PROTOTIPO DE HERRAMIENTA SOFTWARE CON ENFOQUE WEB DEPORTIVO Y CON AUTOMATIZACIÓN DE LA PROGRAMACIÓN DE TORNEOS EN UNIVERSIDADES Y CLUBES DEPORTIVOS

ERIKA VIVIANA FRANCO LÓPEZ
YURI SANGUINO RODRIGUEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2015

PROTOTIPO DE HERRAMIENTA SOFTWARE CON ENFOQUE WEB DEPORTIVO Y CON AUTOMATIZACIÓN DE LA PROGRAMACIÓN DE TORNEOS EN UNIVERSIDADES Y CLUBES DEPORTIVOS

ERIKA VIVIANA FRANCO LÓPEZ
YURI SANGUINO RODRIGUEZ

Trabajo de grado para optar el título de
Ingeniero de Sistemas

Director
MSc. MANUEL GUILLERMO FLOREZ BECERRA
Ingeniero de Sistemas

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2015

Este trabajo está dedicado a papito Dios que me dio fortaleza y sabiduría para culminar este proceso.

Gracias totales a mi "Nita", uno de los seres más importantes y especiales en mi vida, al hombre que papito Dios escogió como mi idóneo, Diego Chaparro, por ser esas personas usadas por Dios para tener palabras de aliento en momentos de angustia y desespero, creyendo en mi todo momento.

A todos y cada uno de los integrantes de mi familia, quienes a pesar de las circunstancias, aportaron todo cuanto pudieron para que cumpliera con mi objetivo.

A la familia Chaparro Cely, a mis excelentes amigas, María Eugenia Suárez y Deyci Carolina Carvajal y a mi compañera de proyecto y su familia, quienes con su apoyo, estuvieron presentes y por supuesto creyeron en mí.

Yuri Sanguino Rodríguez.

A Dios Todopoderoso, quien hace realidad cada cosa, y por supuesto, fue el autor principal de este trabajo.

A mis padres Efraín Franco y Bárbara López, dos hermosos ángeles venidos del cielo, quienes, a pesar de todas las dificultades presentadas, estuvieron dándome apoyo de todo tipo y creyeron en mí.

A mis hermanos Efraín Danilo y María Margarita, mis principales motivaciones de vida, por ser la razón de luchar a diario para entregarles el mejor ejemplo de persona humilde, piadosa, y guerrera.

A mi gran amiga y hermana Jenny Mujica por celebrar mis glorias y apoyarme en los fracasos.

A Carlos Vega, mi viejo amigo, por dar su tiempo y su mano cariñosa para acompañarme y apoyarme de forma más que incondicional.

A Silvia Nossa, Jessica Rueda, Cristian Vega, David Burgos y mi primo Hector Franco por ser mi segunda familia mientras no estuve en casa y enseñarme a siempre brindar una mano de ayuda desinteresada.

A la Coral UIS, lugar donde crecí y aprendí a vivir con pasión, a ser consciente de que convivo con personas diferentes que tienen cada uno su propio toque de luz, y a obtener siempre lo bueno de todos y de todo.

Erika Viviana Franco López

CONTENIDO

pág.

LISTA DE TABLAS

pág.

LISTA DE FIGURAS

pág.

GLOSARIO

ADMINISTRADOR: persona integrante de la institución encargada de la administración del portal. En el sistema es un tipo de usuario.

CATEGORÍA: Cada una de las categorías a las que pertenecen los jugadores de tenis: novato, cuarta, tercera, segunda, intermedia o primera. Por medio de ésta se clasifican los jugadores en un torneo.

CUADROS DE ENFRENTAMIENTO: Estructura configurada por el administrador que indica cómo se enfrentarán los jugadores inscritos en el torneo.

EVENTO: Puede referirse a un torneo, a un partido o a un periodo de inscripciones.

GID-CONUSS: Grupo de Investigación y Desarrollo en Computación en la Nube, Seguridad, Servidores y Servicios. Grupo de investigación al que pertenecen las autoras de éste proyecto y es el grupo que da soporte al modelo Software as a Service.

INSTITUCIÓN: en el documento, una institución hace referencia a la institución deportiva, universidad o empresa, que adquiere el servicio de utilización del portal.

JUEGO: cada una de las partes que componen un set. Por lo general, son seis por cada set, pero en ciertas ocasiones puede cambiar debido a los empates o “iguales”.

JUGADOR: Persona que se inscribe y participa en un torneo.

NIVEL DE DESTREZA: Capacidad de desempeño que posee una persona en un deporte: principiante, básico, intermedio o avanzado.

PARTIDO DE TENIS: o encuentro deportivo, evento en el que dos jugadores se enfrentan jugando el deporte conocido como tenis. De un partido se obtiene siempre un ganador quien dentro de un torneo pasará a la siguiente fase.

PORTAL: en el documento, portal es el término utilizado para identificar el sitio web completo.

SAAS: es la abreviación de Software as a Service, el cual es el modelo de distribución de software que usa el portal.

SET: es cada una de las etapas por las que está compuesto un partido de tenis. La cantidad de éstas que tenga un partido cambia de acuerdo a las políticas del torneo.

SET DE TIE-BREAK: "El primer jugador o equipo que gane seis juegos ganará el "set", siempre que haya un margen de dos juegos sobre el oponente. Si el marcador alcanza los seis juegos iguales para ambos, se jugará un tie-break."[1]

TORNEO: "Conjunto de pruebas en que una serie de contrincantes compiten por conseguir el triunfo"[2]. En este proyecto, nos referimos a torneos específicamente de tenis jugados por personas integrantes de una institución. Los torneos tienen un calendario de fechas, lugares, puntajes y un ganador.

USUARIO: en el documento, un usuario es un término genérico utilizado para identificar a las personas que hacen parte de la institución, quienes son los que utilizan el sistema de información.

[1] *ITF Reglas del tenis*. París, International Tennis Federation 2013, pág. 6 de 46.
[2] Oxford University Press. Oxford Dictionaries: Definición de torneo en español [online]. Reino Unido [citado 2 junio de 2015]. Disponible en Internet: <URL: http://www.oxforddictionaries.com/es/definicion/espanol/torneo>.

RESUMEN

TITULO: PROTOTIPO DE HERRAMIENTA SOFTWARE CON ENFOQUE WEB DEPORTIVO Y CON AUTOMATIZACIÓN DE LA PROGRAMACIÓN DE TORNEOS EN UNIVERSIDADES Y CLUBES DEPORTIVOS.*

AUTORES: ERIKA VIVIANA FRANCO LÓPEZ
YURI SANGUINO RODRIGUEZ**

PALABRAS CLAVE: *LARAVEL*, *SCRUM, SAAS*, AUTOMATIZACIÓN, DEPORTE.

DESCRIPCIÓN:

La implementación de un prototipo de herramienta software para la web, que permita programación de torneos deportivos automáticamente y condense información deportiva completa, surge de una necesidad social detectada, la cual corresponde a la falta de medios tecnológicos para fomentar el deporte, y con ello relaciones interpersonales, los cuales son pilares fundamentales para garantizar la salud física y mental de las personas.

El contenido de este libro inicia con un resumen de las teorías y tecnologías necesarias para escribir la solución, después, plantea el proceso de implementación de la herramienta software propuesta, el cual está enmarcado en la metodología discutida y escogida para el desarrollo: *SCRUM*. Consiste en tres etapas o *sprints*:

- Levantamiento y análisis de los requerimientos funcionales, diseño del software por medio de diagramas *UML* y modelo de datos relacional para definir la base de datos.
- Programación de las interfaces o *front-end.*
- Programación funcional y del lado del servidor o *back-end.*

Como soporte al proceso se documentan las pruebas realizadas al software con sus respectivos resultados, obtenidos de las respuestas que diferentes personas que trabajan en el medio o son de otras áreas dieron al usar y manipular la herramienta. Luego se informa al lector sobre las características que tendrá el software para ser distribuido como servicio (*SaaS*). Por último, el documento incluye anexos para profundizar en parte específicas del contenido.

* Trabajo de grado

** Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: MSc. Manuel Guillermo Flórez Becerra.

ABSTRACT

TITLE: PROTOTYPE SOFTWARE TOOL WITH WEB APPROACH TO SPORTS AND AUTOMATION ESCHEDULE TOURNAMENT IN UNIVERSITIES AND SPORT CLUBS*

AUTHORS: ERIKA VIVIANA FRANCO LÓPEZ
YURI SANGUINO RODRIGUEZ**

KEY WORDS: LARAVEL, SCRUM, SAAS, AUTOMATION, SPORT

DESCRIPTION:

The implementation of a prototype software tool for the Web that allows scheduling of tournaments and condense automatically complete sports information, comes from a social necessity detected, which corresponds to a lack of technological means to promote sport, and with it interpersonal relationships, which are essential to guarantee the physical and mental health of the people.

The contents of this book begins with an overview of the theories and technologies needed to write the solution, after, raises the process of implementation of the proposed software tool, which is framed in the methodology discussed and chosen for development: *SCRUM*. It consists of three stages or *sprints*:

- Lifting and analysis of functional requirements, software design using UML and relational data model to define the database.
- Programming of interfaces or *front-end*.
- Functional and server-side programming or *back-end*

As a support to the process, the software tests are documented with their respective results obtained from the responses that different people working in the media or from other areas are given to use and manipulate the tool. Then it informs the reader about the features that the software have to be distributed as a service (SaaS). Finally, the document includes annexes to deepen specific piece of content.

* Bachelor Thesis

** Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: MSc. Manuel Guillermo Flórez Becerra.

INTRODUCCIÓN

Nos encontramos en una época donde el impacto de la tecnología es realmente significativo; junto a ello la facilidad de encontrar diversa información en internet por medio de redes sociales, informáticas, turísticas, deportivas, motores de búsqueda, entre otras; dicha información provoca la descentralización en cuanto a muchos de los temas tratados allí, para el proyecto en curso nos interesa la información deportiva que presentan entidades que contemplen dentro de sus servicios, torneos deportivos, que es a la que se da solución.

Otro aspecto a resaltar es la limitada existencia de este tipo de software para la web, el cual expone no solo información de los deportes, noticias, entre otros, sino el proceso de desarrollo en su totalidad de torneos deportivos, lo cual se considera un plus para las entidades locales deportivas, en caso de requerir sus servicios.

Se observó un criterio por parte de algunas personas programadoras, las cuales consideran que para un producir software de calidad se debe tener como cimientos las políticas antiguas, las cuales exponen una informalidad, rapidez, intuición y arte, dicha técnica puede ser útil para ciertos casos, pero en la gran mayoría generan falencias para futuros mantenimientos, en usabilidad, aceptabilidad, calidad, robustez, entre otras, que generan fracaso para el producto final.

Para solucionar lo planteado anteriormente se desarrolló un prototipo de herramienta software con enfoque en la web, que proporciona información organizada y confiable, publicada por las mismas entidades que requieran de sus servicios y las cuales tendrán sus propios administradores, responsables de todo aquello que se presente en la herramienta web, ya que esta proporciona facilidad al momento de hacer gestión de sí misma.

En este libro presentamos el proceso que se realizó para el desarrollo de la herramienta, implementando la ingeniería web por medio de la metodología SCRUM, con el objetivo de generar software con calidad y reducir los posibles riesgos del proceso. Para esto se dividió el proyecto en tres (3) *sprints o* incrementos; en el primero se realizaron todas la tareas pertenecientes a diseño de software, en el segundo se realizaron todas las tareas correspondientes al *front-end* (interfaz) de la herramienta web y en el último se realizó el desarrollo de toda la parte *back-end* de la herramienta para dar por terminado el desarrollo total de la herramienta software con éxito.

1 GENERALIDADES

1.1 PLANTEAMIENTO DEL PROBLEMA

A lo largo de la historia una de las premisas centrales en las que la filosofía ha basado sus estudios es la de entender al ser humano como un ser social por naturaleza. Bajo esta perspectiva lo ha definido, desde la antigua Grecia y conforme a los postulados de Aristóteles, como un animal político, otorgándole a la vez la necesidad de desarrollar la habilidad de vivir en sociedad para alcanzar la felicidad.

Las teorías socio humanísticas que plantean que al hombre le es necesario, para ser feliz, la vida en sociedad, definen el planteamiento de este proyecto. Una de las principales formas de interacción es el deporte, las competencias deportivas se remontan a los primeros Juegos Olímpicos celebrados en el año 776 a.C en la ciudad de Olimpia. Hoy la participación en eventos deportivos supone un goce para los competidores y un gran reto para el gerenciamiento de las organizaciones que dedican su tiempo y sus recursos a la realización de torneos.

En la actualidad, las herramientas ofimáticas representan para el componente administrativo y financiero de cualquier negocio una atractiva opción. Los ciclos administrativos requieren de la automatización de algunas tareas que implican recursos humanos y económicos y cuya implementación reduciría drásticamente los costes de operación.

Realizar un evento con éxito requiere de una minuciosa planificación, de la exactitud en el tratamiento de datos y de la reducción de tiempos, desafortunadamente poco de esto está implementado en la actualidad, traduciéndose en un obstáculo para la frecuente realización de eventos deportivos y la generación de espacios para los deportistas de la ciudad.

De esta manera se denota la innegable necesidad de desarrollar una herramienta web que asista a universidades y clubes locales en el fomento e integración de estudiantes y comunidad en general con el deporte; la concepción de dicha herramienta junto con su implantación redundará en la viabilización no solo de eventos deportivos, construcción de conocimiento y desarrollo del ser, sino en la satisfacción de ver implementado el conocimiento de los estudiantes que desarrollamos este proyecto.

1.2 OBJETIVOS

1.2.1 Objetivo general. Implementar un prototipo Software as a Service (SaaS) para la automatización de eventos deportivos en clubes y universidades a través de la web.

1.2.2 Objetivos específicos

- Implementar un portal web configurable a las necesidades de las instituciones y entidades.
- Consolidar información general acerca de los deportes: tenis, fútbol, baloncesto, voleibol.
  - Reglamentos e información general de cada deporte.
  - Noticias relacionadas con eventos deportivos Nacionales, internacionales y propios de la comunidad.
- Automatizar la programación de los eventos.
  - Aplicado a torneos deportivos de tenis.
  - Publicidad y promoción de eventos.
  - Inscripciones a los torneos.
  - Clasificación y selección de participantes.
  - Programación y cuadros de enfrentamientos.
  - Reprogramación de torneos.
  - Calendario de eventos.
  - Publicación de finalistas y ganadores.
  - Historial de eventos realizados.
- Comunicar a los usuarios del portal la información a través de e-mail.
- Facilitar la realización de encuentros en la comunidad deportiva inscrita en el portal.
  - Proporciona el horario de disponibilidad de los usuarios del portal permitiendo la creación de encuentros individuales.
- Promocionar a través de banners publicitarios marcas de compañías deportivas para su financiación.
- Fomentar la salud y el deporte.
  - Implementando una cartelera orientada a información y artículos de terceros relacionados con salud y deporte.
- Socializar los resultados del proyecto mediante la escritura de un artículo.

2 MARCO TEÓRICO Y METODOLÓGICO

En esta parte se presentaran las bases teóricas en las que se fundamenta cada una de las fases del proyecto, así como la metodología utilizada.

2.1 ANTECEDENTES

En la actualidad encontramos sitios web deportivos, dedicados en su mayoría a transmitir información; industrias deportivas como Fox Sport, ESPN y asociaciones como la Asociación Colombiana de Universidades Nacionales, ASCUN, los tienen, sin embargo no se encuentra dentro de ellos una herramienta que pueda utilizarse para la automatización de eventos. Lo anterior nos permite considerar que para esta investigación no servirán como modelos, considerando que nuestros objetivos van más a fondo de la funcionalidad que brindan los sitios existentes en la web.

2.2 CLOUD COMPUTING (COMPUTACIÓN EN LA NUBE)

La palabra “nube” se utiliza como una metáfora de la “internet” y se refiere a un tipo de informática: La computación en la nube o cloud computing se basa en compartir los recursos informáticos en lugar de tener servidores locales o dispositivos personales para manejar aplicaciones, usando para ello internet. [3]

2.2.1 Características

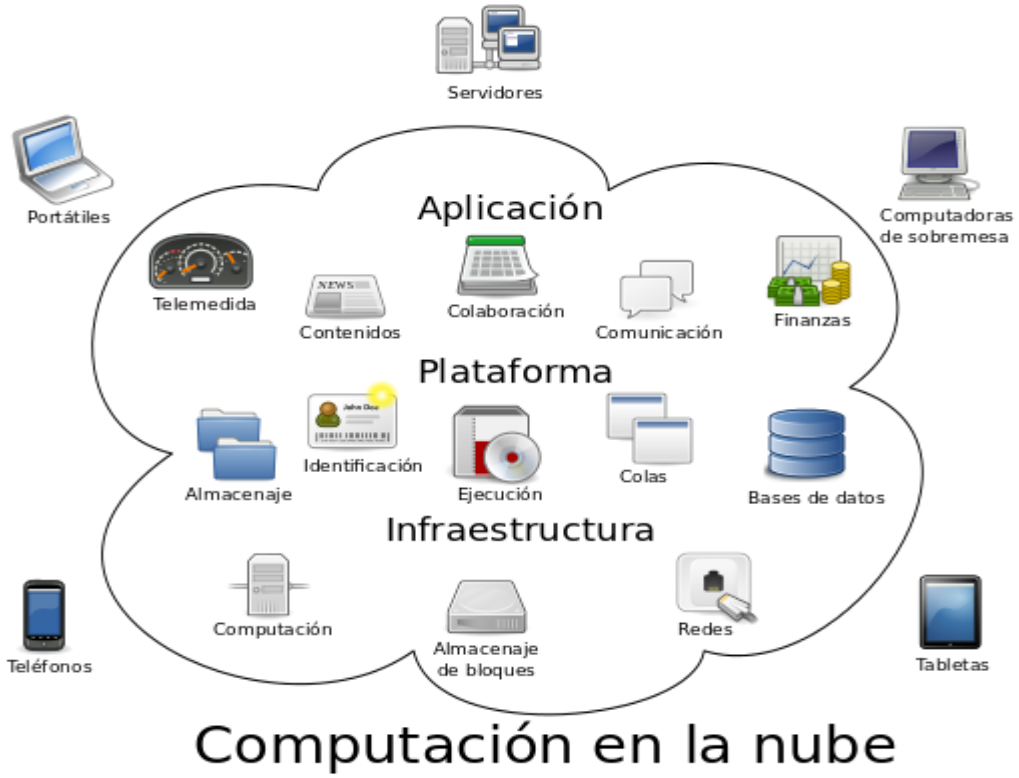
- Autoservicio por demanda: El cliente puede añadir o quitar recursos sin interacción del proveedor.
- Amplio acceso a las redes: Los recursos están disponibles en la red y se acceden a través de dispositivos estándar y heterogéneos.
- Recursos Comunes Compartidos: Los recursos informáticos del proveedor se comparten para servir a múltiples clientes, usando un modelo multi-arrendatario, con diferentes recursos físicos y virtuales asignados en forma dinámica.
- Elasticidad: Escalar rápidamente las capacidades de los servicios, en algunos casos de forma automática. Reacción rápida a cambios.
- Servicio Medido: Una forma automática de controlar y optimizar el uso de recursos, que pueden ser monitoreados brindando transparencia, tanto para el proveedor y el consumidor del servicio utilizado.

[3] Téllez Valdés, Julio. "Computo en la nube: instrumento y objeto del derecho," En: XV Congreso iberoamericano de derecho informática. Buenos Aires, 2011.

2.2.2 Niveles de servicio [4]

- IaaS: (*Infrastructure as a Service*) es entregar hardware y software como un servicio. El ejemplo más común es el *hosting*, el cual, nos provee de hardware como un servidor y de software como un *webserver*.
- PaaS: (*Plataform as a Service*) es entregar una plataforma de desarrollo de aplicaciones como un servicio para desarrolladores en la web. Generalmente se provee de herramientas tipo *middleware*. Además, se ofrece un ambiente de ejecución como el servidor de aplicaciones.
- SaaS: (*Software as a Service*) o software como servicio provee la administración y hosting de aplicaciones con sus propios centros de datos y la hace disponible a los clientes y múltiples usuarios a través de Internet. En el capítulo 8 se referenciará la caracterización como un SaaS del portal web deportivo.

Figura 1: Niveles de servicio del Cloud Computing[5]



[4] S. Bennett, M. Bhuller, and R. Covington, Architectural Strategies for Cloud Computing, Oracle Corporation, 2009.

[5] JOHNSTON, Sam. Disponible bajo la licencia CC BY-SA 3.0 vía Wikimedia Commons – [citado 3 de Junio de 2015] <URL:http://commons.wikimedia.org/wiki/File:Cloud_computing-es.svg#/media/File:Cloud_computing-es.svg>

2.3 LENGUAJE UNIFICADO DE MODELADO (UML)

UML es ampliamente aceptado como el lenguaje de descripción estándar de facto para la especificación y diseño de sistemas de software orientado a objetos. El punto importante a señalar aquí es que UML es un lenguaje para especificar y no un método o procedimiento. Se usa para definir un software, para detallar los artefactos del sistema y para documentar la construcción. Se puede utilizar en una variedad de maneras para apoyar a una metodología de desarrollo de software.

2.3.1 Tipos de Diagramas UML. Los diagramas utilizados en la realización de este proyecto de grado son:

- Diagrama de Casos de Uso: Los Diagramas de Casos de uso se utilizan para capturar la naturaleza dinámica de un sistema. Se compone de casos de uso, actores y sus relaciones. Se utilizan para realizar el diseño de alto nivel para la captura de los requisitos de un sistema.
- Diagramas de Secuencia: Los diagramas de secuencia describen cómo los objetos interactúan a lo largo del tiempo a través de un intercambio de mensajes. Un diagrama de secuencia único, a menudo representa el flujo de eventos para un caso de uso.

2.4 METODOLOGÍA DE TRABAJO

La importancia de usar una metodología para el desarrollo, en este caso de software, se basa en el hecho de tener una adecuada sistematización de los procesos que se llevaran a cabo, dentro de los cuales se encuentra la planificación, conjetura de costes y garantía de la calidad durante todo el proceso de realización del software; todo ello conlleva a tener un producto eficiente y libre de errores. Para la realización de este proyecto se usará la metodología *SCRUM*.

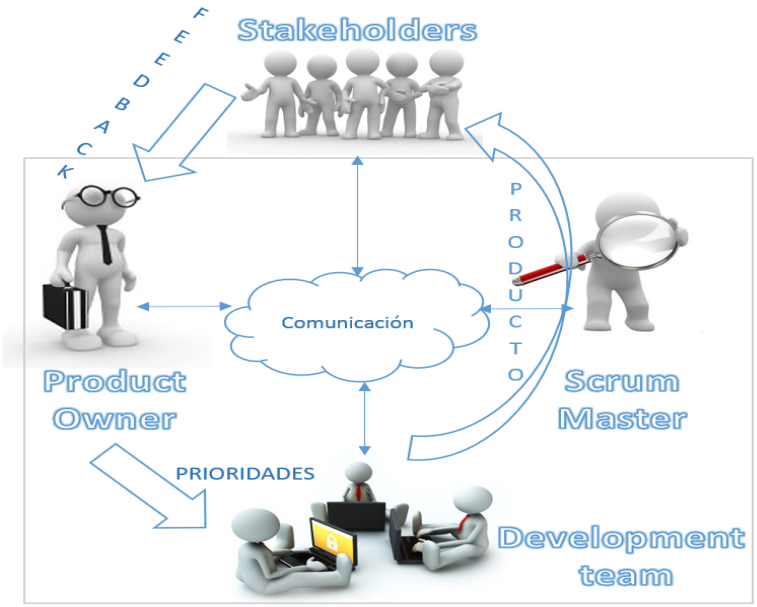
SCRUM es un método ágil, con comportamiento de *Framework* ya que refleja un marco de trabajo, propuesto para el desarrollo y mantenimiento de productos complejos a fin entregarlo con un máximo valor posible, de manera eficaz y creativa ya que se caracteriza por ser ligero y fácil de entender a pesar de poseer complejidad amplia al momento de dominarlo.

Este marco de trabajo de caracteriza por mostrar la eficacia relativa de prácticas de gestión para el producto y prácticas de desarrollo con el fin de mejorar.

SCRUM se basa en la teoría de control de procesos empírica, la cual afirma que el conocimiento se da tras la experiencia y la toma de decisiones, producto de lo que se conoce. *SCRUM* además trabaja bajo un enfoque iterativo e incremental a fin de optimizar la predictibilidad y el control de riesgo, para ello usa tres pilares relevantes en los procesos empíricos: transparencia, inspección y adaptación.

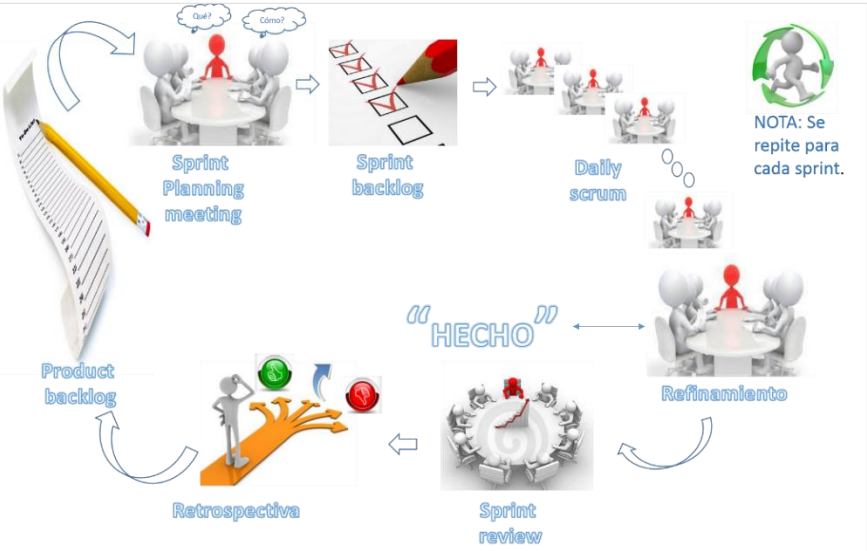
Para lograr cumplir con la entrega de un producto de calidad *SCRUM* se ayuda con el siguiente equipo dentro de su proceso:

Figura 2: Equipo SCRUM.



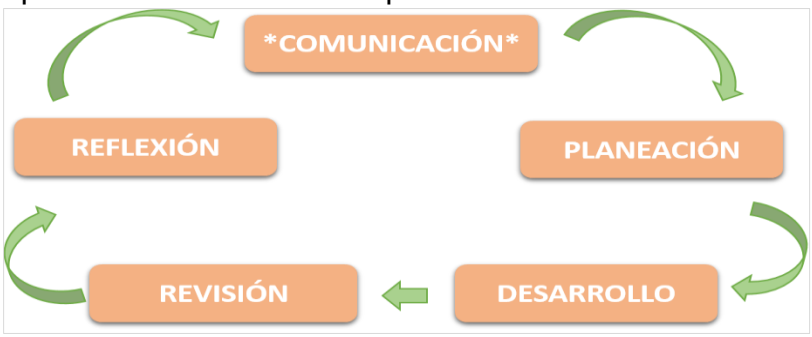
SCRUM posee dentro de su marco de trabajo algunas fases que se aplican iterativamente para cada *Sprint* (incremento) del producto durante toda la ejecución del proyecto, independiente de su tamaño y complejidad, estas fases son:

Figura 3: Fases SCRUM para un Sprint.



Como anexo se establecieron una serie de etapas necesarias para una mejor organización en cada incremento o sprint, las cuales podemos encontrar a continuación:

Figura 4: Etapas de un incremento o Sprint.



- Comunicación**.** Comprende un análisis de todo aquello que puede hacerse necesario en el producto en el desarrollo de cada *Sprint* en curso, para ello se requiere de la presencia del *product owner*, *development team* (equipo de desarrollo) e incluso de los interesados (*stakeholders*); paralelo a ello se lleva a cabo una reunión entre el equipo de desarrollo a fin de discutir posibles ideas para el desarrollo de las funcionalidades.

- Planeación**.** Definir un plan para el *Sprint* (incremento), donde se establecen acciones de riesgos probables, recursos y cronogramas.

- Desarrollo**.** Abarca el desarrollo integral del producto, desde la creación de modelos que ayudan a un mejor entendimiento de las necesidades del cliente desde el punto de vista del desarrollador y un diseño para lograr estos requisitos, hasta la combinación de diferentes lenguajes de programación necesarios para su construcción.

- Revisión**.** Comprende la ejecución de un conjunto de pruebas sobre el *Sprint* del producto, con ello se mitigan riesgos implícitos y se validan supuestos; en el intervienen el *product owner* y el equipo de desarrollo en compañía de los *stakeholders*.

- Reflexión**.** Presentación de aspectos a mejorar o promover en la forma de trabajo, buscando mejorar y dar lugar a las prácticas emergentes.

Para el desarrollo total del sistema se dividieron las acciones (cinco en total) y se asignaron a tres (3) *Sprint* para obtener un flujo en el desarrollo del proyecto más ameno y de calidad.

3 TECNOLOGÍAS APLICADAS AL PROYECTO

En la tabla 1 se presenta un resumen de las diferentes tecnologías usadas para el desarrollo del proyecto:

Tabla 1 Tecnologías aplicadas al proyecto.

Lado local	Lado del servidor	Lado cliente	Base de datos
JetBrains PHP Storm (versión 8.0.3)	Apache (versión 2.0)	JAVASCRIPT	MYSQL
Bitbucket	PHP 5	HTML5	
Git para Windows	Composer	BOOTSTRAP (versión 3.0)	
	Framework Laravel (versión 4.2)		

3.1 LADO LOCAL

3.1.1 JetBrains PHP Storm[6]. Es un IDE multiplataforma comercial para PHP. PhpStorm provee un editor para PHP, HTML y JavaScript con análisis de código en marcha, prevención de errores y refactorizaciones automatizadas para código PHP y JavaScript. Las principales características son: editor de código PHP inteligente y análisis de calidad de código.

3.1.2 Bitbucket. Es una herramienta de control de versiones de Git en la nube. Este servicio representa una ventaja pues elimina la necesidad de administrar un servidor local y otorga la tranquilidad de contar con el respaldo de la información, siempre accesible a través de la red.

En este proyecto se usó esta herramienta para facilitar la comunicación de cambios y centralización de código entre las integrantes del grupo o autoras.

3.1.3 Git para Windows. Git es uno de los sistemas de control de versiones más extendidos actualmente, su principal característica es que es distribuido. Nace de la mano de Linus Torvalds en el año 2005 para gestionar el código fuente de Linux y poco a poco ha ido ganando adeptos en comunidades como GitHub o BitBucket.

[6]JETBRAINS S.R.O. PhpStorm Develop Smarter, Not Harder [online]. República Checa: JetBrains Developer Community, 2014 [citado 4 de mayo de 2015]. Disponible en Internet: <URL: https://www.jetbrains.com/phpstorm/documentation/phpstorm_web.pdf>.

La principal diferencia entre Git y cualquier otro VCS (Subversion y compañía incluidos) es cómo Git modela sus datos. Conceptualmente, la mayoría de los demás sistemas almacenan la información como una lista de cambios en los archivos. Estos sistemas modelan la información que almacenan como un conjunto de archivos y las modificaciones hechas sobre cada uno de ellos a lo largo del tiempo.

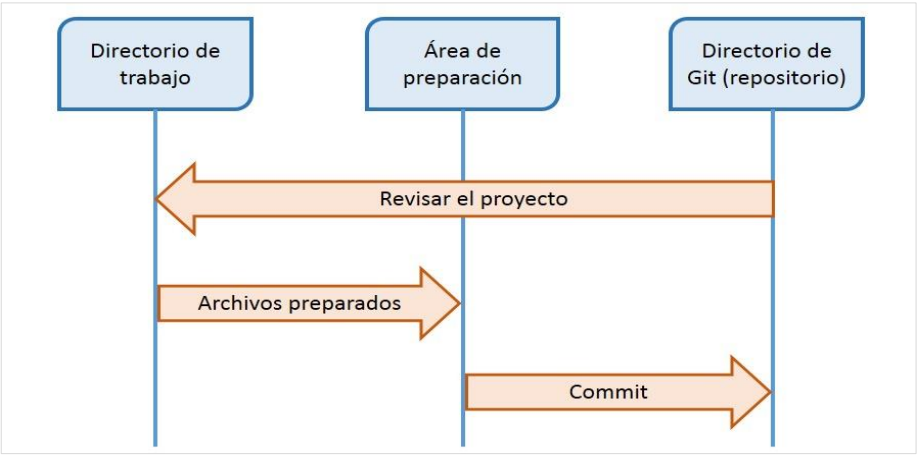
Los archivos en Git se pueden encontrar en tres estados: *Confirmado* (datos almacenados de manera segura en la base de datos local), *Modificado* (archivo modificado pero todavía no se ha confirmado a la base de datos), *Preparado* (archivo modificado marcado en su versión actual para que vaya en la próxima confirmación).

Esto nos lleva a las tres secciones principales de un proyecto de Git:

- El Directorio de Git*:* es donde Git almacena los metadatos y la base de datos de objetos para el proyecto. Es la parte más importante de Git, y es lo que se copia cuando se clona un repositorio desde otro ordenador.
- El Directorio de trabajo*:* es una copia de una versión del proyecto. Estos archivos se sacan de la base de datos comprimida en el directorio de Git, y se colocan en disco para que se puedan usar o modificar.
- El Área de preparación*:* es un archivo, contenido en el directorio de Git, que almacena información acerca de la próxima confirmación.

El flujo de trabajo básico en Git está representado en la figura 5:

Figura 5: Flujo básico de trabajo en Git.



3.2 LADO DEL SERVIDOR

3.2.1 Apache. Herramienta utilizada como servidor web.

3.2.2 Php. Lenguaje de programación de código del lado del servidor.

3.2.3 Composer[7]. Herramienta multiplataforma para gestionar las dependencias de las aplicaciones PHP. Una vez declaradas las librerías de las que depende el proyecto. Composer es un gestor de dependencias. La instalación siempre es local para cada proyecto, ya que las librerías se instalan en un directorio del proyecto (por defecto ese directorio es vendor/). Por defecto Composer no instala ninguna librería globalmente. El problema que resuelve Composer es el siguiente:

- Se dispone de un proyecto que depende de varias librerías.
- A su vez, varias de esas librerías dependen de otras librerías (dependencias "indirectas").
- El desarrollador, solamente declara las dependencias "directas" del proyecto.
- Composer averigua qué librerías deben instalarse (es decir, resuelve todas esas dependencias indirectas) y descarga automáticamente la versión correcta de cada librería.

3.2.4 Laravel. F*ramework* de código abierto para desarrollar aplicaciones con PHP 5. Su filosofía es desarrollar código PHP de forma elegante y simple. Gran parte de Laravel está formado por dependencias, esto implica que el desarrollo de Laravel dependa también del desarrollo de sus dependencias.

3.2.4.1 '*Routes with Closures*'**.** Laravel, propone en el desarrollo usar '*Routes with Closures*', en lugar de un MVC tradicional con el objetivo de hacer el código más claro. Es una vía más rápida en PHP que consiste en programar la interacción HTTP directamente como una función anónima asociada a una Ruta. Esto tiene la ventaja de reducir la cantidad de código, especialmente cuando sólo necesitamos incluir una funcionalidad. Así, donde antes necesitábamos programar una clase, ahora en Laravel sólo requerimos escribir una función en PHP.

[7] WIEDLER, Igor. Composer, Manual oficial [online]. Suiza, 2014 [citado 4 de mayo de 2015]. Disponible en Internet: <URL: https://librosweb.es/libro/composer/>.

3.2.4.2 Manejo de los datos en Laravel**.** Un ORM (*Object Relational Mapper*) en PHP es un software que permite tratar la capa de persistencia de los datos, como simples accesos a métodos de una Clase u Objeto. La funcionalidad interna del ORM es mapear los objetos de PHP a las tablas en la base de datos, para el caso en que la persistencia de los datos de la aplicación es proporcionada por una DB.

Laravel incluye Eloquent ORM. Este ORM se funda en patrón active record. Sin embargo, en Laravel es opcional el uso de Eloquent, que permite la creación de modelos**,** la cual es la forma de interactuar con los datos en un patrón de diseño MVC. Los **Modelos** son clases en PHP que encapsulan todo el trabajo con los datos de una aplicación.

3.2.4.3 Vistas y Layouts**.** Las Vistas en Laravel, son archivos de texto plano, que contienen una plantilla HTML junto con código PHP para desplegar la interfaz web al usuario de la aplicación, representadas por un sistema de procesamiento de plantillas llamado Blade, el cual favorece un código mucho más limpio en las Vistas.

Una gran ventaja de Blade, es el manejo de los Layouts**,** archivos de texto plano que contiene todo el HTML de la página con etiquetas que representan *elementos* o *zonas* a incluir en el Layout, o *vistas parciales*. En Blade, estos elementos incrustados están en un solo archivo, con el fin de mejorar la organización y el rendimiento de las vistas.

3.3 LADO CLIENTE

3.3.1 Javascript. Lenguaje de programación que permite sobrealimentar el lenguaje HTML con interactividad y efectos visuales dinámicos. Se trabajó en conjunto con la biblioteca **JQuery** y la técnica de desarrollo web **Ajax.**

3.3.2 Bootstrap. Es un *framework* para diseño de sitios y aplicaciones web. Permite crear interfaces limpias e intuitivas que se adaptan automáticamente al tamaño de cualquier dispositivo (*responsive design*); Su simpleza les imprime agilidad a la hora de cargar y adaptarse a otros dispositivos.

4 PRIMER SPRINT O INCREMENTO: CAPACITACIÓN DE LOS AUTORES Y DESARROLLO DEL DISEÑO DE SOFTWARE Y BASE DE DATOS

4.1 COMUNICACIÓN

Para este primer sprint se tuvo en cuenta, la capacitación de los autores, dado que el conocimiento en conjunto no era lo suficientemente enriquecedor para el desarrollo del *sprint*. Además, se inició con la construcción del diseño del producto, enmarcado en un flujo que tuvo como punto de entrada un análisis de requerimientos y diseño de software, con el fin de identificar el contexto del negocio, metas y objetivos de la aplicación web.

4.2 PLANEACIÓN

El objetivo principal del *sprint* fue analizar y obtener un diseño óptimo del sistema. Con el fin de cumplirlo, se estableció el siguiente flujo de actividades a realizar: capacitación de las tecnologías por parte de las autoras, análisis de la funcionalidad del producto, especificación de requerimientos, diagramación UML (casos de uso y secuencia), y, modelado de datos del sistema.

Se presentó el siguiente riesgo de atraso en la especificación de requerimientos: dimensionar las funciones del sistema se hizo dispendioso debido a que había variedad de ideas, y acotarlas llevó más tiempo del programado.

Para las actividades se programó tiempo de un (1) mes, el cual tuvo leves alteraciones sin que afectaran el flujo de trabajo. La primera actividad, capacitación de las autoras, se extendió a la par con el desarrollo de todo el sprint, lo cual fue enriquecedor en el marco de la calidad de trabajo. El análisis de funcionalidad y, como ya se referenció en el párrafo anterior, la especificación de requerimientos tomaron más tiempo del asignado (15 días; inicialmente eran 11 días). Diagramación UML, se completó en el tiempo acordado (ocho días), y el modelado de datos se completó en menos tiempo (siete días; inicialmente eran 11 días).

Cada semana se realizó una reunión adicional para controlar los tiempos y el contenido realizado, esto con el fin de garantizar la calidad de los resultados.

4.3 DESARROLLO

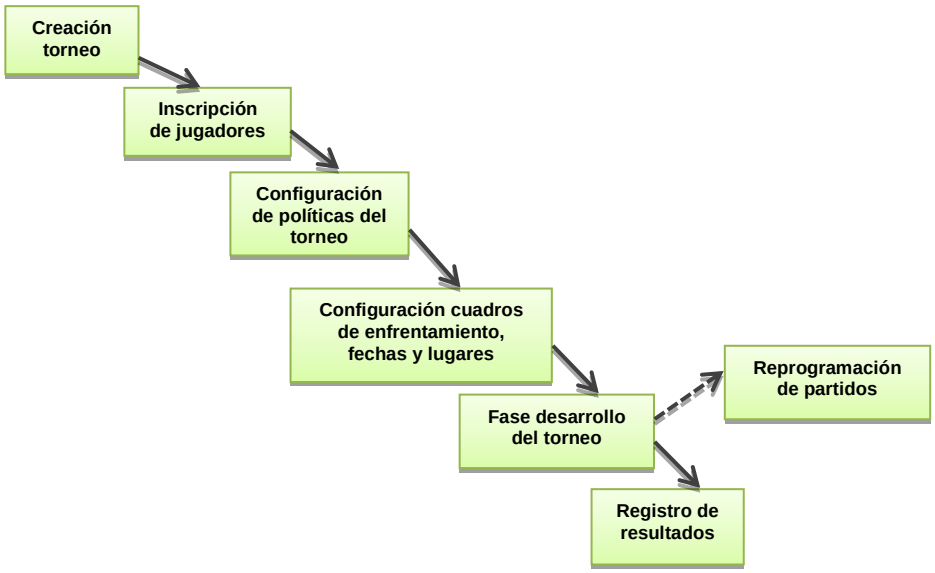
4.3.1 Análisis de funcionalidad del producto. La especificación de requerimientos del sistema portal web deportivo UISport se llevó a cabo con la

colaboración de: MSc. Manuel Guillermo Flórez Becerra, director encargado del proyecto de grado, y el Grupo de Investigación GID-CONUSS, del cual hacen parte activa las autoras.

4.3.1.1 Funcionalidad del producto**.** Debido a las dificultades que se presentan en una institución deportiva a la hora de mantener organizados todos los datos e información necesaria de un torneo, surge la necesidad de apoyarse en un portal en la web diseñado exclusivamente para dicha tarea, el cual facilite la creación y configuración de torneos, historial de los mismos, entre otros aspectos.

En la Figura 6, se ilustra a grandes rasgos el proceso de creación y desarrollo de un torneo.

Figura 6: Procedimiento Creación y Desarrollo de un torneo.



A partir del procedimiento diagramado y teniendo en cuenta sus fases desde la creación del torneo hasta el registro de resultados, el portal web realizará, para suplir ésta necesidad, las siguientes actividades:

- Facilitar la creación y configuración un torneo deportivo de tenis introduciendo todos los parámetros necesarios tales como: fechas de inscripción y de realización del torneo, número de personas por grupo de eliminatoria, número de sets por partido, entre otros.

- Permitir el sencillo y rápido registro de los usuarios que pertenecen a la entidad, así mismo la inscripción a un torneo programado. Seguido a esto, facilitar al administrador clasificar y seleccionar a los participantes.
- Permitir el registro de resultados de cada enfrentamiento, a medida que el torneo se va desarrollando, por parte del administrador.
- En caso necesario, facilitar al administrador reprogramar partidos ingresando nueva fecha y nuevo lugar, según lo amerite.
- Guardar historial de los torneos realizados, con nombre del torneo, ganador, fecha de realización.
- Actualizar y mostrar un calendario de eventos a todos los usuarios, incluso los que no están registrados en el portal web.
- Notificar a los usuarios registrados mediante correo electrónico sobre la creación de un nuevo evento, y permitir al administrador publicar posters y publicidad necesaria del mismo para que los usuarios no registrados también tengan acceso a la información.

Por otra parte, el portal web pretende ser un referente informativo en el cual los usuarios encuentren información completa sobre cada uno de los deportes a los cuales se dará soporte. Dicha información estará compuesta por una reseña histórica, objetivo principal y reglamento, complementados con imágenes y videos ilustrativos; también mostrará una lista de usuarios registrados que lo escogieron como deporte favorito. Ítems que el portal mostrará en forma organizada e intuitiva para facilitar la visualización.

Dentro de otro menú, se encontrarán noticias alusivas a los deportes, las cuales pueden ser internacionales, nacionales o noticias propias de la institución. De la misma manera, artículos y recomendaciones de salud. Para ésta tarea, el sistema realizará las siguientes actividades:

- Actualizar, por medio de sistema RSS, las noticias nacionales e internacionales.
- Permitir al administrador crear, publicar y editar las noticias propias de la institución, los artículos y recomendaciones de salud.
- Facilitar la tarea anterior por medio de editores de texto intuitivos; también inclusión de imágenes, videos y banners publicitarios (en casos de eventos).
- Permitir a los usuarios (administrador y autenticado) realizar comentarios sobre las noticias propias de la institución, los artículos y recomendaciones de salud publicadas.

4.3.1.2 Características de los usuarios del sistema

Usuario	Administrador
Tipo de usuario	Primario
Características.	
Integrante de la institución, encargado de la administración del portal web deportivo. Dentro de las funciones de éste se encuentra la planeación de los torneos deportivos, con tareas como inscripción de jugadores, definir las políticas del torneo, registrar resultados, velar por el desarrollo del torneo y publicación de ganadores.	

Usuario	Usuario autenticado
Tipo de usuario	Primario
Características.	
Persona adscrita a la institución, practicante de uno o varios deportes, cuyo registro en el portal deportivo es avalado por el administrador. Dentro de las funciones de éste se encuentra inscribirse y participar en los torneos programados, consultar y comentar noticias sobre su deporte favorito y artículos sobre el cuidado de la salud.	

Usuario	Visitante
Tipo de usuario	Secundario
Características.	
Persona no integrante de la institución que ingresa al portal web. Las funciones que realiza son consultar información (noticias, artículos, calendario de eventos, entre otros) y registrarse en la página antes de ser avalado por el administrador.	

4.3.1.3 Evolución previsible del sistema**.** A futuro, se pretende que el portal web brinde soporte a funcionalidades nuevas y mejoras, dentro del marco de la implementación de nuevas versiones, de tal manera que se complementen y agreguen aspectos como:

- Versión completa para plataformas móviles.
- Soporte en la creación y configuración de torneos de los deportes faltantes.

4.3.2 Especificación de Requisitos

4.3.2.1 Requisitos comunes de las interfaces

- Interfaces de usuario: La interfaz con el usuario consistirá en un portal web con barras de menús, botones, listas, formularios, imágenes, y sliders. Ésta deberá ser construida específicamente para el sistema propuesto y, será visualizada desde un navegador de internet.

- Interfaces de hardware: Será necesario disponer, en cada institución, de computadores en buen estado los cuales cuenten con acceso a Internet y todos los accesorios básicos como Procesador de 1,66GHz o superior, memoria mínima de 512Mb, mouse y teclado. Por otra parte, el sistema de máquinas en la nube del GID-CONUSS debe tener un soporte que funcione correctamente para evitar la “caída” del sistema.
- Interfaces de software
  - Sistema Operativo: Windows 7 o superior o Linux (cualquier versión).
  - Explorador: Google Chrome o Mozilla Firefox.
- Interfaces de comunicación: Los servidores, clientes y aplicaciones se comunicarán entre sí, mediante protocolos estándares en internet. Por ejemplo, para transferir archivos o documentos deberán utilizarse protocolos existentes (FTP u otros convenientes).

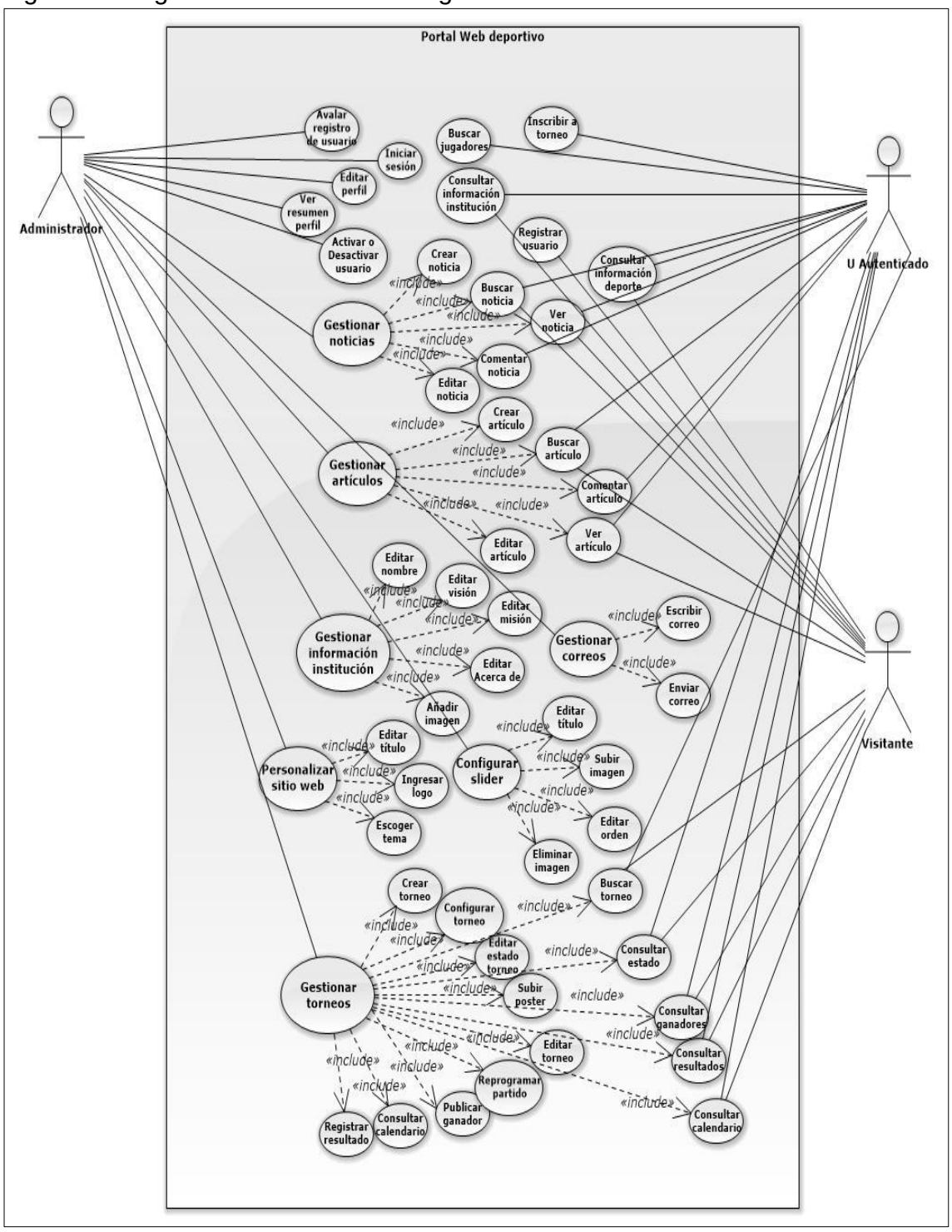
4.3.2.2 Requisitos funcionales y no funcionales del software***.*** Los requisitos funcionales y no funcionales se encuentran en el ANEXO A.

4.3.3 Diagramación UML. Partiendo de la especificación de requisitos funcionales y la descripción de los usuarios del sistema, procedimos a identificar los principales casos de uso. Se listan con sus diagramas a continuación:

1) Registrar usuario
2) Avalar registro de usuario
3) Iniciar sesión
4) Editar perfil
5) Buscar jugadores
6) Activar/Inactivar usuario
7) Gestionar noticias
8) Gestionar artículos
9) Gestionar información institución
10) Personalizar sitio web
11) Gestionar torneos
12) Gestionar correos
13) Configurar slider
14) Consultar información deporte

4.3.3.1 Diagrama casos de uso general

Figura 7: Diagrama de casos de uso general.



4.3.3.2 Diagrama casos de uso de acuerdo a los usuarios.

Figura 8: Diagrama de casos de uso del administrador.

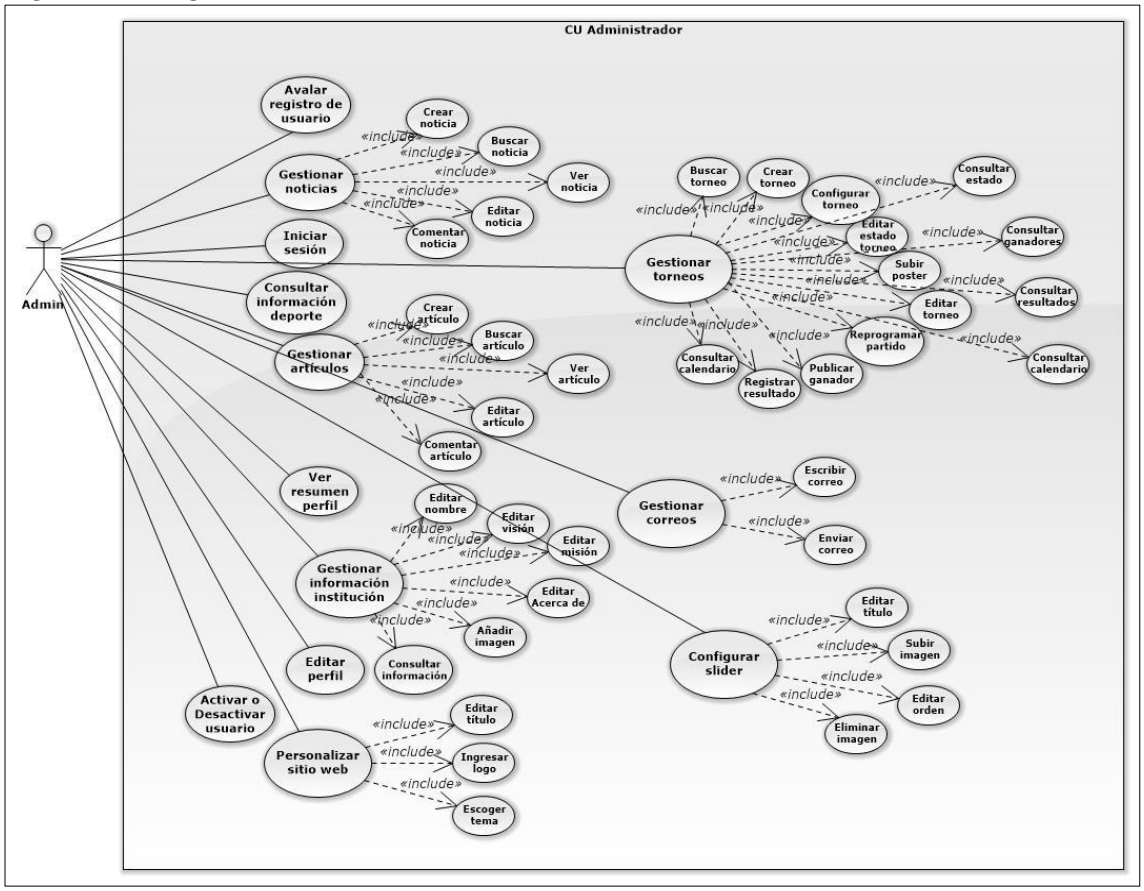


Figura 9: Diagrama de casos de uso del usuario autenticado.

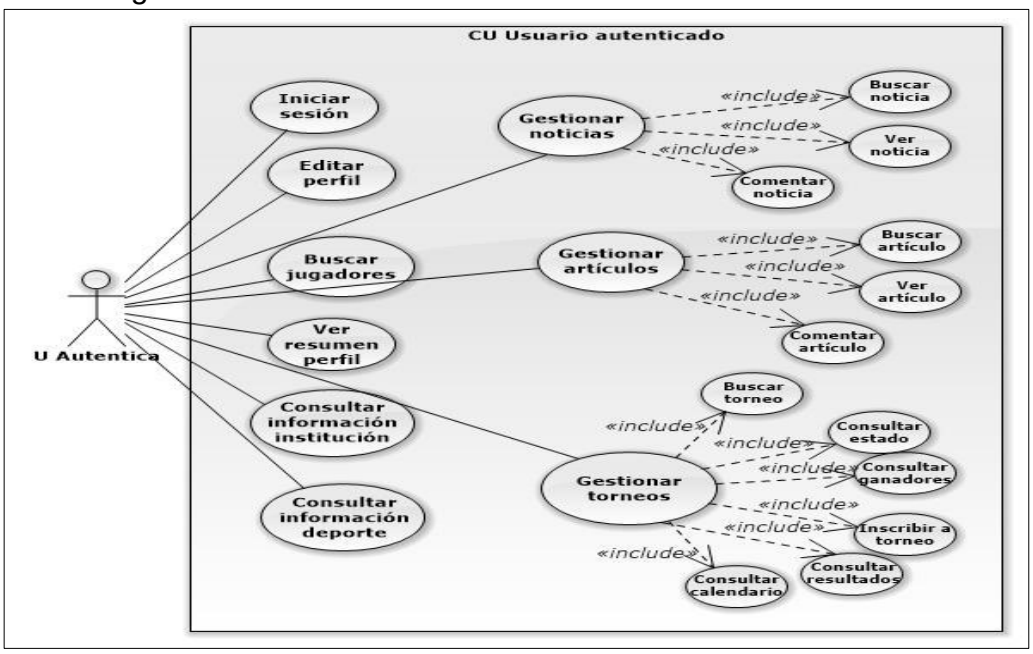
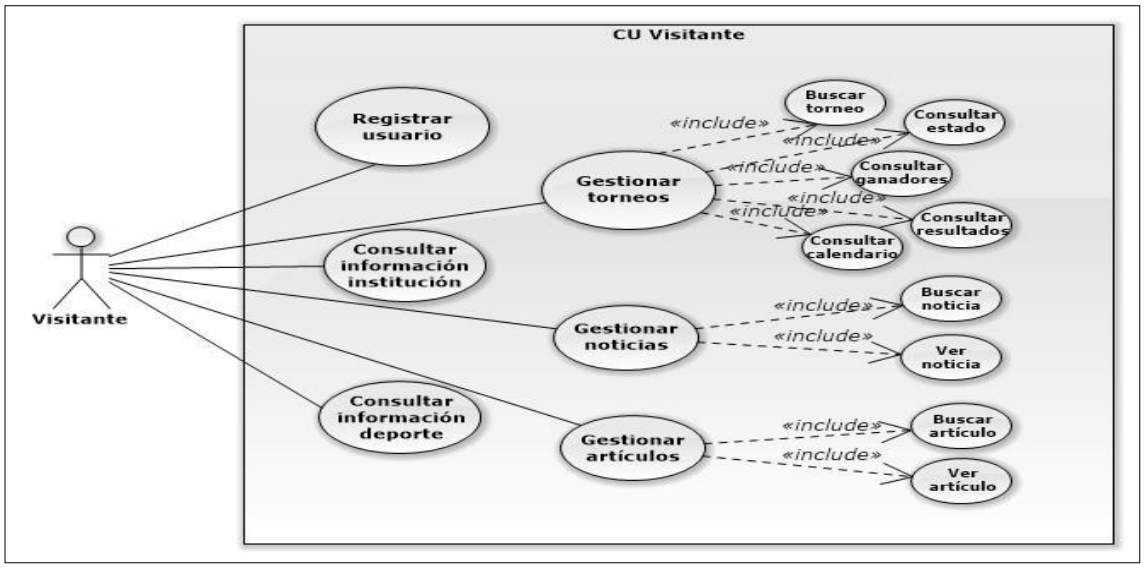


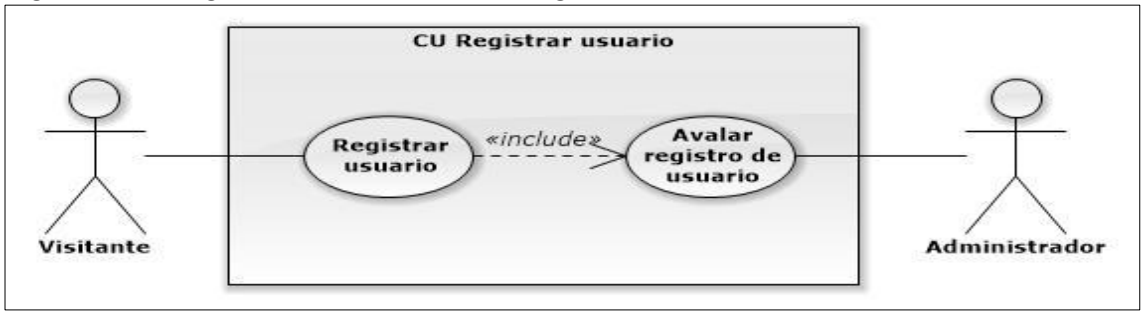
Figura 10: Diagrama de casos de uso del visitante.



4.3.3.3 Descripción de los casos de uso. A continuación se visualizan los diagramas de casos de uso detectados. Las respectivas descripciones de condiciones y escenarios se podrán encontrar en el ANEXO B.

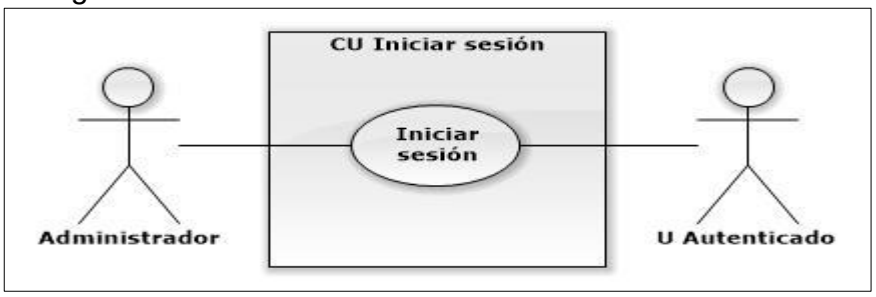
- Registrar usuario

Figura 11: Diagrama Caso de uso - Registrar usuario.



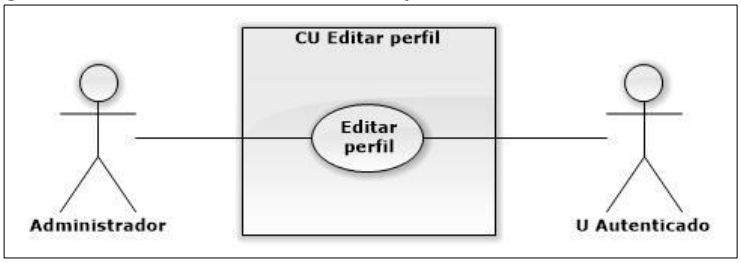
- Iniciar sesión

Figura 12: Diagrama Caso de uso - Iniciar sesión.



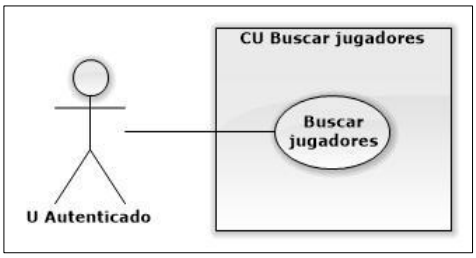
- Editar perfil

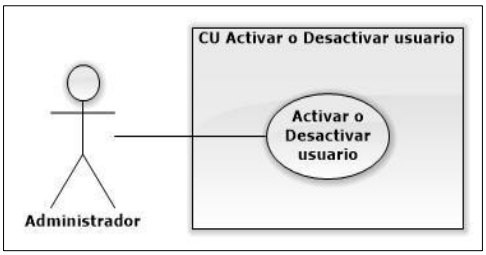
Figura 13: Diagrama Caso de uso - Editar perfil.



- Buscar jugadores | Activar o Desactivar usuario

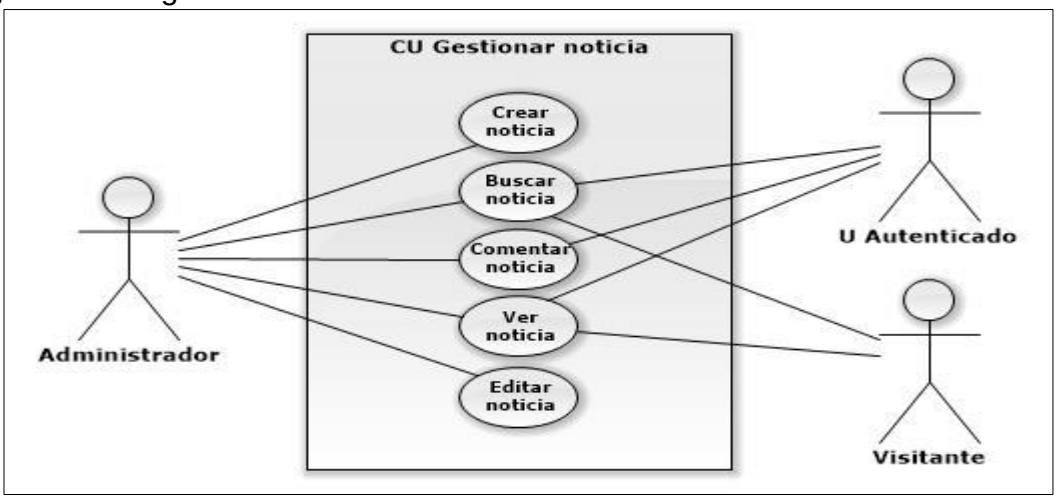
Figura 14: Diagrama Casos de uso - Buscar jugadores - Activar o Desactivar usuario





- Gestionar noticia y Gestionar artículo

Figura 15: Diagrama Caso de uso - Gestionar noticia.



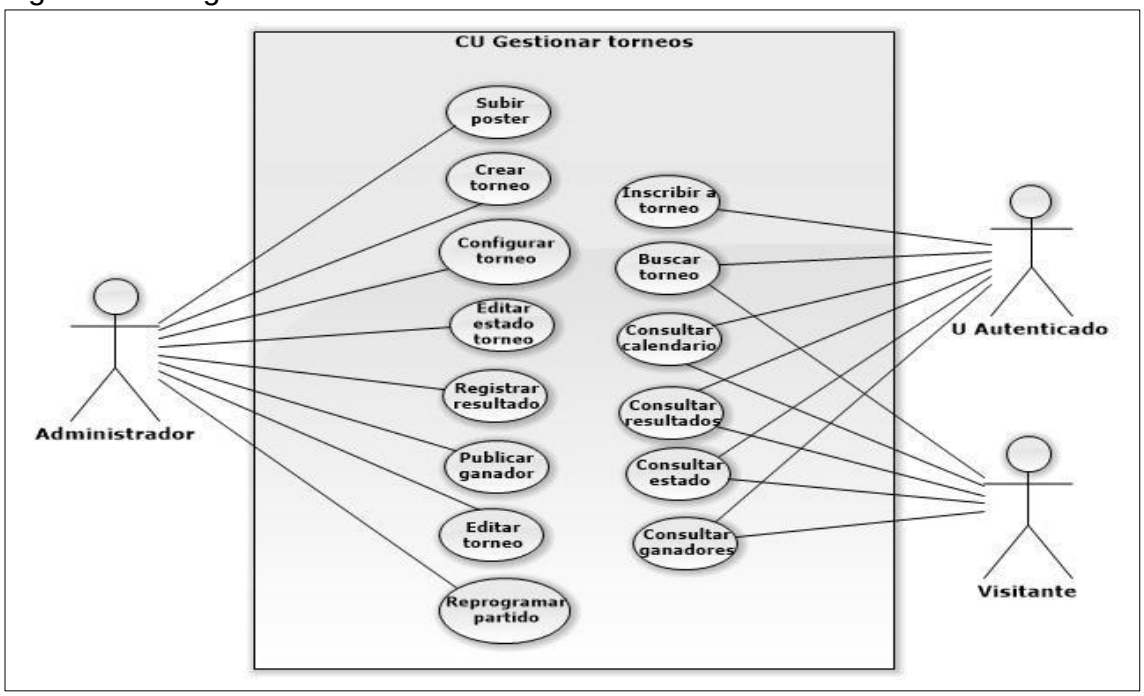
Nota: El diagrama del caso de uso Gestionar artículo funciona igual que el de Gestionar noticia. Dentro del software tienen el mismo manejo los dos ítems.

- Gestionar Información Institución

Figura 16: Diagrama Caso de uso - Gestionar información Institución.

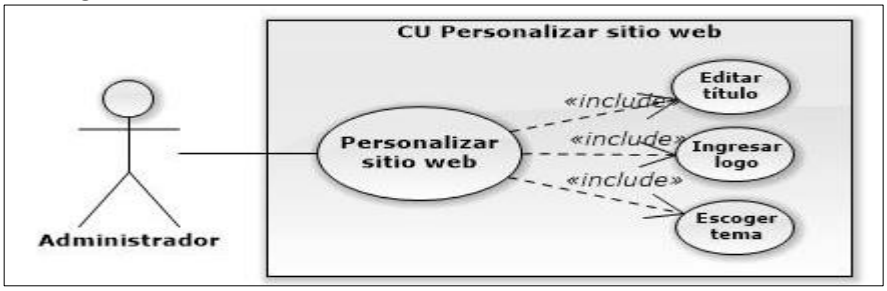
- Gestionar torneos

Figura 17: Diagrama Caso de uso - Gestionar torneos.



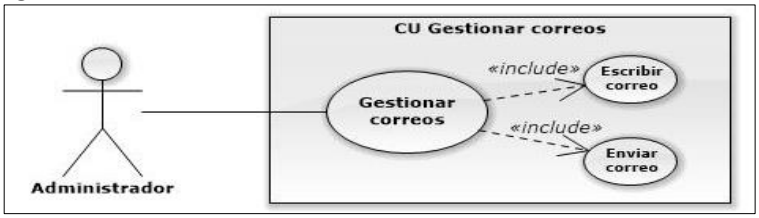
- Personalizar sitio web

Figura 18: Diagrama Caso de uso - Personalizar sitio web.



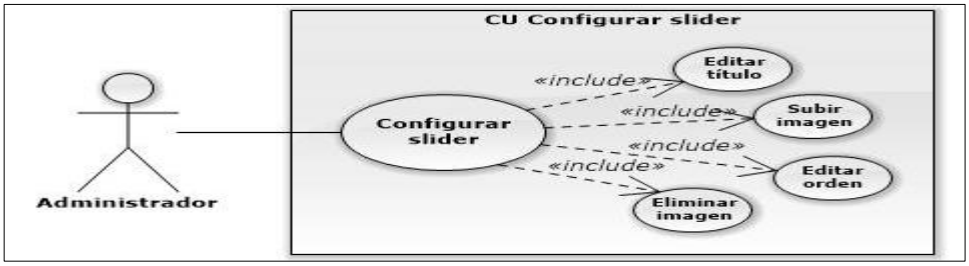
- Gestionar correos

Figura 19: Diagrama Caso de uso - Gestionar correos.



- Configurar slider

Figura 20: Diagrama Caso de uso - Configurar slider.



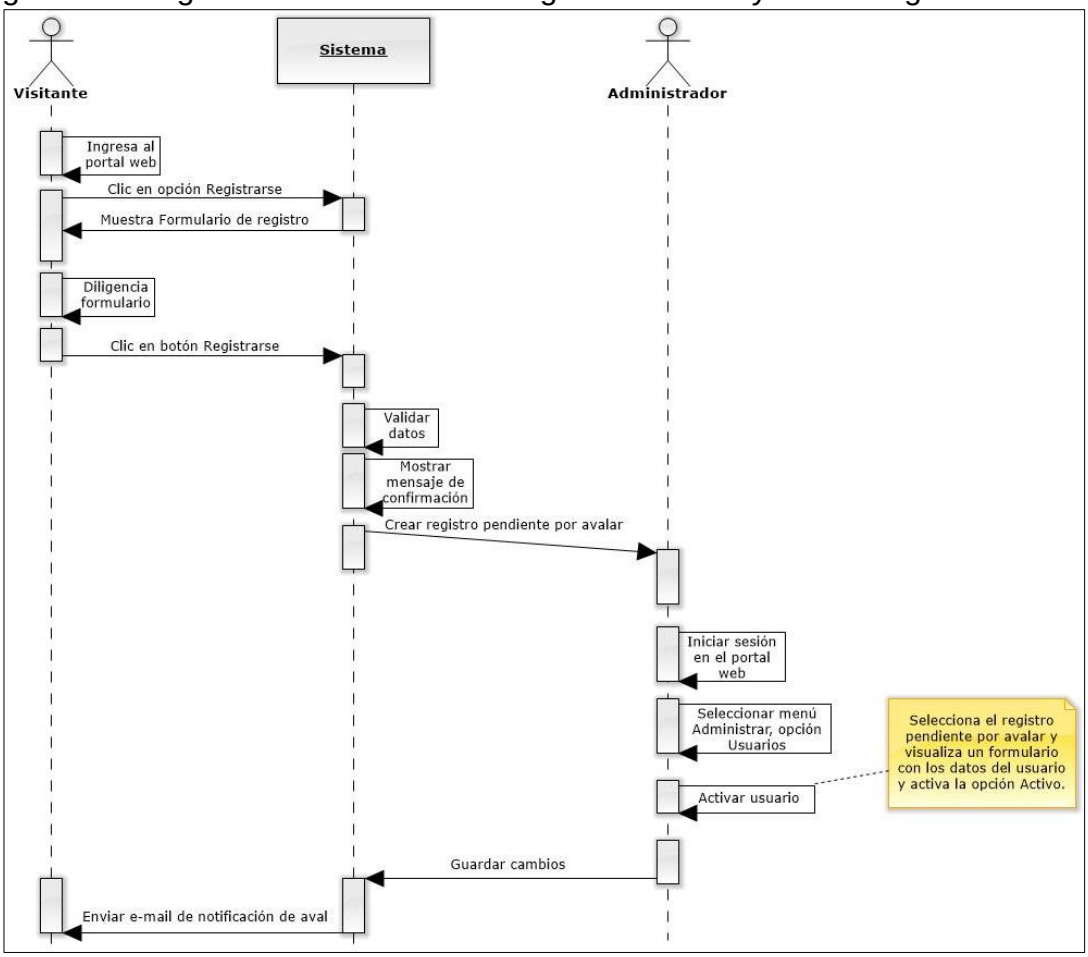
- Consultar información deporte

Figura 21: Diagrama Caso de uso - Consultar información deporte.

4.3.3.4 Diagramas de secuencia

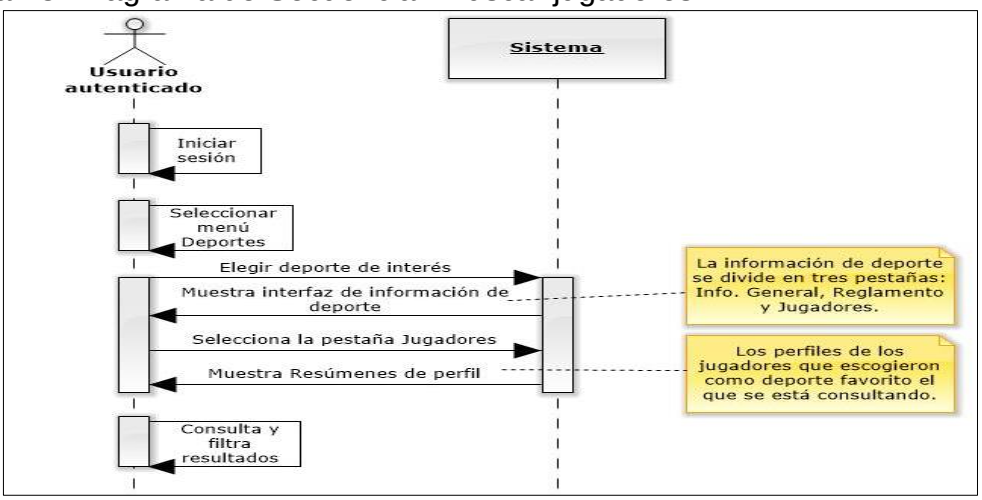
- Registrar usuario y avalar registro

Figura 22: Diagrama de Secuencia - Registrar usuario y Avalar registro.



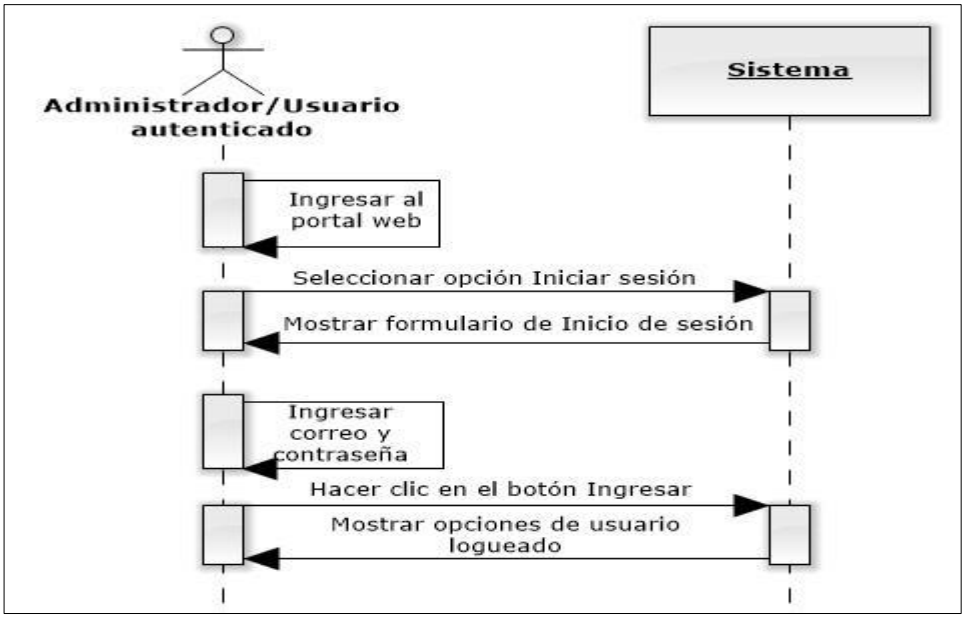
- Buscar jugadores

Figura 23: Diagrama de Secuencia - Buscar jugadores.



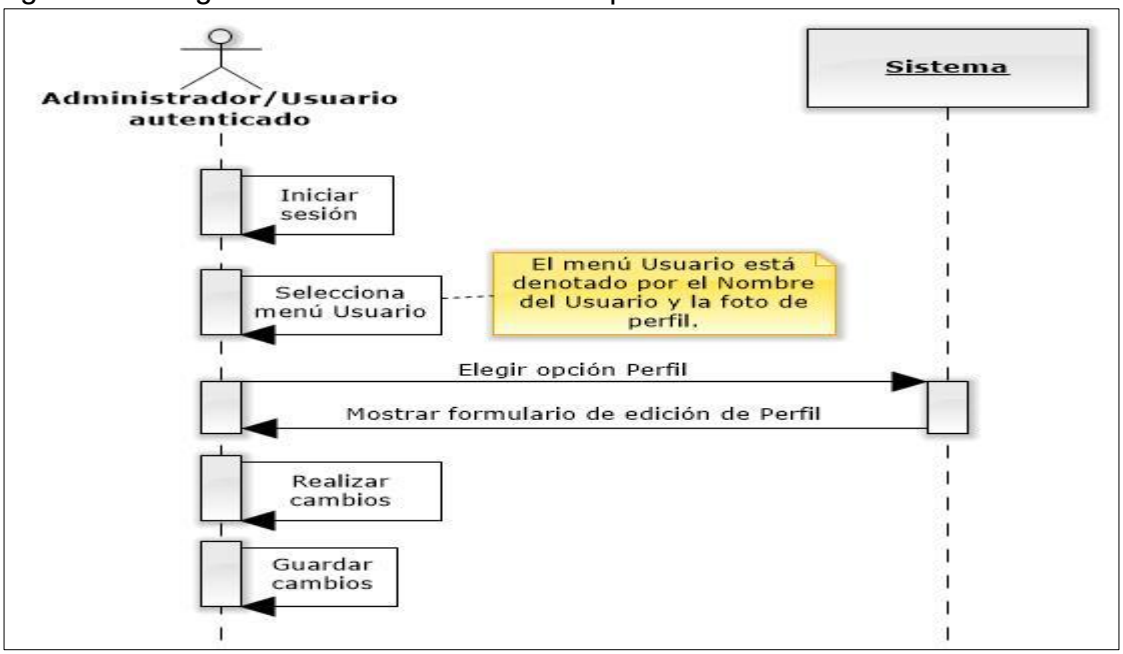
- Iniciar sesión

Figura 24: Diagrama de Secuencia - Iniciar sesión.



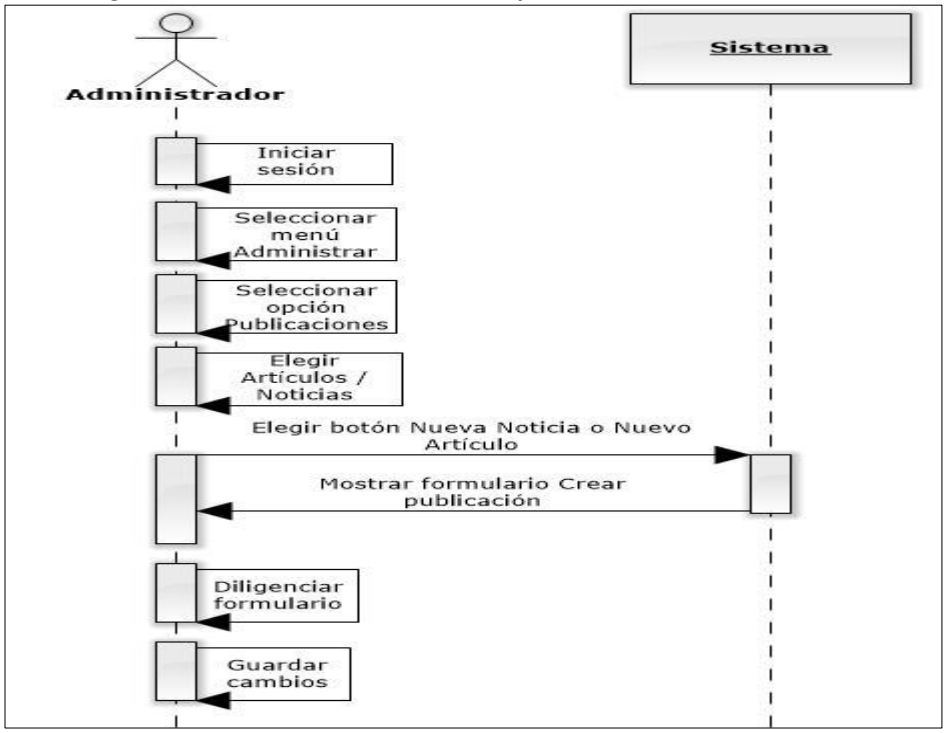
- Editar perfil

Figura 25: Diagrama de Secuencia - Editar perfil.



- Crear publicación

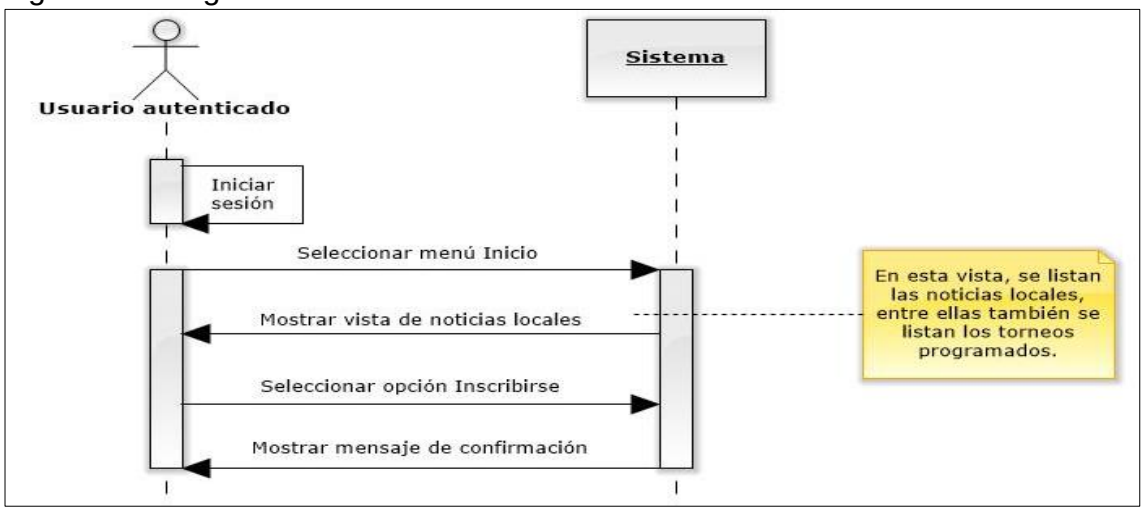
Figura 26: Diagrama de Secuencia - Crear publicación.



Nota: El proceso de creación de ítems por parte del administrador es relativo para todos los casos. Refiérase el diagrama de secuencia de la Figura 26 para estudiar los casos de creación de: comentarios en publicaciones (noticias o artículos), ajustes al portal web, sliders o banners, institución e imágenes en la galería institucional.

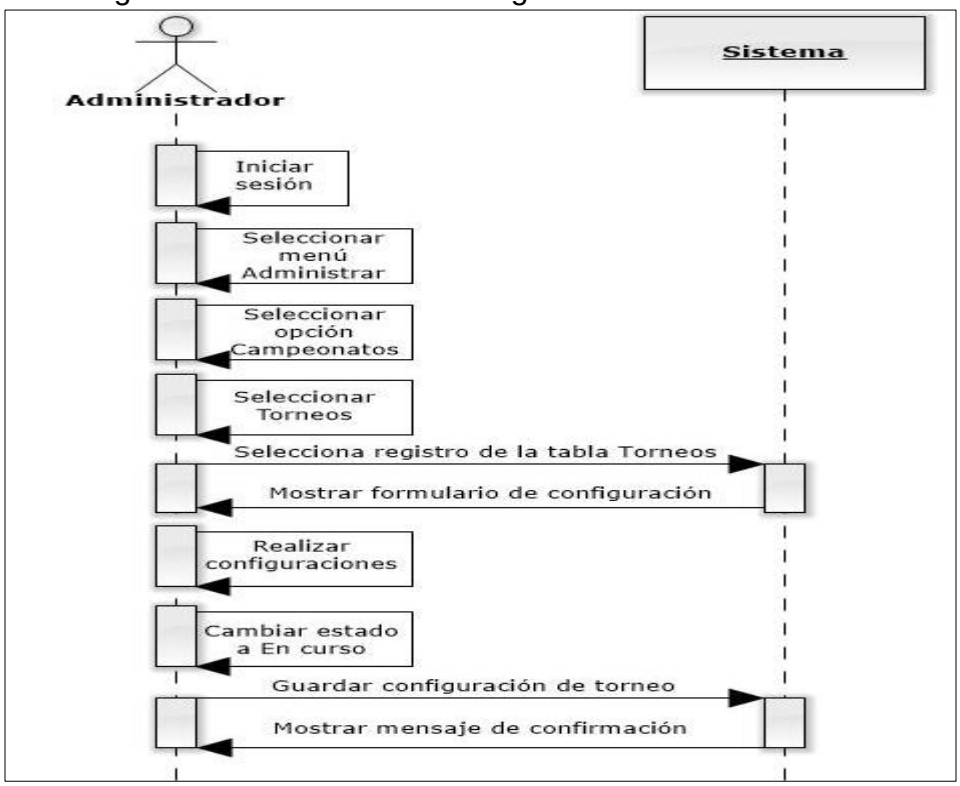
- Inscribir a torneo

Figura 27: Diagrama de Secuencia - Inscribir a torneo.



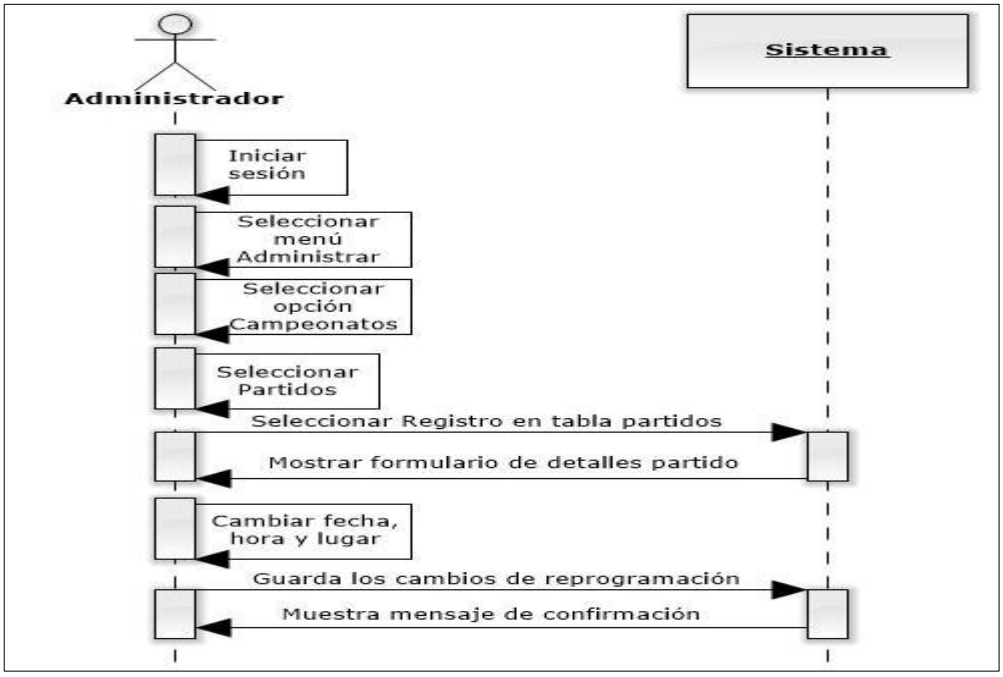
- Configurar torneo

Figura 28: Diagrama de Secuencia - Configurar torneo.



- Reprogramar partido

Figura 29: Diagrama de Secuencia - Reprogramar partido.



4.3.3 Modelo de datos. La última tarea fue construir el modelo de datos que soporta el sistema.

- Usuarios: Son las tablas que representan la administración de todo lo concerniente a los usuarios registrados en el portal. Dichas tablas son:
  - user: guarda los atributos correspondientes a los usuarios.
  - role: guarda los roles que puede tener un usuario en el portal web.
  - category: representa las categorías a las cuales están asociados los usuarios registrados en el deporte tenis.
  - sport: guarda atributos correspondientes a los deportes a los cuales se da soporte de información.
  - user_sport: guarda los deportes a los cuales están asociados los usuarios.

- Publicaciones: son las tablas que denotan las actividades de administración de las publicaciones hechas desde la misma institución. Dichas tablas son:
  - article: guarda los artículos de salud publicados por el administrador.
  - news: guarda las noticias locales publicadas por el administrador.
  - article_comments: guarda los comentarios realizados en artículos de salud publicados por el administrador.
  - news_comments: guarda los comentarios realizados en noticias locales publicados por el administrador.

- Torneos: tablas que representan la creación y administración de un torneo de tenis. La actividad más representativa es la programación automática de los encuentros del torneo. Dichas tablas son:
  - tournament: guarda los atributos propios de los torneos creados.
  - user_tournament: guarda los registros de usuarios que se inscriben a torneos.
  - tournament_category: guarda las categorías, cuyos jugadores pueden participar en un torneo.
  - group: guarda los grupos creados en la fase de grupos.
  - game: guarda los partidos de un torneo creados por el sistema.
  - game_type: indica el tipo de partido actual, es decir, fase de grupos, eliminatorias (octavos de final, cuartos de final, según sea necesario), semifinal, o final.
  - set: guarda la información del set actual: marcadores y si es tipo tie break o no.
  - points: guarda los puntos que el administrador destina para los jugadores de un torneo de acuerdo al puesto que ocupe.

- Personalización Institución: Debido a que el modelo de distribución del sistema es SaaS, está programado para ser personalizado de acuerdo a la institución que tome el servicio. Este modelo de datos presenta las tablas que permiten dicha personalización:
  - institution: guarda la información propia de la institución: nombre, misión, visión, acerca de y una imagen.
  - institution_galery: guarda las imágenes de la sección de galería.
  - config_website: guarda los parámetros de la personalización del sitio web, cuyo fin es que sea acorde con la institución.
  - banner: en puntos estratégicos del portal web habrá banners o sliders informativos. La tabla guarda las imágenes, con nombre y el orden de las mismas dentro del slider.

El Diagrama Entidad-Relación obtenido se encuentra en el ANEXO C.

4.4 REVISIÓN

Se da por terminado el *sprint* y se procede a ejecutar el *sprint review,* en el cual se reúnen el *product owner* (director de proyecto) y el equipo de desarrollo (autores del proyecto), con el fin de fomentar la utilización del producto por parte de los *stakeholder* (en este caso fueron instituciones interesadas) en el producto (software) y extraer información para el próximo s*print* o incremento.

En compañía del director del proyecto se realizó la revisión de los resultados obtenidos y se concluyó que las actividades propuestas se cumplieron a cabalidad y en el tiempo destinado, a pesar de haber tenido el riesgo de la difícil condensación de ideas. Por lo tanto, teniendo la etapa de análisis de requerimientos y diseño del sistema, terminados, se decidió dar inicio al segundo *sprint* planeando las actividades de la implementación del software.

4.5 REFLEXIÓN

En la elaboración de las actividades propuestas en el *sprint* se presentaron inconvenientes en la documentación de cada paso; se incurría en redundancias e inconsistencias en la claridad de los procesos, debido a la poca experiencia en documentación por parte de los autores. Es necesario el manejo basto en formas de lenguaje como conectores, sinónimos, tiempo, persona, género, entre otras.

Sin embargo, cabe destacar la disposición entregada por cada uno de los autores para lograr el objetivo propuesto, el cual se definía en obtener una base concreta y sólida para iniciar la construcción y solución con la misma calidad.

5 SEGUNDO SPRINT O INCREMENTO: DESARROLLO DEL FRONT-END DEL SOFTWARE

5.1 COMUNICACIÓN.

Para este segundo *sprint* se discutió acerca de:

- Dedicar un tiempo para la capacitación de las tecnologías necesarias para el desarrollo del *front-end* del software.
- El equipo *SCRUM* interactuó con el *product owner* para exponer sus decisiones y recibir aporte del mismo y de los *stakeholders* acerca del *front-end.*
- El equipo de desarrolladores decidió realizar pequeños mockups en papel para proyectar un posible diseño de la interfaz del software web
- Se arrancó con el desarrollo de la totalidad de la interfaz.

Para una puesta en marcha ordenada de las tareas discutidas para este segundo *sprint* se realizó lo siguiente:

5.1.1 Definir el material de capacitación

- http://getbootstrap.com/
- http://laravel.com/docs/4.2
- https://www.jetbrains.com/phpstorm/
- Diversos vídeos tutoriales propuestos en youtube.com.
- Libro: HTML5.[8]

5.2 PLANEACIÓN

5.2.1 Tabla de riesgos

Tabla 2 Evaluación de riesgos para el segundo *Sprint.*

RIESGO	PROBABILIDAD	IMPACTO
Personas		
Poca experiencia con *HTML5*	90%	3
Poca experiencia con *CSS3*	90%	3
Poca experiencia con *JavaScript*	90%	3
Poca experiencia con *Bootstrap*	90%	3

[8] ALVAREZ GARCÍA, Alonso. HTML5. Manual imprescindible. Madrid: Anaya Multimedia, 2012.

Poca experiencia con *Laravel*	90%	3
Miembro del equipo de desarrolladores de enferme	50%	2
Miembro del equipo del proyecto se desvincule del mismo	20%	3
Product owner requiera cambios en el *Sprint*	40%	2
Calificadores establezcan correcciones en *Sprint*	30%	2
Producto		
Tecnologías elegidas no adecuadas para el *Sprint*	30%	2
Los temas del software no sean suficientes para lograr satisfacer al *product owner* y *stakeholders*.	50%	2
Proceso		
El alcance del *Sprint* no sea apropiado.	30%	3
Se realicen muchas actividades en comunicación que tomen mucho tiempo en el *Sprint*	50%	3
La tecnología usada no se adapta fácilmente a las tareas de las actividades del *Sprint*	30%	2

Para el desarrollo de este incremento se contempló asignar el máximo tiempo permitido para la realización de un *sprint,* un (1) mes, dada la complejidad, múltiples tareas y baja experiencia con las tecnologías y lenguajes a usar.

5.3 DESARROLLO

Tomando las tareas de las acciones anteriores como cimiento para un desarrollo con calidad, se da paso a crear el *front-end* (interfaz) del software sobre el cual se trabaja.

Se hacen refinamientos a los *mockups* en papel, para iniciar el desarrollo del software. Para ello se establecieron dos principios propuestos en el libro de Pressman para la actividad del diseño: diseño de la interfaz y de la información.

5.3.1 Diseño de la interfaz. Para la creación de una interfaz de calidad se tienen en cuenta los siguientes principios:

- Anticipación: El software se anticipa al hecho de que el usuario se vea confundido, por ello proporciona mensajes que facilitan la navegación del usuario y por ende su interacción con el software.
- Comunicación: El software maneja texto claro en cuanto a las funciones en cada uno de sus botones; así mismo con un log in exitoso del usuario, comunica que el estado de este es ACTIVO.

- Consistencia: Algunos títulos y botones contienen iconos que son coherentes a sus funciones, proporcionando al usuario más claridad.
- Eficiencia: El software es eficaz en cuanto a la posibilidad de generar encuentros individuales entre los miembros del sitio, proporcionando información clave para un contacto rápido.
- Flexibilidad: El software ofrece la posibilidad de corregir errores, si estos se cometieron al momento de registrarse, además notamos que si durante la navegación se cometieron errores en alguna ruta (URL) hay la posibilidad de devolvernos.
- Reducción de latencia: El software genera tiempos de respuesta mínimos en sus consultas de navegación, lo que reduce la latencia del sitio.
- Habilidad de aprender: El software ofrece un aprendizaje en el usuario dado que presenta un diseño intuitivo en donde la organización de los botones y con su respectiva funcionalidad sea obvia.
- Habilidad de leer: El software, ofrece colores, fondos, tipos y tamaños de letra, agradables para la navegación y salud visual del usuario.

5.3.2 Diseño de la información. Tomando como referencia el libro de Pressman[9], el cual nos da a conocer cuatro (4) estructuras importantes para el diseño de información (lineal, jerárquica, red y matriz), concluimos que el diagrama adaptativo de nuestro proyecto es el jerárquico puesto que el usuario tiene acceso a un menú principal y dependiendo de su rol puede navegar en varios de sus pestañas (interfaces) y profundizar la estructura.

Presentando más en concreto la construcción del software y teniendo claro el modelado de la problemática que se quiere atacar y aún las tecnologías a usar para la implementación, se da paso a las siguientes tareas:

5.3.2.1 Tareas base.

- Elección del entorno de desarrollo (IDE).
- Elección del servidor web.
- Elección de los leguajes de programación adecuados para el diseño planteado.
- Elección de *framework* necesario para el mapeo objeto-relacional y creación del *front-end* (interfaces gráficas).
- Elección del control de versiones.

[9] PRESSMAN, Roger. Ingeniería del software. Un enfoque práctico. 7 ed. México: McGraw Hill, 2010.

5.3.2.2 Tareas de la arquitectura.

- Instalación de *framework.*
- Comenzar con la construcción de la interfaz gráfica
- Pruebas de los componentes de la interfaz gráfica.

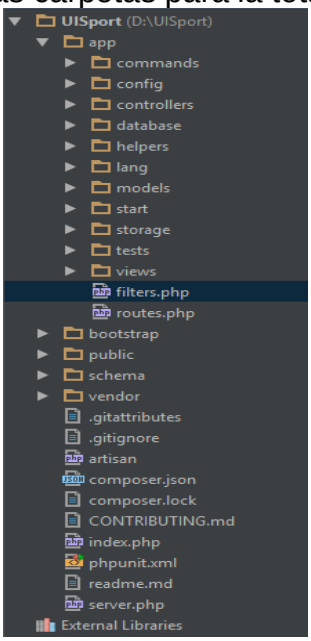
Existen tareas de bajo nivel y tareas de integración implementadas en el próximo *sprint* dado la importancia dentro de este.

5.3.2.3 Herramientas y librerías usadas para el desarrollo del Sprint.

- Jetbrains PhpStorm: las funciones usadas de este son PHP 5, servidor web Apache (versión 2), control de versiones (GIT).
- Diseño de interfaz: Laravel framework (versión 4.2) en conjunto con Bootstrap (versión 3.0).
- Desarrollo de la interfaz: HTML5, JavaScript, jQuery y hojas de estilos CSS3.
- Repositorio remoto: Bitbucked.

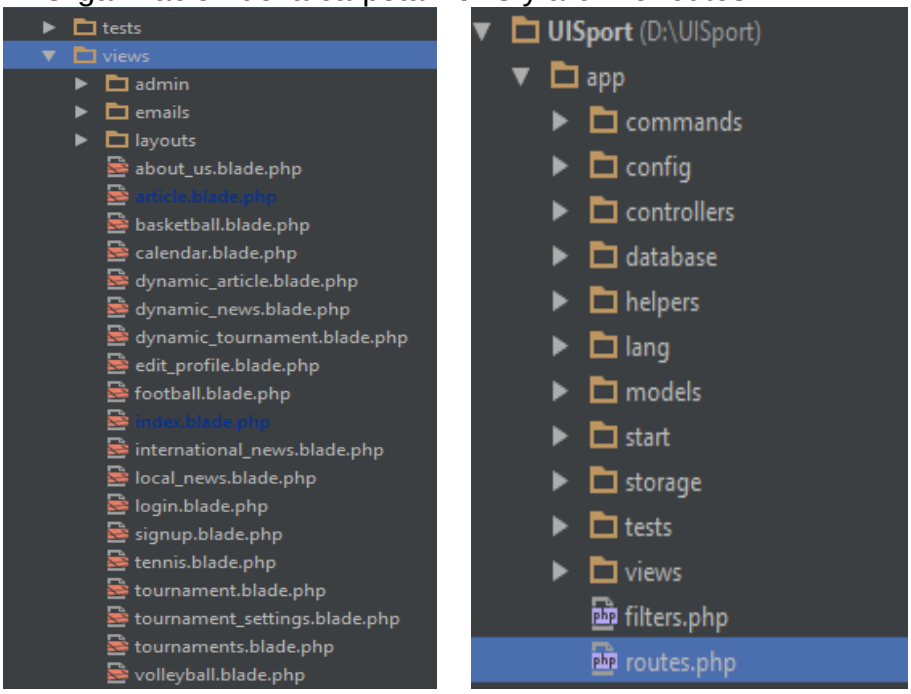
5.3.2.4 Distribución de las carpetas dentro del proyecto en Jetbrains PhpStorm. El proyecto en su totalidad abarca las siguientes carpetas:

Figura 30: Organización de las carpetas para la totalidad del proyecto.



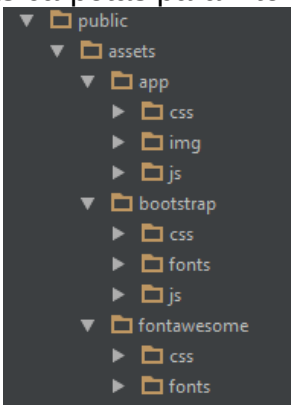
En la carpeta ***Views*** que se encuentra dentro de la carpeta ***App*** se alojan todas las vistas del *front-end* del software, así mismo se alojan las rutas o URL que se asignaron a cada vista.

Figura 31: Organización de la carpeta *views* y archivo *routes*.



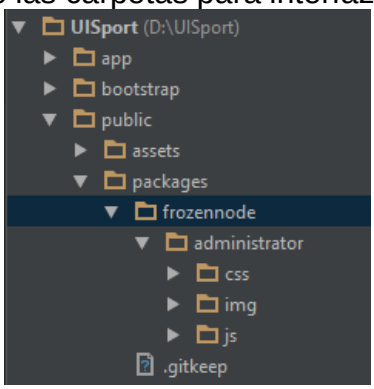
En la carpeta ***assets*** de ***Public*** observamos tres ramas, las cuales corresponden a: carpeta **app**, la cual contiene los archivos personalizados (propios) necesarios para una interfaz agradable al usuario, tales *como css, img* (archivos de imagen estáticos en el software, necesarios para el desarrollo), js, carpeta ***bootstrap***, la cual contiene archivos para una interfaz agradable, propios de *bootstrap*, como *css, fonts, js*, la carpeta ***fontawesome***, la cual posee archivos como css y fonts, necesarios para los iconos que se encuentran en títulos, subtítulos y botones de la interfaz.

Figura 32: Organización de las carpetas para interfaz agradable



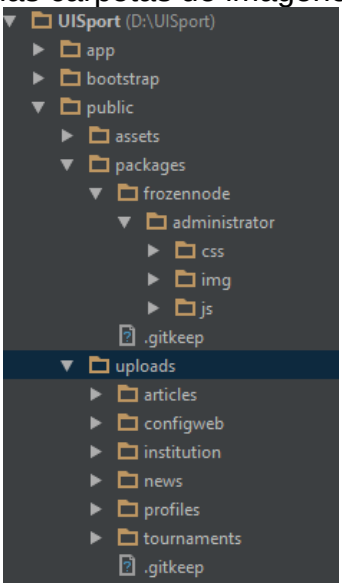
En la carpeta ***packages*** se almacena un paquete ofrecido por el *framework* Laravel llamado ***frozennode***, el cual posee un archivo llamado ***administrator*** que contiene los *css, img* (imágenes propias del paquete) y js usados para la vista de administración del software web proporcionada únicamente al usuario administrador.

Figura 33: Organización de las carpetas para interfaz de administración.



La carpeta ***uploads*** almacena todo el contenido de archivos imagen dinámicas del software web, es decir, aglomera de manera ordenada todas las imágenes suministradas al sitio desde la parte administrativa.

Figura 34: Organización de las carpetas de imágenes dinámicas del sitio.



5.3.3 Resultados de la parte Front-end. El resultado para el segundo *sprint* es la finalización de toda la parte visual, tanto para los usuarios en general, como para la parte administrativa. Las siguientes capturas muestran la visualización final para el usuario.

- Vistas del *front-end* para los usuarios. (Figuras 35 - 40)

Figura 35: Vista menú Inicio para usuario administrador.

Nota: La vista para usuarios autenticados y visitantes varia en que no tiene permiso para los menú Administrar y Correos.

Figura 36: Vista de comentarios a publicaciones.

Figura 37: Vista menú Nosotros (Acerca de)

Figura 38: Vista menú Nosotros (Galería).

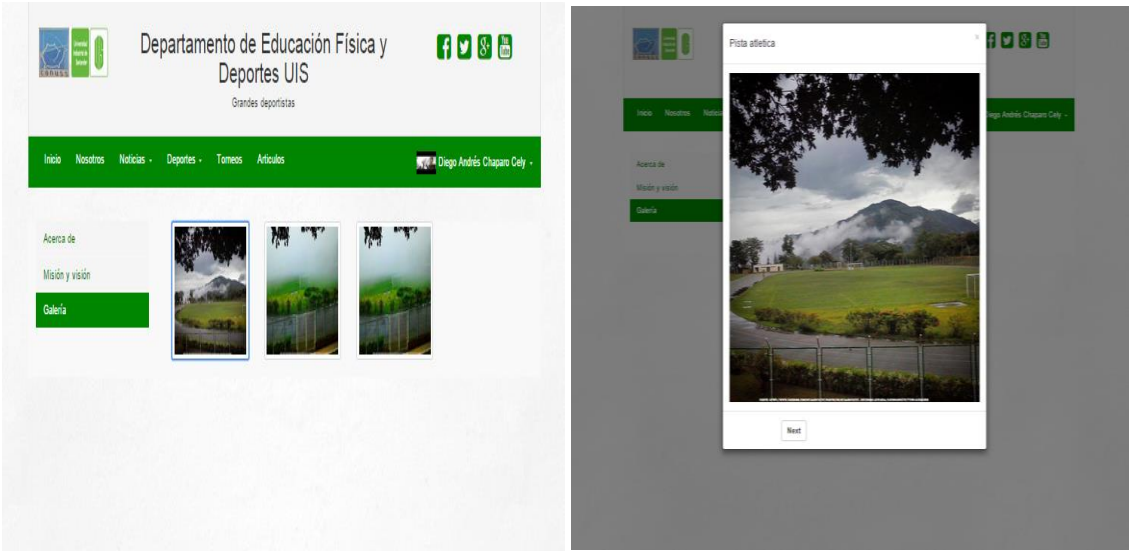


Figura 39: Vista menú Deportes (Tenis/Jugadores)

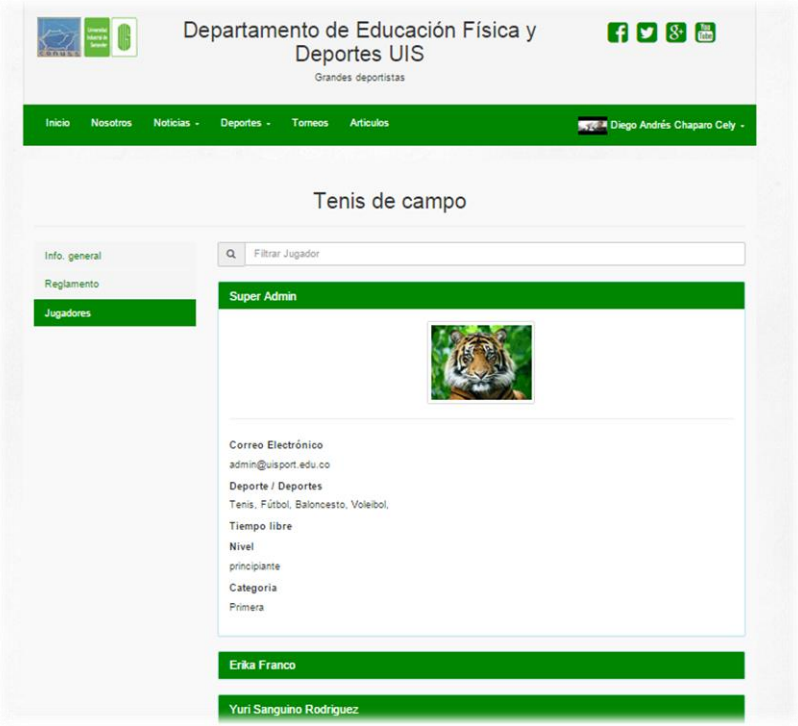


Figura 40: Vista menú Artículos (Notas de salud).

- Vistas del *front-end* para la parte administrativa (Figuras: 41 - 46)

Figura 41: Vista administrativa de la apariencia del software.

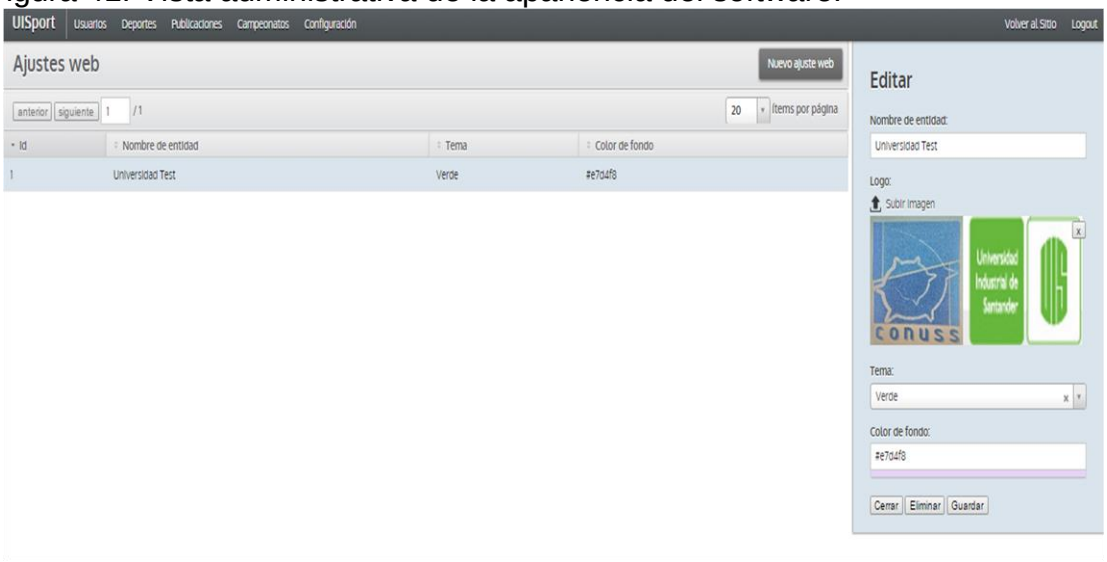


Figura 42: Vista administrativa de los usuarios de la plataforma.

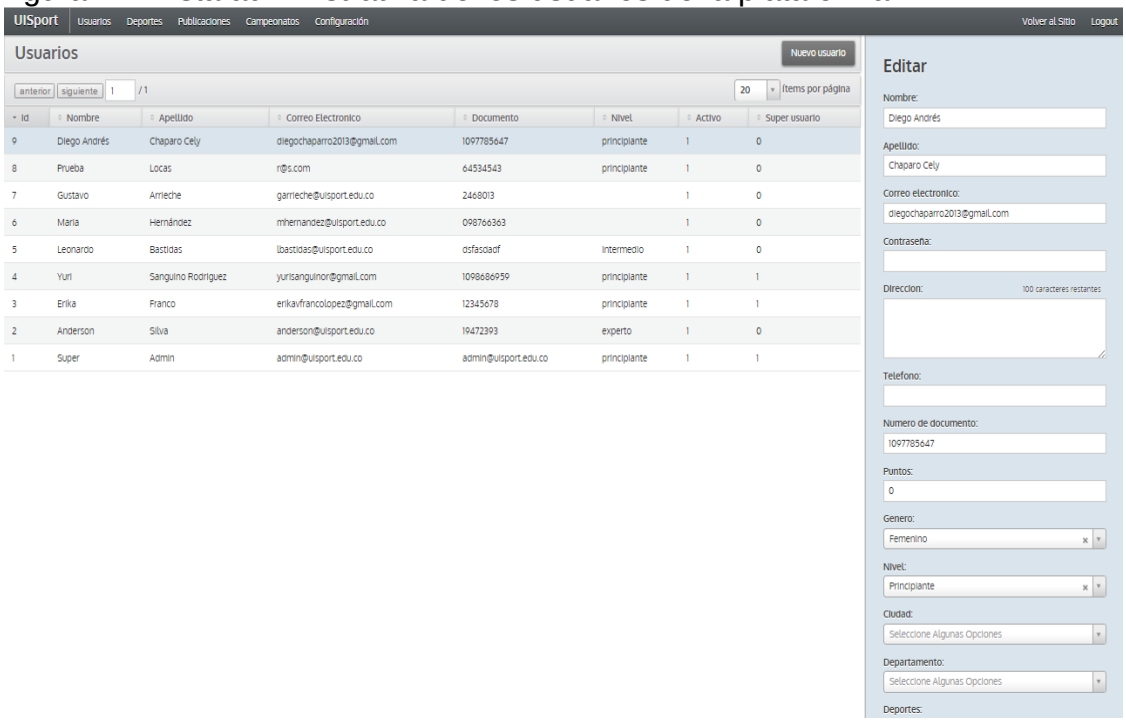


Figura 43: Vista administrativa del menú Deportes.

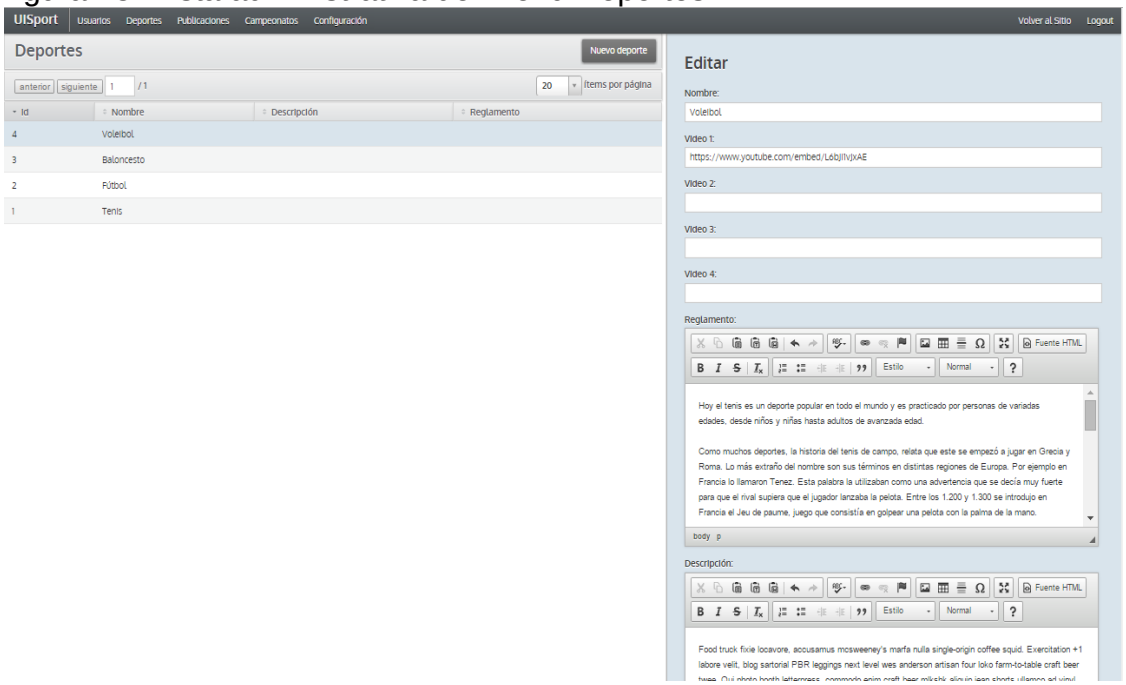


Figura 44: Vista administrativa Publicaciones.

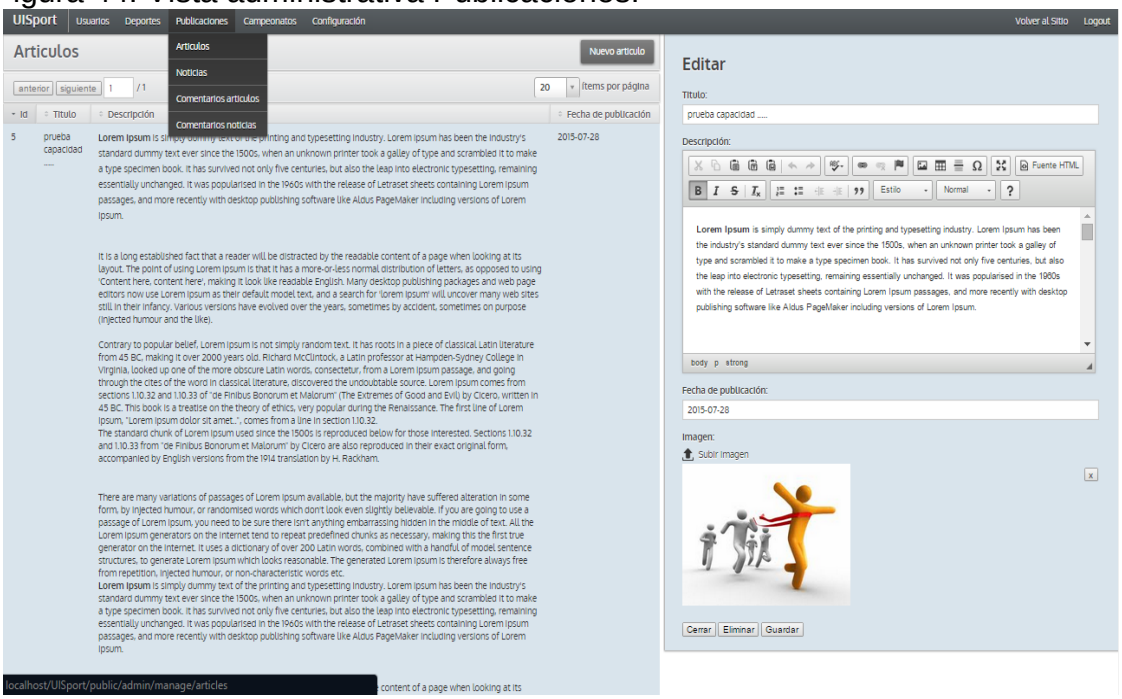


Figura 45: Vista administrativa del menú Torneos.

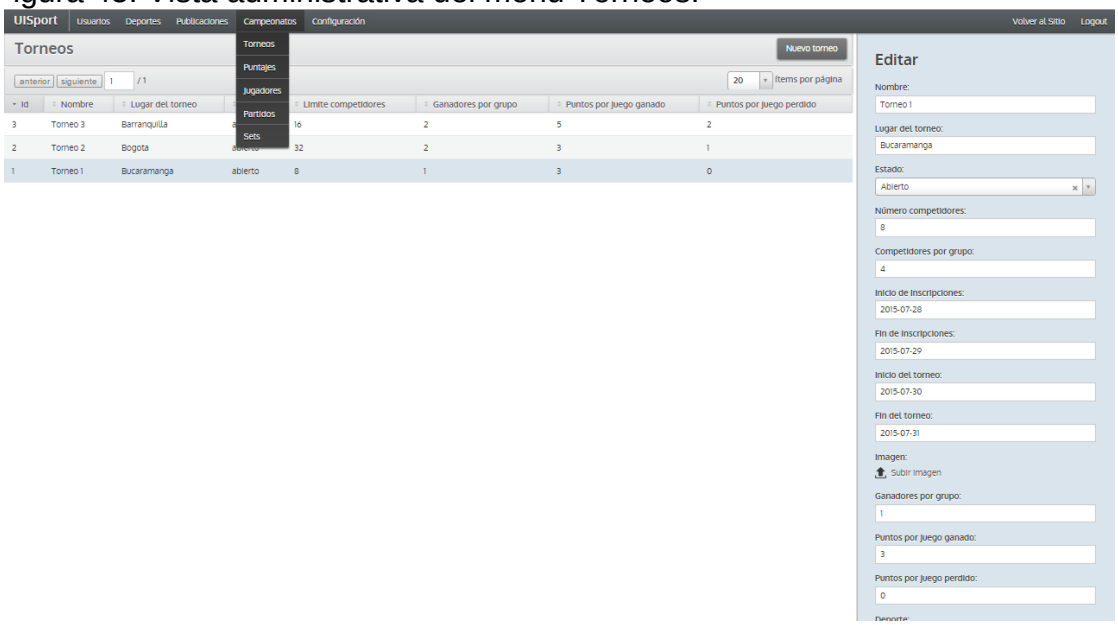
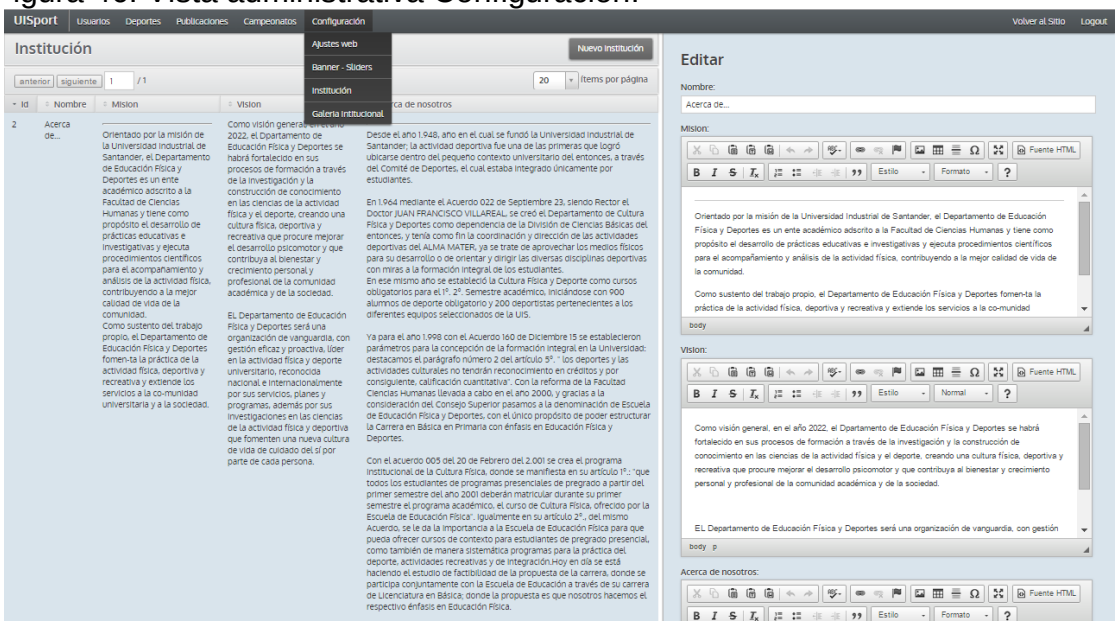


Figura 46: Vista administrativa Configuración.



Nota: Es de vital importancia dejar claro que durante el desarrollo se llevaron a cabo las reuniones diarias que propone la metodología, para avanzar en las tareas de cada desarrollador y concluir con éxito esta fase del incremento.

5.4 REVISIÓN

En esta etapa se da por terminado el *sprint* y se procede a ejecutar el *sprint review,* en el cual se reúnen el *product owner* (director de proyecto) y el equipo de desarrollo (autores del proyecto), con el fin de fomentar la utilización del producto por parte de los *stakeholder* o interesados (en este caso fue la comunidad universitaria interesada en deportes) en el producto (software) y extraer información para el próximo s*print* o incremento.

Todo ello nos permite evitar posibles riesgos que puedan ver afectados no solo el *sprint* en el que se trabajó sino también los próximos.

Se realizó la revisión y análisis del incremento en compañía del director de proyecto y se dio aval, puesto que se cumplían las metas propuestas y se concretó trabajar en el próximo *sprint* toda la parte *back-end* y dar por finalizado el desarrollo de la totalidad del software.

5.5 REFLEXIÓN

Aquí el equipo SCRUM, ejecuta la retrospectiva del *sprint* trabajado, en este caso se presentaron algunos inconvenientes con el trabajo de las tecnologías usadas dado que si surgía alguna inquietud respecto al trabajo con estas y se gestionaba una consulta del tema en la web, se encontraba documentación en inglés, lo cual fue bastante complejo por el bajo nivel de inglés que presentan los autores (equipo de desarrolladores).

Se consideró un aspecto a destacar, el acople del equipo de desarrolladores y los aportes de diseño presentados por cada uno de ellos.

6 TERCER *SPRINT* O INCREMENTO: DESARROLLO DEL BACK-END DEL SOFTWARE

6.1 COMUNICACIÓN.

Para este tercer *sprint* se discutió acerca de:

- De la misma manera que en todos los incrementos, se dedicó un tiempo para la capacitación de los lenguajes necesarios para el desarrollo del *back-end* del software.
- El equipo *SCRUM* interactuó con el *product owner* para exponer sus decisiones y recibir aportes lo más claro posibles y comenzar a reflejar la lógica en la programación del *back-end.*
- Se arrancó con el desarrollo de la lógica y programación interna del *back-end* con el objetivo de acoplar todo ello al *front-end.*

Para una puesta en marcha ordenada de las tareas discutidas para este segundo *sprint* se realizó lo siguiente:

6.1.1 Definir el material de capacitación

- http://www.w3schools.com/php/default.asp
- http://w3schools.com/js/
- http://laravel.com/docs/4.2
- https://www.jetbrains.com/phpstorm/
- Diversos vídeos tutoriales propuestos en youtube.com
- Libro: Javascript.[10]

6.2 PLANEACIÓN

6.2.1 Tabla de riesgos

Tabla 3 Evaluación de riesgos para el tercer *Sprint.*

RIESGO	PROBABILIDAD	IMPACTO
Personas		
Poca experiencia con *Laravel*	90%	3
Poca experiencia con *PHP*	90%	3
Poca experiencia con *JavaSCript*	90%	3

[10] SAWYER MCFARLAND, David. JavaScript. Traducción de JavaScript: The Missing Manual. Madrid: Anaya Multimedia, 2009.

Poca experiencia con *Bootstrap*	90%	3
Poca experiencia con *Laravel*	90%	3
Miembro del equipo de desarrolladores de enferme	50%	2
Miembro del equipo del proyecto se desvincule del mismo	20%	3
Product owner requiera cambios en el *Sprint*	40%	2
Calificadores establezcan correcciones en el *Sprint*	30%	2
Producto		
Las tecnologías elegidas no sean adecuadas para el *Sprint*	30%	2
Los temas del software no sean suficientes para lograr satisfacer al *product owner* y *stakeholders*.	50%	2
Proceso		
El alcance del *Sprint* no sea apropiado.	30%	2
Se realicen muchas actividades en comunicación que tomen mucho tiempo en el *Sprint*	50%	3
La tecnología usada no se adapta fácilmente a las tareas de las actividades del *Sprint*	30%	2

Para el desarrollo total de este incremento se contempló de igual manera, asignar el máximo, un (1) mes, dada la complejidad, múltiples tareas y baja experiencia con las tecnologías y lenguajes a usar.

6.3 DESARROLLO

Llevadas a cabo las tareas de las acciones anteriores y tomadas como cimiento para un desarrollo con calidad se da paso a crear el *back-end* (lógica y programación interna del software).

Teniendo cuenta que en el *sprint* anterior se definieron las tareas base y de arquitectura por ser propias y representativas de dicho incremento; aquí se definirán las tareas de bajo nivel y de integración.

6.3.1 Tareas de bajo nivel

- Codificación de los controladores de la interfaz gráfica y comunicación con el servidor.
- Construcción de mapeo objeto-relación.
- Codificación de los controladores del lado del servidor.
- Realización de pruebas unitarias para cada componente ya ensamblado.
- Refactorizar lo más que se pueda la lógica del código fuente ya creado.

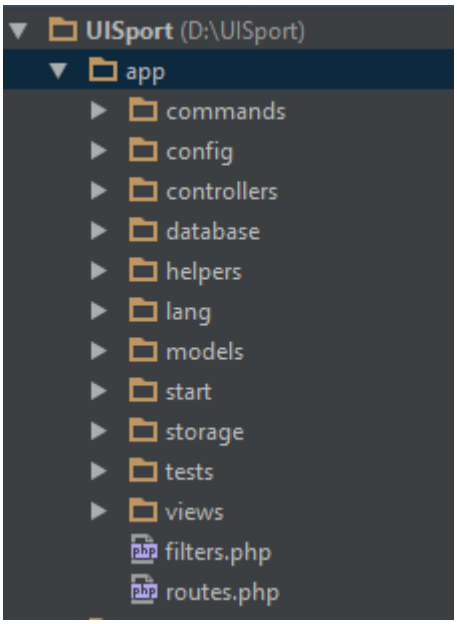
6.3.2 Tareas integración

- Integrar los componentes de lado del servidor con los de clientes.

Para la realización de las tareas, se trabajó sobre las herramientas puestas en marcha en el anterior *sprint*.

6.3.3 Distribución de las carpetas dentro del proyecto en PhpStorm. En la carpeta ***App*** se alojan las carpetas que contienen todo el trabajo realizado para el éxito de este incremento, dichas carpetas son: *config, controllers, helpers, lang, models, start,* entre otras.

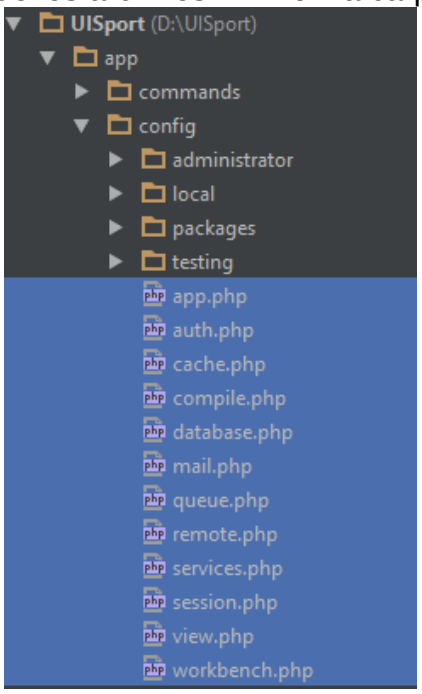
Figura 47: Organización de las carpetas del *back-end.*



- En la carpeta ***config*** de ***app*** observamos cuatro ramas y una serie de archivos php.

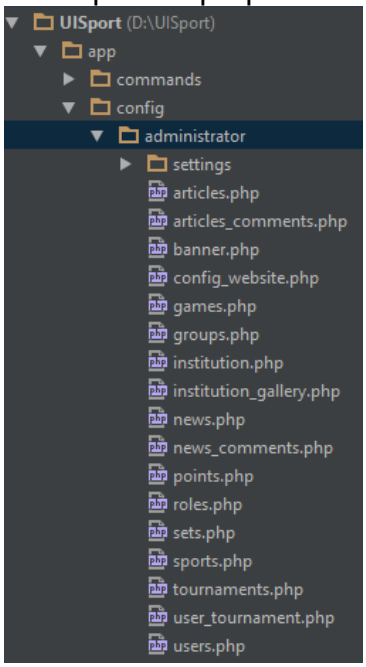
Los archivos PHP contenidos en la carpeta ***config*** pertenecen a toda la configuración necesaria para el desarrollo del *back-end**,*** tales como, configuración de la aplicación, paquetes con clases principales, conexión a la base de datos, usuario y password para el disparo de correos a usuarios autenticados, entre otras.

Figura 48: Organización de los archivos PHP en la carpeta *config.*



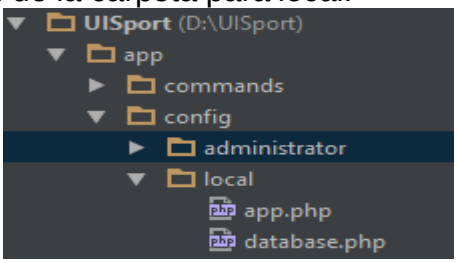
- La carpeta ***administrator*** es el módulo o paquete ***administrator*** que se usó para la parte administrativa del software, esta los archivos PHP correspondientes a la implementación de dicho paquete y reflejado en las vistas correspondientes a la parte administrativa.

Figura 49: Organización de archivos para el paquete *administrator.*



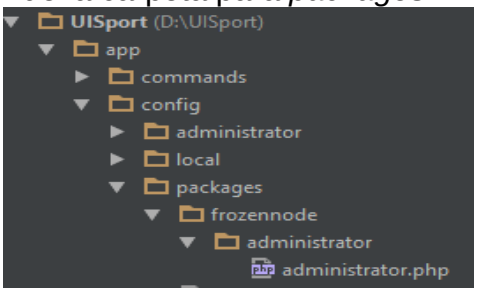
- La carpeta ***local*** almacena archivos de configuración de la base de datos.

Figura 50: Organización de la carpeta para *local.*



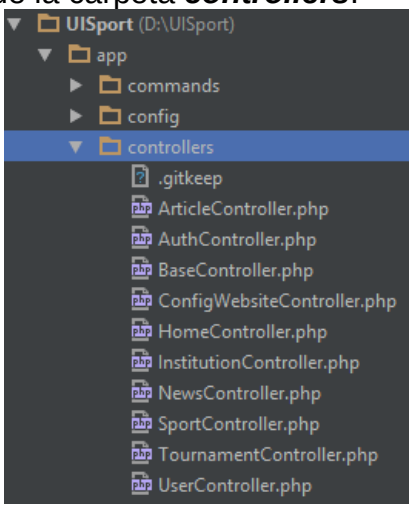
- La carpeta ***packages*** contiene una pequeña configuración del paquete ***administrator.***

Figura 51: Organización de la carpeta para *packages.*



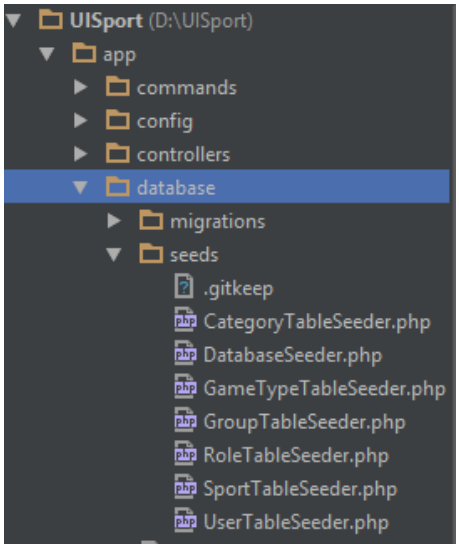
- La carpeta ***controllers*** contiene todas las funciones necesarias para unir el ***front-end*** con el *back-end,* esto es posible gracias a las variables creadas en la base de datos.

Figura 52: Organización de la carpeta ***controllers***.



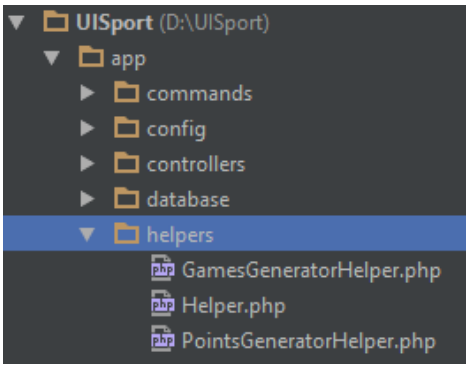
- Dentro de la carpeta ***database,*** encontramos las carpetas ***migrations,*** la cual contiene archivos migrados de otros programas o herramientas, para este caso NO se hizo necesario el uso de esta y la carpeta ***seeds*** contiene la configuración de aquellas tablas de la base de datos con valores estáticos, cabe resaltar que para nuevas versiones se pueden crear nuevos datos.

Figura 53: Organización de las carpetas de *database*



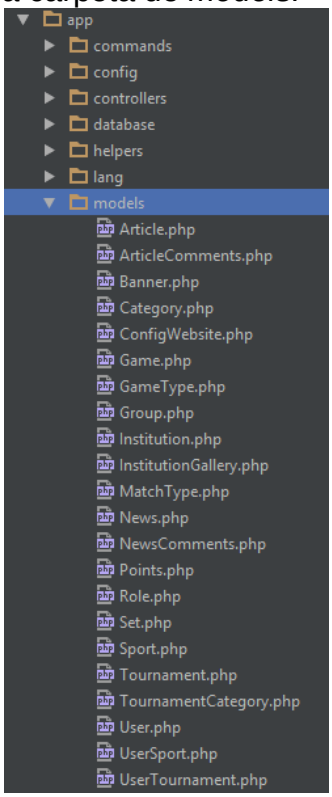
- La carpeta ***helpers*** contiene aquellos archivos PHP que implementan la programación de todo lo relacionado a toneos.

Figura 54: Organización de la carpeta de ***helpers***.



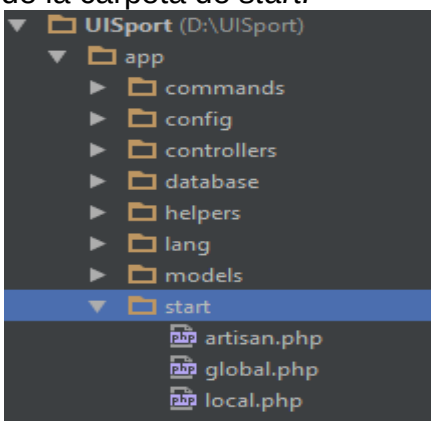
- La carpeta ***models*** contienes todos los modelos representativos a la base de datos, lo cual permite la comunicación entre las vistas creadas en el anterior *sprint* y los controladores, esto con el fin de lograr un modelo vista-controlador exitoso.

Figura 55: Organización de la carpeta de *models.*



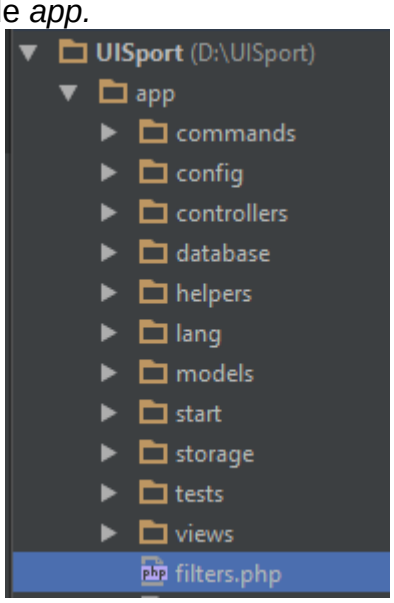
- La carpeta ***start*** contiene todos los archivos que son llamados cuando se inicia el *framework* y durante la producción del software NO son modificados; estos permiten manejar el *framework* por medio de comandos.

Figura 56: Organización de la carpeta de *start.*



- En el archivo PHP, ***filters***, contenido dentro de la carpeta ***app***, se implementan los permisos de cada tipo de usuario para la navegación del software, dichos permisos se encuentran documentados en el primer *sprint.*

Figura 57: Archivo *filters* de *app.*



Nota: Es de vital importancia dejar claro que durante el desarrollo se llevaron a cabo las reuniones diarias que propone la metodología, para avanzar en las tareas de cada desarrollador y concluir con éxito esta fase del incremento.

6.4 REVISIÓN

Se da por terminado el *sprint* y se procede a ejecutar el *sprint review,* en el cual se reúnen el *product owner* (director de proyecto) y el equipo de desarrollo (autores del proyecto), con el fin de fomentar la utilización del producto por parte de los *stakeholder* o interesados (en este caso fue la comunidad universitaria interesada en deportes) en el producto (software).

Como ya se alcanzaron todos y cada uno de los objetivos, no se contempla la planeación de un próximo *sprint.*

Se realizó la revisión y análisis del incremento en compañía del director de proyecto y se dio aval, puesto que se cumplían las metas propuestas.

En el capítulo de **Pruebas** se realizará un análisis exhaustivo en compañía del director de proyecto con el fin de evaluar el software en su totalidad.

6.5 REFLEXIÓN

Aquí el equipo SCRUM, ejecuta la retrospectiva del *sprint* trabajado, en este caso se volvieron a presentar dificultades con documentaciones en inglés, pero ya con menos influencia sobre la ejecución del sprint.

Se destaca nuevamente el acople del equipo de desarrolladores y el gran aprendizaje que se tuvo durante el desarrollo del proyecto.

7 PRUEBAS Y ANÁLISIS DE RESULTADOS

Para la realización del proceso de pruebas en la totalidad del software, nos basamos en el libro de Pressman, en el cual nos sugieren pruebas en nueve (9) dimensiones (contenido, función, estructura, usabilidad, navegabilidad, desempeño, compatibilidad, interoperabilidad, seguridad) para conseguir un proyecto de calidad.

7.1 PRUEBAS DE CONTENIDO

Para un análisis apropiado a este tipo de pruebas, se dará respuesta a una serie de preguntas propuestas por el libro.

- ¿Está la información actualizada y de manera correcta (Gramáticamente)?
  Se revisó todo el software y se llegó a la conclusión de que la información está actualizada; se encontraron algunos errores de ortografía, pero fueron corregidos inmediatamente, por lo que está solucionado.

- ¿Es la información concisa y puntual?
  La total información del software (títulos, textos, botones, etc.), se encuentra de manera concisa y directa, con el objetivo de facilitar no solo la lectura, sino la comprensión del usuario.

- ¿Es el diseño del objeto del contenido de fácil entendimiento para los usuarios?
  Todo el contenido que podemos encontrar en el software fue diseñado de forma intuitiva y ordenada con el objetivo de dar entendimiento al usuario.

- ¿Se proporcionaron referencias adecuadas para toda la información derivada de otras fuentes?
  Tanto en las imágenes como en algunos textos proporcionados por el software se citan las referencias de las cuales es derivada dicha información.

- ¿Es la información del software consistente internamente y consistente con la información presentada en otro objeto de contenido?
  La información interna del software es consistente con su funcionalidad, además se expresó con estándares en inglés de fácil comprensión para usuarios futuros y se consolidó documentación; la información presentada concuerda con otros objetos de contenido en caso de tener la misma función.

- ¿Puede el contenido ser interpretado de forma ofensiva o engañosa, o abre la puerta a los litigios?
  La información presente en el software se encuentra de forma clara y su administración es sencilla, lo cual es un aspecto relevante, pues ésta puede ser usada por diferentes tipos de usuarios. El software no presenta contenido ofensivo o fuera del contexto; el administrador tiene control interno de todo ello y eliminar lo que pueda atentar contra la integridad de cualquier usuario.

- ¿El contenido infringe los derechos de autor o marcas comerciales existentes?
  La información consignada en el software, aunque no se encuentra cobijada bajo ninguna ley por ser casi que un software propio de la institución cliente, cita las fuentes a información de terceros y evitar problemas legales, por ejemplo, información de cada deporte, imágenes, banners publicitarios, etc.

- ¿Qué contenido contiene enlaces internos que contemplan el contenido existente? ¿Esos enlaces están correctos?
  En el software se encuentran enlaces internos en la información (historia, reglamento, etc.) de los deportes. Cada uno de los enlaces es correcto en la actualidad y se someterá a revisiones posteriores para garantizarlos. Se citó la fuente de cada uno de ellos.

- ¿El estilo del contenido genera conflicto con el estilo estético de la interfaz?
  No, dado que se trabajó con el *framework* Bootstrap, el cual ofrece diversas plantillas de diseño agradables, que combinan con el estilo del contenido y de la interfaz, proporcionando entendimiento al usuario.

7.2 PRUEBAS DEL *FRONT-END* (INTERFAZ) DE USUARIO

La destreza de esta prueba se basa principalmente en encontrar errores relacionados con mecanismos específicos de la interfaz, implementando la semántica de la navegación, para el éxito de ello se realizó lo siguiente:

- Las características de la interfaz son probadas con el fin de asegurar el cumplimiento de las reglas de diseño, estética y contenido visual relacionado están disponibles sin errores para los usuarios.
  Aquí se realizó una evaluación en los colores, tipo de fuente, contenedores, bordes, tablas, imágenes, íconos, botones, entre otros, concluyendo que el software maneja características apropiadas para el concepto del mismo; el estilo de la interfaz es dinámico, con el fin de producir un efecto agradable y fácil de entender.

- Cada interfaz es probada con el contenido de los casos de uso o ruta de navegación para cada tipo de usuario.
  Se tomaron los diferentes escenarios de los casos de uso y se compararon las respectivas rutas, concluyendo que el diseño planeado para el software es igual al diseño implementado, obteniendo el éxito de este paso.

- El *front-end* (interfaz) es probado en su totalidad contra casos seleccionados y rutas de navegación para descubrir errores en la semántica de la interfaz.
  Se evaluó cada interfaz haciendo énfasis en la posible existencia de errores semánticos, gracias a ello se encontraron errores en los íconos de algunos títulos, dado que se prestaban para posibles confusiones.

- La interfaz es probada en diferentes entornos para asegurarnos de su compatibilidad.
  Se probó el software en computadores con diferentes sistemas operativos (Windows y Linux), navegadores (Opera, Google Chrome y Firefox), y además se probó en dispositivos móviles con navegador Opera. Para la revisión de resultados, ir a la sección de prueba de compatibilidad.

7.3 PRUEBAS DE USABILIDAD

Para esta prueba, se eligieron usuarios del común para que interactuaran con el software y, posterior a ello, respondieran una serie de preguntas clasificadas en ocho (8) categorías proporcionadas por el libro de Pressman.

- Interactividad
  ¿Los mecanismos de iteración como botones, menús, desplegables, entre otros, son fáciles de entender y usar?
- Diseño
  ¿Son los mecanismos de la navegación, contenido y funciones colocados de una manera que permite al usuario encontrarlos rápidamente?
- Legibilidad
  ¿El texto está bien escrito y entendible? ¿Son las representaciones gráficas intuitivas y fáciles de entender?
- Estética
  ¿El diseño, color, tipografía y características relacionadas conducen a un uso fácil? ¿Los usuarios se sienten cómodos con la apariencia del software web?
- Características de visualización
  ¿El software hace óptimo el uso del tamaño de la pantalla y resolución?

- Sensibilidad del tiempo
  ¿Pueden características, contenidos y funciones importantes, ser utilizados o adquiridos en el momento oportuno?
- Personalización
  ¿La aplicación se adapta apropiadamente a las necesidades específicas de cada tipo de usuario?
- Accesibilidad
  ¿El software es accesible para las personas con discapacidad?

La prueba se realizó a una muestra de 15 personas, quienes respondían, SI o NO a las preguntas formuladas anteriormente conforme a la experiencia vivida con el software. La siguiente tabla nos muestra los resultados.

Tabla 4 Tabulación de los resultados de la prueba de usabilidad.

Categoría	SI	NO
Interactividad	15	0
Diseño	13	2
Legibilidad	15	0
Estética	12	3
Características de visualización	15	0
Sensibilidad del tiempo	13	2
Personalización	11	4
Accesibilidad	-	

Dentro de las personas seleccionadas para las pruebas se encuentran compañeros de la misma carrera (cinco en total), esto con el fin de evaluar la parte administrativa; los resultados por parte de nuestros compañeros resultaron agradables y llamativos dado el hecho de poder administrar el software de una manera intuitiva y fácil. Las otras personas seleccionadas para las pruebas son compañeros del campus universitario (10 en total) interesados, en su gran mayoría, por los deportes y quienes dieron respuesta positiva a cada uno de los aspectos evaluados.

Al hacer análisis de los resultados de la tabla anterior, concluimos que la usabilidad del software fue exitosa y también el objetivo de la prueba según Pressman: determinar el grado en el cual la interfaz del software hace más fácil la vida del usuario.

IMPORTANTE: Como se puede observar, la categoría Accesibilidad se encuentra nula, dado que el software se limita en cuanto a la prestación de servicios a personas con discapacidad visual principalmente. Sin embargo, consideramos que otras discapacidades no limitan al usuario de hacer uso del sistema porque su diseño es muy intuitivo.

7.4 PRUEBA DE COMPATIBILIDAD

Para esta prueba se intenta encontrar errores que puedan interferir con el funcionamiento, apariencia, rendimiento, entre otros aspectos del software, debido a los varios entornos para los cuales se desarrolló. Para cada aspecto se probó el software en diferentes dispositivos de visualización, sistemas operacionales, navegadores y velocidades de conexión a internet. La siguiente tabla muestra los resultados obtenidos:

Tabla 5 Resultados de pruebas en Sistemas operativos y navegadores.

Marca del computador	**Navegador / SO**	**Opera**	**Google Chrome**	**Mozilla Firefox**	**Velocidad de conexión a internet 4 MB**	**Velocidad de conexión a internet 10 MB**
Toshiba	Windows 8.1 Pro	✓	✓	✓	✓	✓
HP	Windows 7 Pro	✓	✓	✓	✓	✓
ASUS	Linux	✓	✓	✓	✓	✓

Además de lo reflejado en la anterior tabla, se aplicaron pruebas a dispositivos móviles con sistemas operativos Android Jelly bean 4.2 y 4.3, y navegadores Google Chrome y Opera respectivamente. Dichas pruebas reflejaron una estructura diferente a la versión de computadora con el fin de generar impacto en la visualización y navegación del usuario en el software. Se aclara, que para la versión de móviles, hay partes administrativas que se salen del *screen*, por ejemplo, los editores de texto en la creación de publicaciones.

7.5 PRUEBAS DE NIVEL DE COMPONENTE

En esta prueba se intentó buscar posibles errores en algunas de las funciones del software. Para ello se realizaron procesos para descartar falencias por medio del método **análisis de valores de límites.**

- Función registro

Figura 58: **Prueba función registro.**

7.6 PRUEBAS DE NAVEGACIÓN

El objetivo de estas pruebas es demostrar el correcto funcionamiento de los mecanismos que permiten la navegación al usuario y comprobar que cada objetivo de navegación se puede conseguir según el tipo de usuario.

- Menús de navegación.
  Para esta sección se evaluaron los menús de navegación del software, externos e internos para cada tipo de usuario; si en la siguiente tabla hay un aval, quiere decir que la navegación dentro del menú está funcionando bien.

Tabla 6 Resultados de la prueba de menús de navegación.

Menú / Usuario	Inicio	Nosotros	Noticias	Deportes	Torneos	Artículos	Administrar
Visitante o anónimo	✓	✓	✓	✓	✓ OJO: Info. Limitada	✓	-
Autenticado	✓	✓	✓	✓	✓	✓	-
Administrador	✓	✓	✓	✓	✓	✓	✓

- Motores de búsqueda externos e internos
  El software presenta motores/filtros externos de búsqueda en la pestaña **Jugadores** del menú **Deportes** y también en todos los menús presentes en la parte administrativa. Se realizaron búsquedas filtradas por el nombre de objeto y resultaron exitosas durante la ejecución de las pruebas.

7.7 PRUEBA DE CONFIGURACIÓN

El software de lado del servidor se ha probado en diferentes entornos:

Tabla 7 Resultados de la prueba de configuración.

Servidor de aplicaciones	Linux (Ubuntu)	Microsoft Windows 7 y 8
Apache 2	✓	✓
Nginx	✓	✓
Lighttpd	✓	✓

El puerto 8080 se habilitó en los diferentes Sistemas operativos, y se configuró el cortafuego o *firewall* para permitir la escucha por dicho puerto de conexiones externas. En los entornos probados la aplicación se ejecutó sin problemas.

7.8 PRUEBA DE SEGURIDAD

- Seguridad en el servidor
  La configuración de la seguridad del servidor depende de la implementada en el grupo de investigación GIB-CONUSS para el cual se realizó este proyecto.

  Para nuestro caso, el software fue probado en una máquina virtual proporcionada por el administrador de servidores de dicho grupo y administrado por los autores del proyecto por la consola *PuTTY*, la cual es en cliente SSH y telnet para la administración de servidores, esta máquina cuenta con un tamaño para la memoria RAM de 1 GB.

  Además de ello se hizo uso de *Filter,* propio del *framework* usado, *Laravel,* por el cual pasan todas las peticiones entrantes, un ejemplo propio es el trabajo sobre el envío de información a cada usuario según sea su rol.

- Encriptación
  Este método se utiliza al momento de guardar las contraseñas en la base de datos, con el fin de evitar la decodificación y robo de las claves por personas que tengan acceso al bando de datos, para lograr pasar la prueba se implementó el método por defecto de Laravel, el cual usa la clase Hash que ofrece Bcrypt seguro para almacenar contraseñas de usuario.

- Autorización o aval
  Para proteger el registro y por ende, el ingreso a la navegación del software, se utiliza la función de Avalar Usuario referida en el requerimiento funcional RF 2.0 que se encuentra en la sección 4.3.2.2.

8 CARACTERIZACIÓN COMO SOFTWARE AS A SERVICE

Como ya se ha referido en ciertos puntos del informe, el modelo de distribución del software implementado será: Software como un servicio (Software as a Service – SaaS). A continuación, se definen las características del sitio web basadas en las características propias del modelo.

8.1 DISPONIBILIDAD GLOBAL

El software está disponible de forma global por medio de Internet, a través de un navegador por demanda. El servicio está alojado en los servidores del grupo de investigación CONUSS de la EISI, los cuales están disponibles las 24 horas del día.

8.2 LICENCIA

La licencia normal se basa en suscripción o uso y se factura en ciclos recurrente. En una minoría de casos, se puede usar una tarifa plana, y este es el caso del sitio web desarrollado.

Considerando ser la primera versión del software enmarcada en la realización de un proyecto de pregrado universitario, lo cual nos informa sobre la madurez baja del mismo, se define la política de cobro con una tarifa plana, con el fin de buscar el mercado para empezar, y más adelante, en otras versiones, implementar en él la licencia basada en suscripción o uso.

8.3 MANTENIMIENTO DE SOFTWARE Y SERVICIO

El proveedor supervisa y mantiene el software y el servicio. Es decir, puede haber código ejecutable en el lado del cliente, pero el usuario no se hace responsable del mantenimiento de este código ni de su interacción con el servicio. La infraestructura que presta el servicio, alojada en el grupo de investigación CONUSS, siempre tiene un supervisor que mantiene el servicio libre de cualquier error posible o mal funcionamiento.

8.4 REDUCCIÓN DE COSTOS

La reducción de costos de mantenimiento y un costo mínimo del sistema de usuario final consiguen que las aplicaciones SaaS sean, por lo general, más baratas que las que utilizan versiones físicas de manejo local. La tarifa plana que maneja el servicio es considerablemente justa y equivalente a la utilidad que se obtiene del mismo. Si sumamos a esto, el hecho de no tener que asumir el mantenimiento físico, podemos observar una reducción real de costos.

8.5 ACTUALIZACIÓN AUTOMÁTICA

Estas aplicaciones gozan de actualizaciones de versión automáticas y administración de parches centralizada. El usuario final tendrá siempre acceso a la versión más actualizada del software deportivo, cuyos cambios serán implementados por el grupo CONUSS cada vez que sea necesario.

8.6 BAJAS BARRERAS DE ENTRADA

Las aplicaciones SaaS suelen tener una barrera de entrada más baja que las tradicionales. Las posibles barreras de entrada a presentarse en la distribución del software deportivo, además de ser solucionables, son pocas. Unas de ellas pueden ser: falta de conocimiento de los beneficios; el cambio abrupto al aprendizaje del manejo de la nube; falta de recursos económicos para adquirir el servicio; y la, muy probable, recolocación del personal encargado de la sección de sistemas de la entidad. A simple vista parece complejo, pero, las soluciones tradicionales incluyen el gasto en hardware que es una barrera más difícil de traspasar y esto las hace poner en desventaja inevitablemente.

8.7 UNIFICACIÓN DE VERSIONES

Todos los usuarios cuentan con la misma versión del software, de modo que el de cada uno siempre es compatible con el de los demás. Los usuarios del portal web deportivo podrán visualizar exactamente la misma versión en terminales diferentes, ya que las actualizaciones son centralizadas.

8.8 MODELO DE MULTITENENCIA

El SaaS ofrece un modelo de datos compartidos mediante un modelo de multitenencia. Sin embargo, también existe la opción de la virtualización de software de instancias individuales, y es éste último el que se implementará para ofrecer el servicio de UISport, basado en una plataforma de máquinas virtuales desde el grupo de investigación CONUSS de la EISI.

9 CONCLUSIONES Y RECOMENDACIONES

- El objetivo general planteado para este proyecto se cumplió en su totalidad, ya que se desarrolló el software con enfoque web con tecnologías que permitieron centralizarlas tareas e información requeridas de manera organizada, facilitando la administración-adaptación de la herramienta de forma gráfica.

- Otro de los objetivos que se cumplió y de hecho de los más importantes, fue la automatización del proceso llevado a cabo al momento de crear, poner en marcha y concluir torneos deportivos, para nuestro caso se dio prioridad al tenis de campo.

- Se trabajó sobre una metodología completa y rigurosa, *SCRUM*, propuesta para la realización de proyectos medianamente grandes, garantizando un trabajo más ameno, organizado y por supuesto de calidad; otro aspecto a destacar de dicha metodología es el enfoque en la comunicación con los usuarios, con el fin de asegurarse de dar solución a sus necesidades. Se recomienda usar esta metodología para proyectos de alta complejidad y que se tengan ideas mínimas sobre el producto que se desea construir, dejando claro que estas ideas pueden ser perfeccionadas durante el desarrollo.

- Se realizaron las respectivas pruebas al software, con el fin de asegurarnos de corregir posibles errores, dichas pruebas se realizaron basándonos en el libro Pressman, el cual sugiere la evaluación de aspectos importantes para lograr un producto de calidad. Basándonos en las falencias encontradas durante la ejecución de pruebas, se confirma el hecho de verificar el software antes de entregar un producto final y con ellos disminuir la probabilidad de fracaso del mismo.

- El desarrollo de este proyecto, ayuda significativamente a la organización en las entidades deportivas que presten este tipo de servicios deportivos (realización de torneos), ya que como mencionábamos al principio, en la actualidad es un trabajo de elaboración dispendiosa, lo cual se presta para presentar información equivocada.

- A través de la herramienta se motiva a las personas a la participación y práctica de deportes, ello con el fin de crear cultura deportiva y que las personas mejoren diferentes aspectos de vida, tales como su rendimiento físico, rendimiento en sus actividades diarias, crear mejores relaciones interpersonales, entre otras.

- El software desarrollado en este proyecto facilitará sin lugar a dudas la adaptación a diferentes entidades, además de que se presta para una gestión o administración muy intuitiva para cualquier usuario, lo que se considera un plus para la entidad, dado que reduce costos en cuanto a la contratación de personal profesional para realizar dicha administración.

- A medida que pasa el tiempo, hay una mayor participación en el mercado del software ofrecido como servicio, aportando con ello grandes beneficios como lo son la disminución de los costos que implica hacerle mantenimiento a los sistemas de información, y contar con una infraestructura tecnológica sin ocupar espacio dentro de las instalaciones propias de las organizaciones, que adquieren estos servicios con un costo también menor.

- Se recomienda para los administradores del software tener una capacitación del manejo del software y estudio total de las condiciones o políticas mínimas consignadas en el ANEXO F. Manual de usuario - UISport versión 1.0., ya que por más de que el software sea intuitivo se requiere de ellos para el éxito en el uso del sistema.

- Para próximas versiones se recomienda crear la implementación algorítmica para la formación de torneos para otros deportes propuestos directamente en el software, tales como fútbol, baloncesto y voleibol.

- Se recomienda para próximas versiones del software hacer más automática la parte que corresponde a la reprogramación de partidos de tenis, es decir que para los partidos reprogramados, se ejecute de manera automática, evitando el cruce con fechas (fecha, hora y cancha) ya programadas, pues en la actualidad es una tarea algo manual.

BIBLIOGRAFÍA

- ALVAREZ GARCÍA, Alonso. HTML5. Manual imprescindible. Madrid: Anaya Multimedia, 2012.

- JETBRAINS S.R.O. PhpStorm Develop Smarter, Not Harder [online]. República Checa: JetBrains Developer Community, 2014 [citado 4 de mayo de 2015]. Disponible en Internet: <URL: https://www.jetbrains.com/phpstorm/documentation/phpstorm_web.pdf>.

- PRESSMAN, Roger. Ingeniería del software. Un enfoque práctico. 7 ed. México: McGraw Hill, 2010.

- SAWYER MCFARLAND, David. JavaScript. Traducción de JavaScript: The Missing Manual. Madrid: Anaya Multimedia, 2009.

- SOSINSKY, Barrie. ¿Qué es la nube? El futuro de los sistemas de información. Traducido por María Pascual Cabrerizo. Madrid: Anaya Multimedia, 2012. Páginas 108 – 110.

- TÉLLEZ VALDÉS, Julio. "Computo en la nube: instrumento y objeto del derecho," En: XV Congreso iberoamericano de derecho informática. Buenos Aires, 2011.

- WIEDLER, Igor. Composer, Manual oficial [online]. Suiza, 2014 [citado 4 de mayo de 2015]. Disponible en Internet: <URL: https://librosweb.es/libro/composer/>.