INNOVATIVE PUBLICATION
Journal of Effective Learning and Sustainable Innovation
Vol.2 №5 (2024). May
innovativepublication.uz

MVP Arxitekturasining Android ilovalaridagi Afzalliklari va Kamchiliklari

Mahkamov Shohruh Sarvar o'g'li
Mirzo Ulug'bek nomidagi O'zbekiston milliy universiteti Jizzax filiali Kompyuter ilmlari va dasturlashtirish kafedra o'qituvchisi
mahkamov@jbnuu.uz

Jayabharath Reddy Dandu
Sambhram universiteti Jizzax filiali BSc fakulteti BSc kafedra o'qituvchisi

Annotatsiya: Ushbu maqolada Android ilovalarini rivojlantirishda Model-View-Presenter (MVP) arxitekturasi qo'llanishining afzalliklari va kamchiliklari tahlil qilinadi. MVP arxitekturasi loyihaning kod bazasini yaxshi tashkil qilish, testlash osonligini ta'minlash va kodning qayta foydalanish imkoniyatini oshirishga yordam beradi. Shu bilan birga, bu arxitekturaning kamchiliklari ham mavjud bo'lib, ular orasida qo'shimcha kod yozish zarurati va komplekslikni oshirishi mumkinligi kiradi. Maqolada ushbu afzalliklar va kamchiliklar batafsil muhokama qilinadi va Android ilovalari uchun MVP arxitekturasi qachon va qanday holatlarda eng yaxshi tanlov bo'lishi mumkinligi haqida tavsiyalar beriladi.

Kalit so'zlar: MVP arxitekturasi, Android rivojlantirish, Model-View-Presenter, Kodni testlash, Kodni qayta foydalanish, Android ilova arxitekturasi, Dasturiy ta'minot arxitekturasi, Ilova dizayni, Kodni tashkil qilish, Arxitektura taqqoslash.

INNOVATIVE PUBLICATION
Journal of Effective Learning and Sustainable Innovation
Vol.2 №5 (2024). May
innovativepublication.uz

Abstract: This article examines the advantages and disadvantages of using the Model-View-Presenter (MVP) architecture in Android application development. The MVP architecture helps in organizing the codebase efficiently, facilitating easy testing, and enhancing code reusability. However, it also has its drawbacks, including the necessity of writing additional code and potential complexity increase. The article discusses these pros and cons in detail and provides recommendations on when and how MVP architecture might be the best choice for Android applications.

Key words: MVP architecture, Android development, Model-View-Presenter, Code testing, Code reusability, Android application architecture, Software architecture, Application design, Code organization, Architecture comparison

Аннотация: В данной статье рассматриваются преимущества и недостатки использования архитектуры Model-View-Presenter (MVP) в разработке приложений для Android. Архитектура MVP помогает эффективно организовать базу кода, облегчает тестирование и повышает возможность повторного использования кода. Однако она также имеет свои недостатки, включая необходимость написания дополнительного кода и возможное увеличение сложности. В статье подробно обсуждаются эти плюсы и минусы, а также даются рекомендации по поводу того, когда и как архитектура MVP может быть наилучшим выбором для Android приложений.

Ключевые слова: Архитектура MVP, Разработка Android, Model-View-Presenter, Тестирование кода, Повторное использование кода, Архитектура Android приложений, Программная архитектура, Дизайн приложений, Организация кода, Сравнение архитектур.

INNOVATIVE PUBLICATION
Journal of Effective Learning and Sustainable Innovation
Vol.2 №5 (2024). May
innovativepublication.uz

KIRISH

MVP yoki Model-View-Presenter - bu Android ilovalarini ishlab chiqishda keng qo'llaniladigan arxitektura naqshidir. Bu eski Model-View-Controller (MVC) naqshining evolyutsiyasi bo'lib, Android rivojlanishiga xos bo'lgan ba'zi muammolarni hal qilish uchun mo'ljallangan.

MVP komponentlari:

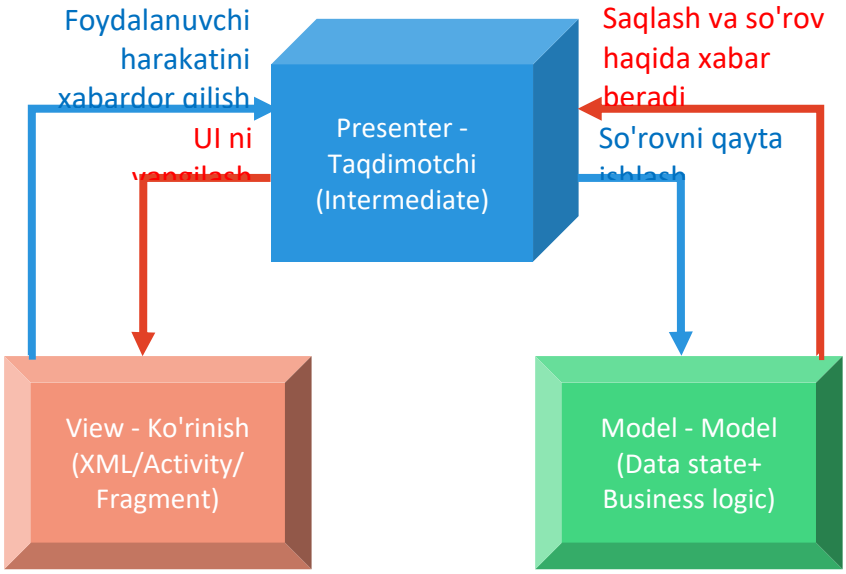
Model - ilovaning ma'lumot va biznes mantiqini boshqaradi, ilovaning ma'lumotlari va biznes mantiqini ifodalaydi va ma'lumotlarni qidirish, saqlash va manipulyatsiya qilish kabi operatsiyalarni bajaradi.

Ko'rinish (View) - foydalanuvchi interfeysini taqdim etadi va uni yangilaydi, ilovaning UI qatlamini ifodalaydi, foydalanuvchiga ma'lumotlarni ko'rsatadi va foydalanuvchi ma'lumotlarini yozib oladi va foydalanuvchi ma'lumotlarini qayta ishlash uchun taqdimotchiga yuboradi.

Taqdimotchi (Presenter) - Model va View o'rtasida vositachilik qiladi, ularning o'zaro aloqa qilishini ta'minlaydi. Model va Ko'rinish o'rtasida vositachi sifatida ishlaydi. Biznes mantig'ini boshqaradi va shunga mos ravishda Model va Ko'rinishni yangilaydi. Ko'rinishni ma'lumotlar manbasidan mustaqil saqlaydi.

Bu usul kodni modullar shaklida ajratib qo'yishga va har bir modulni alohida test qilishga imkon beradi.

INNOVATIVE PUBLICATION
Journal of Effective Learning and Sustainable Innovation
Vol.2 №5 (2024). May
innovativepublication.uz



1-rasm: MVP (Model — View- Presenter) arxitektura patterni

MVP arxitekturasi kodni oson boshqarish, tahlil qilish va qo'llab-quvvatlash imkonini beradi, bu esa dasturchilarga yanada samarali va ishonchli ilovalar yaratishga yordam beradi. Shu bilan birga, bu arxitektura ba'zi kamchiliklarga ham ega bo'lib, qo'shimcha kod yozishni talab qilishi va ba'zi holatlarda komplekslikni oshirishi mumkin.

MUHOKAMA

Xavotirlarni ajratish - MVP mas'uliyatni aniq ajratishda yordam beradi. Model ma'lumotlar va biznes mantig'i uchun javobgardir, View ma'lumotlarni ko'rsatish uchun javobgardir va Taqdimotchi ular orasidagi aloqani boshqaradi.

Sinovga yaroqliligi - MVP ilovangiz komponentlarini birlik sinovini osonlashtiradi. Biznes mantig'i Taqdimotchiga ajratilganligi sababli, siz uni Android ramka komponentlarini jalb qilmasdan sinab ko'rishingiz mumkin, bu test jarayonini soddalashtiradi.

Xizmat ko'rsatish qobiliyati - Xavotirlarni aniq ajratish bilan kod bazasini saqlash va kengaytirish osonroq bo'ladi. Bitta komponentdagi o'zgarishlar (masalan, UI qayta dizayni) boshqalarga ta'sir qilishi shart emas.

Moslashuvchanlik - MVP ko'proq moslashuvchanlik va moslashish imkonini beradi. Masalan, siz biznes mantig'ini (Taqdimotchi) yoki asosiy ma'lumotlarni (Model) o'zgartirmasdan, foydalanuvchi interfeysini amalga oshirishni (Ko'rish) osongina o'zgartirishingiz mumkin.

Oson hamkorlik - MVP ishlab chiquvchilar va dizaynerlar o'rtasidagi hamkorlikni osonlashtiradi. Dizaynerlar UI ustida ishlayotgan paytda ishlab chiquvchilar biznes mantig'iga e'tibor qaratishlari mumkin va bir sohadagi o'zgarishlar boshqasiga kamroq ta'sir qiladi.

MVP qanday ishlaydi:

Foydalanuvchi kiritish - Foydalanuvchi kiritgan ma'lumotlar View (Faoliyat, Fragment va h.k.) tomonidan yozib olinadi.

Taqdimotchi bildirishnomalarni ko‘rishi - Ko'rinish taqdimotchini foydalanuvchining harakatlari haqida xabardor qiladi.

Taqdimotchi biznes mantiqiy jarayonlari - Taqdimotchi Model yordamida biznes mantiqini qayta ishlaydi va shunga mos ravishda Modelni yangilaydi.

Model taqdimotchiga xabar beradi - Model taqdimotchiga ma'lumotlardagi o'zgarishlar haqida xabar beradi.

Taqdimotchi yangilanishlari ko‘rinishi - Taqdimotchi ko'rinishni yangi ma'lumotlar bilan yangilaydi.

Ushbu tsikl foydalanuvchilar ilova bilan o'zaro aloqada bo'lib, tashvishlarni aniq ajratishni ta'minlab, kodlar bazasini yanada modulli va texnik xizmat ko'rsatishni ta'minlaydi.

MVP afzalliklari:

Xavotirlarni ajratish - Biznes mantig'ini (Taqdimotchi) UI (Ko'rish) va ma'lumotlarni qayta ishlash (Model) dan aniq ajratish.

Sinovga yaroqliligi - Biznes mantig'i Android ramka komponentlaridan ajratilganligi sababli birlik sinovini o'tkazish osonroq.

Xizmat ko'rsatish qobiliyati - Bitta komponentdagi o'zgarishlar (masalan, UI qayta dizayni) boshqalarga ta'sir qilishi shart emas, bu esa texnik xizmat ko'rsatishni osonlashtiradi.

MVP ning kamchiliklari:

Murakkablik va qozon plastinkasi - MVP ko'pincha katta miqdordagi qozon kodini talab qiladi. Har bir ko'rinish mos keladigan taqdimotchini talab qiladi, bu interfeyslar va sinflarning ko'payishiga olib keladi. Bu murakkablik kodlar bazasini saqlashni qiyinlashtirishi mumkin, ayniqsa yirik loyihalar uchun.

MVVM ning yuksalishi (Model-View-ViewModel) - MVVM, ayniqsa Google ma'qullashi va LiveData va ViewModel kabi Android arxitektura komponentlarini joriy etishi bilan katta qiziqish uyg'otdi. MVVM hayot aylanishidan xabardor ma'lumotlar bilan ishlashni yaxshiroq qo'llab-quvvatlashni taklif qiladi, xotiraning oqishi xavfini kamaytiradi va kodni boshqarishni osonlashtiradi.

Hayotiy tsiklni boshqarish - MVP tabiatan Android hayotiy tsikli voqealarini boshqarmaydi va uni boshqarishni ishlab chiquvchiga topshiradi. Bu murakkab hayot

aylanishini boshqarish kodiga olib kelishi mumkin. Bundan farqli o'laroq, MVVM kabi arxitektura hayot aylanishidan xabardor komponentlar bilan buni ancha osonlashtiradi.

Ma'lumotlarni ulash - Android-da ma'lumotlarni bog'lashning joriy etilishi MVVM-ni yanada jozibador qildi. Ma'lumotlarni ulash interfeysni yangilashda standart kodga bo'lgan ehtiyojni kamaytiradi, chunki UI komponentlari ma'lumotlar o'zgarishlarini kuzatishi va avtomatik ravishda yangilanishi mumkin.

Sinovga yaroqlilik va masshtablilik muammolari - MVP sinovdan o'tishni targ'ib qilsa-da, Ko'rishlar va Taqdimotchilar o'rtasidagi mahkam bog'liqlik ba'zan sinovni qiyinlashtirishi mumkin. MVVM o'zining ko'proq ajratilgan tabiati bilan ba'zi stsenariylarda yaxshiroq sinovdan o'tishni taklif qilishi mumkin.

Hamjamiyat va yordam - Google va Android hamjamiyati asosan MVVM va MVI (Model-View-Intent) kabi boshqa modellarga o'tish bilan birga, ushbu yangi arxitekturalar uchun ko'proq hamjamiyat yordami, resurslari va asboblari mavjud. Bu yangi ishlab chiquvchilar uchun ularni qabul qilishni osonlashtiradi.

O'rganish egri chizig'i - Yangi ishlab chiquvchilar uchun MVPni o'rganish MVVM bilan solishtirganda qiyinroq bo'lishi mumkin, ayniqsa MVVM uchun mavjud bo'lgan resurslar va jamoat misollari boyligi bilan.

NATIJALAR

MVP arxitekturasidan foydalanadigan holda bir kichik oddiy Android ilovasini yaratish jarayonini ko’rib chiqsak.

1-qadam: Model yaratish (Create Model)

```
public class MvpModel {
    public String getModeling() {
        return "Salom, MVPga Hush kelibsiz";
    }
}
```

2-qadam: Ko'rish interfeysini yaratish (Create View Interface)

```
public interface ModelingView {
    void displayModeling(String modeling);
}
```

3-qadam: Taqdimotchi yaratish (Create Presenter)

```
public class ModelingPresenter {
    private final ModelingView view;
    private final ModelingModel model;
    public ModelingPresenter(ModelingView view, ModelingModel model) {
        this.view = view;
        this.model = model;
    }
    public void loadModeling() {
        String Modeling = model.getModeling();
        view.displayModeling(modeling);
    }
}
```

4-qadam: Ko‘rinishni amalga oshirish (Implement View) (Activity/Fragment)

```
public class ModelingActivity extends AppCompatActivity implements ModelingView {
    private ModelingPresenter presenter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_modeling);
        ModelingModel model = new ModelingModel();
        presenter = new ModelingPresenter(this, model);
        // Trigger the loading of the greeting
        presenter.loadModeling();
    }
    @Override
    public void displayGreeting(String modeling) {
        // Update UI with the greeting
        TextView modelingTextView = findViewById(R.id.modelingTextView);
        modelingTextView.setText(modeling);
    }
}
```

Ushbu misolda:

ModelingModel - ma’lumotlarni ifodalaydi.

ModelingView (ModelingActivity) - tomonidan amalga oshirilgan interfeysdir.

ModelingPresenter - vositachi vazifasini bajaradi va modeldagi ma'lumotlar bilan ko'rinishni yangilaydi.

MVP - bu Android ilovalari uchun mustahkam arxitektura, ayniqsa o'rta va yirik loyihalar uchun foydalidir. Uning asosiy kuchi biznes mantig'ini UIdan ajratishda, ilovani yanada barqaror va sinovdan o'tkazishdadir. Biroq, qo'shimcha murakkablikdagi kichik loyihalar uchun mos kelmasligi mumkin.

XULOSA

Xulosa qilib aytganda MVP arxitekturasi Android ilovalarini ishlab chiqishda ko'plab afzalliklarga ega bo'lsada, ba'zi kamchiliklarni ham e'tiborsiz qoldirib bo'lmaydi.

MVP arxitekturasi Android ilovalarini rivojlantirishda kuchli vosita bo'lib, kodni boshqarish, testlash va qayta foydalanish imkoniyatlarini oshiradi. Shu bilan birga, bu arxitekturaning qo'shimcha kod yozishni talab qilishi va ba'zi holatlarda murakkablikni oshirishi mumkinligini ham hisobga olish lozim. Katta loyihalar uchun MVP arxitekturasi juda mos keladi, kichik loyihalarda esa bu arxitekturaning barcha afzalliklaridan foydalanish ehtiyoji bo'lmasligi mumkin.

Foydalanilgan adabiyotlar:

1. Махкамов, Ш. (2023). Теоретические основы базы данных (мб) и системы управления базами данных (мббт). Информатика и инженерные технологии, 1(1), 90-94.

2. Аликулов, С. Т., & Махкамов, Ш.С. (2018). Дидактическая компетентность педагога. In Развитие интеллектуально-творческого потенциала молодежи: из прошлого в современность (pp. 150-152).

3. Toxir, A., & Lobar, A. (2023). Mobil ilovalar orqali yosh bolalarda uchraydigan nutq buzilishlarini bartaraf etish. In Uz-Conferences (Vol. 1, No. 1, pp. 906-911).

4. Toxir Turg'un o'gli, A. (2023). o‘rinlarni almashtirish usullari. In Uz-Conferences (Vol. 1, No. 1, pp. 133-138).

5. Norqo’ziyev , Q. (2023). Mobil robotlar uchun yo’lni rejalashtirish algoritmi. Research and Implementation. извлечено от https://fer-teach.uz/index.php/rai/article/view/746

6. Тоджиев, А., & Норкузиев, К. (2023). The role of artificial intelligence technology in individualized teaching . Информатика и инженерные технологии, 1(2), 153–156. извлечено от https://inlibrary.uz/index.php/computer-engineering/article/view/25014

7. Абдумаликов А. А. и др. Модель и алгоритмы процесса устройств контроля и мониторинга управления энергоснабжением //Journal of innovations in scientific and educational research. – 2023. – Т. 6. – №. 2. – С. 120-129.