ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

старший преподаватель				С. Ю. Гуков
должность, уч. степень, звание		подпись, дата		инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4

СУБД и использование сетевых сервисов

Вариант 5

по курсу: Разработка мобильных приложений. Разработка мобильных приложений на Kotlin

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	4128				В. А. Воробьев
			подпись, дата		инициалы, фамилия

Санкт-Петербург 2024

СОДЕРЖАНИЕ

1 Постановка задачи

Выполнить проектирование и разработку мобильного приложения под ОС Android на языке программирования высокого уровня Kotlin.

1.1 Задания работы

1. Задание «БД»

- Создать БД в соответствии с вариантом (предметной областью) – для определенного класса;
- Реализовать добавление данных в БД с помощью отдельного Activity;
- Вывести данные из БД в RecyclerView. Также использовать фрагменты.
- Обработка долгого нажатия на элемент списка;
- Создание диалогового окна для выбора «Просмотр», «Удаление», «Обновление»;
- Обработка нажатия элементов диалогового окна, например, для подтверждения удаления или обновления;
- Реализовать обновление данных в БД с помощью отдельного Activity;
- Реализовать удаление данных из БД с помощью отдельного Activity;
- При работе с БД использовать библиотеку Room.

2. Задание «JSON»

- Скачать JSON из интернета (HttpURLConnection/Retrofit);
- Распарсить JSON;
- Использовать Thread для работы с JSON.

2 Выполнение работы

Для начала были подключены нужные зависимости для проекта в `build.gradle.kts` уровня `app`.

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.jetbrains.kotlin.android)
    alias(libs.plugins.kapt)
    alias(libs.plugins.hilt)
}

android {
    namespace = "com.vladcto.lazymeter"
    compileSdk = 34

    defaultConfig {
        applicationId = "com.vladcto.lazymeter"
        minSdk = 24
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.
            AndroidJUnitRunner"
        vectorDrawables {
            useSupportLibrary = true
        }
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-
                    optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8
```

```
36          targetCompatibility = JavaVersion.VERSION_1_8
37      }
38      kotlinOptions {
39          jvmTarget = "1.8"
40  //          allWarningsAsErrors = false
41          freeCompilerArgs += listOf(
42              "-opt-in=androidx.compose.material3.
                    ExperimentalMaterial3Api"
43          )
44      }
45      buildFeatures {
46          compose = true
47      }
48      composeOptions {
49          kotlinCompilerExtensionVersion = "1.5.12"
50      }
51      packaging {
52          resources {
53              excludes += "/META-INF/{AL2.0,LGPL2.1}"
54          }
55      }
56  }
57
58  kapt {
59      correctErrorTypes = false
60  }
61
62  dependencies {
63
64      implementation(libs.kotlin.stdlib)
65      implementation(libs.androidx.core.ktx)
66      implementation(libs.androidx.lifecycle.runtime.ktx)
67      implementation(libs.androidx.activity.compose)
68      implementation(platform(libs.androidx.compose.bom))
69      implementation(libs.androidx.ui)
70      implementation(libs.androidx.ui.graphics)
71      implementation(libs.androidx.ui.tooling.preview)
72      implementation(libs.androidx.material3)
73      implementation(libs.androidx.lifecycle.viewmodel.compose)
74
75      implementation(libs.androidx.room.runtime)
```

```
76      implementation(libs.androidx.room.ktx)
77      kapt(libs.androidx.room.compiler)
78
79      implementation(libs.hilt.android)
80      kapt(libs.hilt.android.compiler)
81
82      implementation(libs.retrofit)
83      implementation(libs.retrofit.converter.gson)
84  }
```

Затем мы приступили к написанию `data-слоя` приложения.

Для хранения данных в локальной БД была выбрана библиотека `Room`, а для работы с сетевыми запросами - `Retrofit`. Вся работы ведется асинхронно, с помощью **корутин**. Выбор в сторону корутин, обусловен их поддержкой со стороны **Kotlin**. Итоговый код доступен в Приложении.

Для DI была выбрана библиотека `Hilt`, с помощью который мы обеспечили внедрение зависимостей через конструктор. Код модуля представлен ниже:

```
1   package com.vladcto.lazymeter.platform.di
2
3   import android.content.Context
4   import androidx.room.Room
5   import com.vladcto.lazymeter.data.lazy.infra.LazyUnitDao
6   import com.vladcto.lazymeter.platform.room.AppDatabase
7   import com.vladcto.lazymeter.platform.room.converter.
        RoomDateLongConverter
8   import dagger.Module
9   import dagger.Provides
10  import dagger.hilt.InstallIn
11  import dagger.hilt.android.components.ViewModelComponent
12  import dagger.hilt.android.qualifiers.ApplicationContext
13
14  @Module
15  @InstallIn(ViewModelComponent::class)
16  class AppModule {
17      @Provides
18      fun provideAppDatabase(@ApplicationContext appContext:
            Context): AppDatabase {
19          return Room.databaseBuilder(
20              context = appContext,
```

```
21                 AppDatabase::class.java, "lazymeter-db",
22             ).build()
23         }
24
25         @Provides
26         fun provideUserDao(appDatabase: AppDatabase): LazyUnitDao
               = appDatabase.lazyUnitDao()
27     }
```

Затем мы приступили к написанию UI части приложения. Для ее написания мы выбрали фреймворк `Jetpack Compose` и библиотеку для работы с `ViewModel`.

Исходный код доступен в Приложении. Здесь мы покажем следующий функционал:

- Начальный экран.
- Отправку данных на сервер.
- Добавление данных в БД.
- Считывание данных из БД.
- Удаление записей из БД.
- Скролл списка.

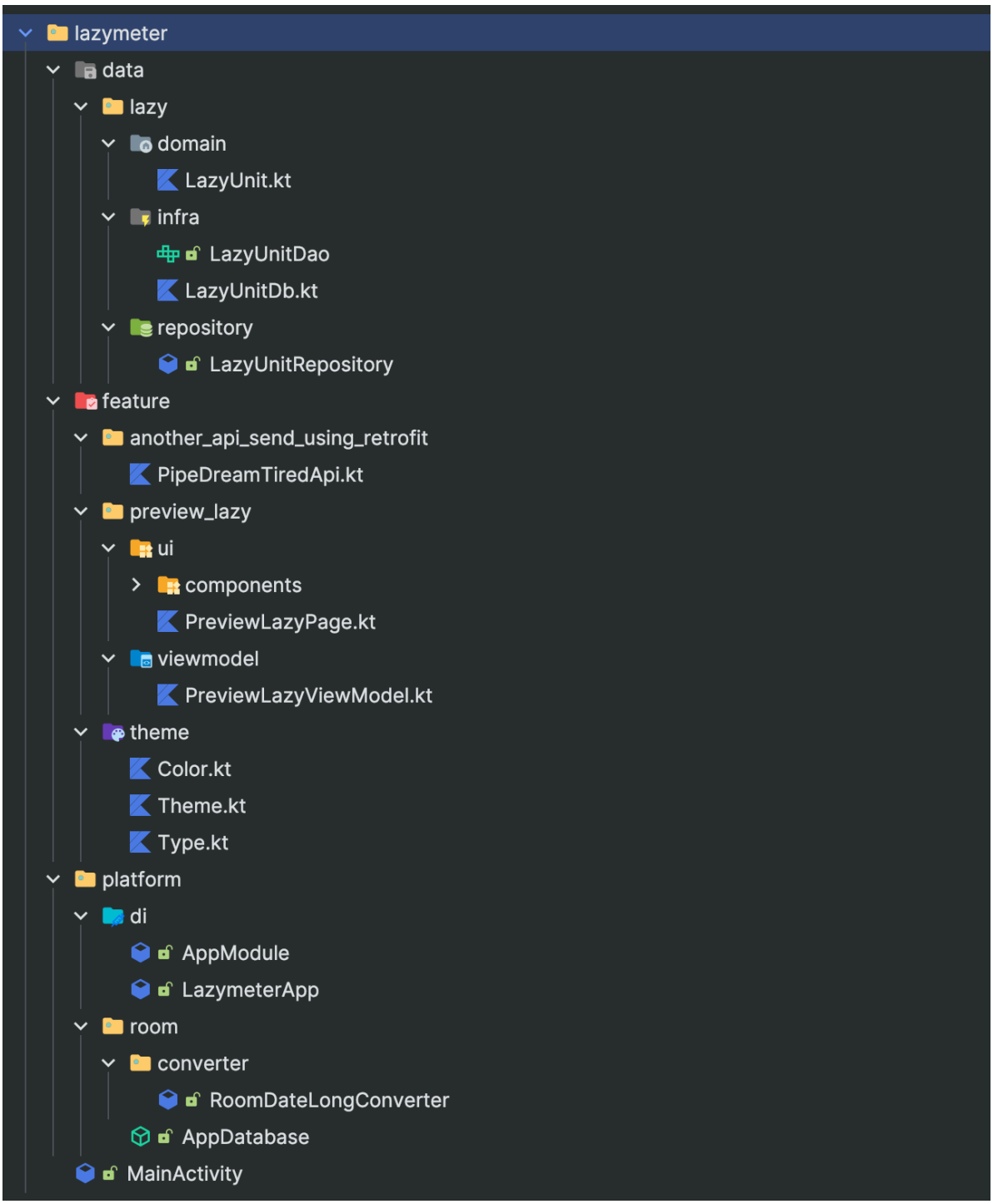

Рисунок 2.1 - Итоговая структура проекта

Рисунок 2.2 - Начальный экран

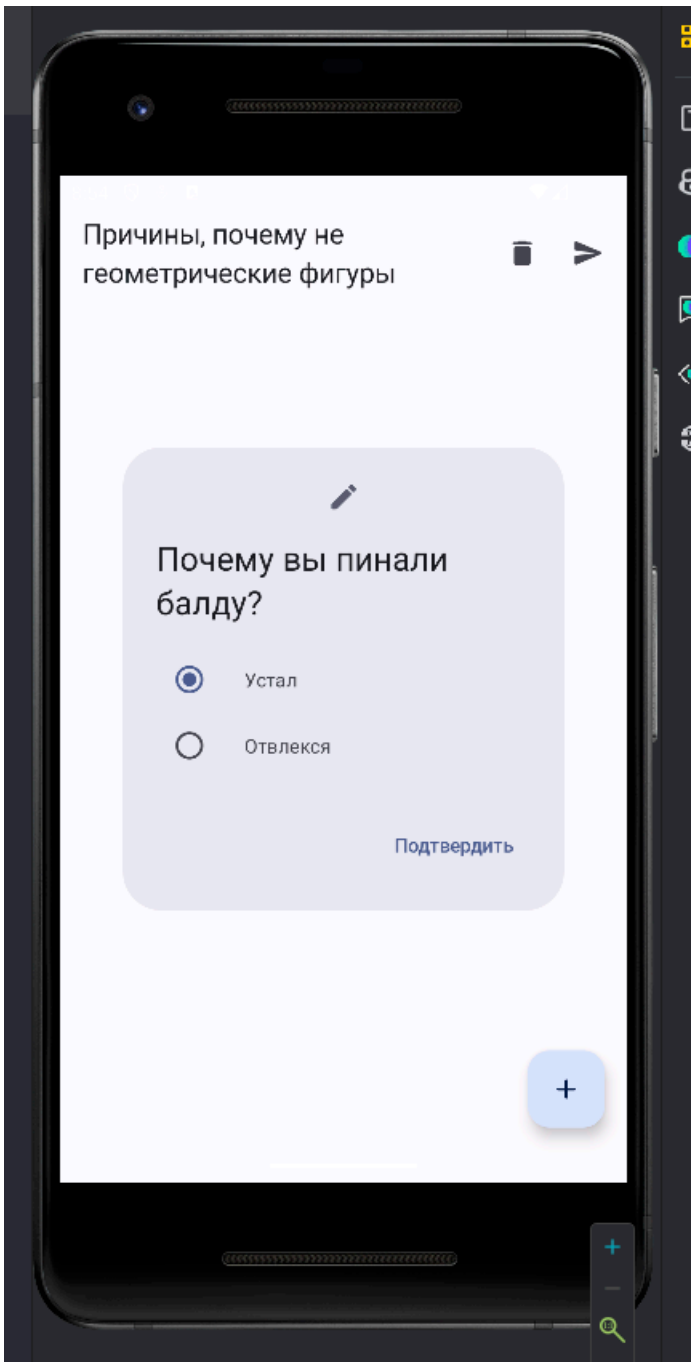

Рисунок 2.3 - Диалог создания обьекта

Рисунок 2.4 - Добавление обьекта в список

Рисунок 2.5 - Холодный перезапуск

Рисунок 2.6 - Удаление всех записей из БД

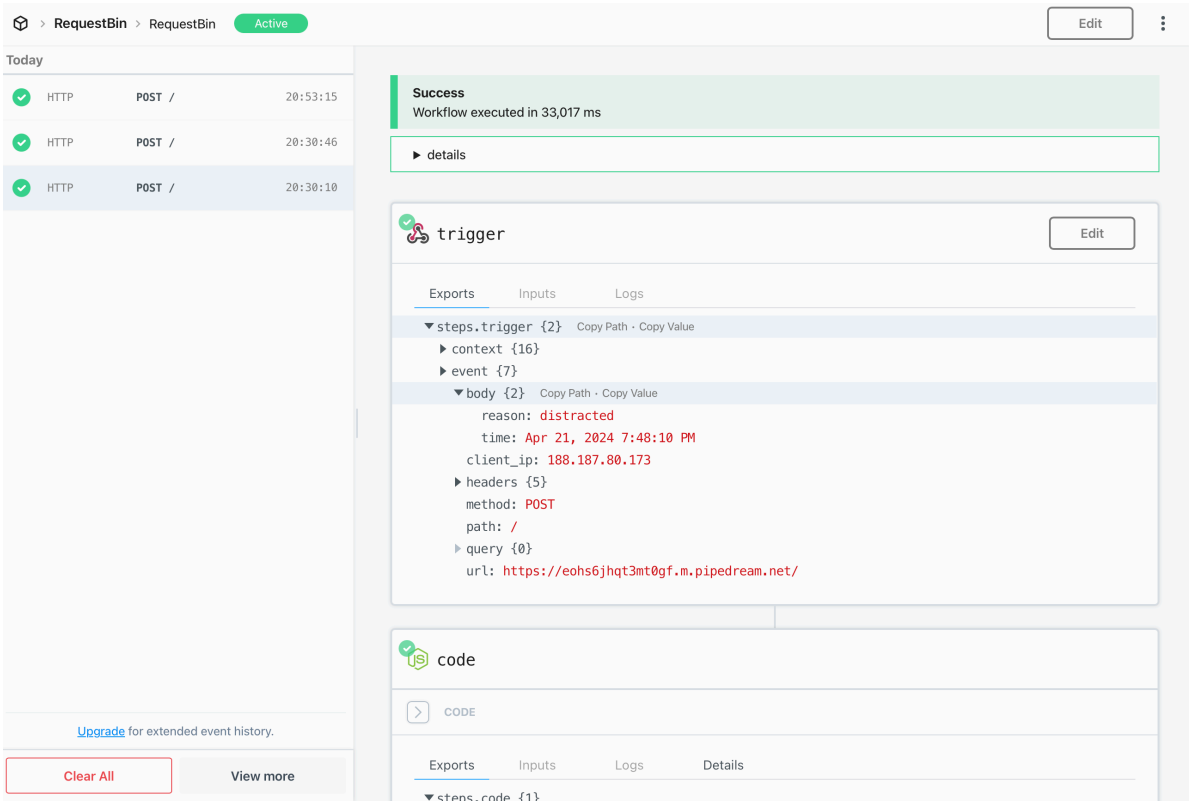

Рисунок 2.7 - Отправка данных на сервер

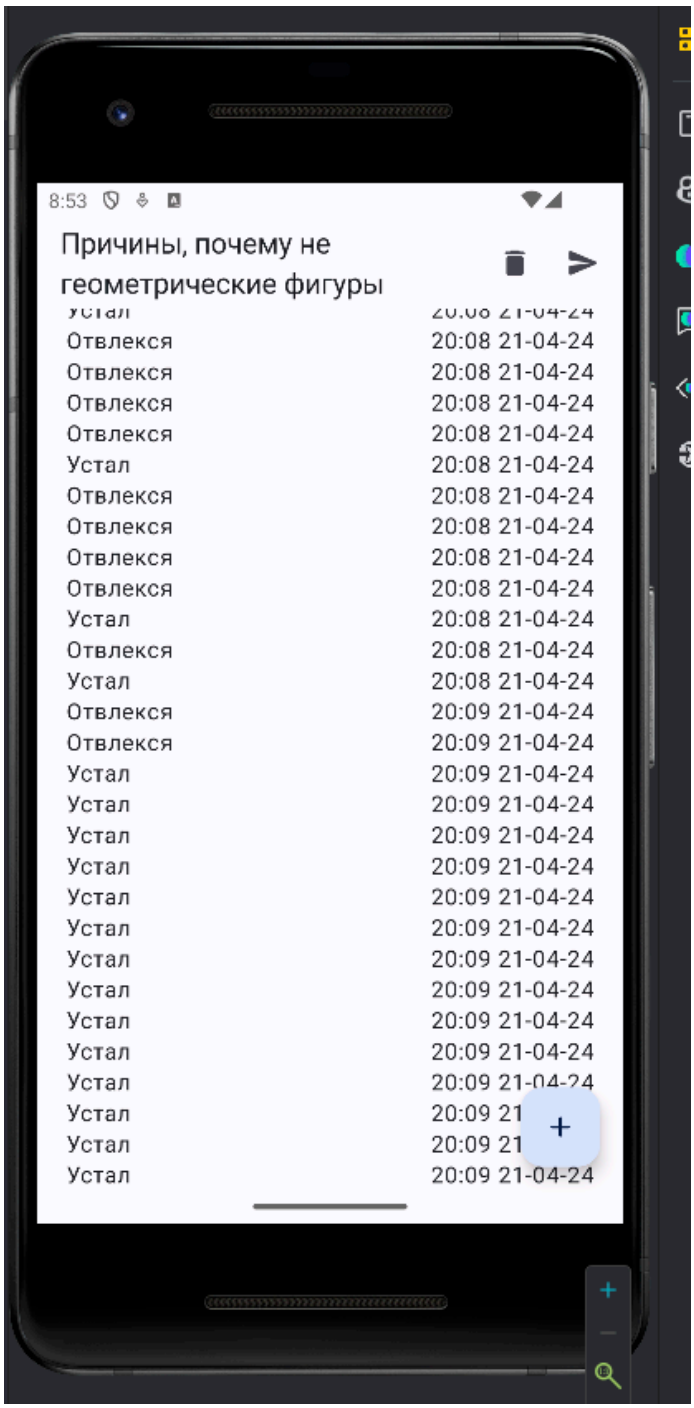

Рисунок 2.8 - Скролл списка

3 Вывод

В результате выполнения лабораторной работы было успешно разработано мобильное приложение под ОС Android на языке программирования Kotlin с использованием фреймворка Jetpack Compose. Приложение включает в себя функционал работы с локальным хранилищем, а именно работа с БД посредством ORM `Room` и работа с сетевыми запросами с помощью `Retrofit`.

Выполненная лабораторная работа способствовала углублению знаний и навыков в области разработки мобильных приложений.

ПРИЛОЖЕНИЕ

```
1  MainActivity.kt
2  // This is a sketch for a course project, so it looks so...
       empty?
3  package com.vladcto.lazymeter
4
5  import android.os.Bundle
6  import androidx.activity.ComponentActivity
7  import androidx.activity.compose.setContent
8  import androidx.activity.enableEdgeToEdge
9  import com.vladcto.lazymeter.feature.preview_lazy.ui.
       LazyPreviewPage
10 import com.vladcto.lazymeter.feature.theme.LazymeterTheme
11 import dagger.hilt.android.AndroidEntryPoint
12
13 @AndroidEntryPoint
14 class MainActivity : ComponentActivity() {
15     override fun onCreate(savedInstanceState: Bundle?) {
16         super.onCreate(savedInstanceState)
17         enableEdgeToEdge()
18         setContent {
19             LazymeterTheme {
20                 LazyPreviewPage()
21             }
22         }
23     }
24 }
25 AppModule.kt
26 package com.vladcto.lazymeter.platform.di
27
28 import android.content.Context
29 import androidx.room.Room
30 import com.vladcto.lazymeter.data.lazy.infra.LazyUnitDao
31 import com.vladcto.lazymeter.platform.room.AppDatabase
32 import com.vladcto.lazymeter.platform.room.converter.
       RoomDateLongConverter
33 import dagger.Module
34 import dagger.Provides
35 import dagger.hilt.InstallIn
36 import dagger.hilt.android.components.ViewModelComponent
37 import dagger.hilt.android.qualifiers.ApplicationContext
```

```
38
39 @Module
40 @InstallIn(ViewModelComponent::class)
41 class AppModule {
42     @Provides
43     fun provideAppDatabase(@ApplicationContext appContext:
       Context): AppDatabase {
44         return Room.databaseBuilder(
45             context = appContext,
46             AppDatabase::class.java, "lazymeter-db",
47         ).build()
48     }
49
50     @Provides
51     fun provideUserDao(appDatabase: AppDatabase): LazyUnitDao
       = appDatabase.lazyUnitDao()
52 }
53 LazymeterApp.kt
54 package com.vladcto.lazymeter.platform.di
55
56 import android.app.Application
57 import androidx.room.Room
58 import com.vladcto.lazymeter.platform.room.AppDatabase
59 import dagger.Provides
60 import dagger.hilt.android.HiltAndroidApp
61
62 @HiltAndroidApp
63 class LazymeterApp : Application()
64 RoomDateLongConverter.kt
65 package com.vladcto.lazymeter.platform.room.converter
66
67 import androidx.room.TypeConverter
68 import java.util.Date
69
70 class RoomDateLongConverter {
71     @TypeConverter
72     fun dateToLong(date: Date?): Long? = date?.time
73
74     @TypeConverter
75     fun longToDate(value: Long?): Date? = value?.let { Date(
       it) }
```

```
}
AppDatabase.kt
package com.vladcto.lazymeter.platform.room

import androidx.room.Database
import androidx.room.RoomDatabase
import androidx.room.TypeConverters
import com.vladcto.lazymeter.data.lazy.infra.LazyUnitDao
import com.vladcto.lazymeter.data.lazy.infra.LazyUnitDb
import com.vladcto.lazymeter.platform.room.converter.
    RoomDateLongConverter

@Database(
    entities = [LazyUnitDb::class],
    version = 1
)
@TypeConverters(RoomDateLongConverter::class)
abstract class AppDatabase : RoomDatabase() {
    abstract fun lazyUnitDao(): LazyUnitDao
}
PipeDreamTiredApi.kt
package com.vladcto.lazymeter.feature.
    another_api_send_using_retrofit

import com.vladcto.lazymeter.data.lazy.domain.LazyUnit
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
import retrofit2.http.Body
import retrofit2.http.POST

// I sketched singletons out of bored.

interface LazyApi {
    @POST(".")
    suspend fun sendLazyUnit(@Body lazyUnit: LazyUnit)
}

object PipeDreamRepository {
    private const val BASE_URL = "https://eohs6jhqt3mt0gf.m.
        pipedream.net/"

```

```
114     private val retrofit: Retrofit = Retrofit.Builder()
115         .baseUrl(BASE_URL)
116         .addConverterFactory(GsonConverterFactory.create())
117         .build()
118
119     private val lazyApi: LazyApi = retrofit.create(LazyApi::
            class.java)
120
121     suspend fun sendLazyUnit(lazyUnit: LazyUnit) {
122         try {
123             lazyApi.sendLazyUnit(lazyUnit)
124         } catch (_: Throwable) {
125         }
126     }
127 }
128 Color.kt
129 package com.vladcto.lazymeter.feature.theme
130
131 import androidx.compose.ui.graphics.Color
132
133 val Purple80 = Color(0xFFD0BCFF)
134 val PurpleGrey80 = Color(0xFFCCC2DC)
135 val Pink80 = Color(0xFFEFB8C8)
136
137 val Purple40 = Color(0xFF6650a4)
138 val PurpleGrey40 = Color(0xFF625b71)
139 val Pink40 = Color(0xFF7D5260)
140 Theme.kt
141 package com.vladcto.lazymeter.feature.theme
142
143 import android.app.Activity
144 import android.os.Build
145 import androidx.compose.foundation.isSystemInDarkTheme
146 import androidx.compose.material3.MaterialTheme
147 import androidx.compose.material3.darkColorScheme
148 import androidx.compose.material3.dynamicDarkColorScheme
149 import androidx.compose.material3.dynamicLightColorScheme
150 import androidx.compose.material3.lightColorScheme
151 import androidx.compose.runtime.Composable
152 import androidx.compose.ui.platform.LocalContext
153
```

```
154 private val DarkColorScheme = darkColorScheme(
155     primary = Purple80,
156     secondary = PurpleGrey80,
157     tertiary = Pink80
158 )
159
160 private val LightColorScheme = lightColorScheme(
161     primary = Purple40,
162     secondary = PurpleGrey40,
163     tertiary = Pink40
164
165     /* Other default colors to override
166     background = Color(0xFFFFFBFE),
167     surface = Color(0xFFFFFBFE),
168     onPrimary = Color.White,
169     onSecondary = Color.White,
170     onTertiary = Color.White,
171     onBackground = Color(0xFF1C1B1F),
172     onSurface = Color(0xFF1C1B1F),
173     */
174 )
175
176 @Composable
177 fun LazymeterTheme(
178     darkTheme: Boolean = isSystemInDarkTheme(),
179     // Dynamic color is available on Android 12+
180     dynamicColor: Boolean = true,
181     content: @Composable () -> Unit
182 ) {
183     val colorScheme = when {
184         dynamicColor && Build.VERSION.SDK_INT >= Build.
            VERSION_CODES.S -> {
185             val context = LocalContext.current
186             if (darkTheme) dynamicDarkColorScheme(context)
                else dynamicLightColorScheme(context)
187         }
188
189         darkTheme -> DarkColorScheme
190         else -> LightColorScheme
191     }
192
```

```
193     MaterialTheme(
194         colorScheme = colorScheme,
195         typography = Typography,
196         content = content
197     )
198 }
199 Type.kt
200 package com.vladcto.lazymeter.feature.theme
201
202 import androidx.compose.material3.Typography
203 import androidx.compose.ui.text.TextStyle
204 import androidx.compose.ui.text.font.FontFamily
205 import androidx.compose.ui.text.font.FontWeight
206 import androidx.compose.ui.unit.sp
207
208 // Set of Material typography styles to start with
209 val Typography = Typography(
210     bodyLarge = TextStyle(
211         fontFamily = FontFamily.Default,
212         fontWeight = FontWeight.Normal,
213         fontSize = 16.sp,
214         lineHeight = 24.sp,
215         letterSpacing = 0.5.sp
216     )
217     /* Other default text styles to override
218     titleLarge = TextStyle(
219         fontFamily = FontFamily.Default,
220         fontWeight = FontWeight.Normal,
221         fontSize = 22.sp,
222         lineHeight = 28.sp,
223         letterSpacing = 0.sp
224     ),
225     labelSmall = TextStyle(
226         fontFamily = FontFamily.Default,
227         fontWeight = FontWeight.Medium,
228         fontSize = 11.sp,
229         lineHeight = 16.sp,
230         letterSpacing = 0.5.sp
231     )
232     */
233 )
```

```
234 PreviewLazyViewModel.kt
235 package com.vladcto.lazymeter.feature.preview_lazy.viewmodel
236
237 import androidx.lifecycle.ViewModel
238 import androidx.lifecycle.viewModelScope
239 import com.vladcto.lazymeter.data.lazy.domain.LazyUnit
240 import com.vladcto.lazymeter.data.lazy.repository.
       LazyUnitRepository
241 import com.vladcto.lazymeter.feature.
       another_api_send_using_retrofit.PipeDreamRepository
242 import dagger.hilt.android.lifecycle.HiltViewModel
243 import kotlinx.coroutines.Dispatchers
244 import kotlinx.coroutines.async
245 import kotlinx.coroutines.flow.MutableStateFlow
246 import kotlinx.coroutines.flow.asStateFlow
247 import kotlinx.coroutines.flow.update
248 import kotlinx.coroutines.launch
249 import javax.inject.Inject
250
251 data class PreviewLazyState(
252     val lazyUnits: List<LazyUnit>,
253 )
254
255 @HiltViewModel
256 class PreviewLazyViewModel @Inject constructor(
257     private val _lazyUnitRepository: LazyUnitRepository,
258 ) : ViewModel() {
259
260     init {
261         viewModelScope.launch {
262             val result = async {
263                 _lazyUnitRepository.getAll()
264             }.await()
265             _previewState.update { _ -> PreviewLazyState(
                    result) }
266         }
267     }
268
269     private val _previewState = MutableStateFlow(
270         PreviewLazyState(listOf())
271     )
```

```
272     val previewState = _previewState.asStateFlow()
273
274     fun addLazyUnit(unit: LazyUnit) {
275         viewModelScope.launch {
276             async { _lazyUnitRepository.add(unit) }.await()
277             _previewState.update { currentState ->
278                 PreviewLazyState(currentState.lazyUnits +
                        unit)
279             }
280         }
281     }
282
283     fun sendLazyUnit(unit: LazyUnit) {
284         viewModelScope.launch(context = Dispatchers.IO) {
285             PipeDreamRepository.sendLazyUnit(unit)
286         }
287     }
288
289     fun clear() = viewModelScope.launch {
290         _lazyUnitRepository.clear()
291         _previewState.update { _ -> PreviewLazyState(listOf()
            ) }
292     }
293
294 }
295 CreateLazyUnitDialog.kt
296 package com.vladcto.lazymeter.feature.preview_lazy.ui.
       components
297
298 import androidx.compose.foundation.layout.Column
299 import androidx.compose.foundation.layout.Row
300 import androidx.compose.foundation.layout.padding
301 import androidx.compose.foundation.selection.selectableGroup
302 import androidx.compose.material.icons.Icons
303 import androidx.compose.material.icons.rounded.Edit
304 import androidx.compose.material3.AlertDialog
305 import androidx.compose.material3.Icon
306 import androidx.compose.material3.RadioButton
307 import androidx.compose.material3.Text
308 import androidx.compose.material3.TextButton
309 import androidx.compose.runtime.Composable
```

```
310 import androidx.compose.runtime.mutableStateOf
311 import androidx.compose.runtime.remember
312 import androidx.compose.ui.Alignment
313 import androidx.compose.ui.Modifier
314 import androidx.compose.ui.unit.dp
315 import com.vladcto.lazymeter.data.lazy.domain.LazyReason
316 import com.vladcto.lazymeter.data.lazy.domain.LazyUnit
317 import java.util.Date
318
319 @Composable
320 fun CreateLazyUnitDialog(onDismissRequest: () -> Unit,
        onComplete: (LazyUnit) -> Unit) {
321     val choiceTired = remember { mutableStateOf(true) }
322     AlertDialog(
323         onDismissRequest = onDismissRequest,
324         icon = {
325             Icon(
326                 Icons.Rounded.Edit,
327                 contentDescription = ""
328             )
329         },
330         title = {
331             Text(text = "Почему вы пинали балду?")
332         },
333         text = {
334             Column(modifier = Modifier.selectableGroup()) {
335                 Row(
336                     verticalAlignment = Alignment.
                        CenterVertically
337                 ) {
338                     RadioButton(
339                         selected = choiceTired.value,
340                         onClick = { choiceTired.value = true
                            })
341                     Text(text = "Устал", modifier = Modifier.
                        padding(start = 16.dp))
342
343                 }
344                 Row(
345                     verticalAlignment = Alignment.
                        CenterVertically
```

```
346                 ) {
347                     RadioButton(
348                         selected = !choiceTired.value,
349                         onClick = { choiceTired.value = false
                            })
350                     Text(text = "Отвлекся", modifier =
                        Modifier.padding(start = 16.dp))
351                 }
352             }
353         },
354         confirmButton = {
355             TextButton(
356                 onClick = {
357                     onComplete(
358                         LazyUnit(
359                             time = Date(),
360                             reason = LazyReason.Tired.takeIf
                                { choiceTired.value }
361                                 ?: LazyReason.Distracted
362                         )
363                     )
364                     onDismissRequest()
365                 }
366             ) {
367                 Text(text = "Подтвердить")
368             }
369         },
370     )
371 }
372 LazyUnitCard.kt
373 package com.vladcto.lazymeter.feature.preview_lazy.ui.
       components
374
375 import androidx.compose.foundation.layout.Row
376 import androidx.compose.foundation.layout.fillMaxWidth
377 import androidx.compose.material3.Text
378 import androidx.compose.runtime.Composable
379 import androidx.compose.ui.Modifier
380 import androidx.compose.ui.platform.LocalConfiguration
381 import androidx.compose.ui.text.style.TextAlign
382 import com.vladcto.lazymeter.data.lazy.domain.LazyReason
```

```
383 import com.vladcto.lazymeter.data.lazy.domain.LazyUnit
384 import java.text.SimpleDateFormat
385
386 private const val DATE_PATTERN = "HH:mm dd-MM-yy"
387
388 @Composable
389 fun LazyUnitCard(lazyUnit: LazyUnit, modifier: Modifier) {
390     val locale = LocalConfiguration.current.locales[0]
391     val dateFormat = SimpleDateFormat(DATE_PATTERN, locale)
392     Row(
393         modifier = modifier.fillMaxWidth()
394     ) {
395         Text(
396             text = when (lazyUnit.reason) {
397                 LazyReason.Tired -> "Устал"
398                 LazyReason.Distracted -> "Отвлекся"
399             }
400         )
401         Text(
402             modifier = Modifier.fillMaxWidth(),
403             textAlign = TextAlign.End,
404             text = dateFormat.format(lazyUnit.time)
405         )
406     }
407 }
408 PreviewLazyPage.kt
409 package com.vladcto.lazymeter.feature.preview_lazy.ui
410
411 import androidx.compose.foundation.layout.fillMaxHeight
412 import androidx.compose.foundation.layout.padding
413 import androidx.compose.foundation.lazy.LazyColumn
414 import androidx.compose.foundation.lazy.items
415 import androidx.compose.material.icons.Icons
416 import androidx.compose.material.icons.rounded.Add
417 import androidx.compose.material.icons.rounded.Delete
418 import androidx.compose.material.icons.rounded.Send
419 import androidx.compose.material3.FloatingActionButton
420 import androidx.compose.material3.Icon
421 import androidx.compose.material3.IconButton
422 import androidx.compose.material3.Scaffold
423 import androidx.compose.material3.Text
```

```
424 import androidx.compose.material3.TopAppBar
425 import androidx.compose.runtime.Composable
426 import androidx.compose.runtime.collectAsState
427 import androidx.compose.runtime.getValue
428 import androidx.compose.runtime.mutableStateOf
429 import androidx.compose.runtime.remember
430 import androidx.compose.ui.Modifier
431 import androidx.compose.ui.unit.dp
432 import androidx.compose.ui.unit.sp
433 import androidx.lifecycle.viewmodel.compose.viewModel
434 import com.vladcto.lazymeter.feature.preview_lazy.ui.
       components.CreateLazyUnitDialog
435 import com.vladcto.lazymeter.feature.preview_lazy.ui.
       components.LazyUnitCard
436 import com.vladcto.lazymeter.feature.preview_lazy.viewmodel.
       PreviewLazyViewModel
437
438 @Composable
439 fun LazyPreviewPage(
440     previewLazyViewModel: PreviewLazyViewModel = viewModel()
441 ) {
442     val lazyUnitData by previewLazyViewModel.previewState.
          collectAsState()
443     val creationDialog = remember { mutableStateOf(false) }
444     Scaffold(
445         topBar = {
446             TopAppBar(
447                 title = {
448                     Text(
449                         "Причины, почему не геометрические
                              фигуры",
450                         fontSize = 20.sp
451                     )
452                 },
453                 actions = {
454                     IconButton(
455                         onClick = { previewLazyViewModel.
                              clear() }) {
456                         Icon(
457                             Icons.Rounded.Delete,
458                             contentDescription = ""
```

```
459                         )
460                     }
461                     IconButton(onClick = {
462                         previewLazyViewModel.sendLazyUnit(
                                lazyUnitData.lazyUnits[0])
463                     }) {
464                         Icon(
465                             Icons.Rounded.Send,
466                             contentDescription = ""
467                         )
468                     }
469                 }
470             )
471         },
472         floatingActionButton = {
473             FloatingActionButton(
474                 onClick = { creationDialog.value = true }
475             ) {
476                 Icon(
477                     Icons.Rounded.Add,
478                     contentDescription = "",
479                 )
480             }
481         }
482     ) {
483         if (creationDialog.value) {
484             CreateLazyUnitDialog(
485                 onDismissRequest = { creationDialog.value =
                        false },
486                 onComplete = previewLazyViewModel::
                        addLazyUnit,
487             )
488         }
489         LazyColumn(
490             modifier = Modifier
491                 .padding(it)
492                 .fillMaxHeight()
493                 .padding(horizontal = 16.dp)
494         ) {
495             items(lazyUnitData.lazyUnits) { unit ->
496                 LazyUnitCard(
```

```
                    unit,
                    modifier = Modifier.padding(horizontal =
                        4.dp)
                )
            }
        }
    }
}
LazyUnitRepository.kt
package com.vladcto.lazymeter.data.lazy.repository

import com.vladcto.lazymeter.data.lazy.domain.LazyReason
import com.vladcto.lazymeter.data.lazy.domain.LazyUnit
import com.vladcto.lazymeter.data.lazy.infra.LazyReasonDb
import com.vladcto.lazymeter.data.lazy.infra.LazyUnitDao
import com.vladcto.lazymeter.data.lazy.infra.LazyUnitDb
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.async
import kotlinx.coroutines.withContext
import javax.inject.Inject

class LazyUnitRepository @Inject constructor(private val
    _lazyUnitDao: LazyUnitDao) {
    suspend fun getAll(): List<LazyUnit> = withContext(
        Dispatchers.IO) {
        val result = async {
            _lazyUnitDao.getAll()
        }.await()
        return@withContext result.map { it.toDomain() }
    }

    suspend fun clear() = withContext(Dispatchers.IO) {
        return@withContext _lazyUnitDao.clear()
    }

    suspend fun add(unit: LazyUnit) = addAll(listOf(unit))

    suspend fun addAll(units: List<LazyUnit>) = withContext(
        Dispatchers.IO) {
        return@withContext _lazyUnitDao.insertAll(units.map {
            it.toDb() })
```

```
533         }
534     }
535
536     // Mappers
537
538     private fun LazyUnit.toDb(): LazyUnitDb = LazyUnitDb(
539         time = this.time,
540         reason = LazyReasonDb.valueOf(this.reason.name),
541     )
542
543     private fun LazyUnitDb.toDomain(): LazyUnit = LazyUnit(
544         time = this.time,
545         reason = LazyReason.valueOf(this.reason.name),
546     )
547
548     LazyUnitDao.kt
549     package com.vladcto.lazymeter.data.lazy.infra
550
551     import androidx.room.Dao
552     import androidx.room.Insert
553     import androidx.room.Query
554
555     @Dao
556     interface LazyUnitDao {
557         @Query("SELECT * FROM lazyUnit")
558         suspend fun getAll(): List<LazyUnitDb>
559
560         @Query("DELETE FROM lazyUnit")
561         suspend fun clear()
562
563         @Insert
564         suspend fun insertAll(units: List<LazyUnitDb>)
565     }
566     LazyUnitDb.kt
567     package com.vladcto.lazymeter.data.lazy.infra
568
569     import androidx.room.Entity
570     import androidx.room.PrimaryKey
571     import java.util.Date
572
573     enum class LazyReasonDb {
```

```
574         Tired,
575         Distracted,
576     }
577
578     @Entity(tableName = "lazyUnit")
579     data class LazyUnitDb(
580         @PrimaryKey(autoGenerate = true) val uid: Int = 0,
581         val reason: LazyReasonDb,
582         val time: Date,
583     )
584     LazyUnit.kt
585     package com.vladcto.lazymeter.data.lazy.domain
586
587     import java.util.Date
588
589     enum class LazyReason {
590         Tired,
591         Distracted,
592     }
593
594     data class LazyUnit(
595         val time: Date,
596         val reason: LazyReason,
597     )
```