УДК 004.273

АНАЛИЗ СОВРЕМЕННОЙ АРХИТЕКТУРЫ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ОПЕРАЦИОННОЙ СИСТЕМЫ АНДРОИД

Е.Н. ГОРОВОЙ
(*Представлено: М.В. ДЕКАНОВА*)

В данной статье рассматривается архитектура и общие подходы к реализации мобильного приложения для операционной системы Андроид, выделяются основные слои и уровни архитектуры, необходимые для создания качественного проекта.

Введение. Процесс разработки приложения представляет собой написание огромного количества кода, создания новых экранов и модулей, зачастую сопровождается изменением изначальных требований, добавлением новых разработчиков к работе. Правильный выбор базовой архитектуры приложения позволит создать легко масштабируемый и модифицируемый проект, позволит уменьшить время, требующееся новым разработчикам, чтобы присоединиться к работе над проектом, обеспечит организованность и чистоту кода.

Подход, позволяющий реализовать хорошо организованный и легко модифицируемый проект, должен отвечать следующим требованиям:

- независимость от способа предоставления данных;
- независимость от пользовательского интерфейса;
- независимость от используемых библиотек;
- тестируемость.

Независимость от способа представления данных. Данные могут предоставляться через веб-сервис, или храниться в локальной базе данных. Бизнес-логика не зависит от способа предоставления данных.

Независимость от пользовательского интерфейса. Пользовательский интерфейс может быть изменен без изменения остальной части системы, также возможно создать несколько реализаций интерфейса, например, для мобильной версии, и версии для Android TV.

Независимость от используемых библиотек. Архитектура приложения не должна зависеть от наличия каких-либо библиотек, это позволит использовать их в качестве инструмента, а не встраивать систему в ограничения, создаваемые ими.

Тестируемость. Бизнес-логика может быть протестирована без использования пользовательского интерфейса, базы данных, веб-сервера или любого другого внешнего элемента [2].

Исходя из данных требований можно выделить четыре основных слоя приложения:

- Entities. Слой данных;
- Use Cases (Interactors). Бизнес-логика приложения;
- Interface Adapters. Адаптеры между бизнес-логикой и пользовательским интерфейсом, а также между бизнес-логикой и внешними данными.
- Frameworks. Внешний слой, в нем находится все остальное: пользовательский интерфейс, база данных, http-клиент и другое.

Расположив данные слои приложения на уровни можно построить схему процесса обработки данных в приложении (рисунок). Событие пользователя идет в Presenter, тот передает в Use Case. Use Case делает запрос в Repository. Repository получает данные из источника данных, создает Entity, передает его в UseCase. Так Use Case получает все нужные ему Entity. Затем, применив их и свою логику, получает результат, который передает обратно в Presenter. А тот, в свою очередь, отображает результат в пользовательском интерфейсе [1].

Рассмотрим уровни приложения подробнее.

Data. Все данные, необходимые для приложения, поставляются из этого слоя через реализацию Repository (интерфейс находится в domain — слое бизнес-логики), который использует Repository Pattern со стратегией, которая, в свою очередь, выбирает нужный источник данных, например, веб-сервис или базу данных [3].

Например, для получения конкретного пользователя по ключу источником данных выбирается база данных, если пользователь уже загружен в неё, в противном случае отправляется запрос на получение данных в веб-сервис, и сохраняется в базу данных.

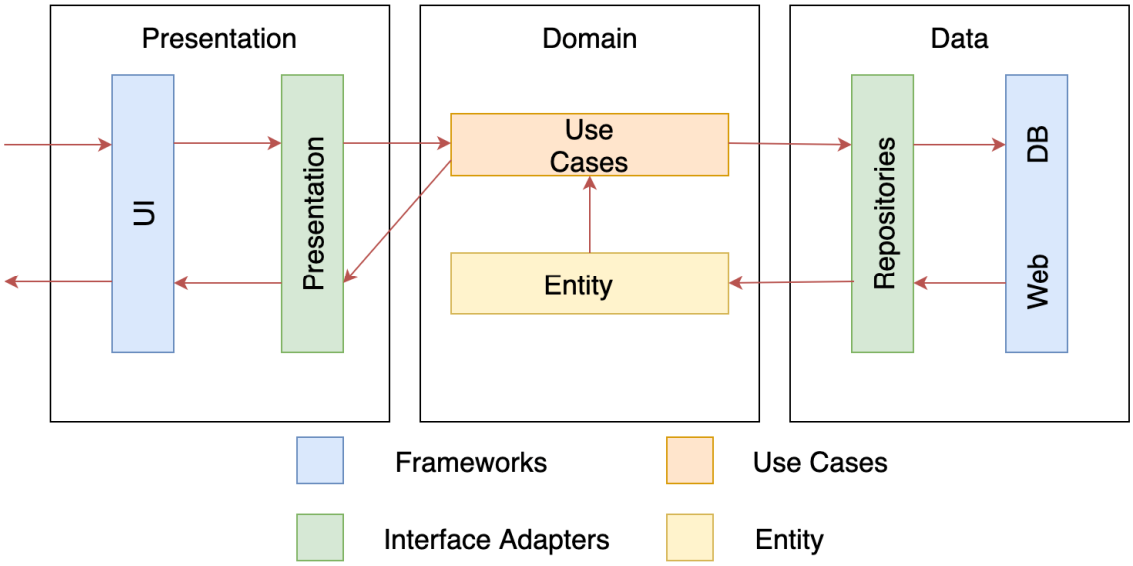

Рисунок. – Схема процесса обработки данных в приложении

Главная идея всего заключается в том, что происхождение данных является понятным для клиента, которого не волнует, поступают данные из памяти, базы данных или веб-сервиса, ему важно только то, что данные будут получены и доступны.

Domain. На данном уровне происходит вся бизнес-логика приложения и преобразование данных в данные, подходящие для отображения, если это требуется. Все внешние компоненты используют интерфейсы для связи с бизнес-объектами.

Presentation. Presenters на данном уровне связываются с Use Cases для получения данных для отображения и дальнейшей их передачи на пользовательский интерфейс. Так же на данном уровне происходят все анимации и отображение данных.

Стоит заметить, что существует множество паттернов, для реализации уровня представления данных, наиболее популярные из них – Model-View-Presenter (MVP), Model-View-ViewModel (MVVM), а также Model-View-Controller (MVC).

Заключение. Архитектура приложения является одной из самых важных вещей в разработке программы. Изменение архитектуры – это сложная и длительная задача с использованием уже существующего кода, которая может привести к появлению многих новых ошибок в работе приложения. Поэтому нужно быть осторожным при выборе архитектуры еще на этапе разработки приложения.

В этой статье рассмотрен один из подходов к разработке мобильных приложений, а именно подход «Clean Architecture», определяющий слои, разделяющие уровни мобильного приложения, что, в свою очередь, улучшает тестируемость кода. Модули становятся независимыми друг от друга, что дает возможность их самостоятельной разработки и повторного использования, а также упрощает внедрение новых функций в проект.

ЛИТЕРАТУРА

1. A Guided Tour inside a clean architecture code base. [Электронный ресурс]. Режим доступа: https://proandroiddev.com/a-guided-tour-inside-a-clean-architecture-code-base-48bb5cc9fc97. – Дата доступа: 15.09.2019.
2. Basic concepts of software architecture patterns in Android. [Электронный ресурс]. Режим доступа: https://android.jlelse.eu/basic-concepts-of-software-architecture-patterns-in-android-c76e53f46cba. – Дата доступа: 16.09.2019.
3. Clean architecture for Android with Kotlin: a pragmatic approach for starters. [Электронный ресурс]. Режим доступа: https://antonioleiva.com/clean-architecture-android/. – Дата доступа: 16.09.2019.