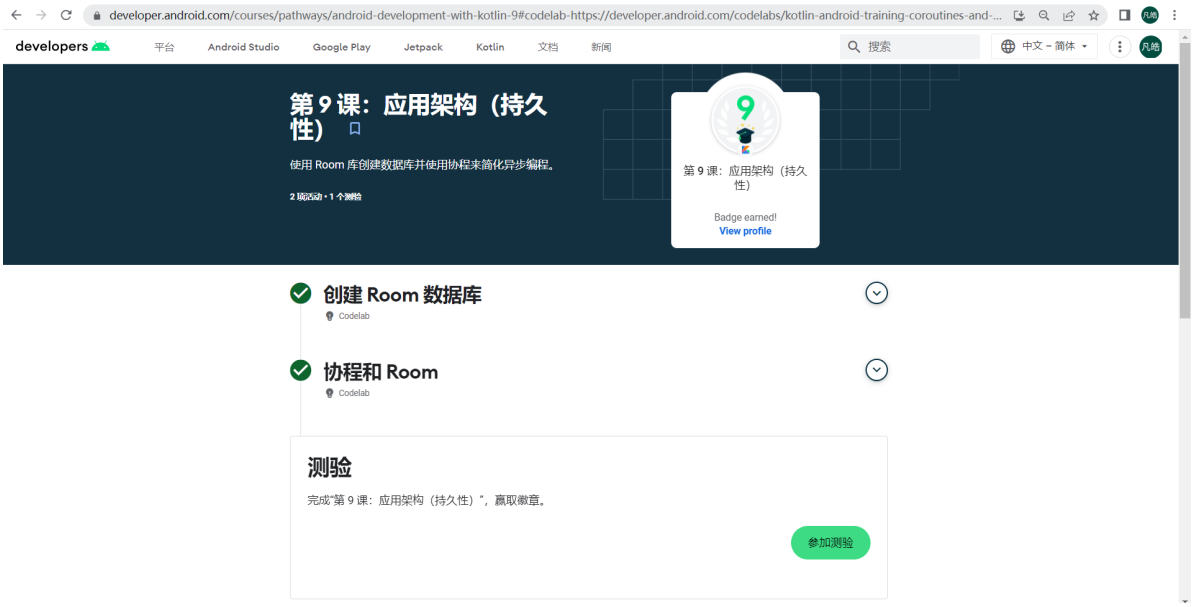
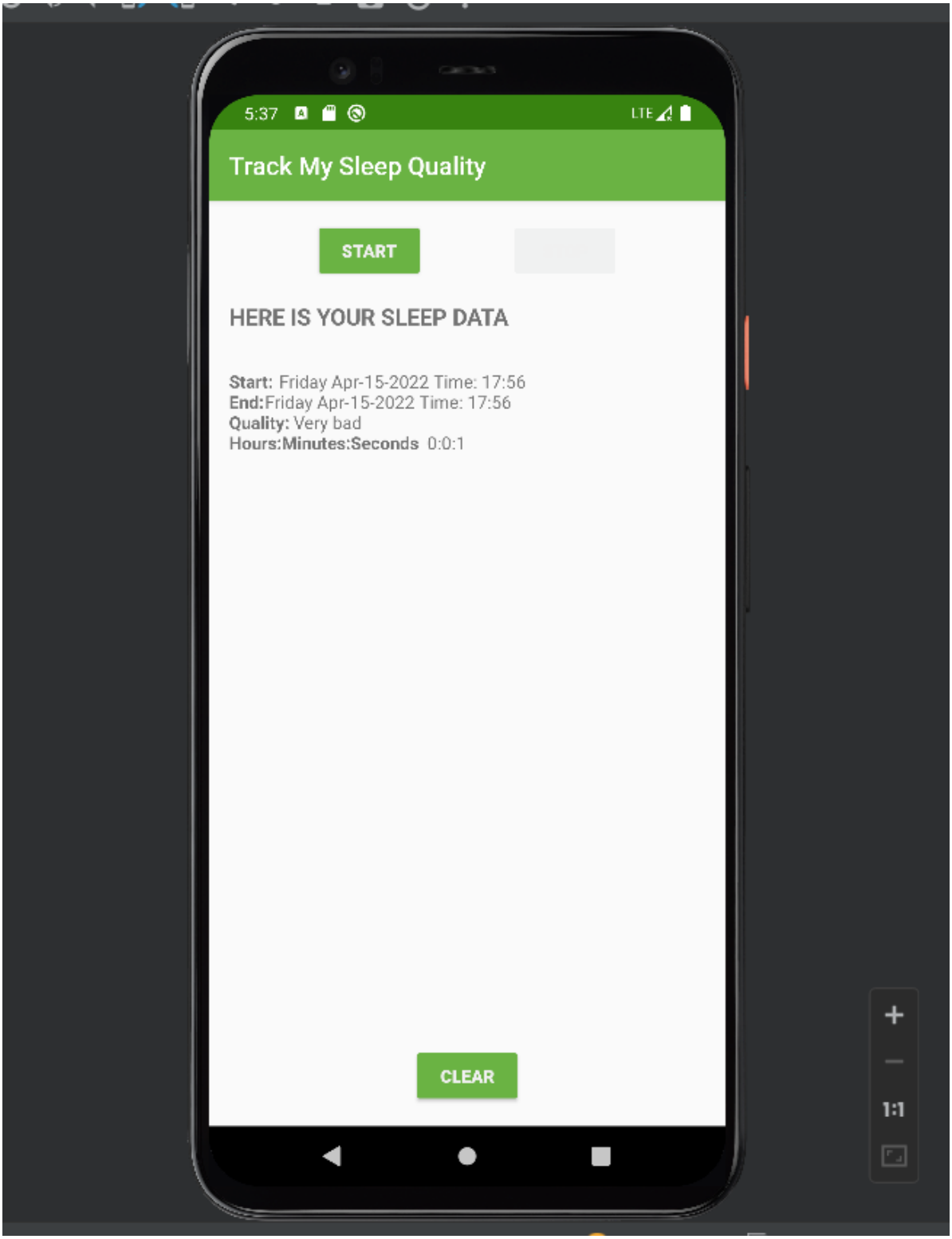
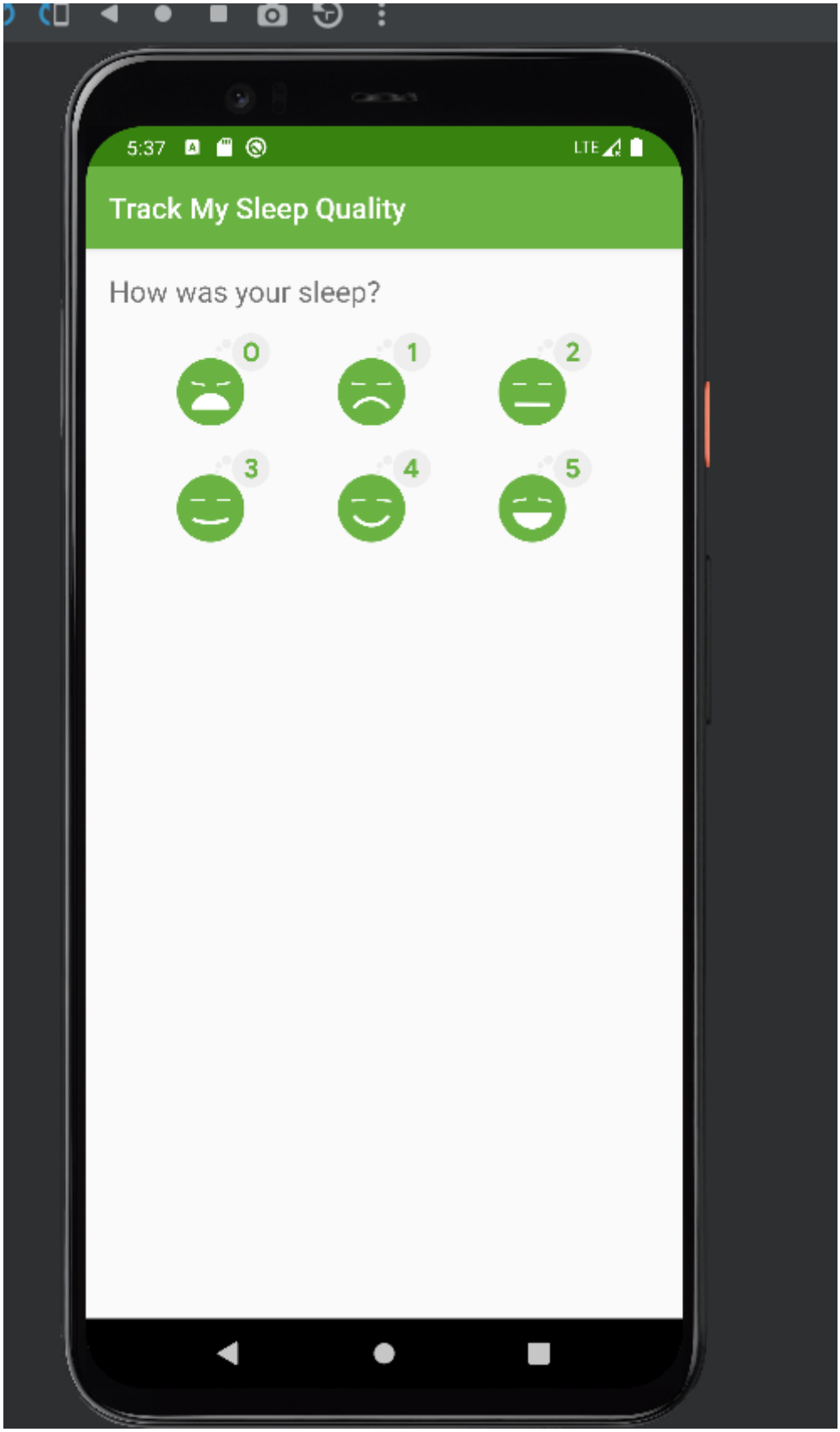
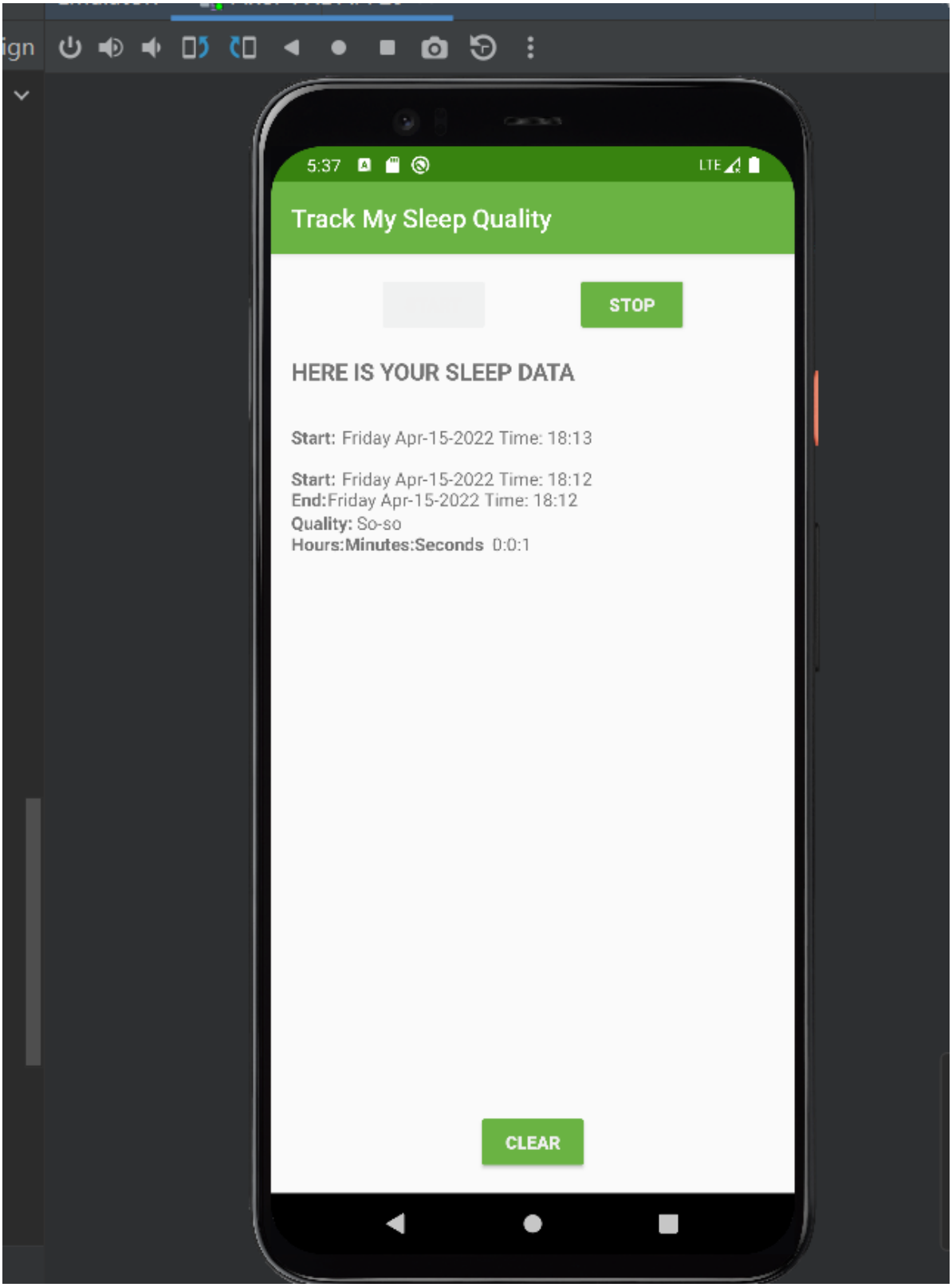
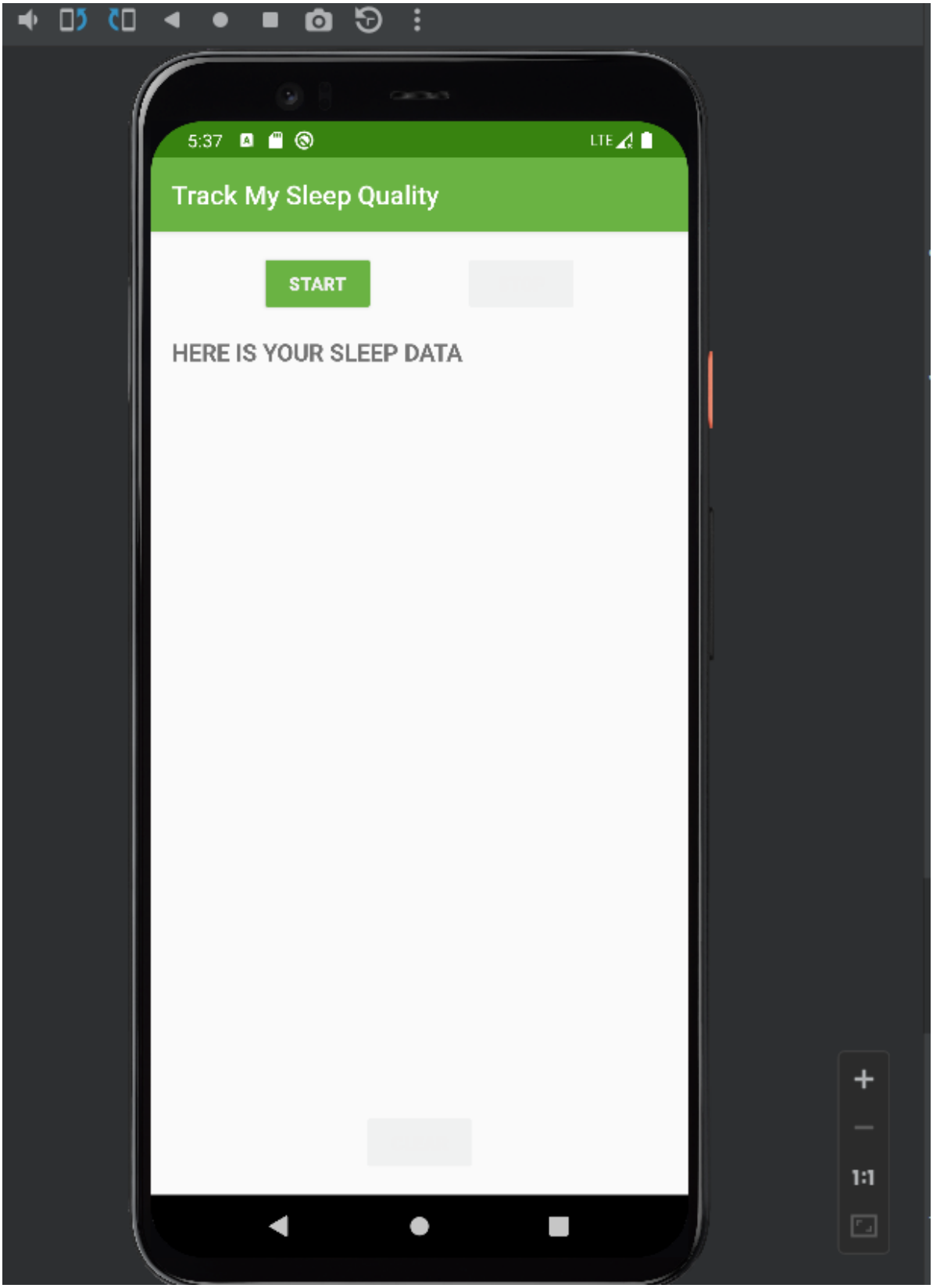
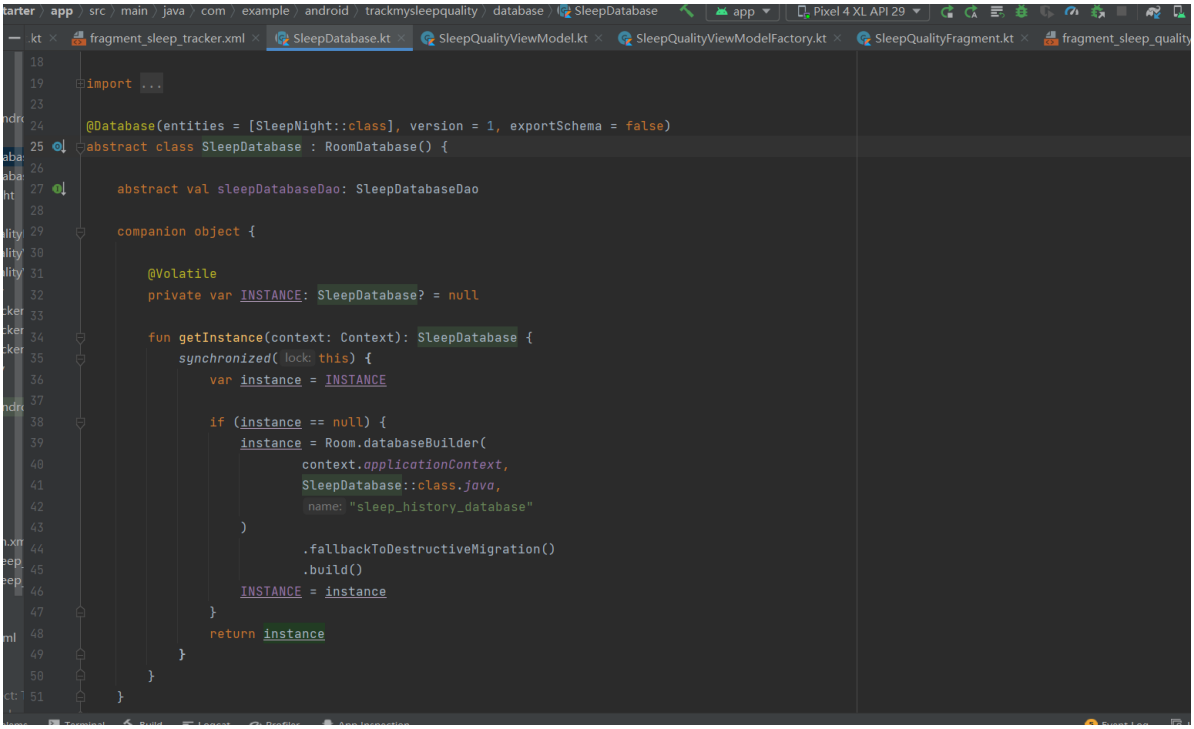

developer.android.com/courses/pathways/android-development-with-kotlin-9#codelab-https://developer.android.com/codelabs/kotlin-android-training-coroutines-and-...
developers
平台
Android Studio
Google Play
Jetpack
Kotlin
文档
新闻
搜索
中文 – 简体
第 9 课：应用架构（持久性）
使用 Room 库创建数据库并使用协程来简化异步编程。
2 项活动 • 1 个测验
第 9 课：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


5:37
LTE
Track My Sleep Quality
START
HERE IS YOUR SLEEP DATA
Start: Friday Apr-15-2022 Time: 17:56
End:Friday Apr-15-2022 Time: 17:56
Quality: Very bad
Hours:Minutes:Seconds 0:0:1
CLEAR
1:1

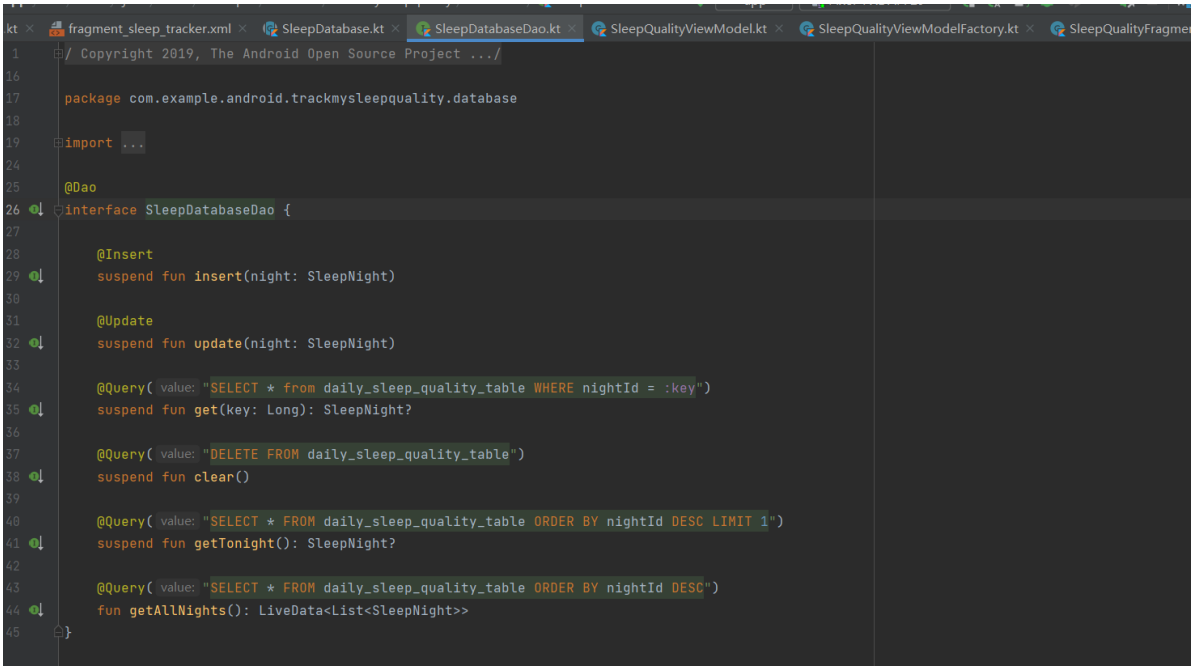

5:37
LTE
Track My Sleep Quality
How was your sleep?
0
1
2
3
4
5


5:37
LTE
Track My Sleep Quality
STOP
HERE IS YOUR SLEEP DATA
Start: Friday Apr-15-2022 Time: 18:13
Start: Friday Apr-15-2022 Time: 18:12
End:Friday Apr-15-2022 Time: 18:12
Quality: So-so
Hours:Minutes:Seconds 0:0:1
CLEAR

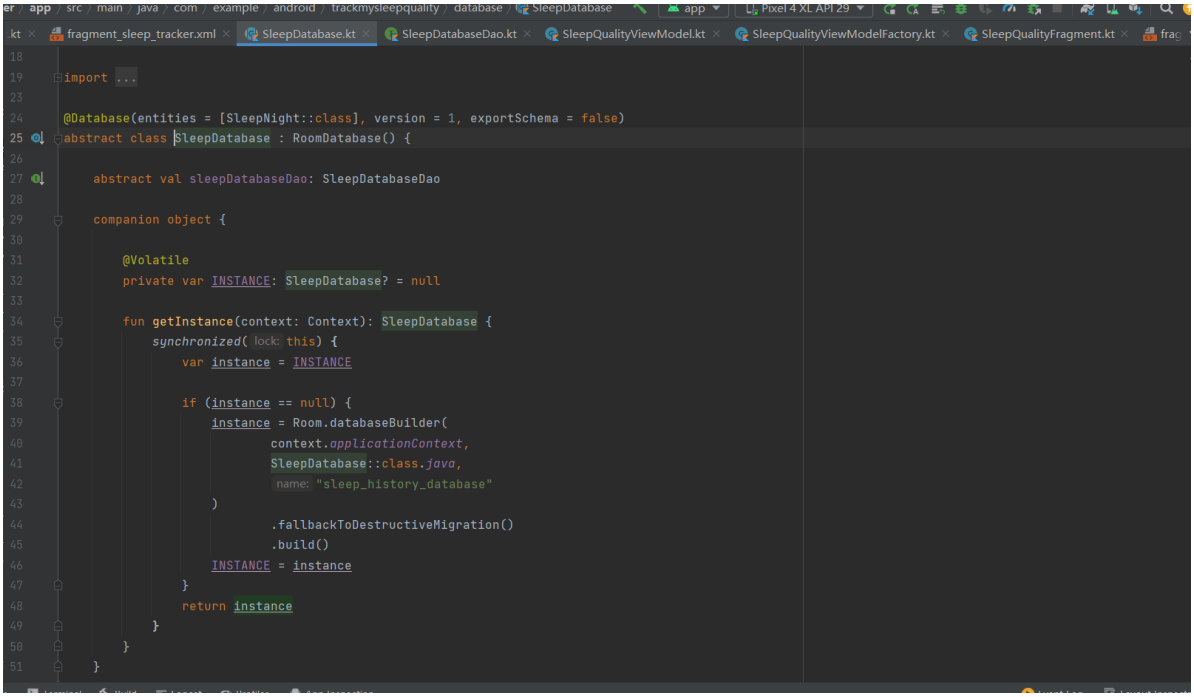

5:37
LTE
Track My Sleep Quality
START
HERE IS YOUR SLEEP DATA
1:1



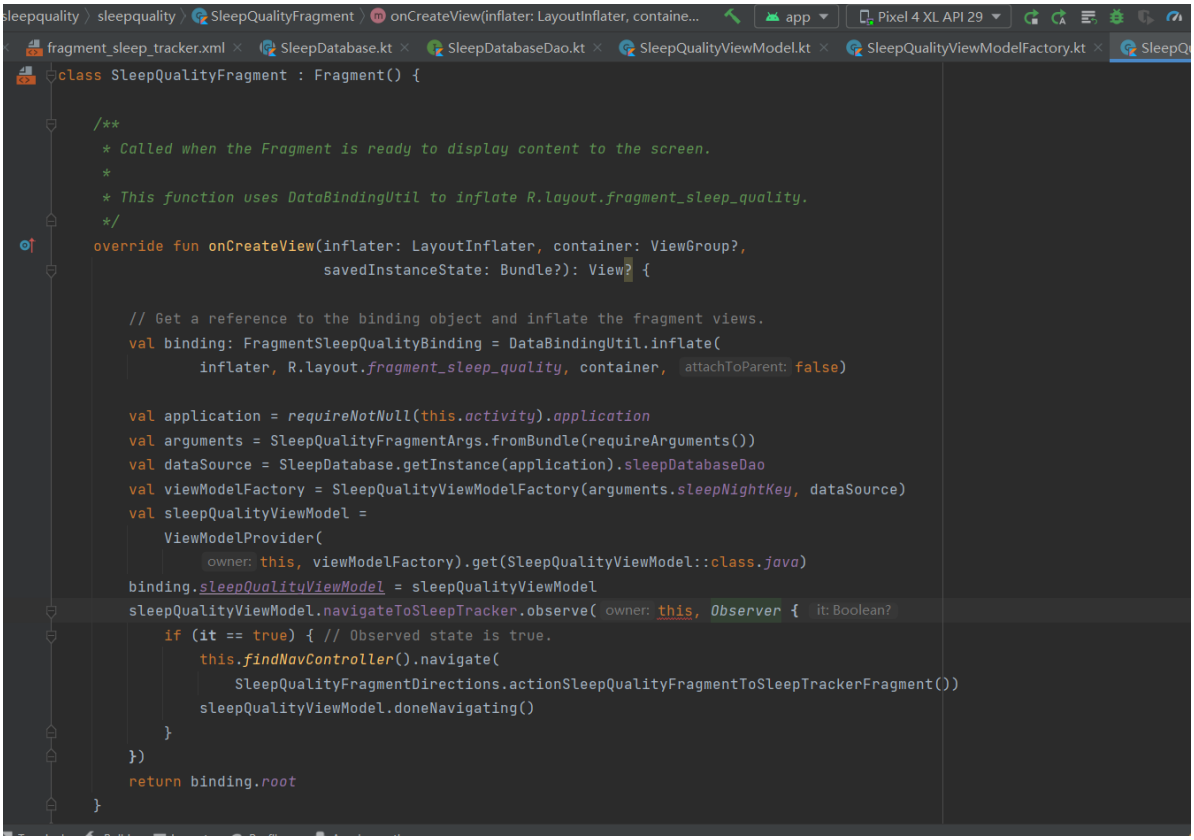
```
import ...

@Database(entities = [SleepNight::class], version = 1, exportSchema = false)
abstract class SleepDatabase : RoomDatabase() {

    abstract val sleepDatabaseDao: SleepDatabaseDao

    companion object {

        @Volatile
        private var INSTANCE: SleepDatabase? = null

        fun getInstance(context: Context): SleepDatabase {
            synchronized( lock: this) {
                var instance = INSTANCE

                if (instance == null) {
                    instance = Room.databaseBuilder(
                            context.applicationContext,
                            SleepDatabase::class.java,
                             name: "sleep_history_database"
                    )
                            .fallbackToDestructiveMigration()
                            .build()
                    INSTANCE = instance
                }
                return instance
            }
        }
    }
```



```
/ Copyright 2019, The Android Open Source Project .../

package com.example.android.trackmysleepquality.database

import ...

@Dao
interface SleepDatabaseDao {

    @Insert
    suspend fun insert(night: SleepNight)

    @Update
    suspend fun update(night: SleepNight)

    @Query( value: "SELECT * from daily_sleep_quality_table WHERE nightId = :key")
    suspend fun get(key: Long): SleepNight?

    @Query( value: "DELETE FROM daily_sleep_quality_table")
    suspend fun clear()

    @Query( value: "SELECT * FROM daily_sleep_quality_table ORDER BY nightId DESC LIMIT 1")
    suspend fun getTonight(): SleepNight?

    @Query( value: "SELECT * FROM daily_sleep_quality_table ORDER BY nightId DESC")
    fun getAllNights(): LiveData<List<SleepNight>>
}
```



```
import ...

@Database(entities = [SleepNight::class], version = 1, exportSchema = false)
abstract class SleepDatabase : RoomDatabase() {

    abstract val sleepDatabaseDao: SleepDatabaseDao

    companion object {

        @Volatile
        private var INSTANCE: SleepDatabase? = null

        fun getInstance(context: Context): SleepDatabase {
            synchronized( lock: this) {
                var instance = INSTANCE

                if (instance == null) {
                    instance = Room.databaseBuilder(
                            context.applicationContext,
                            SleepDatabase::class.java,
                             name: "sleep_history_database"
                    )
                            .fallbackToDestructiveMigration()
                            .build()
                    INSTANCE = instance
                }
                return instance
            }
        }
    }
}
```



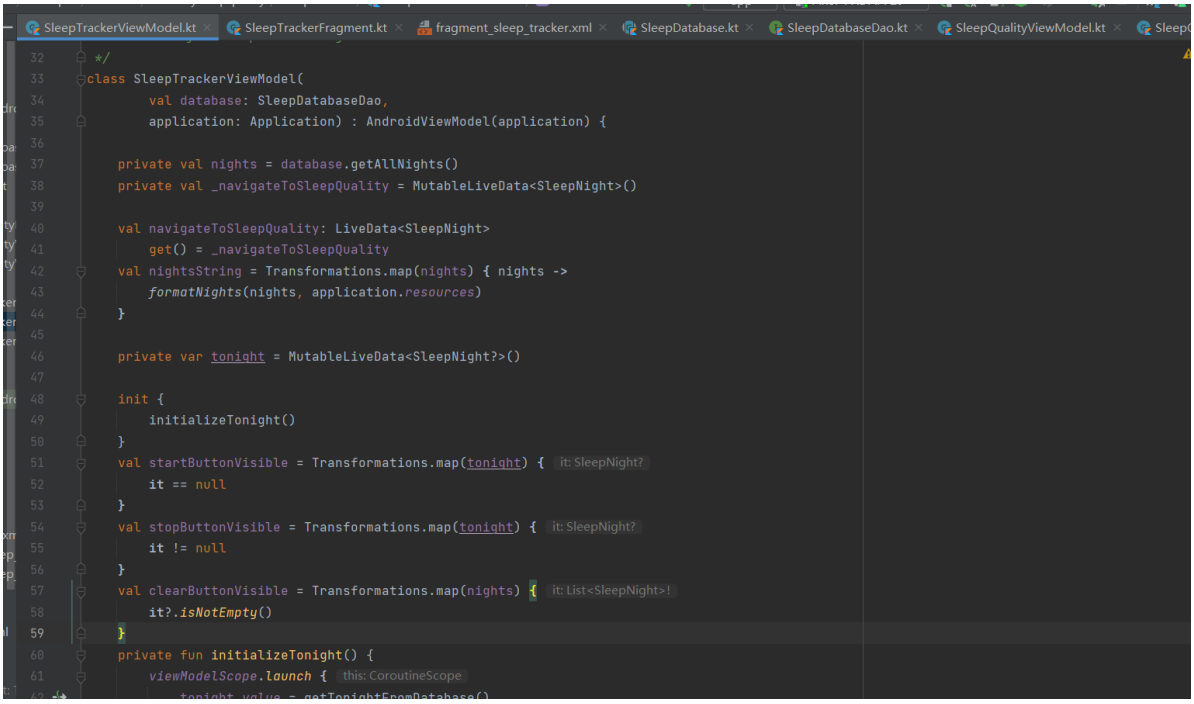
```
class SleepQualityFragment : Fragment() {

    /**
     * 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
    }
```

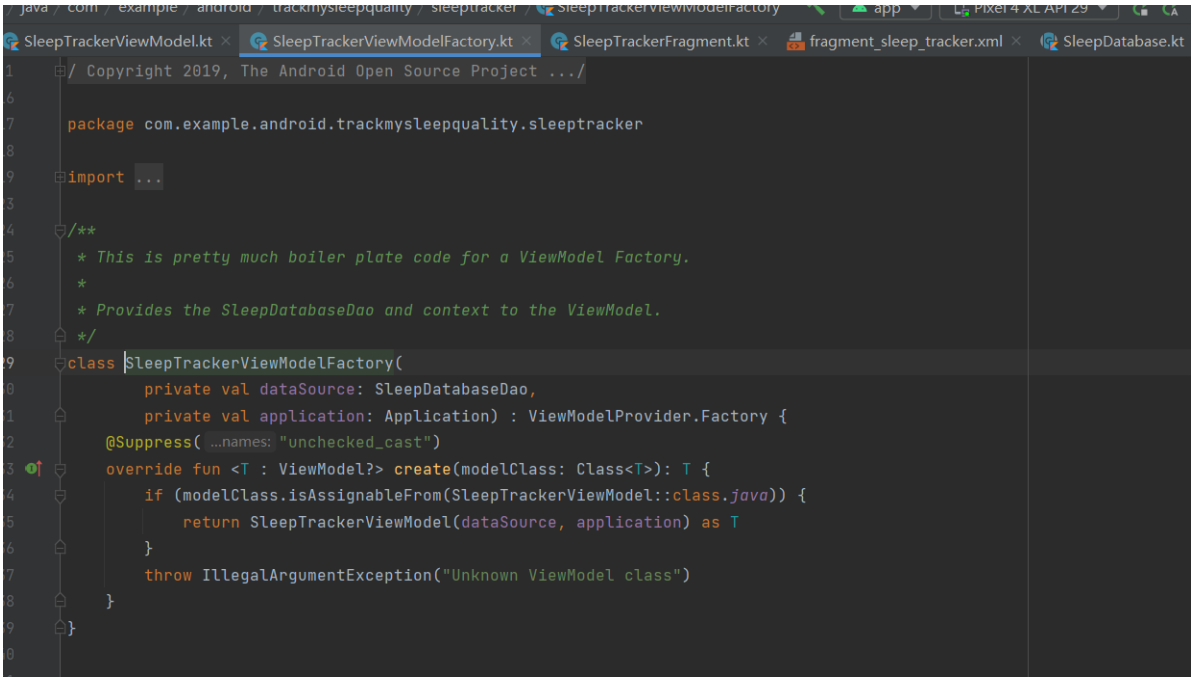
```kotlin
class SleepQualityViewModel(
    private val sleepNightKey: Long = 0L,
    val database: SleepDatabaseDao
) : ViewModel() {
    private val _navigateToSleepTracker = MutableLiveData<Boolean?>()

    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
        }
    }
}
```

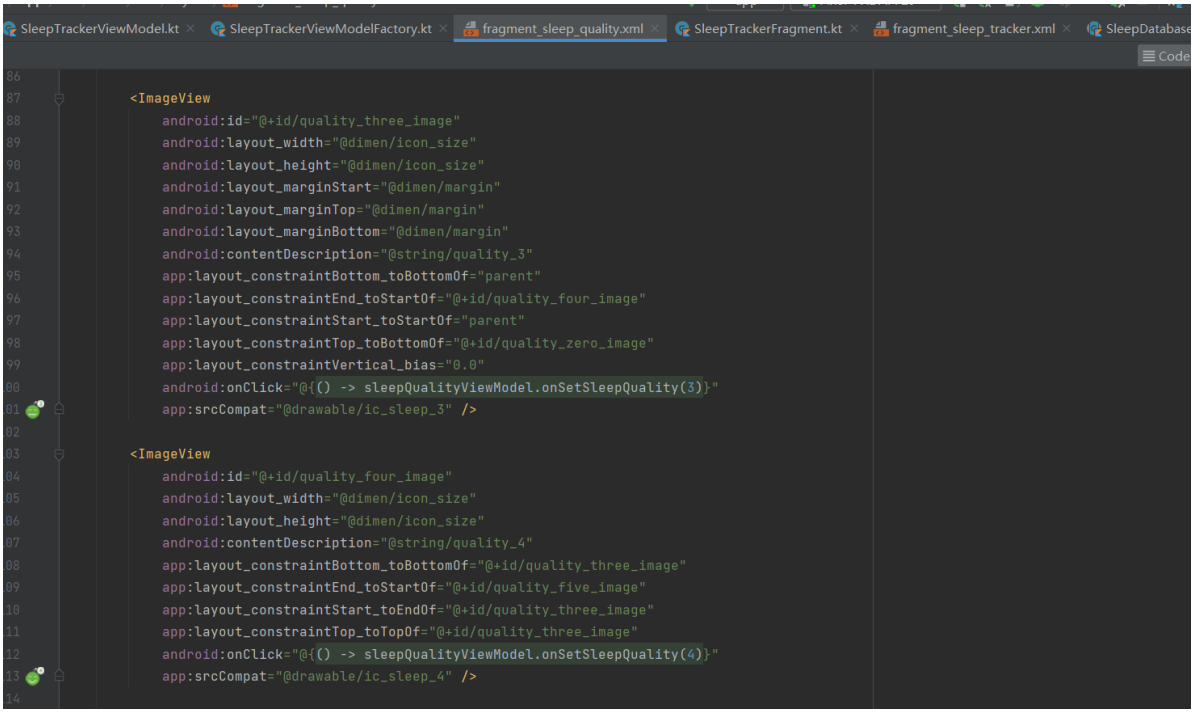
```kotlin
*/

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( ...names: "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")
    }
}
```

SleepTrackerFragment.kt | fragment_sleep_tracker.xml | SleepDatabase.kt | SleepDatabaseDao.kt | SleepQualityViewModel.kt | SleepQualityViewModelFactory.kt | Sleep

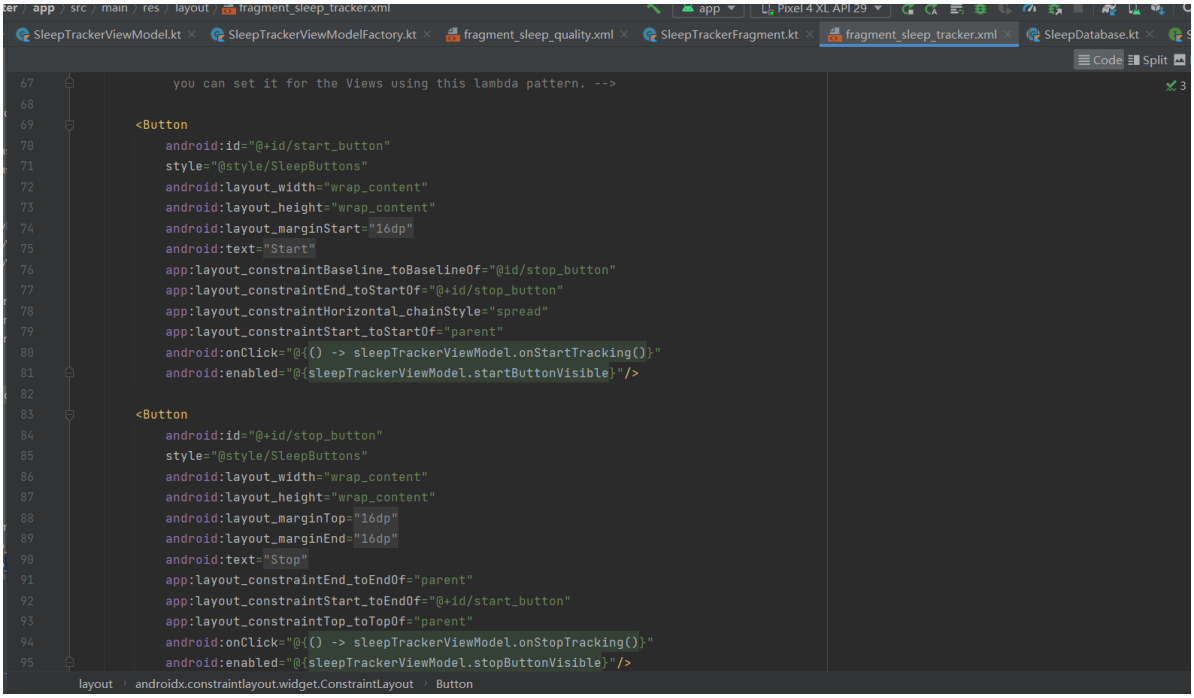
```kotlin
    /**
     * This function uses DataBindingUtil to inflate R.layout.fragment_sleep_quality.
     */
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {

        // Get a reference to the binding object and inflate the fragment views.
        val binding: FragmentSleepTrackerBinding = DataBindingUtil.inflate(
                inflater, R.layout.fragment_sleep_tracker, container,  attachToParent: false)

        val application = requireNotNull(this.activity).application

        val dataSource = SleepDatabase.getInstance(application).sleepDatabaseDao

        val viewModelFactory = SleepTrackerViewModelFactory(dataSource, application)

        val sleepTrackerViewModel =
                ViewModelProvider(
                        owner: this, viewModelFactory).get(SleepTrackerViewModel::class.java)

        binding.setLifecycleOwner(this)

        binding.sleepTrackerViewModel = sleepTrackerViewModel

        sleepTrackerViewModel.navigateToSleepQuality.observe( owner: this, Observer { night ->
            night?.let {  it: SleepNight
                this.findNavController().navigate(
                    SleepTrackerFragmentDirections
                        .actionSleepTrackerFragmentToSleepQualityFragment(night.nightId))
                sleepTrackerViewModel.doneNavigating()
            }
```


SleepTrackerViewModel.kt
SleepTrackerFragment.kt
fragment_sleep_tracker.xml
SleepDatabase.kt
SleepDatabaseDao.kt
SleepQualityViewModel.kt
*/
class SleepTrackerViewModel(
val database: SleepDatabaseDao,
application: Application) : AndroidViewModel(application) {
private val nights = database.getAllNights()
private val _navigateToSleepQuality = MutableLiveData<SleepNight>()
val navigateToSleepQuality: LiveData<SleepNight>
get() = _navigateToSleepQuality
val nightsString = Transformations.map(nights) { nights ->
formatNights(nights, application.resources)
}
private var tonight = MutableLiveData<SleepNight?>()
init {
initializeTonight()
}
val startButtonVisible = Transformations.map(tonight) {
it == null
}
val stopButtonVisible = Transformations.map(tonight) {
it != null
}
val clearButtonVisible = Transformations.map(nights) {
it?.isNotEmpty()
}
private fun initializeTonight() {
viewModelScope.launch {


SleepTrackerViewModel.kt
SleepTrackerViewModelFactory.kt
SleepTrackerFragment.kt
fragment_sleep_tracker.xml
SleepDatabase.kt
/ Copyright 2019, The Android Open Source Project .../
package com.example.android.trackmysleepquality.sleeptracker
import ...
/**
* This is pretty much boiler plate code for a ViewModel Factory.
*
* Provides the SleepDatabaseDao and context to the ViewModel.
*/
class SleepTrackerViewModelFactory(
private val dataSource: SleepDatabaseDao,
private val application: Application) : ViewModelProvider.Factory {
@Suppress("unchecked_cast")
override fun <T : ViewModel?> create(modelClass: Class<T>): T {
if (modelClass.isAssignableFrom(SleepTrackerViewModel::class.java)) {
return SleepTrackerViewModel(dataSource, application) as T
}
throw IllegalArgumentException("Unknown ViewModel class")
}
}




```
<ImageView
    android:id="@+id/quality_three_image"
    android:layout_width="@dimen/icon_size"
    android:layout_height="@dimen/icon_size"
    android:layout_marginStart="@dimen/margin"
    android:layout_marginTop="@dimen/margin"
    android:layout_marginBottom="@dimen/margin"
    android:contentDescription="@string/quality_3"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/quality_four_image"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/quality_zero_image"
    app:layout_constraintVertical_bias="0.0"
    android:onClick="@{() -> sleepQualityViewModel.onSetSleepQuality(3)}"
    app:srcCompat="@drawable/ic_sleep_3" />

<ImageView
    android:id="@+id/quality_four_image"
    android:layout_width="@dimen/icon_size"
    android:layout_height="@dimen/icon_size"
    android:contentDescription="@string/quality_4"
    app:layout_constraintBottom_toBottomOf="@+id/quality_three_image"
    app:layout_constraintEnd_toStartOf="@+id/quality_five_image"
    app:layout_constraintStart_toEndOf="@+id/quality_three_image"
    app:layout_constraintTop_toTopOf="@+id/quality_three_image"
    android:onClick="@{() -> sleepQualityViewModel.onSetSleepQuality(4)}"
    app:srcCompat="@drawable/ic_sleep_4" />
```




```
        you can set it for the Views using this lambda pattern. -->

    <Button
        android:id="@+id/start_button"
        style="@style/SleepButtons"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:text="Start"
        app:layout_constraintBaseline_toBaselineOf="@id/stop_button"
        app:layout_constraintEnd_toStartOf="@+id/stop_button"
        app:layout_constraintHorizontal_chainStyle="spread"
        app:layout_constraintStart_toStartOf="parent"
        android:onClick="@{() -> sleepTrackerViewModel.onStartTracking()}"
        android:enabled="@{sleepTrackerViewModel.startButtonVisible}"/>

    <Button
        android:id="@+id/stop_button"
        style="@style/SleepButtons"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:text="Stop"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/start_button"
        app:layout_constraintTop_toTopOf="parent"
        android:onClick="@{() -> sleepTrackerViewModel.onStopTracking()}"
        android:enabled="@{sleepTrackerViewModel.stopButtonVisible}"/>
```