PRÁCTICA 3:

DISEÑO ARQUITECTONICO DE UNA APLICACIÓN

Óscar Bellón García
Sara Sánchez García
Saúl Santana Santana

Práctica 3: Diseño arquitectónico de una aplicación

Dominio y funcionalidad:

Continuando con la idea propuesta en la práctica anterior, para esta asignatura hemos decidido diseñar una aplicación nativa de android de asistencia para el tiro con arco. El objetivo principal es automatizar el proceso de puntaje de dicha actividad y el análisis de los resultados de los puntajes del arquero.

Los puntajes se realizan lanzando una cantidad determinada de tandas de flechas, tanto la cantidad de tandas como la de flechas por tanda varían según el formato o la categoría utilizada, nuestro objetivo es permitir que el usuario seleccione sus preferencias y la aplicación se encargue de calcular las puntuaciones y de darle información relevante al respecto.

Además queremos almacenar la información de manera temporal en la base de datos para ofrecer consejos e información en relación a la progresión deportiva del arquero.

Identificación de los principales elementos de la arquitectura:

La arquitectura que se ha decidido utilizar es la MVP. MVP es un patrón arquitectónico diseñado para facilitar las pruebas de unidad automatizada y mejorar la separación de inquietudes en la unidad lógica del presentador.

La arquitectura consta de tres elementos principales:

- El *modelo* es una interfaz que define los datos que se mostrarán y sobre los que actuará el usuario. Es la encargada de encapsular la lógica de negocio
- La *vista* es una interfaz pasiva que exhibe los datos del modelo y órdenes de usuario de las rutas (eventos) al presentador para actuar sobre los datos.
- El *presentador* actuará principalmente sobre la vista y hará peticiones al modelo para recuperar la información del mismo. Escucha a la vista para atender los eventos que suceden en la misma y actúa sobre el modelo recuperando los datos del modelo para presentarlos sobre la vista y actualizando los que va recibiendo por la misma.

En este caso particular, el modelo consta de tres tipos de objetos: usuarios, competición y puntuación.

El objeto usuario contiene los datos personales del usuario como el nombre, los apellidos, una foto de perfil, el club a que pertenece, el color de su dragonera (que indica hasta qué distancia tiene permitido tirar) y el arco que utiliza. También contiene un elemento booleano

para indicar si tiene las notificaciones activas, y una lista del objeto puntuación con todas las puntuaciones que ha guardado.

El objeto de la competición contiene información de las características de la competición como el nombre, el tipo de competición, la modalidad, la categoría, la distancia, número de flechas que se van a lanzar y una lista que almacena usuarios y sus puntuaciones.

La puntuación sería el objeto que almacena un diccionario de los puntos conseguidos y un dato fecha de tipo Date().

La vista de nuestra aplicación va a constar de varias pantallas las cuales van a tener la función de servir como vía para tomar y mostrar las estadísticas y datos relacionados con las tiradas.

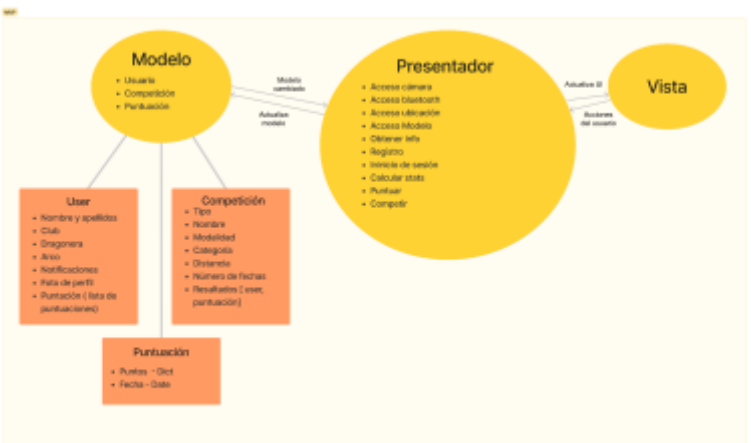
Nuestro presentador va a tener varias funcionalidades importantes puesto que va a ser el componente principal que interactúe con la vista y el modelo de datos. En primer lugar queremos aprovechar las ventajas de la programación nativa de dispositivos móviles con el acceso a distintos dispositivos del teléfono como la cámara para el registro de imágenes de las tiradas así como la ubicación de la misma. Además también queremos que nuestro presentador tenga la capacidad de interactuar con el bluetooth para obtener información de los dispositivos cercanos en las competiciones a través de la app.

Por otra parte también le brindaremos la posibilidad de interactuar con el modelo para obtener los datos de usuarios, puntuaciones y competiciones así como el soporte para registrar e iniciar sesión para los usuarios.

Además también incorporaremos un módulo de cálculo estadístico encargado de generar las gráficas de rendimiento a partir de los datos tomados y calculados de puntuaciones y competiciones realizados por el usuario.

Diseño de la Arquitectura:

En el siguiente diagrama se muestran los principales elementos de la arquitectura MVP, como se describieron en el apartado anterior: consta del modelo, el presentador y la vista. Las interacciones básicas de estos elementos también están descritas en el diagrama. El flujo habitual consiste en que la vista informe de las acciones que realiza el usuario al presentador, para que luego, el presentador actualice el modelo y, acto seguido, el modelo le devuelva la información al presentador; por último, el presentador actualizará la vista de la interfaz de usuario (UI).



Casos de uso:

Dentro de esta aplicación desarrollaremos dos casos de uso esenciales para el funcionamiento de la misma.

Iniciar Sesión:

Al abrir la aplicación por defecto se carga en la vista un call to action que urge al usuario a iniciar sesión o crear una nueva cuenta de usuario. Para el inicio de sesión la vista cuenta con un formulario que contiene dos campos a rellenar por el usuario. El usuario interactúa con la vista y rellena ambos campos. De nuevo el usuario ha de interactuar con la vista para que el presentador opere sobre estos elementos enviando el formulario. El presentador envía los datos recibidos al modelo, que conectará con firebase authentication recibiendo una respuesta que le permitirá conectar con la base de datos y recuperar el usuario de la misma.

En el caso de que la respuesta del servicio de autenticación sea correcta se procederá a hacer un retrieve de la información de la base de datos, desde el modelo. Una vez la información del usuario es cargada en el modelo se envía al presentador. El presentador del modelo recibe ciertos, entre ellos que el usuario está logueado. El presentador actualiza la vista indicando que se debe acceder a la pantalla de estadísticas. Finalmente el presentador opera sobre los datos obtenidos del usuario como sus puntuaciones para enviar toda esta información ya procesada a la vista, actualizando en este caso las gráficas del usuario.

En el caso de que la autenticación no haya sido exitosa el modelo enviará este fallo al presentador, y el presentador actualizará la vista indicando que ha habido un fallo de autenticación (por ejemplo una contraseña que no coincide con el usuario). El presentador procederá a actualizar la vista indicando el error que ha recibido e intentará repetir el proceso una vez el usuario interaccione con la vista.

Puntuar:

Damos por hecho que el usuario ya está logueado en el sistema y que su información se encuentra precargada en el modelo.

Al iniciar sesión la vista cargada inicialmente por el presentador es la pantalla de estadísticas. El usuario interactúa con la vista y clica sobre la barra de navegación para indicar que quiere acceder al apartado de puntuar. El presentador recibe el evento de la vista y procede a actualizar la misma con la pantalla de datos de puntuación. De nuevo el usuario tendrá que interactuar con la vista y rellenar los campos en el formulario que define el tipo de puntuación, una vez clique siguiente el presentador actualizará a la siguiente pantalla (tabla de puntuación) y enviará los datos recibidos a través de la vista al modelo. El modelo añadirá una entrada de puntuaciones sobre el usuario con los parámetros recibidos por el presentador.

En esta nueva vista de tabla el usuario irá rellenando la tabla a medida que vaya tirando. Cada vez que el usuario introduce nuevos datos el presentador enviará los datos al modelo para que actualice la entrada de referida a esta puntuación. El presentador con los datos que recibe de vuelta del modelo opera para averiguar el total de puntos y el número de dieces para actualizar el campo correspondiente en la vista. Al terminar de apuntar y clicar siguiente el presentador actualizará la vista y avanzará a la pantalla de resumen de puntuaciones.

El presentador recibe del modelo la entrada de puntuación completa y opera sobre los datos para conseguir los puntos totales, el promedio, los dieces, el promedio de dieces y la desviación típica. El usuario en esta vista tendrá la oportunidad de actualizar una vez más el modelo indicando su ubicación, y de la misma manera que anteriormente, los datos los leerá el presentador y los enviará al modelo para que se actualice. Cuando el usuario clique sobre guardar puntuación el presentador deberá indicar al modelo que actualice la información de la base de datos con la información contenida actualmente en el modelo, y, por otra parte, el presentador también mandará información a la vista para que esta se actualice de nuevo con la pantalla de estadísticas.

Conclusión:

Esta arquitectura nos permite separar y aislar los procesos críticos de la aplicación. El modelo trabajará con los datos y el presentador recibirá la información y actualizará las vistas.

Esta segregación reducirá la complejidad de los elementos de las vistas, que simplemente desplegarán la información que reciban por parte del presentador, el presentador se encargará de la lógica de interacción con la vista, pero los procesos relativos a la lógica de negocio serán transparentes pàra el, y por último el modelo llevará la mayor parte del trabajo, siendo el encargado de realizar la comunicación con la base de datos, el servicio de autenticación y manteniendo dentro su lógica de negocio.

La elección de esta arquitectura atiende a la pequeña escala del proyecto. El “MVP” nos permite encapsular la lógica de negocio separándola de las vistas y de la lógica de interacción, manteniendo una escalabilidad suficiente dentro del dominio de la aplicación sin tener que complicar de manera excesiva la arquitectura para una aplicación de pequeñas dimensiones.