



**LICENCIATURA EN INGENIERÍA DE TECNOLOGÍAS DE LA
INFORMACIÓN E INNOVACIÓN DIGITAL**

Desarrollo de software

Unidad IV.

Actividad. Capturas de Pantalla

Materia

Desarrollo de aplicaciones móviles

Alumno:

Ramiro Padierna Delgado **1224100710**

Carlos Emanuel Valentino **1224100**

Profesor:

Anastasio Rodríguez García

Índice

Copia del README	1
# MedicTrack.....	1
## Características	1
## Capturas de Pantalla	2
## Tecnologías Utilizadas.....	4
## Instalación.....	4
##  Ejemplos de Código Documentado (KDoc / JSDoc).....	6
Ejemplo 1	6
Ejemplo 2	9
## Autores.....	11

Copia del README

MedicTrack

Esta es una aplicacion de recordatorio de toma de medicamentos diriida a quellas personas que padecen alguna enfermedad y mantienen una vida demaciado ocupada o agetriada y suelen olvidar tomar sus medicamentos esta aplicacion ayudara a recordarle la hora de la toma de su medicamento al igual que tiene una funcion de alergias a medicamento que al registrar tu alergia algun medicamento si lo quieres agregar para tomar te mandar una alerta de que eres alergico a este medicamentto

Caracteristicas

1. Diseño innovador
2. con UX
3. ergonomico para el publico al que esta dirigido
4. Historial
5. Almacenamiento en la nube
6. Alerta de alergias
7. Sistema de autenticacion seguro
8. Seguimiento de adherencia en el medicamento

Capturas de Pantalla

Crear Cuenta

Nombre completo

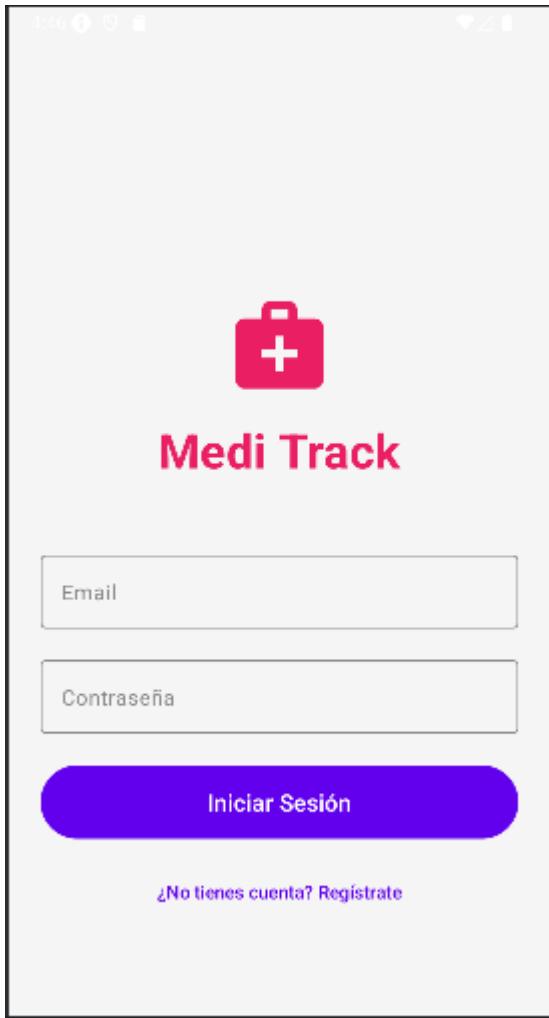
Email

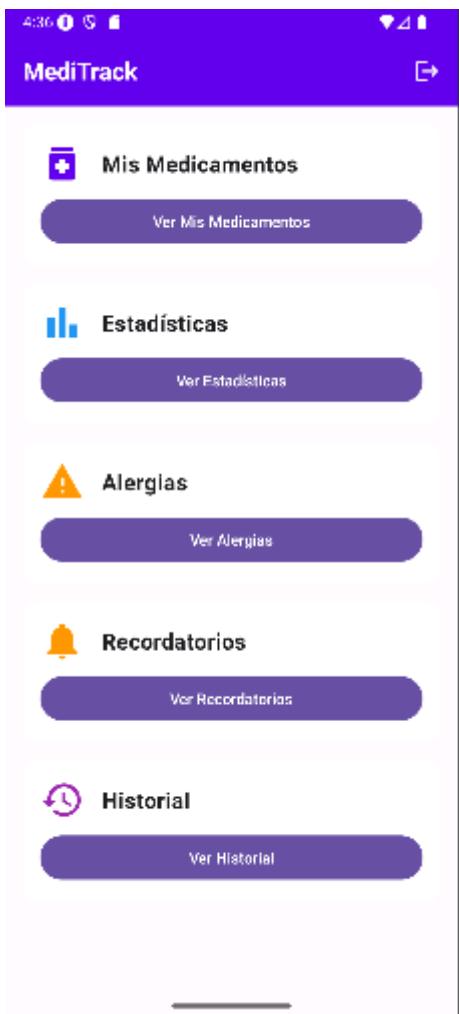
Contraseña

Confirmar contraseña

Crear Cuenta

¿Ya tienes cuenta? Inicia sesión





Tecnologías Utilizadas

- Kotlin
- Jetpack Compose
- MVVM
- Firebase
- Android Studio

Instalación

1. Clona este repositorio:

<https://github.com/RamiroPaD/MedicTrack.git>

2. Abre el proyecto en Android Studio.

3. Espera a que Gradle termine de compilar.
4. Ejecuta la app en un emulador o dispositivo físico.

Estructura del Proyecto

ESTRUCTURA DE CARPETAS:

```
```plaintext
app/src/main/java/mx/edu/utng/rpd/medittrack/
|
├── models/
| ├── Usuario.kt
| ├── Medicamento.kt
| ├── Recordatorio.kt
| ├── Historial.kt
| ├── Alergia.kt
| └── MedicamentoAPI.kt
|
├── repository/
| ├── FirebaseRepository.kt
| └── MedicamentosAPIRepository.kt
|
└── viewmodels/
 ├── AuthViewModel.kt
 ├── MedicamentosViewModel.kt
 └── RecordatoriosViewModel.kt
|
```

```

├── services/
| ├── MediTrackMessagingService.kt
| └── MedicationAlarmReceiver.kt
|
| ...
|
└── utils/
 └── NotificationHelper.kt
|
| ...
|
└── MediTrackApplication.kt
└── MainActivity.kt
```
---
```

📄 Ejemplos de Código Documentado (KDoc / JSDoc)

Ejemplo 1

```

12.  * ViewModel encargado de manejar la lógica relacionada con la gestión
13.  * de medicamentos, incluyendo carga, creación y eliminación.
14.  *
15.  * Utiliza rutinas y StateFlow para exponer estados inmutables a la interfaz de usuario.
16.  */
17. class MedicamentosViewModel(
18.     private val repositorio: RepositorioDeMedicamentos
19. ) : ModeloDeVista() {
20.
21. /**
22.  * Observables de StateFlows
23.  */
24.
25. /**
26.  * Lista de medicamentos obtenida de Firebase
27.  * privado val _medicamentos = FlujodeEstado<MutableList<Medicamento>>(listEmpty())
28.  * val medicamentos=_medicamentos.asStateFlow()
29.
30. /**
31.  * Estado de carga (loading) para mostrar u ocultar un indicador en la UI
32.  * privado val _cargando = FlujodeEstado<Boolean>(false)
33.  * val cargando=_cargando.asStateFlow()
34.
35. /**
36.  * Mensajes de error o éxito para mostrar comentarlos al usuario
37.  * privado val _mensaje = FlujodeEstado<Cadena?>(null)
38.  * val mensaje=_mensaje.asStateFlow()
39.
40. /**
41.  * Se ejecuta al inicializar el ViewModel.
42.  * Carga automáticamente la lista de medicamentos.
43.  */
44. init{
45.     cargarMedicamentos()
46. }
47.
48. /**
49.  * Métodos principales
50. */
51.
```

```
49     /**
50      * Obtiene los medicamentos desde el repositorio y actualiza el estado.
51      */
52     diversión cargar medicamentos() {
53         viewModelScope.launch {
54             _cargando.valor= verdadero
55             valresultado=repo.obtenerMedicamentos()
56
57             si(resultado.esExito) {
58                 _medicamentos.valor=resultado.obtenerONulo()?.emptyList()
59             }de lo contrario{
60                 _mensaje.valor= "Error al cargar medicamentos"
61             }
62
63             _cargando.valor= falso
64         }
65     }
66
67     /**
68      * Agrega un medicamento a Firebase.
69      *
70      * @param medicamento objeto Medicamento a agregar.
71      * @param onSuccess callback cuando la operación es exitosa.
72      * @param onError callback con un mensaje de error si falla.
73      */
74     diversión agregar medicamento(
75         medicamento: Medicamento,
76         onSuccess:{}-> Unidad,
77         onError:(Cadena)-> Unidad
78     ) {
79         viewModelScope.launch {
80             _cargando.valor= verdadero
81
82             intentar{
83                 valresultado=repo.agregarMedicamento(medicamento)
84             }de lo contrario{
85                 _mensaje.valor= resultado.error
86             }
87         }
88     }
89 }
```

```

        si(resultado.esExito) {
            cargarMedicamentos()//Refresca la lista
            _mensaje.valor= "Medicamento agregado correctamente"
            onSuccess()
        }de lo contrario{
            valerror=resultado.excepcionONulo()?.mensaje?: "Error desconocido"
            _mensaje.valor=error
            onError(error)
        }

    }captura(e: Excepción) {
    valerror=e.mensaje?: "Error al agregar medicamento"
    _mensaje.valor=error
    onError(error)

}finalmente{
    _cargando.valor=> falso
}
}

/***
 * Elimina un medicamento de Firebase mediante su ID.
 *
 * @param medId ID del medicamento a eliminar
 */
división eliminarMedicamento(ID de med: Cadena) {
    viewModelScope.launch {
        _cargando.valor= verdadero

        valresultado=repo.eliminarMedicamento(ID de med)

        si(resultado.esExito) {
            cargarMedicamentos()
            _mensaje.valor= "Medicamento eliminado"
        }
    }
}

```

Ejemplo 2

```
* ViewModel encargado de manejar la lógica relacionada con:  
* - Recordatorios del día  
* - Marcado de recordatorios como tomados u omitidos  
* - Historial de medicamentos  
* - Estadísticas de cumplimiento  
*  
* Utiliza corrutinas y StateFlow para actualizar y exponer datos reactivos a la UI.  
*/  
clase RecordatoriosViewModel(  
    privado val repositorio: Repositorio de Firebase=Repositorio de Firebase()  
) : Modelo de vista() {  
  
    //=====  
    //Observables de StateFlows  
    //=====  
  
    /** Lista de recordatorios del día actual*/  
    privado val _recordatorios = Flujo de estado mutable<Lista<Recordatorio>>(listaEmpty())  
    val recordatorios= _recordatorios.asStateFlow()  
  
    /** Historial completo de eventos (tomas y omisiones)*/  
    privado val _histórico = Flujo de estado mutable<Lista<Historial>>(listaEmpty())  
    val histórico= _histórico.asStateFlow()  
  
    /** Estadísticas de cumplimiento (ej. tomados vs. omitidos)*/  
    privado val _estadísticas = Flujo de estado mutable<Mapa<Cadena,Int>>(emptyMap())  
    val estadísticas= _estadísticas.asStateFlow()  
  
    /** Estado de carga para mostrar u ocultar indicadores*/  
    privado val _cargando = Flujo de estado mutable(falso)  
    val cargando= _cargando.asStateFlow()  
  
    //=====  
    //Métodos principales  
    //=====
```

```

    * Carga los recordatorios correspondientes al día de hoy
    */
    diversión cargarRecordatorios() {
        viewModelScope.launch {
            _cargando.valor= verdadero

            valresultado=repo.obtenerRecordatoriosHoy()

            si(resultado.esÉxito) {
                _recordatorios.valor=resultado.obtenerONulo()?:emptyList()
            }

            _cargando.valor= falso
        }
    }

    /**
     * Marca un recordatorio como tomado
     *
     * - Actualiza el estado del recordatorio en Firebase.
     * - Registre el evento en el historial.
     * - Recarga la lista de recordatorios.
     *
     * @param recordatorio Recordatorio a marcar como tomado.
     */
    diversión marcarComoTomado(recordatorio: Recordatorio) {
        viewModelScope.launch {

            //1. Actualizar el estado del recordatorio
            repo.actualizarEstadoRecordatorio(recordatorio.id,"completado")

            //2. Registrar evento en historial
            valhist= Historial(
                medicamentoId=recordatorio.medicamentoId,
                nombreMedicamento=recordatorio.nombreMedicamento,
        }
    }
}

```

```

87             dosis=recordatorio.dosis,
88             fechaHora= Marca de tiempo.now(),
89             estado= "tomado"
90         )
91         repo.agregarHistorial(hist)
92
93         //3. Recargar recordatorios
94         cargarRecordatorios()
95     }
96 }
97
98 /**
99  * Marca un recordatorio como omitido.
100 *
101 * - Actualiza su estado en Firebase.
102 * - Agrega una entrada al historial.
103 * - Recarga la lista de recordatorios.
104 *
105 * @param recordatorio Recordatorio a marcar como omitido.
106 */
107 diversion marcarComoOmitido(recordatorio: Recordatorio) {
108     viewModelScope.launch {
109
110         repo.actualizarEstadoRecordatorio(recordatorio.id,"omitido")
111
112         valhist= Historial(
113             medicamentoId=recordatorio.medicamentoId,
114             nombreMedicamento=recordatorio.nombreMedicamento,
115             dosis=recordatorio.dosis,
116             fechaHora= Marca de tiempo.now(),
117             estado= "omitido"
118         )
119         repo.agregarHistorial(hist)
120
121         cargarRecordatorios()
122     }

```

Autores

- Ramiro Padierna Delgado
- Carlos Emanuel Valentino Martinez