13-14 червня, 2017
ДонНТУ, ФКНТ, корпус 3, ауд. 3.319/1

Тренінг GameHub

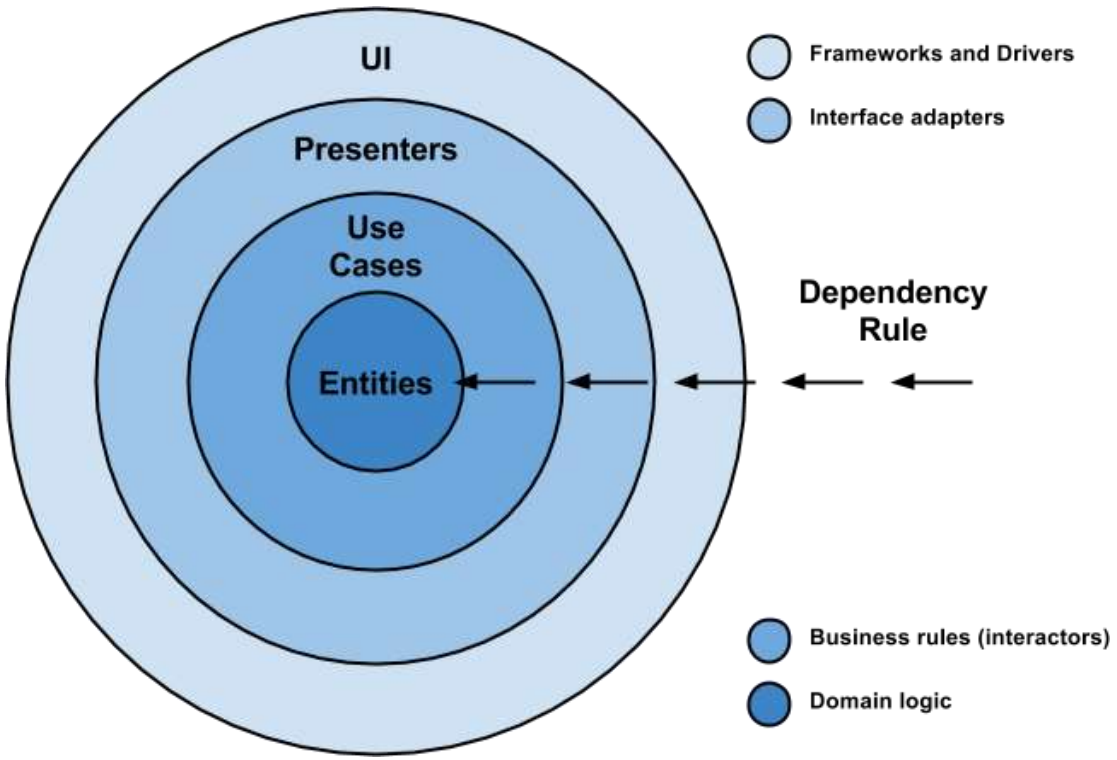
Розробка ігрових додатків на базі Unity та ОС Android

Архітектура і базові принципи проектування ігрових додатків для мобільних пристроїв під керуванням ОС Android

доцент Сергій Цололо, старший викладач Юлія Дікова

Поняття «стрункої архітектури»

Co-funded by the Erasmus+ Programme of the European Union



- Незалежність від фреймворків
- Тестованість
- Незалежність від UI
- Незалежність від БД
- Незалежність від будь-якої зовнішньої служби

Правило залежностей: Код повинен мати залежності тільки у внутрішні кола і не повинен мати ніякого поняття, що відбувається в зовнішніх колах.

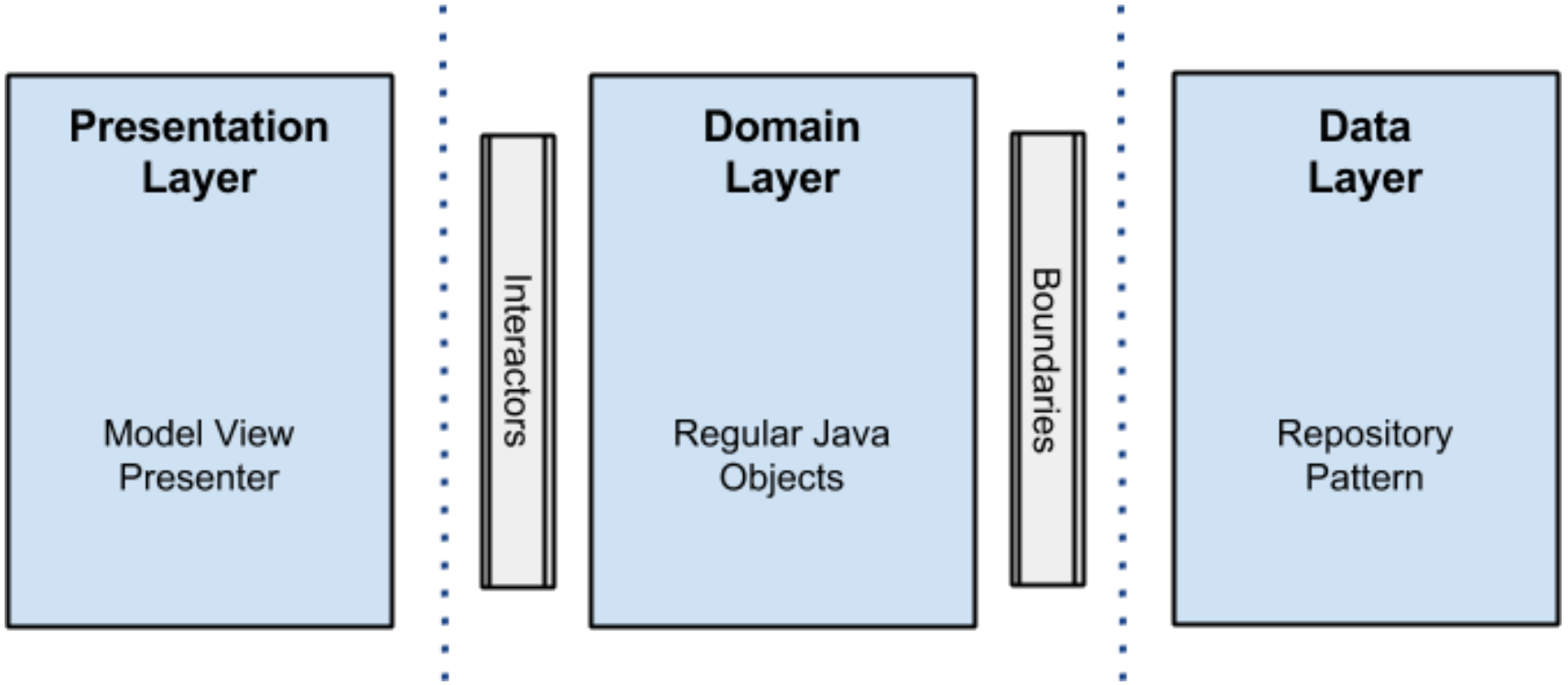
Основні терміни

Co-funded by the Erasmus+ Programme of the European Union
GAMEHUB University Enterprises Cooperation in Game Industry in Ukraine

- ***Entities (Сутності)*** – бізнес-логіка програми
- ***Use Cases (Методи використання)*** – ці методи організовують потік даних в *Entities* і з них. Також їх називають *Interactors* (*Посередниками*)
- ***Interface Adapters (Інтерфейс-Адаптери)*** – цей набір адаптерів перетворює дані з формату, зручного для *Методів використання* і *Сутностей*. До цих адаптерами належать *Presenter*-и і *Controller*-и
- ***Frameworks and Drivers*** – місце скупчення деталей: UI, інструменти, фреймворки, БД тощо

Загальна архітектура Android-додатка

Co-funded by the Erasmus+ Programme of the European Union



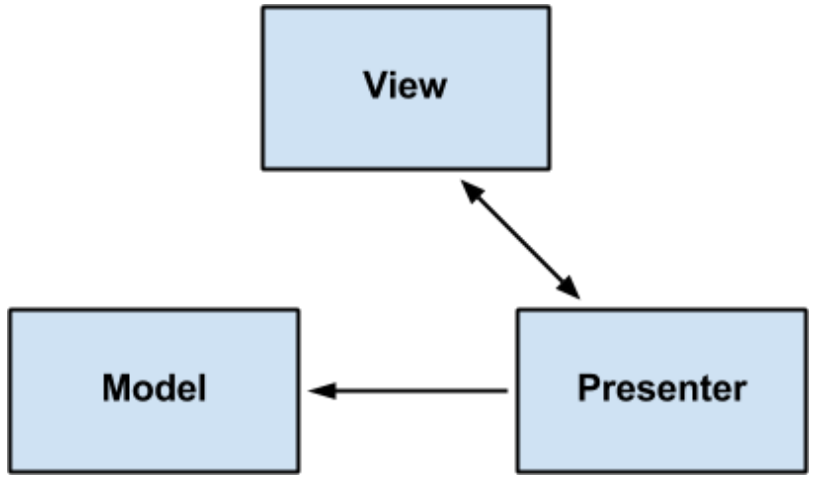
Головна мета – розділення завдань таким чином, щоб бізнес-логіка нічого не знала про зовнішній світ, так, щоб можна було тестувати її без жодних залежностей і зовнішніх елементів

Presentation Layer (шар представлення

Co-funded by the Erasmus+ Programme of the European Union

У цьому шарі логіка зв'язується з ***Views (Уявленнями)*** і запускаються анімації.

Тут можно використовувати різні патерни - наприклад, Model View Presenter (MVP), MVC або MVVM.

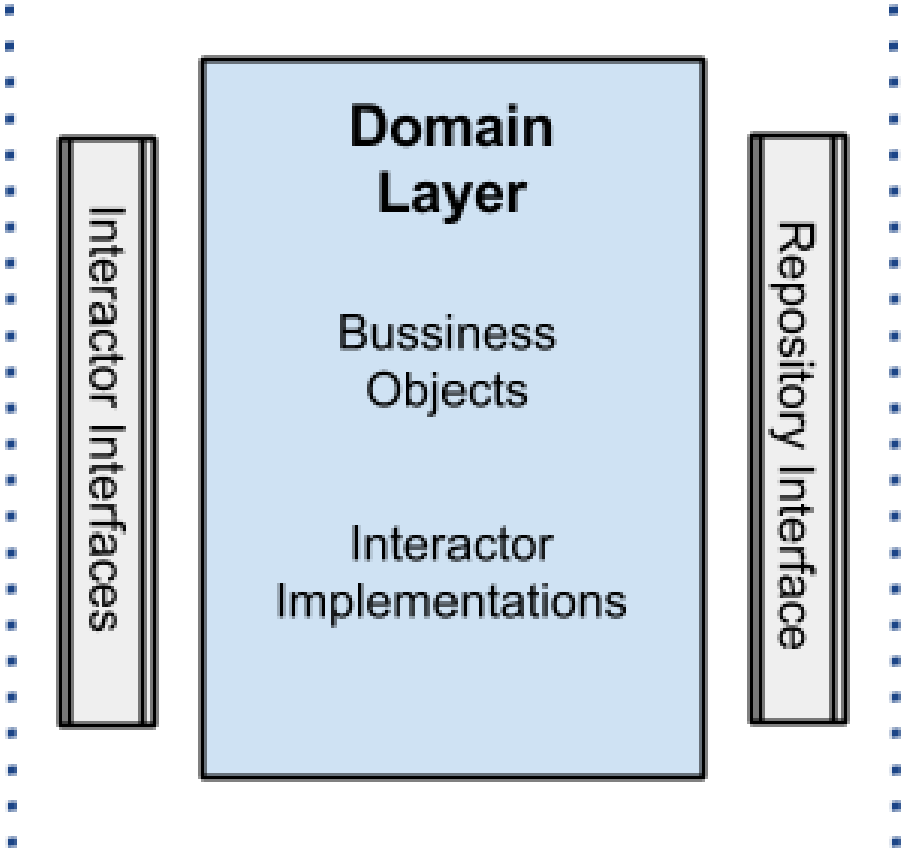


Fragments і ***activities*** є екземплярами Views для логіки UI і рендеринга відображення.

Presenter-и на цьому шарі пов'язуються з **Interactors (посередниками)**, що передбачає роботу в новому потоці (не в UI-потоці), і передачі через callback'и інформації, яка буде відображена у Views.

Domain Layer (шар бізнес-логіки)

Co-funded by the Erasmus+ Programme of the European Union



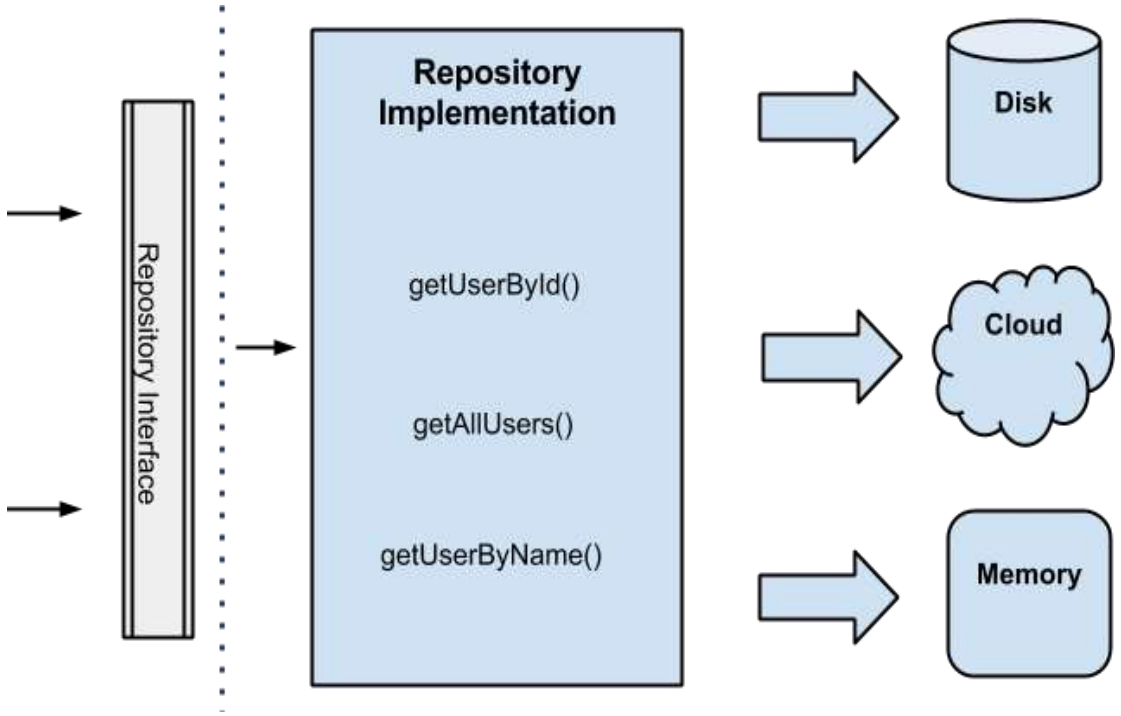
Вся логіка реалізується в цьому шарі шляхом задання **Interactor-ів (Use Cases – методи використання)**.

Частіше за все цей шар є модулем на «чистій» Java без жодних Android-залежностей.

Всі зовнішні компоненти використовують інтерфейси для зв'язку з бізнес-об'єктами.

Data Layer (шар даних)

Co-funded by the Erasmus+ Programme of the European Union



Всі дані, необхідні для додатку, поставляються з цього шару через реалізацію ***UserRepository*** (інтерфейс знаходиться в Domain Layer).

Для реалізації UserRepository можна використати ***Repository Pattern*** зі стратегією, яка через фабрику вибирає різні джерела даних, в залежності від певних умов.

Обробка помилок

Co-funded by the Erasmus+ Programme of the European Union
GAMEHUB University Enterprises Cooperation in Game Industry in Ukraine

Для обробки помилок використовуються callback’и, які повинні мати два базові методи:

- onResponse();
- onError().

Ці методи інкапсулюють виключення у wrapper class (клас-обгортку) під назвою "ErrorBundle".

Такий підхід пов'язаний з деякими труднощами із ланцюжками зворотних викликів «один за іншим», поки помилка не виходить на Presentation Layer, щоб відобразитися.

Тестування кожного з шарів

Co-funded by the Erasmus+ Programme of the European Union

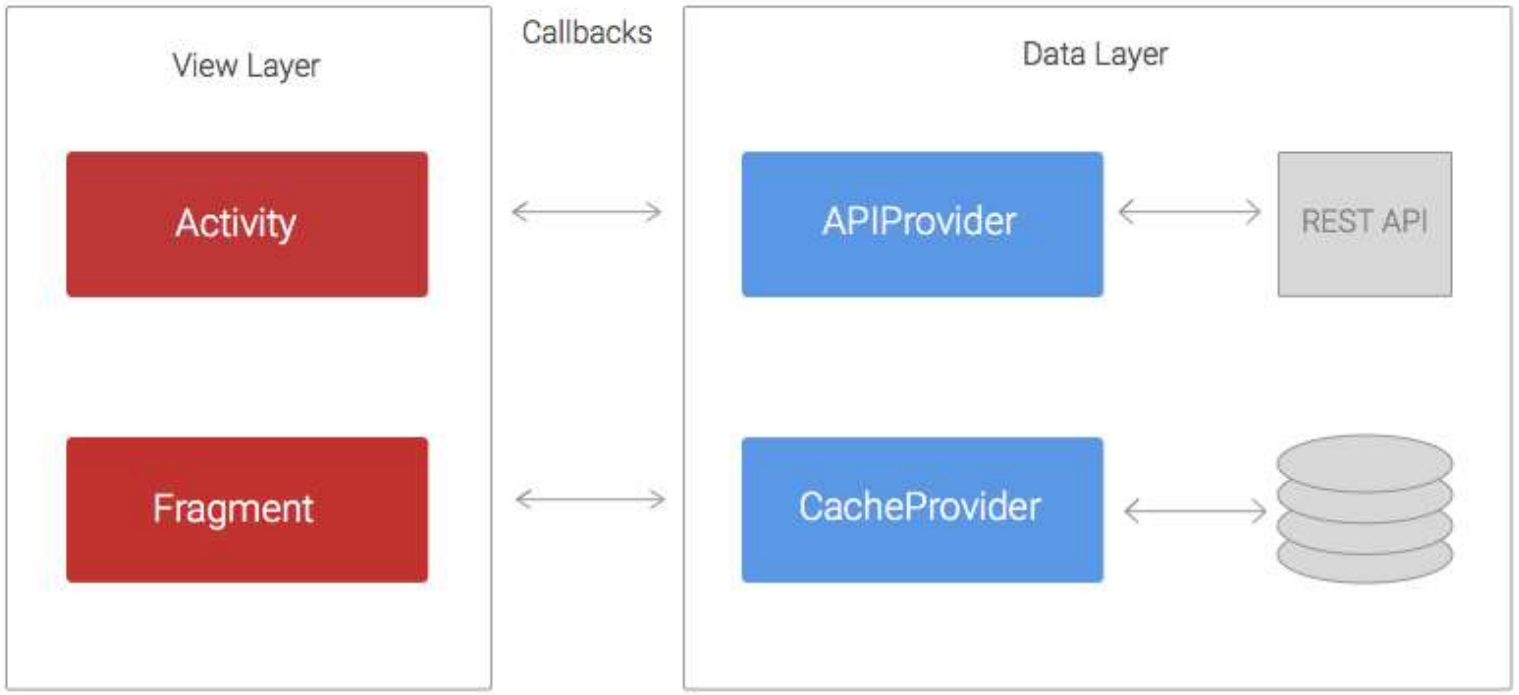
Для тестування усього додатку неможливо використати однакові методи – кожен шар має свої особливості, які повинні бути враховані.

Загальноприйнятої методики немає, але, наприклад, може бути використаний такий розподіл:

1. *Presentation Layer* – існуючі Android-інструменти і espresso для інтеграції і функціонального тестування
2. *Domain Layer* – JUnit + mockito
3. *Data Layer* – Robolectric (шар має android-залежності) + junit для інтеграції і юніт-тестів.

Архітектура на стандартних Activity та

Co-funded by the Erasmus+ Programme of the European Union



Код розділяється на два рівні:

- **рівень даних (Data Layer)**, який відповідав за отримання/збереження даних, що одержуються як через REST API, так і через різні локальні сховища
- **рівень представлення (View Layer)**, що відповідає за обробку та відображення даних

View Layer / Data Layer

Co-funded by the Erasmus+ Programme of the European Union

APIProvider надає методи, що дозволяють Activity і фрагментами взаємодіяти з REST API. Ці методи використовують ***URLConnection*** та ***AsyncTask***, щоб виконати запит у фоновому потоці, а потім доставляють результати в Activity через функції зворотного виклику.

Аналогічно працює ***CacheProvider*** – є методи, які дістають дані з SharedPreferences або SQLite, і є функції зворотного виклику, які повертають результати.

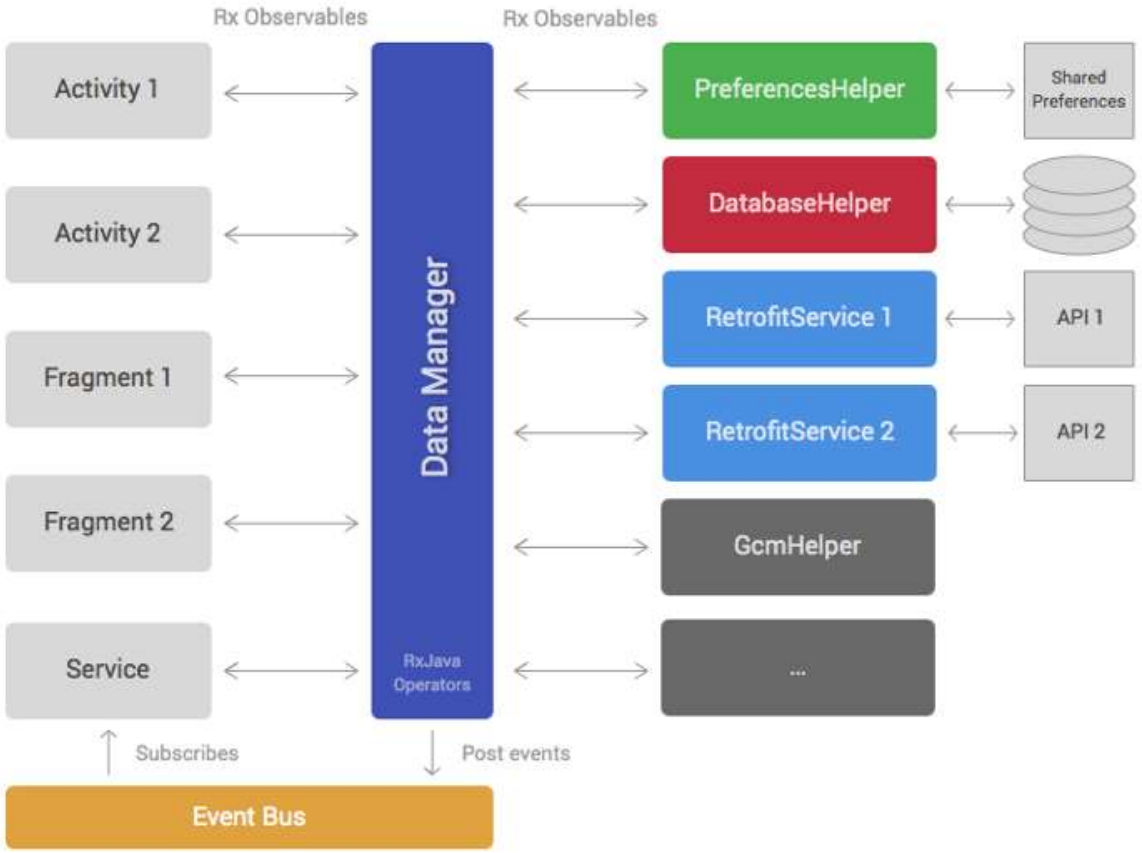
Проблеми архітектури на Activity та A

Co-funded by the Erasmus+ Programme of the European Union

1. Activity та фрагменти стають занадто великими і їх важко підтримувати.
2. Занадто багато рівнів вкладеності призводять до того, що код стає потворним і недоступним для розуміння – це призводить до ускладнення додавання нового функціоналу або внесення змін.
3. Юніт-тестування ускладнюється (якщо не стає взагалі неможливим), тому що багато логіки знаходиться в Activity або фрагментах, які не дуже-то сприяють юніт-тестуванню.

Архітектура з використанням RxJava

Co-funded by the Erasmus+ Programme of the European Union



Код розділяється на два рівні:

- ***рівень даних*** містить DataManager і набір класів-помічників,
- ***рівень уявлення*** складається з класів Android SDK, таких як Activity, Fragment, ViewGroup тощо.

Архітектура з використанням RxJava –

Co-funded by the Erasmus+ Programme of the European Union

DataManager є центральною частиною архітектури. Він використовує оператори RxJava для того, щоб комбінувати, фільтрувати і трансформувати дані, отримані від помічників.

Тобто Activity і фрагменти звільняються від роботи по впорядковуванню даних – DataManager буде виробляти всі потрібні трансформації всередині себе і віддавати дані, готові до відображення.

Архітектура з використанням RxJava –

Co-funded by the Erasmus+ Programme of the European Union

Класи-помічники мають дуже обмежені області відповідальності, і реалізують їх в послідовній манері.

Наприклад, більшість проектів мають класи для доступу до REST API, читання даних з БД або взаємодії з сторонніми SDK.

У різних додатків буде різний набір класів-помічників, наприклад:

- PreferencesHelper – працює з даними в SharedPreferences
- DatabaseHelper – працює з SQLite
- Сервіси Retrofit, що виконують звернення до REST API
- тощо...

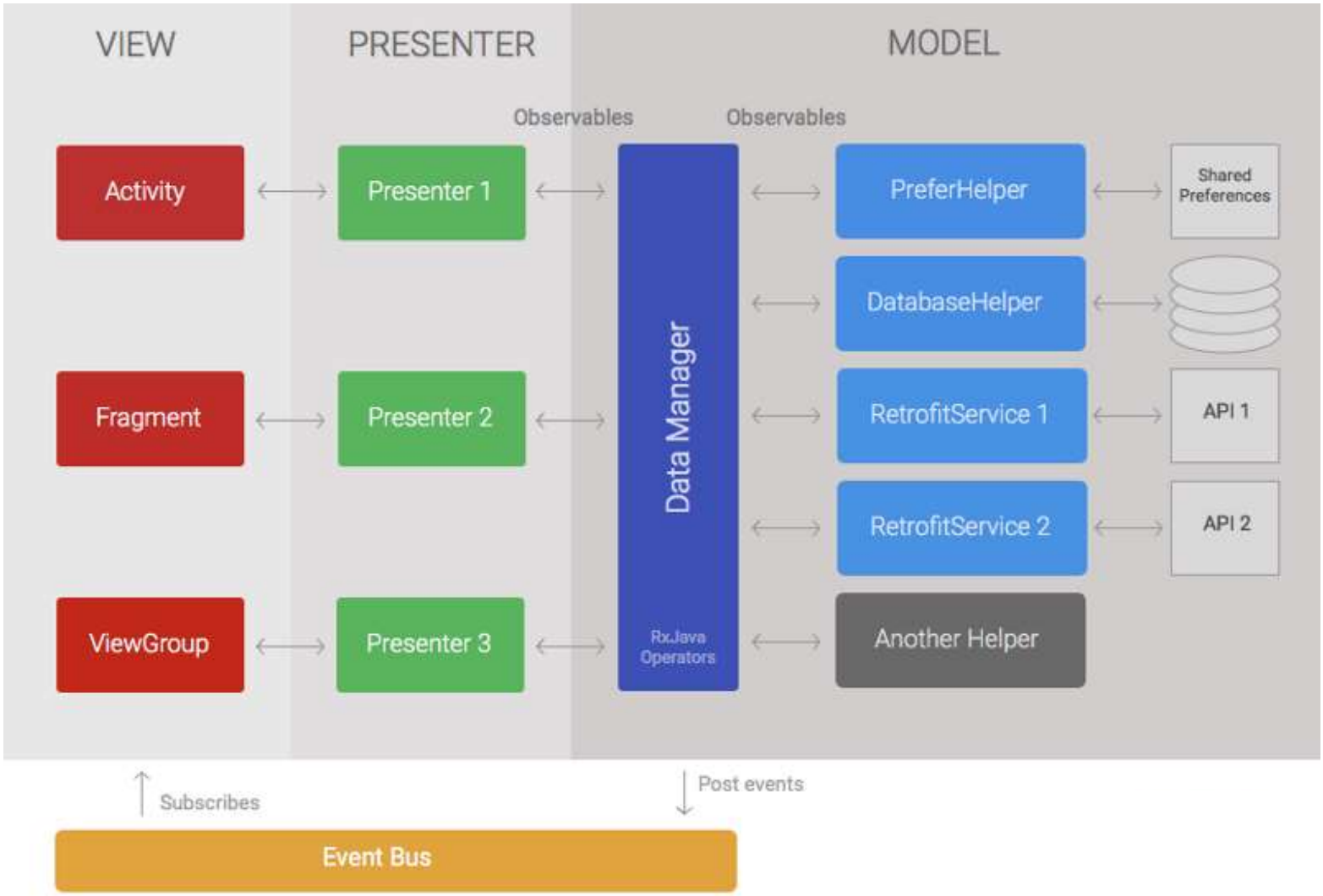
Переваги архітектури з використанням

Co-funded by the Erasmus+ Programme of the European Union

1. Observables і оператори з RxJava позбавляють від вкладених функцій зворотного виклику.
2. DataManager бере на себе роботу, яка раніше виконувалася на рівні уявлення, розвантажуючи таким чином Activity і фрагменти.
3. Переміщення частини коду в DataManager і класи-помічники робить юніт-тестування Activity і фрагментів простішим.
4. Чітке розділення відповідальності і виділення DataManager робить всю архітектуру більш дружньою до тестування.
5. Класи-помічники, або DataManager, можуть бути легко підмінені на спеціальні заглушки.

Перехід до Model View Presenter

Co-funded by the Erasmus+ Programme of the European Union



Рівень Presenter моделі MVP

Co-funded by the Erasmus+ Programme of the European Union

Presenter'и відповідають за завантаження даних з моделі і виклик відповідних методів на рівні уявлення, коли дані завантажені.

Presenter'и підписуються на Observables, які повертаються DataManager. Отже, вони повинні працювати з такими сутностями як підписки і планувальники. Більш того, вони можуть аналізувати виникаючі помилки, або застосовувати додаткові оператори до потокам даних, якщо необхідно.

Наприклад, якщо потрібно відфільтрувати деякі дані, і цей фільтр швидше за все ніде більше використовуватися не буде, є сенс винести цей фільтр на рівень presenter'а, а не DataManager.