Raptor

Generated by Doxygen 1.13.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 ExampleUnitTest Class Reference

Example local unit test, which will execute on the development machine (host).

4.1.1 Detailed Description

Example local unit test, which will execute on the development machine (host).

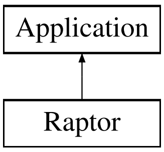
See `testing documentation`.

4.2 LogicDao Interface Reference

4.3 Raptor Class Reference

Dummy class for dependency injection.

Inheritance diagram for Raptor:



4.3.1 Detailed Description

Dummy class for dependency injection.

Can store application-wide data.

4.4 UIDao Interface Reference

Chapter 5

File Documentation

5.1 ExampleInstrumentedTest.kt File Reference

Functions

- RunWith (AndroidJUnit4::class) class ExampleInstrumentedTest

  Instrumented test, which will execute on an Android device.

5.1.1 Function Documentation

5.1.1.1 RunWith()

```
RunWith (
             AndroidJUnit4.class )
```

Instrumented test, which will execute on an Android device.

See `testing documentation`.

5.2 AudioPlayer.kt File Reference

Functions

- class AudioPlayer Inject constructor (@ApplicationContext context:Context)

  This class handles audio playback in the media player, by wrapping around an `ExoPlayer` *instance.*

5.2.1 Function Documentation

5.2.1.1 constructor()

```
class AudioPlayer Inject constructor (
            @ApplicationContext context:Context )
```

This class handles audio playback in the media player, by wrapping around an `ExoPlayer` instance.

It handles the logic of media controls, what song to play, at what time, etc. Gets the current playback position of the currently playing song in milliseconds

Gets the duration of the currently playing song in milliseconds

Enum describing all states that the player can be in

Gets the current playback state of the player, as described in `PlaybackStates`

Plays a given `Song` and sets it as the current song

Player can't be in the playing state at the time of calling.

Parameters

song	`Song` to play

Restarts playback from the beginning of the current song

Pauses the currently playing song

Player must be in the playing state when calling

Changes the currently playing song

Parameters

song	song to change to

Changes the timeline position of the playback to some value in milliseconds

Parameters

time	value in milliseconds to change playback to

Releases the internal `ExoPlayer` object

Must be called in order to free memory, when an object of this class is no longer used

5.3 BiometricAuthenticator.kt File Reference

Functions

- enum BiometricAuthenticationStatus (val id:Int)
- class BiometricAuthenticator (private val context:Context)

5.3.1 Function Documentation

5.3.1.1 BiometricAuthenticationStatus()

```
enum BiometricAuthenticationStatus (
             val id:Int )
```

5.3.1.2 BiometricAuthenticator()

```
class BiometricAuthenticator (
             private val context:Context )
```

5.4 DatabaseManager.kt File Reference

Functions

- Singleton class DatabaseManager Inject constructor (@ApplicationContext val context:Context)

  This class serves as an intermediary between the application database and the rest of the application.

5.4.1 Function Documentation

5.4.1.1 constructor()

```
Singleton class DatabaseManager Inject constructor (
              @ApplicationContext val context:Context )
```

This class serves as an intermediary between the application database and the rest of the application.

This class is a singleton Collects a flow of all authors from the database.

Returns

A Flow emitting a list of Author objects.

Collects a flow of albums by a specific author from the database.

Parameters

authorName	The name of the author whose albums are to be collected.

Returns

A Flow emitting a list of Album objects.

Collects a flow of songs by a specific album from the database.

Parameters

albumId	The ID of the album whose songs are to be collected.

Returns

A Flow emitting a list of Song objects.

Collects a flow of a specific song from the database.

Parameters

songId	The ID of the song to be collected.

Returns

A Flow emitting a Song object.

Retrieves a specific song from the database.

Parameters

song↩ Id	The ID of the song to be retrieved.

Returns

A Song object.

Collects a flow of authors of a specific song from the database.

Parameters

song	The Song object whose authors are to be collected.

Returns

A Flow emitting a list of Author objects or null.

Collects a flow of a specific album from the database.

Parameters

album↩ Id	The ID of the album to be collected.

Returns

A Flow emitting an Album object.

Populate the database with a list of `SongInfo`

5.5 Album.kt File Reference

5.6 AlbumAuthorCrossRef.kt File Reference

5.7 Author.kt File Reference

5.8 Song.kt File Reference

5.9 LibraryDb.kt File Reference

Functions

- Database (entities=[Song::class, Author::class, Album::class, AlbumAuthorCrossRef::class], version=1) override class LibraryDb

5.9.1 Function Documentation

5.9.1.1 Database()

```
Database (
             entities  = [Song::class, Author::class, Album::class, AlbumAuthorCrossRef:↩
:class],
             version  = 1)  [pure virtual]
```

5.10 LogicDao.kt File Reference

Classes

- interface LogicDao

5.11 AlbumWithAuthors.kt File Reference

5.12 AlbumWithSongs.kt File Reference

5.13 AuthorWithAlbums.kt File Reference

5.14 UIDao.kt File Reference

Classes

- interface UIDao

5.15 ImageManager.kt File Reference

Functions

- class ImageManager Inject constructor (@ApplicationContext private val context:Context)

  Handles the extraction of album images from app storage and provides images from app storage to the rest of the program.

5.15.1 Function Documentation

5.15.1.1 constructor()

```
class ImageManager Inject constructor (
             @ApplicationContext private val context:Context )
```

Handles the extraction of album images from app storage and provides images from app storage to the rest of the program.

Extracts the image directly from some audio file and copies it into app storage

Parameters

uri	Image URI
artistNames	List of names of the composers of the file
albumName	Name of the album of the file

Returns

URI of the file created in app storage or `null`

Retrieves a image bitmap in local storage

Parameters

uri	URI of the image file from local storage

Returns

ImageBitmap from storage or blank bitmap

5.16 MainActivity.kt File Reference

Functions

- Composable fun MainScreen (libraryViewModel:LibraryViewModel=hiltViewModel())
- Composable fun MainScreenContent (navController:NavHostController, libraryViewModel:LibraryViewModel)
- Composable fun AuthorTile (author:String, onClick:() -> Unit)
- Composable fun AlbumsScreen (navController:NavHostController, author:String,)
- Composable fun PortraitView (selectedAlbum:Pair< Album, ImageBitmap >, navController:NavHost↩ Controller)
- Composable fun LandscapeView (selectedAlbum:Pair< Album, ImageBitmap >, navController:NavHost↩ Controller)
- Composable fun SongsScreen (navController:NavHostController, libraryViewModel:LibraryViewModel, albumId:Long?, modifier:Modifier=Modifier)
- Composable fun SongItem (song:Song, navController:NavHostController)
- Composable fun isPortrait ()
- Composable fun AuthenticationScreen (onAuthenticate:() -> Unit)

Variables

- lateinit var sensorManager
- lateinit var SensorEventListener

5.16.1 Function Documentation

5.16.1.1 AlbumsScreen()

```
Composable fun AlbumsScreen (
            navController:NavHostController ,
            author:String )
```

5.16.1.2 AuthenticationScreen()

```
Composable fun AuthenticationScreen (
            onAuthenticate:() -> Unit)
```

5.16.1.3 AuthorTile()

```
Composable fun AuthorTile (
            author:String ,
            onClick:() -> Unit)
```

5.16.1.4 isPortrait()

```
Composable fun isPortrait ()
```

5.16.1.5 LandscapeView()

```
Composable fun LandscapeView (
            selectedAlbum:Pair< Album, ImageBitmap > ,
            navController:NavHostController )
```

5.16.1.6 MainScreen()

```
Composable fun MainScreen (
            libraryViewModel:LibraryViewModel  = hiltViewModel())
```

5.16.1.7 MainScreenContent()

```
Composable fun MainScreenContent (
            navController:NavHostController ,
            libraryViewModel:LibraryViewModel )
```

5.16.1.8 PortraitView()

```
Composable fun PortraitView (
            selectedAlbum:Pair< Album, ImageBitmap > ,
            navController:NavHostController )
```

5.16.1.9 SongItem()

```
Composable fun SongItem (
            song:Song ,
            navController:NavHostController )
```

5.16.1.10 SongsScreen()

```
Composable fun SongsScreen (
             navController:NavHostController ,
             libraryViewModel:LibraryViewModel ,
             albumId:Long?  ,
             modifier:Modifier  = Modifier)
```

5.16.2 Variable Documentation

5.16.2.1 SensorEventListener

```
lateinit var SensorEventListener
```

5.16.2.2 sensorManager

```
lateinit var sensorManager
```

5.17 MusicFileLoader.kt File Reference

Functions

- class MusicFileLoader Inject constructor (@ApplicationContext private val context:Context)

  Handles the loading of music files in a given directory and its subdirectories as well as launching a file picker instance.

5.17.1 Function Documentation

5.17.1.1 constructor()

```
class MusicFileLoader Inject constructor (
              @ApplicationContext private val context:Context )
```

Handles the loading of music files in a given directory and its subdirectories as well as launching a file picker instance.

Dataclass representing a song file

Parameters

filename	Name of the file
uri	`Uri` corresponding to the file
mimeType	The type of the file

Observable list of `SongFile` currently loaded

@Composable function that prepares the file picker launcher

Must be called before `launch()`

Launches the file picker

5.18 Raptor.kt File Reference

Classes

- class Raptor

  Dummy class for dependency injection.

5.19 SongPlayUI.kt File Reference

Functions

- Composable fun MediaControls (playerViewModel:PlayerViewModel)
- Composable fun CurrentSongInfo (title:String?, artists:String?, cover:ImageBitmap, modifier:Modifier)
- SuppressLint ("StateFlowValueCalledInComposition") @Composable fun SongPlayUI(songId

5.19.1 Function Documentation

5.19.1.1 CurrentSongInfo()

```
Composable fun CurrentSongInfo (
             title:String?  ,
             artists:String?  ,
             cover:ImageBitmap ,
             modifier:Modifier )
```

5.19.1.2 MediaControls()

```
Composable fun MediaControls (
             playerViewModel:PlayerViewModel )
```

5.19.1.3 SuppressLint()

```
SuppressLint (
             "StateFlowValueCalledInComposition" )
```

5.20 WaveFormSeekBar.kt File Reference

Functions

- Composable fun WaveformSeekBar (waveform:List< Float >, progress:Float, modifier:Modifier=Modifier, onProgressChanged:(Float) -> Unit)

5.20.1 Function Documentation

5.20.1.1 WaveformSeekBar()

```
Composable fun WaveformSeekBar (
            waveform:List< Float > ,
            progress:Float ,
            modifier:Modifier  = Modifier,
            onProgressChanged:(Float) -> Unit)
```

5.21 TagExtractor.kt File Reference

Functions

- class TagExtractor Inject constructor (@ApplicationContext private val context:Context, private val imageManager:ImageManager)

  This class handles metadata extraction from a collection of `SongFile`.

5.21.1 Function Documentation

5.21.1.1 constructor()

```
class TagExtractor Inject constructor (
             @ApplicationContext private val context:Context ,
            private val imageManager:ImageManager )
```

This class handles metadata extraction from a collection of `SongFile`.

It will also copy all album covers to app storage Dataclass representing the tags a song has.

Parameters

artists	A list of artist names that authored the song
album	A list of artist names that authored the album of the song
title	The title of the song
releaseDate	A string representing the release date of the song
albumArtists	The title of the album of the song
fileUri	`Uri` of the song
coverUri	`Uri` of the song cover art, in app storage

Extracts the tags and album covers to app storage of a list of `SongFile`

Parameters

fileList	List of `SongFile`, each element should have it's data extracted

Returns

List of `SongInfo` corresponding to each input song file

5.22 Color.kt File Reference

5.23 Theme.kt File Reference

5.24 Type.kt File Reference

5.25 AlbumsScreenViewModel.kt File Reference

Functions

- HiltViewModel (assistedFactory=AlbumsScreenViewModel.Factory::class) class AlbumsScreenViewModel @AssistedInject const ructor(@Assisted currentAuthor

  `ViewModel` used for the albums screen

5.25.1 Function Documentation

5.25.1.1 HiltViewModel()

```
HiltViewModel (
            assistedFactory  = AlbumsScreenViewModel::Factory::class) const
```

`ViewModel` used for the albums screen

Parameters

currentAuthor	Author whose albums to display

Flow of a pair of `Album` objects and bitmaps of their covers

5.26 AlbumTileViewModel.kt File Reference

5.27 LibraryViewModel.kt File Reference

Functions

- HiltViewModel class LibraryViewModel Inject constructor (private val picker:MusicFileLoader, private val databaseManager:DatabaseManager, private val tagExtractor:TagExtractor,)

  Viewmodel presenting the starting, author screen.

5.27.1 Function Documentation

5.27.1.1 constructor()

```
HiltViewModel class LibraryViewModel Inject constructor (
            private val picker:MusicFileLoader ,
            private val databaseManager:DatabaseManager ,
            private val tagExtractor:TagExtractor )
```

Viewmodel presenting the starting, author screen.

Also handles user interaction with the file picker All authors in the database. Used for displaying author tiles

Gets a list of songs in a given album

Parameters

album↩ Id	Id of the album

Returns

Flow of a list of `Song`

Initialize the flow used by the file picker

See also

MusicFileLoader.PrepareFilePicker

Launch the file picker

5.28 PlayerViewModel.kt File Reference

Functions

- HiltViewModel class PlayerViewModel Inject constructor (private val audioPlayer:AudioPlayer, private val databaseManager:DatabaseManager, private val imageManager:ImageManager, @ApplicationContext private val context:Context, savedStateHandle:SavedStateHandle)

  Viewmodel for the player screen.

5.28.1 Function Documentation

5.28.1.1 constructor()

```
HiltViewModel class PlayerViewModel Inject constructor (
            private val audioPlayer:AudioPlayer ,
            private val databaseManager:DatabaseManager ,
            private val imageManager:ImageManager ,
             @ApplicationContext private val context:Context ,
            savedStateHandle:SavedStateHandle )
```

Viewmodel for the player screen.

Flow that represents the position of the playback progress bar

Album icon of the currently playing song

Flow with the title of the currently playing song

Flow with the artists of the current song joined into a string

Flow with the `Album` of the current song

Flow with an `ImageBitmap` of the cover of the current song

Observable list of points corresponding to the height of each point on the waveform

This property handles toast sent by the viewmodel in case of an exception or user error

Handle the user tapping on the progress bar and change the playback position

Handle the user tapping the center button. Depending on the playback state the behaviour will be different

If the playback is playing, the button will pause If the playback is paused, the playback will resume If the playback is stopped, the playback will restart

Initialize the `currentSong` Flow

5.29 WaveFormExtractor.kt File Reference

Functions

- suspend fun extractWaveformDataFromUri (context:Context, uri:Uri)

5.29.1 Function Documentation

5.29.1.1 extractWaveformDataFromUri()

```
suspend fun extractWaveformDataFromUri (
             context:Context ,
             uri:Uri )
```

5.30 ExampleUnitTest.kt File Reference

Classes

- class ExampleUnitTest

  Example local unit test, which will execute on the development machine (host).

Index