# Software architecture document for project Lima App

## 0. Авторы

- Вольф Илья
- Чиесов Игорь
- Вайс Андрей
- Романов Никита

## 1. Цели и ограничения

1.1. Основные функциональные требования

1) Возможность добавлять / удалять проект
2) Возможность задавать цвет и название проекта
3) Возможность сделать задачу выполненной
4) Возможность скрывать выполненные задачи
5) Возможность удалить все выполненные задачи
6) Возможность совершать поиск по имени задачи
7) Приложение должно отображать задачи по приоритетности
8) Возможность добавлять / удалять / редактировать задачу
9) Возможность отменять действие удаления задачи
10) Возможность задать название/описание/приоритетность/категорию/время начала/время конца/день задачи
11) Возможность видеть задачи только на сегодня в виде списка и виде календаря с отображением в зависимости от времени
12) Возможность видеть задачи только на последующие дни в виде списка
13) Реализация иерархии задач: одна задача является подзадачей другой задачи

1.2. Основные не функциональные требования

1) Приложение должно быстро реагировать на действия пользователей.
2) Приложение должно поддерживаться как можно большим количеством устройств Android

1.3. Архитектурные цели

1) Возможность легко расширять список атрибутов задач / категорий.
2) Возможность видеть изменения в задачах / категориях в реальном времени (то есть мы что-то изменили и это сразу отрисовалось)
3) Реализовать паттерн seperation of concerns и dependency injection
4) Реализовать паттерн one activity with multiple fragments
5) Реализовать архитектуру MVVM
6) Реализовать сохранение данных в локальной базе данных SQL
7) Реализовать сохранения состояния при убийстве приложения или ухода в background.

1.4. Дополнительные цели, ограничения и предпочтения

1) Тенденция к высокой совместимости текущей Android-реализации и возможной iOS-реализации в будущем.
2) Основной язык программирования - Kotlin. Освоить Kotlin Flows & Coroutines

## 2. Описание решения

В соответствии с нашими целями, мы получили следующее описание решения:

## 2.1. Модули и подсистемы

UI модуль.
Отвечает за отрисовку всех данных. Должен содержать в себе только логику, которая зависит от элементов интерфейса (Вьюшек). Все данные для отрисовки получает из ViewModel посредством паттерна наблюдателя. То есть мы подписываемся на какую-то переменную типа LiveData из ViewModel и как бы “слушаем её изменения”. Как только данные изменились мы их перерисовываем.

ViewModel
Отвечает за подготовку и получения всех данных, относящихся к определенному фрагменту, а также за сохранения данных при убийстве приложения или перехода в background. Здесь у нас происходят вызовы функций dao для получения данных из бд. Также данный модель содержит в себя модификацию данных. Все вызовы к бд происходят многопоточно, тем самым, не блокируя UI thread.

DataBase
Отвечает за хранение данных в локальной базе данных. Представляет интерфейс методов получения данных по запросам из бд.

Hilt
Отвечает за реализацию паттерна dependency Injection