Clean Achitecture — принцип разработки приложений, предложенный Uncle Bob'ом. Код, спроектированный с учётом этой архитектуры, легче тестировать и переиспользовать.

Преимуществами данной архитектуры являются:

Простота написания тестов;

Независимость от фреймворков;

Независимость от UI;

Независимость от Баз Данных;

Независимость от внешних сервисов, с которыми взаимодействует приложение.

Суть Clean Architecture заключается в разделении логики приложения на несколько составляющих слоёв: слой бизнес-логики, слой представления и слой данных.

При этом чтобы обеспечить максимальную независимость этих слоев, на каждом из них используется своя модель данных, которая конвертируется при взаимодействии между слоями.Так же выделяются отдельные интерфейсы для взаимодействия между слоями.

Схема данных слоев выглядит следующим образом:

Слой представления предназначен в первую очередь для взаимодействия с пользователем, так же он отвечает за логику отображения данных на экране и за другие процессы, связанные с UI. Этот слой не должен содержать логику приложения, не связанную с UI. Именно слой представления привязывается к экранам и помогает организовать взаимодействие со слоем бизнес-логики и работу с данными. Этот слой может быть реализован с использованием любого предпочитаемого паттерна, к примеру, MVC, MVP, MVVM и других.

При реализации данного приложения слой представления организуем согласно паттерну MVP. Он позволит нам разделить экран на UI-часть (View), на логику работы с UI (Presenter) и объекты для взаимодействия с UI (Model).

Слое бизнес-логики содержит всю бизнес-логика приложения. Этот слой является неким объединением слоев сценариев взаимодействия и бизнес-логики. Именно к этому слою обращается слои представления для выполнения запросов и получения данных. В данном приложении слой бизнес-логики будет реализован в виде Java-модуля, который не содержит никаких зависимостей от Android-классов. Преимуществом данного подхода является то, что для реализации бизнес-логики нам нужны только классы моделей и стандартные средства языка Java. Более того, такой подход позволит легко тестировать этот слой с помощью обычных тестов на JUnit, что очень удобно. В таком случае иногда не будет возможности выполнить какой-либо метод или использовать некоторые классы из других слоев. Поэтому для взаимодействия с этим слоем используются интерфейсы.

Слой данных отвечает в первую очередь за получение данных из различных источников и их кэширование. Он реализуется за счет паттерна Repository, и его общую схему можно представить следующим образом:

Существует несколько плюсов от использования такого подхода. Во-первых, другие слои, которые запрашивают данные, не знают о том, откуда эти данные приходят. Более того, им не нужно этого знать, так как это усложняет логику работы и модуль берет на себя лишнюю ответственность. Во-вторых, слой данных в таком случае выступает единственным источником информации.