ПРЕИМУЩЕСТВА И НЕДОСТАТКИ АРХИТЕКТУРЫ MVP ПРИ РАЗРАБОТКЕ ANDROID-ПРИЛОЖЕНИЯ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Макаревич В. А.

Тонкович И. Н. – канд. хим. наук, доцент

Актуализируется задача отделения бизнес-логики от логики отображения при разработке Android-приложения. Предложено использовать паттерн MVP, обеспечивающий полную независимость Presenter от View, что позволяет учесть особенность Android, а именно сохранение Presenter между жизненными циклами View.

Целью исследования является создание клиент-серверного Android-приложения для проверки знаний правил дорожного движения [1].

Разработка архитектуры является одним из самых важных этапов создания программного средства. При разработке Android-приложения была поставлена задача максимально отделить бизнес-логику от логики отображения, тем самым облегчив понимание и тестирование программного кода.

Для решения данной задачи существует ряд архитектурных шаблонов: MVP, MVC, MVVM, HMVC и т.д. Каждый из паттернов обладает преимуществами и недостатками по отношению к другим.

MVC возник первым и на его базе были созданы похожие по сути, но разные по возможностям архитектуры. Несмотря на то, что паттерн MVC широко используется, однако в контексте разработки Android-приложений паттерн MVP является предпочтительным. Среди отличий MVP относительно других MV паттернов следует выделить полную независимость Presenter от View, что позволяет учесть особенность Android, а именно сохранение Presenter между жизненными циклами View.

В результате сравнения MV паттернов для разработки данного Android-приложения был выбран паттерн MVP.

Диаграмма паттерна MVP представлена на рисунке 1.

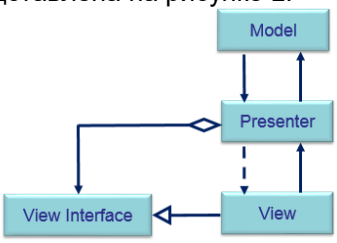

Рисунок 1 – Диаграмма паттерна MVP

Архитектура MVP имеет ряд преимуществ:

presenter можно использовать повторно. Например, при создании Android-приложений элементы пользовательского интерфейса могут иметь одинаковый вид, но разное поведение в зависимости от роли в приложении (набор элементов для ввода данных, который представлен на трех разных экранах, и в одном случае используется только для отображения данных сущности, в другом – для создания данной сущности, в третьем – для изменения), что позволяет не копировать программный код для повторения внешнего вида экрана, а подменяя presenter, менять поведение элементов (поля доступны или недоступны для изменения).

presenter легко тестируется, благодаря его отделенности от элементов View и механизму взаимодействия через интерфейс.

К минусам данной архитектуры можно отнести:

– необходимость создания и поддержки интерфейсов для представлений, что практически всегда приводит к созданию одного дополнительного интерфейса для каждого элемента представления, при этом на одном пользовательском экране может быть размещено несколько View элементов;

– лишний шаблонный код, который, исходя из цели MVP, позволяет максимально облегчить переиспользование и модификацию, с другой же – увеличивает размер проекта

Однако отмеченные минусы паттерна MVC незначительны по сравнению с преимуществами, которые позволяют улучшать архитектуру.

Список использованных источников:

1. Макаревич, В.А. Клиент-серверное программное средство для проверки знаний правил дорожного движения / В.А. Макаревич // Новые информационные технологии в научных исследованиях: м-лы XXIII Всероссийской научно-технической конференции студентов, молодых ученых и специалистов. Том 2. Рязанский государственный радиотехнический университет, 2018. – С.23.