Tarea de las unidades 3 y 4

Unidad 3: Arquitecturas de software

1.- Marca con una X los ejercicios y prácticas realizados:

(X) **Ejercicio:** Encapsular código para guardar último libro leído

(X) **Ejercicio:** Uso del patrón Observer

(X) **Ejercicio:** Uso del patrón Null Object en la clase Libro

(X) **Ejercicio:** Uso del patrón Command

(X) **Práctica:** Uso del patrón Command con LongClickListener

(X) **Ejercicio:** Uso del patrón Adapter

(X) **Ejercicio:** Aplicar patrón builder en la clase Libro

(X) **Práctica:** Uso del patrón builder en la clase Libro

(X) **Ejercicio:** Acceso a último libro leído con el patrón Singleton

(X) **Práctica:** Acceso a los libros con el patrón Singleton

(X) **Práctica:** Acceso a Volley con el patrón Singleton

(X) **Ejercicio:** Uso de Model View Controller

(X) **Ejercicio:** Uso de Model View Presenter

(X) **Práctica:** Uso de MVP en DetalleFragment\*

(X) **Ejercicio:** Uso de Casos de uso

(X) **Ejercicio:** Uso de Repositorio para abstraer

(\_) **Práctica:** Uso de la arquitectura CLEAN en DetalleFragment\*

2.- En caso de haber realizado alguna práctica marcada con \*, copia el código obtenido:

Detalle Fragment

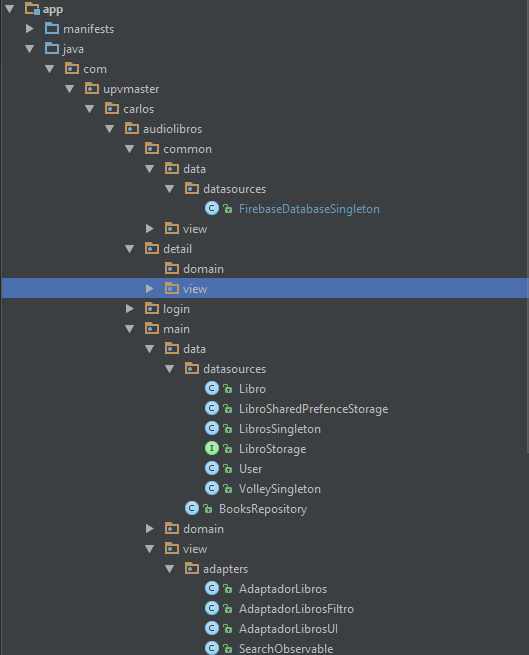
public class DetalleFragment extends Fragment implements View.OnTouchListener, MediaPlayer.OnPreparedListener, MediaController.MediaPlayerControl,DetallePresenter.View {  
 public static String *ARG\_ID\_LIBRO* = "id\_libro";  
 private View vista;  
 private DetallePresenter presenter;  
 private ZoomSeekBar zoombar;  
  
  
 @Override  
 public View onCreateView(LayoutInflater inflador, ViewGroup contenedor, Bundle savedInstanceState) {  
 vista = inflador.inflate(R.layout.*fragment\_detalle*, contenedor, false);  
 Bundle args = getArguments();  
 String key="";  
 if (args != null) {  
 key = args.getString(*ARG\_ID\_LIBRO*);  
 }  
 presenter = new DetallePresenter(vista.getContext(),new MediaController(getActivity()), this);  
 presenter.ponInfoLibro(key);  
 //Poner aquí los cambios en ZoomSeekBar  
 zoombar = (ZoomSeekBar) vista.findViewById(R.id.*zoombar*);  
 zoombar.setVisibility(View.*INVISIBLE*);  
 //Probar con el seekTo para cambiar el zoomSeekBar  
 zoombar.setOnZoomSeekBarListener(presenter.getlistenerZoomBar());  
 //Make sure you update Seekbar on UI thread  
 getActivity().runOnUiThread(presenter.getRun\_tiempo(zoombar));  
 return vista;  
 }  
  
 public void ponInfoLibro(String key) {  
 presenter.ponInfoLibro(key);  
 }  
  
 @Override  
 public void onPrepared(MediaPlayer mediaPlayer) {  
 presenter.onPrepareMediaPlayer(getActivity(),getView().findViewById(R.id.*fragment\_detalle*),this,zoombar);  
 }  
  
 @Override  
 public void onResume() {  
 DetalleFragment detalleFragment = (DetalleFragment) getFragmentManager().findFragmentById(R.id.*detalle\_fragment*);  
 if (detalleFragment == null) {  
 ((MainActivity) getActivity()).mostrarElementos(false);  
 }  
 super.onResume();  
 }  
  
 @Override  
 public boolean onTouch(View vista, MotionEvent evento) {  
 presenter.showMediaController();  
 return false;  
 }  
  
 @Override  
 public void onStop() {  
 presenter.onStop();  
 super.onStop();  
 }  
  
 @Override  
 public boolean canPause() {  
 return true;  
 }  
  
 @Override  
 public boolean canSeekBackward() {  
 return true;  
 }  
  
 @Override  
 public boolean canSeekForward() {  
 return true;  
 }  
  
 @Override  
 public int getBufferPercentage() {  
 return 0;  
 }  
  
 @Override  
 public int getCurrentPosition() {  
 try {  
 return presenter.getCurrentPosition();  
 } catch (Exception e) {  
 return 0;  
 }  
 }  
  
 @Override  
 public int getDuration() {  
 return presenter.getDuration();  
 }  
  
 @Override  
 public boolean isPlaying() {  
 return presenter.audioIsPlaying();  
 }  
  
 @Override  
 public void pause() {  
 presenter.pauseAudio();  
 }  
  
 @Override  
 public void seekTo(int pos) {  
 presenter.seekToAudio(pos);  
 }  
  
 @Override  
 public void start() {  
 presenter.startAudio();  
 }  
  
 @Override  
 public int getAudioSessionId() {  
 return 0;  
 }  
  
 @Override  
 public void ponInfoLibro(Libro book) {  
 ((TextView) vista.findViewById(R.id.*titulo*)).setText(book.getTitulo());  
 ((TextView) vista.findViewById(R.id.*autor*)).setText(book.getAutor());  
 ((NetworkImageView) vista.findViewById(R.id.*portada*)).  
 setImageUrl(book.getUrlImagen(), VolleySingleton.*getInstance*(vista.getContext()).*getLectorImagenes*());  
 vista.setOnTouchListener(this);  
 Uri audio = Uri.*parse*(book.getUrlAudio());  
 presenter.initMediaPlayer(this,audio);  
 }

Detalle Presenter

public class DetallePresenter {  
 private Libro book;  
 private MediaPlayer mediaPlayer;  
 private MediaController mediaController;  
 private Handler mHandler;  
 private Runnable run\_tiempo;  
 private Context context;  
 private View view;  
  
 public DetallePresenter(Context context,MediaController mediaController, View view) {  
 this.context = context;  
 this.mediaPlayer = new MediaPlayer();  
 this.mediaController = mediaController;  
 this.mHandler = new Handler();  
 this.view = view;  
 }  
  
 public OnZoomSeekBarListener getlistenerZoomBar(){  
 return new OnZoomSeekBarListener() {  
 @Override  
 public void colocarAudio(int posicion) {  
 if(mediaPlayer!=null)  
 mediaPlayer.seekTo(posicion\*1000);  
 }  
 };  
 }  
  
 public void ponInfoLibro(String key) {  
 this.book = LibrosSingleton.*getInstance*(context).getAdaptadorLibros().getItemByKey(key);  
 view.ponInfoLibro(book);  
 }  
  
 public void onPrepareMediaPlayer(Context context, android.view.View anchovista, MediaController.MediaPlayerControl controller, ZoomSeekBar zoombar){  
 Log.*d*("Audiolibros", "Entramos en onPrepared de MediaPlayer");  
 SharedPreferences preferencias = PreferenceManager.*getDefaultSharedPreferences*(context);  
 if (preferencias.getBoolean("pref\_autoreproducir", true)) {  
 mediaPlayer.start();  
 }  
 mediaController.setMediaPlayer(controller);  
 mediaController.setAnchorView(anchovista);  
 mediaController.setPadding(0, 0, 0, 110);  
 mediaController.setEnabled(true);  
 mediaController.show();  
 //Poner el Zoombar  
 int duracionAudio = mediaPlayer.getDuration() / 1000;  
 zoombar.setValMin(0);  
 zoombar.setEscalaMin(0);  
 zoombar.setEscalaIni(0);  
 zoombar.setEscalaRaya(duracionAudio/50);  
 zoombar.setEscalaRayaLarga(10);  
 zoombar.setValMax(duracionAudio);  
 zoombar.setEscalaMax(duracionAudio);  
 zoombar.setVisibility(android.view.View.*VISIBLE*);  
 }  
  
 public void onStop(){  
 hideMediaController();  
 mHandler.removeCallbacks(run\_tiempo);  
 try {  
 mediaPlayer.stop();  
 mediaPlayer.release();  
 } catch (Exception e) {  
 Log.*d*("Audiolibros", "Error en mediaPlayer.stop()");  
 }  
 }  
  
 public Runnable getRun\_tiempo(final ZoomSeekBar zoomView){  
 run\_tiempo = new Runnable() {  
  
 @Override  
 public void run() {  
 if(mediaPlayer != null){  
 int mCurrentPosition = mediaPlayer.getCurrentPosition() / 1000;  
 zoomView.setVal(mCurrentPosition);  
 }  
 mHandler.postDelayed(this, 1000);  
 }  
 };  
 return run\_tiempo;  
 }  
  
 public void initMediaPlayer(MediaPlayer.OnPreparedListener onPreparedListener, Uri audio){  
 if (mediaPlayer != null) {  
 mediaPlayer.release();  
 }  
 mediaPlayer = new MediaPlayer();  
 mediaPlayer.setOnPreparedListener(onPreparedListener);  
 try {  
 mediaPlayer.setDataSource(context,audio);  
 mediaPlayer.prepareAsync();  
 } catch (IOException e) {  
 Log.*e*("Audiolibros", "ERROR: No se puede reproducir " + audio, e);  
 }  
 }  
  
 public void startAudio(){  
 mediaPlayer.start();  
 }  
 public void pauseAudio(){  
 mediaPlayer.pause();  
 }  
 public void stopAudio(){  
 mediaPlayer.stop();  
 }  
 public void seekToAudio(int pos){  
 mediaPlayer.seekTo(pos);  
 }  
 public int getDuration(){  
 return mediaPlayer.getDuration();  
 }  
 public boolean audioIsPlaying(){  
 return mediaPlayer.isPlaying();  
 }  
 public int getCurrentPosition(){  
 return mediaPlayer.getCurrentPosition();  
 }  
 public void showMediaController(){  
 mediaController.show();  
 }  
 public void hideMediaController(){  
 mediaController.hide();  
 }  
  
  
  
 public interface View {  
 void ponInfoLibro(Libro book);  
 }  
}

3.- En caso de haber realizado alguna refactorización adicional, explica brevemente y copia el código obtenido:

Se ha restructurado el proyecto



Unidad 4: Firebase: Autentificación y bases de datos

1.- Marca con una X los ejercicios y prácticas realizados:

(X) **Ejercicio:** Ejercicio: Añadir Firebase a Audiolibros.

(X) **Ejercicio:** Autentificación con correo y contraseña y Google

(X) **Ejercicio:** Acceso a datos del usuario

(X) **Ejercicio:** Cerrar sesión

(\_) **Práctica:** Verificación de correo

(X) **Ejercicio:** Autentificación con Facebook

(X) **Ejercicio:** Autentificación con Twitter

(X) **Ejercicio:** Añadir fotografía de usuario

(X) **Ejercicio:** Autentificación personalizada con correo y contraseña

(X) **Ejercicio:** Autentificación personalizada con Google

(X) **Ejercicio:** Autentificación personalizada con Facebook

(X) **Ejercicio:** Autentificación personalizada con Twitter

(X) **Práctica:** Permitir las dos alternativas de autentificación

(X) **Ejercicio:** Importar/Exportar datos desde un fichero JSON

(X) **Ejercicio:** Configurar la base datos y referencias

(X) **Ejercicio:** Modificar POJO de libros

(X) **Ejercicio:** Crear POJO de usuarios

(X) **Ejercicio:** Guardar en la base de datos la información del usuario

(X) **Ejercicio:** Mostrar datos con FirebaseRecyclerAdapter

(X) **Ejercicio:** Mostrar datos con filtrado local

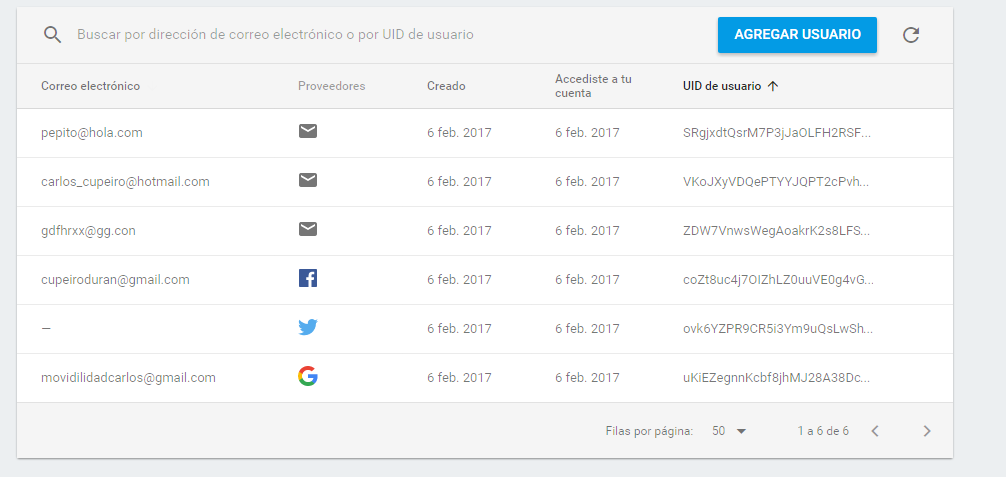
(X) **Ejercicio:** AdaptadorLibros basado en Firebase SDK

(\_) **Práctica:** Almacenar lecturas de cada usuario

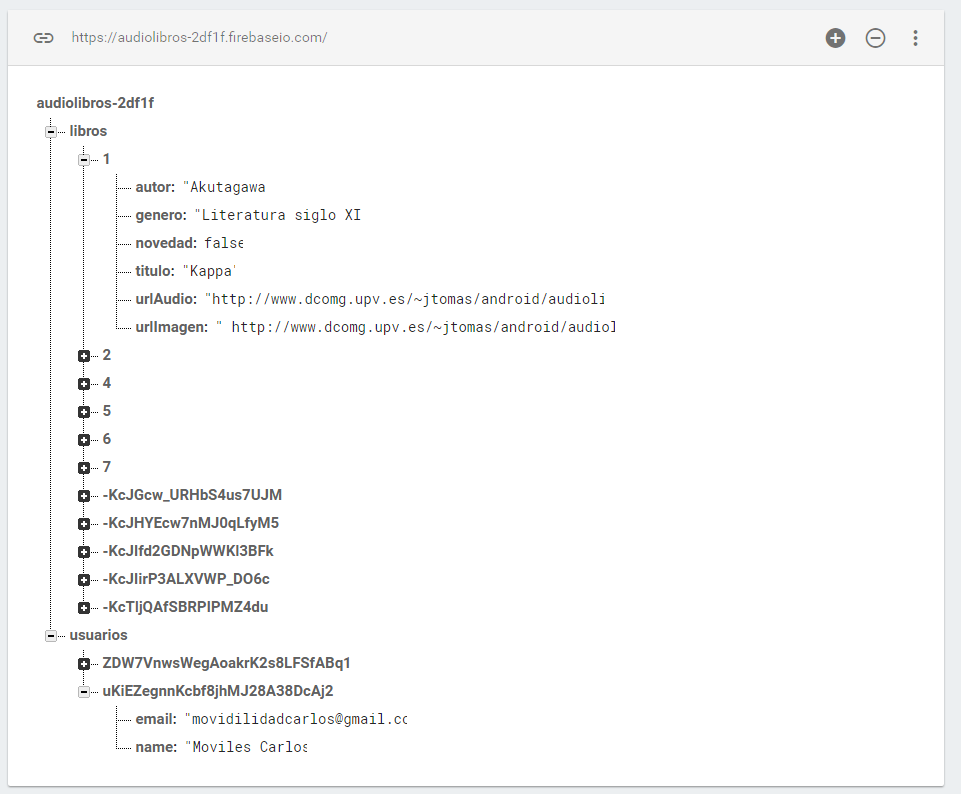
(\_) **Práctica:** Almacenar lecturas en estructura eficiente

(\_) **Práctica:** Almacenar usuario que creó cada libro

2.- Pega una captura, donde se muestren los usuarios que han accedido a tu aplicación:



3.- Pega una captura, donde se muestre la estructura de datos de tu aplicación (trata de que se vea un nodo de cada tipo, totalmente extendido):



4.- Respuesta de la Práctica Almacenar lecturas en estructura eficiente:

5.- En caso de haber realizado alguna mejora adicional, explica brevemente y copia el código obtenido: