**LAPORAN PRAKTIKUM**

|  |  |  |
| --- | --- | --- |
| **IDENTITAS MAHASISWA** | | |
| **Nama Mahasiswa** | : | Firza Aji Karya |
| **NPM** | : | 223307007 |
| **Kelas** | : | 3A |

|  |
| --- |
| **DASAR TEORI** |
| Arsitektur aplikasi adalah seperangkat aturan desain. Mirip dengan cetak biru sebuah rumah, arsitektur memberikan struktur bagi aplikasi Anda. Arsitektur aplikasi yang baik dapat membuat kode Anda andal, fleksibel, skalabel, dan mudah dikelola selama bertahun-tahun.  Mempelajari cara menggunakan ViewModel, salah satu komponen Arsitektur untuk menyimpan data aplikasi Anda. Data yang disimpan tidak akan hilang jika framework menghancurkan dan membuat ulang aktivitas dan fragmen selama perubahan konfigurasi atau peristiwa lainnya.  LiveData adalah class holder data yang dapat diamati dan peka terhadap siklus proses.  Beberapa karakteristik LiveData:   * LiveData menyimpan data; LiveData adalah wrapper yang dapat digunakan dengan semua jenis data. * LiveData dapat diamati, yang berarti bahwa observer akan diberi tahu saat data yang dimiliki objek LiveData berubah. |

|  |
| --- |
| **ALAT DAN BAHAN** |
| 1. Laptop 2. Android Studio 3. GitHub |

|  |
| --- |
| **KODE PROGRAM DAN HASIL** |
| 4. Menambahkan LiveData ke kata acak saat ini   * Di GameViewModel, ubah jenis variabel \_currentScrambledWord menjadi MutableLiveData<String>. LiveData dan MutableLiveData adalah class generik, jadi Anda perlu menentukan jenis data yang disimpannya. * Ubah jenis variabel \_currentScrambledWord menjadi val karena nilai objek LiveData/MutableLiveData * Ubah jenis kolom cadangan currentScrambledWord ke LiveData<String>, karena tidak dapat diubah.      * Untuk mengakses data dalam objek LiveData, gunakan properti value. Di GameViewModel dalam metode getNextWord(), dalam blok else, ubah referensi \_currentScrambledWord menjadi \_currentScrambledWord.value.     5. Melampirkan observer ke objek LiveData   * Di GameFragment, hapus metode updateNextWordOnScreen() dan semua panggilan ke dalamnya. Anda tidak memerlukan metode ini, karena Anda akan melampirkan observer ke LiveData. * Di onSubmitWord(), ubah blok if-else kosong sebagai berikut. Metode yang lengkap akan terlihat seperti ini.      * Teruskan *viewLifecycleOwner* sebagai parameter pertama ke metode observe(). *viewLifecycleOwner* merepresentasikan siklus proses [Tampilan Fragment](https://developer.android.com/reference/kotlin/androidx/fragment/app/Fragment.html?hl=id#getView()). Parameter ini membantu LiveData mengetahui siklus proses GameFragment dan memberi tahu observer hanya jika GameFragment dalam status aktif * Tambahkan lambda sebagai parameter kedua dengan newWord sebagai parameter fungsi. * Dalam isi fungsi ekspresi lambda, tetapkan newWord ke tampilan teks kata acak.      1. Memasang observer ke skor dan jumlah kata  * Dalam GameViewModel, ubah jenis variabel class \_score dan \_currentWordCount menjadi val. * Ubah jenis data variabel \_score dan \_currentWordCount menjadi MutableLiveData, lalu inisialisasikan ke 0. * Ubah jenis kolom cadangan ke LiveData<Int>.      * Di GameViewModel pada awal metode reinitializeData(), ubah referensi \_score dan \_currentWordCount masing-masing menjadi \_score.value dan \_currentWordCount.value.      * Di GameViewModel, dalam metode nextWord(), ubah referensi \_currentWordCount menjadi \_currentWordCount. * Di GameViewModel, di dalam metode increaseScore() dan getNextWord(), ubah referensi \_score dan \_currentWordCount menjadi \_score.value dan \_currentWordCount.value.      * gunakan fungsi Kotlin inc() untuk meningkatkan nilai satu per satu dengan keamanan null.      * Di GameFragment, akses nilai score menggunakan properti value. Dalam metode showFinalScoreDialog(), ubah viewModel.score menjadi viewModel.score.value.      * Di GameFragment dalam metode onViewCreated(), hapus kode yang memperbarui tampilan teks skor dan jumlah kata. * Pada GameFragment di akhir metode onViewCreated(), lampirkan observer untuk score. Teruskan viewLifecycleOwner sebagai parameter pertama ke observer dan ekspresi lambda untuk parameter kedua. * Di akhir metode onViewCreated(), lampirkan observer untuk currentWordCount LiveData. Teruskan viewLifecycleOwner sebagai parameter pertama ke observer dan ekspresi lambda untuk parameter kedua. Di dalam ekspresi lambda, teruskan jumlah kata baru sebagai parameter dan di isi fungsi, tetapkan jumlah kata baru dengan MAX\_NO\_OF\_WORDS     7. Menggunakan LiveData dengan data binding   * Di file build.gradle(Module), aktifkan properti dataBinding pada bagian buildFeatures.      * Mengkonversikan file tata letak menjadi tata letak data binding dengan cara buka game\_fragment.xml pilih tab code      * Di GameFragment, pada awal metode onCreateView(), ubah pembuatan instance variabel binding untuk menggunakan data binding.     8. Menambahkan variable data binding   * Di game\_fragment.xml, di dalam tag <data>, tambahkan tag turunan yang disebut <variable>, deklarasikan properti yang disebut gameViewModel dan jenis GameViewModel. Hal tersebut digunakan untuk mengikat data di ViewModel ke tata letak * Di bawah deklarasi gameViewModel, tambahkan variabel lain di dalam tag <data> jenis Integer, dan beri nama maxNoOfWords. Anda akan menggunakan ini untuk mengikat ke variabel di ViewModel guna menyimpan jumlah kata per game.      * Pada GameFragment di awal metode onViewCreated(), inisialisasi variabel tata letak gameViewModel dan maxNoOfWords.     9. Menggunakan ekspresi binding   * Di game\_fragment.xml, tambahkan atribut text ke tampilan teks textView\_unscrambled\_word. Gunakan variabel tata letak baru, gameViewModel dan tetapkan @{gameViewModel.currentScrambledWord} ke atribut text.      * Perbarui atribut text untuk tampilan teks score dengan ekspresi binding berikut. Gunakan resource string score dan teruskan gameViewModel.score sebagai parameter resource.     10. Menguji aplikasi Unscramble dengan Talkback diaktifkan   * Dalam GameViewModel, konversikan kata acak String menjadi string Spannable. String spannable adalah string yang berisi beberapa informasi tambahan.     11. Menghapus kode yang tidak digunakan   * Dalam GameFragment, hapus metode getNextScrambledWord() dan onDetach().        * Dalam GameViewModel, hapus metode onCleared().     **Hasil Akhir :** |

|  |
| --- |
| **KESIMPULAN** |
| Pada praktikum ini kita dapat mengetahui apa yang dimaksud dengan Livedata, kita dapat mengerti bagaimana cara menambahkan LiveData ke kata acak saat ini, Melampirkan observer ke objek LiveData, Menggunakan LiveData dengan data binding, Menambahkan variabel data binding dsb. |