18. Тришарова архітетктура. MVVM

Тришарова архітектура розділяє програму на три основні шари чи рівні:

1. Представлення
   - Відповідає за відображення інтерфейсу користувача та обробку подій, що приходять від користувача.
   - Сюди входять усі елементи графічного інтерфейсу, такі як кнопки, тексти, вікна тощо.

2. Бізнес-логіка
   - Містить логіку, яка визначає, як програма повинна працювати, та взаємодіє з даними через шар даних.
   - Обробляє всі обчислення, логіку бізнес-процесів та приймає рішення на основі вхідних даних.

3.Дані
   - Відповідає за зберігання та отримання даних. Це може включати роботу з базами даних, веб-сервісами, файловою системою тощо.

MVVM розширює концепцію тришарової архітектури та вводить новий рівень, ViewModel. Основні компоненти MVVM:

1. Модель
   - Представляє дані та логіку, що стосується обробки даних.
   - Може включати доступ до бази даних, операції зчитування/запису, обробку даних тощо.

2. Вид
   - Відповідає за відображення інтерфейсу користувача та реагує на події від користувача.
   - Не має логіки взаємодії з даними, а лише відображає дані, які подаються з ViewModel.

3. ViewModel:
   - Містить логіку, яка потрібна для відображення даних на Виді.
   - Забезпечує прослойку між Моделлю та Видом, конвертуючи дані так, щоб їх можна було легко відобразити в інтерфейсі користувача.
   - Реагує на дії користувача та відправляє відповідні запити Моделі.

Таким чином, MVVM дозволяє відокремити бізнес-логіку від логіки інтерфейсу користувача та забезпечує більшу гнучкість та тестову можливість. Вона використовує концепції тришарової архітектури та додає нові елементи для ефективної розробки додатків з інтерфейсом користувача.

Плюси MVVM (Model-View-ViewModel):

1. Розділення обов'язків:
   - MVVM дозволяє чітко розділити логіку бізнес-процесів, представлення та обробку даних, що полегшує розробку та підтримку коду.

2. Тестова можливість:
   - ViewModel може бути легко тестованою, оскільки вона не пов'язана з конкретним інтерфейсом користувача. Це сприяє розвитку та підтримці високоякісного коду.

3.Гнучкість та перевикористання:
   - Завдяки розділенню різних компонентів, можливе більше гнучкості та перевикористання коду. ViewModel може використовуватися в різних частинах програми або навіть в інших проектах.

Мінуси MVVM:

1. Складність:
   - Впровадження MVVM може здаватися складним, оскільки вимагає розуміння концепцій та дотримання правил патерну.

2. Зайві шари:
   - Додавання ViewModel може здатися надмірним для простих додатків або там, де потрібно швидко створити прототип.

3. Додатковий код:
   - Іноді вигода від розділення може бути недостатньою для обґрунтування додаткового обсягу коду, який виникає при використанні MVVM.