УДК 004.02

Студ. А.В. Рубан
Науч. рук. Е.А. Блинова
(кафедра информационных систем и технологий, БГТУ)

РЕАЛИЗАЦИЯ ПАТТЕРНА ПРОЕКИРОВАНИЯ MVP ДЛЯ ANDROID ПРИЛОЖЕНИЯ “WOD(WORKOUT OF DAY)”

Архитектурный шаблон — это единственный способ сохранить проект чистым, расширяемым и проверяемым. Шаблоны являются признанными решениями, которые были разработаны на протяжении многих лет и считаются отраслевыми стандартами.

Когда мы кратко анализируем Android SDK, в частности, отношения между данными, у нас создается впечатление, что модель, которая лучше всего подходит Android, является Model View Controller (MVC). Однако, когда проект приобретает сложность, разделение проблем, предлагаемых MVC, недостаточно, особенно для реализации модульных тестов. Должен ли я использовать MVC или MVP в моем проекте? На этот вопрос нет правильного ответа. Это означает, что единственный способ ответить на вопрос - понять плюсы и минусы каждого решения. Model-view-controller (MVC) — это архитектурный образец программного обеспечения, в основном (но не исключительно) для реализации пользовательских интерфейсов на компьютерах. Он делит данное программное приложение на три взаимосвязанные части, чтобы отделить внутренние представления информации от способов представления или принятия информации от пользователя.

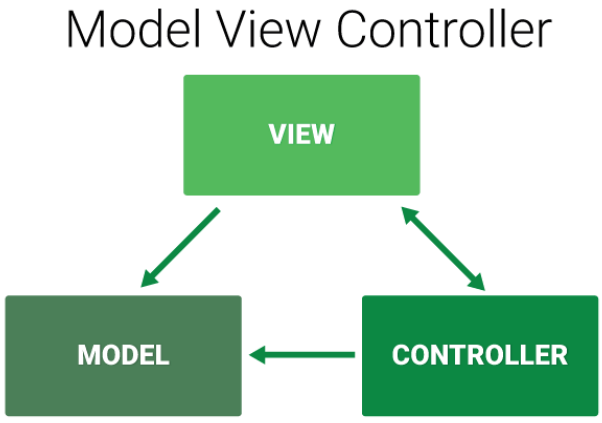

Рисунок 1 - Схема паттерна MVC

Model-view-presenter (MVP) — это шаблон проектирования, производный от MVC, который используется в основном для построения пользовательского интерфейса.

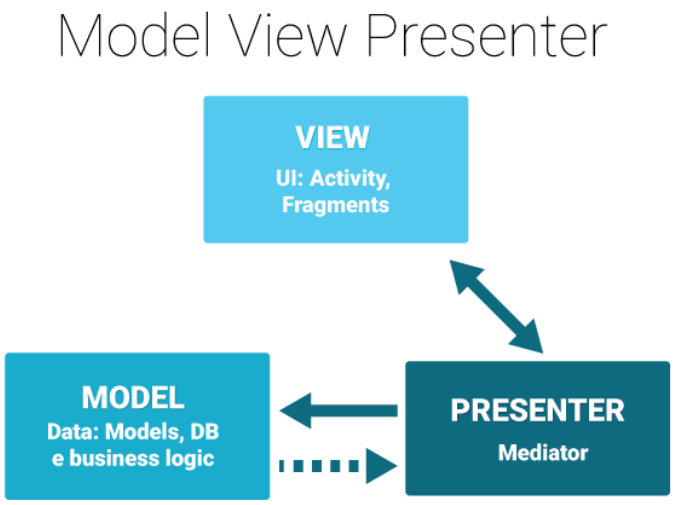

Рисунок 2 - Схема паттерна MVP

Для того чтобы приложение стало расширяемым, поддерживаемым и проверяемым, чрезвычайно важно создать глубокое разделение проблем, и это, вероятно, самое большое преимущество, которое мы получим при принятии MVP.

Таблица 1 - Отличия шаблонов проектирования MVC MVP

Отличия MVP от MVC	Отличия MVC от MVP
Для каждого представления есть свой Presenter.	Предствление знает о данных и может на прямую обращяться к ним
Представление ничего не знает о данных.	Контроллер может взаимодействовать с несколькими представлениями
Unit – тесты пишутся проще.	

Способ реализации шаблона изменяется в зависимости от роли, которую принимает Presenter. Но независимо от выбора, обязательно должны соблюдаться 3 правила:

1. Отделить код View, который мы показываем пользователю от бизнес- логики, то есть Presenter'а. Данные, которые отображаются во View должны быть представлены модулем Model и тогда код становится понятным, читабельным, поддерживаемым.
2. Связь между View и Model должна происходить через Presenter. Ни Model, ни View не должны иметь ссылок друг на друга.
3. Используя MVP, мы должны иметь возможность вносить изменения в один компонент, не затрагивая код в двух других. Так же, если мы хотим полностью заменить View или Model это должно быть возможно, например, мы используем контент провайдер для получения списка упражнений, который в ближайшее время должен быть заменен на ORMRealm без изменений в любой другой части проекта.

- **Presenter**выступает в роли посредника. Он извлекает данные из**Model**и показывает их в**View**. Он также обрабатывает действия пользователя, переданные ему из**View**.
- **View**это интерфейс, который отображает данные и направляет действия пользователя в**Presenter**.
- **Model**содержит бизнес-логику приложения. Он контролирует создание, сохранение и изменение данных.

ЛИТЕРАТУРА

1. Annotation Basic of MVP – The Android Way. [Электронный ресурс] / Medium Corporation. – 2018. / Режим доступа: https://hackernoon.com/basics-of-mvp-the-android-way-f75da407019d: 31.03.2018

УДК 004.02

Студ. А.А. Подолянчик
Науч. рук. Е.А. Блинова
(кафедра информационных систем и технологий, БГТУ)

РЕАЛИЗАЦИЯ ФУНКЦИОНАЛА ГЕОПОЗИЦИОНИРОВАНИЯ ПРИ ПОМОЩИ ФРЕЙМВОРКА MAPKIT ДЛЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ ПОД УПРАВЛЕНИЕМ ОС IOS

Актуальность сервисов геопозиционирования на сегодняшний день очевидна: большинство мобильных приложений используют данные сервисы для определения местоположения мобильного устройства, с целью предоставления различного спектра услуг: check-in логика, отслеживание движения транспорта, навигация, обеспечение пользователя актуальной информацией. Таким образом, любое современное мобильное приложение использует тот или иной функционал, предоставляемый данными сервисами. И если для операционной системы Android существует огромное количество примеров использования различных фреймворков для реализации данного функционала, то для устройств IOS-смейств, набор документации ограничен и в большинстве случаев написан на иностранных языках.

Наиболее распространённым фреймворком, позволяющим решать проблему отображения карт, является библиотека MapKit, предоставля APIs для разработчиков, что дает возможность работать с картами: отображать карты, перемещаться по карте, добавлять аннотации для определенныхмест, добавлять пометки на существующих картах и т.д. При разработке приложения вы также можете предоста-