Instituto Tecnológico y de Estudios

Superiores de Occidente.

Practica2: Hive y Bloc

**Materia**: Programación de dispositivos móviles

**Profesor**: Francisco Javier Camacho Gil

**Fecha**: 19/10/2020

**Autor(es)**: Francisco Andres De La Torre Valle

## Introducción

El propósito de la tarea es aplicar los temas vistos en clase, en el transcurso de las sesiones previas al examen, de esta forma servirá de práctica para estudiar antes de la evaluación parcial.

Se requiere hacer una aplicación con diseño basado en lineamientos de material design, la cual aproveche de manera óptima el espacio en pantalla, además de implementar base de datos local para persistencia de datos y BLoC para manejo de estados en la UI de la aplicación.

# Desarrollo

***Fase 0:***

**Explica con tus palabras lo que es la persistencia de datos y para qué se utiliza:**

* **¿Cuál es la función principal de serializar y deserializar JSON? (describir)**

Esto se encarga de poder transformar objetos con todos sus atributos en texto plano para poder ser manejados o enviados de manera sencilla y posteriormente reinterpretados como los objetos originales a base de signos de puntuación. (Serialización/deserialización de objetos y transmisión de datos con JSON: una revisión de la literatura, 2015)

* **¿Qué  es un stream y para qué se utiliza?**

Estos serían los flujos continuos de datos, esto sirve para poder mostrar información que se actualiza en tiempo real y que puede reaccionar de manera inmediata. (Garcia, 2016)

* **¿En qué  consiste el patrón de arquitectura BLoC?.**

Este es un patron que se encarga de centralizar la lógica de negocio, centralizar cambios de estado y mapear al formato que necesita la vista con base en que Por cada concepto lo suficientemente importante vamos a tener su correspondiente BloC. (Fernández, 2020)

* **¿Cuál es el objetivo de utilizar un *FutureBuilder()*?**

Poder crear o manejar los estados y constructores sin preocuparse por deadlocks u otras amenazas por diseño a base de construir con el estado actual y un constructor a futuro, esto comprobando también el estado futuro. (Google Developers, 2018)

* **¿Para que se utiliza el *Build Runner*? (describir)**

Para generar tanto archivos nuevos como archivos de testing, genera archivos output basándose en archivos existentes de tipo input. (Thomsen, 2019)

* **Describe el proceso para implementar BLoC en una app (cómo se estructura, etc.)**

Lo primero sería descargar las bibliotecas respectivas en el archivo pubspec.yaml, luego de esto es hacer click derecho sobre la carpeta en la que se quiera estar trabajando y seleccionar la opción de un bloc nuevo, esto generara los archivos de bloc,state y events, en estos se ven la logística de los eventos, sus operaciones y en que estados resultan, los posibles estados en que podría caer una operación de la aplicación, y los eventos que esta estará experimentando, respectivamente.

**Enlista cuatro opciones para almacenamiento de datos local con una pequeña descripción y al menos un pro y un contra por opción.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Opción** | **Descripción** | **Pros** | **Contras** |
| **SQLite** | “Sistema gestor de bases de datos multiplataforma con un motor ligero.” | “Las bases de datos se guardan como ficheros, con esto se facilita el translado de las bases de datos” | No es escalabre ni rcapaz de operar con bases de datos grandes |
| **Apache Derby** | “Sirve para bases de datos relacionales, es de código abierto, implementado totalmente en Java.” | Fácil de usar, tamaño reducido y soporta paradigma cliente-servidor | Lento y no tan escalable |
| **HSQLDB** **(Hyperthreaded Structured** **Query Language database)** | “Se distribuye bajo la licencia BSD (Berkeley Software Distribution) es una licencia muy cercana a dominio público.” | “Ofrece un engine rápido multihilo para el manejo de datos” | Difícil de acceder a la información fuera de la app, entre más información exista más lenta será. |
| **H2** | “Es un sistema gestor de bases de datos relacionales programado en Java. Disponible como software de código abierto, bajo la licencia pública de Mozilla.” | Rápida, pequeña en tamaño y puede manejar estructura de cliente servidor | Movimientos de combinado de información le son imposibles en ciertos casos (merge y join) y el problema de lentitud relacionada a la cantidad de información es peor que en HSQLBD |

Para el desarrollo de la aplicación se comenzó con descargar el proyecto existente y usando el comando flutter pub get para poder correr el código existente y ver las anotaciones que hacían falta. Lo primero que se hiso fue establecer el objeto de base de datos del Hive, y seguido a eso el adapter, pues facilita tener estas piezas ya establecidas e instaladas, porque se necesitaban para la lógica en los bloc y para poder correr los comandos de creación.

El procedimiento desde ahí fue revisar todos los TODO e irlos resolviendo uno por uno, los más sencillos fueron los de añadir y eliminar elementos en el bloc, una vez con esto se encontró un problema con la manera en que se cargaban los datos al principio, principalmente debido a que no se estaban cargando, se opto por aplazar esa parte hasta cambiar un poco el diseño de la app con colores personalizados para familiarizarse mejor con su estructura, luego se experimento con el objeto de la base de datos en si hasta que poco a poco se llego a una manera de sacar los datos que funcionara.

Trabajar con la lógica de Bloc no fue muy difícil pues esta es clara, ya que tiene estados que describen si la base esta vacía, si se ah añadido un elemento, si está esperando después de añadir un elemento, si fue borrado etc, y la estructura general es que mientras no exista la base de datos o esta esté vacía solo se mostrara un mensaje de agregar contenido, luego la lógica de mostrar la lista solo se activa si la bd no esta vacía, y la lógica añadida fue un snackbar cuando se esperará después de añadir un elemento.

En la interfaz solo se aprecia un botón, pero al activarlo esta manda llamar una función que construye un pequeño recuadro que contiene un cuadro de texto y un reloj para escoger la hora, este campo se actualiza no al poner la hora, sino al volver a seleccionar ese recuadro.

# Conclusiones

El uso de Bloc para el manejo de estados es necesario para mantener un control y orden de las acciones que una aplicación podría tener, ya que simplifica la manera en que se harán las tareas después de interactuar con alguna parte de la app, Así mismo el uso de Hive para el almacenamiento local y de un adapter facilitan el uso de los datos dinámicos que se pueden emplear para darle funcionalidad al proyecto.

Observaciones

Lo que más me costó en esta práctica fue la logística para sacar los datos de la base de datos desde el inicio, pues fue lo ultimo que pude hacer, hasta mostrar la base después de interactuar con ella fue más sencillo, pero después de evaluar todas las piezas móviles que tiene el objeto de Box y entender cómo se podía parsear fue más sencillo, fuera de eso todo lo demás fue sencillo de entender pues se ah estado revisando en todas las otras tareas y trabajos.

Video:

Se encuentra en el repositorio.

GITHUB:

<https://github.com/FrankieDLT/Practica2>

# Bibliografía

A., D. (20 de Diciembre de 2019). *SQLite vs MySQL: ¿Cuál es la diferencia?* Obtenido de https://www.hostinger.es/tutoriales/sqlite-vs-mysql-cual-es-la-diferencia/

*Apache Derby- Great for localized Risk Modeling*. (13 de Junio de 2018). Obtenido de https://www.trustradius.com/products/derby/reviews?qs=pros-and-cons

Fernández, J. S. (26 de Marzo de 2020). *Introducción al patrón BLoC*. Obtenido de Xurxodev: http://xurxodev.com/introduccion-al-patron-bloc/

Garcia, A. (1 de Diciembre de 2016). *¿Qué es y para qué sirve un stream?* Obtenido de Quora: https://es.quora.com/Qu%C3%A9-es-y-para-qu%C3%A9-sirve-un-stream

Google Developers. (18 de Septiembre de 2018). *FutureBuilder - El widget de Flutter de la semana*. Obtenido de Youtube: https://www.youtube.com/watch?v=ek8ZPdWj4Qo

Romero, O. (2 de Febrero de 2016). *Bases de datos embebidas*. Obtenido de Blog Informático de Óscar : https://oscarromeroblog.wordpress.com/2016/02/21/bases-de-datos-embebidas/

*Serialización/deserialización de objetos y transmisión de datos con JSON: una revisión de la literatura*. (22 de Abril de 2015). Obtenido de Juan Antonio Mora-Castillo: https://www.scielo.sa.cr/pdf/tem/v29n1/0379-3982-tem-29-01-00118.pdf

Thomsen, M. (21 de Mayo de 2019). *build\_runner*. Obtenido de Dart: https://dart.dev/tools/build\_runner

Unknown. (30 de Diciembre de 2012). *Choosing Light Weight Java Database* . Obtenido de http://lightjavadatabase.blogspot.com/2012/12/choosing-light-weight-java-database.html

# Criterio de evaluación ponderado.

|  |  |  |  |
| --- | --- | --- | --- |
| **Fase** | **Descripción** | **Puntos alcanzables** | **Puntos obtenidos** |
| **Fase 0** | Respuesta a la preguntas de manera correcta, además de fuentes bibliográficas. | **10 pts** | 10 |
| **Fase 1** | Se implementa persistencia de datos de manera correcta con Hive.  Se guardan los datos y se cargan después de cerrar y abrir la app. | **35 pts** | 35 |
| **Fase 2** | Se utiliza BLoC para guardar y cargar datos, así como manejo de estados y eventos que actualizan la UI. | **40 pts** | 40 |
| **Fase 3** | Se muestra una bottom sheet para agregar a la lista, al igual que al hacer swipe de un elemento, este se elimina de la lista y de la base de datos local. | **15 pts** | 15 |
| **Extra\*** | Link de video con su app funcionando. | **2 pts** | 2 |