trim($_POST['fecha_inicio']) : "";
    $fecha_fin = isset($_POST['fecha_fin']) ? trim($_POST['fecha_fin']) : "";
    $canchas = isset($_POST['canchas']) ? trim($_POST['canchas']) : "";
    $grupo = isset($_POST['grupo']) ? trim($_POST['grupo']) : "";
```

```
    $num_clasificados = isset($_POST['num_clasificados']) ?
trim($_POST['num_clasificados']) : "";
    $id_categoria = isset($_POST['id_categoria']) ?
intval(trim($_POST['id_categoria'])) : 0;

    // Crea instancias de las clases a ser usadas.
    $objeto_torneo = new torneo();
    $objeto_categoria = new categoria();

    // Verifica si existe ese ID de categoria.
    $id_categoria = $objeto_categoria->verificar_ID($id_categoria,
$conexion);
    if ($id_categoria > 0) {

        // Verifica que los datos no estén vacíos.
        if (!empty($etapa) && !empty($fecha_inicio) && !empty($fecha_fin) &&
!empty($canchas) && !empty($num_clasificados)) {

            // Verifica que los datos no estén vacíos si es etapa de grupos.
            if(!empty($grupo)){
                $array = explode(",", $num_clasificados);
                // Inserta los nuevos torneos en la base de datos.
                for($i=0;$i<$grupo;$i++){
                    $resultado[$i] = $objeto_torneo->insertar_torneo($etapa,
$fecha_inicio, $fecha_fin, "GRUPO ".$i+1, $array[$i], $id_categoria,
$canchas, $conexion);
                }

                // Devuelve los id de los torneos insertados.
                if (!empty($resultado)) {
                    echo json_encode(array('success' => true,'datos' =>
$resultado));
                } else {
                    echo json_encode(array('noInserto' => true));
                }
            }
            else{

                // Inserta el nuevo torneo en la base de datos.
                $resultado = $objeto_torneo->insertar_torneo($etapa,
$fecha_inicio, $fecha_fin, $grupo, $num_clasificados, $id_categoria,
$canchas, $conexion);

                // Verificar que se inserto el torneo.
                if ($resultado>0) {
                    echo json_encode(array('success' => true,'datos' =>
$resultado));
                } else {
```

```
                    echo json_encode(array('noInserto' => true));
                }
            }
        } else {
            // Al menos una de las variables está vacía.
            echo json_encode(array('noHayDatos' => true));
        }
    } else {
        // Si la categoría no existe.
        echo json_encode(array('no_existe_categoria' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “GT001-2 Generar partidos”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la generación de
partidos en un torneo.
 */

/**
 * @mainpage Documentación de Generación de Partidos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la generación de partidos en un
torneo.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_partido.php');
include('clases/clase_torneo.php');
include('clases/clase_equipo.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
```

```php
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Maneja la generación de partidos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_torneo El ID del torneo que se quieren generar partido.
    * @param array $equipos Un array con los id de los equipos participantes
en el torneo
    */
    $id_torneo = isset($_POST['id_torneo']) ?
intval(trim($_POST['id_torneo'])) : 0;
    $equipos = isset($_POST['equipos']) ? $_POST['equipos'] : "";

    // Crea instancias de las clases a ser usadas.
    $objeto_torneo = new torneo();
    $objeto_partido = new partido();
    $objeto_equipo = new equipo();

    // Verifica si existe ese torneo.
    $verificar = $objeto_torneo->verificar_ID($id_torneo, $conexion);
    if(!empty($verificar)){
        // Verificar si ya existen partidos creados en ese torneo.
        $existen_partidos = $objeto_partido-
>mostrar_partidos($id_torneo,$conexion);
        if(empty($existen_partidos)){

            // Verificar que los datos no esten vacios.
            if(!empty($equipos)){

               // Obtener el id de categoria asociado a ese torneo.
                $id_categoria=$verificar['id_categoria'];

                // Verficar los id de equipos.
                $cont=0;
                $valido=false;
                for($i=0; $i < count($equipos); $i++){
                    $verificar=$objeto_equipo-
>verificar_ID($equipos[$i],$conexion);
                    if($verificar){
```

```
                        $cont++;
                    }
                    if($i==count($equipos)-1){
                        if($i==$cont-1){
                            $valido=true;
                        }
                    }
                }

                // Verificar si todos los id de equipos son correctos.
                if($valido){
                    $num_equipos=count($equipos);// Obtener le numero de
equipos

                    // Se registran las estadisticas de los equipos
inicialmente en 0
                    $objeto_equipo-
>registar_estadisticas_equipo($equipos,$id_torneo,$conexion);

                    // Verificar si el numero de equipos es impar
                    if($num_equipos%2!=0){
                        $equipos[$num_equipos]=30; //se asigna el id de
equipo descansa
                        $num_equipos++;//se aumenta el num de equipos porque
es impar
                    }
                    $num_partidos=$num_equipos/2;

                    // Consulta para ganerar los partidos para ese torneo.
                    $resultado = $objeto_partido-
>generar_partidos($equipos,$num_equipos);

                    // Consulta para insertar equipos.
                    $respuesta = $objeto_partido-
>insertar_partido($resultado, $num_equipos, $id_torneo, $conexion);

                    // Verificar que se inserto los partidos.
                    if ($respuesta) {
                        echo json_encode(array('success' => true));
                    } else {
                        echo json_encode(array('noInserto' => true));
                    }
                } else {
                    // Si no existe equipo o equipos.
                    echo json_encode(array('no_existen_equipo' => true));
                }
            } else {
                // Al menos una de las variables está vacía.
```

```
                echo json_encode(array('noHayDatos' => true));
            }
        } else {
            // Si no existen partidos.
            echo json_encode(array('existen_partidos' => true));
        }
    } else {
        // Si el torneo no existe.
        echo json_encode(array('no_existe_torneo' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

Código “GT001-3 Mostrar torneos”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para mostrar los torneos de una
categoría de una base de datos.
 */

/**
 * @mainpage Documentación de Mostrar Torneos por Categoría
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para mostrar torneos de una categoría en una
base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_torneo.php');
include('clases/clase_categoria.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
```

```php
    return conexion_DB();
}

/**
 * Muestra los torneos de una categoria.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_categoria El ID de la categoría para mostrar los
torneo.
    */
    $id_categoria = isset($_POST['id_categoria']) ?
intval(trim($_POST['id_categoria'])) : 0;

    // Crea instancias de las clases a ser usadas.
    $objeto_torneo = new torneo();
    $objeto_categoria = new categoria();

    // Verifica si existe ese ID de categoría.
    $id_categoria = $objeto_categoria->verificar_ID($id_categoria,
$conexion);
    if ($id_categoria > 0) {

        // Obtiene los resultados para mostrar los torneos de una categoría.
        $resultado = $objeto_torneo->mostrar_torneos($id_categoria,
$conexion);

        // Devuelve los torneos de una categoria en formato JSON, si no hay
datos retorna array vacio.
        echo json_encode(array('datos' => $resultado));
    } else {
        // Si la categoría no existe.
        echo json_encode(array('no_existe_categoria' => true));
    }
    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);

}

// Ejecuta la función principal.
main();

?>
```

Código “GP001-1 Programar partidos”

```php
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la actualización de un partido en la base de datos.
 */

/**
 * @mainpage Documentación de Actualización de Partidos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la actualización de un partido en la base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_partido.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Maneja la actualización de un partido en la base de datos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_partido El ID del partido que se quiere programa, el partido debe estar en programa o pospuesto.
    * @param array $partido Un array con los datos del partido. Por ejemplo
    * [YYYY-MM-DD,hh:mm,cancha1,equipo_escogido,""],el dato de veedor es opcional
    * El orden de cada array de partido debe ser [fecha_partido,hora_partido,cancha,vocal,veedor]
```

```
    */
    $id_partido = isset($_POST['id_partido']) ?
intval(trim($_POST['id_partido'])) : 0;
    $partido = isset($_POST['partido']) ? $_POST['partido'] : "";
    $estado = "porjugar";

    // Crea una instancia de la clase partido.
    $objeto_partido = new partido();

    // Verifica si existe ese torneo.
    $verificar = $objeto_partido->verificar_ID($id_partido, $conexion);
    if(!empty($verificar)){

        // Obtener el estado del partido.
        $estado_BD=$verificar['estado'];

        // Verificar el estado del partido.
        switch ($estado_BD) {
            case "programar":
            case "posponer":
                // Verificar que los datos no esten vacios.
                if(!empty($partido)){

                    // Actualizar partido en la base de datos.
                    $respuesta=$objeto_partido-
>modificar_partido($id_partido, $partido, $estado, $conexion);

                    // Verificar que se actualizo el partido.
                    if ($respuesta) {
                        echo json_encode(array('success' => true));
                    } else {
                        echo json_encode(array('noInserto' => true));
                    }
                } else {
                    // Al menos una de las variables está vacía.
                    echo json_encode(array('noHayDatos' => true));
                }
                break;
            case "porjugar":
                echo json_encode(array('partido_programado' => true));
                break;
            case "jugado":
                echo json_encode(array('partido_jugado' => true));
                break;
            default:
                echo json_encode(array('error' => true));
        }
    } else {
```

```
        // Si el partido no existe.
        echo json_encode(array('no_existe_partido' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

Código “PI001-1 Mostrar todas las ligas inscritas”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para mostrar todas las ligas activas
en la base de datos.
 */

/**
 * @mainpage Documentación de Mostrar Todas las Ligas activas
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para mostrar todas las ligas activas en una
base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_liga.php');
include('conexion.php');



/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Muestra todas las ligas activas en la base de datos.
 */
```

```php
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    // Crea una instancia de la clase liga.
    $objeto_liga = new liga();

    // Obtiene los datos de todas las ligas en la BD.
    $resultado = $objeto_liga->mostrar_todas_ligas('', $conexion);

    //Devuelve la información de todas las liga activas en formato JSON.
    echo json_encode(array('datos' => $resultado));

    // Cierra la conexión.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

Código “PI001-2 Mostrar equipos de categoría”

```php
<?php
/**
 * @file
 * Este archivo contiene un script PHP para mostrar equipos de una categoría en una base de datos.
 */

/**
 * @mainpage Documentación de Mostrar Equipos de una Categoría en una Base de Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la obtención de equipos de una categoría en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_equipo.php');
include('clases/clase_categoria.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
```

```php
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Obtiene la información de los equipos de una categoría.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_categoria El ID de la categoría para mostrar los
equipos.
    */
    $id_categoria = isset($_POST['id_categoria']) ?
intval(trim($_POST['id_categoria'])) : 0;

    // Crea instancias de las clases a ser usadas.
    $objeto_categoria = new categoria();
    $objeto_equipo = new equipo();

    // Verifica si existe ese ID de categoria.
    $verificar=$objeto_categoria->verificar_ID($id_categoria,$conexion);
    if ($id_categoria>0) {

        // Obtiene los resultados para mostrar los equipos de una categoria.
        $resultado = $objeto_equipo-
>mostrar_equipos_categoria($id_categoria, $conexion);

        // Devuelve los equipos de una categoria en formato JSON, si no hay
datos retorna array vacio.
        echo json_encode(array('datos' => $resultado));
    } else {
        // Si la categoría no existe.
        echo json_encode(array('no_existe_categoria' => true));
    }
    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
```

```
?>
```

Código “GE001-5 Mostrar jugadores”

```php
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la visualización de
jugadores de un equipo en una base de datos.
 */

/**
 * @mainpage Documentación de Manejar Visualización de Jugadores de un
Equipo en una Base de Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la visualización de jugadores de
un equipo en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_jugador.php');
include('clases/clase_equipo.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Muestra los jugadores de un equipo.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_equipo El ID del equipo para mostrar los jugadores.
    */
    $id_equipo = isset($_POST['id_equipo']) ?
intval(trim($_POST['id_equipo'])) : 0;
```

```php
    // Crea instancias de las clases a ser usadas.
    $objeto_jugador = new jugador();
    $objeto_equipo = new equipo();

    // Verifica si existe ese ID de equipo.
    $id_equipo = $objeto_equipo->verificar_ID($id_equipo, $conexion);
    if ($id_equipo > 0) {
        // Obtiene los jugadores de un equipo.
        $resultado = $objeto_jugador->mostrar_jugadores($id_equipo,
$conexion);

        // Devuelve los jugadores de un equipo en formato JSON, si no hay
datos retorna array vacio.
        echo json_encode(array('datos' => $resultado));
    } else {
        // Si el equipo no existe.
        echo json_encode(array('no_existe_equipo' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

Código “PI001-3 Mostrar posiciones”

```php
<?php
/**
 * @file
 * Este archivo contiene un script PHP para mostrar las posiciones de un
torneo en una base de datos.
 */

/**
 * @mainpage Documentación de Mostrar las Posiciones de un Torneo en una
Base de Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la obtención de las posiciones de
un torneo en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */
```

```php
// Incluye las clases y archivos necesarios.
include('clases/clase_partido.php');
include('clases/clase_torneo.php');
include('clases/clase_equipo.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Obtiene las posiciones de un torneo y la devuelve en formato JSON.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_torneo El ID del torneo para mostrar las posiciones.
    */
    $id_torneo = isset($_POST['id_torneo']) ?
intval(trim($_POST['id_torneo'])) : 0;

    // Crea instancias de las clases a ser usadas.
    $objeto_torneo = new torneo();
    $objeto_equipo = new equipo();

    // Verifica si existe ese torneo.
    $verificar = $objeto_torneo->verificar_ID($id_torneo, $conexion);
    if(!empty($verificar)){

        // Obtiene las posiciones del torneo.
        $resultado = $objeto_equipo->mostrar_posiciones($id_torneo,
$conexion);

        // Devuelve las posiciones en formato JSON, si no hay datos retorna
array vacio.
        echo json_encode(array('tabla' => $resultado));
    } else {
        // Si el torneo no existe.
        echo json_encode(array('no_existe_torneo' => true));
    }
```

```php
    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

ANEXO VII

Código “GP001-2 Registrar resultados”

```php
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la insercción de
resultados en una base de datos.
 */

/**
 * @mainpage Documentación de Inserción de Resultados
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la inserción de resultados en
partidos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_partido.php');
include('clases/clase_torneo.php');
include('clases/clase_jugador.php');
include('clases/clase_equipo.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Maneja la insercción de resultados en la base de datos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_partido El ID del partido que se quiere insertar los
resultados debe estar en estado jugado.
```

```
    * @param int $id_torneo El ID del torneo al que pertenece ese partido
    * @param array $partido Un array con los datos del partido. Por ejemplo
    * [67,2,3,69]
    * El orden de cada array de partido debe ser
[id_equipo_local,goles_local,goles_visitantes,id_equipo_visitante]
    */
    $id_torneo = isset($_POST['id_torneo']) ?
intval(trim($_POST['id_torneo'])) : 0;
    $id_partido = isset($_POST['id_partido']) ?
intval(trim($_POST['id_partido'])) : 0;
    $partido = isset($_POST['partido']) ? $_POST['partido'] : "";
    $estado='jugado';

    // Crea instancias de las clases a ser usadas.
    $objeto_torneo = new torneo();
    $objeto_partido = new partido();
    $objeto_jugador= new jugador();
    $objeto_equipo= new equipo();

    // Verifica si existe ese torneo.
    $verificar = $objeto_torneo->verificar_ID($id_torneo, $conexion);
    if(!empty($verificar)){

        // Verificar si ya existen partidos creados en ese torneo.
        $existen_partidos = $objeto_partido-
>mostrar_partidos($id_torneo,$conexion);
        if(!empty($existen_partidos)){

            // Verifica si existe ese partido.
            $verificar = $objeto_partido->verificar_ID($id_partido,
$conexion);
            if(!empty($verificar)){

                // Obtener el estado del partido.
                $estado_BD=$verificar['estado'];

                // Verificar el estado del partido.
                switch ($estado_BD) {
                    case "porjugar":

                        // Verificar que los datos no esten vacios.
                        if(!empty($partido)){

                            // Actualizar resultado en la base de datos.
                            $respuesta=$objeto_partido-
>modificar_resultado($id_partido, $partido, $estado, $conexion);
                            if ($respuesta) {
```

```
                                // Actualizar las estadisticas de equipo en
la base de datos.
                                $respuesta=$objeto_equipo-
>actualizar_estadisticas_equipo($partido, $id_torneo, $conexion);

                                // Verificar que se actualizo las
estadisticas.
                                if($respuesta){
                                    echo json_encode(array('success' =>
true));
                                } else {
                                    echo
json_encode(array('no_inserto_estadistica' => true));
                                }
                            } else {
                                echo json_encode(array('no_modifico_partido'
=> true));
                            }
                        } else {
                            // Al menos una de las variables está vacía.
                            echo json_encode(array('noHayDatos' => true));
                        }
                        break;
                    case "jugado":
                        echo json_encode(array('partido_jugado' => true));
                        break;
                    case "programar":
                        echo json_encode(array('partido_no_programado' =>
true));
                        break;
                    case "posponer":
                        echo json_encode(array('partido_pospuesto' =>
true));
                        break;
                    default:
                        echo json_encode(array('error' => true));
                }
            } else {
                // Si el partido no existe.
                echo json_encode(array('no_existe_partido' => true));
            }
        } else {
            // Si no existe partidos.
            echo json_encode(array('no_existen_partidos' => true));
        }
    } else {
        // Si el torneo no existe.
        echo json_encode(array('no_existe_torneo' => true));
```

```
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

Código “GP001-3 Comprobar registro de estadísticas de jugadores”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la verificación de
estadisticas de jugadores.
 */

/**
 * @mainpage Documentación de Verificación de estadisticas de jugadores.
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la estadisticas de jugadores. en
un torneo.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_partido.php');
include('clases/clase_torneo.php');
include('clases/clase_jugador.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Maneja la verificación de estadisticas de jugadores.
 */
function main() {
    // Obtiene la conexión a la base de datos.
```

```php
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_partido El ID del partido que ya fue jugado.
    * @param int $id_torneo El ID del torneo al que pertenece ese partido.
    */
    $id_partido = isset($_POST['id_partido']) ?
intval(trim($_POST['id_partido'])) : 0;
    $id_torneo = isset($_POST['id_torneo']) ?
intval(trim($_POST['id_torneo'])) : 0;

    // Crea instancias de las clases a ser usadas.
    $objeto_partido = new partido();
    $objeto_torneo= new torneo();
    $objeto_jugador= new jugador();

    // Verifica si existe ese torneo.
    $verificar=$objeto_torneo->verificar_ID($id_torneo,$conexion);
    if(!empty($verificar)){

        // Verifica si existe ese partido.
        $verificar = $objeto_partido->verificar_ID($id_partido, $conexion);

        if(!empty($verificar)){

            // Obtener el estado del partido.
            $estado_BD=$verificar['estado'];

            // Verificar el estado del partido.
            switch ($estado_BD) {
                case "jugado":
                    // Obtener las estadisticas de jugadores.
                    $respuesta=$objeto_jugador-
>extraer_partidos_jugados($id_torneo,$id_partido,$conexion);

                    // Verificar que existen estadisticas de jugadores.
                    if($respuesta){
                        echo json_encode(array('existen_registros' =>
true));
                    } else {
                        echo json_encode(array('no_existen_registros' =>
true));
                    }
                    break;
                case "porjugar":
                    echo json_encode(array('partido_por_jugar' => true));
```

```
                    break;
                case "programar":
                    echo json_encode(array('partido_no_programado' =>
true));
                    break;
                case "posponer":
                    echo json_encode(array('partido_pospuesto' => true));
                    break;
                default:
                    echo json_encode(array('error' => true));
            }
        } else {
            // Si el partido no existe.
            echo json_encode(array('no_existe_partido' => true));
        }
    } else {
        // Si el torneo no existe.
        echo json_encode(array('no_existe_torneo' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

Código “GP001-4 Registrar estadísticas jugadores: rojas, amarillas, goles, goles recibidos en el caso de porteros, autogoles y partidos jugados.”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la inserción de
estadísticas de jugadores en partidos.
 */
/**
 * @mainpage Documentación de Inserción de Estadísticas de Jugadores
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la inserción de estadísticas de
jugadores en partidos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
```

```php
include('clases/clase_partido.php');
include('clases/clase_torneo.php');
include('clases/clase_jugador.php');
include('clases/clase_equipo.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Maneja la inserción de estadísticas de jugadores en partidos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_partido El ID del partido que se quiere insertar las
estaditicas debe estar en estado jugado.
    * @param int $id_torneo El ID del torneo al que pertenece ese partido
    * @param array $jugadores Una matriz con un array de las estaditicas de
jugadores. Por ejemplo
    * [[1234567890,1,0,1,0,1],[1234567891,1,0,1,0,1]]
    * El orden de cada array de jugador debe ser
[CI_jugador,rojas,amarillas,goles,goles_recibidos,autogoles]
    */
    $id_torneo = isset($_POST['id_torneo']) ?
intval(trim($_POST['id_torneo'])) : 0;
    $id_partido = isset($_POST['id_partido']) ?
intval(trim($_POST['id_partido'])) : 0;
    $jugadores = isset($_POST['jugadores']) ? $_POST['jugadores'] : "";

    // Crea instancias de las clases a ser usadas.
    $objeto_torneo = new torneo();
    $objeto_partido = new partido();
    $objeto_jugador= new jugador();
    $objeto_equipo= new equipo();

    // Verifica si existe ese torneo.
    $verificar = $objeto_torneo->verificar_ID($id_torneo, $conexion);
    if(!empty($verificar)){
```

```
        // Verificar si ya existen partidos creados en ese torneo.
        $existen_partidos = $objeto_partido-
>mostrar_partidos($id_torneo,$conexion);
        if(!empty($existen_partidos)){

            // Verifica si existe ese partido.
            $verificar = $objeto_partido->verificar_ID($id_partido,
$conexion);
            if(!empty($verificar)){

                // Obtener el estado del partido.
                $estado_BD=$verificar['estado'];

                // Verificar el estado del partido.
                switch ($estado_BD) {
                    case "jugado":

                        // Obtener las estadisticas de jugadores.
                        $respuesta=$objeto_jugador-
>extraer_partidos_jugados($id_torneo,$id_partido,$conexion);
                        if($respuesta){
                            echo json_encode(array('existen_registros' =>
true));
                        } else {

                            // Verificar que los datos no esten vacios.
                            if(!empty($jugadores)){
                                // Consulta para insertar estadisticas de
jugadores.
                                $respuesta=$objeto_jugador-
>insertar_actualizar_estadisticas_jugador($jugadores,$id_torneo,$id_partido,
$conexion);

                                // Verificar que se inserto las estadisticas
de jugadores.
                                if($respuesta){
                                    echo json_encode(array('success' =>
true));
                                } else {
                                    echo json_encode(array('noInserto' =>
true));
                                }
                            } else {
                                // Al menos una de las variables está vacía.
                                echo json_encode(array('noHayDatos' =>
true));
                            }
                        }
```

```
                        break;
                    case "porjugar":
                        echo json_encode(array('partido_por_jugar' =>
true));
                        break;
                    case "programar":
                        echo json_encode(array('partido_no_programado' =>
true));
                        break;
                    case "posponer":
                        echo json_encode(array('partido_pospuesto' =>
true));
                        break;
                    default:
                        echo json_encode(array('error' => true));
                }
            } else {
                // Si el partido no existe.
                echo json_encode(array('no_existe_partido' => true));
            }
        } else {
            // Si no existe partidos.
            echo json_encode(array('no_existen_partidos' => true));
        }
    } else {
        // Si el torneo no existe.
        echo json_encode(array('no_existe_torneo' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

Código “PI001-4 Mostrar partidos”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la obtención de partidos
en un torneo.
 */

/**
 * @mainpage Documentación de Obtención de Partidos en un Torneo
```

```php
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la obtención de partidos en un
torneo.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_torneo.php');
include('clases/clase_partido.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Maneja la obtención de partidos en un torneo.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_torneo El ID del torneo para mostrar los partidos.
    */
    $id_torneo = isset($_POST['id_torneo']) ?
intval(trim($_POST['id_torneo'])) : 0;

    // Crea instancias de las clases a ser usadas.
    $objeto_torneo = new torneo();
    $objeto_partido = new partido();

    // Verifica si existe ese torneo.
    $verificar = $objeto_torneo->verificar_ID($id_torneo, $conexion);
    if(!empty($verificar)){
        // Obtiene los partidos de ese torneo.
        $resultado = $objeto_partido-
>mostrar_partidos($id_torneo,$conexion);
```

```
        // Devuelve los partidos en formato JSON, si no hay datos retorna
array vacio.
        echo json_encode(array('partidos' => $resultado));
    } else {
        // Si el torneo no existe.
        echo json_encode(array('no_existe_torneo' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “PI001-6 Mostrar partidos de cada equipo”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la obtención de partidos
por equipo en un torneo.
 */

/**
 * @mainpage Documentación de Obtención de Partidos por Equipo en un Torneo
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la obtención de partidos por
equipo en un torneo.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_partido.php');
include('clases/clase_torneo.php');
include('clases/clase_equipo.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}
```

```php
/**
 * Maneja la obtención de partidos por equipo en un torneo.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_torneo El ID del torneo.
    * @param int $id_equipo El ID del equipo para mostra los partidos en ese
torneo.
    */
    $id_torneo = isset($_POST['id_torneo']) ?
intval(trim($_POST['id_torneo'])) : 0;
    $id_equipo = isset($_POST['id_equipo']) ?
intval(trim($_POST['id_equipo'])) : 0;

    // Crea instancias de las clases a ser usadas.
    $objeto_torneo = new torneo();
    $objeto_partido = new partido();
    $objeto_equipo = new equipo();

    // Verifica si existe ese torneo.
    $verificar = $objeto_torneo->verificar_ID($id_torneo, $conexion);
    if(!empty($verificar)){

        // Verficar si existe ese equipo.
        $verificar = $objeto_equipo->verificar_ID($id_equipo,$conexion);
        if(!empty($verificar)){

            // Consulta para obtener los partidos de un equipo.
            $resultado = $objeto_partido-
>mostrar_partidos_equipos($id_torneo, $id_equipo, $conexion);

            // Devuelve los partidos de un equipo en formato JSON, si no hay
datos retorna array vacio.
            echo json_encode(array('partidos' => $resultado));
        } else {
            // Si el equipo no existe.
            echo json_encode(array('no_existe_equipo' => true));
        }
    } else {
        // Si el torneo no existe.
        echo json_encode(array('no_existe_torneo' => true));
    }
```

```
    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

Código “PI001-7 Mostrar estadísticas de jugadores de equipo”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la visualización de
estadísticas de los jugadores de un equipo en una base de datos.
 */

/**
 * @mainpage Documentación de Manejar Visualización de Estadísticas de los
Jugadores de un Equipo en una Base de Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la visualización de estadísticas
de los jugadores de un equipo en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_jugador.php');
include('clases/clase_equipo.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Muestra las estadísticas de los jugador de un equipo.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();
```

```
    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_equipo El ID del equipo para mostrar las estadisticas.
    */
    $id_equipo = isset($_POST['id_equipo']) ?
intval(trim($_POST['id_equipo'])) : 0;

    // Crea instancias de las clases a ser usadas.
    $objeto_jugador = new jugador();
    $objeto_equipo = new equipo();

    // Verifica si existe ese ID de equipo.
    $id_equipo = $objeto_equipo->verificar_ID($id_equipo, $conexion);
    if ($id_equipo > 0) {

        // Obtiene las estadísticas de los jugadores de un equipo.
        $resultado = $objeto_jugador-
>mostrar_estadisticas_jugador_equipo($id_equipo, $conexion);

        // Devuelve las estadisticas de los jugadores de un equipo formato
JSON, si no hay datos retorna array vacio.
        echo json_encode(array('datos' => $resultado));
    } else {
        // Si el equipo no existe.
        echo json_encode(array('no_existe_equipo' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

Código “PI001-8 Mostrar información de jugador”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la visualización de
información detallada de un jugador en una base de datos.
 */

/**
 * @mainpage Documentación de Manejar Visualización de Información Detallada
de un Jugador en una Base de Datos
```

```php
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la visualización de información
detallada de un jugador en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_jugador.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Muestra la información detallada de un jugador.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $CI El número de cédula del jugador para mostrar la
información.
    */
    $CI = isset($_POST['CI']) ? intval(trim($_POST['CI'])) : 0;

    // Crea una instancia de la clase jugador.
    $objeto_jugador = new jugador();

    // Verifica si existe esa CI.
    $verificar=$objeto_jugador->verificar_CI($CI,$conexion);
    if ($verificar > 0) {
        // Obtiene los resultados para mostrar la informacion del jugador.
        $resultado = $objeto_jugador->mostrar_informacion_jugador($CI,
$conexion);

        // Devuelve la informacion de un jugador en formato JSON, si no hay
datos retorna array vacio.
        echo json_encode(array('datos' => $resultado));
```

```
    } else {
        // Si la CI no existe.
        echo json_encode(array('no_existe_CI' => true));
    }
    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “GP001-5 Registrar alineación”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la inserción de
alineaciones en partidos.
 */

/**
 * @mainpage Documentación de Inserción de Alineaciones
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la inserción de alineaciones en
partidos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_partido.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Maneja la inserción de alineaciones en partidos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();
```

```php
    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_partido El ID del partido que se quiere insertar la
alineacion, el partido debe estar en programa.
    * @param json $alineacion Un json con la informacion de la alineación de
los equipos local y visitante.
    */
    $id_partido = isset($_POST['id_partido']) ?
intval(trim($_POST['id_partido'])) : 0;
    $alineacion = isset($_POST['alineacion']) ? $_POST['alineacion'] : "";

    // Crea una instancia de la clase partido.
    $objeto_partido = new partido();

    // Verifica si existe ese partido.
    $verificar = $objeto_partido->verificar_ID($id_partido, $conexion);
    if(!empty($verificar)){

        // Obtener el estado del partido.
        $estado_BD=$verificar['estado'];

        // Verificar el estado del partido.
        switch ($estado_BD) {
            case "porjugar":
                // Verificar que los datos no esten vacios.
                if(!empty($alineacion)){

                    // Consulta para verificar si existen una alineación
registrada para ese partido.
                    $verfificar_id_partido=$objeto_partido-
>verificar_existencia_alineacion($id_partido, $conexion);
                    if(!empty($verfificar_id_partido)){
                        // Si existe registro de alineación.
                        echo json_encode(array('existe_alienacion' =>
true));
                    } else {

                        // Consulta para insertar alineación.
                        $respuesta=$objeto_partido-
>insertar_alineacion($alineacion,$id_partido,$conexion);

                        // Verificar que se inserto alineación.
                        if($respuesta){
                            echo json_encode(array('success' => true));
                        } else {
                            echo json_encode(array('noInserto' => true));
```

```
                        }
                    }
                } else {
                    // Al menos una de las variables está vacía.
                    echo json_encode(array('noHayDatos' => true));
                }
                break;
            case "jugado":
                echo json_encode(array('partido_jugado' =>
true));
                break;
            case "programar":
                echo json_encode(array('partido_no_programado' => true));
                break;
            case "posponer":
                echo json_encode(array('partido_pospuesto' => true));
                break;
            default:
                echo json_encode(array('error' => true));
        }
    } else {
        // Si el partido no existe.
        echo json_encode(array('no_existe_partido' => true));
    }
    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

Código “PI001-9 Mostrar alineación”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la obtención de
alineaciones de jugadores en partidos.
 */

/**
 * @mainpage Documentación de Obtención de Alineaciones de Jugadores
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la obtención de alineaciones de
jugadores en partidos.
```

```php
 * Utiliza clases y archivos necesarios para conectar y manipular la base de datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_partido.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Maneja la obtención de alineaciones de jugadores en partidos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_partido El ID del partido que se quiere mostrar la alineación el partido debe estar en estado jugado o por jugar.
    */
    $id_partido = isset($_POST['id_partido']) ? intval(trim($_POST['id_partido'])) : 0;

    // Crea una instancia de la clase partido.
    $objeto_partido = new partido();

    // Verifica si existe ese partido.
    $verificar = $objeto_partido->verificar_ID($id_partido, $conexion);
    if(!empty($verificar)){

        // Obtener el estado del partido.
        $estado_BD=$verificar['estado'];

        // Verificar el estado del partido.
        switch ($estado_BD) {
            case "porjugar":
            case "jugado":
                // Consulta para obtener la alineacion del partido.
```

```
                $alineacion=$objeto_partido-
>verificar_existencia_alineacion($id_partido, $conexion);

                // Devuelve la alineación en formato JSON, si no hay datos
retorna array vacio.
                echo json_encode(array('datos' =>
$alineacion['datos_alineacion']));
                break;
            case "programar":
                echo json_encode(array('partido_no_programado' => true));
                break;
            case "posponer":
                echo json_encode(array('partido_pospuesto' => true));
                break;
            default:
                echo json_encode(array('error' => true));
        }
    } else {
        // Si el partido no existe.
        echo json_encode(array('no_existe_partido' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “GP001-6 Registrar informe de sanciones”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la inserción del informe
de sanciones en partidos.
 */

/**
 * @mainpage Documentación de Inserción del informe de Sanciones
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la inserción del informe de
sanciones en partidos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */
```

```php
// Incluye las clases y archivos necesarios.
include('clases/clase_sanciones.php');
include('clases/clase_partido.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Maneja la inserción del informe de sanciones en partidos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_partido El ID del partido al que se asocia la sanción.
    * @param string $informe_local El informe del equipo local sobre la
sanción puede no exitir.
    * @param string $informe_visitante El informe del equipo visitante sobre
la sanción puede no exitir.
    * @param string $arbitro El nombre del árbitro que sancionó.
    */
    $id_partido = isset($_POST['id_partido']) ?
intval(trim($_POST['id_partido'])) : 0;
    $informe_local = isset($_POST['informe_local']) ?
trim($_POST['informe_local']) : "";
    $informe_visitante = isset($_POST['informe_visitante']) ?
trim($_POST['informe_visitante']) : "";
    $arbitro = isset($_POST['arbitro']) ? trim($_POST['arbitro']) : "";

    // Crea instancias de las clases a ser usadas.
    $objeto_sanciones = new sanciones();
    $objeto_partido = new partido();

    // Verifica si existe ese ID de partido.
    $verificar = $objeto_partido->verificar_ID($id_partido, $conexion);
    if (!empty($verificar)){

        // Obtener el estado del partido.
        $estado=$verificar['estado'];
```

```php
        // Verificar el esatdo del partido.
        if($estado==='jugado'){

            // Consulta para verificar si existe una sancion registrada.
            $verificar=$objeto_sanciones->verificar_sancion($id_partido,
$conexion);
            if($verificar==0){

                // Verificar que los datos no esten vacios.
                if(!empty($arbitro)){

                    // Inseratar sanciones en la base de datos.
                    $respuesta=$objeto_sanciones-
>insertar_sancion($id_partido, $informe_local, $informe_visitante, $arbitro,
$conexion);

                    // Verificar que se inserto la sanción.
                    if ($respuesta) {
                        echo json_encode(array('success' => true));
                    } else {
                        echo json_encode(array('noInserto' => true));
                    }
                } else {
                    // Al menos una de las variables está vacía.
                    echo json_encode(array('noHayDatos' => true));
                }
            } else {
                // Si existe sanción.
                echo json_encode(array('existe_sancion' => true));
            }
        } else {
            // Si el partido no esta en estado jugado.
            echo json_encode(array('partido_no_jugado' => true));
        }
    } else {
        // Si el partido no existe.
        echo json_encode(array('no_existe_partido' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

ANEXO VIII

Código “GP001-7 Registrar tribunal de sanciones”

```php
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la inserción del
tribunal de sanciones en partidos.
 */

/**
 * @mainpage Documentación de Inserción del tribunal de Sanciones
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la inserción del tribunal de
sanciones en partidos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_sanciones.php');
include('clases/clase_partido.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Maneja la inserción del tribunal de sanciones en partidos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_partido El ID del partido al que se asocia la sanción.
    * @param string $tribunal_local El informe del tribunal del equipo local
sobre la sanción.
```

```
    * @param string $tribunal_visitante El informe del tribunal del equipo
visitante sobre la sanción.
    * @param string $responsables Los nombres de los responsables de aceptar
el tribunal, debe ser separado
    * por comas ejemplo (Responsable 1,Responsable 2)
    */
    $id_partido = isset($_POST['id_partido']) ?
intval(trim($_POST['id_partido'])) : 0;
    $tribunal_local = isset($_POST['informe_local']) ?
trim($_POST['informe_local']) : "";
    $tribunal_visitante = isset($_POST['informe_visitante']) ?
trim($_POST['informe_visitante']) : "";
    $responsables = isset($_POST['responsables']) ?
trim($_POST['responsables']) : "";

    // Crea instancias de las clases a ser usadas.
    $objeto_sanciones = new sanciones();
    $objeto_partido = new partido();

    // Verifica si existe ese ID de partido.
    $verificar = $objeto_partido->verificar_ID($id_partido, $conexion);
    if (!empty($verificar)){

        // Obtener el estado del partido.
        $estado=$verificar['estado'];

        // Verificar el esatdo del partido.
        if($estado==='jugado'){

            // Consulta para verificar si existe una sancion registrada.
            $verificar=$objeto_sanciones->verificar_sancion($id_partido,
$conexion);
            if($verificar>0){

                // Verificar que los datos no esten vacios.
                if(!empty($tribunal_local) & !empty($tribunal_visitante) &
!empty($responsables)){

                    // Inseratar sanciones en la base de datos.
                    $respuesta=$objeto_sanciones-
>insertar_tribunal($id_partido, $tribunal_local, $tribunal_visitante,
$responsables, $conexion);

                    // Verificar que se inserto la sanción.
                    if ($respuesta) {
                        echo json_encode(array('success' => true));
                    } else {
                        echo json_encode(array('noInserto' => true));
```

```
                    }
                } else {
                    // Al menos una de las variables está vacía.
                    echo json_encode(array('noHayDatos' => true));
                }
            } else {
                // Si la sancion no existe.
                echo json_encode(array('no_existe_sancion' => true));
            }
        } else {
            // Si el partido no fue jugado.
            echo json_encode(array('partido_no_jugado' => true));
        }
    } else {
        // Si el partido no existe.
        echo json_encode(array('no_existe_partido' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

Código “GP001-8 Mostrar acta de juego”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la obtención de detalles
de un partido.
 */

/**
 * @mainpage Documentación de la obtención de detalles de un partido
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la obtención de detalles de un
partido, incluyendo estadísticas de jugadores y sanciones.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_partido.php');
include('clases/clase_torneo.php');
```

```php
include('clases/clase_jugador.php');
include('clases/clase_sanciones.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Maneja la obtención de detalles de un partido.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_partido El ID del partido que se quiere ver la
informacion.
    * @param int $id_torneo El ID del torneo.
    */
    $id_torneo = isset($_POST['id_torneo']) ?
intval(trim($_POST['id_torneo'])) : 0;
    $id_partido = isset($_POST['id_partido']) ?
intval(trim($_POST['id_partido'])) : 0;

    // Crea instancias de las clases a ser usadas.
    $objeto_torneo = new torneo();
    $objeto_partido = new partido();
    $objeto_jugador = new jugador();
    $objeto_sancion = new sanciones();

    // Verifica si existe ese torneo.
    $verificar = $objeto_torneo->verificar_ID($id_torneo, $conexion);
    if(!empty($verificar)){
        // Verificar si existe ese partido.
        $verificar = $objeto_partido->verificar_ID($id_partido, $conexion);
        if(!empty($verificar)){
            $estado_BD=$verificar['estado'];
            // Verificar si el estado del partido es jugado.
            switch ($estado_BD) {
                case "jugado":
```

```
                    // Consultas para obtener el partido, las estadisticas
de jugadores y las sanciones de ese partido.
                    $partido=$objeto_partido-
>mostrar_partido($id_partido,$conexion);
                    $estadisticas=$objeto_jugador-
>mostrar_estadisticas_jugador_partido($id_partido, $id_torneo, $conexion);
                    $sanciones=$objeto_sancion-
>mostrar_sancion($id_partido,$conexion);

                    // Devuelve la informacion detallada en formato JSON, si
no hay datos retorna array vacios.
                    echo json_encode(array('datos_partido' =>
$partido,'estadisticas' => $estadisticas,'sanciones'=>$sanciones));
                    break;
                case "porjugar":
                    echo json_encode(array('partido_por_jugar' => true));
                    break;
                case "programar":
                    echo json_encode(array('partido_no_programado' =>
true));
                    break;
                case "posponer":
                    echo json_encode(array('partido_pospuesto' => true));
                    break;
                default:
                    echo json_encode(array('error' => true));
            }
        } else {
            // Si el partido no existe.
            echo json_encode(array('no_existe_partido' => true));
        }
    } else {
        // Si el torneo no existe.
        echo json_encode(array('no_existe_torneo' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “GU001-3 Recuperar contraseña”

```
<?php
/**
 * @file
```

```php
 * Este archivo contiene un script PHP para recuperar la contraseña de usuarios en una base de datos.
 */

/**
 * @mainpage Documentación de Recuperación de Contraseña de Usuario
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la recuperación de la contraseña de un usuario en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_usuario.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Recuperar la contraseña de un usuario registrado en la base de datos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param string $correo_usuario El correo del usuario.
    */
    $correo_usuario = isset($_POST['correo']) ? trim($_POST['correo']) : "";

    $correo_usuario = "carlospnppm@gmail.com";

    // Crea una instancia de la clase usuario.
    $objeto_usuario = new usuario();

    // Verifica que los datos no estén vacíos.
    if (!empty($correo_usuario)) {
```

```
        // Consulta para verificar que ese usuario existe.
        $existe_usuario = $objeto_usuario-
>verificar_usuario($correo_usuario, $conexion);
        if (!empty($existe_usuario)) {

            // Obtener el nombre del usuario
            $nombreUsuario = $existe_usuario['nombre'];

            // Generar un código de verificación aleatorio
            $codigoVerificacion = generarCodigoVerificacion();


            // Envío de correo electrónico con el código de verificación
            enviarCorreoRecuperacionContraseña($correo_usuario,
$nombreUsuario, $codigoVerificacion);

            // Crear un arreglo con los datos a devolver en la respuesta
JSON
            $response = array(
                'success' => true,
                'message' => 'Se ha enviado un correo con instrucciones para
recuperar tu contraseña.',
                'correoUsuario' => $correo_usuario,
                'codigoVerificacion' => $codigoVerificacion
            );
        } else {
             // El usuario no existe y no se puede modificar.
            echo json_encode(array('success' => false));
        }
    } else {
        // Al menos una de las variables está vacía.
        echo json_encode(array('noHayDatos' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

// Función para generar un código de verificación
function generarCodigoVerificacion() {
    // Ejemplo básico: Generar una cadena de caracteres aleatoria de
longitud 6
    $caracteres = '0123456789';
    $codigoVerificacion = '';
    for ($i = 0; $i < 6; $i++) {
```

```
        $codigoVerificacion .= $caracteres[rand(0, strlen($caracteres) -
1)];
    }

    return $codigoVerificacion;
}

// Función para enviar el correo de recuperación de contraseña
function enviarCorreoRecuperacionContraseña($correoDestino, $nombreUsuario,
$codigoVerificacion) {
    $asunto = 'Recuperación de contraseña';
    $mensaje = "Estimado/a $nombreUsuario,\r\n\r\n";
    $mensaje .= "Hemos recibido una solicitud para restablecer la contraseña
de tu cuenta. Por favor, utiliza el siguiente código de verificación:
$codigoVerificacion\r\n\r\n";
    $mensaje .= "Si no has solicitado esta acción, te recomendamos que tomes
las medidas necesarias para proteger tu cuenta, como cambiar tu contraseña y
habilitar la autenticación de dos factores.\r\n\r\n";
    $mensaje .= "Si tienes alguna pregunta o necesitas asistencia adicional,
no dudes en contactarnos. Estamos aquí para ayudarte.\r\n\r\n";
    $mensaje .= "Atentamente:\r\n";
    $mensaje .= "SGLigas\r\n";
    $cabeceras = 'From: tu_correo@example.com' . "\r\n" .
                 'Reply-To: tu_correo@example.com' . "\r\n" .
                 'X-Mailer: PHP/' . phpversion();


    mail($correoDestino, $asunto, $mensaje, $cabeceras);
}
?>
```

Código “GU001-4 Actualizar usuario”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para actualizar los datos de usuarios
en una base de datos.
 */

/**
 * @mainpage Documentación de Actualización de Datos de Usuario
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la actualización de los datos de
un usuario en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */
```

```php
 // Incluye las clases y archivos necesarios.
include('clases/clase_usuario.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Actualiza los datos de un usuario registrado en la base de datos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param string $correo_usuario El correo del usuario.
    * @param string $nombre El nuevo nombre del usuario.
    * @param string $tipo El tipo de usuario (por ejemplo, 'presidente' o
'hincha').
    */
    $correo_usuario = isset($_POST['correo']) ? trim($_POST['correo']) : "";
    $nombre = isset($_POST['nombre']) ? trim($_POST['nombre']) : "";
    $tipo = isset($_POST['tipo']) ? trim($_POST['tipo']) : "";

    // Crea una instancia de la clase usuario.
    $objeto_usuario = new usuario();

    // Verifica que los datos no estén vacíos.
    if (!empty($correo_usuario) && !empty($nombre) && !empty($tipo)) {

        // Consulta para verificar que ese usuario existe.
        $existe_usuario = $objeto_usuario-
>verificar_usuario($correo_usuario, $conexion);
        if (!empty($existe_usuario)) {

            // Consulta para actualizar los datos del usuario.
            $resultado = $objeto_usuario->actualizar_usuario($nombre, $tipo,
"", $correo_usuario, $conexion);

            // Verifica que se actualizo los datos del usuario.
```

```
            if ($resultado) {
                echo json_encode(array('success' => true));
            } else {
                echo json_encode(array('noInserto' => true));
            }
        } else {
            // El usuario no existe y no se puede modificar.
            echo json_encode(array('success' => false));
        }
    } else {
        // Al menos una de las variables está vacía.
        echo json_encode(array('noHayDatos' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para actualizar la contraseña de
usuarios en una base de datos.
 */

/**
 * @mainpage Documentación de Actualización de Contraseña de Usuario
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la actualización de la contraseña
de un usuario en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_usuario.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
```

```php
    return conexion_DB();
}

/**
 * Actualizar la contraseña de un usuario registrado en la base de datos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param string $correo_usuario El correo del usuario.
    * @param string $contrasena_nueva La contraseña segura del usuario.
    */
    $correo_usuario = isset($_POST['correo']) ? trim($_POST['correo']) : "";
    $contrasena_nueva = isset($_POST['password']) ? trim($_POST['password'])
: "";

    // Crea una instancia de la clase usuario.
    $objeto_usuario = new usuario();

    // Verifica que los datos no estén vacíos.
    if (!empty($correo_usuario) && !empty($contrasena_nueva)) {

        // Consulta para verificar que ese usuario existe.
        $existe_usuario = $objeto_usuario-
>verificar_usuario($correo_usuario, $conexion);
        if (!empty($existe_usuario)) {

            // Encripta el password
            $pass_fuerte = password_hash($contrasena_nueva,
PASSWORD_DEFAULT);

            // Consulta para actualizar la contraseña del usuario.
            $resultado = $objeto_usuario->actualizar_usuario("", "",
$pass_fuerte, $correo_usuario, $conexion);

            // Verifica que se actualizo la contraseña.
            if ($resultado) {
                echo json_encode(array('success' => true));
            } else {
                echo json_encode(array('noInserto' => true));
            }
        } else {
             // El usuario no existe y no se puede modificar.
            echo json_encode(array('success' => false));
```

```
        }
    } else {
        // Al menos una de las variables está vacía.
        echo json_encode(array('noHayDatos' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “GL001-5 Actualizar categoría”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la actualización de una
categoría en una base de datos.
 */

/**
 * @mainpage Documentación de Actualización de Categoría en una Base de
Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la actualización de una categoría
en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_categoria.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Actualiza una categoría en la base de datos.
 */
```

```php
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param string $nombre El nombre de la categoría a actualizar.
    * @param int $num_equipos El número de equipos en la categoría.
    * @param int $id_categoria El ID de la categoría a actualizar.
    */
    $nombre = isset($_POST['categoria']) ? trim($_POST['categoria']) : "";
    $num_equipos = isset($_POST['num_equipos']) ?
trim($_POST['num_equipos']) : "";
    $id_categoria = isset($_POST['id_categoria']) ?
intval(trim($_POST['id_categoria'])) : 0;

    // Crea una instancia de la clase categoria.
    $objeto_categoria = new categoria();

    // Verifica si existe ese ID de categoría.
    $id_categoria = $objeto_categoria->verificar_ID($id_categoria,
$conexion);
    if ($id_categoria > 0) {

        // Verifica que los datos no estén vacíos.
        if (!empty($nombre) && !empty($num_equipos)) {

            // Actualizar la nueva categoría en la base de datos.
            $resultado = $objeto_categoria->actualizar_categoria($nombre,
$num_equipos, $id_categoria, $conexion);

            // Verifica que se actualizo la categoría.
            if ($resultado) {
                echo json_encode(array('success' => true));
            } else {
                echo json_encode(array('noInserto' => true));
            }
        } else {
            // Al menos una de las variables está vacía.
            echo json_encode(array('noHayDatos' => true));
        }
    } else {
        // Si la categoría no existe.
        echo json_encode(array('no_existe_categoria' => true));
    }

    // Cierra la conexión a la base de datos.
```

```
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “GE001-6 Actualizar equipo”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la actualización de un
equipo en una base de datos.
 */

/**
 * @mainpage Documentación de Actualización de Equipo en una Base de Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la actualización de un equipo en
una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_equipo.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Actualiza un equipo en la base de datos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param string $nombre_equipo El nombre del equipo a actualizar.
```

```
    * @param string $presidente El nombre del presidente del equipo.
    * @param string $escudo La URL del escudo del equipo.
    * @param int $id_equipo El ID del equipo que se desea mostrar.
    */
    $nombre_equipo=isset($_POST['nombre_equipo']) ?
trim($_POST['nombre_equipo']) : "";
    $presidente=isset($_POST['presidente']) ? trim($_POST['presidente']) :
"";
    $escudo=isset($_POST['escudo']) ? trim($_POST['escudo']) : "";
    $id_equipo = isset($_POST['id_equipo']) ?
intval(trim($_POST['id_equipo'])) : 0;

    // Crea una instancia de la clase equipo.
    $objeto_equipo = new equipo();

    // Verifica si existe ese ID de equipo.
    $id_equipo = $objeto_equipo->verificar_ID($id_equipo, $conexion);
    if ($id_equipo > 0) {

        // Verifica que los datos no estén vacíos.
        if (!empty($nombre_equipo) && !empty($presidente) &&
!empty($escudo)) {

            // Actualizar el equipo en la base de datos.
            $resultado = $objeto_equipo->actualizar_equipo($nombre_equipo,
$presidente, $escudo, $id_equipo, $conexion);

            // Verifica que se actualizo los datos del equipo.
            if ($resultado) {
                echo json_encode(array('success' => true));
            } else {
                echo json_encode(array('noInserto' => true));
            }
        } else {
            // Al menos una de las variables está vacía.
            echo json_encode(array('noHayDatos' => true));
        }
    } else {
        // Si el equipo no existe.
        echo json_encode(array('no_existe_equipo' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
```

```
?>
```

Código “GE001-7 Actualizar jugador”

```php
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la actualizacion de
jugadores en una base de datos.
 */

/**
 * @mainpage Documentación de Manejar actualizacion de Jugadores en una Base
de Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la actualizacion de jugadores en
una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_equipo.php');
include('clases/clase_jugador.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Ejecuta la función principal para actualizar un nuevo jugador.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $CI El número de cédula del jugador.
    * @param string $posicion La posición del jugador.
    * @param string $foto La URL de la foto del jugador.
    * @param int $estatura La estatura del jugador en cm ejemplo (160).
```

```
    * @param int $num_camiseta El numero de camiseta del jugador.
    * @param int $id_equipo El ID del equipo al que pertenece el jugador.
    */
    $CI = isset($_POST['CI']) ? intval(trim($_POST['CI'])) : 0;
    $posicion = isset($_POST['posicion']) ? trim($_POST['posicion']) : "";
    $foto = isset($_POST['foto']) ? trim($_POST['foto']) : "";
    $estatura = isset($_POST['estatura']) ? intval(trim($_POST['estatura']))
: 0;
    $num_camiseta = isset($_POST['num_camiseta']) ?
intval(trim($_POST['num_camiseta'])) : 0;
    $id_equipo = isset($_POST['id_equipo']) ?
intval(trim($_POST['id_equipo'])) : 0;

    // Crea instancias de las clases a ser usadas.
    $objeto_equipo = new equipo();
    $objeto_jugador = new jugador();

    // Verifica si existe ese ID de equipo.
    $id_equipo = $objeto_equipo->verificar_ID($id_equipo, $conexion);
    if ($id_equipo > 0) {

        // Verificar si la CI de ese jugador no existe.
        $verificar=$objeto_jugador->verificar_CI($CI,$conexion);
        if(!empty($verificar)){

            // Verifica que los datos no estén vacíos.
            if (!empty($posicion) && !empty($foto) && !empty($estatura) &&
!empty($num_camiseta)) {

                // Consulta para verificar el numero de camiseta.
                $verificar=$objeto_jugador->verificar_camiseta($id_equipo,
$num_camiseta, $conexion);
                if($verificar == 0){

                    // Actualizar el nuevo jugador en la base de datos.
                    $resultado = $objeto_jugador->actualizar_jugador($CI,
$posicion, $foto, $estatura, $num_camiseta, $id_equipo, $conexion);

                    // Verificar que se inserto el jugador.
                    if ($resultado) {
                        echo json_encode(array('success' => true));
                    } else {
                        echo json_encode(array('noInserto' => true));
                    }
                } else {
                    // Existe el numero de camiseta.
                    echo json_encode(array('exite_numero_camiseta' =>
true));
```

```
                }
            } else {
                // Al menos una de las variables está vacía.
                echo json_encode(array('noHayDatos' => true));
            }
        } else {
            // Si existe el jugador.
            echo json_encode(array('no_existe_jugador' => true));
        }
    } else {
        // Si el equipo no existe.
        echo json_encode(array('no_existe_equipo' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();

?>
```

Código “GU001-5 Eliminar usuario”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para eliminar un usuario de tipo
hincha en una base de datos.
 */

/**
 * @mainpage Documentación de Eliminación de Usuarios Tipo Hincha
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la eliminación de un usuario de
tipo hincha en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_usuario.php');
include('conexion.php');



/**
```

```php
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Elimina un usuario de tipo hincha registrado en la base de datos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param string $correo_usuario El correo del usuario.
    */
    $correo_usuario = isset($_POST['correo']) ? trim($_POST['correo']) : "";

    // Crea una instancia de la clase usuario.
    $objeto_usuario = new usuario();

    // Verifica que los datos no estén vacíos.
    if (!empty($correo_usuario)) {

        // Consulta para verificar que ese usuario existe.
        $existe_usuario = $objeto_usuario-
>verificar_usuario($correo_usuario, $conexion);
        if (!empty($existe_usuario)) {

            //Verifica el tipo de usuario que es.
            if($existe_usuario['tipo_usuario'] != 'presidente'){

                // Consulta para eliminar usuario.
                $resultado = $objeto_usuario-
>eliminar_usuario_hincha($correo_usuario, $conexion);

                // Verifica que se elimino el usuario.
                if ($resultado) {
                    echo json_encode(array('success' => true));
                } else {
                    echo json_encode(array('noInserto' => true));
                }
            } else {
                echo json_encode(array('es_presidente' => true));
            }
```

```
        } else {
            // El usuario no existe y no se puede modificar.
            echo json_encode(array('success' => false));
        }
    } else {
        // Al menos una de las variables está vacía.
        echo json_encode(array('noHayDatos' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “GL001-6 Eliminar liga”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para eliminar una liga en una base de
datos.
 */

/**
 * @mainpage Documentación de Desactivación de Liga
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la eliminación de una liga en una
base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_liga.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
```

```
 * Elimina una liga en la base de datos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_liga El ID de la liga que se desea eliminar.
    */
    $id_liga = isset($_POST['id_liga']) ? intval(trim($_POST['id_liga'])) :
0;

    // Crea una instancia de la clase liga.
    $objeto_liga = new liga();

    // Verifica si existe ese ID  de liga.
    $id_liga = $objeto_liga->verificar_ID($id_liga, $conexion);
    if ($id_liga > 0) {

        // Consulta para eliminar la liga.
        $resultado = $objeto_liga->eliminar_liga($id_liga, $conexion);

        // Verifica si se eliminó la liga.
        if ($resultado) {
            echo json_encode(array('success' => true));
        } else {
            echo json_encode(array('noInserto' => true));
        }
    } else {
        // Al menos una de las variables está vacía.
        echo json_encode(array('no_existe_liga' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “GL001-7 Eliminar categoría”

```
<?php
/**
 * @file
```

```php
 * Este archivo contiene un script PHP para manejar la eliminación de una categoría en una base de datos.
 */

/**
 * @mainpage Documentación de Inserción de Categorías en una Base de Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la eliminación de categorías en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_categoria.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Elimina una categoría en la base de datos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_categoria El ID de la categoría a eliminar.
    */
    $id_categoria = isset($_POST['id_categoria']) ? intval(trim($_POST['id_categoria'])) : 0;

    // Crea una instancia de la clase categoria.
    $objeto_categoria = new categoria();

    // Verifica si existe ese ID de liga.
    $id_categoria = $objeto_categoria->verificar_ID($id_categoria, $conexion);
    if ($id_categoria>0) {
```

```
        // Consulta para eliminar categoría.
        $resultado = $objeto_categoria->eliminar_categoria($id_categoria,
$conexion);

        // Verifica si se desactivó la categoría.
        if ($resultado) {
            echo json_encode(array('success' => true));
        } else {
            echo json_encode(array('noInserto' => true));
        }
    } else {
        // Si el id de categoría no existe
        echo json_encode(array('no_existe_categoria' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

Código “GE001-8 Eliminar equipo”

```
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la eliminación de un
equipo en una base de datos.
 */

/**
 * @mainpage Documentación de Eliminacion de un equipo en una Base de Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la eliminación de un equipo en
una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de
datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_equipo.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
```

```php
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Elimina un equipo en la base de datos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $id_equipo El ID del equipo a eliminar.
    */
    $id_equipo = isset($_POST['id_equipo']) ?
intval(trim($_POST['id_equipo'])) : 0;

    // Crea una instancia de la clase categoria.
    $objeto_equipo = new equipo();

    // Verifica si existe ese ID de liga.
    $id_categoria = $objeto_equipo->verificar_ID($id_equipo, $conexion);
    if ($id_equipo>0) {

        // Consulta para eliminar categoría.
        $resultado = $objeto_equipo->eliminar_equipo($id_equipo, $conexion);

        // Verifica si se desactivó la categoría.
        if ($resultado) {
            echo json_encode(array('success' => true));
        } else {
            echo json_encode(array('noInserto' => true));
        }
    } else {
        // Si el id de categoría no existe
        echo json_encode(array('no_existe_equipo' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
```

```
?>
```

Código “GE001-9 Eliminar jugador”

```php
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la eliminación de un jugador en una base de datos.
 */

/**
 * @mainpage Documentación de Eliminacion de jugadores en una Base de Datos
 *
 * @section intro_sec Introducción
 * Este script PHP se utiliza para manejar la eliminación de jugadores en una base de datos.
 * Utiliza clases y archivos necesarios para conectar y manipular la base de datos.
 */

// Incluye las clases y archivos necesarios.
include('clases/clase_jugador.php');
include('conexion.php');

/**
 * Obtiene la conexión a la base de datos.
 * @return mysqli La conexión a la base de datos.
 */
function obtener_conexion() {
    return conexion_DB();
}

/**
 * Elimina un jugador en la base de datos.
 */
function main() {
    // Obtiene la conexión a la base de datos.
    $conexion = obtener_conexion();

    /**
    * Obtiene los datos ingresados por parte del cliente.
    *
    * @param int $CI La CI del jugador a eliminar.
    */
    $CI = isset($_POST['CI']) ? intval(trim($_POST['CI'])) : 0;

    // Crea una instancia de la clase categoria.
    $objeto_jugador = new jugador();
```

```php
    // Verifica si existe ese ID de liga.
    $CI = $objeto_jugador->verificar_CI($CI, $conexion);
    if ($CI>0) {

        // Consulta para eliminar categoría.
        $resultado = $objeto_jugador->eliminar_jugador($CI, $conexion);

        // Verifica si se desactivó el jugador.
        if ($resultado) {
            echo json_encode(array('success' => true));
        } else {
            echo json_encode(array('noInserto' => true));
        }
    } else {
        // Si el id de categoría no existe
        echo json_encode(array('no_existe_jugador' => true));
    }

    // Cierra la conexión a la base de datos.
    mysqli_close($conexion);
}

// Ejecuta la función principal.
main();
?>
```

ANEXO IX

https://github.com/CarlosVelasquezZ/SGLigas_backend/tree/main/Documentacion

ANEXO X

Sprint 1			
Código	**Historia de Usuario**	**Criterios de aceptación**	**Cumplimiento**
GU001-1	Registrar usuario	Envió de datos correctos	✓
		Envió de datos vacios	✓
		Envió de correo registrado	✓
		Error en inserción	✓
GU001-2	Iniciar de sesión	Envió de datos correctos	✓
		Envió de datos vacios	✓
		Envió de correo electrónico no registrado	✓
		Contraseña incorrecta	✓
GL001-1	Registrar liga	Envió de datos correctos	✓
		Envió de datos vacios	✓
		Envió de correo electronico no registrado	✓
		Envió de correo electronico que no es presidente	✓
		Envió de correo electronico que tiene liga registrada	✓
		Envió de nombre de liga existente	✓
		Error en inserción	✓
GL001-2	Registrar categoría	Envió de datos correctos	✓
		Envió de un id de liga incorrecto	✓
		Envió de datos vacios	✓
		Envió de nombre de categoría existente en la liga	✓
		Error en inserción	✓
GL001-3	Mostrar liga	Envió de datos correctos	✓
		Envió de correo electrónico no registrado	✓
		Envió de correo electrónico que no es presidente	✓
		Error en la consulta	✓
GL001-4	Mostrar categorías	Envió de datos correctos	✓
		Envió de un id de liga incorrecto	✓
		Error en la consulta	✓
GE001-2	Validar numero de equipos	Envió de datos correctos y limite alcanzado	✓
		Envió de datos correctos y limite no alcanzado	✓

		Envió de un id de categoría incorrecto	✓
		Error en la consulta	✓
GE001-3	Registrar equipo	Envió de datos correctos	✓
		Envió de un id de liga incorrecto	✓
		Envió de un id de categoría incorrecto	✓
		Envió de datos vacíos	✓
		Envió de nombre de equipo existente en la liga	✓
		Error en inserción	✓
GE001-1	Mostrar equipos de liga	Envió de datos correctos	✓
		Envió de un id de liga incorrecto	✓
		Error en la consulta	✓
PI001-5	Mostrar equipo	Envió de datos correctos	✓
		Envió de un id de liga incorrecto	✓
		Error en la consulta	✓
Sprint 2			
Código	**Historia de Usuario**	**Criterios de aceptación**	**Cumplimiento**
GE001-4	Registrar jugador	Envió de datos correctos	✓
		Envió de un id de equipo incorrecto	✓
		Envió de cédula existente	✓
		Envió de datos vacíos	✓
		Envió de número de camiseta existen en equipo	✓
		Error en inserción	✓
GT001-1	Registrar torneos	Envió de datos correctos sin grupos	✓
		Envió de datos correctos con grupos	✓
		Envió de un id de categoría incorrecto	✓
		Envió de datos vacios	✓
		Error en inserción	✓
GT001-2	Generar partidos	Envió de datos correctos	✓
		Envió de id de torneo incorrecto	✓
		Envió de id de torneo que tiene partidos registrados	✓
		Envió de datos vacios	✓
		Envió de id de equipo incorrecto	✓
		Error en la consulta	✓

		Error en inserción	✓
GT001-3	Mostrar torneos	Envió de datos correctos	✓
		Envió de un id de categoría incorrecto	✓
		Error en la consulta	✓
GP001-1	Programar partidos	Envió de datos correctos	✓
		Envió de un id de partido incorrecto	✓
		Envió de un id de partido en un estado que no es programar o pospuesto	✓
		Envió de datos vacios	✓
		Error en la actualización de estadísticas partido	✓
PI001-1	Mostrar todas las ligas inscritas	Consulta correcta	✓
		Error en la consulta	✓
PI001-2	Mostrar equipos de categoría	Envió de datos correctos	✓
		Envió de un id de categoría incorrecto	✓
		Error en la consulta	✓
GE001-5	Mostrar jugadores	Envió de datos correctos	✓
		Envió de un id de equipo incorrecto	✓
		Error en la consulta	✓
PI001-3	Mostrar posiciones	Envió de datos correctos	✓
		Envió de un id de torneo incorrecto	✓
		Error en la consulta	✓
Sprint 3			
Código	**Historia de Usuario**	**Criterios de aceptación**	**Cumplimiento**
GP001-2	Registrar resultados	Envió de datos correctos	✓
		Envió de un id de torneo incorrecto	✓
		Envió de un id de torneo sin partidos	✓
		Envió de un id de partido incorrecto	✓
		Envió de un id de partido en un estado que no es por jugar	✓
		Envió de datos vacíos	✓
		Error en la actualización de partido	✓
		Error en la actualización de estadística equipo	✓
GP001-3		Envió de datos correctos	✓
		Envió de id de torneo incorrecto	✓

	Comprobar registro de estadisticas de jugadores	Envió de id de partido incorrecto	✓
		Envió de id de partido en un estado que no es jugado	✓
		Error en la consulta	✓
GP001-4	Registrar estadísticas jugadores: rojas, amarillas, goles, goles recibidos en el caso de porteros, autogoles y partidos jugados.	Envió de datos correctos	✓
		Envió de un id de torneo incorrecto	✓
		Envió de un id de torneo sin partidos	✓
		Envió de un id de partido incorrecto	✓
		Envió de un id de partido en un estado que no es jugado	✓
		Envió de datos vacíos	✓
		Error en la actualización de estadísticas de jugador	✓
PI001-4	Mostrar partidos	Envió de datos correctos	✓
		Envió de id de torneo incorrecto	✓
		Error en la consulta	✓
PI001-6	Mostrar partidos de cada equipo	Envió de datos correctos	✓
		Envió de id de torneo incorrecto	✓
		Envió de un id de equipo incorrecto	✓
		Error en la consulta	✓
PI001-7	Mostrar estadisticas de jugadores de equipo	Envió de datos correctos	✓
		Envió de un id de equipo incorrecto	✓
		Error en la consulta	✓
PI001-8	Mostrar informacion de jugador	Envió de datos correctos	✓
		Envió de una cédula incorrecta	✓
		Error en la consulta	✓
GP001-5	Registrar alineación	Envió de datos correctos	✓
		Envió de un id de partido incorrecto	✓
		Envió de un id de partido en un estado que no es por jugar	✓
		Envió de datos vacíos	✓
		Envió de un id de partido que ya tiene registrado alineación	✓
		Error en inserción	✓
PI001-9	Mostrar alineación	Envió de datos correctos	✓
		Envió de un id de partido incorrecto	✓
		Envió de un id de partido en un estado que no es jugado	✓

		Error en la consulta	✓
GP001-6	Registrar informe de sanciones	Envió de datos correctos	✓
		Envió de un id de partido incorrecto	✓
		Envió de un id de partido en un estado que no es jugado	✓
		Envió de un id de partido que ya tiene registrado una sanción	✓
		Envió de datos vacíos	✓
		Error en inserción	✓
Sprint 4			
Código	**Historia de Usuario**	**Criterios de aceptación**	**Cumplimiento**
GP001-7	Registrar tribunal de sanciones	Envió de datos correctos	✓
		Envió de un id de partido incorrecto	✓
		Envió de un id de partido en un estado que no es jugado	✓
		Envió de un id de partido que no tiene registrado una sanción	✓
		Envió de datos vacíos	✓
		Error en inserción	✓
GP001-8	Mostrar acta de juego	Envió de datos correctos	✓
		Envió de un id de torneo incorrecto	✓
		Envió de un id de partido incorrecto	✓
		Envió de un id de partido en un estado que no es jugado	✓
		Error en la consulta	✓
GU001-3	Recuperar contraseña	Envió de datos correctos	✓
		Envió de correo electrónico no registrado	✓
GU001-4	Actualizar usuario	Envió de datos correctos	✓
		Envió de datos vacíos	✓
		Envió de correo electrónico no registrado	✓
GL001-5	Actualizar categoría	Envió de datos correctos	✓
		Envió de un id de categoría incorrecto	✓
		Envió de datos vacios	✓
		Error en actualización	✓
GE001-6	Actualizar equipo	Envió de datos correctos	✓
		Envió de id equipo incorrecto	✓
		Envió de datos vacios	✓
		Error en actualización	✓
GE001-7	Actualizar jugador	Envió de datos correctos	✓

		Envió de un id de equipo incorrecto	✓
		Envió de cédula existente	✓
		Envió de datos vacíos	✓
GU001-5	Eliminar usuario	Envió de datos correctos	✓
		Envió de datos vacios	✓
		Envió de correo electronico no registrado	✓
		Envió de correo electronico que es presidente	✓
GL001-6	Eliminar liga	Envió de datos correctos	✓
		Envió de un id de liga incorrecto	✓
GL001-7	Eliminar categoría	Envió de datos correctos	✓
		Envió de un id de categoría incorrecto	✓
GE001-8	Eliminar equipo	Envió de datos correctos	✓
		Envió de un id de equipo incorrecto	✓
GE001-9	Eliminar jugador	Envió de datos correctos	✓
		Envió de un CI incorrecto	✓

ANEXO XI

Sprint 1
Prueba “GU001-1 Registrar Usuario”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_1_insertar_usuario extends TestCase {

    // Prueba de envio de datos correctos
    public function test_insercion_correcta() {
        // Datos del nuevo usuario
        $datosUsuario = [
            'correo' => 'nuevo_usuario@example.com',
            'nombre' => 'Nuevo Usuario',
            'password' => 'contraseña_segura',
            'tipo' => 'hincha'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitar_post('http://localhost:8080/SGLIGAS/Modelo/1_insertar_usuario.php
', $datosUsuario);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    // Prueba de envio de datos vacios
    public function test_datos_vacios() {
        // Datos vacíos
        $datosUsuario = [
            'correo' => '',
            'nombre' => '',
            'password' => '',
            'tipo' => ''
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitar_post('http://localhost:8080/SGLIGAS/Modelo/1_insertar_usuario.php
', $datosUsuario);
```

```php
        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    // Prueba de envio de correo registrado
    public function test_correo_registrado() {
        // Datos de usuario existente
        $datosUsuario = [
            'correo' => 'nuevo_usuario@example.com',
            'nombre' => 'Usuario Registrado',
            'password' => 'contraseña_segura',
            'tipo' => 'hincha'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitar_post('http://localhost:8080/SGLIGAS/Modelo/1_insertar_usuario.php', $datosUsuario);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertFalse($cuerpoRespuesta['success']);
    }

    private function solicitar_post($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_1_insertar_usuario.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

...                                                                 3 / 3 (100%)

Time: 00:02.975, Memory: 8.00 MB

OK (3 tests, 9 assertions)
```

Prueba “GU001-2 Iniciar de sesión”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_1_iniciar_sesion extends TestCase {

    // Prueba de envio de datos correctos
    public function testEnvioDatosCorrectos() {
        // Datos del usuario
        $datosUsuario = [
            'correo' => 'nuevo_usuario@example.com',
            'password' => 'contraseña_segura'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/1_iniciar_sesion.php',
$datosUsuario);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('token', $cuerpoRespuesta);
        $this->assertArrayHasKey('userData', $cuerpoRespuesta);
        $this->assertEquals('nuevo_usuario@example.com',
$cuerpoRespuesta['userData']['correo']);
        $this->assertEquals('hincha',
$cuerpoRespuesta['userData']['tipo_usuario']);
    }

    public function testEnvioDatosVacios() {
        // Datos vacíos
        $datosUsuario = [
            'correo' => '',
            'password' => ''
        ];

        // Realizar la solicitud POST al script PHP
```

```
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/1_iniciar_sesion.php', $datosUsuario);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    public function testCorreoNoRegistrado() {
        // Datos de correo no registrado
        $datosUsuario = [
            'correo' => 'correo_no_registrado@example.com',
            'password' => 'contraseña'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/1_iniciar_sesion.php', $datosUsuario);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('correo', $cuerpoRespuesta);
        $this->assertFalse($cuerpoRespuesta['correo']);
    }

    public function testContrasenaIncorrecta() {
        // Datos de contraseña incorrecta
        $datosUsuario = [
            'correo' => 'nuevo_usuario@example.com',
            'password' => 'contraseña_incorrecta'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/1_iniciar_sesion.php', $datosUsuario);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());
```

```
        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertFalse($cuerpoRespuesta['success']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_1_iniciar_sesion.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

....                                                              4 / 4 (100%)

Time: 00:03.438, Memory: 8.00 MB

OK (4 tests, 14 assertions)
```

Prueba “GL001-1 Registrar liga”

```
<?php
use PHPUnit\Framework\TestCase;

class prueba_2_insertar_liga extends TestCase {

    public function testEnvioDatosCorrectos() {
        // Datos del usuario
        $datosLiga = [
            'nombre_liga' => 'Nueva Liga',
            'fecha_fundacion' => '2022-02-14',
            'direccion' => 'Dirección de la Liga',
            'correo_admin' => 'correo_admin@example.com'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/2_insertar_liga.php', $datosLiga);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
```

```
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioDatosVacios() {
        // Datos vacíos
        $datosLiga = [
            'nombre_liga' => '',
            'fecha_fundacion' => '',
            'direccion' => '',
            'correo_admin' => ''
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/2_insertar_liga.php',
$datosLiga);

            // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    public function testCorreoNoRegistrado() {
        // Datos de correo no registrado
        $datosLiga = [
            'nombre_liga' => 'Nueva Liga',
            'fecha_fundacion' => '2022-02-14',
            'direccion' => 'Dirección de la Liga',
            'correo_admin' => 'correo_no_registrado@example.com'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/2_insertar_liga.php',
$datosLiga);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        //$this->assertArrayHasKey('no_existe_usuario', $cuerpoRespuesta);
```

```
        $this->assertTrue($cuerpoRespuesta['no_existe_usuario']);
    }

    public function testCorreoNoPresidente() {
        // Datos de correo no presidente
        $datosLiga = [
            'nombre_liga' => 'Nueva Liga',
            'fecha_fundacion' => '2022-02-14',
            'direccion' => 'Dirección de la Liga',
            'correo_admin' => 'nuevo_usuario@example.com'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/2_insertar_liga.php',
$datosLiga);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_es_presidente', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_es_presidente']);
    }

    public function testCorreoLigaRegistrada() {
        // Datos de correo con liga registrada
        $datosLiga = [
            'nombre_liga' => 'Nueva Liga',
            'fecha_fundacion' => '2022-02-14',
            'direccion' => 'Dirección de la Liga',
            'correo_admin' => 'correo_admin@example.com'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/2_insertar_liga.php',
$datosLiga);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('existe_registro', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['existe_registro']);
    }
```

```php
    public function testNombreLigaExistente() {
        // Datos de nombre de liga existente
        $datosLiga = [
            'nombre_liga' => 'Nueva Liga',
            'fecha_fundacion' => '2022-02-14',
            'direccion' => 'Dirección de la Liga',
            'correo_admin' => 'correo_admin2@example.com'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/2_insertar_liga.php',
$datosLiga);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('existe_nombre', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['existe_nombre']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_2_insertar_liga.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

......                                                            6 / 6 (100%)

Time: 00:07.488, Memory: 8.00 MB

OK (6 tests, 17 assertions)
```

Prueba “GL001-2 Registrar categoría”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_3_insertar_categoria extends TestCase {

    public function testEnvioDatosCorrectos() {
        // Datos de la categoría
```

```
        $datosCategoria = [
            'categoria' => 'Maxima',
            'num_equipos' => 4,
            'id_liga' => 1
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/3_insertar_categoria.php', $datosCategoria);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testIdLigaIncorrecto() {
        // Datos de categoría con ID de liga incorrecto
        $datosCategoria = [
            'categoria' => 'Maxima',
            'num_equipos' => 4,
            'id_liga' => 100
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/3_insertar_categoria.php', $datosCategoria);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_liga', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_liga']);
    }

    public function testEnvioDatosVacios() {
        // Datos vacíos
        $datosCategoria = [
            'categoria' => '',
            'num_equipos' => '',
            'id_liga' => 1
```

```
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/3_insertar_categoria.php', $datosCategoria);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    public function testNombreCategoriaExistente() {
        // Datos de nombre de liga existente
        $datosCategoria = [
            'categoria' => 'Maxima',
            'num_equipos' => 4,
            'id_liga' => 1
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/3_insertar_categoria.php', $datosCategoria);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('existe_nombre', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['existe_nombre']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_3_insertar_categoria.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

....                                                                4 / 4 (100%)

Time: 00:04.517, Memory: 8.00 MB

OK (4 tests, 12 assertions)
```

Prueba “GL001-3 Mostrar liga”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_2_mostrar_liga_presidente extends TestCase {

    public function testEnvioDatosCorrectos() {
        // Datos del presidente
        $datosPresidente = [
            'correo_admin' => 'correo_admin@example.com'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/2_mostrar_liga_presidente.php', $datosPresidente);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('datos', $cuerpoRespuesta);
        $this->assertTrue(count($cuerpoRespuesta['datos']) > 0);
    }

    public function testCorreoNoRegistrado() {
        // Datos de correo no registrado
        $datosPresidente = [
            'correo_admin' => 'correo_no_registrado@example.com'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/2_mostrar_liga_presidente.php', $datosPresidente);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());
```

```php
        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_usuario', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_usuario']);
    }

    public function testCorreoNoPresidente() {
        // Datos de correo no presidente
        $datosPresidente = [
            'correo_admin' => 'nuevo_usuario@example.com'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/2_mostrar_liga_presidente.php', $datosPresidente);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_es_presidente', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_es_presidente']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}

?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_2_mostrar_liga_presidente.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

...                                                                 3 / 3 (100%)

Time: 00:03.262, Memory: 8.00 MB

OK (3 tests, 9 assertions)
```

Prueba “GL001-4 Mostrar categorías”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_3_mostrar_categorias extends TestCase {
```

```php
    public function testEnvioDatosCorrectos() {
        // Datos del cliente
        $datosCliente = [
            'id_liga' => 1
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/3_mostrar_categorias.php', $datosCliente);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('datos', $cuerpoRespuesta);
        $this->assertNotEmpty($cuerpoRespuesta['datos']);
    }

    public function testEnvioIdLigaIncorrecto() {
        // Datos del cliente
        $datosCliente = [
            'id_liga' => 0
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/3_mostrar_categorias.php', $datosCliente);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_liga', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_liga']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
```

```
?>
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_3_mostrar_categorias.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

..                                                                  2 / 2 (100%)

Time: 00:02.367, Memory: 8.00 MB

OK (2 tests, 6 assertions)
```

Prueba “GE001-2 Validar número de equipos”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_4_validar_numero_equipos extends TestCase {

    public function testEnvioDatosCorrectosYLimiteAlcanzado() {
        // Datos del equipo
        $datosEquipo = [
            'id_categoria' => 82 // ID de una categoría que alcanzo el
límite de equipos
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_validar_numero_equipo
s.php', $datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('limite_equipos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['limite_equipos']);
    }

    public function testEnvioDatosCorrectosYLimiteNoAlcanzado() {
        // Datos del equipo
        $datosEquipo = [
            'id_categoria' => 111 // ID de una categoría que no haya
alcanzado el límite de equipos
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_validar_numero_equipo
s.php', $datosEquipo);
```

```php
        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('limite_equipos', $cuerpoRespuesta);
        $this->assertFalse($cuerpoRespuesta['limite_equipos']);
    }

    public function testEnvioIdCategoriaIncorrecto() {
        // Datos del equipo
        $datosEquipo = [
            'id_categoria' => 0 // Asignar un ID de una categoría que no
exista
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_validar_numero_equipo
s.php', $datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_categoria', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_categoria']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_4_validar_numero_equipos.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

...                                                                 3 / 3 (100%)

Time: 00:04.062, Memory: 8.00 MB

OK (3 tests, 9 assertions)
```

Prueba “GE001-3 Registrar equipo”

```php
<?php
```

```php
use PHPUnit\Framework\TestCase;

class prueba_4_insertar_equipo extends TestCase {

    public function testEnvioDatosCorrectos() {
        // Datos del nuevo equipo
        $datosEquipo = [
            'nombre_equipo' => 'Nuevo Equipo',
            'fecha_fundacion' => '2022-01-01',
            'presidente' => 'Presidente Nuevo',
            'color' => 'Rojo',
            'escudo' => 'http://example.com/escudo',
            'id_categoria' => 111,
            'id_liga' => 1
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_insertar_equipo.php', $datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioIdLigaIncorrecto() {
        // Datos del nuevo equipo con un ID de liga incorrecto
        $datosEquipo = [
            'nombre_equipo' => 'Nuevo Equipo',
            'fecha_fundacion' => '2022-01-01',
            'presidente' => 'Presidente Nuevo',
            'color' => 'Rojo',
            'escudo' => 'http://example.com/escudo',
            'id_categoria' => 111,
            'id_liga' => 0
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_insertar_equipo.php', $datosEquipo);

        // Verificar si la solicitud fue exitosa
```

```
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_liga', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_liga']);
    }

    public function testEnvioIdCategoriaIncorrecto() {
        // Datos del nuevo equipo con un ID de categoría incorrecto
        $datosEquipo = [
            'nombre_equipo' => 'Nuevo Equipo',
            'fecha_fundacion' => '2022-01-01',
            'presidente' => 'Presidente Nuevo',
            'color' => 'Rojo',
            'escudo' => 'http://example.com/escudo',
            'id_categoria' => 0,
            'id_liga' => 1
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_insertar_equipo.php',
$datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_categoria', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_categoria']);
    }

    public function testEnvioDatosVacios() {
        // Datos vacíos
        $datosEquipo = [
            'nombre_equipo' => '',
            'fecha_fundacion' => '',
            'presidente' => '',
            'color' => '',
            'escudo' => '',
            'id_categoria' => 111,
            'id_liga' => 1
        ];

        // Realizar la solicitud POST al script PHP
```

```
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_insertar_equipo.php', $datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    public function testEnvioNombreEquipoExistenteEnLiga() {
        // Datos del nuevo equipo con un nombre ya existente en la liga
        $datosEquipo = [
            'nombre_equipo' => 'Nuevo Equipo',
            'fecha_fundacion' => '2022-01-01',
            'presidente' => 'Presidente Nuevo',
            'color' => 'Rojo',
            'escudo' => 'http://example.com/escudo',
            'id_categoria' => 111,
            'id_liga' => 1
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_insertar_equipo.php', $datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('existe_nombre', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['existe_nombre']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_4_insertar_equipo.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

.....                                                               5 / 5 (100%)

Time: 00:07.950, Memory: 8.00 MB

OK (5 tests, 15 assertions)
```

Prueba “GE001-1 Mostrar equipos de liga”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_4_mostrar_equipos_liga extends TestCase {

    public function testEnvioDatosCorrectos() {
        // Datos del ID de liga
        $datosLiga = [
            'id_liga' => 1
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_mostrar_equipos_liga.php', $datosLiga);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('datos', $cuerpoRespuesta);
        // Comprueba si la respuesta contiene al menos un equipo
        $this->assertGreaterThanOrEqual(1, count($cuerpoRespuesta['datos']));
    }

    public function testIdLigaIncorrecto() {
        // Datos del ID de liga incorrecto
        $datosLiga = [
            'id_liga' => "dos"
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_mostrar_equipos_liga.php', $datosLiga);

        // Verificar si la solicitud fue exitosa
```

```
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_liga', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_liga']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_4_mostrar_equipos_liga.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

..                                                                  2 / 2 (100%)

Time: 00:02.156, Memory: 8.00 MB

OK (2 tests, 7 assertions)
```

Prueba “PI001-5 Mostrar equipo”

```
<?php
use PHPUnit\Framework\TestCase;

class prueba_4_mostrar_equipo extends TestCase {

    public function testEnvioDatosCorrectos() {
        // Datos del equipo
        $datosEquipo = [
            'id_equipo' => 86,
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_mostrar_equipo.php',
$datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('datos', $cuerpoRespuesta);
        $this->assertNotEmpty($cuerpoRespuesta['datos']);
```

```
    }

    public function testIdEquipoIncorrecto() {
        // Datos de ID de equipo incorrecto
        $datosEquipo = [
            'id_equipo' => -1,
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_mostrar_equipo.php',
$datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_equipo', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_equipo']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_4_mostrar_equipo.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

..                                                                  2 / 2 (100%)

Time: 00:04.115, Memory: 8.00 MB

OK (2 tests, 6 assertions)
```

Sprint 2

Prueba “GE001-4 Registrar jugador”

```
<?php
use PHPUnit\Framework\TestCase;

class prueba_5_insertar_jugador extends TestCase {

    public function testEnvioDatosCorrectos() {
        // Datos del jugador
        $datosJugador = [
```

```
            'CI' => 12345678,
            'nombre' => 'Jugador de Prueba',
            'posicion' => 'Delantero',
            'fecha_nacimiento' => '1990-01-01',
            'foto' => 'https://ruta-de-la-imagen.com/foto.jpg',
            'estatura' => 180,
            'num_camiseta' => 10,
            'id_equipo' => 86
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_insertar_jugador.php', $datosJugador);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioIdEquipoIncorrecto() {
        // Datos del jugador
        $datosJugador = [
            'CI' => 12345677,
            'nombre' => 'Jugador de Prueba',
            'posicion' => 'Delantero',
            'fecha_nacimiento' => '1990-01-01',
            'foto' => 'https://ruta-de-la-imagen.com/foto.jpg',
            'estatura' => 180,
            'num_camiseta' => 11,
            'id_equipo' => 9999 // Un ID de equipo inexistente
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_insertar_jugador.php', $datosJugador);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_equipo', $cuerpoRespuesta);
```

```
        $this->assertTrue($cuerpoRespuesta['no_existe_equipo']);
    }

    public function testEnvioCedulaExistente() {
        // Datos del jugador con una cédula existente
        $datosJugador = [
            'CI' => 12345678, // Una cédula que ya existe en la base de
datos
            'nombre' => 'Jugador de Prueba',
            'posicion' => 'Delantero',
            'fecha_nacimiento' => '1990-01-01',
            'foto' => 'https://ruta-de-la-imagen.com/foto.jpg',
            'estatura' => 180,
            'num_camiseta' => 11,
            'id_equipo' => 86
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_insertar_jugador.php'
, $datosJugador);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('existe_jugador', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['existe_jugador']);
    }

    public function testEnvioDatosVacios() {
        // Datos del jugador vacíos
        $datosJugador = [
            'CI' => 12345677,
            'nombre' => '',
            'posicion' => '',
            'fecha_nacimiento' => '',
            'foto' => '',
            'estatura' => '',
            'num_camiseta' => '',
            'id_equipo' => 86
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_insertar_jugador.php'
, $datosJugador);
```

```php
        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    public function testEnvioNumeroCamisetaExistente() {
        // Datos del jugador con un número de camiseta existente
        $datosJugador = [
            'CI' => 12345677,
            'nombre' => 'Jugador de Prueba',
            'posicion' => 'Delantero',
            'fecha_nacimiento' => '1990-01-01',
            'foto' => 'https://ruta-de-la-imagen.com/foto.jpg',
            'estatura' => 180,
            'num_camiseta' => 10,
            'id_equipo' => 86
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_insertar_jugador.php'
, $datosJugador);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('exite_numero_camiseta', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['exite_numero_camiseta']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_5_insertar_jugador.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

.....                                                               5 / 5 (100%)

Time: 00:07.346, Memory: 8.00 MB

OK (5 tests, 15 assertions)
```

Prueba “GT001-1 Registrar torneos”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_6_insertar_torneo extends TestCase {
    public function testEnvioDeDatosCorrectosSinGrupos() {
        // Datos del torneo
        $datosTorneo = [
            'etapa' => 'Primera etapa',
            'fecha_inicio' => '2024-03-01',
            'fecha_fin' => '2024-03-10',
            'canchas' => 'Cancha1,Cancha2',
            'num_clasificados' => 2,
            'id_categoria' => 111
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/6_insertar_torneo.php', $datosTorneo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioDeDatosCorrectosConGrupos() {
        // Datos del torneo
        $datosTorneo = [
            'etapa' => 'Fase de grupos',
            'fecha_inicio' => '2024-03-01',
            'fecha_fin' => '2024-03-10',
            'canchas' => 'Cancha1,Cancha2',
            'grupo' => 2,
            'num_clasificados' => '2,2',
            'id_categoria' => 112
```

```
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/6_insertar_torneo.php', $datosTorneo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioDeIdCategoriaIncorrecto() {
        // Datos del torneo
        $datosTorneo = [
            'etapa' => 'Fase de grupos',
            'fecha_inicio' => '2024-03-01',
            'fecha_fin' => '2024-03-10',
            'canchas' => 'Cancha1,Cancha2',
            'num_clasificados' => 2,
            'id_categoria' => 0
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/6_insertar_torneo.php', $datosTorneo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_categoria', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_categoria']);
    }

    public function testEnvioDeDatosVacios() {
        // Datos del torneo
        $datosTorneo = [
            'etapa' => '',
            'fecha_inicio' => '',
            'fecha_fin' => '',
            'canchas' => '',
```

```
            'num_clasificados' => '',
            'id_categoria' => 113
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/6_insertar_torneo.php', $datosTorneo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_6_insertar_torneo.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

....                                                                4 / 4 (100%)

Time: 00:05.002, Memory: 8.00 MB

OK (4 tests, 12 assertions)
```

Prueba “GT001-2 Generar partidos”

```
<?php
use PHPUnit\Framework\TestCase;

class prueba_7_generar_partidos extends TestCase {
    public function testEnvioDeDatosCorrectos() {
        // Datos del torneo
        $datosPartidos = [
            'id_torneo' => 42,
            'equipos' => [86, 87, 88, 89]
        ];

        // Realizar la solicitud POST al script PHP
```

```
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_generar_partidos.php', $datosPartidos);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioDeIdTorneoIncorrecto() {
        // Datos del torneo
        $datosPartidos = [
            'id_torneo' => 1000,
            'equipos' => [1, 2, 3, 4]
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_generar_partidos.php', $datosPartidos);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_torneo', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_torneo']);
    }

    public function testEnvioDeIdTorneoConPartidosRegistrados() {
        // Datos del torneo
        $datosPartidos = [
            'id_torneo' => 42,
            'equipos' => [86, 87, 88, 89]
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_generar_partidos.php', $datosPartidos);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());
```

```
        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('existen_partidos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['existen_partidos']);
    }

    public function testEnvioDeDatosVacios() {
        // Datos del torneo
        $datosPartidos = [
            'id_torneo' => 43,
            'equipos' => []
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_generar_partidos.php'
, $datosPartidos);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    public function testEnvioDeIdEquipoIncorrecto() {
        // Datos del torneo
        $datosPartidos = [
            'id_torneo' => 43,
            'equipos' => [1, 2, 99, 4] // 99 es un ID de equipo que no
existe
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_generar_partidos.php'
, $datosPartidos);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existen_equipo', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existen_equipo']);
```

```
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_7_generar_partidos.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

.....                                                               5 / 5 (100%)

Time: 00:11.396, Memory: 8.00 MB

OK (5 tests, 15 assertions)
```

Prueba “GT001-3 Mostrar torneos”

```
<?php
use PHPUnit\Framework\TestCase;

class prueba_6_mostar_torneos extends TestCase {
    public function testEnvioDeDatosCorrectos() {
        // Datos del torneo
        $datos = [
            'id_categoria' => 111
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/6_mostar_torneos.php',
$datos);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('datos', $cuerpoRespuesta);
    }

    public function testEnvioDeIdCategoriaIncorrecto() {
        // Datos del torneo
        $datos = [
            'id_categoria' => 9999
        ];
```

```
        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/6_mostar_torneos.php',
$datos);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_categoria', $cuerpoRespuesta);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_6_mostar_torneos.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

..                                                                  2 / 2 (100%)

Time: 00:02.247, Memory: 8.00 MB

OK (2 tests, 4 assertions)
```

Prueba “GP001-1 Programar partidos”

```
<?php
use PHPUnit\Framework\TestCase;

class prueba_7_programar_partidos extends TestCase {
    public function testEnvioDeDatosCorrectos() {
        // Datos del partido
        $datos = [
            'id_partido' => 52,
            'partido' => [
                '2024-02-14',
                '08:00',
                'cancha 1',
                'equipo 1',
                ''
            ]
        ];

        // Realizar la solicitud POST al script PHP
```

```
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_programar_partidos.php', $datos);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
    }

    public function testEnvioDeIdPartidoIncorrecto() {
        // Datos del partido
        $datos = [
            'id_partido' => 9999,
            'partido' => [
                '2024-02-14',
                '08:00',
                'cancha 1',
                'equipo 1',
                ''
            ]
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_programar_partidos.php', $datos);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_partido', $cuerpoRespuesta);
    }

    public function testEnvioDeIdPartidoEnEstadoIncorrecto() {
        // Datos del partido
        $datos = [
            'id_partido' => 52,
            'partido' => [
                '2024-02-14',
                '08:00',
                'cancha 1',
                'equipo 1',
                ''
```

```
            ]
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_programar_partidos.php', $datos);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('partido_programado', $cuerpoRespuesta);
    }

    public function testEnvioDeDatosVacios() {
        // Datos del partido
        $datos = [
            'id_partido' => 53,
            'partido' => []
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_programar_partidos.php', $datos);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_7_programar_partidos.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

....                                                                4 / 4 (100%)

Time: 00:04.151, Memory: 8.00 MB

OK (4 tests, 8 assertions)
```

Prueba “PI001-1 Mostrar todas las ligas inscritas”

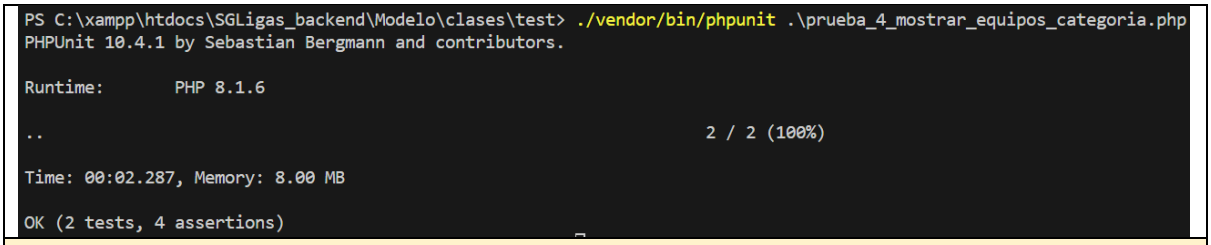
```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_2_mostrar_todas_ligas extends TestCase {
    public function testConsultaCorrecta() {
        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/2_mostrar_todas_ligas.php', []);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('datos', $cuerpoRespuesta);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_2_mostrar_todas_ligas.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

.                                                                   1 / 1 (100%)

Time: 00:00.803, Memory: 8.00 MB

OK (1 test, 2 assertions)
```

Prueba “PI001-2 Mostrar equipos de categoría”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_4_mostrar_equipos_categoria extends TestCase {
```

```php
    public function testConsultaCorrecta() {
        // Datos de la categoria
        $datosCategoria = [
            'id_categoria' => 111 // ID de la categoria
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_mostrar_equipos_categoria.php', $datosCategoria);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('datos', $cuerpoRespuesta);
    }

    public function testConsultaIncorrecta() {
        // Datos de la categoria
        $datosCategoria = [
            'id_categoria' => 0 // ID de la categoria incorrecto
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_mostrar_equipos_categoria.php', $datosCategoria);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_categoria', $cuerpoRespuesta);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```



Prueba “GE001-5 Mostrar jugadores”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_5_mostrar_jugadores extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del equipo
        $datosEquipo = [
            'id_equipo' => 86 // ID de un equipo existente
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_mostrar_jugadores.php', $datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('datos', $cuerpoRespuesta);
    }

    public function testEnvioIdEquipoIncorrecto() {
        // Datos del equipo
        $datosEquipo = [
            'id_equipo' => 999999 // ID de un equipo no existente
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_mostrar_jugadores.php', $datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_equipo', $cuerpoRespuesta);
```

```
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_5_mostrar_jugadores.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

..                                                                  2 / 2 (100%)

Time: 00:02.261, Memory: 8.00 MB

OK (2 tests, 4 assertions)
```

Prueba “PI001-3 Mostrar posiciones”

```
<?php
use PHPUnit\Framework\TestCase;

class prueba_6_mostrar_posiciones extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del torneo
        $datosTorneo = [
            'id_torneo' => 42 // ID del torneo válido
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/6_mostrar_posiciones.ph
p', $datosTorneo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('tabla', $cuerpoRespuesta);
    }

    public function testEnvioIdTorneoIncorrecto() {
        // Datos del torneo
        $datosTorneo = [
            'id_torneo' => 0 // ID del torneo no válido
        ];
```

```
        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/6_mostrar_posiciones.ph
p', $datosTorneo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_torneo', $cuerpoRespuesta);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_6_mostrar_posiciones.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

..                                                                  2 / 2 (100%)

Time: 00:02.285, Memory: 8.00 MB

OK (2 tests, 4 assertions)
```

Prueba “GP001-2 Registrar resultados”

```
<?php
use PHPUnit\Framework\TestCase;

class prueba_7_insertar_resultados extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 42,
            'id_partido' => 52,
            'partido' => [86,2,3,89]
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_resultados.p
hp', $datosPartido);

        // Verificar si la solicitud fue exitosa
```

```
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioIdTorneoIncorrecto() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 0,
            'id_partido' => 1,
            'partido' => [86,2,3,89]
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_resultados.p
hp', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_torneo', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_torneo']);
    }

    public function testEnvioIdTorneoSinPartidos() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 43,
            'id_partido' => 1,
            'partido' => [86,2,3,89]
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_resultados.p
hp', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
```

```
        $this->assertArrayHasKey('no_existen_partidos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existen_partidos']);
    }

    public function testEnvioIdPartidoIncorrecto() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 42,
            'id_partido' => 0,
            'partido' => [86,2,3,89]
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_resultados.p
hp', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_partido', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_partido']);
    }

    public function testEnvioIdPartidoEstadoIncorrecto() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 42,
            'id_partido' => 53,
            'partido' => [86,2,3,89]
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_resultados.p
hp', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('partido_no_programado', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['partido_no_programado']);
    }
```

```php
    public function testEnvioDatosVacios() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 42,
            'id_partido' => 54,
            'partido' => ''
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_resultados.p
hp', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_7_insertar_resultados.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

......                                                            6 / 6 (100%)

Time: 00:10.168, Memory: 8.00 MB

OK (6 tests, 18 assertions)
```

Prueba “GP001-3 Comprobar registro de estadísticas de jugadores”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_7_comprobar_estadisticas_jugador extends TestCase {
    public function testEnvioDatosCorrectosTieneRegistros() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 36,
            'id_partido' => 17,
```

```
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_comprobar_estadisticas_jugador.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('existen_registros', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['existen_registros']);
    }

    public function testEnvioDatosCorrectosNoTieneRegistros() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 42,
            'id_partido' => 52,
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_comprobar_estadisticas_jugador.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existen_registros', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existen_registros']);
    }

    public function testEnvioIdTorneoIncorrecto() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 0,
            'id_partido' => 1,
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_comprobar_estadisticas_jugador.php', $datosPartido);
```

```
        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_torneo', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_torneo']);
    }

    public function testEnvioIdPartidoIncorrecto() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 36,
            'id_partido' => 0,
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_comprobar_estadistica
s_jugador.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_partido', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_partido']);
    }

    public function testEnvioIdPartidoEstadoIncorrecto() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 42,
            'id_partido' => 55,
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_comprobar_estadistica
s_jugador.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
```

```
        $this->assertArrayHasKey('partido_no_programado', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['partido_no_programado']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}

?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_7_comprobar_estadisticas_jugador.php

PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

.....                                                               5 / 5 (100%)

Time: 00:07.791, Memory: 8.00 MB

OK (5 tests, 15 assertions)
```

Prueba “GP001-4 Registrar estadísticas jugadores: rojas, amarillas, goles, goles recibidos en el caso de porteros, autogoles y partidos jugados.”

```
<?php
use PHPUnit\Framework\TestCase;

class prueba_7_insertar_estaditicas_jugadores extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 42,
            'id_partido' => 52,
            'jugadores' => [
                [1313131313,1,0,1,0,1],
                [1515151515,1,0,1,0,1]
            ]
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_estaditicas_
jugadores.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
```

```
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioIdTorneoIncorrecto() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 0,
            'id_partido' => 1,
            'jugadores' => [
                [1313131313,1,0,1,0,1],
                [1515151515,1,0,1,0,1]
            ]
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_estaditicas_
jugadores.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_torneo', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_torneo']);
    }

    public function testEnvioIdTorneoSinPartidos() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 43,
            'id_partido' => 1,
            'jugadores' => [
                [1313131313,1,0,1,0,1],
                [1515151515,1,0,1,0,1]
            ]
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_estaditicas_
jugadores.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());
```

```
        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existen_partidos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existen_partidos']);
    }

    public function testEnvioIdPartidoIncorrecto() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 42,
            'id_partido' => 0,
            'jugadores' => [
                [1313131313,1,0,1,0,1],
                [1515151515,1,0,1,0,1]
            ]
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_estaditicas_
jugadores.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_partido', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_partido']);
    }

    public function testEnvioIdPartidoEstadoIncorrecto() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 42,
            'id_partido' => 53,
            'jugadores' => [
                [1313131313,1,0,1,0,1],
                [1515151515,1,0,1,0,1]
            ]
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_estaditicas_
jugadores.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
```

```php
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('partido_no_programado', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['partido_no_programado']);
    }

    public function testEnvioDatosVacios() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 42, // Datos vacíos
            'id_partido' => 52,
            'jugadores' => ''
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_estaditicas_jugadores.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_7_insertar_estaditicas_jugadores.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.
Runtime:       PHP 8.1.6
......                                                            6 / 6 (100%)
Time: 00:11.477, Memory: 8.00 MB
OK (6 tests, 18 assertions)
```

Prueba “PI001-4 Mostrar partidos”

```php
<?php
```

```php
use PHPUnit\Framework\TestCase;

class prueba_7_mostrar_partidos extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del torneo
        $datosTorneo = [
            'id_torneo' => 42
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_mostrar_partidos.php', $datosTorneo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('partidos', $cuerpoRespuesta);
        $this->assertIsArray($cuerpoRespuesta['partidos']);
        $this->assertNotEmpty($cuerpoRespuesta['partidos']);
    }

    public function testEnvioIdTorneoIncorrecto() {
        // Datos del torneo
        $datosTorneo = [
            'id_torneo' => 0
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_mostrar_partidos.php', $datosTorneo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_torneo', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_torneo']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
```

```
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_7_mostrar_partidos.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

..                                                                  2 / 2 (100%)

Time: 00:02.476, Memory: 8.00 MB

OK (2 tests, 7 assertions)
```

Prueba “PI001-6 Mostrar partidos de cada equipo”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_7_mostrar_partidos_equipo extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 42,
            'id_equipo' => 86
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_mostrar_partidos_equipo.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('partidos', $cuerpoRespuesta);
        $this->assertNotEmpty($cuerpoRespuesta['partidos']);
    }

    public function testEnvioIdTorneoIncorrecto() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 0, // ID de un torneo que no existe
            'id_equipo' => 1 // ID de un equipo existente
        ];

        // Realizar la solicitud POST al script PHP
```

```php
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_mostrar_partidos_equipo.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_torneo', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_torneo']);
    }

    public function testEnvioIdEquipoIncorrecto() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 42, // ID de un torneo existente con partidos
            'id_equipo' => 0 // ID de un equipo que no existe
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_mostrar_partidos_equipo.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_equipo', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_equipo']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_7_mostrar_partidos_equipo.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

...                                                                 3 / 3 (100%)

Time: 00:03.946, Memory: 8.00 MB

OK (3 tests, 9 assertions)
```

Prueba “PI001-7 Mostrar estadísticas de jugadores de equipo”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_5_mostrar_estadisticas_jugador_equipo extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del equipo
        $datosEquipo = [
            'id_equipo' => 86,
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_mostrar_estadisticas_jugador_equipo.php', $datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('datos', $cuerpoRespuesta);
        $this->assertNotEmpty($cuerpoRespuesta['datos']);
    }

    public function testEnvioIdEquipoIncorrecto() {
        // Datos del equipo
        $datosEquipo = [
            'id_equipo' => 0,
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_mostrar_estadisticas_jugador_equipo.php', $datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_equipo', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_equipo']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
```

```
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_5_mostrar_estadisticas_jugador_equipo.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

..                                                                  2 / 2 (100%)

Time: 00:02.365, Memory: 8.00 MB

OK (2 tests, 6 assertions)
```

Prueba “PI001-8 Mostrar información de jugador”

```
<?php
use PHPUnit\Framework\TestCase;

class prueba_5_mostrar_informacion_jugador extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del jugador
        $datosJugador = ['CI' => 1313131313];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_mostrar_informacion_jugador.php', $datosJugador);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('datos', $cuerpoRespuesta);
        $this->assertNotEmpty($cuerpoRespuesta['datos']);
    }

    public function testEnvioCedulaIncorrecta() {
        // Datos del jugador con CI incorrecta
        $datosJugador = ['CI' => 0];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_mostrar_informacion_jugador.php', $datosJugador);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());
```

```php
        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_CI', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_CI']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_5_mostrar_informacion_jugador.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

..                                                                  2 / 2 (100%)

Time: 00:02.240, Memory: 8.00 MB

OK (2 tests, 6 assertions)
```

Prueba “GP001-5 Registrar alineación”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_7_insertar_alineacion extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos de la alineación
        $datosAlineacion = [
            'id_partido' => 54,
            'alineacion' =>
'{"local":[{"id_jugador":1,"numero_camisa":10},{"id_jugador":2,"numero_camisa":5}],"visitante":[{"id_jugador":3,"numero_camisa":7},{"id_jugador":4,"numero_camisa":3}]}'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_alineacion.php', $datosAlineacion);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
```

```
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioIdPartidoIncorrecto() {
        // Datos de la alineación
        $datosAlineacion = [
            'id_partido' => 0,
            'alineacion' =>
'{"local":[{"id_jugador":1,"numero_camisa":10},{"id_jugador":2,"numero_camis
a":5}],"visitante":[{"id_jugador":3,"numero_camisa":7},{"id_jugador":4,"nume
ro_camisa":3}]}'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_alineacion.p
hp', $datosAlineacion);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_partido', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_partido']);
    }

    public function testEnvioIdPartidoEstadoIncorrecto() {
        // Datos de la alineación
        $datosAlineacion = [
            'id_partido' => 52,
            'alineacion' =>
'{"local":[{"id_jugador":1,"numero_camisa":10},{"id_jugador":2,"numero_camis
a":5}],"visitante":[{"id_jugador":3,"numero_camisa":7},{"id_jugador":4,"nume
ro_camisa":3}]}'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_alineacion.p
hp', $datosAlineacion);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('partido_jugado', $cuerpoRespuesta);
```

```
        $this->assertTrue($cuerpoRespuesta['partido_jugado']);
    }

    public function testEnvioDatosVacios() {
        // Datos de la alineación
        $datosAlineacion = [
            'id_partido' => 54,
            'alineacion' => ''
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_alineacion.p
hp', $datosAlineacion);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    public function testEnvioIdPartidoConAlineacionRegistrada() {
        // Datos de la alineación
        $datosAlineacion = [
            'id_partido' => 7,
            'alineacion' =>
'{"local":[{"id_jugador":1,"numero_camisa":10},{"id_jugador":2,"numero_camis
a":5}],"visitante":[{"id_jugador":3,"numero_camisa":7},{"id_jugador":4,"nume
ro_camisa":3}]}'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_insertar_alineacion.p
hp', $datosAlineacion);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('existe_alienacion', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['existe_alienacion']);
    }
```

```
    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_7_insertar_alineacion.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

.....                                                               5 / 5 (100%)

Time: 00:06.597, Memory: 8.00 MB

OK (5 tests, 15 assertions)
```

Prueba “PI001-9 Mostrar alineación”

```
<?php
use PHPUnit\Framework\TestCase;

class prueba_7_mostrar_alineacion extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del partido
        $datosPartido = [
            'id_partido' => 54 // ID de un partido existente en estado
jugado
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_mostrar_alineacion.ph
p', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('datos', $cuerpoRespuesta);
        // Verificar que se devuelvan datos de la alineación del partido
        $this->assertNotEmpty($cuerpoRespuesta['datos']);
    }

    public function testEnvioIdPartidoIncorrecto() {
        // Datos del partido
        $datosPartido = [
            'id_partido' => 0 // ID de un partido que no existe
```

```
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_mostrar_alineacion.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_partido', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_partido']);
    }

    public function testEnvioIdPartidoEstadoIncorrecto() {
        // Datos del partido
        $datosPartido = [
            'id_partido' => 53 // ID de un partido en estado que no es jugado
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/7_mostrar_alineacion.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('partido_no_programado', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['partido_no_programado']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_7_mostrar_alineacion.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

...                                                                 3 / 3 (100%)

Time: 00:03.161, Memory: 8.00 MB

OK (3 tests, 9 assertions)
```

Prueba “GP001-6 Registrar informe de sanciones”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_8_insertar_sancion_informe extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del informe de sanciones
        $datosInformeSanciones = [
            'id_partido' => 52,
            'informe_local' => 'Este es el informe local',
            'informe_visitante' => 'Este es el informe visitante',
            'arbitro' => 'Juan Pérez'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/8_insertar_sancion_info
rme.php', $datosInformeSanciones);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioIdPartidoIncorrecto() {
        // Datos del informe de sanciones
        $datosInformeSanciones = [
            'id_partido' => 0,
            'informe_local' => 'Este es el informe local',
            'informe_visitante' => 'Este es el informe visitante',
            'arbitro' => 'Juan Pérez'
        ];

        // Realizar la solicitud POST al script PHP
```

```
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/8_insertar_sancion_informe.php', $datosInformeSanciones);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_partido', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_partido']);
    }

    public function testEnvioIdPartidoEstadoIncorrecto() {
        // Datos del informe de sanciones
        $datosInformeSanciones = [
            'id_partido' => 54,
            'informe_local' => 'Este es el informe local',
            'informe_visitante' => 'Este es el informe visitante',
            'arbitro' => 'Juan Pérez'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/8_insertar_sancion_informe.php', $datosInformeSanciones);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('partido_no_jugado', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['partido_no_jugado']);
    }

    public function testEnvioIdPartidoConSancionRegistrada() {
        // Datos del informe de sanciones
        $datosInformeSanciones = [
            'id_partido' => 1,
            'informe_local' => 'Este es el informe local',
            'informe_visitante' => 'Este es el informe visitante',
            'arbitro' => 'Juan Pérez'
        ];

        // Realizar la solicitud POST al script PHP
```

```
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/8_insertar_sancion_informe.php', $datosInformeSanciones);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('existe_sancion', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['existe_sancion']);
    }

    public function testEnvioDatosVacios() {
        // Datos del informe de sanciones
        $datosInformeSanciones = [
            'id_partido' => 2, // ID de un partido vacío
            'informe_local' => '', // Informe local vacío
            'informe_visitante' => '', // Informe visitante vacío
            'arbitro' => '' // Nombre del árbitro vacío
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/8_insertar_sancion_informe.php', $datosInformeSanciones);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_8_insertar_sancion_informe.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

.....                                                               5 / 5 (100%)

Time: 00:06.578, Memory: 8.00 MB

OK (5 tests, 15 assertions)
```

Prueba “GP001-7 Registrar tribunal de sanciones”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_8_insertar_sancion_tribunal extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del partido y sanciones
        $datosSanciones = [
            'id_partido' => 52,
            'informe_local' => 'Informe sanción local',
            'informe_visitante' => 'Informe sanción visitante',
            'responsables' => 'Responsable 1, Responsable 2'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/8_insertar_sancion_tribunal.php', $datosSanciones);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioIdPartidoIncorrecto() {
        // Datos del partido y sanciones
        $datosSanciones = [
            'id_partido' => 0, // ID de un partido que no existe
            'informe_local' => 'Informe sanción local',
            'informe_visitante' => 'Informe sanción visitante',
            'responsables' => 'Responsable 1, Responsable 2'
        ];

        // Realizar la solicitud POST al script PHP
```

```
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/8_insertar_sancion_trib
unal.php', $datosSanciones);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_partido', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_partido']);
    }

    public function testEnvioIdPartidoEstadoIncorrecto() {
        // Datos del partido y sanciones
        $datosSanciones = [
            'id_partido' => 54,
            'informe_local' => 'Informe sanción local',
            'informe_visitante' => 'Informe sanción visitante',
            'responsables' => 'Responsable 1, Responsable 2'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/8_insertar_sancion_trib
unal.php', $datosSanciones);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('partido_no_jugado', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['partido_no_jugado']);
    }

    public function testEnvioIdPartidoSinSanciones() {
        // Datos del partido y sanciones
        $datosSanciones = [
            'id_partido' => 2, // ID de un partido existente que está jugado
pero no tiene sanciones
            'informe_local' => 'Informe sanción local',
            'informe_visitante' => 'Informe sanción visitante',
            'responsables' => 'Responsable 1, Responsable 2'
        ];

        // Realizar la solicitud POST al script PHP
```

```php
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/8_insertar_sancion_tribunal.php', $datosSanciones);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_sancion', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_sancion']);
    }

    public function testEnvioDatosVacios() {
        // Datos del partido y sanciones
        $datosSanciones = [
            'id_partido' => 52, // Datos vacíos
            'informe_local' => '',
            'informe_visitante' => '',
            'responsables' => ''
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/8_insertar_sancion_tribunal.php', $datosSanciones);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_8_insertar_sancion_tribunal.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

.....                                                               5 / 5 (100%)

Time: 00:06.193, Memory: 8.00 MB

OK (5 tests, 15 assertions)
```

Prueba “GP001-8 Mostrar acta de juego”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_8_mostrar_acta_de_juego extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 42,
            'id_partido' => 52
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/8_mostrar_acta_de_juego.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('datos_partido', $cuerpoRespuesta);
        $this->assertArrayHasKey('estadisticas', $cuerpoRespuesta);
        $this->assertArrayHasKey('sanciones', $cuerpoRespuesta);
    }

    public function testEnvioIdTorneoIncorrecto() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 0, // ID de un torneo que no existe
            'id_partido' => 1 // ID de un partido existente en un torneo existente
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/8_mostrar_acta_de_juego.php', $datosPartido);
```

```
        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_torneo', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_torneo']);
    }

    public function testEnvioIdPartidoIncorrecto() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 42, // ID de un torneo existente
            'id_partido' => 0 // ID de un partido que no existe
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/8_mostrar_acta_de_juego
.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_partido', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_partido']);
    }

    public function testEnvioIdPartidoEstadoIncorrecto() {
        // Datos del partido
        $datosPartido = [
            'id_torneo' => 42, // ID de un torneo existente
            'id_partido' => 54 // ID de un partido existente en un torneo
existente
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/8_mostrar_acta_de_juego
.php', $datosPartido);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
```

```
        $this->assertArrayHasKey('partido_por_jugar', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['partido_por_jugar']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_8_mostrar_acta_de_juego.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

....                                                                4 / 4 (100%)

Time: 00:06.827, Memory: 8.00 MB

OK (4 tests, 13 assertions)
```

Prueba “GU001-3 Recuperar contraseña”

```
<?php
use PHPUnit\Framework\TestCase;

class prueba_1_recuperar_contraseña extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del usuario
        $datosUsuario = [
            'correo' => 'carlospnppm@gmail.com' // Correo de un usuario
registrado en tu sistema
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/1_recuperar_contrase%C3
%B1a.php', $datosUsuario);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
        $this->assertArrayHasKey('message', $cuerpoRespuesta);
        $this->assertArrayHasKey('correoUsuario', $cuerpoRespuesta);
        $this->assertEquals($datosUsuario['correo'],
$cuerpoRespuesta['correoUsuario']);
```

```
        $this->assertArrayHasKey('codigoVerificacion', $cuerpoRespuesta);
        $this->assertEquals(6,
strlen($cuerpoRespuesta['codigoVerificacion'])); // Verificar que el código
de verificación tiene 6 caracteres
    }

    public function testEnvioCorreoNoRegistrado() {
        // Datos del usuario
        $datosUsuario = [
            'correo' => 'correo_que_no_existe@ejemplo.com' // Correo que no
está registrado en tu sistema
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/1_recuperar_contrase%C3
%B1a.php', $datosUsuario);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertFalse($cuerpoRespuesta['success']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}

?>
```

Prueba “GU001-4 Actualizar usuario”

```
<?php
use PHPUnit\Framework\TestCase;

class prueba_1_actualizar_datos_usuario extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del usuario
        $datosUsuario = [
            'correo' => 'nuevo_usuario@example.com', // Correo de un usuario
existente
```

```
            'nombre' => 'Nuevo Nombre', // Nuevo nombre para el usuario
            'tipo' => 'hincha' // Tipo de usuario
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/1_actualizar_datos_usuario.php', $datosUsuario);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioDatosCorrectosContraseña() {
        // Datos del usuario
        $datosUsuario = [
            'correo' => 'nuevo_usuario@example.com', // Correo de un usuario existente
            'password' => 'nueva_contra', // Nuevo nombre para el usuario
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/1_actualizar_contraseña_usuario.php', $datosUsuario);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioDatosVacios() {
        // Datos del usuario
        $datosUsuario = [
            'correo' => '', // Correo vacío
            'nombre' => '', // Nombre vacío
            'tipo' => '' // Tipo vacío
        ];
```

```php
        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/1_actualizar_datos_usuario.php', $datosUsuario);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    public function testEnvioCorreoNoRegistrado() {
        // Datos del usuario
        $datosUsuario = [
            'correo' => 'correo_no_registrado@example.com', // Correo de un usuario no registrado
            'nombre' => 'Nuevo Nombre', // Nuevo nombre para el usuario
            'tipo' => 'presidente' // Tipo de usuario
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/1_actualizar_datos_usuario.php', $datosUsuario);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertFalse($cuerpoRespuesta['success']);
    }
    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_1_actualizar_datos_usuario.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

....                                                                4 / 4 (100%)

Time: 00:04.982, Memory: 8.00 MB

OK (4 tests, 12 assertions)
```

Prueba “GL001-5 Actualizar categoría”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_3_actualizar_categoria extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos de la categoría
        $datosCategoria = [
            'categoria' => 'Super Maxima', // Nombre de la categoría
            'num_equipos' => 4, // Número de equipos
            'id_categoria' => 111 // ID de una categoría existente en la base de datos
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/3_actualizar_categoria.php', $datosCategoria);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioIdCategoriaIncorrecto() {
        // Datos de la categoría
        $datosCategoria = [
            'categoria' => 'NombreCategoria', // Nombre de la categoría
            'num_equipos' => 10, // Número de equipos
            'id_categoria' => 0 // ID de una categoría que no existe en la base de datos
        ];

        // Realizar la solicitud POST al script PHP
```

```php
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/3_actualizar_categoria.php', $datosCategoria);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_categoria', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_categoria']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_3_actualizar_categoria.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

..                                                                  2 / 2 (100%)

Time: 00:03.005, Memory: 8.00 MB

OK (2 tests, 6 assertions)
```

Prueba “GE001-6 Actualizar equipo”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_4_actualizar_equipo extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del equipo
        $datosEquipo = [
            'nombre_equipo' => 'Nuevo nombre', // Nuevo nombre para el equipo
            'presidente' => 'Nuevo presidente', // Nuevo presidente del equipo
            'escudo' => 'http://url.com/escudo', // Nueva URL del escudo
            'id_equipo' => 86 // ID de un equipo existente
        ];

        // Realizar la solicitud POST al script PHP
```

```
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_actualizar_equipo.php', $datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioIdEquipoIncorrecto() {
        // Datos del equipo
        $datosEquipo = [
            'nombre_equipo' => 'Nuevo nombre', // Nuevo nombre para el equipo
            'presidente' => 'Nuevo presidente', // Nuevo presidente del equipo
            'escudo' => 'http://url.com/escudo', // Nueva URL del escudo
            'id_equipo' => 0 // ID de un equipo que no existe
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_actualizar_equipo.php', $datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_equipo', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_equipo']);
    }

    public function testEnvioDatosVacios() {
        // Datos del equipo
        $datosEquipo = [
            'nombre_equipo' => '', // Datos vacíos
            'presidente' => '',
            'escudo' => '',
            'id_equipo' => 86
        ];

        // Realizar la solicitud POST al script PHP
```

```
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_actualizar_equipo.php
', $datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

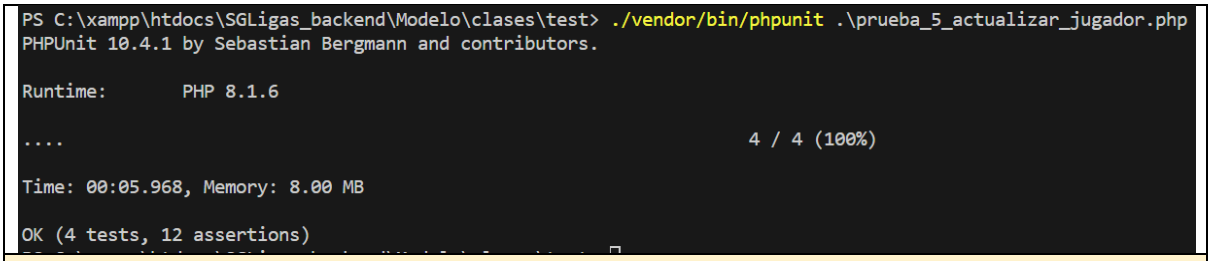
```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_4_actualizar_equipo.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

...                                                                 3 / 3 (100%)

Time: 00:03.332, Memory: 8.00 MB

OK (3 tests, 9 assertions)
```

Prueba “GE001-7 Actualizar jugador”

```
<?php
use PHPUnit\Framework\TestCase;

class prueba_5_actualizar_jugador extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del jugador
        $datosJugador = [
            'CI' => 1313131313, // CI no existente
            'posicion' => 'Delantero',
            'foto' => 'https://example.com/jugador1.jpg',
            'estatura' => 180, // cm
            'num_camiseta' => 11, // Número de camiseta
            'id_equipo' => 86 // ID de un equipo existente
        ];

        // Realizar la solicitud POST al script PHP
```

```
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_actualizar_jugador.php', $datosJugador);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioIdEquipoIncorrecto() {
        // Datos del jugador
        $datosJugador = [
            'CI' => 123456789, // CI no existente
            'posicion' => 'Delantero',
            'foto' => 'https://example.com/jugador1.jpg',
            'estatura' => 180, // cm
            'num_camiseta' => 10, // Número de camiseta
            'id_equipo' => 0 // ID de un equipo que no existe
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_actualizar_jugador.php', $datosJugador);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_equipo', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_equipo']);
    }

    public function testEnvioCIExistente() {
        // Datos del jugador
        $datosJugador = [
            'CI' => 0, // CI que ya existe en la base de datos
            'posicion' => 'Delantero',
            'foto' => 'https://example.com/jugador1.jpg',
            'estatura' => 180, // cm
            'num_camiseta' => 10, // Número de camiseta
            'id_equipo' => 86 // ID de un equipo existente
        ];
```

```php
        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_actualizar_jugador.php', $datosJugador);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_jugador', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_jugador']);
    }

    public function testEnvioDatosVacios() {
        // Datos del jugador
        $datosJugador = [
            'CI' => 1313131313, // Datos vacíos
            'posicion' => '',
            'foto' => '',
            'estatura' => '',
            'num_camiseta' => '',
            'id_equipo' => 86
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_actualizar_jugador.php', $datosJugador);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```



Prueba “GU001-5 Eliminar usuario”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_1_eliminar_usuario_hincha extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del usuario
        $datosUsuario = [
            'correo' => 'wilmertituana10@gmail.com'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/1_eliminar_usuario_hincha.php', $datosUsuario);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioDatosVacios() {
        // Datos vacíos
        $datosUsuario = [
            'correo' => ''
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/1_eliminar_usuario_hincha.php', $datosUsuario);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
```

```
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('noHayDatos', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['noHayDatos']);
    }

    public function testEnvioCorreoNoRegistrado() {
        // Datos del usuario con correo no registrado
        $datosUsuario = [
            'correo' => 'correo_no_registrado@ejemplo.com'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/1_eliminar_usuario_hinc
ha.php', $datosUsuario);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertFalse($cuerpoRespuesta['success']);
    }

    public function testEnvioCorreoEsPresidente() {
        // Datos del usuario con correo del presidente
        $datosUsuario = [
            'correo' => 'admin2@gmail.com'
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this-
>solicitarPost('http://localhost:8080/SGLIGAS/Modelo/1_eliminar_usuario_hinc
ha.php', $datosUsuario);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('es_presidente', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['es_presidente']);
    }

    private function solicitarPost($url, $datos) {
        // Simular una solicitud POST a través de GuzzleHTTP
        $cliente = new \GuzzleHttp\Client();
```

```
        $respuesta = $cliente->post($url, [
            'form_params' => $datos
        ]);
        return $respuesta;
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_1_eliminar_usuario_hincha.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

....                                                              4 / 4 (100%)

Time: 00:04.047, Memory: 8.00 MB

OK (4 tests, 12 assertions)
```

Prueba “GL001-6 Eliminar liga”

```
<?php
use PHPUnit\Framework\TestCase;

class prueba_2_eliminar_liga extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos de la liga
        $datosLiga = [
            'id_liga' => 1 // ID de una liga existente
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/2_eliminar_liga.php', $datosLiga);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioIdLigaIncorrecto() {
        // Datos de la liga
        $datosLiga = [
            'id_liga' => 0 // ID de una liga que no existe
        ];

        // Realizar la solicitud POST al script PHP
```

```
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/2_eliminar_liga.php', $datosLiga);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_liga', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_liga']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_2_eliminar_liga.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

..                                                                  2 / 2 (100%)

Time: 00:03.430, Memory: 8.00 MB

OK (2 tests, 6 assertions)
```

Prueba “GL001-7 Eliminar categoría”

```
<?php
use PHPUnit\Framework\TestCase;

class prueba_3_eliminar_categoria extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos de la categoría
        $datosCategoria = [
            'id_categoria' => 112 // ID de una categoría existente
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/3_eliminar_categoria.php', $datosCategoria);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());
```

```php
        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioIdCategoriaIncorrecto() {
        // Datos de la categoría
        $datosCategoria = [
            'id_categoria' => 0 // ID de una categoría que no existe
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/3_eliminar_categoria.php', $datosCategoria);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_categoria', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_categoria']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_3_eliminar_categoria.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

..                                                                  2 / 2 (100%)

Time: 00:02.439, Memory: 8.00 MB

OK (2 tests, 6 assertions)
```

Prueba “GE001-8 Eliminar equipo”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_4_eliminar_equipo extends TestCase {
    public function testEnvioDatosCorrectos() {
```

```php
        // Datos del equipo a eliminar
        $datosEquipo = [
            'id_equipo' => 1, // ID de un equipo existente
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_eliminar_equipo.php', $datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioIdEquipoIncorrecto() {
        // Datos del equipo a eliminar
        $datosEquipo = [
            'id_equipo' => 0, // ID de un equipo que no existe
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/4_eliminar_equipo.php', $datosEquipo);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_equipo', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_equipo']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_4_eliminar_equipo.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

..                                                                  2 / 2 (100%)

Time: 00:02.762, Memory: 8.00 MB

OK (2 tests, 6 assertions)
```

Prueba “GE001-9 Eliminar jugador”

```php
<?php
use PHPUnit\Framework\TestCase;

class prueba_5_eliminar_jugador extends TestCase {
    public function testEnvioDatosCorrectos() {
        // Datos del jugador
        $datosJugador = [
            'CI' => 2121212121 // CI de un jugador existente
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_eliminar_jugador.php', $datosJugador);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('success', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['success']);
    }

    public function testEnvioCIIncorrecto() {
        // Datos del jugador
        $datosJugador = [
            'CI' => 0 // CI de un jugador que no existe
        ];

        // Realizar la solicitud POST al script PHP
        $respuesta = $this->solicitarPost('http://localhost:8080/SGLIGAS/Modelo/5_eliminar_jugador.php', $datosJugador);

        // Verificar si la solicitud fue exitosa
        $this->assertEquals(200, $respuesta->getStatusCode());

        // Verificar el cuerpo de la respuesta
```

```php
        $cuerpoRespuesta = json_decode($respuesta->getBody(), true);
        $this->assertArrayHasKey('no_existe_jugador', $cuerpoRespuesta);
        $this->assertTrue($cuerpoRespuesta['no_existe_jugador']);
    }

    private function solicitarPost($url, $datos) {
        $cliente = new \GuzzleHttp\Client();
        return $cliente->post($url, [
            'form_params' => $datos
        ]);
    }
}
?>
```

```
PS C:\xampp\htdocs\SGLigas_backend\Modelo\clases\test> ./vendor/bin/phpunit .\prueba_5_eliminar_jugador.php
PHPUnit 10.4.1 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.6

..                                                                  2 / 2 (100%)

Time: 00:02.539, Memory: 8.00 MB

OK (2 tests, 6 assertions)
```