eLive

软件设计文档

项目	eLive
文档	软件设计文档
说明	V2.0
作者	张凯鑫
最后更新时间	2017-06-12

版本更新概要			
版本号	时间	更新人	更新摘要
V1.0	2017-05-9	张凯鑫	完成架构设计初稿
V2.0	2017-6-12	张凯鑫	修改设计图并定稿

目录

1. 引言

1.1 编写目的

编写本文档的目的是为了使开发人员了解 eLive 应用的软件设计，便于在软件需求规格说明书的基础上完成软件设计规定的各模块的具体实现工作。

1.2 读者对象

本文档的读者主要是应用开发人员，即 eLive 团队成员。

1.3 软件应用概述

1.3.1 软件名称

软件名称为“eLive（易生活）”。简称：eLive。

1.3.2 软件功能概述

从整体上说，功能主要有“三看两写”，即看新闻、看天气、看笑话，和写日记、写备忘。其中，新闻模块又根据新闻的热门情况划分为五个模块：头条、娱乐、军事、科技、财经。天气模块直接获取所在城市天气状况、生活指数及接下来五天的天气预报。笑话模块是娱乐功能，获取并显示笑话。日记模块和备忘模块是为个人提供的应用功能，用于记录个人事宜。

1.3.3 软件性能要求

硬件要求：此应用是 android 应用，只能使用安卓手机或安卓模拟器运行，且安卓必须为 4.4 及以上版本。

网络要求：新闻、天气、笑话模块均需求联网，如果没有网络，新闻、天气和笑话无法获取数据，日记中实现了定位和天气获取的功能，理论也需要联网，但无网状态下，应用采用默认的定位和天气数据。

权限要求：应用需要访问网络，所以需求网编访问权限；应用需求对日记等文件进行读写操作，所以需求文件读写权限；应用需求定位功能，所以需求要位权限。

1.4 文档概述

本文档主要说明 eLive 应用所选用的软件设计技术，并分析其整体架构、模块划分及接口设计、模块架构设计，并按此内容流程组织文档。

2. 软件设计约束

2.1 设计目标和原则

应用的设计应实现用户的需求，即实现需求规格说明书中所提及的全部功能。此外，应

用具有良好的可扩充性，便于之后功能的添加与实现。在此基础上，应用设计应尽可能得清晰明确。

为实现上述目标，在设计软件过程中将会遵循软件设计的以下原则：

- 开闭原则：对组件功能的扩展是开放的，对原有代码的修改是封闭的。
- 依赖倒置原则：依赖于抽象，不依赖于实现，针对接口编程，不针对实现编程。

2.2 设计约束

- 硬件平台：android 手机或模拟器
- 开发语言：Java
- 开发工具：Android Studio 2.3

3. 软件设计描述

3.1 总体结构设计与模块划分

本应用分模块设计，每一个大功能为一个模块，根据应用的功能可划分如下图所示，各模块的架构设计分别设计。

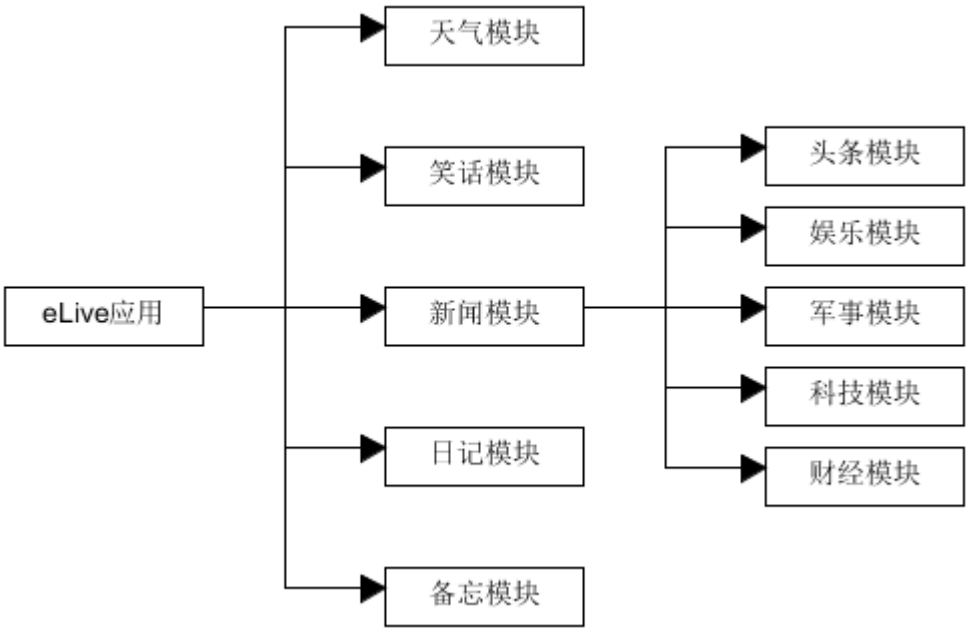


图 1 eLive 应用模块划分图

3.2 设计类图

经过详细的分析讨论，本应用涉及到的 Activity（包含 Fragment）主要有以下几个：MainActivity、NewsFragment、NewsSubFragment、NewsDetailsActivity、JokesFragment、WeatherFragment 、 DiaryFragment 、 SetDiaryLock 、 DiaryActivity 、 MarkerFragment 、MarkerActivity、MarkerDetialsActivity，Activity 的设计类图如下：

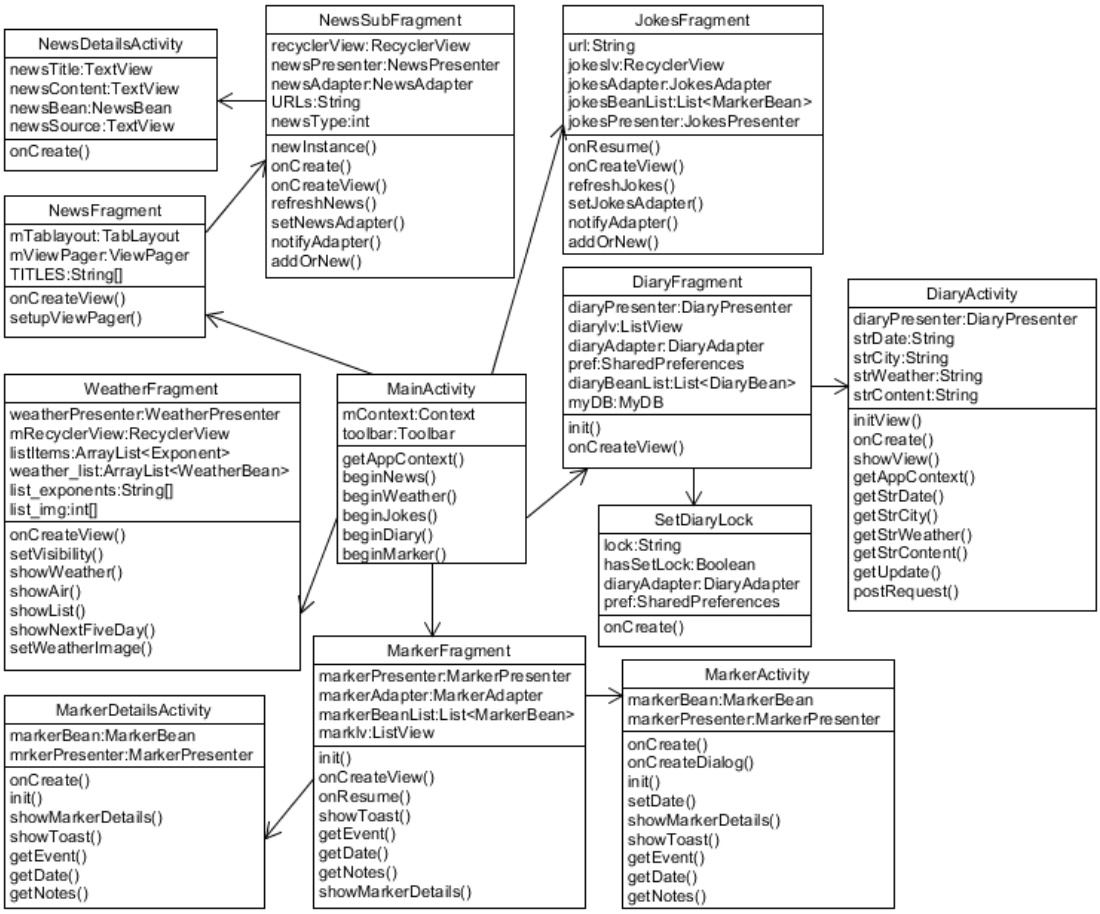


图 2 eLive Activity 设计类图

3.3 子系统及其接口设计

eLive 应用采用安卓抽屉式布局，拥有侧滑抽屉，侧滑出现功能列表，可以跳转到对应的模块系统中，具体接口设计如下图所示：

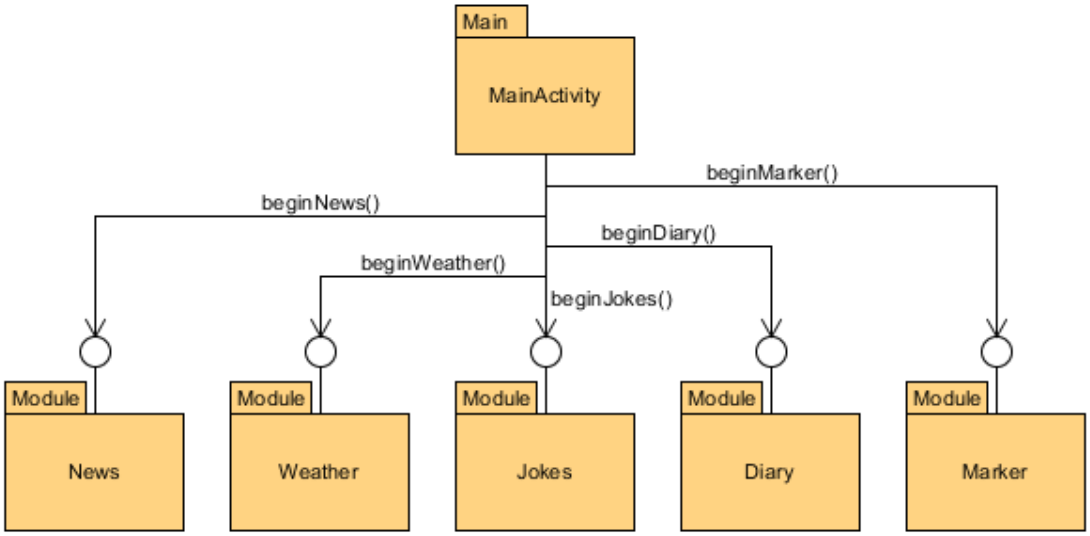
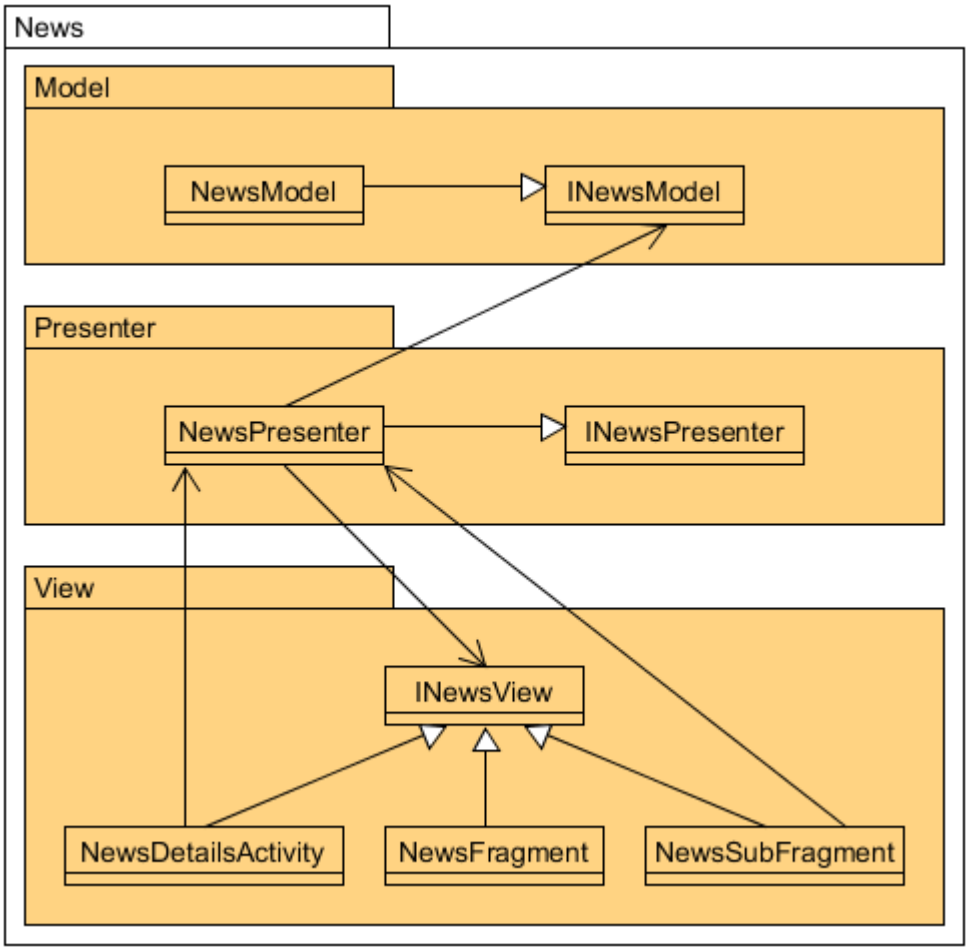


图 3 eLive 接口设计

3.3.1 新闻模块子系统设计

新闻模块遵循 MVP 架构，分别为数据层（Model）、视图层（View）、发布层（Presenter），按新闻的热门情况将新闻划分为五个子模块，这五个子模块均遵循 MVP 架构，因此此处不区分不同模块的设计，具体如下图所示。



1） 视图层（View）

该层对应于 News 模块中与 News 相关的 NewsDetailsActivity 和 NewsFragment、NewsSubFragment，负责显示新闻数据（model）并且将用户对这一模块的指令（events）传送到 presenter 层以便作用于那些数据的接口，它含有 Presenter 的引用，由获取新闻列表、上拉刷新列表、下拉加载更多、显示新闻详情的界面功能组成，并提供对应的操作方法。

2） 数据层（Model）

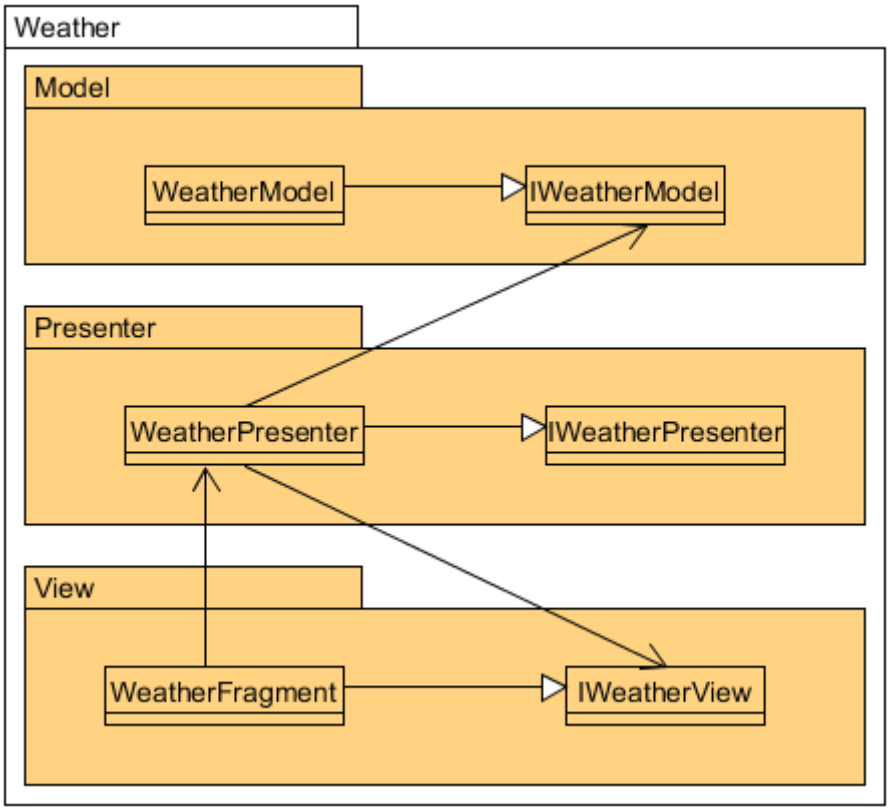
该层负责对新闻数据的存取操作，包含新闻 API 网址的获取、新闻 API 网络连接、新闻数据请求、数据传输。

3） 发布层（Presenter）

新闻模块中的 Presenter 层是连接 View 层和 Model 层桥梁，负责对新闻业务逻辑进行处理，它将从 Model 层获得的新闻数据，进行一些适当的处理后交由 View 层进行显示，使得 View 和 Model 之间不存在耦合，同时也将新闻业务逻辑从 View 中抽离。

3.3.2 天气子系统设计

天气模块遵循 MVP 架构，分别为数据层（Model）、视图层（View）、发布层（Presenter），设计如下图所示。



1) 视图层（View）

该层对应于天气模块中与 Weather 相关的 WeatherFragment，负责显示天气数据（model）并且将用户对这一模块的指令（events）传送到 presenter 层以便作用于那些数据的接口，它含有 Presenter 的引用，由显示当天天气状况、显示接下来五天天气状况、显示当天各类生活指数等界面功能组成。

2) 数据层（Model）

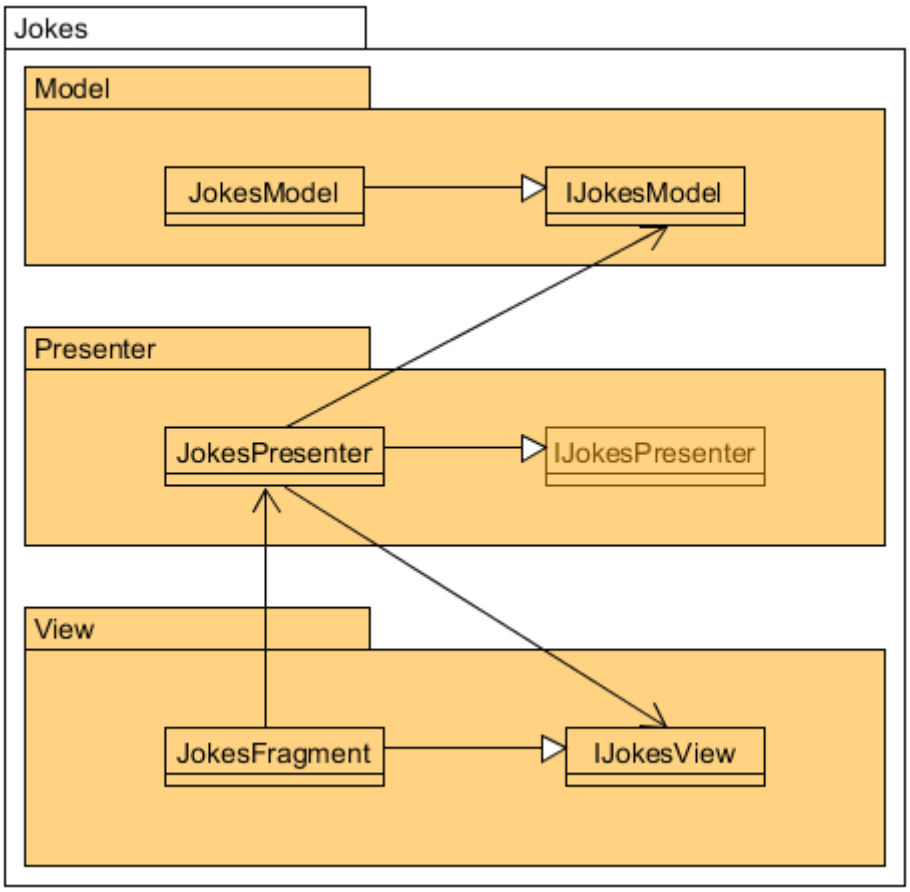
该层负责对天气数据的存取操作，包含天气 API 网址的获取、天气 API 网络连接、天气数据请求、数据传输。

3) 发布层（Presenter）

天气模块中的 Presenter 层是连接 View 层和 Model 层桥梁，负责对天气业务逻辑进行处理，它将从 Model 层获得的天气数据，进行一些适当的分情况处理后交由 View 层进行显示，使得 View 和 Model 之间不存在耦合，同时也将天气业务逻辑从 View 中抽离。

3.3.3 笑话子系统设计

笑话模块遵循 MVP 架构，分别为数据层（Model）、视图层（View）、发布层（Presenter），设计如下图所示：



1) 视图层（View）

该层对应于笑话模块中与 Jokes 相关的 JokesFragment，负责显示笑话数据（model）并且将用户对这一模块的指令（events）传送到 presenter 层以便作用于那些数据的接口，它含有 Presenter 的引用，由获取笑话、上拉刷新列表、下拉加载更多等界面功能组成。

2) 数据层（Model）

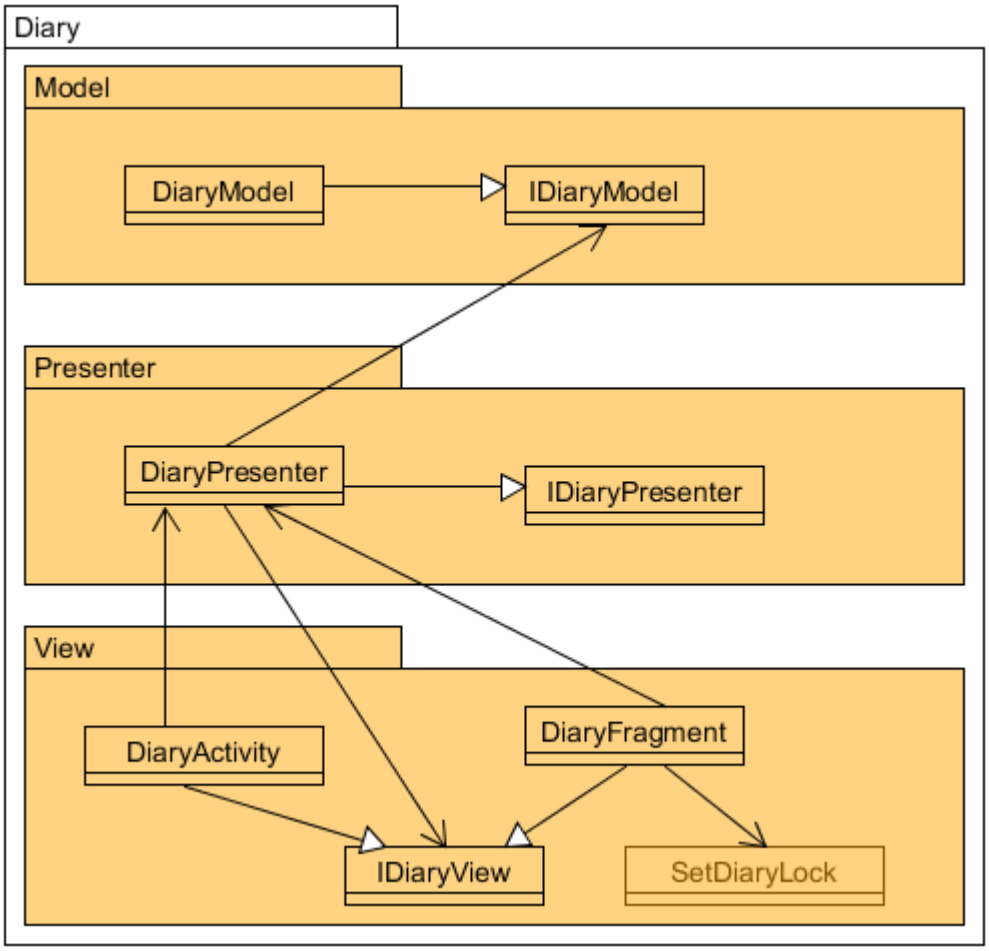
该层负责对笑话数据的存取操作，包含笑话 API 网址的获取、笑话 API 网络连接、天气数据请求、数据传输。

3) 发布层（Presenter）

笑话模块中的 Presenter 层是连接 View 层和 Model 层桥梁，负责对笑话业务逻辑进行处理，它将从 Model 层获得的笑话数据，进行一些适当的处理后交由 View 层进行显示，使得 View 和 Model 之间不存在耦合，同时也将笑话业务逻辑从 View 中抽离。

3.3.4 日记子系统设计

日记模块遵循 MVP 架构，分别为数据层（Model）、视图层（View）、发布层（Presenter），设计如下图所示：



1) 视图层（View）

该层对应于日记模块中与 Diary 相关的 Fragment 和 Activity，具体有 DiaryFragment、DiaryActivity、SetDiaryLock，负责显示日记数据（model）并且将用户对这一模块的指令（events）传送到 presenter 层以便作用于那些数据的接口，它含有 Presenter 的引用，由设置日记锁、修改日记锁密码、获取日记列表、添加日记、查看日记、修改日记、删除日记等界面功能组成。

2) 数据层（Model）

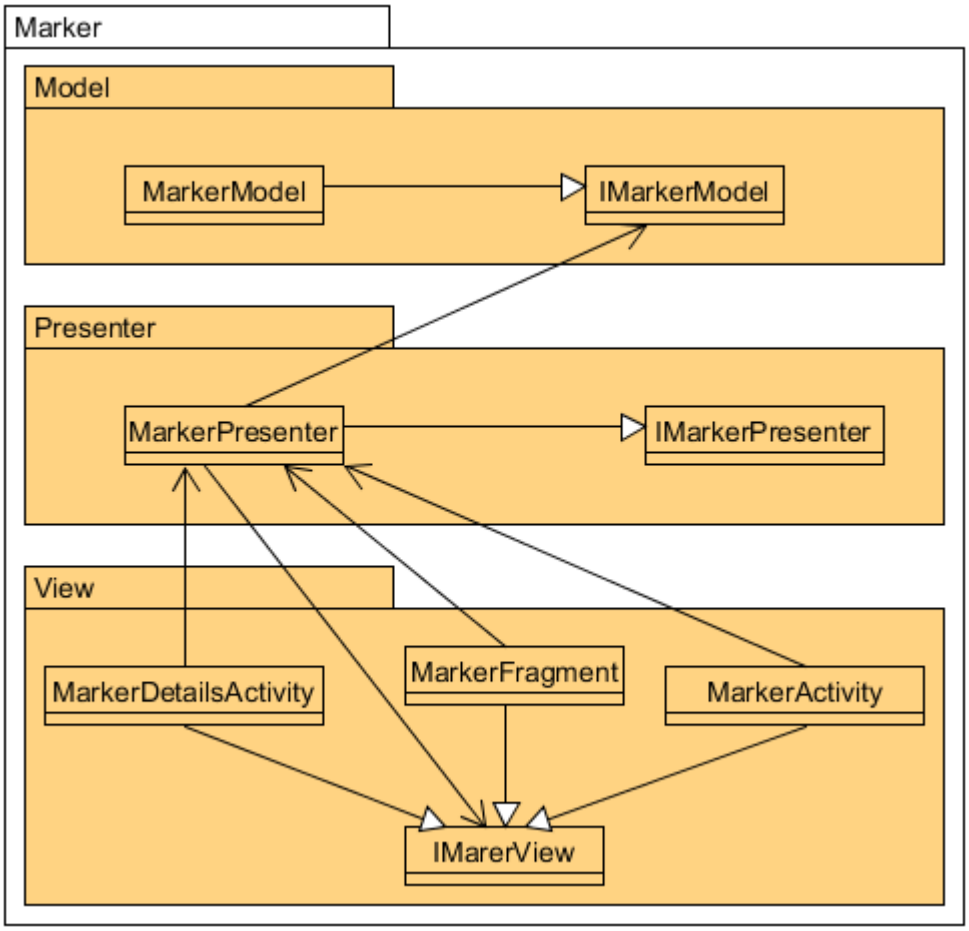
该层负责对日记数据的存取操作，包含数据库的连接、数据库中日记表的“增、删、查、改”操作、对应日记文件的存取、删除等。

3) 发布层（Presenter）

日记模块中的 Presenter 层是连接 View 层和 Model 层桥梁，负责对日记业务逻辑进行处理，它将从 Model 层获得的日记数据，进行一些适当的处理后交由 View 层进行显示，使得 View 和 Model 之间不存在耦合，同时也将日记业务逻辑从 View 中抽离。

3.3.5 备忘子系统设计

备忘模块遵循 MVP 架构，分别为数据层（Model）、视图层（View）、发布层（Presenter），设计如下图所示：



1) 视图层（View）

该层对应于备忘模块中与 Marker 相关的 Fragment 和 Activity，具体有 MarkerFragment、MarkerActivity、MarkerDetailsActivity，负责显示备忘的数据（model）并且将用户对这一模块的指令（events）传送到 presenter 层以便作用于那些数据的接口，它含有 Presenter 的引用，由获取备忘列表、添加备忘、查看备忘、修改备忘、删除备忘等界面功能组成。

2) 数据层（Model）

该层负责对备忘数据的存取操作，包含数据库的连接、数据库中备忘表的“增、删、查、改”操作等。

3) 发布层（Presenter）

备忘模块中的 Presenter 层是连接 View 层和 Model 层桥梁，负责对备忘业务逻辑进行处理，它将从 Model 层获得的日记数据，进行一些适当的处理后交由 View 层进行显示，使得 View 和 Model 之间不存在耦合，同时也将备忘业务逻辑从 View 中抽离。

3.4 设计技术选型理由

3.4.1 面向对象设计

eLive 软件设计选用面向对象设计，此设计技术主要体现在各模块模型的设计中。每个功能都有一个 Bean 类，分别有 NewsBean、WeatherBean、JokesBean、DiaryBean、MarkBean，这些类的属性和方法是对各自模块中的模型的抽象设计。面向对象设计的基本思想是使用类、对象、继承、封闭、消息等基本概念进行程序设计，它符合人类的自然思维方式，并根

据事物的本质特征将它们抽象表示为应用中的类，这使得 eLive 软件中的组件功能更为直观，利于开发。此外，面向对象中类的封闭提高了类的内聚性，降低了对象之间的耦合性。

3.4.2 MVP 模式

eLive 软件设计中，应用的每一个模块均采用 MVP 架构设计模式。MVP 是 MVC 架构的一种演变，分别表示数据层（Model）、视图层（View）、发布层（Presenter）。它将模型与视图完全分离，可以只修改视图而不影响模型，在 MVP 中，View 不直接使用 Model，它们之间的通信是通过 Presenter 来进行的，所有的交互都发生在 Presenter 内部。一个 Presenter 可以用于多个视图，因为视图的变化总是比模型的变化频繁。

此外，MVP 对于 android 应用开发有着很好的优势，它能实现模块分工，这能大大简化开发人员对代码的理解难度，遵循单一职责原则，将复杂的业务逻辑分解。它具有良好的易用性，而且维护成本低。