Informe técnico

Aylín Álvarez Santos C-312
Rocio Ortiz Gancedo C-311
Carlos Toledo Silva C-311
Ariel A. Triana Pérez C-311

Índice

1. Diccionario de datos

En esta sección describiremos cómo funciona el flujo de datos en la aplicación. Para esto se dividela explicación entre las diferentes aplicaciones que componen el proyecto.

1.1. Aplicación principal

Esta aplicación es la que permite el mostrado de los objetos que se aprecian en la página principal

1.2. Aplicación torneos

Esta aplicación contempla las operaciones relacionadas con los torneos.

El único modelo es "Torneo" y los formularios que se utilizan son "TorneoForm" y "EditarTorneoForm"

La vista basada en la clase "TorneoList" pinta en el template "torneos.html", todos los torneos registrados en la base de datos y los atributos de los mismos, dividiéndolos en torneos iniciados y no iniciados.

La vista basada en la clase "TorneoCreate" utiliza el modelo "Torneo" y el formulario "TorneoForm". Su función es la pintar el formulario mencionadao anteriormente en el template "crear_torneo.html" y permitir llenar dicho formulario a un administrador de torneo para así poder registrar un nuevo torneo en la base de datos.

La vista basada en la clase "TorneoUpdate" utiliza el modelo "Torneo" y el formulario "EditTorneoForm". Su función es la de pintar el formulario con los campos de un determinado torneo en el template "crear_torneo.html" para que un administrador de torneos pueda editarlos y luego ha de guardar dichos cambios en la base de datos.

La vista basada en la clase "TorneoDelete" utiliza el modelo "Torneo" y su función es permitir a un administrador de torneos eliminar un determinado torneo de la base de datos. Para poder hacer esto es necesario que el torneo no tenga ni participantes, ni rondas, ni partidas asociadas a él.

1.3. Aplicación admins_torneos

Esta aplicación contempla la creación de administradores de torneos. Para crear un administrador de torneos se utiliza un formulario que hereda del formulario que trae Django para el registro de usuarios. Los campos que se utilizan son los de nombre de usuario, email y contraseña. El modelo que se utiliza es similar el modelo de usuario de Django. Se utiliza la vista basada en la clase "AdminTorneosCreate" , la cual pinta el formulario en el template "registrar_admin_torneos.html" y después guarda la información que recibe en la base de datos y otorga a dicho usuario los permisos correspondientes..

1.4. Aplicación admins_partidas

Esta aplicación contempla la creación de administradores de partidas. Para crear un administrador de partida se utiliza el mismo formulario que para los administradores de torneo, el módelo que se utiliza es similar al modelo de usuario de Django. Se utiliza la vista basada en la clase "AdminPartidasCreate" la cual pinta el formulario en el template "registrar_admin_partidas.html" y

después guarda la información que recibe en la base de datos y otorga a dicho usuario los permisos correspondientes.

1.5. Aplicación users

Esta aplicación contempla las operaciones relacionadas con los jugadores.

Se utilizan los modelos "JugadorUser", "Provincia", "Municipio", "Jugador" y "AuthUser". Además se utilizan los formularios "UserRegisterForm", "JugadorRegisterForm", "MunicipioRegisterForm", "ProvinciaRegisterForm", los cuales se fusionan en el formulario "JugadorUserRegisterForm" y los tres últimos se fusionan en el formulario "JugadorUserEditForm".

La vista basada en la clase "UserCreate" utiliza el formulario "JugadorUserRegisterForm" y su función es la de pintar dicho formulario en el template "registrar_usuario.html". Una vez llenados los campos el jugador se registra tanto como jugador como un usuario que puede loguarse en el sistema, guardándose además toda la información registrada en la base de datos.

La vista basada en la clase "UserUpdate" utiliza el formulario "JugadorUserEditForm" y su función es la de permitir a un jugador modificar su información personal editando los campos del formulario antes mencionados y registrando los nuevos datos en la base de datos. La información se pinta y edita en el template "editar_usuario.html"

La vista basada en la clase "TorneoUserList" utiliza el modelo participante de la aplicación "eventos" y su función es pintar en el template "torneos_inscrito.html" los torneos en los que está registrado el usuario.

1.6. Aplicación decks

Esta aplicación contempla las operaciones con los decks.

Para crear un deck se utilizan los formularios "DeckForm", "ArquetipoPrincipalForm" y "ArquetipoAdicionalForm" para registrar información correspondiente a un deck y su arquetipo principal y secundario respectivamente. Estos, para mayor comodidad, se fusionan en el formulario "DeckCreateForm". Los modelos que se utilizan son "Arquetipo" y "Deck". Para crear un deck se utiliza la vista basada en la clase "DeckCreate", la cual pinta en el template "create_deck.html" el formulario "DeckCreateForm" y guarda los valores que asigne el usuario en la base de datos. Además esta detecta que usuario es el que crea el deck y guarda dicha información en la base de datos.

Para la edición de un deck se utiliza la vista basada en la clase DeckUpdate, la cual utiliza el mismo formulario y template que "DeckCreate" y además pinta la información actual que está guardada en la base de datos y registra en la misma los cambios que el usuario haga.

La vista basada en la clase "DeckDelete" se utiliza para eliminar un deck de la base de datos.

Las vista basada en clase "DeckList" se utiliza para listar los decks correspondientes al jugador logueado actualmente en el template "decks.html".

La vista basada en la clase "ArquetipoList" permite listar los arquetipos registrados en el template "arquetipos.tml".

1.7. Aplicación eventos

Esta aplicación contempla las operaciones relacionadas a los eventos de un torneo.

Para que un participante se pueda inscribrir en un torneo se utiliza la vista basada en la función "crearParticipante", la cual redirecciona al template "inscripción.html" donde el jugador logueado

actualmente deberá seleccionar uno de sus decks para poder inscribirse. Si el jugador no tiene ninguno creado se le redirecciona a la página de crear torneos para que se cree uno con el que pudiera inscrbirse. Una vez seleccionado el deck, se registra el identificador del torneo, el jugador y el deck seleccionado en la base de datos.

La vista basada en la función "listar" se utiliza para pintar en el template "torneo_eventos.html" los participantes en un torneo específico, así como el lugar o instancia competitiva que hayan alcanzado.

La vista basada en la función "editarLugarAlcanzado" utiliza el modelo "Participante" y el formulario "LugarAlcanzadoForm". Lo que hace es pintar dicho formulario en el template "lugar_alcanzado.html" para que un administrados de torneos modifique el lugar o instancia competitia que un participantete alcanzó en dicho torneo y posteriormente guardarlo en la base de datos.

La vista basada en la función "crear_participacion" utiliza el modelo "Participa" y el formulario "ParticipaRegisterForm". Lo que hace es pintar dicho formulario en el template "crear_participacion.html" para que un administrador de torneos agrgue la participación de un jugador en una determinada ronda del torneo. Luego esta información se registra en la base de datos.

La vista basada en la función "rondaInfo" utiliza los modelos "Torneo","Participa" y "Partida". Su función fundamental es mostrar información sobre los participantes que intervienen en la ronda de un torneo y las partidas que se desarrollan en la misma en el template "ronda_eventos.html".

La vista basada en la función "editarParticipacion" utiliza los modelos "Participa" y el formulario "EditarParticipacionForm". Su función es la de permitir a un administrador de torneo, asignar si un determinado supera la ronda o no y guardar dicha información en la base de datos.

La vista basada en la función "eliminarParticipacion" utiliza el modelo "Participa" y su función es la de eliminar la participación de un jugador en una ronda. Para esto debe asegurarse que el jugador no esté registrado en ninguna partida de la ronda.

La vista basada en la función "crearPartida" utiliza el modelo "Partida" y el formulario "PartidaRegistgerForm". Su función es pintar el formulario en el template "crear_partida.html" y guardar en la base de datos, los datos ingresados por un administrador de partidas.

La vista basada en la función "editarPartida" utiliza el modelo "Partida" y el formulario "PartidaRegistgerForm". Su función es la de pintar dicho formulario en el template "crear_partida.html" con los datos de una determinada partida para que estos puedan ser editados por un administrador de partidas. Luego los cambios realizados serán guardados en la base de datos.

La vista basada en la función "eliminarPartida" utiliza el modelo "Partida" y su función es eliminar una determinada partida de la base de datos.

1.8. Aplicación de consultas

Esta aplicación contempla todo lo relacionado con las consultas, desde su implementación a nivel de modelos, las vistas y los templates.

Para el desarrollo de la aplicación, se tiene una clase abstracta `Query` que representa una consulta que se efectúa a través de su método `execute(context)` y devuelve una instancia de `context` que es un diccionario donde están los elementos para enviar al template. Se tiene una clase concreta para cada consulta del sistema.

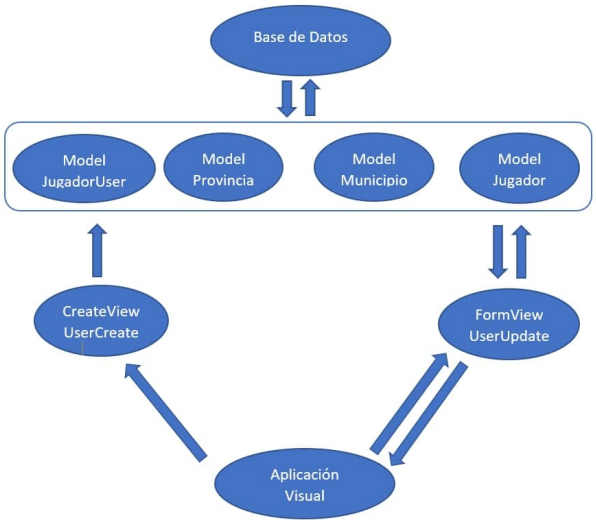
Además se aplicó el patrón de diseño Factory para la construcción de las instancias de las clases herederas de Query de forma dinámica.

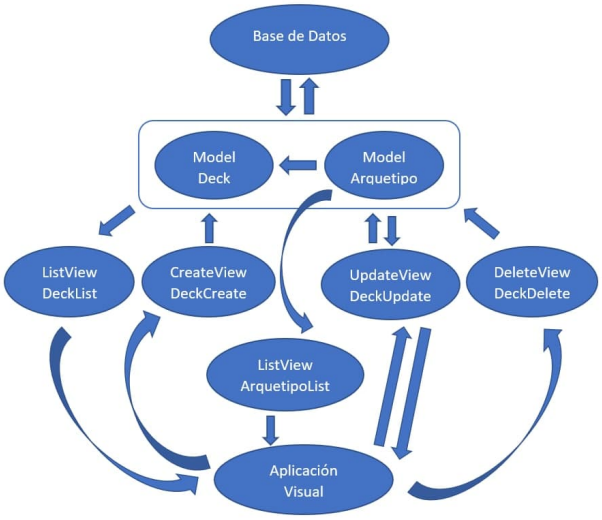
Luego se tiene un formulario para cada consulta del sistema. Todos estos formularios se almacenan en un diccionario y se envian al template "consultas.html", donde son renderizados los

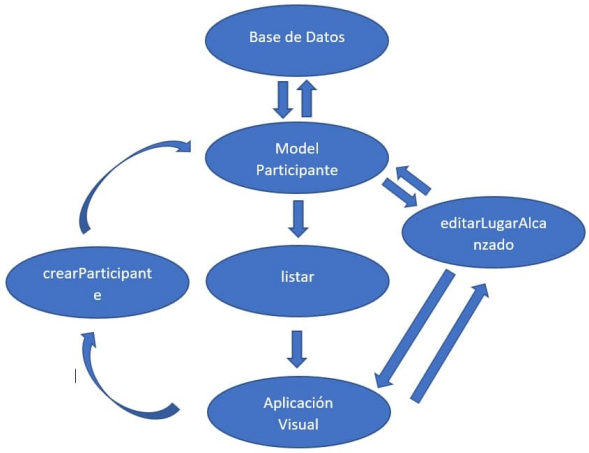
formularios. Alguno de estos formularios es utilizado y se envía al servidor utilizando el método POST, y se toma el nombre del formulario de un campo oculto en el template. Se constuye una instancia de la clase Query correspondiente utilizando las bondades de Factory. Se ejecuta la consulta y se devuelve la tabla de respuesta, la cual se almacena en el diccionario de contexto, y se renderiza la tabla en el template "consultas.html" debajo del formulario en cuestión.

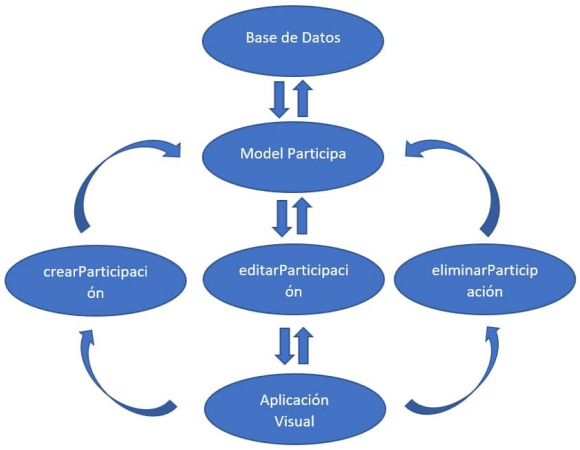
2. Esquemas con el diseño de la aplicación y algunas clases definidas

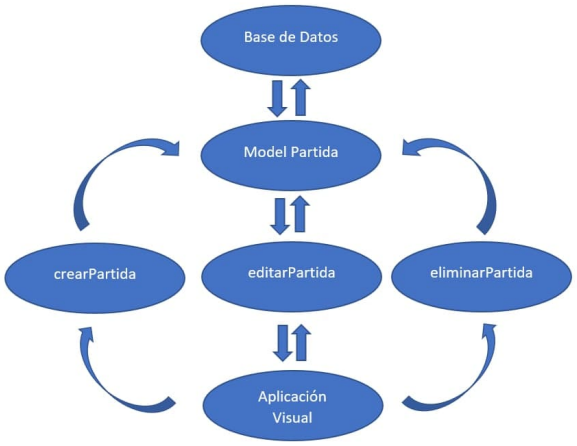
En esta sección se presentan algunos esquemas para facilitar de forma visual el entendimiento del funcionamiento de la aplicación. Algunas clases aparecen marcadas con Model para indicar que son modelos y otras con View para indicar que son clases de vista. Las que no aparecen marcadas son funciones. El sentido de las flechas indica el flujo de los datos por la aplicación.


Base de Datos
Model JugadorUser
Model Provincia
Model Municipio
Model Jugador
CreateView UserCreate
FormView UserUpdate
Aplicación Visual


Base de Datos
Model Deck
Model Arquetipo
ListView DeckList
CreateView DeckCreate
UpdateView DeckUpdate
DeleteView DeckDelete
ListView ArquetipoList
Aplicación Visual


Base de Datos
Model Participante
editarLugarAlcanzado
crearParticipante
listar
Aplicación Visual


Base de Datos
Model Participa
crearParticipaci
ón
editarParticipaci
ón
eliminarParticip
ación
Aplicación
Visual


Base de Datos
Model Partida
crearPartida
editarPartida
eliminarPartida
Aplicación
Visual