

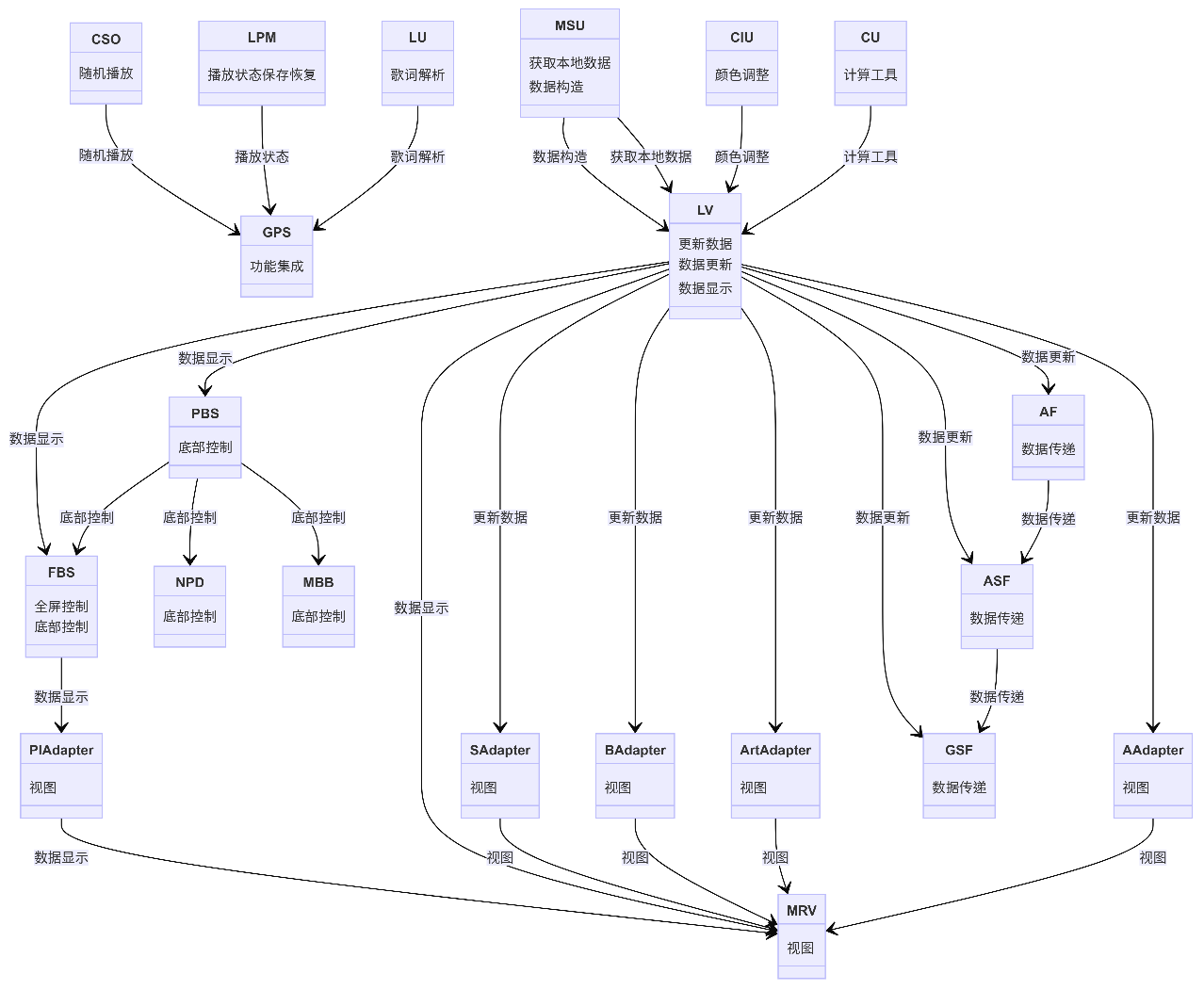
图1 架构图与模块分工

图2 文件调用协作图

图中文件名称的对应表如下：

表1 文件名功能对照表

|  |  |
| --- | --- |
| 文件名 | 功能说明 |
| AAdapter (AlbumAdapter) | 专用于显示专辑列表的适配器 |
| ArtAdapter (ArtistAdapter) | 用于显示艺术家的适配器 |
| BAdapter (BaseAdapter) | 通用的RecyclerView适配器，用于展示数据列表 |
| SAdapter (SongAdapter) | 用于显示歌曲列表的适配器 |
| PlAdapter（PlaylistAdapter） | 用于显示歌单以及播放队列适配器 |
| MRV (MyRecyclerView) | 自定义的RecyclerView，主要提供与AppBarLayout交互还有快速滚动 |
| FBS (FullBottomSheet) | 自定义控件，底部弹出窗口，包含各个组件的点击事件和控制逻辑 |
| PBS (PlayerBottomSheet) | 自定义FrameLayout，显示音乐播放器底部弹出框（抽屉式面板） |
| NPD (NowPlayingDrawable) | 自定义动画实现三根柱子随音乐变化的效果 |
| MBB (MyBottomSheetBehavior) | 管理底部弹出框（Bottom Sheet）的行为，提供多种选项 |
| LU (LrcUtils) | 从音频文件或特定的.lrc歌词文件中提取和解析歌词信息 |
| GPS (GramophonePlaybackService) | 集成了ExoPlayer，用于媒体播放、管理状态、自定义播放命令等等 |
| LPM (LastPlayedManager) | 管理音乐播放器中的播放状态和播放列表 |
| CU (CalculationUtils) | 执行各种内部计算和数据转换 |
| ClU (ColorUtils) | 根据设备的当前模式调整颜色的HSL（色调、饱和度、亮度）值 |
| CSO (CircularShuffleOrder) | 用于实现自定义的循环随机播放顺序 |
| AF (AdapterFragment) | 通用的容器，用于显示RecyclerView的各种列表 |
| GSF (GeneralSubFragment) | 用于显示不同类型（专辑、流派等）的项目列表 |
| LV (LibraryViewModel) | 囊括所有媒体库数据的ViewModel类，用于管理应用程序中的媒体库数据 |
| MSU（MediaStoreUtils） | 获取设备上的本地数据，并构造应用所需的各种列表和媒体项 |
| CSO (CircularShuffleOrder) | 随机播放实现 |

**应用整体逻辑分析**：

为实现该项目，我们采用了MVVM（Model-View-ViewModel）框架和分层架构设计模式，将数据管理、界面展示和播放控制逻辑进行模块化分离，使用kotlin语言开发。利用Media3库以及Media3.session中的MediaController实现音乐播放以及相关管理。

以下是该项目的整体架构说明：

MainAcitvity，该活动作为应用的入口以及核心，协调和组件的初始化和交互。在MainActivity中，LibraryViewModel负责数据的管理；MediaControllerViewModel负责播放相关的服务注册（GramophonePlaybackService及其使用的LastPlayedManager）以及本地歌曲数据的获取（借由MediaStoreUtils）；多片段组成的视图以及底部FullBottomSheet负责数据展示以及用户交互。

数据上：主要由MediaStoreUtils实现对本地歌曲的扫描和数据的构建。

播放控制上：使用Meida3库以及Media3.session中的MediaController实现播放及其相关控制（包括通知栏中的控制）。

界面上：包含一个容器和一个播放器窗口。容器主要负责按类别（带导航）以一定形式展示数据，由ViewPager2、自定义的recyclerView、以及各个适配器等相关组件实现；播放器窗口负责播放相关管的UI展示和交互，主要由PlayerBottomSheet和FullBottomSheet等组件实现。

下面是具体的架构和组件分析：

1. Model

数据和业务逻辑层，负责数据获取、存储和处理，主要包括：

(1) MediaStoreUtils：负责检索设备媒体库中的音频文件信息，构建媒体库的数据结构，并通过协程更新视图模型。

(2) LastPlayedManager：管理播放状态和播放列表的持久化，使用SharedPreferences存储和恢复数据。

(3) GramophonePlaybackService：集成了ExoPlayer用于媒体播放，管理播放状态和自定义播放命令。

(4) 各类比较器：如AlphaNumericComparator和SupportComparator，用于对歌曲、艺术家、专辑等进行排序。

2. View

用户界面层，负责显示数据并与用户交互，主要包括：

(1) MainActivity：主活动，负责启动应用、权限申请、读取文件和界面绘制。

(2) 各类Fragment（如AdapterFragment、ArtistSubFragment、SearchFragment等）：负责显示具体的数据列表或详细信息。

(3) 自定义视图组件：

MyRecyclerView、MyBottomSheetBehavior、PlayerBottomSheet、FullBottomSheet等：提供自定义的RecyclerView、底部弹出框和播放控制视图。

NowPlayingDrawable、MarqueeTextView等：提供动画效果和滚动文本显示。

3. ViewModel

视图模型层，充当Model和View之间的桥梁，主要包括：

(1) LibraryViewModel：管理媒体库的数据和状态。

(2) MediaControllerViewModel：管理媒体播放控制器的逻辑和生命周期。

(3) LifecycleCallbackList和LifecycleCallbackListImpl：管理具有生命周期感知能力的回调对象。

4. Utils

工具类和辅助函数，提供各种辅助功能，主要包括：

(1) CalculationUtils：执行时间格式转换、颜色设置、数值约束、插值等计算。

(2) ColorUtils：根据设备的当前模式调整颜色的HSL值。

(3) LrcUtils：解析音频文件的歌词信息。

(4) CircularShuffleOrder：实现自定义的循环随机播放顺序。

(5) ExoPlayer相关的工具类：如EndedWorkaroundPlayer、GramophoneExtractorsFactory、GramophoneMediaSourceFactory等，处理播放结束问题、创建媒体源和自定义渲染器。

5. Adapter

适配器层，用于RecyclerView和ViewPager2的数据绑定，主要包括：

(1) AlbumAdapter、ArtistAdapter、BaseAdapter、SongAdapter等：用于显示专辑、艺术家、歌曲等列表，并处理点击事件。

(2) ViewPager2Adapter：管理和显示多个标签页中的Fragment。

6. 关键组件和功能

(1) 播放控制：使用ExoPlayer库实现媒体播放，使用MediaController进行播放控制（播放、暂停、上一首、下一首、播放模式等）。

(2) 歌词解析：通过LrcUtils解析歌词文件，并在播放时同步显示。

(3) 随机播放：通过CircularShuffleOrder实现自定义的循环随机播放顺序。

综上所述，该项目结合了MVVM架构和自定义组件，通过ViewModel管理数据状态和逻辑，通过自定义的Adapter和View优化用户界面和交互体验，同时利用ExoPlayer库实现强大的媒体播放功能。