ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

Implementación de un sistema web para las ligas barriales de Quito

Componente Backend para el sistema web

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN COMPUTACIÓN

CARLOS ANDRÉS VELÁSQUEZ ZALDUMBIDE

carlos.velasquez01@epn.edu.ec

DIRECTOR: EVELYN MARCELA MOSQUERA ESPINOSA

evelyn.mosquerae@epn.edu.ec

DMQ, febrero 2024

CERTIFICACIONES

Yo, Carlos Andrés Velásquez Zaldumbide declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Carlos Andrés Velásquez Zaldumbide

Certifico que el presente trabajo de integración curricular fue desarrollado por Carlos Andrés Velásquez Zaldumbide, bajo mi supervisión.

Evelyn Marcela Mosquera Espinosa
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

CARLOS ANDRÉS VELÁSQUEZ ZALDUMBIDE

EVELYN MARCELA MOSQUERA ESPINOSA

AGRADECIMIENTO

Expreso mi profundo agradecimiento a todas las personas que de alguna manera colaboraron en la culminación de esta tesis. Agradezco a mi tutora, MSc. Marcela Mosquera, por su dirección, sus recomendaciones, y su inmensa paciencia durante la realización de este trabajo. Agradezco a mi familia, especialmente a mi madre Ligia y hermanos Billy y Carolina, por su comprensión, amor incondicional y constante apoyo. También agradezco a mis amigos por estar siempre presentes y acompañarme en este viaje de aprendizaje y crecimiento personal.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Este trabajo se centró en desarrollar el componente backend para la gestión de ligas barriales de Quito. La metodología empleada fue Scrum, aplicando buenas prácticas de Programación Extrema (XP). Se realizaron cinco Sprints, con un total de 224 story points, abordando desde la arquitectura y base de datos hasta la publicación de información y gestión de usuarios.

El Sprint 0 se enfocó en configurar el ambiente de desarrollo y modelar la base de datos. El Sprint 1 se centró en registrar usuarios, iniciar sesión, y comenzar la gestión de ligas y publicación de información. En el Sprint 2, se agilizó el registro de jugadores, torneos, programación de partidos y generación de estos. El Sprint 3 buscó agilizar el proceso de registro de resultados, estadísticas de jugadores y alineaciones. En el Sprint 4, se

completaron tareas relacionadas con la actualización y eliminación de usuarios, ligas, equipos y jugadores.

Con esto el componente backend ofrece una solución integral para la gestión de ligas barriales de Quito, con características clave como gestión completa de equipos y jugadores, programación y organización de partidos, registro de resultados y generación de estadísticas, y APIs para integraciones externas. La documentación técnica y el código desarrollado están disponibles para su uso y revisión.

PALABRAS CLAVE: gestión de ligas barriales, backend, fútbol, ligas barriales de Quito, Scrum.

ABSTRACT

This work focused on developing the backend component for the management of neighborhood leagues in Quito. The methodology used was Scrum, applying good practices of Extreme Programming (XP). Five Sprints were carried out, with a total of 224 story points, addressing everything from architecture and database to the publication of information and user management.

Sprint 0 focused on setting up the development environment and modeling the database. Sprint 1 focused on registering users, logging in, and beginning league management and information publishing. In Sprint 2, player registration, tournaments, match scheduling and match generation were streamlined. Sprint 3 sought to streamline the process of recording results, player statistics and lineups. In Sprint 4, tasks related to updating and deleting users, leagues, teams, and players were completed.

With this, the backend component offers a comprehensive solution for the management of neighborhood leagues in Quito, with key features such as complete management of teams and players, scheduling and organization of matches, registration of results and generation of statistics, and APIs for external integrations. Technical documentation and developed code are available for use and review.

KEYWORDS: neighborhood league management, backend, soccer, Quito neighborhood leagues, Scrum.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

En la actualidad la gestión de las ligas barriales de Quito se lleva principalmente en documentos Excel o registros físicos. Este proceso dificulta el acceso y la actualización eficiente de la información relacionada con los equipos, jugadores y torneos. La dependencia de documentos físicos también implica el riesgo de pérdida o deterioro de los registros importantes.

Como resultado, los presidentes de los clubes, los jugadores y los hinchas carecen de una plataforma centralizada que les permita estar informados sobre los torneos, las estadísticas de los jugadores y las tablas de posiciones de manera fácil y accesible.

Para esta problemática el propósito del componente backend es centralizar la información, lo que incluirá una amplia gama de funcionalidades para abordar las necesidades específicas de la gestión de ligas barriales de Quito. Esto abarcará la gestión del torneo, que permitirá inscribir equipos, registrar ligas y almacenar la programación de partidos de manera eficiente. La gestión de equipos permitirá registrar jugadores y la información relevante de cada equipo. La gestión de partidos estará a cargo del registro de resultados y estadísticas, incluyendo tarjetas, goles, autogoles y sanciones que haya su citado en el partido.

Para asegurar la transparencia y el acceso público a la información, el componente también permitirá la extracción de datos importantes para su publicación, como la tabla de posiciones y los detalles de los partidos. Por último, se implementarán funciones de gestión de usuarios, que permitirán registrar y obtener información de los usuarios relacionados con el sistema. Finalmente, se llevarán a cabo pruebas unitarias para verificar el correcto funcionamiento del backend.

Además, se proporcionará una documentación técnica para desarrolladores que describirá la estructura, las API y las pruebas del sistema. También se ofrece una documentación lo cual asegurará una implementación exitosa y un uso sencillo del sistema, además de facilitar futuras actualizaciones y solución de problemas.

Con todo lo descrito el componente backend ayudará a resolverá la falta de una plataforma centralizada que permita a los presidentes de los clubes, jugadores y aficionados acceder fácilmente a información sobre las ligas barriales de Quito de su interés.

1.1 Objetivo general

Desarrollar el componente backend que brinde soporte al sistema web para la gestión de las ligas barriales de fútbol en Quito.

1.2 Objetivos específicos

1. Levantar los requerimientos del backend a través de reuniones con los desarrolladores del frontend.
2. Diseñar la arquitectura y base de datos para el sistema basada en los requerimientos.
3. Implementar el backend en base al diseño obtenido.
4. Implementar la lógica necesaria para la gestión de las ligas barriales de Quito.
5. Realizar pruebas unitarias.

1.3 Alcance

Para el desarrollo del backend del sistema web para la gestión de las ligas barriales de Quito se diseñará una arquitectura simple que se adapte a las necesidades actuales del sistema, se implementará la lógica de negocio necesaria para la gestión de torneos, partidos, equipos y usuarios. Se va a configurar y administrar la base de datos que almacenará los datos esenciales del sistema. Se utilizarán buenas prácticas de diseño de bases de datos para garantizar la integridad y la eficiencia en el almacenamiento y recuperación de datos. Además, se debe preparar documentación técnica que abarque aspectos de diseño, API, modelos de datos y procedimientos de prueba. Esta documentación servirá como recurso esencial para los desarrolladores y usuarios finales, asegurando una implementación exitosa y un uso efectivo del sistema.

Para todo lo mencionado se hará uso del marco de trabajo Scrum ya que nos permite presentar historias de usuario para implementar todas las funcionalidades en un proceso iterativo e incremental, adicionalmente se seguirán las buenas prácticas de Programación Extrema (XP).

1.4 Marco teórico

A continuación, se presentan conceptos fundamentales que facilitarán la comprensión del backend del sistema, el marco de trabajo y las herramientas utilizadas para su desarrollo

Liga barrial

Una liga barrial es una estructura intermedia en el sistema organizativo del deporte comunitario, que convoca y organiza actividades deportivas recreativas en el ámbito local, especialmente en barrios y parroquias. Aunque no es la organización de base ni la superestructura que toma grandes decisiones, juega un papel crucial en la coordinación y promoción del deporte a nivel comunitario [1].

Torneo

Un torneo es una serie de pruebas en las que un grupo de competidores se enfrentan entre sí para lograr la victoria, principalmente en deportes y juegos. Pueden ser individuales o grupales y tienen como objetivo obtener uno o más premios en una o varias categorías. Estas competiciones han existido desde la Edad Media como una forma de demostrar fuerza, habilidades o destrezas. Por lo general, los torneos siguen un formato eliminatorio y jerárquico, donde los competidores van siendo eliminados en cada etapa.

Los torneos deportivos tienen atributos comparables a otros tipos de competiciones donde los contendientes se congregan para competir por un premio. En numerosas disciplinas deportivas, se llevan a cabo ligas, siendo las de equipos de fútbol las más destacadas [2].

Backend

Se hace referencia a los programas y scripts que funcionan en servidores de manera invisible para los usuarios. El backend se encarga de realizar tareas relacionadas con el almacenamiento y la recuperación de datos de una base de datos, siendo esencialmente un depósito de la lógica fundamental y el funcionamiento de aplicaciones de software o sistemas de información. La función principal del sistema backend es asegurar que los datos o servicios solicitados por el sistema o la aplicación frontend se entreguen de manera eficiente a través de métodos de programación. Esta parte del sistema abarca la lógica central de la aplicación, la gestión de la base de datos, la integración de datos y la implementación de la aplicación [3].

Base de datos (BD)

Una base de datos es un conjunto organizado de datos que se almacenan electrónicamente. Los datos pueden ser de diversos tipos, incluyendo texto, números, imágenes, videos y archivos. Un sistema de manejo de bases de datos (en inglés DBMS *Database Management System*) es un software que facilita la gestión de estos datos [4].

A continuación, se presentan algunas razones que destacan la importancia de las bases de datos.

- **Escalabilidad eficiente:** Las aplicaciones de bases de datos pueden manejar volúmenes masivos de datos, que van desde millones hasta miles de millones o más. Sin una base de datos, sería imposible almacenar esta cantidad de datos digitales [4].
- **Integridad de datos:** Las bases de datos suelen incorporar reglas y condiciones que garantizan la coherencia de los datos [4].
- **Seguridad de datos:** Las bases de datos cumplen con los requisitos de privacidad y cumplimiento normativo asociados con cualquier tipo de datos. Por ejemplo, el acceso a la base de datos requiere que los usuarios inicien sesión, y diferentes usuarios pueden tener niveles de acceso distintos, como el de solo lectura [4].

Database Management System (DBMS)

Es un conjunto de elementos interconectados y una serie de programas que permiten el almacenamiento, la recuperación y la edición de la información. El DBMS tiene como objetivo principal proporcionar a los usuarios una visión abstracta de la información, lo que implica ocultar detalles sobre cómo se almacenan los datos [5]. Esta abstracción se manifiesta en tres niveles:

- **Nivel físico:** Relativo a la forma que los datos se almacenan en los medios de almacenamiento, es decir, su disposición como bloques de bytes consecutivos [5].
- **Nivel conceptual:** Donde se describen los datos almacenados y las relaciones entre ellos. La Figura 1 muestra una representación de este nivel [5].

```
CREATE TABLE liga (
    id_liga INTEGER PRIMARY KEY NOT NULL,
    nombre_liga CHAR(100) NOT NULL,
    fecha_fundacion DATE NOT NULL,
    direccion CHAR(150) NOT NULL,
    correoAdmin CHAR(50) NOT NULL,
    estado CHAR(10) NOT NULL,
    FOREIGN KEY (correoAdmin) REFERENCES usuarios(correo)
);
```

Figura 1. Representación nivel conceptual.

- **Nivel de vista:** Describe solo una parte de la base de datos, y esta descripción puede representarse de diversas maneras, ofreciendo así una representación personalizada de los datos sin revelar toda la complejidad de la base de datos [5].

Modelo relacional

El modelo relacional es en el cual los datos y las relaciones entre ellos se representan mediante tablas, cada una con varias columnas identificadas por nombres únicos. La Figura 2 muestra un ejemplo de cómo sería un modelo relacional [5].

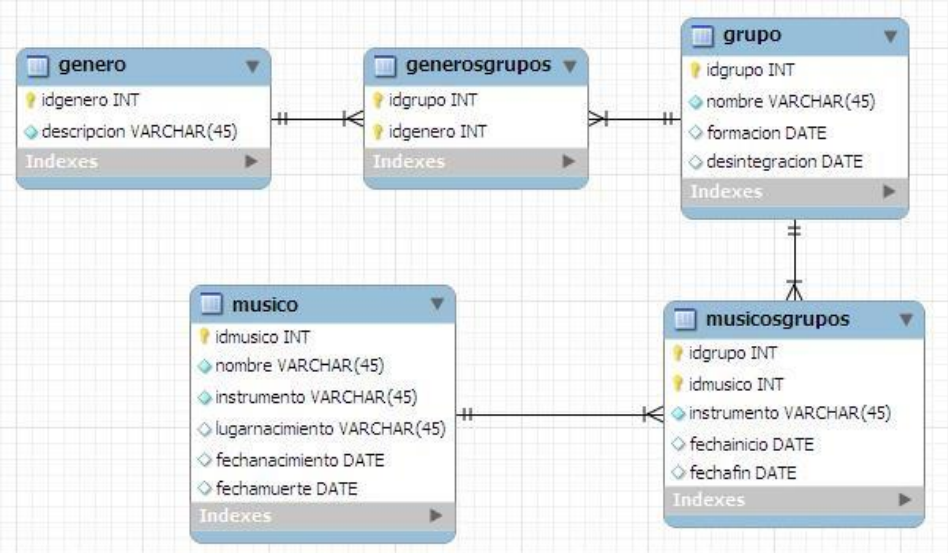

Figura 2. Ejemplo de Modelo relacional. [6]

Structure Query Language (SQL)

El lenguaje SQL es el estándar de alto nivel utilizado en sistemas de bases de datos relacionales, y su adopción como estándar es responsabilidad de la ANSI y la ISO [7].

El SQL clasifica tres categorías de declaraciones con propósitos específicos, utilizando los siguientes lenguajes:

- **Lenguaje de Definición de Datos (DDL, Data Definiton Language):** Conjunto de instrucciones en SQL destinadas a respaldar la definición y declaración de los elementos de la base de datos. Estos elementos incluyen en sí una base de datos, incluyendo las tablas, vistas, índices, procedimientos almacenados, disparadores, reglas, dominios y valores predeterminados. Estos elementos pueden ser creados, modificados o eliminados mediante comandos como CREATE, ALTER y DROP [7].
- **Lenguaje de Manipulación de Datos (DML, Data Management Language):** Conjunto de instrucciones en SQL diseñadas para gestionar los datos almacenados en las bases de datos, tanto a nivel de filas (tuplas) como de columnas (atributos). Estas incluyen comandos como INSERT, UPDATE, DELETE y SELECT [7].

- **Lenguaje de Control de Datos (DCL, Data Control Language):** Conjunto de instrucciones en SQL destinadas a supervisar las funciones administrativas ejecutadas por el DBMS, incluyendo aspectos como la atomicidad y la seguridad. Estas incluyen comandos como COMMIT TRANSACTION, ROLLBACK TRANSACTION, GRANT y REVOKE [7].

Pruebas unitarias

Las pruebas unitarias de software permiten evaluar de manera individual el correcto funcionamiento de los componentes de código, determinando si las funciones cumplen con el comportamiento esperado. Para facilitar la interacción con librerías o frameworks, es esencial contar con un componente intermedio que simplifique este proceso de manera eficaz y rápida. De esta manera, se logra ejecutar las pruebas, evaluar la funcionalidad del código y analizar los resultados obtenidos de manera efectiva [8].

Scrum

Scrum es un marco de trabajo que tiene como objetivo principal mantener un seguimiento continuo del estado actual del software en desarrollo. Una de sus principales fortalezas reside en la integración del cliente como un miembro integral del equipo de trabajo, lo que fomenta un compromiso compartido hacia la meta final. Scrum también establece claramente una serie de eventos, como la reunión diaria, la revisión del sprint y la retrospectiva, que permiten a la organización consolidar la metodología e identificar áreas de mejora en cada equipo y proceso. Además, Scrum es altamente adaptable a diversas circunstancias y promueve un ambiente de trabajo colaborativo y en equipo [9].

Roles

Scrum posee tres roles principales que son:

- **Product Owner:** la persona que toma las decisiones y es el responsable de maximizar el valor del producto y de gestionar adecuadamente el backlog del producto [10].
- **Scrum Master:** es el responsable de maximizar el valor del producto y garantizar que se sigan los principios y prácticas de Scrum [10].
- **Equipo de desarrollo:** Suele ser un equipo pequeño de unas 5-9 personas y son los encargados de convertir los requisitos del producto en incrementos potencialmente entregables en cada sprint [10].

La Figura 3 muestra el proceso Scrum.

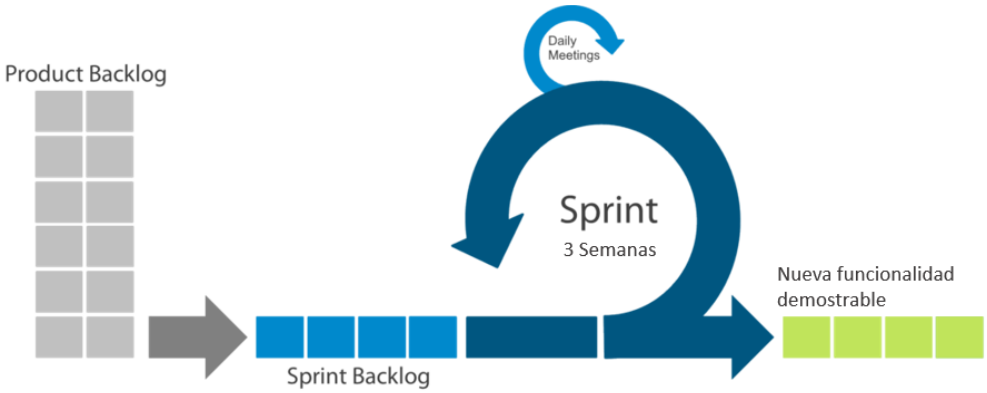

Figura 3. Proceso Scrum.

Fases de Scrum

Este proceso iterativo e incremental se agrupa en las siguientes fases [11]:

- **Fase de Inicio:** aquí se establece una visión clara del producto a desarrollar, se configura el equipo Scrum con miembros esenciales. Durante esta fase, se elaboran las épicas, las cuales son sometidas a un proceso de refinamiento y priorización para dar lugar al Product Backlog. Como parte de esta etapa inicial, también se define un plan de implementación por fases y se determina la duración de los Sprints.
- **Fase de Planificación y Estimación:** se generan Historias de Usuarios que detallan las funcionalidades. Estas historias son revisadas, aprobadas y estimadas para su posterior inclusión en los Sprints planificados. Para una mayor precisión, las Historias de Usuarios se descomponen en tareas más pequeñas, y se estima el esfuerzo necesario para llevar a cabo cada una de ellas.
- **Fase de Implementación:** durante esta fase en base al Sprint Backlog se trabaja en las tareas para crear los entregables del Sprint. Se realizan reuniones diarias de standup para actualizar el progreso y solucionar impedimentos. Se mantiene y actualiza el Product Backlog en función de las necesidades cambiantes.
- **Fase de Revisión y Retrospectiva:** En esta fase, se lleva a cabo una reunión de revisión del Sprint para mostrar los entregables del Sprint al Product Owner. Luego, el equipo Scrum se reúne para una retrospectiva del Sprint, donde se discuten lecciones aprendidas y se documentan mejoras para futuros Sprints.

- **Fase de Lanzamiento:** en la fase final, se entregan los entregables aceptados y se documenta su aceptación. Se realiza una retrospectiva del proyecto, para identificar y documentar las lecciones aprendidas y las recomendaciones para futuros proyectos.

Artefactos de Scrum

Scrum emplea los artefactos que se describen continuación [12]:

- **Product Backlog:** es una lista priorizada u ordenada de todas las características, cambios, defectos, mejoras técnicas y otros elementos que se requieren para cumplir con la visión del propietario del producto.
- **Sprint Backlog:** Lista de pendientes del Sprint (un inventario de actividades requeridas para transformar una parte del Backlog de Producto en un incremento de funcionalidad del software).
- **Incremento:** se define como la suma de todos los elementos del Product Backlog que han sido completados durante un Sprint y los Sprints anteriores. Al finalizar un Sprint, este nuevo incremento debe ser potencialmente entregable, o en otras palabras, debe estar "hecho", lo que significa que está en condiciones de ser utilizado, independientemente de si el Product Owner decide lanzarlo o no.

Actividades de Scrum

Estas actividades están diseñadas para mantener al equipo enfocado, organizado y alineado con los objetivos del proyecto, asegurando una entrega eficiente y efectiva del producto. En Scrum, se hablan de cuatro reuniones que se llevan a cabo de manera iterativa en cada Sprint [12]:

- **Sprint Planning:** es el primer paso que se lleva a cabo al iniciar cada ciclo de trabajo en Scrum. Durante esta reunión, participan el Product Owner, el Scrum Master y el equipo de desarrollo. Durante esta reunión, el Product Owner presenta las historias de usuario prioritarias del Backlog de Producto al equipo. Posteriormente, el equipo discute y negocia el alcance de estas historias, dividiéndolas en tareas con estimaciones de esfuerzo. Estas tareas se agregan al Backlog del Sprint y se asignan a los miembros del equipo.
- **Daily Scrum:** son breves "conversaciones" de 5 a 15 minutos que el Scrum Master lleva a cabo al inicio de cada día con cada miembro del equipo. Durante estas conversaciones, el Scrum Master se pone al día sobre lo que cada miembro del

equipo ha logrado en el día anterior y lo que planea hacer en el día actual. Además, el Scrum Master busca identificar cualquier impedimento que esté surgiendo para resolverlo rápidamente y permitir que el equipo continúe su trabajo sin contratiempos.

- **Sprint Review:** el equipo presenta al Product Owner las funcionalidades que han desarrollado. Explican y demuestran estas funcionalidades para que el Product Owner y cualquier otra audiencia puedan experimentarlas. El Product Owner puede sugerir mejoras, aprobarlas por completo o rechazarlas si considera que no se ha cumplido el objetivo. La reunión de revisión se realiza en el último día del Sprint y su duración no está predeterminada. En la práctica, se emplea el tiempo que sea necesario.

- **Sprint Retrospective:** el equipo de desarrollo, junto con el Scrum Master y opcionalmente el Product Owner, analiza lo que se ha hecho y sus resultados. El objetivo de esta retrospectiva es identificar medidas concretas para mejorar los resultados futuros. Se considera una oportunidad para aprender de los éxitos y errores, y mejorar en lo que sea posible. La retrospectiva en Scrum se ve como una oportunidad para el aprendizaje y la mejora continua.

XP (eXtrem programing)

La metodología conocida como eXtreme Programming (Programación Extrema), o XP, surgió en 1996 gracias a la colaboración de Kent Beck, Ward Cunningham y Ron Jeffries. A diferencia de Scrum, XP se centra exclusivamente en un conjunto específico de prácticas técnicas que, cuando se aplican de manera conjunta, buscan maximizar los resultados positivos en un proyecto de desarrollo de software. XP propone 12 prácticas técnicas que son simples y fáciles de entender, que, cuando se implementan de manera conjunta, garantizan un mejor resultado para el proyecto [12].

A continuación, se describen estas doce prácticas:

- **Cliente in-situ (on-site customer):** se enfatiza la presencia activa del cliente en el desarrollo del proyecto. Esto implica que el cliente debe estar disponible y dispuesto a participar activamente en todas las etapas del proceso.

- **Semana de 40 horas (40 hour week):** se promueve una semana laboral de 40 horas como estándar. Esta práctica se fundamenta en la idea de salvaguardar la calidad del equipo, reconociendo los límites humanos en términos de esfuerzo y dedicación.

- **Metáfora (metaphor):** Para mejorar la comunicación entre técnicos y usuarios y evitar problemas comunes, XP sugiere el uso de metáforas. Estas metáforas sirven como puntos de referencia que relacionan conceptos técnicos con situaciones de la vida cotidiana, facilitando así la comprensión.
- **Diseño simple (simple design):** se basa en el principio técnico conocido como KISS, cuyas siglas en inglés significan "Keep it simple, stupid!" (¡Manténlo simple, estúpido! En esencia, el principio KISS implica mantener un diseño sencillo, estandarizado, fácil de entender y de refactorizar.
- **Refactorización (refactoring):** una técnica que implica modificar el código fuente de un software sin alterar su comportamiento externo. La refactorización se utiliza para mejorar la estructura y la legibilidad del código, eliminando redundancias y aumentando su cohesión.
- **Programación de a pares (pair programming):** en la programación de a pares, dos desarrolladores trabajan juntos en una misma computadora, intercambiando roles de manera fluida durante las sesiones de Pair Programming. Los roles pueden variar ampliamente, lo que permite una gran libertad creativa.
- **Entregas cortas (short releases):** esta práctica implica realizar entregas de software en períodos de tiempo cortos, agregando pequeñas funcionalidades en cada iteración.
- **Testing:** se emplean tres tipos de pruebas Test Unitarios, de Aceptación y de Integración.
- **Código estándar (coding standards):** establecer estándares de escritura del código fuente es fundamental en el proceso de programación. Estos estándares permiten que el código sea más legible, limpio y facilitan la comprensión por parte de otros programadores.
- **Propiedad colectiva (collective ownership):** para XP no existe un único programador "dueño" de un código o una funcionalidad específica en el equipo. En cambio, el código pertenece colectivamente a todos los miembros del equipo.
- **Integración continua (continuous integración):** esta práctica propone que todo el código desarrollado por los miembros del equipo se integre en un repositorio común diariamente.

- **Juego de planificación (planning game):** consiste en un proceso colaborativo al inicio de cada iteración donde el cliente presenta una lista de funcionalidades deseada en formato de "Historias de Usuario", el equipo de desarrollo estima el esfuerzo necesario para cada historia y el cliente decide qué historias se desarrollarán y en qué orden, basándose en estas estimaciones y las prioridades del proyecto, asegurando así una planificación efectiva y transparente para alcanzar los objetivos de la iteración.

Herramientas de desarrollo

Tabla 1. Herramientas utilizadas para el desarrollo del backend

NOMBRE	DESCRIPCIÓN	VERSIÓN	APLICACIÓN
Lenguaje de programación utilizado			
PHP 	Es un lenguaje de programación de código abierto que es ampliamente utilizado, especialmente en el ámbito del desarrollo web, y que puede ser integrado dentro de HTML. [13].	8.1.6	• Establecer conexión con la base de datos y modificar los datos de este. • Facilitar la integración con el frontend al proporcionar la lógica necesaria para interactuar con él mismo.
Herramientas de desarrollo			
Visual Studio Code 	Se trata de un editor de código liviano pero poderoso, diseñado para funcionar en su computadora y que es compatible con sistemas operativos como Windows, macOS y Linux. [14].	1.86.0	• Crear el código fuente para el lado del servido, además del código para las pruebas unitarias.
XAMPP 	Es el entorno más comúnmente utilizado para desarrollar con PHP, que incluye una distribución gratuita y sencilla de	3.3.0	• Ejecutar la aplicación PHP en la propia máquina sin necesidad de configurar servidores adicionales.

	Apache que contiene MariaDB, PHP y Perl. [15].		
MySQL Workbench 	MySQL Workbench es una herramienta visual completa, diseñada para arquitectos, desarrolladores y administradores de bases de datos. [16].	8.0.33	• Administrar la base de datos MySQL alojada en la nube.
GitHub 	Es una plataforma en la nube que ofrece servicios de control de versiones (VCS) a través de Git. Este sistema permite que los desarrolladores trabajen juntos en proyectos compartidos, manteniendo un registro detallado de los cambios y el progreso del proyecto. [17].	Online	• Gestionar las versiones de manera efectiva. • Almacenar el proyecto de forma segura.
GitHub Desktop 	Es una aplicación gratuita de código abierto que le ayuda a trabajar con código alojado en GitHub u otros servicios de alojamiento Git [18].	3.3.8	• Simplificar la gestión de versiones del proyecto. • Proporcionar una interfaz gráfica que hace más accesible la utilización de los comandos de GIT.
Azure DevOps 	Reúne personas, procesos y tecnología, automatizando la entrega de software para brindar valor continuo a sus usuarios. DevOps automatiza y acelera la entrega de software. Hace que su proceso y sus productos sean más confiables [19].	Online	• Estructurar las historias de usuarios y las tareas a realizar. • Evaluar el estado actual del proceso de cada historia.

PHPUnit 	Es un marco de pruebas para PHP orientado al programador. Es una instancia de la arquitectura xUnit para marcos de pruebas unitarias [20].	10.4.1	• Ejecutar las pruebas unitarias para las diferentes clases.
Clever Cloud 	Proporciona un servicio de plataforma como servicio, con sede en Europa. PaaS ayuda a los equipos de desarrollo a poner en producción aplicaciones y servicios digitales en una infraestructura confiable, con escalabilidad automática y precios transparentes [21].	Online	• Almacenar la base de datos en un entorno de nube.
Doxygen 	Es una herramienta generadora de documentación ampliamente utilizada en el desarrollo de software. Automatiza la generación de documentación a partir de comentarios del código fuente, analizando información sobre clases, funciones y variables para producir resultados en formatos como HTML y PDF [22].	1.9.8	• Crear documentación a partir del código fuente de programas desarrollados en PHP.
Diseño de base de datos			
ERD Plus 	Es una herramienta en línea para crear diagramas entidad-relación (ERD. ERDPlus permite visualizar los diagramas de las bases de datos y crear las bases de datos de forma rápida y eficiente [23].	Online	• Desarrollar el modelo entidad-relación que refleje la lógica del negocio.

2 METODOLOGÍA

Este trabajo se centra en desarrollar el backend del sistema para la gestión de las ligas barriales de Quito. La metodología empleada en este estudio se basa en el marco de trabajo ágil conocido como Scrum, adicionalmente se aplicaron buenas prácticas de XP. Scrum posee fases que son iterativas y se aplican a lo largo del desarrollo del componente backend, permitiendo la adaptación constante y la mejora continua según las necesidades y resultados del proyecto. A lo largo de esta sección, se detallará el proceso de implementación de Scrum.

2.1 Fase de inicio

Para el desarrollo del componente backend se creará una plataforma centralizada que permita a las ligas barriales gestionar eficientemente sus equipos, jugadores, partidos y resultados. Además, el backend se integrará con otras partes del sistema, como el frontend y aplicaciones móviles, para proporcionar una experiencia de usuario completa.

Durante este período, se lleva a cabo la configuración del equipo Scrum, los roles de cada miembro se encuentra en la Tabla 2.

Tabla 2. Roles Scrum en el desarrollo del sistema

ROL SCRUM	**RESPONSABLE**
Product Owner	MSc. Marcela Mosquera
Scrum Master	MSc. Marcela Mosquera
Equipo de desarrollo	Wilmer Jiménez, Cristhian Lemache, Carlos Velásquez

A partir de la colaboración del equipo se define y refina las funcionalidades del sistema, llamadas épicas.

Se han identificado como épicas:

- Gestión de Torneos (GT001)
- Gestión de Partidos (GP001)
- Gestión de Equipos (GE001)
- Publicar información (PI001)
- Gestión de Usuarios (GU001)
- Gestión de Ligas (GL001)

Este proceso prioriza las épicas según su valor y viabilidad técnica, formando así el Product Backlog.

Product Backlog

En la Tabla 3 se presenta el producto backlog, el cual se ha generado a partir de las épicas identificadas previamente.

Tabla 3. Product backlog

Épica	Código	Historia de Usuario
Gestión de usuarios (GU001)	GU001-1	Registrar usuario
	GU001-2	Iniciar de sesión
	GU001-3	Recuperar contraseña
	GU001-4	Actualizar usuario
	GU001-5	Eliminar usuario
Gestión de Ligas (GL001)	GL001-1	Registrar liga
	GL001-2	Registrar categoría
	GL001-3	Mostrar liga
	GL001-4	Mostrar categorías
	GL001-5	Actualizar categoría
	GL001-6	Eliminar liga
	GL001-7	Eliminar categoría
Gestión de Equipos (GE001)	GE001-1	Mostrar equipos de liga
	GE001-2	Validar número de equipos
	GE001-3	Registrar equipo
	GE001-4	Registrar jugador
	GE001-5	Mostrar jugadores
	GE001-6	Actualizar equipo
	GE001-7	Actualizar jugador
	GE001-8	Eliminar equipo
	GE001-9	Eliminar jugador
Gestión de torneos (GT001)	GT001-1	Registrar torneos
	GT001-2	Generar partidos
	GT001-3	Mostrar torneos
Gestión de partidos (GP001)	GP001-1	Programar partidos
	GP001-2	Registrar resultados
	GP001-3	Comprobar registro de estadistícas de jugadores
	GP001-4	Registrar estadísticas jugadores: rojas, amarillas, goles, goles recibidos en el caso de porteros, autogoles y partidos jugados.
	GP001-5	Registrar alineación
	GP001-6	Registrar informe de sanciones

	GP001-7	Registrar tribunal de sanciones
	GP001-8	Mostrar acta de juego
Publicar información (PI001)	PI001-1	Mostrar todas las ligas inscritas
	PI001-2	Mostrar equipos de categoría
	PI001-3	Mostrar posiciones
	PI001-4	Mostrar partidos
	PI001-5	Mostrar equipo
	PI001-6	Mostrar partidos de cada equipo
	PI001-7	Mostrar estadísticas de jugadores de equipo
	PI001-8	Mostrar información de jugador
	PI001-9	Mostrar alineación

Además, en esta etapa inicial, se establece un plan de implementación por fases y la duración de los Sprints, que será de 3 semanas.

2.2 Fase de Planificación y Estimación

Durante esta etapa, se generan las Historias de Usuario que describen las funcionalidades específicas que se deben implementar en el componente backend del sistema. Estas historias son revisadas, aprobadas y estimadas utilizando poker planning para su posterior inclusión en los Sprints planificados. A continuación, se muestra una historia de usuario, las restantes se encuentran en el apartado Anexo I.

Tabla 4. 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.

Estimación de Historias de Usuario

A continuación, se presentan las prioridades asignadas a cada una de las historias de usuario identificadas en el Producto Backlog.

Tabla 5. Prioridades del Product Backlog

Código	Historia de Usuario	Prioridad
GU001-1	Registrar usuario	ALTA
GU001-2	Iniciar de sesión	ALTA
GU001-3	Recuperar contraseña	MEDIA
GU001-4	Actualizar usuario	MEDIA
GU001-5	Eliminar usuario	BAJA
GL001-1	Registrar liga	ALTA

GL001-2	Registrar categoría	ALTA
GL001-3	Mostrar liga	ALTA
GL001-4	Mostrar categorías	ALTA
GL001-5	Actualizar categoría	MEDIA
GL001-6	Eliminar liga	BAJA
GL001-7	Eliminar categoría	BAJA
GE001-1	Mostrar equipos de liga	ALTA
GE001-2	Validar número de equipos	ALTA
GE001-3	Registrar equipo	ALTA
GE001-4	Registrar jugador	ALTA
GE001-5	Mostrar jugadores	MEDIA
GE001-6	Actualizar equipo	MEDIA
GE001-7	Actualizar jugador	MEDIA
GE001-8	Eliminar equipo	BAJA
GE001-9	Eliminar jugador	BAJA
GT001-1	Registrar torneos	ALTA
GT001-2	Generar partidos	ALTA
GT001-3	Mostrar torneos	ALTA
GP001-1	Programar partidos	ALTA
GP001-2	Registrar resultados	ALTA
GP001-3	Comprobar registro de estadisticas de jugadores	ALTA
GP001-4	Registrar estadísticas jugadores: rojas, amarillas, goles, goles recibidos en el caso de porteros, autogoles y partidos jugados.	ALTA
GP001-5	Registrar alineación	MEDIA
GP001-6	Registrar informe de sanciones	MEDIA
GP001-7	Registrar tribunal de sanciones	MEDIA
GP001-8	Mostrar acta de juego	MEDIA
PI001-1	Mostrar todas las ligas inscritas	ALTA
PI001-2	Mostrar equipos de categoría	ALTA
PI001-3	Mostrar posiciones	ALTA
PI001-4	Mostrar partidos	ALTA
PI001-5	Mostrar equipo	ALTA
PI001-6	Mostrar partidos de cada equipo	ALTA
PI001-7	Mostrar estadísticas de jugadores de equipo	ALTA
PI001-8	Mostrar información de jugador	ALTA
PI001-9	Mostrar alineación	ALTA

Product Backlog ordenado

Después de realizar las estimaciones se ordena el producto backlog como se muestra en la siguiente tabla

Tabla 6. Product Backlog ordenado

Código	Historia de Usuario	Prioridad
GU001-1	Registrar usuario	ALTA
GU001-2	Iniciar de sesión	ALTA
GL001-1	Registrar liga	ALTA
GL001-2	Registrar categoría	ALTA
GL001-3	Mostrar liga	ALTA
GL001-4	Mostrar categorías	ALTA
GE001-1	Mostrar equipos de liga	ALTA
GE001-2	Validar número de equipos	ALTA
GE001-3	Registrar equipo	ALTA
PI001-5	Mostrar equipo	ALTA
GE001-4	Registrar jugador	ALTA
GT001-1	Registrar torneos	ALTA
GT001-2	Generar partidos	ALTA
GT001-3	Mostrar torneos	ALTA
GP001-1	Programar partidos	ALTA
PI001-1	Mostrar todas las ligas inscritas	ALTA
PI001-2	Mostrar equipos de categoría	ALTA
GE001-5	Mostrar jugadores	MEDIA
PI001-3	Mostrar posiciones	ALTA
GP001-2	Registrar resultados	ALTA
GP001-3	Comprobar registro de estadísticas de jugadores	ALTA
GP001-4	Registrar estadísticas jugadores: rojas, amarillas, goles, goles recibidos en el caso de porteros, autogoles y partidos jugados.	ALTA
PI001-4	Mostrar partidos	ALTA
PI001-6	Mostrar partidos de cada equipo	ALTA
PI001-7	Mostrar estadísticas de jugadores de equipo	ALTA
PI001-8	Mostrar información de jugador	ALTA
GP001-5	Registrar alineación	MEDIA
PI001-9	Mostrar alineación	ALTA
GP001-6	Registrar informe de sanciones	MEDIA
GP001-7	Registrar tribunal de sanciones	MEDIA
GP001-8	Mostrar acta de juego	MEDIA
GU001-3	Recuperar contraseña	MEDIA
GU001-4	Actualizar usuario	MEDIA
GL001-5	Actualizar categoría	MEDIA
GE001-6	Actualizar equipo	MEDIA
GE001-7	Actualizar jugador	MEDIA
GU001-5	Eliminar usuario	BAJA
GL001-6	Eliminar liga	BAJA

GL001-7	Eliminar categoría	BAJA
GE001-8	Eliminar equipo	BAJA
GE001-9	Eliminar jugador	BAJA

Descomposición de Historias de Usuario en Tareas

Para garantizar una estimación precisa del esfuerzo requerido para cada historia, estas son desglosadas en tareas más pequeñas. Una vez que las tareas están definidas, el equipo de desarrollo estima el esfuerzo necesario para completar cada una de ellas, utilizando la técnica de estimación en puntos de historia o la estimación en horas. La siguiente tabla muestra las tareas que deben ser realizadas por el componente backend de cinco historias de usuario. El resto de las tareas para cada historia de usuario se pueden observar en el Anexo II.

Tabla 7. Historias de Usuario

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

2.3 Sprint 0

Sprint Planning

El propósito del Sprint 0 es preparar el ambiente de desarrollo y modelar la base de datos. En la Tabla 8 se detallan las tareas que fueron consideradas para el Sprint Backlog 0. Para este Sprint se estimaron un total de 62 story-points.

Tabla 8. Sprint Backlog 0

Nro.	Tarea	Estimación
1	Instalación de Visual Studio Code	3
2	Instalación de PHP en Visual Studio Code	3
3	Instalación de XAMPP	5
4	Instalación de workbeanch	3
5	Crear un repositorio en GitHub	3
6	Instalar GitHub Desktop	3
7	Creación de ramas para el control de versiones	5

8	Crear una cuenta en Azure DevOps	3
9	Crear cuenta en ERD Plus	2
10	Realizar el modelo entidad relación	8
11	Crear cuenta en Clever Cloud	3
12	Establecer una conexión de PHP con la base de datos en la nube.	5
13	Instalar composer	3
14	Instalar GuzzleHttp	3
15	Instalar PHPUnit	5
16	Instalar Doxygen	5

Implementación

- **Ambiente de desarrollo**

El ambiente de desarrollo presentado en la Tabla 1 se configuró en una laptop con sistema operativo Windows 10. Las especificaciones de hardware incluyen un procesador Intel(R) Core(TM) i5-8265U, 8,00 GB de memoria RAM y un disco duro sólido de 117 GB. Además, para la cuenta en Clever Cloud, se optó por el plan DEV para MySQL, el cual es gratuito y permite un máximo de 10MiB de tamaño de disco y hasta 5 conexiones simultáneas.

- **Ramas para el control de versiones**

Para este proyecto, adoptamos una estrategia de ramificación donde se estableció un repositorio principal en Git, identificado como la rama 'main'. En un siguiente paso, se creó una rama denominada 'RamaCarlos' específicamente para este componente, permitiendo así un desarrollo separado y enfocado en el proyecto. El enlace al repositorio correspondiente está disponible en el Anexo III.

- **Modelo entidad relación**

El modelo entidad relación obtenido se puede observar en la Figura 4.

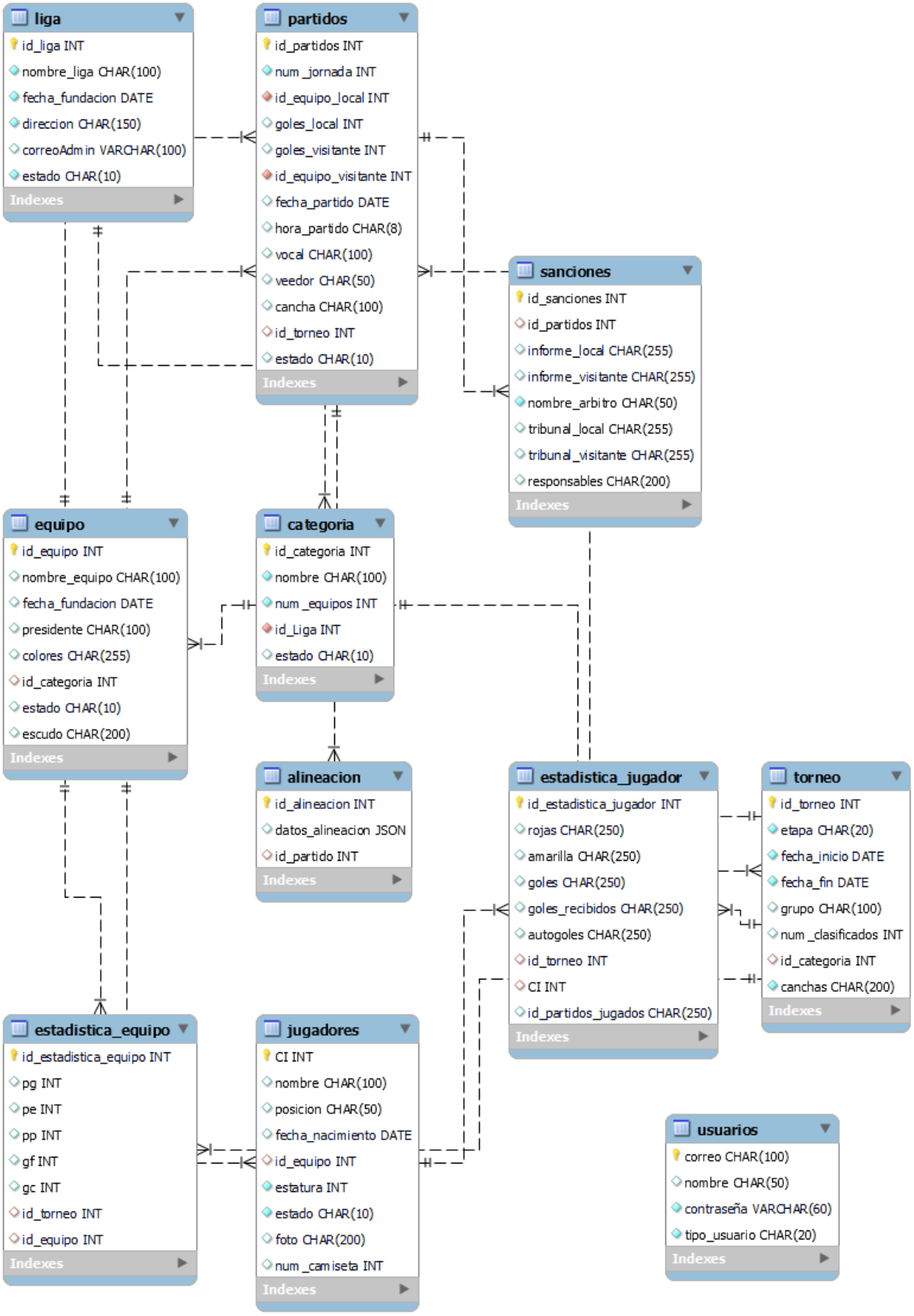
liga
id_liga INT
nombre_liga CHAR(100)
fecha_fundacion DATE
direccion CHAR(150)
correoAdmin VARCHAR(100)
estado CHAR(10)
Indexes
partidos
id_partidos INT
num_jornada INT
id_equipo_local INT
goles_local INT
goles_visitante INT
id_equipo_visitante INT
fecha_partido DATE
hora_partido CHAR(8)
vocal CHAR(100)
veedor CHAR(50)
cancha CHAR(100)
id_torneo INT
estado CHAR(10)
Indexes
sanciones
id_sanciones INT
id_partidos INT
informe_local CHAR(255)
informe_visitante CHAR(255)
nombre_arbitro CHAR(50)
tribunal_local CHAR(255)
tribunal_visitante CHAR(255)
responsables CHAR(200)
Indexes
equipo
id_equipo INT
nombre_equipo CHAR(100)
fecha_fundacion DATE
presidente CHAR(100)
colores CHAR(255)
id_categoria INT
estado CHAR(10)
escudo CHAR(200)
Indexes
categoria
id_categoria INT
nombre CHAR(100)
num_equipos INT
id_Liga INT
estado CHAR(10)
Indexes
alineacion
id_alineacion INT
datos_alineacion JSON
id_partido INT
Indexes
estadistica_jugador
id_estadistica_jugador INT
rojas CHAR(250)
amarilla CHAR(250)
goles CHAR(250)
goles_recibidos CHAR(250)
autogoles CHAR(250)
id_torneo INT
CI INT
id_partidos_jugados CHAR(250)
Indexes
torneo
id_torneo INT
etapa CHAR(20)
fecha_inicio DATE
fecha_fin DATE
grupo CHAR(100)
num_clasificados INT
id_categoria INT
canchas CHAR(200)
Indexes
estadistica_equipo
id_estadistica_equipo INT
pg INT
pe INT
pp INT
gf INT
gc INT
id_torneo INT
id_equipo INT
Indexes
jugadores
CI INT
nombre CHAR(100)
posicion CHAR(50)
fecha_nacimiento DATE
id_equipo INT
estatura INT
estado CHAR(10)
foto CHAR(200)
num_camiseta INT
Indexes
usuarios
correo CHAR(100)
nombre CHAR(50)
contraseña VARCHAR(60)
tipo_usuario CHAR(20)
Indexes

Figura 4. Diagrama entidad relación.

Sprint review

En la Tabla 9 se detallan las tareas que fueron realizadas satisfactoriamente dentro del plazo establecido.

Tabla 9. Criterios de aceptación Sprint 0

Nro.	Tarea	Estimación	Cumplimiento
1	Instalación de Visual Studio Code	3	✓
2	Instalación de PHP en Visual Studio Code	3	✓
3	Instalación de XAMPP	5	✓
4	Instalación de workbeanch	3	✓
5	Crear un repositorio en GitHub	3	✓
6	Instalar GitHub Desktop	3	✓
7	Creación de ramas para el control de versiones	5	✓
8	Crear una cuenta en Azure DevOps	3	✓
9	Crear cuenta en ERD Plus	2	✓
10	Realizar el modelo entidad relación	8	✓
11	Crear cuenta en Clever Cloud	3	✓
12	Establecer una conexión de PHP con la base de datos en la nube.	5	✓
13	Instalar composer	3	✓
14	Instalar GuzzleHttp	3	✓
15	Instalar PHPUnit	5	✓
16	Instalar Doxygen	5	✓

Sprint Retrospective

- **¿Qué se hizo bien?** Se implementó el ambiente de desarrollo correctamente. Para trabajar de manera colaborativa se creó un repositorio en GitHub para almacenar el proyecto y se definió una estrategia de ramificación para facilitar el manejo de versiones.
- **¿Qué se hizo mal?** Los nombres de las ramas deberían haber sido designados de manera que reflejaran el nombre del componente en desarrollo. Además, el diseño del modelo relacional no fue el más apropiado, ya que no incluía ciertos datos relevantes, como los informes y los tribunales de sanciones relacionados con la gestión de partidos.

- **¿Qué se puede mejorar?** La nomenclatura de la rama de desarrollo debería reflejar claramente el nombre del componente en desarrollo. En cuanto al diseño del modelo relacional, se requiere una modificación para incluir aspectos relacionados con sanciones, como informes y tribunales de sanciones. Además, es importante realizar un breve repaso del uso de Git para garantizar una integración exitosa con la rama principal ('main').

2.4 SPRINT 1

Sprint Planning

El propósito del Sprint 1 es permitir registrar usuarios e iniciar sesión en la aplicación además de iniciar con la gestión de Ligas y publicar información.

En la Tabla 10 se detallan las tareas de ciertas historias de usuario que fueron consideradas para el Sprint Backlog 1, el resto se encuentra en el ANEXO IV. Para este Sprint se estimaron un total de 41 story-points.

Tabla 10. Sprint Backlog 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

Implementación

- **Historia de Usuario GU001-1**

A continuación, se presenta el código desarrollado para la implementación de la historia de usuario “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();
?>
```

El código relacionado con las otras historias de usuario consideradas para este sprint se puede encontrar en el Anexo V.

Se generó la documentación técnica utilizando la herramienta Doxygen. La aplicación se ejecutó, mostrando una ventana similar a la que se presenta a continuación:

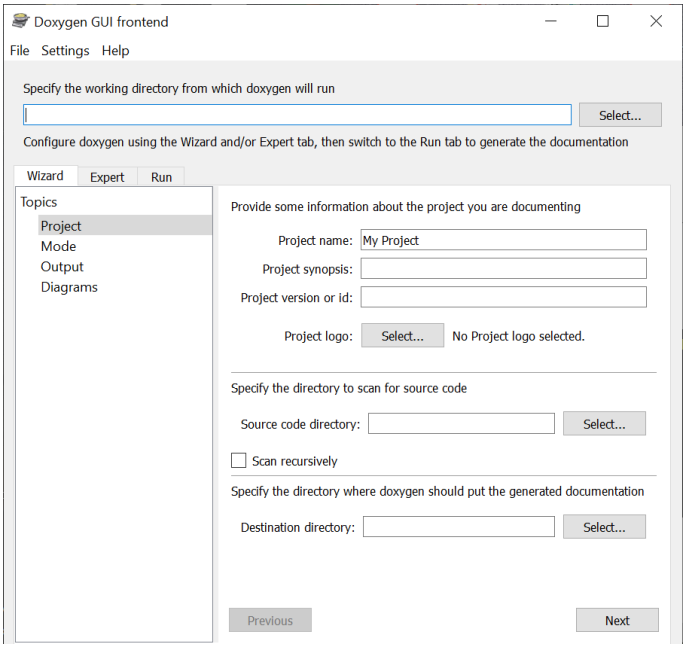

Figura 5. Ventana principal de Doxygen

- Seleccionar una carpeta para almacenar el código. En este caso, se guardó en la ruta C:/Users/BILLY/Desktop/tesis/DOCUMENTACION_DOXY.
- Nombrar el proyecto como "SGLigas", incluir una descripción y especificar la versión del proyecto (en este caso, la versión 1).
- Seleccionar la ruta donde se encuentra el código del proyecto.
- Elegir la carpeta donde se guardará la documentación generada. La siguiente figura muestra los diferentes campos configurados:

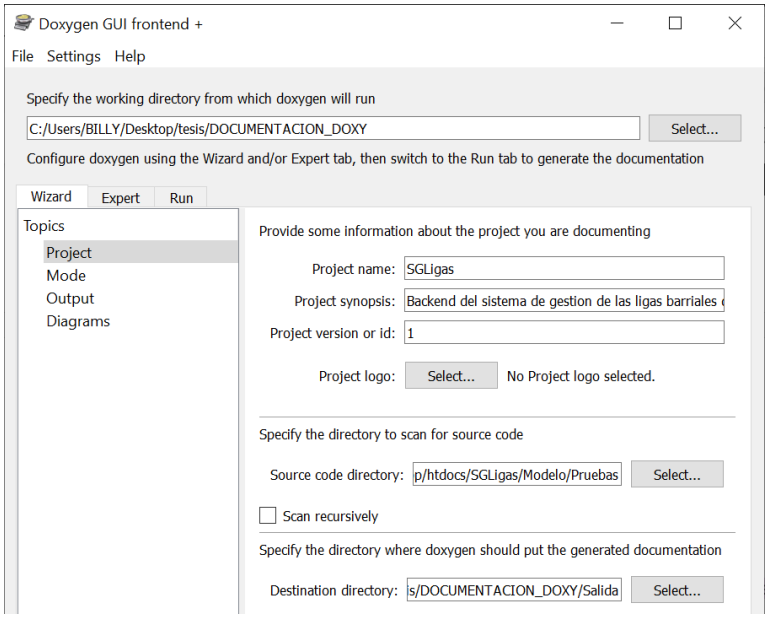

Figura 6. Campos configurados en Doxygen

- Hacer clic en "Siguiente" ("Next") y seleccionar la documentación para todo el código y el tipo de lenguaje (en este caso, PHP).
- La generación de la documentación será en formato HTML, como se muestra en la siguiente figura:

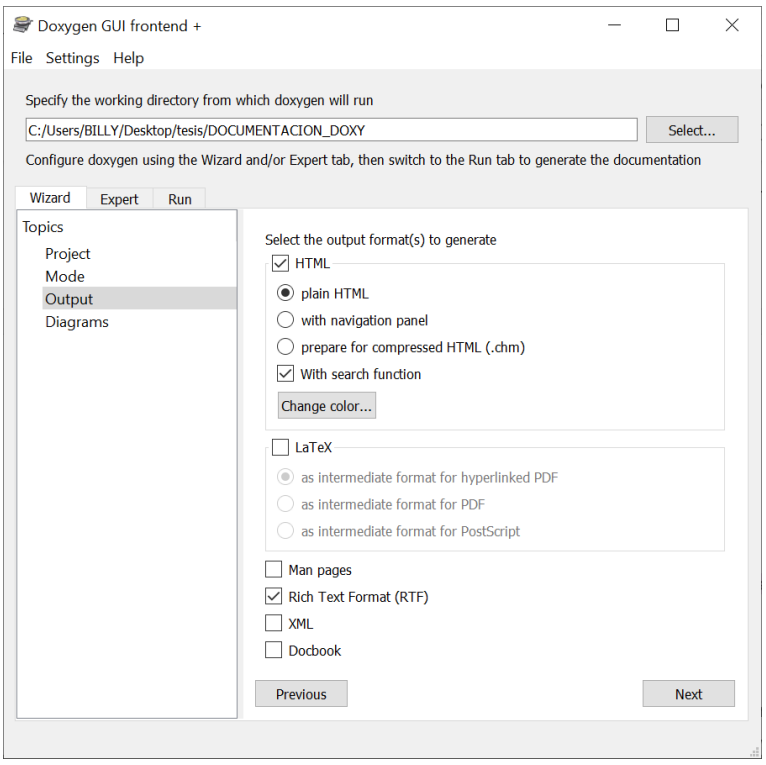

Figura 7. Escoger el formato de presentación.

- En la siguiente ventana, hacer clic en "Siguiente" ("Next") y luego seleccionar el idioma de salida. En la casilla de entrada en "Archivo: patrones", elegir PHP.
- Posteriormente, en la pestaña "Run", hacer clic en "Run doxygen" para generar la documentación.
- Finalmente, como se puede observar en la siguiente imagen, se muestra la documentación generada para la historia de usuario "GU001-1 Registrar Usuario":

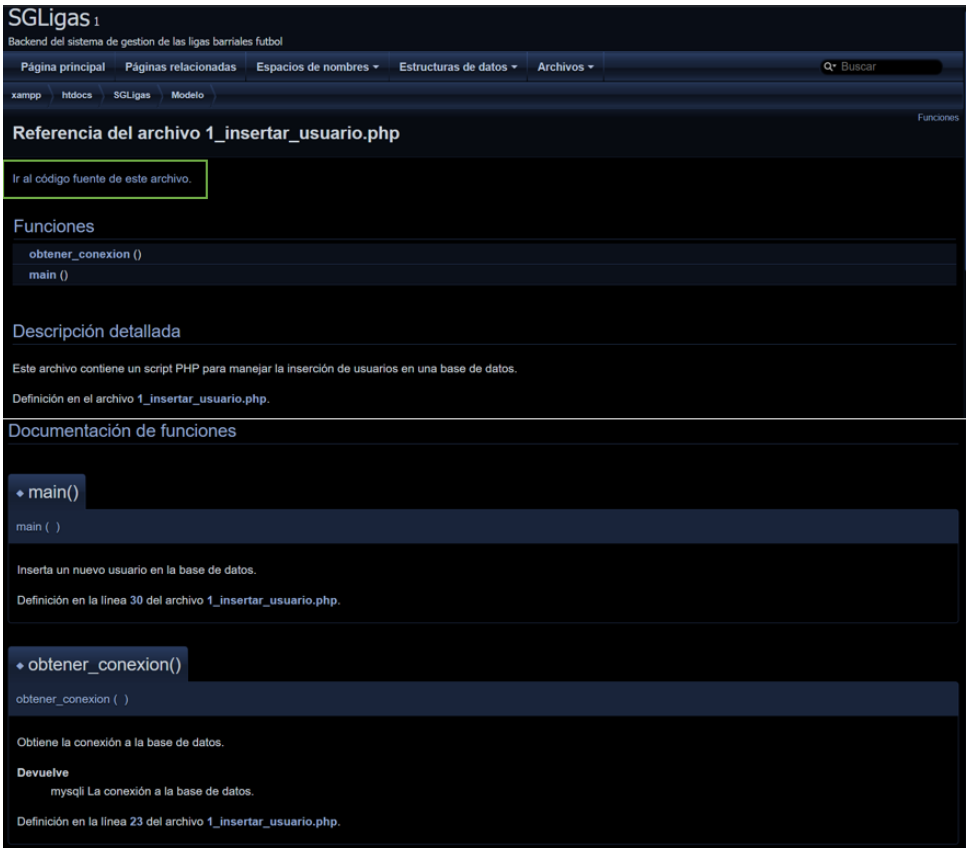

Figura 8. Documentación del archivo 1_insertar_usuario.php

En la Figura, se destaca un recuadro de color verde el cual permite visualizar el código fuente del archivo. En este aspecto, se puede observar una porción del código documentado en la siguiente Figura.

Figura 9. Porción del código del archivo 1_insertar_usuario.php documentado

Es fundamental incluir comentarios en cada archivo que faciliten a Doxygen generar una documentación precisa. Por esta razón, en el código mostrado para la historia de usuario "GU001-1 Registrar Usuario", se han añadido comentarios específicos para Doxygen.

El resto de la documentación está disponible en el enlace almacenado en el repositorio correspondiente, el cual se puede encontrar en el Anexo IX.

Sprint review

A continuación, en la Tabla 11 se detallan las tareas que se completaron satisfactoriamente dentro del plazo establecido para las historias de usuario GU001-1 y GU001-2 junto con los criterios de aceptación correspondientes a cada una. Las restantes se encuentran en el Anexo X.

Tabla 11. Criterios Aceptación Sprint 1

Código	Historia de Usuario	Criterios de aceptación	Cumplimiento
GU001-1	Registrar usuario	Envió de datos correctos	✓
		Envió de datos vacíos	✓
		Envió de correo registrado	✓
GU001-2	Iniciar de sesión	Envió de datos correctos	✓
		Envió de datos vacíos	✓
		Envió de correo electrónico no registrado	✓

		Envió de correo electrónico que no es presidente	✓
		Envió de correo electrónico que tiene liga registrada	✓
		Envió de nombre de liga existente	✓

Sprint Retrospective

- **¿Qué se hizo bien?** Las diversas historias de usuarios se completaron satisfactoriamente, lo que sugiere una implementación efectiva con los requisitos establecidos.
- **¿Qué se hizo mal?** Los métodos que interactúan con la base de datos fueron escritos en el mismo archivo que contenía la recepción de datos y validación. Almacenar el escudo del equipo en formato base64 en la base de datos resultó en consultas muy lentas.
- **¿Qué se puede mejorar?** La estructura de los métodos que interactúan con la base de datos debería reorganizarse; es decir, crear clases y en los archivos que utilicen estos métodos, instanciar objetos de esta clase para acceder a los archivos de manera más eficiente. Es necesario modificar la forma de almacenar el escudo en la base de datos para evitar demoras al extraer la información del equipo.

2.5 SPRINT 2

Sprint Planning

El propósito del Sprint 2 es facilitar el registro de jugadores, torneos y la programación de partidos, así como la generación de estos últimos con sus respectivos horarios, fecha y canchas. Además, se espera que se muestre la tabla de posiciones, que inicialmente estará vacía ya que aún no se han registrado resultados de los diferentes partidos.

En la Tabla 12 se detallan las tareas de ciertas historias de usuario que fueron consideradas para el Sprint Backlog 2, el resto se encuentra en el ANEXO IV. Para este Sprint se estimaron un total de 46 story-points.

Tabla 12. Sprint Backlog 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

Implementación

- **Historia de Usuario GT001-1**

A continuación, se presenta el código desarrollado para la implementación de la historia de usuario “GT001-1 Registrar torneos”.

```php
<?php
/**
 * @file
 * Este archivo contiene un script PHP para manejar la inserción de
torneos en una 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 string $grupo Los id de los torneos para cada grupo separados
por coma ejemplo (1,2) es opcional si y solo si la etapa es de grupos.
    * @param int $num_clasificados El número de equipos que clasifican en
cada etapa.
    * @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($grupo)){
                // Inserta los nuevos torneos en la base de datos.
                for($i=0;$i<$grupo;$i++){
                    $resultado[$i] = $objeto_torneo-
>insertar_torneo($etapa, $fecha_inicio, $fecha_fin, "GRUPO ".$i+1,
$num_clasificados[$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 (!empty($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();
?>
```

El código relacionado con las otras historias de usuario consideradas para este sprint se puede encontrar en el Anexo VI.

La documentación generada para este sprint está disponible en el enlace almacenado en el repositorio correspondiente, el cual se puede encontrar en el Anexo IX.

Sprint review

A continuación, a modo de ejemplo, se detallan dos tareas que se completaron satisfactoriamente dentro del plazo establecido, junto con los criterios de aceptación correspondientes a cada una. Las restantes se encuentran en el Anexo X.

Tabla 13. Criterios Aceptación Sprint 2

Código	Historia de Usuario	Criterios de aceptación	Cumplimiento
GT001-1	Registrar torneos	Envió de datos correctos	✓
		Envió de id de categoría incorrecto	✓
		Envió de datos vacíos	✓
GT001-2	Generar partidos	Envió de datos correctos	✓
		Envió de id de torneo incorrecto	✓
		Envió de id de torneo que tiene partidos generados	✓
		Envió de datos vacíos	✓
		Envió de id de equipo incorrecto	✓

Sprint Retrospective

- **¿Qué se hizo bien?** Se completaron satisfactoriamente diversas historias de usuario, lo que sugiere una implementación efectiva conforme a los requisitos establecidos. Además, se corrigió la estructura de los métodos que interactúan con la base de datos, lo que mejora la eficiencia y mantenibilidad del sistema.
- **¿Qué se hizo mal?** Al guardar los partidos generados, inicialmente se creaba una matriz para almacenar cada partido con sus atributos correspondientes. Este enfoque resultó en un proceso costoso tanto en la creación como en la extracción de datos. Además, la actualización de la matriz no se llevaba a cabo de manera eficiente.
- **¿Qué se puede mejorar?** Se sugiere crear una tabla en la base de datos para almacenar directamente cada partido con sus respectivos atributos. Esto simplificaría el proceso de generación de partidos, eliminando la necesidad de crear matrices previas. Además, facilitaría la actualización de datos, mejorando la eficiencia del sistema en general.

2.6 SPRINT 3

Sprint Planning

El propósito del Sprint 3 es agilizar el proceso de registro de resultados, estadísticas de jugadores en cada partido, alineaciones y los informes de árbitros correspondientes. Además, se espera completar todas las historias de usuario relacionadas con la épica de publicar información. Esto permitirá a los aficionados acceder a toda la información relevante sobre ligas, categorías, torneos, equipos y jugadores de manera fácil y rápida.

En la Tabla 14 se detallan las tareas de ciertas historias de usuario que fueron consideradas para el Sprint Backlog 3, el resto se encuentra en el ANEXO IV. Para este Sprint se estimaron un total de 45 story-points.

Tabla 14. Sprint Backlog 3

Código	Historia de Usuario	Tareas
GP001-2	Registrar resultados	Método verificar ID
		Método mostrar partidos
		Método modificar partidos
		Método actualizar estadisticas de equipo
		Método modificar partido
		Prueba del método verificar ID
		Prueba del método mostrar partidos
		Prueba del método modificar partidos
		Prueba del método actualizar estadisticas de equipo
		Prueba del método modificar partido
GP001-3	Comprobar registro de estadisticas de jugadores	Método verificar ID
		Método extraer partidos jugados
		Prueba del método verificar ID
		Pruebas del método extraer partidos jugados

Implementación

- **Historia de Usuario PI001-4**

A continuación, se presenta el código desarrollado para la implementación de la historia de usuario “PI001-4 Mostrar partidos”.

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

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

```php
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();
?>
```

El código relacionado con las otras historias de usuario consideradas para este sprint se puede encontrar en el Anexo VII.

La documentación generada para este sprint está disponible en el enlace almacenado en el repositorio correspondiente, el cual se puede encontrar en el Anexo IX.

Sprint review

A continuación, a modo de ejemplo, se detallan dos tareas que se completaron satisfactoriamente dentro del plazo establecido, junto con los criterios de aceptación correspondientes a cada una. Las restantes se encuentran en el Anexo X.

Tabla 15. Criterios de Aceptación Sprint 3

Código	Historia de Usuario	Criterios de aceptación	Cumplimiento
GP001-3	Comprobar registro de estadísticas de jugadores	Envió de datos correctos	✓
		Envió de id de partido incorrecto	✓
		Envió de id de torneo incorrecto	✓
		Envió de id de partido por jugar	✓
		Envió de id de partido por programar	✓
		Envió de id de partido pospuesto	✓
PI001-4	Mostrar partidos	Envió de datos correctos	✓
		Envió de id de torneo incorrecto	✓
		Envió de id de torneo que no tiene partidos	✓

Sprint Retrospective

- **¿Qué se hizo bien?** Se completaron satisfactoriamente diversas historias de usuario, lo que sugiere una implementación efectiva conforme a los requisitos establecidos. Además, se corrigió la forma de almacenamiento de partidos en la base de datos, mejorando así la eficiencia y la estructura del sistema.
- **¿Qué se hizo mal?** Hubo algunas deficiencias en el proceso de registro de estadísticas de jugadores y resultados. Primero, ambos se realizaban en el mismo

archivo, lo que puede generar confusiones y complicar el mantenimiento del código. Además, al registrar estadísticas de jugadores, se omitió la implementación de un método que verificara si estas ya habían sido registradas previamente. Por otro lado, en el registro de sanciones, se descuidó la creación de una columna para insertar a los responsables del tribunal de sanciones.

- **¿Qué se puede mejorar?** En el próximo sprint, es crucial abordar estas deficiencias. Primero, crear la columna faltante en la tabla de sanciones para incluir a los responsables del tribunal de sanciones. Luego, implementar un método de verificación para evitar la inserción de registros duplicados de estadísticas de jugadores del mismo partido. Además, se debería considerar la separación de archivos para el registro de estadísticas de jugadores y resultados, lo que facilitaría la organización y mantenimiento del código.

2.7 SPRINT 4

Sprint Planning

El propósito del Sprint 3 es llevar a cabo tareas relacionadas con la actualización y eliminación de usuarios, ligas, equipos y jugadores. Además, se llevarán a cabo acciones para recuperar contraseña en caso de que los usuarios la olvide, así como registrar el tribunal con los responsables correspondientes.

En la Tabla 16 se detallan las tareas de ciertas historias de usuario que fueron consideradas para el Sprint Backlog 4, el resto se encuentra en el ANEXO IV. Para este Sprint se estimaron un total de 36 story-points.

Tabla 16. Sprint Backlog 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

Implementación

- **Historia de Usuario GU001-5**

A continuación, se presenta el código desarrollado para la implementación de la historia de usuario “GU001-5 Eliminar usuario”.

```php
<?php
/**
 * @file
 * Este archivo contiene un script PHP para eliminar un usuario de tipo
hincha en una 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();
}

/**
 * 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();
?>
```

El código relacionado con las otras historias de usuario consideradas para este sprint se puede encontrar en el Anexo VIII.

La documentación generada para este sprint está disponible en el enlace almacenado en el repositorio correspondiente, el cual se puede encontrar en el Anexo IX.

Sprint review

A continuación, a modo de ejemplo, se detallan dos tareas que se completaron satisfactoriamente dentro del plazo establecido, junto con los criterios de aceptación correspondientes a cada una. Las restantes se encuentran en el Anexo X.

Tabla 17. Criterios Aceptación Sprint 3

Código	Historia de Usuario	Criterios de aceptación	Cumplimiento
GE001-6	Actualizar equipo	Envió de datos correctos	✓
		Envió de id equipo incorrecto	✓
		Envió de datos vacíos	✓
		Envió de nombre existente	✓
GU001-5	Eliminar usuario	Envió de datos correctos	✓
		Envió de datos vacíos	✓
		Envió de correo electrónico no registrado	✓
		Envió de correo electrónico que es presidente	✓
		Envió de correo electrónico que tiene liga registrada	✓

Sprint Retrospective

- **¿Qué se hizo bien?** Se completaron satisfactoriamente diversas historias de usuario, lo que sugiere una implementación efectiva conforme a los requisitos establecidos. Además, se corrigió la columna faltante, se implementó el método de comprobación de estadísticas de jugadores y se separaron los archivos de registros.
- **¿Qué se hizo mal?** A la hora de actualizar la información del equipo y la categoría, no se realizó una verificación previa de los nombres antes de proceder con la actualización. Esta omisión podría conducir a errores o inconsistencias en los datos.
- **¿Qué se puede mejorar?** Se debe implementar una verificación previa de los nombres antes de proceder con la actualización de la información del equipo y la categoría. Esto garantizará la integridad y la coherencia de los datos almacenados en el sistema.

2.8 Fase de Lanzamiento:

El backend del sistema ofrece una solución integral para la gestión de las ligas barriales de Quito. Sus características clave incluyen:

- Gestión completa de equipos y jugadores.
- Programación y organización de partidos.
- Registro de resultados y generación de estadísticas.
- APIs para integraciones externas.

Este backend proporciona la base sólida necesaria para una gestión eficiente y exitosa de las ligas barriales de Quito. El código del backend completo y la documentación necesaria, se encuentra en el Anexo III

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Resultados

Para finalizar el proyecto se ejecutaron 5 Sprints con un total de 224 story points, el detalle de cada uno se presenta en la Tabla 18.

Tabla 18. Story Points por Sprints

Sprint	Story Points
0	56
1	41
2	46
3	45
4	36

A continuación, se presenta el código desarrollado para de la prueba unitaria de la historia de usuario “GU001-1 Registrar Usuario”.

```php
<?php
use PHPUnit\Framework\TestCase;
include('../clase_usuario.php');
include('../../conexion.php');
require 'vendor/autoload.php';
class prueba_1_insertar_usuario extends TestCase {
```

```php
    // 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);

        // 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
        ]);
```

```
    }
}
?>
```

Al ejecutar este archivo con la herramienta PHPUnit, obtuvimos el siguiente resultado, que puede observarse en la Figura 10.

```
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)
```

Figura 10. Resultado de la prueba unitaria.

Este resultado indica que se han ejecutado en total 3 pruebas (tests), realizando 9 aserciones (assertions), que son las comprobaciones que se llevaron a cabo en cada prueba para verificar que el comportamiento del código probado es el esperado. Con ello, se considera que esta historia de usuario ha sido aprobada.

En el Anexo XI se encuentra el código desarrollado para cada historia de usuario, junto con su correspondiente confirmación.

Luego del proceso de desarrollo y pruebas, el componente de backend para las ligas barriales se encuentra en una fase de finalización, listo para su integración con la interfaz de usuario del frontend web.

3.2 Conclusiones

- La colaboración efectiva entre los equipos de desarrollo frontend y backend resultó en una comprensión detallada de los requisitos necesarios para el desarrollo del backend. Esta estrecha colaboración aseguró que el sistema web desarrollado cumpliera con las necesidades específicas para la gestión de ligas barriales en Quito, lo que a su vez mejoró significativamente
- L a implementación de una arquitectura y base de datos diseñada específicamente para atender los requerimientos recopilados proporcionó una solución escalable y eficiente. Este diseño facilitó la gestión de datos complejos relacionados con ligas, equipos y jugadores. La implementación de este diseño aseguró una estructura robusta y adaptable, lo que permitió una fácil expansión y modificación del sistema según las necesidades cambiantes.

- La implementación del backend se realizó con éxito, logrando un sistema funcional que apoya eficazmente la gestión de las ligas barriales. La correcta interpretación de los requerimientos y la solidez del diseño previamente establecido aseguraron la conformidad con las expectativas y la satisfacción del usuario final.
- Las pruebas unitarias realizadas aseguraron la calidad y el correcto funcionamiento del backend. Estas pruebas permitieron identificar y corregir errores de manera eficiente, lo que contribuyó a la estabilidad y eficacia general del sistema. Además, proporcionaron una base sólida para futuras actualizaciones y mantenimiento.

3.3 Recomendaciones

- Es recomendable a la hora de generar el modelo relacional del sistema tener una visión clara de como se van a almacenar los datos, dado que esto llevaría a tener que realizar consultas extras para buscar nuevas formas de almacenamiento lo cual implica para el desarrollo normal de la codificación.
- Se recomienda dedicar tiempo a identificar y eliminar duplicaciones, corregir malentendidos y mejorar la legibilidad del código. La revisión de código y su refactorización es una actividad esencial para garantizar que el código sea limpio, eficiente y fácil de mantener.
- Se recomienda realizar una auditoría de seguridad completa y abordar cualquier vulnerabilidad identificada. Además, implementar las mejores prácticas de seguridad, a las ya implementadas validaciones de datos de entrada y la prevención de ataques de inyección.
- Se recomienda revisar y mejorar la documentación técnica, incluyendo comentarios en el código, guías de instalación y uso, y manuales de resolución de problemas.
- Es importante mantener el código y las dependencias actualizadas para mitigar riesgos de seguridad y mantener la compatibilidad con las últimas tecnologías. Se recomienda establecer un proceso de mantenimiento regular y una estrategia para manejar actualizaciones de dependencias y frameworks.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] P. Yacelga, «Uso y administración Liga Deportiva Barrial Los,» 17 Febrero 2021. [En línea]. Available: https://www7.quito.gob.ec/mdmq_ordenanzas/Administraci%C3%B3n%202019-2023/Comisiones%20del%20Concejo%20Metropolitano/Propiedad%20y%20Espacio%20P

%C3%BAblico/2022/2022-08-24/Documentos%20de%20tratamient/3.%20Los%20Libertadores/anexos/dmdr-afr-cdu-0007-2021_.

[2] «Guía gratis: ¿Cómo organizar o crear un torneo? o competición | Foro Vivetix,» Foro Vivetix, 2024. [En línea]. Available: https://vivetix.com/f/p/b/como-organizar-crear-torneo#:~:text=Un%20torneo%20es%20un%20conjunto,de%20una%20o%20m%C3%A1s%20categor%C3%ADas..

[3] B. M. Adán, A. R. A. Besari y M. M. Bachtiar, «Diseño de sistema de servidor backend basado en API REST para sistema de pago sin efectivo en comunidad minorista,» *Simposio Internacional de Electrónica (IES) 2019,* pp. 208-213, 2019.

[4] AWS, «¿Qué es una base de datos?,» Amazon Web Services, Inc., 2023. [En línea]. Available: https://aws.amazon.com/es/what-is/database/.

[5] F. L. Rivera Osorio, Base de datos relacionales, Medellin: Fondo Editorial ITM, 2008.

[6] D. Gabriela, «MODELO RELACIONAL,» Blogspot.com, 2024. [En línea]. Available: https://katyygaby.blogspot.com/p/modelo-relacional.html.

[7] E. G. Alvaro , «Manual práctico de SQL,» 2003.

[8] A. Villa Betancur y J. E. Giraldo Plaza, «Automatización de pruebas unitarias de códigos,» *Scientia Et Technica,* vol. XVII, nº 50, pp. 147-151, 2012.

[9] C. Rodríguez y R. D. Vicente, «¿Por qué implementar Scrum?,» *Revista Ontare,* vol. 3, pp. 125-144, 30 Octubre 2015.

[10] M. Trigás Gallego, U. O. d. Catalunya y A. C. Domingo Troncho, «Metodología Scrum,» Universitat Oberta de Catalunya, 2012.

[11] T. Satpathy, «Una guía para el CONOCIMIENTO DE SCRUM (Guía SBOK™) - 3ra Edición,» VMEdu, Inc, Avondale, 2013.

[12] E. Bahit, Scrum y eXtreme Programming para Programadores, Buenos Aires: Alfaomega Grupo Editor, 2012.

[13] PHP, «PHP: ¿Qué es PHP? - Manual,» PHP, 2024. [En línea]. Available: https://www.php.net/manual/es/intro-whatis.php.

[14] Microsoft, «Visual Studio Code,» Microsoft, 03 Noviembre 2021. [En línea]. Available: https://code.visualstudio.com/docs.

[15] Apache, «XAMPP Installers and Downloads for Apache Friends,» Apachefriends.org, 2023. [En línea]. Available: https://www.apachefriends.org/es/index.html.

[16] MySQL, «MySQL :: MySQL Workbench,» Mysql.com, 2024. [En línea]. Available: https://www.mysql.com/products/workbench/.

[17] G. B., «Qué es GitHub y cómo empezar a usarlo,» Tutoriales Hostinger, 12 Abril 2019. [En línea]. Available: https://www.hostinger.es/tutoriales/que-es-github.

[18] «Getting started with GitHub Desktop - GitHub Docs,» GitHub Docs, 2024. [En línea]. Available: https://docs.github.com/en/desktop/overview/getting-started-with-github-desktop.

[19] «Azure DevOps Hands-On Labs,» Azuredevopslabs.com, 2023. [En línea]. Available: https://azuredevopslabs.com/.

[20] «PHPUnit – The PHP Testing Framework,» Phpunit.de, 2 Febrero 2024. [En línea]. Available: https://phpunit.de/.

[21] «About Clever Cloud | Clever Cloud,» Clever Cloud, 19 Enero 2024. [En línea]. Available: https://www.clever-cloud.com/about/.

[22] «Doxygen homepage,» Doxygen.nl, 2023. [En línea]. Available: https://www.doxygen.nl/.

[23] info@erdplus.com, «ERDPlus,» Erdplus.com, 2024. [En línea]. Available: https://erdplus.com/about.

5 ANEXOS

ANEXO I. Historias de Usuario

ANEXO II. Historias de Usuario en tareas

ANEXO III. Enlace al repositor del código

ANEXO IV. Sprint backlog de cada sprint

ANEXO V. Código desarrollado para el Sprint 1

ANEXO VI. Código desarrollado para el Sprint 2

ANEXO VII. Código desarrollado para el sprint 3

ANEXO VIII. Código desarrollado para el sprint 4

ANEXO IX. Enlace al repositor donde se encuentra la documentación

ANEXO X. Historias de Usuario con sus respectivos criterios de aceptación y cumplimiento

ANEXO XI. Pruebas Unitarias

Los anexos se encuentran en el siguiente repositorio:

https://github.com/CarlosVelasquezZ/SGLigas_backend/blob/main/Anexos/Anexos.pdf