Metodología

Tecnologías Investigadas

Frameworks

Para realizar este proyecto, primero se requiere investigar sobre las tecnologías para desarrollar aplicaciones web. En este caso se investigó dos tecnologías: Java Server Faces y Django. Java Server Faces es un framework que utiliza el lenguaje de programación Java y Django utiliza el lenguaje de programación Python. Para investigar las tecnologías se hicieron aplicaciones sencillas. Un ejemplo de una aplicación sencilla es mostrar una página web en un servidor. Java Server Faces utilizaba a Jboss como servidor de aplicaciones. Fue muy difícil configurar la conexión de la base de datos con Jboss. Se hizo varias pruebas y no se pudo utilizar el ORM de esta tecnología. Debido a estos inconvenientes se eligió Django. Se hicieron las mismas pruebas con Django y resultaron ser exitosas. No hubo errores en la configuración de la base de datos. El ORM de Django funcionaba a la perfección en las pequeñas pruebas que se hizo.

Bases de Datos y Sincronización con Django

Se utilizará MySQL como base de datos el cual es un sistema libre con mucha documentación disponible para el programador. Luego, se instalará Django, el cual proporcionará en el momento de la instalación un súper administrador. Al instalar Django, la base de datos se sincronizará con la que se haya diseñado para generar las tablas que Django utiliza en el sistema.

Para la restauración de la base de datos se implementará una recuperación diaria con el motor de la misma y será guardada en el servidor. Justamente, si se pierde la información se la puede recuperar actualizada a un día anterior. Esto será programado automáticamente por el motor de búsqueda que tiene MySQL y se hará una prueba de fallos para comprobar que el respaldo funcione correctamente.

Django Registration

Django provee una tecnología llamada Django-Registration que permite el uso de un registro y un login seguro. Por eso no se tiene que implementar este módulo y simplemente se instala y se utiliza. La aplicación utiliza esta tecnología para crear el módulo de registro. Se habla del módulo de registro en la sección de desarrollo de la metodología.

Control de Versiones: Git

Muchas veces al escribir código se necesita una forma de mantenerlo y corregirlo. Es muy común en el desarrollo de aplicaciones tener un controlador de versiones. Un controlador de versiones es una tecnología que permite guardar el código de diferentes estados de una aplicación. Precisamente, si se necesita volver al estado de una aplicación que estaba funcionando anteriormente es posible hacerlo. El controlador de versiones sirve para que varios desarrolladores trabajen en el mismo archivo. Después, el archivo se mezcla con las dos implementaciones de los dos programadores distintos. En el caso de que exista un error en la mezcla de los archivos, el controlador de versiones notifica al programador sobre el conflicto que existe. El controlador de versiones que se utiliza en esta aplicación es Git. Es decir, si la aplicación sufriera algún imperfecto se podría recuperar a un punto anterior sin tener que desarrollarla de nuevo.

Diseño

Posteriormente se debe hacer un diseño elemental de las pantallas de la aplicación. Las pantallas de la aplicación mostrarán la estructura de la aplicación en términos de su uso. Se mostrará la pantalla principal y luego como están relacionadas las distintas pantallas entre sí. De este modo se podrá percibir la navegación de la aplicación. Al mostrar la navegación de la aplicación se facilitará la implementación de modelos y controladores que necesiten ser utilizados en cada vista. Además, la planificación del diseño de las pantallas antes de programarlas otorga una gran ventaja al programador para poder desarrollarlas

correctamente. Así, no se tendrá que hacer doble trabajo cuando el diseño no funcione con la programación.

Desarrollo

Se continúa con el módulo del súper administrador. Este módulo permite dar cualquier tipo de privilegio a los usuarios como usuario digitador, usuario representante de equipo y usuario visita. También puede editar o eliminar todo tipo de torneo, cambiar el estado de cualquier torneo de público a privado o viceversa, permite editar los resultados, inserción de resultados, eliminar partidos. El súper administrador es el administrador general del sistema, el cual tiene la mayor cantidad de permisos y privilegios para el sistema.

Luego, se continúa trabajando con el módulo de login para la autenticación y el registro para establecer los datos iniciales de los nuevos usuarios. Este módulo permite a cualquier persona registrarse como administrador de torneos y crear su torneo. En el módulo de registro se ingresan los datos: nombre de usuario y la contraseña. Así, el usuario podrá ingresar sus datos iniciales al sistema. En caso que se trate de ingresar un usuario que ya exista en el sistema, un mensaje de error que notificará al usuario que su nombre de usuario ya se encuentra en la base de datos. El súper administrador tendrá la opción de cambiar la contraseña de cualquier usuario en caso de que el usuario olvidase su contraseña. Después de que los datos ya estén ingresados al sistema se trabaja en el módulo de login. El usuario escribirá su nombre de usuario y su contraseña para ingresar al sistema. Si los datos de nombre de usuario y contraseña coinciden con los de la base de datos, el usuario podrá ingresar al sistema y crear su torneo. Es posible que en este escenario ocurran dos errores. El primero será que el usuario no exista en la base de datos, entonces el sistema deberá notificar al usuario que no se ha registrado y darle la opción de hacerlo. El segundo error es que su nombre de usuario y su contraseña al momento de hacer login no coincidan, se le deberá notificar al usuario que no coincide el nombre de usuario con la contraseña, con la

opción de intentarlo de nuevo.

Luego, se implementa el módulo de administrador de torneos. El administrador de torneos podrá crear sus propios torneos, establecerlos como privados o públicos. Los torneos públicos son los torneos accesibles para todos los usuarios y podrán ver la tabla de posiciones. Los torneos privados son accesibles a usuarios por medio de permisos del administrador de torneos. Posteriormente, el torneo debe definirse en términos, si el equipo juega contra todos los otros equipos una vez o dos veces, esto se llama el ida y vuelta de los partidos. El administrador de torneos al generar uno deberá ingresar el nombre del torneo, la fecha de inicio y la fecha final del mismo. Al haber creado un torneo, el administrador de torneos creará los equipos para el torneo con el nombre del equipo y una descripción opcional. Después de que cree los equipos tendrá la opción para generar los partidos de manera automática. Una vez que esté satisfecho con los partidos que se muestran con el equipo local y equipo visitante para cada jornada se guardaran sin la posibilidad de edición. El sistema controlará que cada equipo se distancie lo suficiente en fechas para poder descansar, si existe un número impar de equipos se le otorgará una jornada de descanso a cada equipo durante el torneo, además se verificará de que jueguen todas las permutaciones o combinaciones posibles antes de generar todos los partidos para los distintos tipos de torneos. Luego se podrá insertar datos de los partidos como son los goles del equipo local y el equipo visitante. De esta manera, el administrador de torneos podrá tener varios torneos y administrar los que desee según los permisos que otorgue a otros usuarios. El sistema generará la tabla de posiciones dinámicamente mientras se sigan ingresando los resultados de los partidos.

El módulo de generación de torneos permite crear un torneo y agregarle datos como nombre de torneo, fecha de inicio y fecha final del torneo. Este módulo ya estará actuando con la base de datos, lo que permitirá probar el funcionamiento de la base de datos. Después

de realizar el módulo de generación de torneos se procede a implementar el módulo de creación de equipos por torneo. En este módulo deben estar relacionados los equipos a los torneos que se hayan creado. En la creación de equipos se ingresa el nombre de cada equipo y una breve descripción de los mismos. Después se comprueba que se haya guardado en la base de datos al ver en el sistema que existen los torneos creados aunque se cambien de sesión, además se pueden ver los equipos participantes del torneo.

Terminado el módulo de creación de equipos se procede a generar la funcionalidad de generación de partidos, verificando que ningún equipo juegue contra sí mismo y que todos los equipos jueguen todos contra todos. Este módulo pretende generar la lista de partidos y el digitador o administrador de torneos ingresará el resultado de cada uno de ellos. Al terminar este módulo se continúa con el siguiente, el generador de la tabla de posiciones. La tabla de posiciones muestra el orden de los equipos según sus puntos a favor, puntos diferencia y puntos en contra. Este es el momento donde se aprecia el funcionamiento global de la aplicación. Finalmente, se revisa si la aplicación funciona correctamente y se añade lo que falta si es que fuera necesario antes de subirla en línea. Luego se hacen las pruebas en línea con los datos generados desde el sistema y se finaliza el proyecto cuando todos los requerimientos descritos hayan sido implementados.

Revisión Literaria

Hoy en día, existen diferentes opciones para el desarrollo de aplicaciones web. Una de esas opciones es Java Server Faces. Java Server Faces es una tecnología de java que permite al desarrollador tener componentes gráficos los cuales se mapean 1:1 con HTML (Bengsen, 2004). Es decir, que la vista de la aplicación pueda ser controlada fácilmente con esta tecnología. También tiene un API (Application Programming Interface) que permite desarrollar nuevos componentes fácilmente al antojo del programador (Bengsen, 2004). Esto quiere decir que esta tecnología puede utilizarse para programar componentes que no existen

en HTML y podrían ser muy útiles para la implementación. Otra opción que tiene integrado los Java Server Faces son los Validadores (Dudney, Lehr, & Mattingly, 2004). Los validadores verifican los formularios que se utilizan para el registro de usuario y para el envío de mensajes. De esta manera el programador podrá hacer uso de estos validadores para verificar si un campo del formulario ha sido ingresado con un e-mail correcto. También, puede revisar que el usuario haya ingresado todos los campos reglamentarios y el tipo de dato correcto para registrarse o enviar su mensaje. Otra de las ventajas de Java Server Faces es la creación de páginas HTML dinámicas (FIelds & Bayern, 2001). Dinámico se refiere a que ocurren cambios en tiempo real, y la página de HTML puede cambiar según las necesidades del usuario. Esta tecnología podría ser muy útil para desarrollar aplicaciones web que requieren mucho cambio dinámico.

No solamente existe java como recurso para implementar aplicaciones web. Existen muchos otros elementos que podrían facilitar al desarrollador con su proyecto. Existe el lenguaje de programación PHP que es un producto de software libre que se ejecuta del lado del servidor (Welling & Thomson, 2003). Esto significa que se implementa desde el servidor y no hay llamadas a la aplicación. Los desarrolladores pensarían que es eficiente pero no lo es debido a que perjudica el modelo MVC. El modelo MVC es un modelo en el cual se exige que el modelo, la vista y el controlador sean implementados por separado (Wojciechowski, Sakowicz, Dura, & Napieralski, 2004). El modelo se refiere a la base de datos, la vista se refiere a todo lo que pueda ver el usuario y el controlador se refiere a la parte interna de cómo interactúa la base de datos con la vista (Wojciechowski et al., 2004). Es uno de los mejores diseños para implementar una aplicación escalable y estructurada. Ya que PHP puede escribir código dentro del código HTML (Welling & Thomson, 2003) perjudica al modelo MVC. Si se destruye el modelo MVC se pierde la escalabilidad, la estructura, el diseño de la aplicación y se hace muy difícil generar versiones futuras con otros cambios.

Otro framework que podría ser utilizado para el desarrollo de aplicaciones web es Django. El desarrollo web era muy caótico porque tenía limitaciones de banda ancha, vistas y formularios repetidos. Django simplifica estas repeticiones y desarrolla módulos más eficientes que han sido probados durante años (Forcier et al., 2008). Además, Django al separar las vistas, el modelo y el controlador no destruye el modelo MVC manteniendo la escalabilidad y estructura de la aplicación.

Como se ha visto, hay una variedad de tecnologías web. Unas son mejores a otras en términos de la promoción de buenos diseños de software. Un programador utiliza la tecnología más a fin a su experiencia, que además le permita un buen diseño.

Al principio las tecnologías web que se proponían para la aplicación eran tecnologías que utilizaban java. Debido a que se tenía un alto conocimiento del funcionamiento de java y se podía implementar el modelo MVC manteniendo la estabilidad de proyectos futuros. Como no se pudo desarrollar una aplicación simple se cambió de tecnología. Ahora, se propone utilizar Django que usa Python, este es un lenguaje con orientación a objetos facilitando el diseño MVC y además la utilización de este framework para desarrollar aplicaciones web es más fácil, eficiente y robusto.

Capítulo 3 –Tecnologías e Implementación

Tecnologías Utilizadas Para la Aplicación de Generación de Torneos

Para el desarrollo de la aplicación del generador y administrador de torneos se utilizó el framework Django 1.6 y el lenguaje de Programación Python 2.7. Como ambiente de desarrollo o IDE (Integrated Development Environment) se utilizó Pycharm Communtiy Edition que es totalmente gratuito y permite escribir código de manera flexible. Además, como motor de base de datos se utilizó MySQL 5.6.16 y como diseñador de base de datos se utilizó MySQL Workbench 5.2.47 para poder crear el núcleo de la aplicación.

Para mantener el control de versiones se utilizó Git. Git es un controlador de versiones totalmente gratuito. Se utilizó GitHub para guardar el código en línea (https://github.com). Para poder acceder al código se utiliza la siguiente ruta: https://github.com/calimat32/tourngen. Se utilizó esta tecnología para realizar todos los cambios que fueron necesarios durante el desarrollo de la aplicación. Se puede ver todas las versiones anteriores y la evolución del código mientras se fue desarrollando. Además, desde Github es posible descargar el código para instalar la aplicación en cualquier otra computadora.

Para mantener el control de los privilegios de los usuarios se instaló Django-Guardian. Los privilegios que controla Django-Guardian son por instancias de objetos. Es decir, un usuario tiene un permiso para cierto torneo. Django-Guardian sincroniza la base de datos insertando su tabla de privilegios en la cual se mapea el usuario, el privilegio, el tipo de instancia y la instancia a la cual está otorgando los privilegios. Así se pueden controlar los privilegios de los usuarios para diferentes torneos que se deseen crear después.

Introducción a MVT y ORM de Django

Django provee una arquitectura MVT (Model View Template) para crear aplicaciones. Django tiene un archivo llamado settings.py donde se agregan todas las

facilidades que se van a utilizar como la configuración para la conexión a la base de datos y otras configuraciones para el inicio de la aplicación. Django automáticamente sincroniza la base de datos con el modelo. Esto consiste en hacer que las tablas que provee Django son insertadas inmediatamente a la base de datos con la cual se está trabajando sin alterar ninguna otra. El ORM de Django se encarga de hacer clases en el archivo models.py para cada una de las tablas que está presente en la base de datos con sus respectivas relaciones. Lo que en Django se llama View es lo que en el modelo MVC (Model View Controller) se llama controlador. Existen archivos views.py para cada vista de las aplicaciones que se crean; las cuales procesan los datos para luego mostrarlos en las Templates. Lo que en Django se llama Template es la vista en el modelo MVC. Estos archivos son páginas HTML que se muestran al usuario y se puede decidir cómo se va a mostrar la información. Además, Django tiene un archivo para cada aplicación llamado urls.py, el cual permite relacionar los URL con las vistas creadas para las aplicaciones. Este es un controlador del cual el framework se encarga y el programador no tiene que preocuparse por implementarlo.

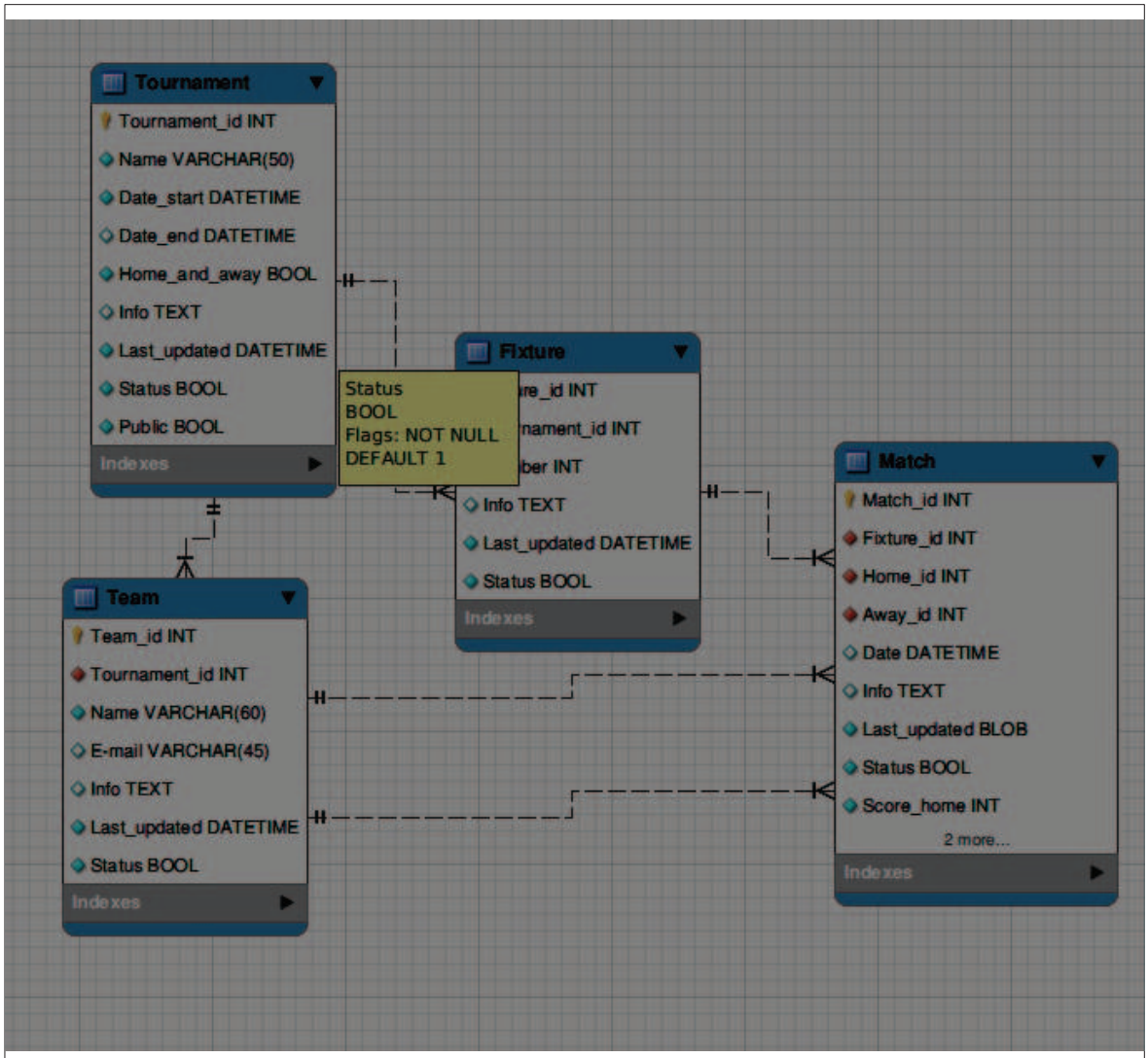

Figura 1.1: Diseño Principal de la Base de Datos

Desarrollo de la Aplicación de Generador y Administrador de Torneos

Diseño Inicial de la Base de Datos

Para poder implementar la aplicación primero se necesita hacer el diseño de la base de datos. El diseño de la base datos consiste en planificar todas las entidades que interactuarán con el usuario. Las entidades que se plantearon son: torneos (Tournament), equipos (Team), partidos (Match) y jornadas (Fixtures). En la Figura 1.1 se puede ver el diseño de la base de datos. Todas las tablas tienen los siguientes atributos: Tabla_id como referencia de la tabla, estado (Status) para implementar un borrado lógico si fuese necesario y un atributo de última vez actualizado (Last_Updated). Este último atributo no se utiliza en la aplicación web. La

base de datos se comparte con otra aplicación que es móvil. Las aplicaciones móviles necesitan sincronizar los datos que están en línea con los que no lo están. Para sincronizar los datos es necesario saber la fecha de su última actualización. Sin embargo, para las funcionalidades del generador y administrador de torneos de fútbol no es necesario el conocimiento de este atributo.

La tabla torneo tiene los atributos: nombre (Name), fecha de inicio (Date_start), fecha final de término del torneo (Date_end), torneo ida y vuelta (Home_and_away), información y si es público o no (Public). Un torneo puede tener muchos equipos pero un equipo solo puede pertenecer a un torneo. Eso es lo que se puede apreciar en la Figura 1.1 con la línea del diagrama. Del mismo modo un torneo puede tener varias jornadas pero la jornada pertenece a solo un torneo. La tabla de partidos tiene otra particularidad, un partido puede tener varios equipos locales pero solo un equipo local pertenece a un partido. De la misma manera, un partido puede tener varios equipos visitantes pero solo un equipo visitante puede pertenecer a un partido. La tabla de los partidos tiene dos referencias a la tabla de los equipos. Un partido consiste de dos equipos. Es por esto que se hace doble referencia a la tabla equipos. Esto es de gran ayuda para realizar los cálculos de la tabla de posiciones que se implementa.

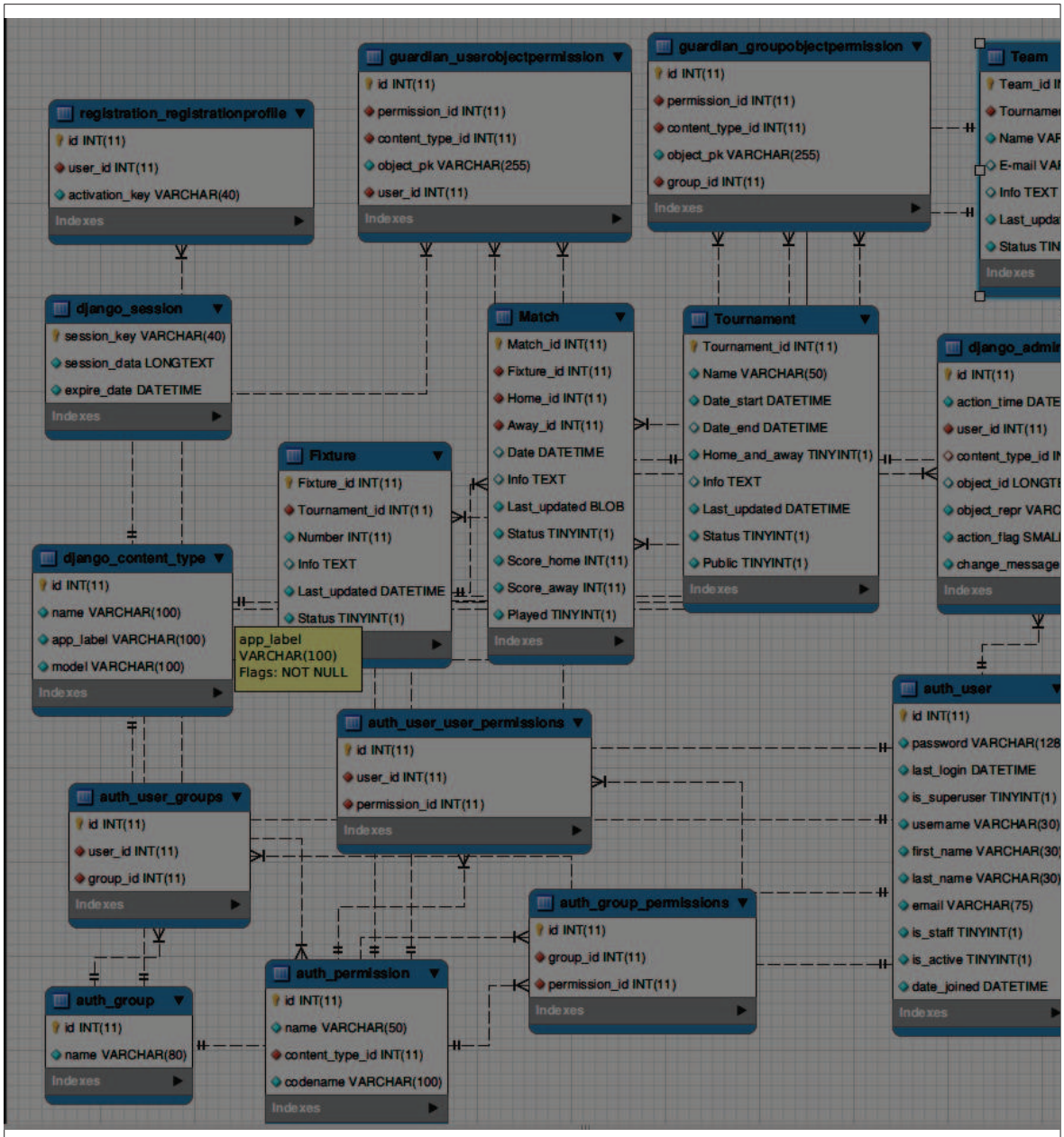

Figura 1.2: Diseño de la base de Datos después de Actualización con Django

Desarrollador	Django	Django Guardian
Tournament	django_admin_log	guardian_userobjectpermissions
Team	auth_user	guardian_groupobjectpermissions
Match	auth_user_user_permissions	
Fixture	auth_group_permissions	
	auth_permissions	
	auth_group	
	auth_user_groups	
	django_content_type	
	django_session	
	registration_registrationprofile	

Tabla 1: Tablas de insertadas por el Desarrollador, Django y Django-Guardian

Diseño final de la Base de Datos

Después de sincronizar la base de datos, Django y Django-Guardian insertan nuevas tablas que no fueron diseñadas por el desarrollador. Sin embargo, estas tablas son indispensables para que el sistema funcione correctamente. La Tabla 1 muestra las tablas que fueron insertadas por el desarrollador, Django y Django-Guardian. El diseño final que se muestra en la Figura 1.2 fue creado después de la inserción de las nuevas tablas. Las tablas que han sido resaltadas con forma de rectángulo y con rojo son las tablas diseñadas por el desarrollador. Las tablas que fueron resaltadas con rectángulo ovalado y de color azul son las tablas que fueron insertadas por Django-Guardian. Las tablas que no han sido resaltadas son las tablas que fueron insertadas por Django. Ahora, dentro de la base de datos, la tabla usuarios (auth_user) es la que controla todos los usuarios en la aplicación. Tiene el atributo "is_superuser" para saber si es que el usuario es un súper administrador o no. Django agrega la tabla de permisos (auth_permissions) que en esta aplicación se la utiliza para diferenciar un administrador de torneos de otros usuarios. También tiene una tabla de permisos por

usuario (auth_user_user_permissions) la cual es una tabla intermedia para saber qué permiso le pertenece a qué usuario. Además, se utiliza la tabla (guardian_userobjectpermissions) para asignar los permisos por instancias. Así es como Django-Guardian provee la facilidad de crear un permiso por cada instancia. El resto de las tablas son tablas nativas de Django y son las que no están resaltadas en la Tabla 1. Estas tablas a lo largo del tiempo fueron utilizadas por más programadores facilitando las funcionalidades típicas de aplicaciones web. Finalmente, se obtiene el diseño final de la base de datos para la aplicación del generador y administrador de torneos de fútbol.

Arquitectura de la aplicación de Generador y Administrador de Torneos de Fútbol.

Django divide su directorio en diferentes partes. Una de ellas son las aplicaciones que crea. El módulo principal se llama tourngen. Esta aplicación se crea apenas se inicializa un proyecto en Django. El nombre puede ser editado pero en este módulo se crean todas las configuraciones iniciales de la aplicación. Las configuraciones iniciales de la aplicación consisten en: la conexión a la base de datos, los módulos que han sido desarrollados por otras personas para facilitar funcionalidades, las configuraciones iniciales de un servidor para poder correr la aplicación en línea, el directorio de los archivos estáticos para proveer un buen diseño, la configuración de un archivo de log de errores y muchas otras estructuras que una aplicación web debe tener.

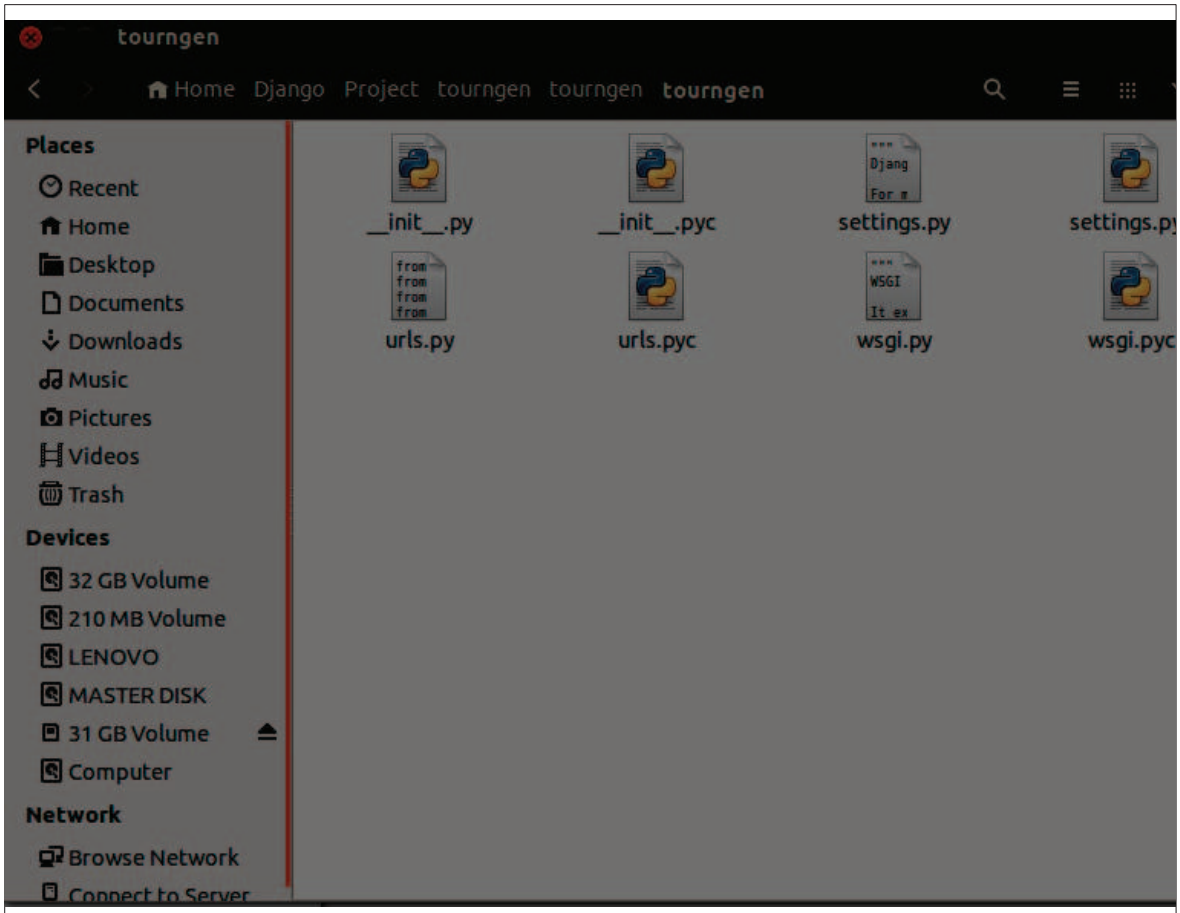

Figura 2.1 Estructura de directorio de aplicación del Generador y Administrador de torneos de fútbol

Configuraciones Iniciales

En la figura 2.1 se puede ver los archivos que contiene la aplicación del generador y administrador de torneos de fútbol. El archivo settings.py es un archivo que contiene toda la configuración necesaria para iniciar. Aquí se declaran todas las tecnologías que han sido descargadas como Django-Guardian. Django busca la aplicación en el directorio, si la aplicación no está instalada se produce un error y notifica al programador con un mensaje para instalarla. Si la aplicación existe, entonces Django utiliza la tecnología descargada. Además en este archivo se hace la conexión a la base de datos. De esta manera, no se necesitará escribir la configuración de la base de datos en otros archivos. La configuración de la base de datos consiste en el motor, el usuario y la contraseña. Así es como se conecta

la aplicación web a la base de datos de MySQL. Además se pueden insertar todas las configuraciones que una aplicación web debería tener que fueron mencionadas anteriormente.

El archivo urls.py es otro archivo importante. En este archivo lo que se relacionan son los URL con las vistas. Las vistas son los controladores de la base de datos. Las vistas son el lugar donde se escribe la lógica de la aplicación. Esto permite que el usuario escriba una dirección URL y dependiendo de la dirección se muestra una vista distancia. Por ejemplo, si el usuario escribe tourngen.com aparece la página principal de la aplicación.

El resto de los archivos que se muestran en la Figura 2.1 son archivos que fueron generados por Django. Son archivos de configuración para que se pueda correr una aplicación escrita en Python como aplicación web. Estos archivos no fueron editados de ninguna manera y son esenciales para que la aplicación web funcione adecuadamente.

Controladores de Django

El directorio principal de la aplicación que se ha creado con Django se muestra en la figura 2.2. El archivo manage.py es un archivo generado por Django. Lo que hace este archivo es generar un servidor local en la máquina del cliente para poder visualizar la aplicación antes de producirla. En la Figura 2.2 se muestran todos los directorios los cuales se utilizan para poder desarrollar la aplicación. Cada una de las carpetas y su importancia será explicada en las siguientes secciones de este capítulo.

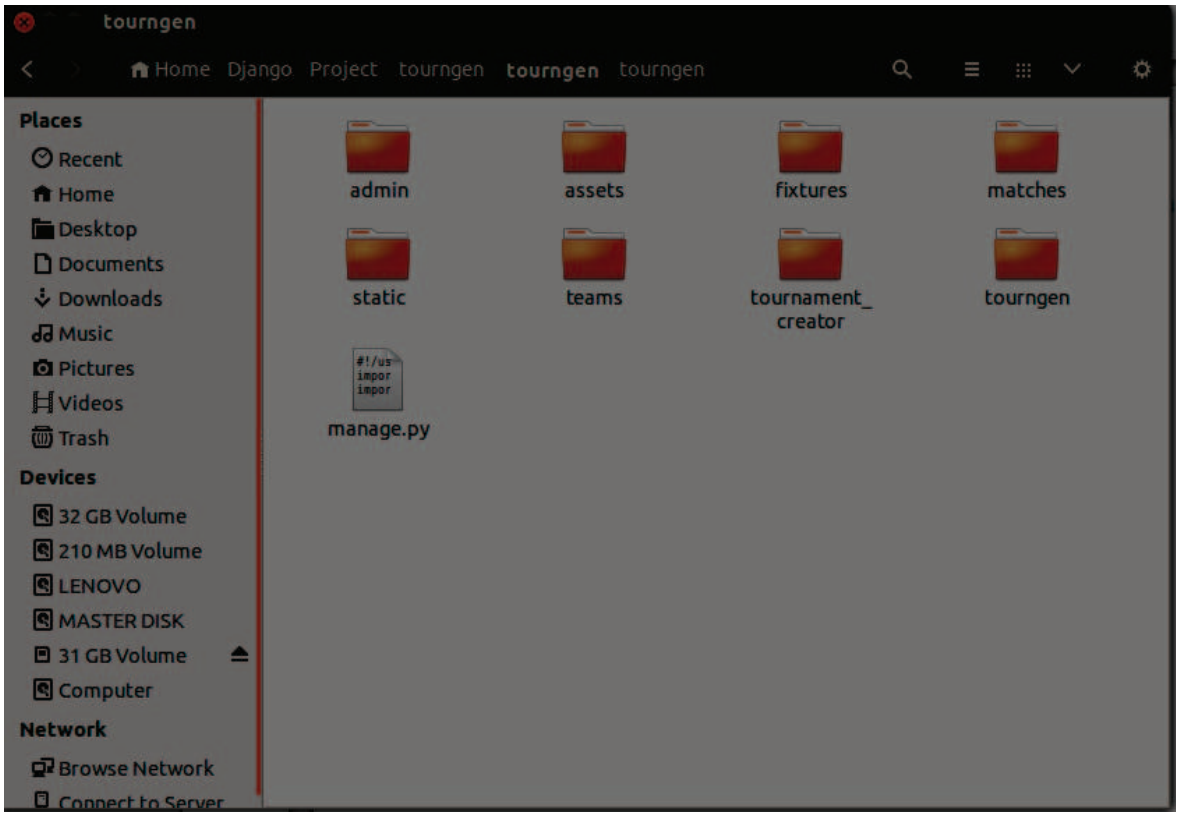

Figura 2.2 Estructura de directorio de aplicación general

Carpeta	Explicación
admin	Configuraciones de Súper Administrador
assets	Repositorio de Archivos Estáticos
fixtures matches	Módulo de jornadas Módulo de partidos
static teams	Creación de Archivos Estáticos Módulo de Equipos
tournament_creator tourngen	Módulo de Torneos Configuraciones Iniciales

Tabla 2: Explicación Breve de Directorio General de Django

Módulo de Súper Administrador de Django

La carpeta admin contiene todo lo relacionado con la vista del módulo de administración como se muestra en la Tabla 2. Django provee un URL en el cual se agrega /admin y se puede ingresar al módulo de administración. El módulo de administración es creado automáticamente por Django. En la aplicación de generador de torneos esto es el módulo del

súper administrador, el usuario con todos los privilegios automáticamente.

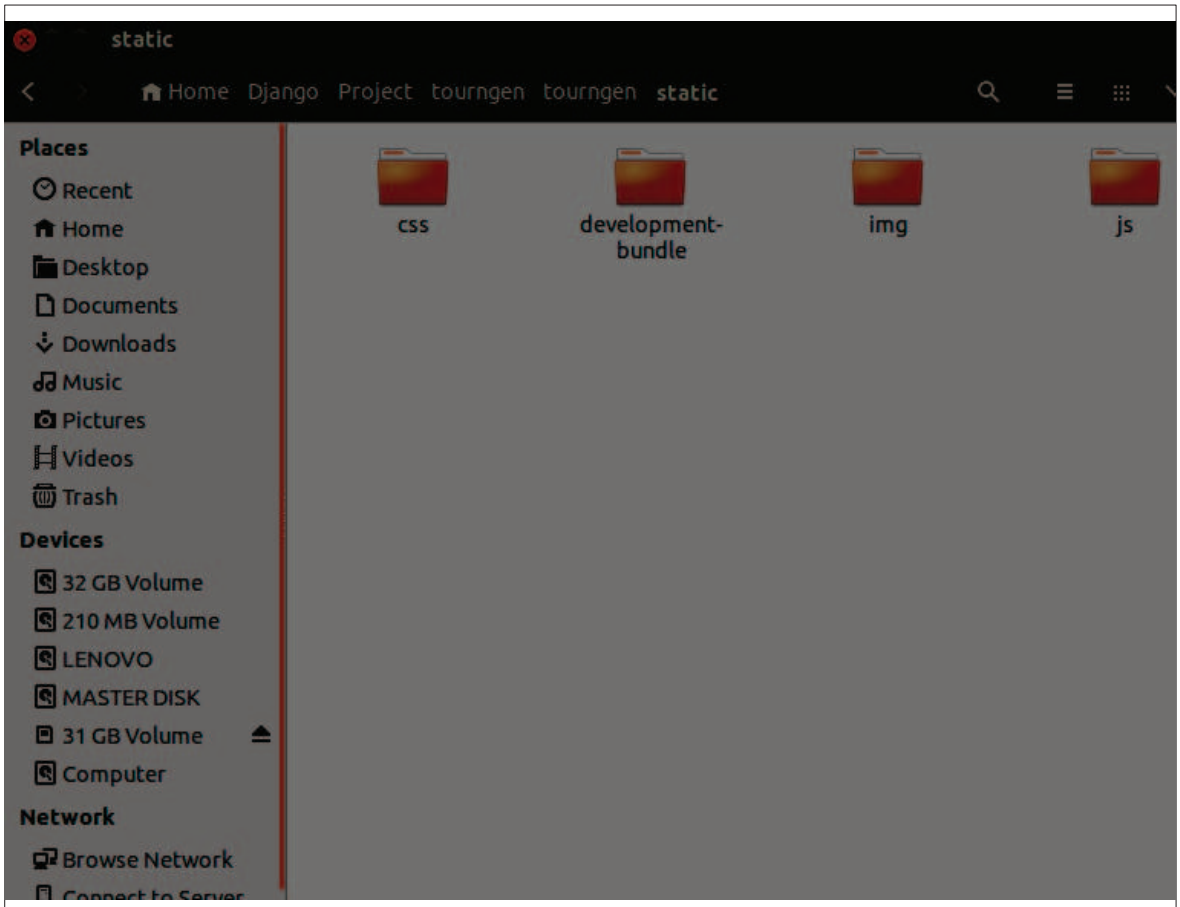

Figura 2.3 Estructura de directorio de Archivos Estáticos

Archivos Estáticos	Explicación
CSS JavaScript (JS)	Archivos para dar diseño Archivos para programar en front-end
Imágenes	Imágenes en la aplicación

Tabla 3: Explicación de Archivos Estáticos

Diseño de Django

Los assets es una carpeta que crea Django a partir de su carpeta static como se explica en la Tabla 2. En la carpeta static se guardan todos los archivos estáticos de la aplicación. En la

Tabla 3 se puede apreciar los tipos de archivos estáticos los cuales se utilizan para la aplicación y con su breve explicación. Existen archivos CSS, que se utilizan para darle estilo a las páginas. Archivos Javascript (js) que se utilizan para dar una programación de front-end en el desarrollo de aplicaciones web. Es otra forma de generar páginas HTML dinámicas. Otros archivos que se guardan en la carpeta static son las imágenes. Estas carpetas por el momento están vacías pero se las construye para luego aplicar un diseño a la aplicación web y hacer a la aplicación más llamativa. Esto mostraría las imágenes, el estilo y funcionalidades de front-end de Javascript si es que hubiese en la aplicación de generador de torneos. De todas maneras la carpeta permanece por si es necesario utilizarla en algún futuro.

La carpeta development-bundle es una carpeta que genera Django automáticamente. En el transcurso del desarrollo de la aplicación web no se modificó esta carpeta. Por lo tanto, es una carpeta que es necesaria para que funcione la aplicación pero sin modificaciones.

Vista de Django (Template)

En la figura 2.2 se puede ver que se tiene la carpeta tournament_creator. Esta es la aplicación principal que se ejecuta cuando se inicia el programa. En la figura 2.4 se puede ver que archivos contiene esta aplicación. En la Tabla 4 se puede apreciar una explicación breve de los archivos que son generales para las carpetas: tournament_creator, fixtures, teams y matches que se muestran en la Figura 2.2.

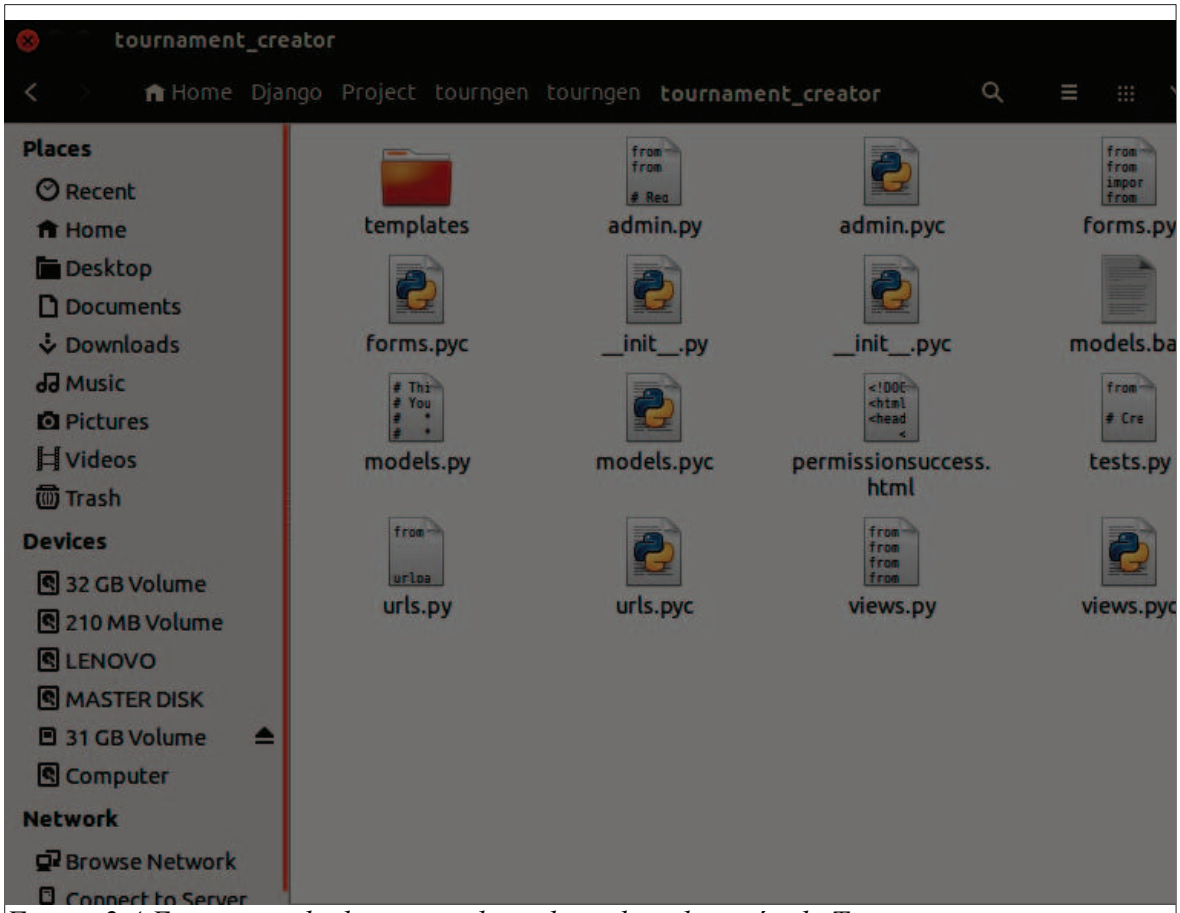

Figura 2.4 Estructura de directorio de archivo de aplicación de Torneos

Archivos o Carpetas	Funcionalidad
admin.py	Registro de funcionalidades para el Súper Administrador
forms.py	Archivo que crea formularios para que el usuario los pueda ingresar
models.py	Las tablas de la base de datos convertidas en clases
views.py	La interacción de los datos con la vista. (La ejecución del código)
urls.py	La referencia URL del módulo con su vista
templates	Todas las páginas HTML

Tabla 4: Archivos Generales de Módulos

Django crea una carpeta llamada templates. En la carpeta templates se encuentran todas las páginas HTML que serán mostradas al usuario. De este modo se separa la vista del controlador respetando al modelo MVC. Tiene un archivo admin.py en el cual contiene todas las entidades registradas las cuales el súper administrador puede controlar. Una vez que se registren, el súper administrador creado por Django tiene el control sobre las mismas. Es decir, en la aplicación de generación de torneos de fútbol se pueden crear los torneos, equipos, partidos y jornadas. Django crea un archivo llamado forms.py donde se guardan todos los formularios necesarios para el usuario. Los formularios consisten en el formulario de registro, el formulario para agregar un torneo. Así un usuario puede agregar los datos del torneo para guardarlos en la base de datos. Debido a que está adentro de tournament_creator entonces solo se muestran de los torneos. Existe el archivo models.py, en este archivo están todas las tablas de la base de datos como tablas y sus respectivas relaciones. Es decir que a las tablas en esta aplicación se las trata como clases. Aquí es donde actúa el ORM de Django con la base de datos MYSQL de la aplicación. Todos estos archivos están presentes en las otras carpetas que se mencionaron anteriormente.

Diagramas UML

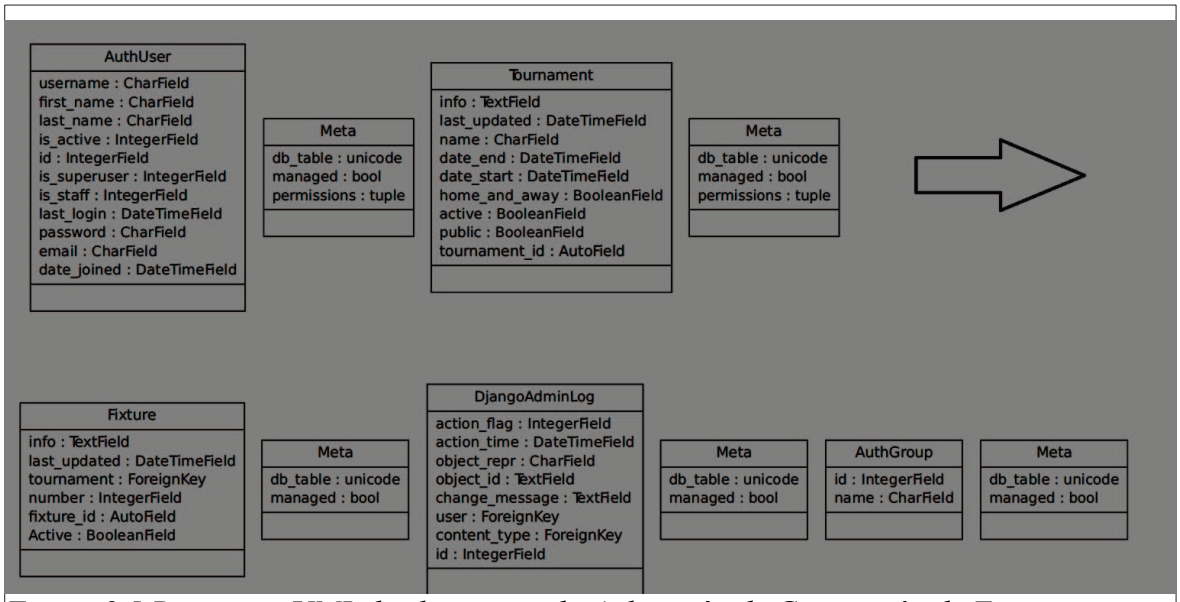

Figura 2.5 Diagrama UML de clases para la Aplicación de Generación de Torneos

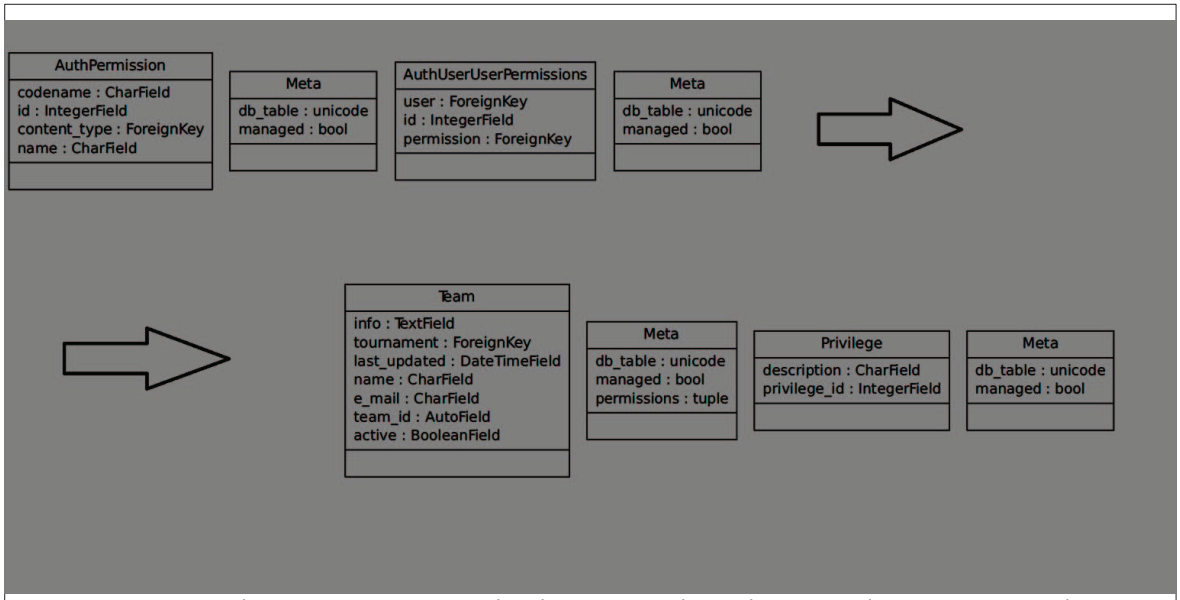

Figura 2.6 Segundo Diagrama UML de clases para la Aplicación de Generación de Torneos

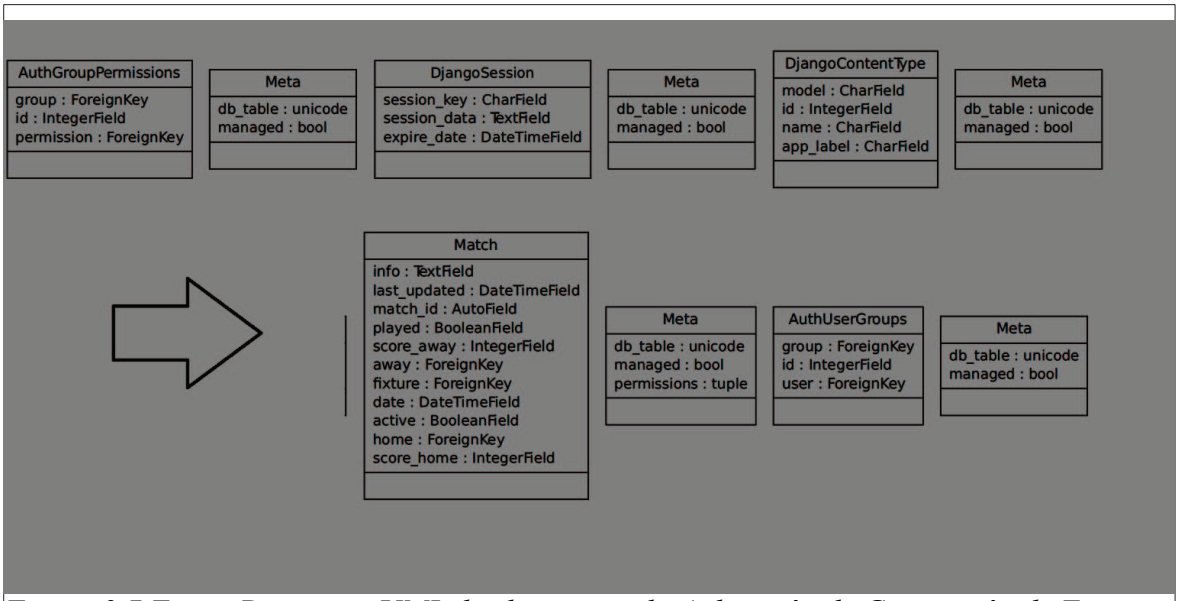

Figura 2.7 Tercer Diagrama UML de clases para la Aplicación de Generación de Torneos

En la Figura 2.5, Figura 2.6 y Figura 2.7 se puede ver el diagrama UML de las clases. Se puede ver que existe una clase por cada tabla que se diseñó en la base de datos. Esto es debido al ORM ya que su trabajo es convertir todas las tablas en clases. Por eso es que se puede ver una clase Torneo (Tournament) que es la que representa la tabla torneo. Se tiene una clase equipo (Team) que representa a la tabla de equipos y así sucesivamente. Se han agregado algunos atributos a la clase que no existían en la base de datos. Esto se debe a que Django necesita estos atributos para que puede ejecutar la aplicación correctamente. Además de eso se han creado nuevas clases llamadas Meta. Las clases Meta son las que utiliza Django para relacionar que tabla pertenece a que clase. Esto es una de las implementaciones de la tecnología y es por eso que se la deja como está.

Requerimientos de la aplicación

Casos de Uso

Los desarrolladores utilizan casos de uso para diseñar la funcionalidad de su aplicación. Se estudia cada caso común de la aplicación. La aplicación de generador de torneos tiene varios casos. Tiene un caso de registro, en el cual el usuario se registra. Un

caso de login el cual es cuando el usuario ingresa al sistema. Además, tiene el caso de generación de torneos que es lo propuesto a generar desde el sistema en el desarrollo de la aplicación. Esto permite a los desarrolladores explorar exactamente cómo funcionará su sistema.

Caso de Registro

El caso de registro del sistema funciona de la siguiente manera. El usuario navega hacia la página del sistema, en este caso no puede entrar porque le pide un usuario y una contraseña. Habrá un botón para que cualquier usuario se registre. Al hacer clic en ese botón, el usuario es llevado a otra página que contiene el formulario para ingresar un nombre de usuario y una contraseña. Luego el usuario hace clic para registrarse, y pueden ocurrir dos cosas: si es que el usuario no existe en la base de datos, entonces se ingresan los datos y se le muestra al usuario un mensaje de confirmación. Si el usuario existe en la base de datos, es decir su nombre de usuario, entonces se le notifica con un mensaje explicando que su e-mail ha sido registrado.

Caso de Login

En el caso de login se presenta al usuario con dos campos, el nombre de usuario y la contraseña. Si el nombre de usuario no existe en la base de datos, entonces se procede a verificar que la contraseña sea la misma que ingresó el usuario. Si el nombre de usuario y la contraseña coinciden se le da acceso a los módulos dependiendo del privilegio que tenga en el sistema. Si es que el e-mail no existe, entonces se procede a notificar al usuario que el e-mail que ingresó no está registrado y que por favor lo registre y se va al caso del registro. Si el nombre de usuario si existe pero no coincide con la contraseña, se le notifica al usuario que los datos no corresponden, y se le da la opción de restablecer su contraseña.

Caso de Torneos

Una vez dentro del sistema el usuario puede crear su torneo. Cuando crea su torneo, define un nombre para el torneo, una fecha de inicio, una fecha fin, establece si es ida y vuelta y establece si es público o privado. Aquí el administrador de torneos puede elegir los permisos para otorgarles a otros usuarios como el privilegio de digitador y el privilegio de participante de equipo. De esta manera los usuarios tendrán acceso a este torneo. El usuario con privilegio digitador podrá ingresar los datos de los resultados de los partidos, goles del equipo local y goles del equipo visitante. El usuario con privilegio de participante de equipo podrá ver el rendimiento de su equipo incluso si el torneo es privado pero no podrá hacer ningún cambio. Luego se dispone a crear los equipos para los torneos, se ingresa el nombre de equipo y una descripción del equipo, la cual puede ser opcional. Después de la creación de los equipos de los torneos se procede a generar los partidos. Los partidos se pueden generar manualmente o automáticamente. Cuando se generan manualmente el administrador de torneos tiene la libertad de elegir cuales equipos se enfrentan entre sí para cada jornada. El sistema verificará si existen todas las permutaciones posibles, si es que no están todas se enviará un mensaje de error el cual notificará al usuario que hay algún problema con sus jornadas. Si se generan automáticamente el algoritmo hará que los partidos se generen con todas las permutaciones posibles y dará una distancia según las jornadas para que los equipos tengan un descanso. Se da clic en grabar, una vez grabado no habrá marcha atrás y no se podrá editar. La tabla de posiciones se creará con valores iniciales de cero para todos sus campos los cuales son puntos, goles a favor y goles diferencia. Al ir ingresando los datos de los partidos generados por el sistema, la tabla de posiciones irá cambiando sus resultados y además se ordenará automáticamente desde puntos, goles a favor y goles diferencia.

Los casos de uso se dividen por diferentes actores. Los actores que se tienen en la aplicación de generador de torneos son el usuario administrador de torneos, el súper

administrador, el usuario digitador, el usuario representante de equipos y el usuario visita. En la figura 3.1 se puede ver todas las acciones que el usuario súper administrador puede hacer con el sistema. Es evidente que pueda hacer login y todas las ediciones e inserciones necesarias con torneos, equipos y partidos. No ingresa usuarios debido a que eso lo debe hacer el módulo de registro.

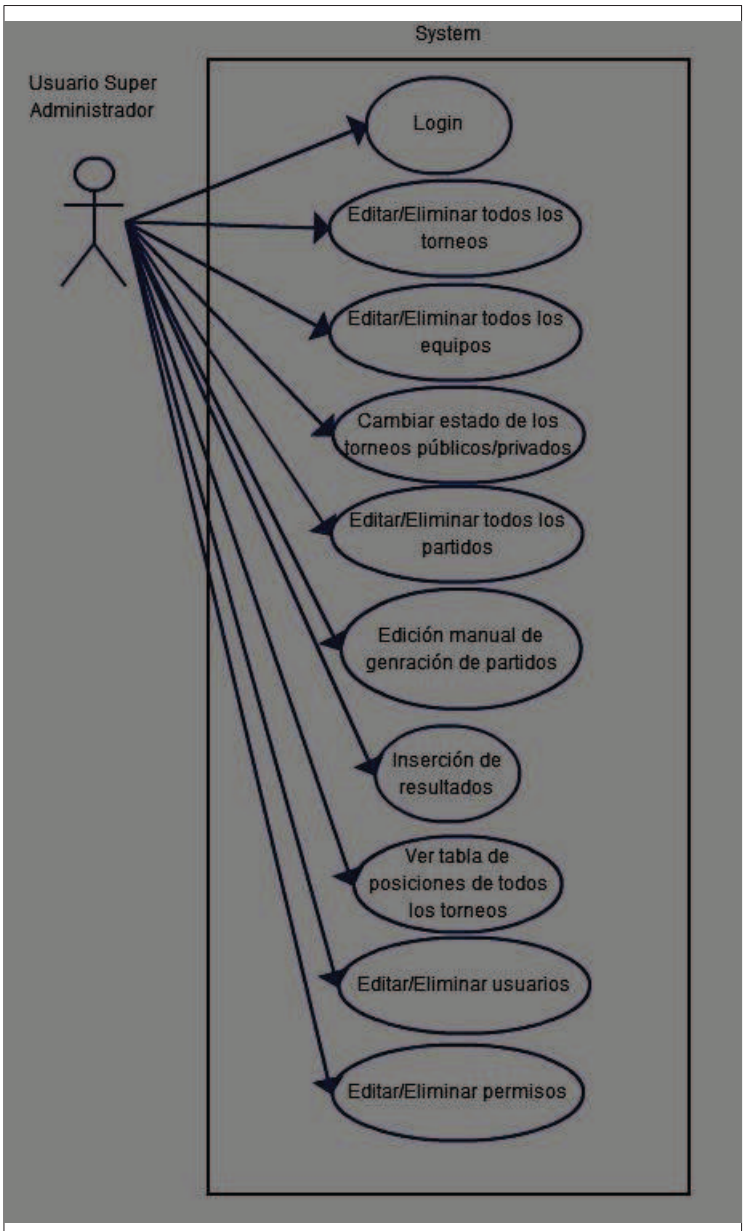

Figura 3.1 Caso de Uso de Súper Administrador

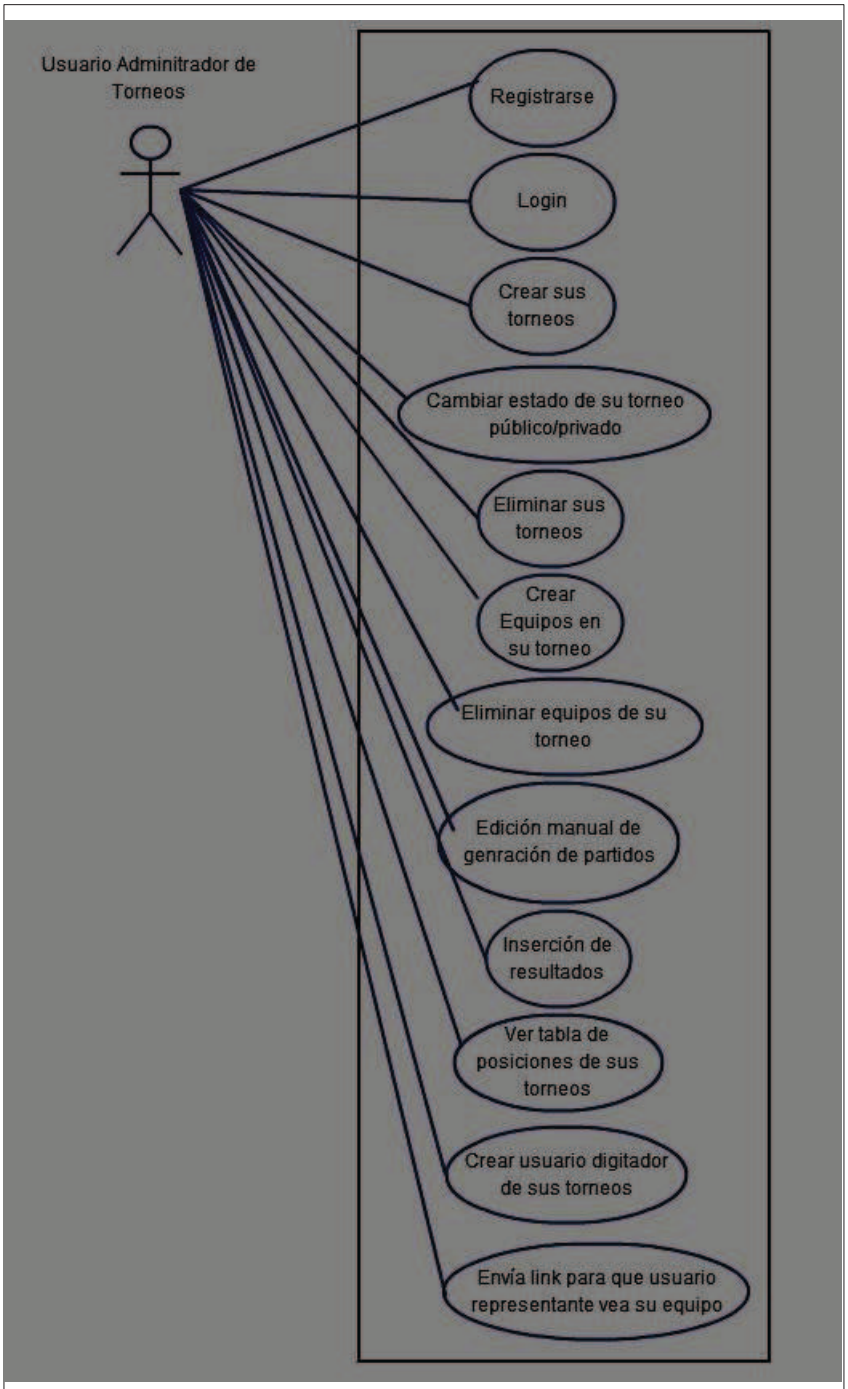

Figura 3.2 Caso de Uso de Administrador de Torneos

En la figura 3.2 se puede ver todas las acciones que puede hacer el usuario administrador de torneos con el sistema. En este caso puede crear torneos, crear equipos y crear los partidos. Además puede crear usuarios con ciertos privilegios. En la figura 3.2 se puede ver más claro las acciones que tiene el administrador generador de torneos con el sistema.

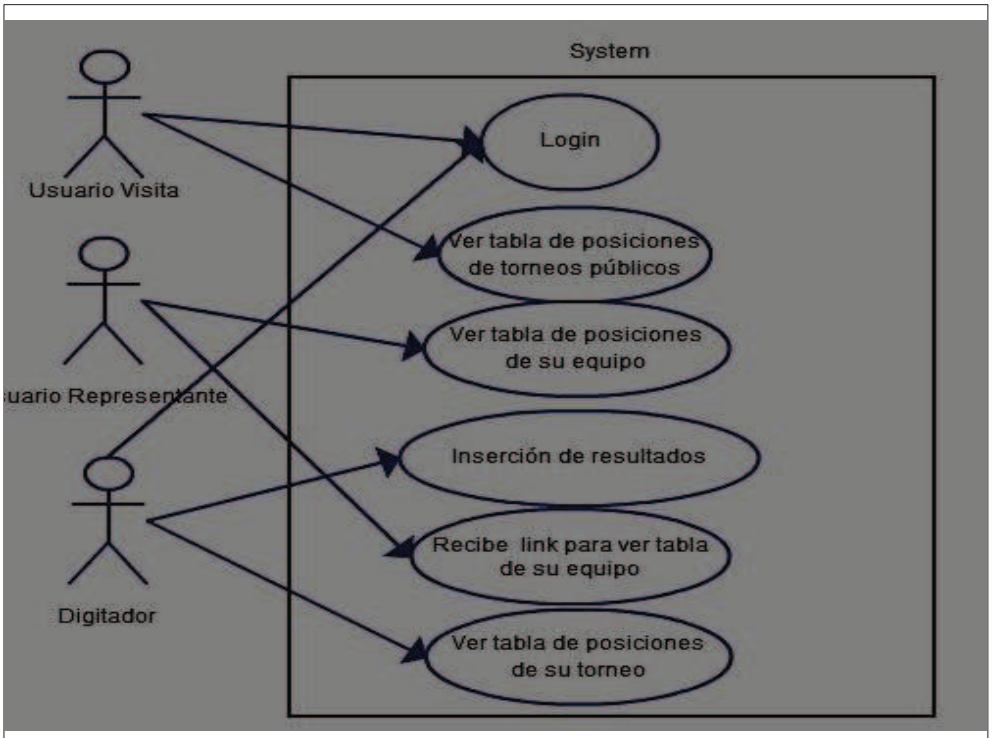

Figura 3.3 Caso de Uso de Usuario Visita, Representante y Digitador

En la figura 3.3 se pueden ver las acciones que tienen el resto de los actores con el sistema. EL usuario visita tiene únicamente puede hacer login y ver la tabla de posiciones de los partidos públicos. El usuario representante de equipos puede únicamente hacer login y ver la tabla de posiciones del torneo del equipo al cual fue asignado aunque sea privado. El usuario represente de equipo puede actualizar los resultados de los partidos del torneo el cual el administrador de torneos le dio acceso y además puede ver esa tabla de posiciones.