He creado las pantallitas para acceder a la Configuración y la Clasificación aunque no tienen nada, pero bueno, para familiarizarme de nuevo con el código.

Después me he puesto con la maldita base de datos, al final lo he hecho no sé si igual o prácticamente igual que como está en el ejemplo que está subido al aula (copiado de alguien de internet btw), menos que ahí tiene una forma para mostrarlo todo por pantalla y blablablá y para insertar nuevos datos y eso a mí no me interesaba

Sin embargo, he creado otra función que devuelve una pregunta en concreto (QuestionEntity), pasándole el índice de la misma (tenemos una base de datos a la que aún hay que meter las preguntas correspondientes a los siguientes niveles). En principio, tal y como lo he hecho, creo que lo mejor sería hacer tres rangos de índices en función de la dificultad y ya está, porque si no toca hacer otras tablas y creo que todas las funciones y toda la pesca y buuuuufffff.

La cosa principal de esta mierda es que no se puede hacer con el hilo principal porque lo peta, así que te obliga a hacerlo a través de AsyncTasks, una movida porque además no te devuelve el valor en el momento y tal, así que de momento y hasta que no encuentre otra solución tengo puesto que la primera pregunta la cargue (se la meta al Quiz usando una variable estática) aún en la MainActivity, y que se vayan cargando siempre las preguntas siguientes. Por otro lado lo de no hacerlo en el thread principal te obliga a usar el modelview que tengo por ahí creado (se llama dbNew) para acceder a las funciones. Simplemente dbNew.loadQuestion(index que sea) y listo, pero que lo sepas.

A ver, qué más. Ahora mismo las preguntas no tienen imagen ni nada de eso (porque primero quería que funcionara), ni ninguna forma de comprobar si ya se han mostrado o no (creo que lo mejor sería hacer un mapa en el que guardar los índices de las que se han mostrado que andar modificando la base de datos y tener que hacer más búsquedas ahí y demás).

Además, aún se sigue haciendo la otra forma que teníamos de hacerlo en paralelo y por eso las imágenes que se muestran y la puntuación no tienen nada que ver.

Para mostrar las preguntas ahora lo hago hardcodeando, hay que pasarle el índice y creo que he descubierto por qué me estaba petando antes al hacerlo, así que lo voy a dejar preparado. Antes es que tenía el id como un long, y luego le pasaba un int y quizá no le gustaba, pero ya se me caen los ojos así que no voy a probarlo porque he borrado todo lo que tenía. Vale, sí que era por eso, simplemente calcula el índice y pásaselo. Hay que hacerlo tanto en la MainActivity para la primera, como en el Quizz donde lo teníamos antes también, me parece.

Tenía pensado que el índice se calculara como int index = r.nextInt(NUM\_PREGUNTAS\_POR\_DIFICULTAD-1) +dificultad (tendría que valer 0, 1 y 2 así que cambiar un par de cosas de cómo está hecho)\* NUM\_PREGUNTAS\_POR\_DIFICULTAD.

Así se cumple el rango de separación de ids

*\* index preguntas  
\* [0-13] Fácil  
\* [14-27] Medio  
\* [28-40] Difícil  
\*/*

Lo que haría ahora sería eso, comprobar que pasando el índice devuelve la pregunta que toca y tal, y cambiar lo que teníamos de antes para que deje de hacer cosas en paralelo y de momento, aunque no se vea ninguna imagen, no pasa nada, pero que el acceso a la base de datos funcione y tal, en modo fácil. Luego ya iremos añadiendo el resto de preguntas.

Y ya no se me ocurre qué más decir, así que cualquier duda, me preguntas. Quizá no te la puedo resolver pero :DDD

DÍA 27:

Creado en el Dao getPreguntasDificultad, que devuelve una lista con las QuestionEntity que pertenezcan a la dificultad pasada como parámetro. Cuando se hace la consulta realmente usando el loadQuestionAsyncTask, al recibir las preguntas que cumplen la consulta, se pasan a un arrayList de la Main Activity llamado preguntas. (He creado un ArrayList estático en la Main Activity porque es desde donde estoy llamando a que se carguen las preguntas, así al iniciar el quiz como tal es más probable que ya estén cargadas, y no podía pasárselo directamente al array de Quizz porque aún no existía, no estaba inicializado).

En la clase Quizz paso por referencia a q el valor de MainActivity.preguntas y así ya sólo trabajamos con q para coger las preguntas y demás. Cosas a saber: he tenido que poner que QuestionEntity implemente Serializable porque si no no podía meterlo en el ArrayList.

Además ya he puesto que se calcule el índice y tal más o menos como lo teníamos en la práctica anterior (cuidado no confundir n\_question (que es el número ordinal de las preguntas para mostrarlo por pantalla) con index\_question(que es el índice de la pregunta real que se va a mostrar)). Ah, otra cosa a saber, en las bases de datos no hay que poner cosas con id 0, no sé por qué, pero no le gustan, me lo pillaba a partir del 1 todo el rato, así que le he puesto al que tenía el 0 el 14 y tirando (internamente se ordena así que cuando te devuelve las preguntas la de índice 0 corresponde al id 1 y tal, da igual en qué orden se hayan metido en la base de datos).

Creo que por último así para saber, he creado un HashMap al que pasarle el valor de index\_question de las preguntas que se muestran y así al calcular el index\_question de la siguiente primero comprueba que no esté ya ahí.

AHH, y hay que intentar poner que verifique si la base de datos ya tiene las cosas o no y comprobar si funciona, para que no esté todo el rato borrando todo y metiéndolo de cero, que lo que tenía lo he comentado y hay que comprobar si funciona.

He metido todas las preguntas de diferentes dificultades y pues que cargue las preguntas una vez seleccionada la dificultad y luego el loadQuestionAsyncTask coja del parámetro mandado la dificultad de la que tiene que buscar las preguntas. De esa forma no da tiempo y peta porque quiz intenta acceder antes de tenerlas así que le he metido un sleep de 3 segundos y no resulta molesto que se pare a cargar, si vemos que es demasiado hacemos entre medias una activity simplemente de carga en la que ponga Cargando y listo.

He mirado muchas otras páginas pero pasando

<https://android.jlelse.eu/5-steps-to-implement-room-persistence-library-in-android-47b10cd47b24>

<https://www.upwork.com/hiring/mobile/why-you-should-use-asynctask-in-android-development/>

<https://android.jlelse.eu/5-steps-to-implement-room-persistence-library-in-android-47b10cd47b24>