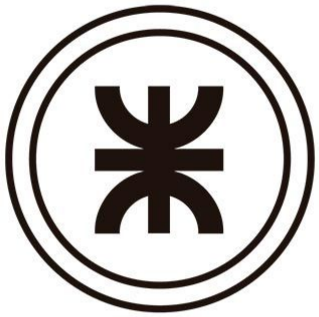

UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Justificaciones de Diseño

Materia: Diseño de Sistemas

Profesor: Luciano Straccia

Integrantes:
Franco Passarelli
Giuliana Nicole Vetere Portillo

Año: 2020

Especificación y justificación de requerimientos:

1. **Ofrecer registración e inscripción vía web. Cada participante ingresa nombre, apellido, tipo y número de documento e identificador de la FIDE (FIDE ID). Si el participante no tiene usuario, el sistema deberá permitir el alta.**

   Para este punto, en el diagrama implementamos una clase Usuario con los distintos tipos de usuario que heredan de la misma (Administrador, Juez, Supervisor y Jugador). Tienen atributos en común excepto por el FIDEid y el número de orden que son específicos para el Jugador.
   La parte del requerimiento que pide registro e inscripción forma parte de la interfaz, por esto, no lo contemplamos en el diagrama.

2. **Cobrar la inscripción a través de diferentes medios: tarjeta de crédito, débito, transferencia bancaria y otros a incorporar próximamente.**

   Para este requerimiento, la clase Torneo tiene el método inscribirJugador que recibe un Jugador y un Medio de pago, y se encarga tanto de cobrarle al Jugador la inscripción con el medio de pago indicado, como de agregarlo a la lista de participantes del torneo. Utilizamos el patrón Strategy para manejar la forma en que se le cobra al jugador según el medio de pago elegido.

3. **Automatizar el proceso de adquisición de ELO sabiendo que la FIDE (Federación Internacional de Ajedrez) actualiza el ELO de cada miembro el primer día de cada mes. Actualmente el Director del Torneo obtiene su valor consultando la pagina web: https://ratings.fide.com/ o bien descargando el mismo día del inicio del Torneo de ahí mismo archivos TXT (separado por tabulación) o en formato XML. El sistema debe ser capaz**

de importar e interpretar estos archivos y cargar automáticamente el ELO correspondiente a cada participante.

Para este punto, incorporamos un método obtenerELOs, que tendría el procedimiento necesario para leer los archivos, interpretarlos y asignarle el ELO a cada jugador del torneo. El mismo retornaría el FIDEid del jugador a modo de identificación, y el ELO correspondiente en formato hash, variando la forma de obtenerlo dependiendo del tipo de obtención de ELO seleccionada para el torneo, implementado con el patrón Strategy.

4. **Notificar vía SMS o WhatsApp a los participantes su próximo oponente, mesa y horario.**

   En este caso, la ronda posee el método notificarJugadores, el cual se ejecuta cada vez que se arma una partida ya que es esta la que posee los datos que deben ser enviados. Utilizando un Strategy, el tipo de notificación está dado por la clase TipoNotificación, de la que heredan Whatsapp y SMS, que se encargarán de enviar la información a los jugadores.

5. **Crear y mantener usuarios para jueces (quienes podrán validar resultados), supervisores (que cargarán resultados) y un administrador con capacidad para crear, modificar y eliminar estos usuarios. Todas las acciones de cada usuario, deberán actualizar un registro de eventos.**

   Para este punto, se incorporaron métodos específicos para las clases Juez y Supervisor, que les permitan validar resultados y cargar resultados respectivamente. No se incorporaron métodos al Administrador dado que simplemente implicaría ejecutar métodos de la clase Usuario para las altas y modificaciones, y una eliminación de un registro directamente desde la base de datos para las bajas.

Para implementar la actualización del registro de eventos, se incorporó una clase RegistroDeEventos con una lista de eventos como atributo. Esta lista de strings, indicando qué acción se realizó y sobre qué entidad, sería actualizada por un Visitor que permitiría la ejecución del método necesario para cada caso en particular.

6. **Generar un informe de resultados parciales por cada ronda y otro de resultados generales del torneo en formato XLSX, HTML y PDF.**

   Dentro de los métodos generarInformesTorneo y generarInformesRonda, se encuentra la lógica para generar los informes en los tres formatos solicitados. Dichos métodos serán ejecutados desde la interfaz cuando sea necesario.

7. **Funcionamiento del Sistema Suizo**

   El resto de los métodos y clases incorporadas al diagrama, se utilizaron para emular el funcionamiento del sistema suizo. Cabe destacar la clase Criterio, que permite cumplir el requerimiento de que los jugadores que no tengan ranking sean agregados al final de la lista utilizando uno de los criterios del enunciado (orden por apellido o sorteo al azar).