

← → ↺

developer.android.com/courses/pathways/android-development-with-kotlin-9#codelab-https://developer.android.com/codelabs/kotlin-android-training-coroutines-and-...

🔍 搜索

🌐 中文 - 简体

👤 凡德

developers

平台 Android Studio Google Play Jetpack Kotlin 文档 新闻

第 9 课：应用架构（持久性）

使用 Room 库创建数据库并使用协程来简化异步编程。

2 篇文章 • 1 个测验

9

第 9 课：应用架构（持久性）

Badge earned!

View profile

✓ 创建 Room 数据库

Codelab

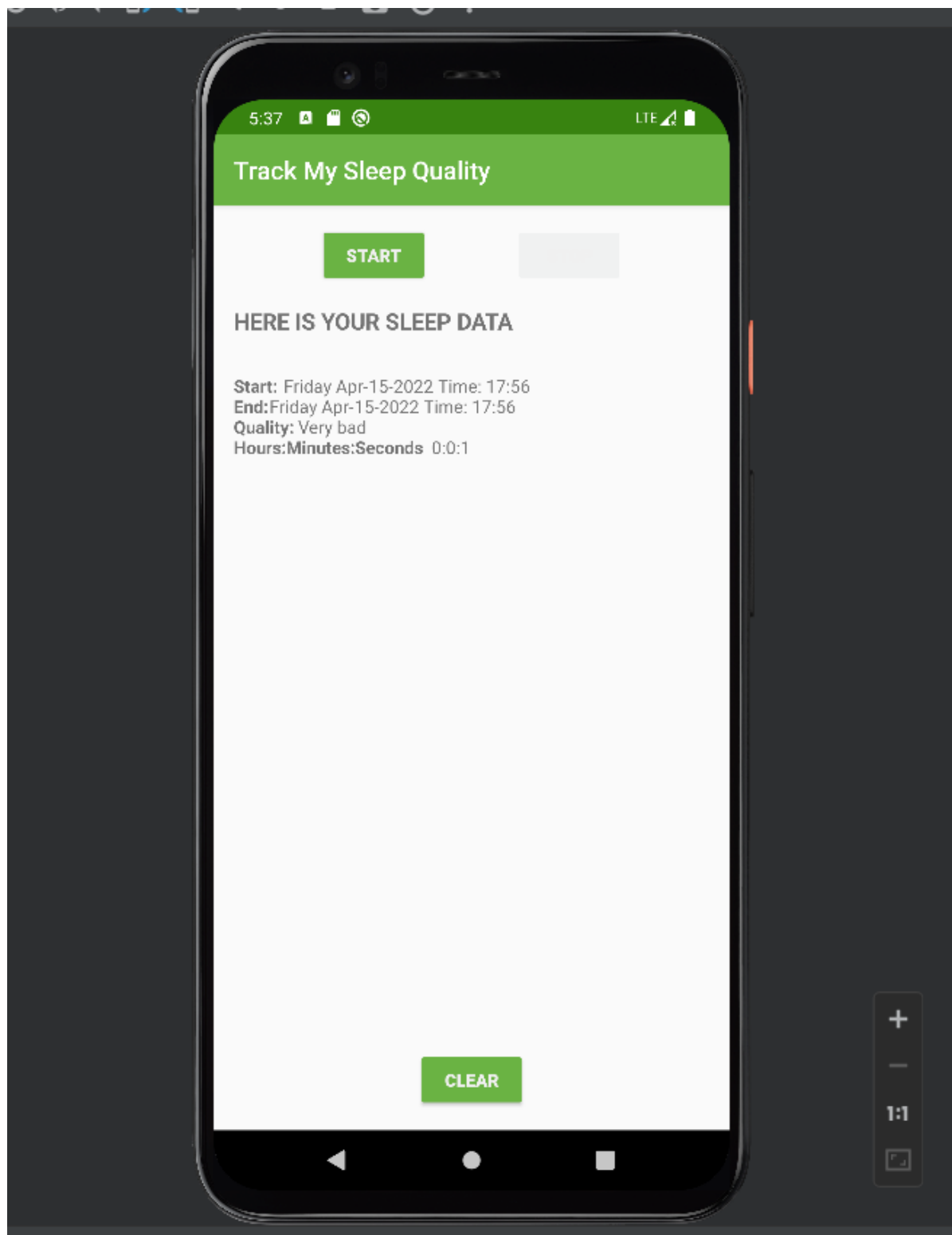
✓ 协程和 Room

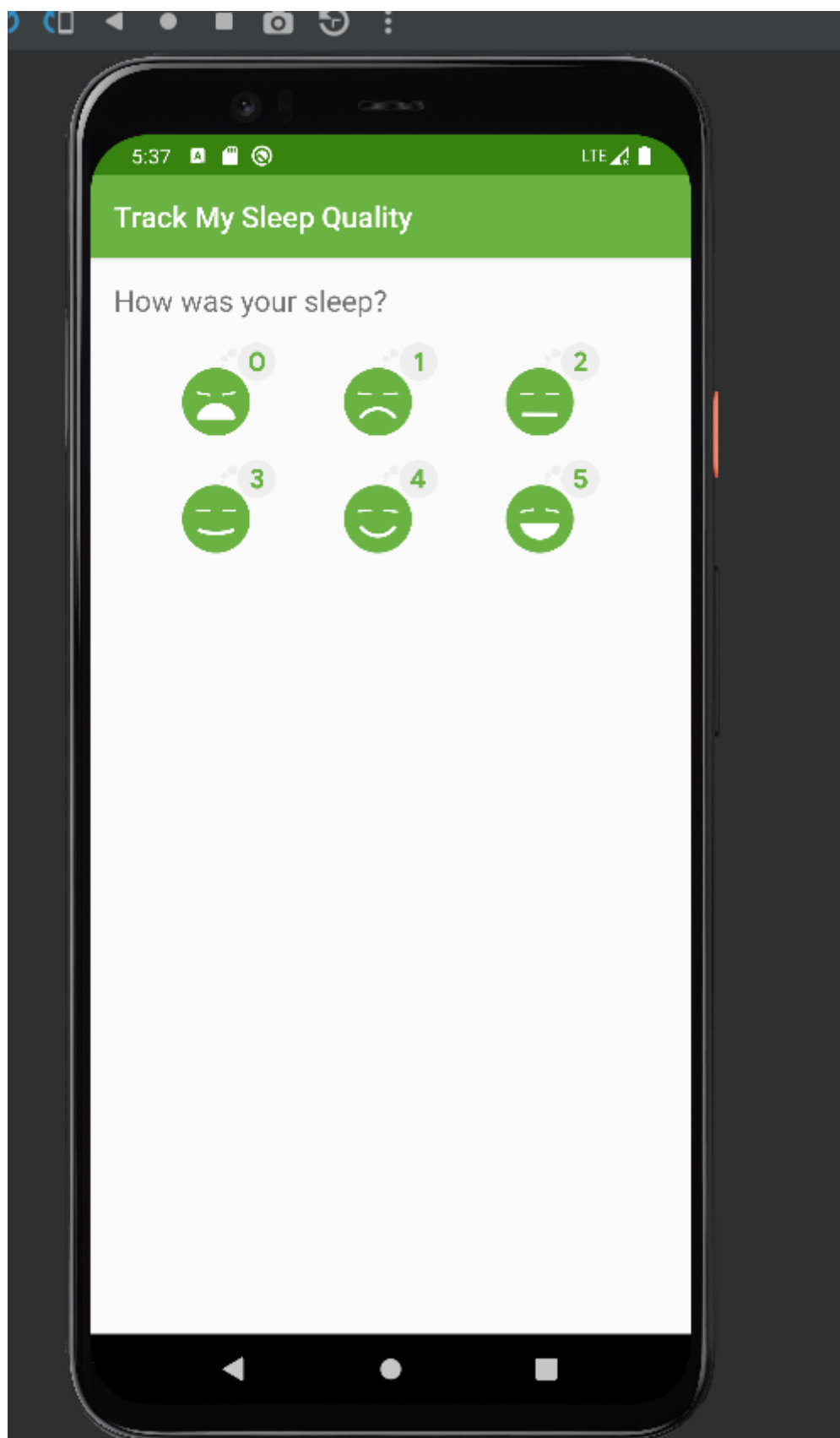
Codelab

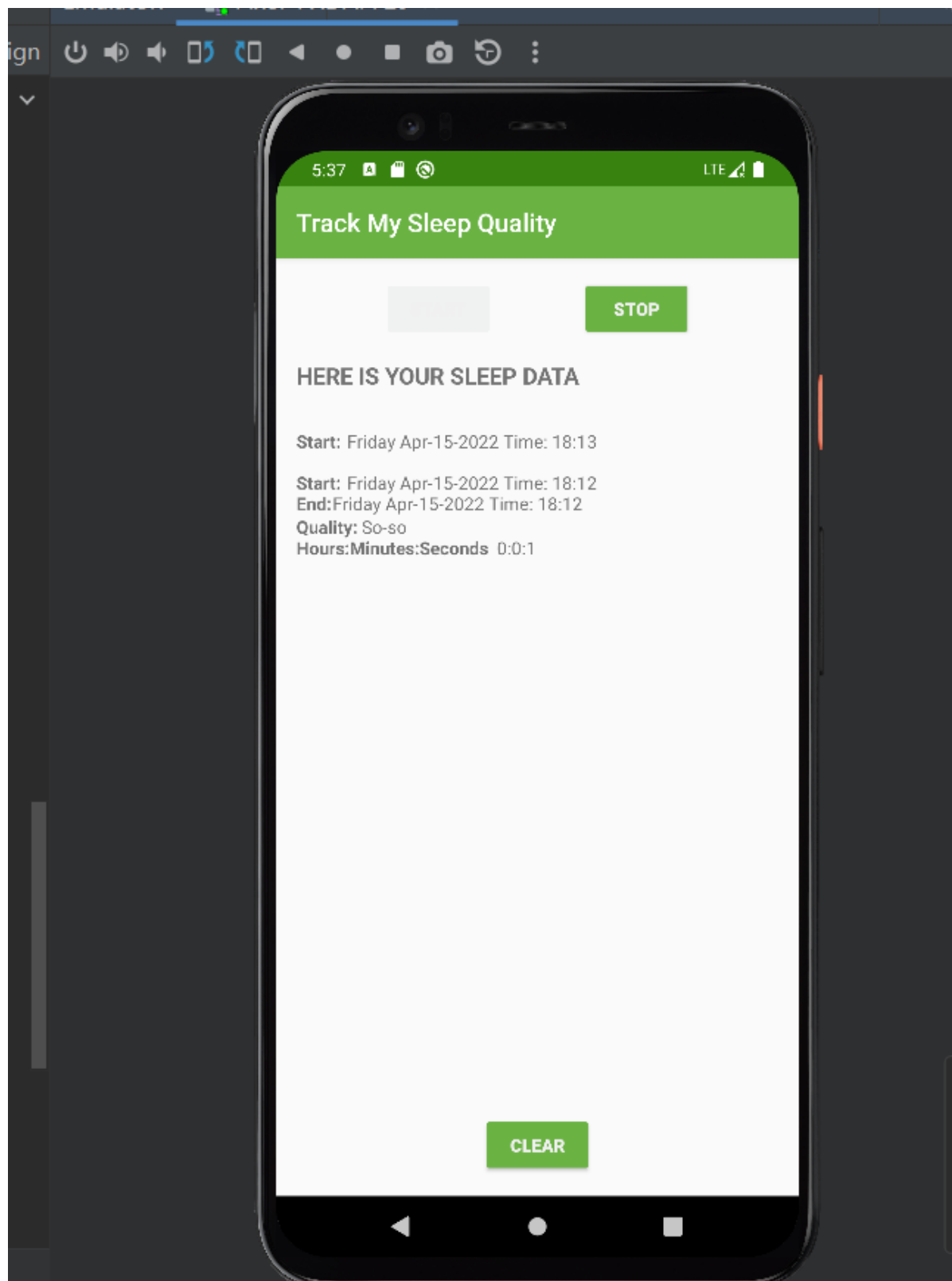
测验

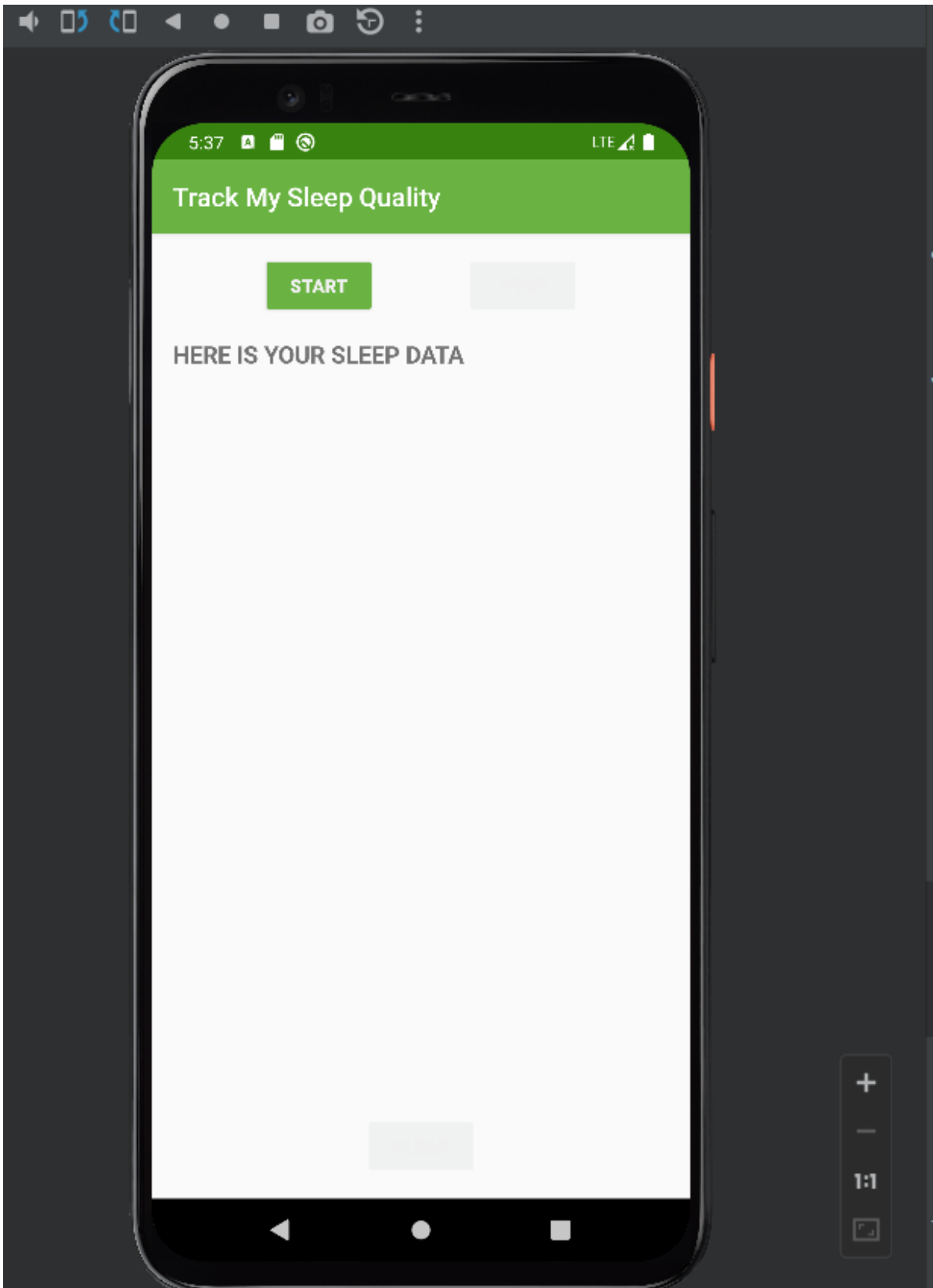
完成“第 9 课：应用架构（持久性）”，赢取徽章。

参加测验









```
kt x fragment_sleep_tracker.xml x SleepDatabase.kt x SleepQualityViewModel.kt x SleepQualityViewModelFactory.kt x SleepQualityFragment.kt x fragment_sleep_quality
18
19 import ...
23
24 @Database(entities = [SleepNight::class], version = 1, exportSchema = false)
25 abstract class SleepDatabase : RoomDatabase() {
26
27     abstract val sleepDatabaseDao: SleepDatabaseDao
28
29     companion object {
30
31         @Volatile
32         private var INSTANCE: SleepDatabase? = null
33
34         fun getInstance(context: Context): SleepDatabase {
35             synchronized(lock = this) {
36                 var instance = INSTANCE
37
38                 if (instance == null) {
39                     instance = Room.databaseBuilder(
40                         context.applicationContext,
41                         SleepDatabase::class.java,
42                         name = "sleep_history_database"
43                     )
44                         .fallbackToDestructiveMigration()
45                         .build()
46                     INSTANCE = instance
47                 }
48                 return instance
49             }
50         }
51     }
}
```

```
kt x fragment_sleep_tracker.xml x SleepDatabase.kt x SleepDatabaseDao.kt x SleepQualityViewModel.kt x SleepQualityViewModelFactory.kt x SleepQualityFragment.kt
1 Copyright 2019, The Android Open Source Project .../
16
17 package com.example.android.trackmysleepquality.database
18
19 import ...
24
25 @Dao
26 interface SleepDatabaseDao {
27
28     @Insert
29     suspend fun insert(night: SleepNight)
30
31     @Update
32     suspend fun update(night: SleepNight)
33
34     @Query(value = "SELECT * from daily_sleep_quality_table WHERE nightId = :key")
35     suspend fun get(key: Long): SleepNight?
36
37     @Query(value = "DELETE FROM daily_sleep_quality_table")
38     suspend fun clear()
39
40     @Query(value = "SELECT * FROM daily_sleep_quality_table ORDER BY nightId DESC LIMIT 1")
41     suspend fun getTonight(): SleepNight?
42
43     @Query(value = "SELECT * FROM daily_sleep_quality_table ORDER BY nightId DESC")
44     fun getAllNights(): LiveData<List<SleepNight>>
45 }
```

```

18
19 import ...
20
21
22
23
24 @Database(entities = [SleepNight::class], version = 1, exportSchema = false)
25 abstract class SleepDatabase : RoomDatabase() {
26
27     abstract val sleepDatabaseDao: SleepDatabaseDao
28
29     companion object {
30
31         @Volatile
32         private var INSTANCE: SleepDatabase? = null
33
34         fun getInstance(context: Context): SleepDatabase {
35             synchronized(lock: this) {
36                 var instance = INSTANCE
37
38                 if (instance == null) {
39                     instance = Room.databaseBuilder(
40                         context.applicationContext,
41                         SleepDatabase::class.java,
42                         name: "sleep_history_database"
43                     )
44                         .fallbackToDestructiveMigration()
45                         .build()
46                     INSTANCE = instance
47                 }
48                 return instance
49             }
50         }
51     }
52 }

```

```

sleepquality / sleepquality / SleepQualityFragment / onCreateView(inflater: LayoutInflater, containe...
fragment_sleep_tracker.xml x SleepDatabase.kt x SleepDatabaseDao.kt x SleepQualityViewModel.kt x SleepQualityViewModelFactory.kt x SleepQ
class SleepQualityFragment : Fragment() {
    /**
     * Called when the Fragment is ready to display content to the screen.
     *
     * This function uses DataBindingUtil to inflate R.layout.fragment_sleep_quality.
     */
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?): View? {
        // Get a reference to the binding object and inflate the fragment views.
        val binding: FragmentSleepQualityBinding = DataBindingUtil.inflate(
            inflater, R.layout.fragment_sleep_quality, container, attachToParent: false)

        val application = requireNotNull(this.activity).application
        val arguments = SleepQualityFragmentArgs.fromBundle(requireArguments())
        val dataSource = SleepDatabase.getInstance(application).sleepDatabaseDao
        val viewModelFactory = SleepQualityViewModelFactory(arguments.sleepNightKey, dataSource)
        val sleepQualityViewModel =
            ViewModelProvider(
                owner: this, viewModelFactory).get(SleepQualityViewModel::class.java)
        binding.sleepQualityViewModel = sleepQualityViewModel
        sleepQualityViewModel.navigateToSleepTracker.observe( owner: this, Observer { it: Boolean?
            if (it == true) { // Observed state is true.
                this.findNavController().navigate(
                    SleepQualityFragmentDirections.actionSleepQualityFragmentToSleepTrackerFragment())
                sleepQualityViewModel.doneNavigating()
            }
        })
        return binding.root
    }
}

```

```

class SleepQualityViewModel(
    private val sleepNightKey: Long = 0L,
    val database: SleepDatabaseDao
) : ViewModel() {
    private val _navigateToSleepTracker = MutableLiveData<Boolean?>()

    val navigateToSleepTracker: LiveData<Boolean?>
        get() = _navigateToSleepTracker

    fun doneNavigating() {
        _navigateToSleepTracker.value = null
    }

    fun onSetSleepQuality(quality: Int) {
        viewModelScope.launch { this: CoroutineScope
            val tonight = database.get(sleepNightKey) ?: return@launch
            tonight.sleepQuality = quality
            database.update(tonight)

            // Setting this state variable to true will alert the observer and trigger navigation.
            _navigateToSleepTracker.value = true
        }
    }
}

```

```

package com.example.android.trackmysleepquality.sleepquality

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import com.example.android.trackmysleepquality.database.SleepDatabaseDao

class SleepQualityViewModelFactory(
    private val sleepNightKey: Long,
    private val dataSource: SleepDatabaseDao
) : ViewModelProvider.Factory {
    @Suppress("unchecked_cast")
    override fun <T : ViewModel?> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(SleepQualityViewModel::class.java)) {
            return SleepQualityViewModel(sleepNightKey, dataSource) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}

```

```

42  * This function uses DataBindingUtil to inflate R.layout.fragment_sleep_quality.
43  */
44  override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
45                          savedInstanceState: Bundle?): View? {
46
47      // Get a reference to the binding object and inflate the fragment views.
48      val binding: FragmentSleepTrackerBinding = DataBindingUtil.inflate(
49          inflater, R.layout.fragment_sleep_tracker, container, attachToParent: false)
50
51      val application = requireNotNull(this.activity).application
52
53      val dataSource = SleepDatabase.getInstance(application).sleepDatabaseDao
54
55      val viewModelFactory = SleepTrackerViewModelFactory(dataSource, application)
56
57      val sleepTrackerViewModel =
58          ViewModelProvider(
59              owner: this, viewModelFactory).get(SleepTrackerViewModel::class.java)
60
61      binding.setLifecycleOwner(this)
62
63      binding.sleepTrackerViewModel = sleepTrackerViewModel
64
65      sleepTrackerViewModel.navigateToSleepQuality.observe(owner: this, Observer { night ->
66          night?.let { it: SleepNight
67              this.findNavController().navigate(
68                  SleepTrackerFragmentDirections
69                      .actionSleepTrackerFragmentToSleepQualityFragment(night.nightId))
70              sleepTrackerViewModel.doneNavigating()
71          }
72      })

```



```

32  */
33  class SleepTrackerViewModel(
34      val database: SleepDatabaseDao,
35      application: Application) : AndroidViewModel(application) {
36
37      private val nights = database.getAllNights()
38      private val _navigateToSleepQuality = MutableLiveData<SleepNight>()
39
40      val navigateToSleepQuality: LiveData<SleepNight>
41          get() = _navigateToSleepQuality
42      val nightsString = Transformations.map(nights) { nights ->
43          formatNights(nights, application.resources)
44      }
45
46      private var _tonight = MutableLiveData<SleepNight?>()
47
48      init {
49          initializeTonight()
50      }
51      val startButtonVisible = Transformations.map(_tonight) { it: SleepNight?
52          it == null
53      }
54      val stopButtonVisible = Transformations.map(_tonight) { it: SleepNight?
55          it != null
56      }
57      val clearButtonVisible = Transformations.map(nights) { it: List<SleepNight>?
58          it?.isEmpty()
59      }
60
61      private fun initializeTonight() {
62          viewModelScope.launch { this: CoroutineScope
63              tonight.value = getTonightFromDatabase()
64          }
65      }

```

```

1  // Copyright 2019, The Android Open Source Project .../
2
3  package com.example.android.trackmysleepquality.sleeptracker
4
5  import ...
6
7  /**
8   * This is pretty much boiler plate code for a ViewModel Factory.
9   * Provides the SleepDatabaseDao and context to the ViewModel.
10  */
11  class SleepTrackerViewModelFactory(
12      private val dataSource: SleepDatabaseDao,
13      private val application: Application) : ViewModelProvider.Factory {
14      @Suppress("unchecked_cast")
15      override fun <T : ViewModel?> create(modelClass: Class<T>): T {
16          if (modelClass.isAssignableFrom(SleepTrackerViewModel::class.java)) {
17              return SleepTrackerViewModel(dataSource, application) as T
18          }
19          throw IllegalArgumentException("Unknown ViewModel class")
20      }
21  }

```

```
SleepTrackerViewModel.kt x SleepTrackerViewModelFactory.kt x fragment_sleep_quality.xml x SleepTrackerFragment.kt x fragment_sleep_tracker.xml x SleepDatabase.kt x Code
86
87
88 <ImageView
89     android:id="@+id/quality_three_image"
90     android:layout_width="@dimen/icon_size"
91     android:layout_height="@dimen/icon_size"
92     android:layout_marginStart="@dimen/margin"
93     android:layout_marginTop="@dimen/margin"
94     android:layout_marginBottom="@dimen/margin"
95     android:contentDescription="@string/quality_3"
96     app:layout_constraintBottom_toBottomOf="parent"
97     app:layout_constraintEnd_toStartOf="@+id/quality_four_image"
98     app:layout_constraintStart_toStartOf="parent"
99     app:layout_constraintTop_toBottomOf="@+id/quality_zero_image"
100     app:layout_constraintVertical_bias="0.0"
101     android:onClick="@{() -> sleepQualityViewModel.onSetSleepQuality(3)}"
102     app:srcCompat="@drawable/ic_sleep_3" />
103
104 <ImageView
105     android:id="@+id/quality_four_image"
106     android:layout_width="@dimen/icon_size"
107     android:layout_height="@dimen/icon_size"
108     android:contentDescription="@string/quality_4"
109     app:layout_constraintBottom_toBottomOf="@+id/quality_three_image"
110     app:layout_constraintEnd_toStartOf="@+id/quality_five_image"
111     app:layout_constraintStart_toEndOf="@+id/quality_three_image"
112     app:layout_constraintTop_toTopOf="@+id/quality_three_image"
113     android:onClick="@{() -> sleepQualityViewModel.onSetSleepQuality(4)}"
114     app:srcCompat="@drawable/ic_sleep_4" />
```

```
app / src / main / res / layout / fragment_sleep_tracker.xml
SleepTrackerViewModel.kt x SleepTrackerViewModelFactory.kt x fragment_sleep_quality.xml x SleepTrackerFragment.kt x fragment_sleep_tracker.xml x SleepDatabase.kt x Split
67 you can set it for the Views using this lambda pattern. -->
68
69 <Button
70     android:id="@+id/start_button"
71     style="@style/SleepButtons"
72     android:layout_width="wrap_content"
73     android:layout_height="wrap_content"
74     android:layout_marginStart="16dp"
75     android:text="Start"
76     app:layout_constraintBaseline_toBaselineOf="@id/stop_button"
77     app:layout_constraintEnd_toStartOf="@+id/stop_button"
78     app:layout_constraintHorizontal_chainStyle="spread"
79     app:layout_constraintStart_toStartOf="parent"
80     android:onClick="@{() -> sleepTrackerViewModel.onStartTracking()}"
81     android:enabled="@{sleepTrackerViewModel.startButtonVisible}" />
82
83 <Button
84     android:id="@+id/stop_button"
85     style="@style/SleepButtons"
86     android:layout_width="wrap_content"
87     android:layout_height="wrap_content"
88     android:layout_marginTop="16dp"
89     android:layout_marginEnd="16dp"
90     android:text="Stop"
91     app:layout_constraintEnd_toEndOf="parent"
92     app:layout_constraintStart_toEndOf="@+id/start_button"
93     app:layout_constraintTop_toTopOf="parent"
94     android:onClick="@{() -> sleepTrackerViewModel.onStopTracking()}"
95     android:enabled="@{sleepTrackerViewModel.stopButtonVisible}" />
layout > androidx.constraintlayout.widget.ConstraintLayout > Button
```