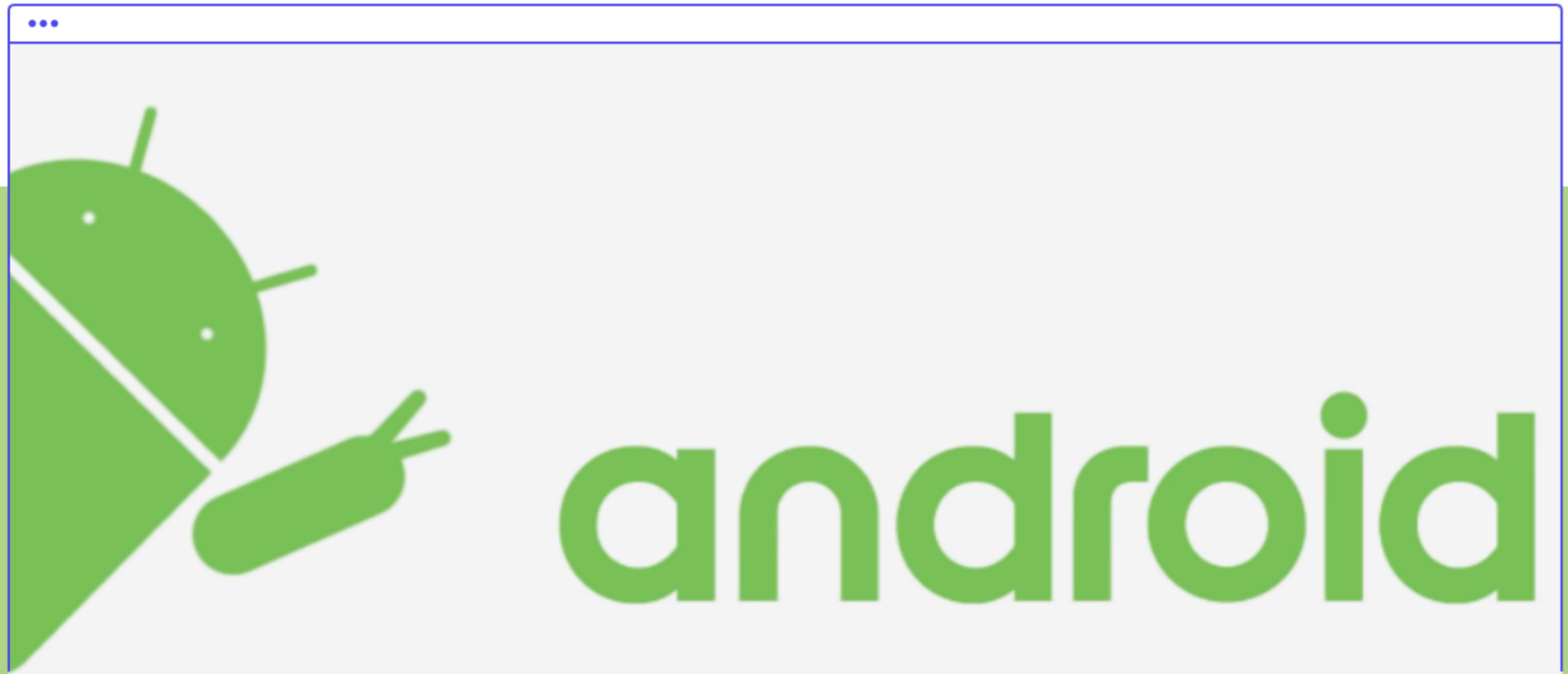
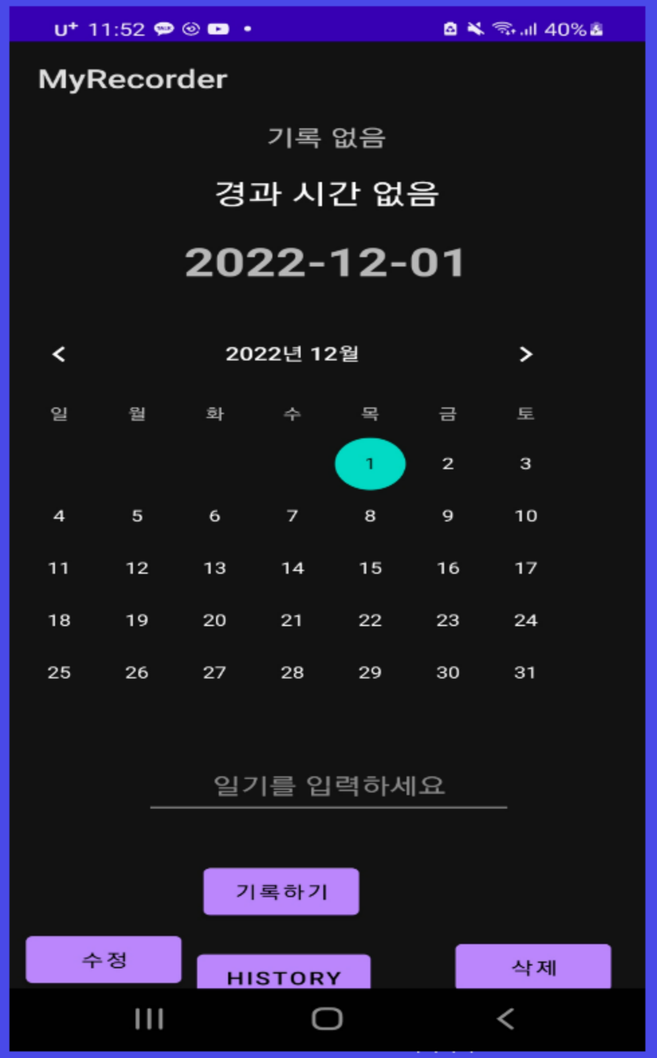
2022

1801005 강신성

일기장 만들기 과제 설명





activity_main.xml

메인화면 입니다

달력 => android:id="@+id/calendarView"

기록하기
android:id="@+id/buttonRecord"

HISTORY
android:id="@+id/buttonHistory"

수정
android:id="@+id/buttonUpdate"

삭제
android:id="@+id/buttonDelete"

```
binding.buttonRecord.setOnClickListener {onSave()}

binding.buttonHistory.setOnClickListener {  it: View!
    startActivity(Intent( packageContext: this, HistoryActivity::class.java))
}
binding.buttonUpdate.setOnClickListener {  it: View!
    onUpdate()
}
binding.buttonDelete.setOnClickListener {  it: View!
    onDelete()
}
```

MainActivity.kt

각 버튼에 해당하는 함수를 넣기

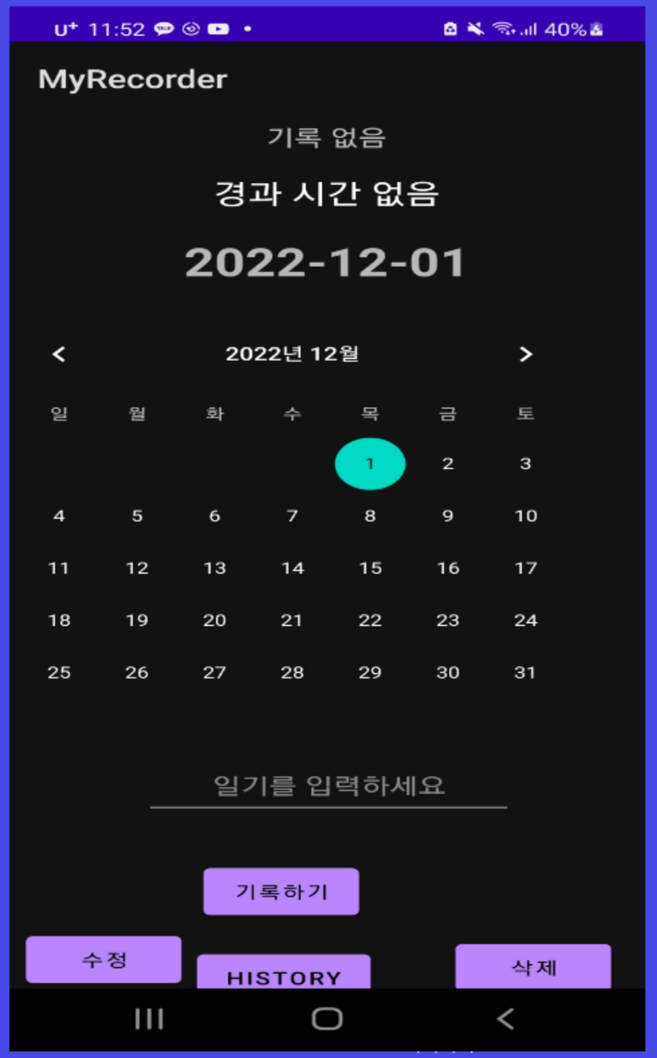
```kotlin
fun onSave(){
    with(sharedPreference.edit()){ this: SharedPreferences.Editor!
        val food = binding.editTextFoodName.text.toString()

        if(food.isNotEmpty()) {
            this.putString("food", food)
              val time = LocalDateTime.now().toString()
            this.putString("time", time)
            this.apply()

            CoroutineScope(Dispatchers.IO).launch { this: CoroutineScope
                val db = MyDatabase.getInstance( context: this@MainActivity)
                db?.myDao()?.insert(food, time)
            }
        }
    }
}
```

```kotlin
fun onUpdate(){
    with(sharedPreference.edit()){ this: SharedPreferences.Editor!
        val food = binding.editTextFoodName.text.toString()

        if(food.isNotEmpty()) {
            this.putString("food", food)
              val time = LocalDateTime.now().toString()
            this.putString("time", time)
            this.apply()

            CoroutineScope(Dispatchers.IO).launch { this: CoroutineScope
                val db = MyDatabase.getInstance( context: this@MainActivity)
                db?.myDao()?.update(food, time)
            }
        }
    }
}
```

```kotlin
fun onDelete(){
    with(sharedPreference.edit()){ this: SharedPreferences.Editor!
        val food = binding.editTextFoodName.text.toString()

        if(food.isNotEmpty()) {
            this.putString("food", food)
              val time = LocalDateTime.now().toString()
            this.putString("time", time)
            this.apply()
        }
            CoroutineScope(Dispatchers.IO).launch { this: CoroutineScope
                val db = MyDatabase.getInstance( context: this@MainActivity)
                db?.myDao()?.delete(time)
                  db?.myDao()?.deleteAll()
```

insert , update , delete 의 기능을 하는 함수 구현 부분입니다

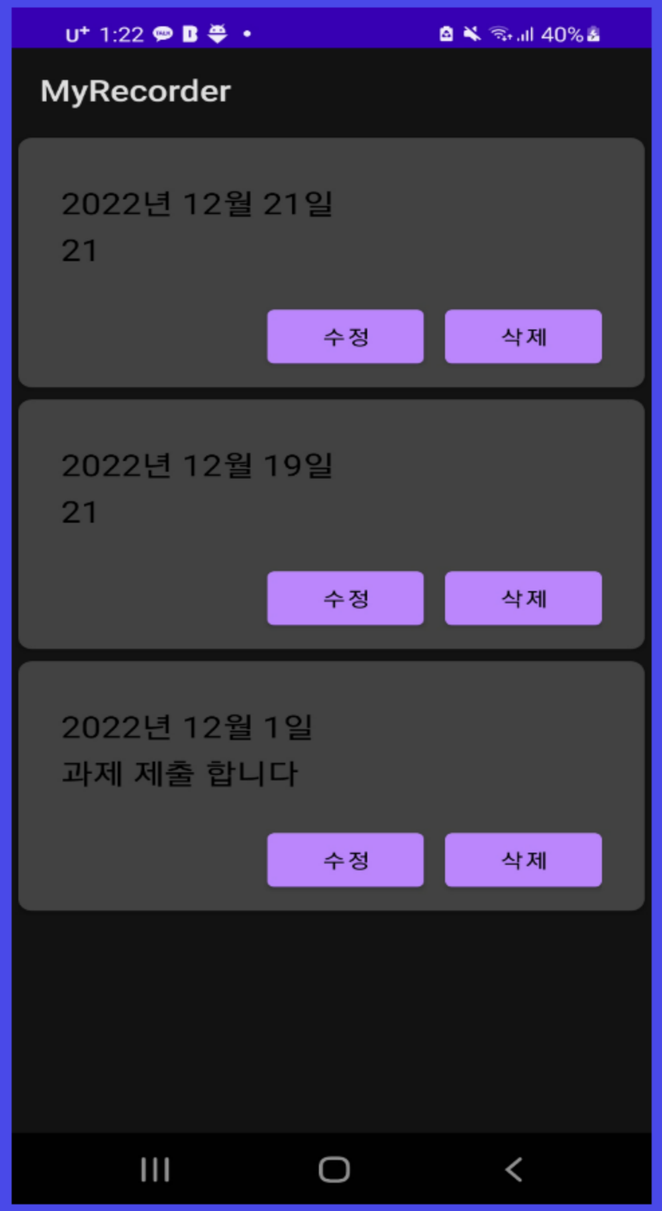
* DB를 불러와서 myDao를 통해 조작




```kotlin
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(binding.root)

    val calendarView: CalendarView = findViewById(R.id.calendarView)
    val dayText: TextView = findViewById(R.id.day_text)
    val dateFormat: DateFormat = SimpleDateFormat( pattern: "yyyy-MM-dd")

    val date: Date = Date(calendarView.date)
    dayText.text = dateFormat.format(date)
    var time: String = dateFormat.format(date)
    calendarView.setOnDateChangeListener{calendarView, year, month, dayOfMonth ->
        var day: String = "${year}년 ${month + 1}월 ${dayOfMonth}일"
        dayText.text = day
        time= day
    }
```

달력의 id 값인 calendatView를 바인딩하여
지정한SimpleDateFormat의 맞추어
dateFormat 변수에 넣은 후

data를 format에 맞춰 daytext.text 에 보여주고
setOnDateChangeListener를 사용해서
클릭한 날짜에 데이터를 바인딩 한다




```
        CoroutineScope(Dispatchers.IO).launch {  this: CoroutineScope
            val dao = MyDatabase.getInstance( context: this@HistoryActivity)?.myDao()
            val result = dao?.selectAll()!!
            withContext(Dispatchers.Main) {  this: CoroutineScope
                adapter.updateData(result)
            }
        }
    }
}
```

Mydao에 정의 되어있는 selectAll( ) 함수를 result에 담아서 adapter에 updateData 실행

리사이클 뷰를 이용해서 일기 list를 출력합니다

activity_history.xml

item_history.xml

```kotlin
@Dao
interface MyDao {
    @Query("select * from MyRecord")
    fun selectAll():List<MyRecord>

    @Query("update MyRecord set food = :food where time = :time ")
    fun update(food: String, time: String)

    @Query("delete from MyRecord where time = :time")
    fun delete(time: String)

    @Query("delete from MyRecord")
    fun deleteAll()

    @Query("Insert into MyRecord (food, time) values (:food, :time )")
    fun insert(food: String, time: String)
```

MyDao의 정의된 @Query를 통해 Room db를 조작할 수 있게 됩니다

where 조건절에 날짜의 정보가 담긴 time 을 넣게되면, 선택한 time에 해당하는 일기만 조작합니다

```kotlin
@Entity
data class MyRecord(
    @PrimaryKey(autoGenerate = true) val rid:Int,
    @ColumnInfo val food:String,
    @ColumnInfo val time:String

)
```

MyRecord 파일에는 Primarykey로 지정한 rid에 autoincrement 적용한 코드입니다

코틀린 언어를 처음 접하는 학생들에게
따라서 만들어 볼 수 있는 ppt를 제공해
주셔서 정말 감사합니다

한 학기 동안 고생 많으셨습니다
감사합니다