PEI - DESARROLLADOR FULL STACK

EJERCICIO FINAL

MARZO DE 2021

1. Criterios de Trabajo

A los fines prácticos del ejercicio final, se deberán tomar las siguientes consideraciones antes de comenzar con la resolución del mismo:

- Lee detenidamente los requerimientos funcionales antes de comenzar a construirlos.
- Intentá resolver los ejercicios en el orden en que aparecen. Si alguno no te sale, continuá con el siguiente y luego volvé a intentar resolverlo cuando hayas terminado el resto.
- Si bien la solución se va a parecer a la que estuviste trabajando durante la segunda semana, es una versión homogénea para todos por igual, que incluye la resolución del ejercicio de repaso de una forma estándar. Tomate tu tiempo para darle una mirada y entenderla.
- Es preferible contar con un único requerimiento completamente construido a tener varios de ellos construidos por la mitad o con fallas graves.
- Tenés 4 horas para poder desarrollar la mayor cantidad de funcionalidad posible de forma correcta. Organizate y optimizá tú tiempo.
- Se considerará una funcionalidad como terminada si: tiene todos los requerimientos desarrollados, no posee bugs críticos o invalidantes y además posee los Tests Unitarios correspondientes.
- No olvides realizar el Commit y Push de los cambios realizados en el GIT. Para ayudarte, podés hacer Commits parciales a medida que vayas terminando cosas.
- Más allá de los desarrollos que realices, la Solución debe compilar. Si no lo hace, es preferible que el código que no hayas terminado **lo comentes y no lo borres**. Una buena práctica es agregar un Tag de `//TODO: [Comment]` para indicar qué faltó o qué falla.
- Se pueden usar los apuntes, el material de las clases e Internet. No se puede conversar con otras personas por ningún medio (verbal, chat, teléfono, etc)

2. Listado de Requerimientos Funcionales

Se solicita extender la funcionalidad del sistema actual de administración de partidos de tenis para usar en un torneo interno de **Baufest**.

2.1 Agregar el Concepto de Entrenador

- Agregar la Entidad **Entrenador** y todas las clases necesarias para poder desarrollar su correspondiente ABML
- El entrenador deberá contar con una propiedad “nombre”.
- Validar que exista, o agregar una opción en el menú general de la aplicación que permita acceder a este nuevo ABML
- Al crear un jugador, permitir la posibilidad de seleccionar y asignar un entrenador
- Se requiere realizar dos Unit Tests de servicio teniendo en cuenta los siguientes pasos:
    - Crear un nuevo entrenador y validar el resultado obtenido
    - Recuperar todos los entrenadores y validar que coinciden con los entrenadores definidos en el mock del test unitario
    - Recordá que es un test de servicio y deberás mockear el repositorio

Nota: ABML es Alta-Baja-Modificación-Listado. Comprende la creación de pantallas, servicios y métodos de acceso a datos, en este caso para la entidad Entrenador.

2.2 Agregar Información de Ganancias de Jugadores

- Agregar al listado de jugadores una nueva columna informativa en la que se muestre la ganancia total obtenida por jugador de acuerdo a los partidos que ha ganado o perdido, en base a las siguientes reglas:
  - Si gana un partido con diferencia de set igual o superior a 3 (6-0, 6-1, 6-2 y 6-3), recibe $300
  - Si gana un partido con diferencia de set menor a 3 (6-4, y 6-5), recibe $200
  - Si pierde el partido, no recibe ningún premio
- Esta información debe obtenerse de los partidos finalizados registrados en la aplicación, y no se debe crear ningún campo nuevo en la tabla de Jugadores, sino que debe ser calculado.
- Esta información deberá mostrarse de forma automática al mostrar el listado de jugadores, sin requerir presionar ningún botón ni otra acción del usuario.

2.3 Crear los partidos iniciales de un torneo a partir de una lista de Jugadores

- Se deberá generar una nueva pantalla en donde el usuario del sitio pueda seleccionar 4 Jugadores para generar los partidos iniciales de un torneo. Estos jugadores deben ser distintos entre sí.
- Adicionalmente, deberá poder ingresar la fecha en la cual se jugarán los partidos. Esta fecha será la misma para todos los partidos generados.
- Al presionar el botón “Generar”, la aplicación debe generar 2 partidos usando los 4 jugadores seleccionados, respetando las siguientes reglas:
  - Cada jugador participa en un único partido.
  - Todos los jugadores deben estar asignados, es decir, participar de los partidos.
  - Se ordenan los jugadores por ranking y se arman estos partidos: 1ro vs 4to, 2do vs 3ro. Si dos jugadores tienen el mismo ranking, da lo mismo el criterio que se use para desempatar.
- Respecto la selección de Cancha, se deja a libre elección del alumno cómo implementar la asignación de la misma (se recomienda buscar una manera sencilla de asignarla de forma de no complicarse). El objetivo es que la solución funcione y permita crear los partidos, indistintamente de la Cancha asignada.
- Luego de generados los partidos, la aplicación debe volver a la pantalla de generación con un mensaje indicando que los partidos se generaron correctamente.
- Realizar como mínimo los siguientes Unit Tests de servicio que verifiquen lo siguiente:
  - El servicio valida que los 4 jugadores seleccionados por el usuario para crear el torneo sean distintos entre sí.
  - Los partidos generados respetan la regla de emparejamiento por ranking.