Guía Completa: Agregar Nueva Métrica a Health Connect Export

Resumen

Esta guía explica cómo agregar una nueva métrica (como Blood Pressure, Body Temperature, Heart Rate, etc.) al sistema de exportación de Health Connect, asegurando que funcione en los 3 modos de exportación.

Arquitectura del Sistema

3 Modos de Exportación

1. **Manual Export** (`WelcomeScreen.kt`) - 30 días, solo Samsung Health
2. **Auto Full Export** (`AutoExportWorker.kt`) - Histórico completo, todas las apps
3. **Auto Differential Export** (`AutoExportWorker.kt`) - Solo cambios desde último export

Pasos para Agregar una Nueva Métrica

Paso 1: AndroidManifest.xml

Agregar permisos de lectura/escritura para la nueva métrica.

Ubicación: `app/src/main/AndroidManifest.xml`

```xml
<!-- Ejemplo para Blood Pressure -->
<uses-permission android:name="android.permission.health.READ_BLOOD_PRESSURE"/>
<uses-permission android:name="android.permission.health.WRITE_BLOOD_PRESSURE"/>

<!-- Ejemplo para Heart Rate -->
<uses-permission android:name="android.permission.health.READ_HEART_RATE"/>
<uses-permission android:name="android.permission.health.WRITE_HEART_RATE"/>

<!-- Ejemplo para Body Temperature -->
<uses-permission android:name="android.permission.health.READ_BODY_TEMPERATURE"/>
<uses-permission android:name="android.permission.health.WRITE_BODY_TEMPERATURE"/>
```

Paso 2: InputReadingsViewModel.kt

Agregar el import y el permiso en el conjunto de permisos.

Ubicación: `app/src/main/java/.../presentation/screen/inputreadings/InputReadingsViewModel.kt`

```kotlin
```

```kotlin
// Agregar import
import androidx.health.connect.client.records.BloodPressureRecord
import androidx.health.connect.client.records.HeartRateRecord
import androidx.health.connect.client.records.BodyTemperatureRecord

// En la variable permissions, agregar:
val permissions = setOf(
    // ... permisos existentes ...
    HealthPermission.getReadPermission(BloodPressureRecord::class),
    HealthPermission.getWritePermission(BloodPressureRecord::class)
)
```

Paso 3: HealthConnectManager.kt

Agregar función para leer la nueva métrica.

Ubicación: app/src/main/java/.../data/HealthConnectManager.kt

```kotlin
// Ejemplo para Blood Pressure
suspend fun readBloodPressureRecords(
    startTime: Instant,
    endTime: Instant
): List<BloodPressureRecord> {
    val request = ReadRecordsRequest(
        recordType = BloodPressureRecord::class,
        timeRangeFilter = TimeRangeFilter.between(startTime, endTime)
    )
    val response = healthConnectClient.readRecords(request)
    return response.records
}

// Ejemplo para Heart Rate
suspend fun readHeartRateRecords(
    startTime: Instant,
    endTime: Instant
): List<HeartRateRecord> {
    val request = ReadRecordsRequest(
        recordType = HeartRateRecord::class,
        timeRangeFilter = TimeRangeFilter.between(startTime, endTime)
    )
    val response = healthConnectClient.readRecords(request)
    return response.records
}
```

Paso 4: HealthDataSerializer.kt

Modificar la función `generateHealthJSON` para incluir la nueva métrica.

Ubicación: `app/src/main/java/.../data/HealthDataSerializer.kt`

kotlin

```kotlin
fun generateHealthJSON(
    weightRecords: List<WeightData>,
    exerciseData: List<Map<String, Any?>>,
    sleepData: List<Map<String, Any?>>,
    bloodPressureData: List<Map<String, Any?>>,  // AGREGAR
    exportType: String
): String {
    return buildString {
        // ... código existente ...

        // Agregar nueva sección antes del cierre
        append("  },\n")
        append("  \"blood_pressure_records\": {\n")
        append("    \"count\": ${bloodPressureData.size},\n")
        append("    \"data\": ")
        append(serializeMapList(bloodPressureData))
        append("\n  }\n")

        append("}")
    }
}
```

Paso 5: WelcomeScreen.kt (Export Manual)

Agregar lectura y serialización en `performManualExport()`.

Ubicación: `app/src/main/java/.../presentation/screen/welcome/WelcomeScreen.kt`

kotlin

```kotlin
// En la función shareWeightDataToGoogleDrive(), agregar:

// Leer Blood Pressure
val bloodPressureRecords = if (healthConnectManager != null) {
    healthConnectManager.read[NombreMetrica]Records(startTime, endTime).filter {
        it.metadata.dataOrigin.packageName == "com.sec.android.app.shealth"
    }
} else {
    emptyList()
}

// Actualizar llamada al serializer
val jsonContent = HealthDataSerializer.generateHealthJSON(
    filteredWeightRecords,
    exerciseData,
    sleepData,
    bloodPressureRecords,  // AGREGAR
    "MANUAL_SAMSUNG_ONLY"
)
```

Paso 6: AutoExportWorker.kt (Export Automático)

A) Agregar imports

```kotlin
import androidx.health.connect.client.records.BloodPressureRecord
```

B) Actualizar permisos requeridos (línea ~55)

```kotlin
val requiredPermissions = setOf(
    HealthPermission.getReadPermission(WeightRecord::class),
    HealthPermission.getReadPermission(ExerciseSessionRecord::class),
    HealthPermission.getReadPermission(SleepSessionRecord::class),
    HealthPermission.getReadPermission(BloodPressureRecord::class)  // AGREGAR
)
```

C) En performFullExport(), agregar lectura

```kotlin
```

```
// Después de leer weight y exercise
val bloodPressureData = readBloodPressureRecords(client, startTime, endTime)

// Actualizar generateHealthJSON
val jsonContent = generateHealthJSON(
    weightData,
    exerciseData,
    emptyList(),  // sleep si corresponde
    bloodPressureData,  // AGREGAR
    "AUTO_FULL_EXPORT"
)
```

D) En performDifferentialExport():

1. Actualizar ChangesTokenRequest (línea ~115):

```
kotlin

val token = client.getChangesToken(
    ChangesTokenRequest(
        recordTypes = setOf(
            WeightRecord::class,
            ExerciseSessionRecord::class,
            SleepSessionRecord::class,
            BloodPressureRecord::class  // AGREGAR
        )
    )
)
```

2. Agregar lista para cambios:

```
kotlin

val bloodPressureChanges = mutableListOf<Map<String, Any?>>()
```

3. Agregar caso en el when statement:

```
kotlin
```

```
is BloodPressureRecord -> {
    bloodPressureChanges.add(mapOf(
        "timestamp" to record.time.toString(),
        "systolic" to record.systolic.inMillimetersOfMercury,
        "diastolic" to record.diastolic.inMillimetersOfMercury,
        "source" to record.metadata.dataOrigin.packageName,
        "change_type" to "UPSERT"
    ))
    Log.d(TAG, "  + Blood Pressure change detected")
}
```

4. Actualizar condición de "no hay cambios":

```
kotlin

if (weightChanges.isEmpty() &&
    exerciseChanges.isEmpty() &&
    sleepChanges.isEmpty() &&
    bloodPressureChanges.isEmpty() &&  // AGREGAR
    deletions.isEmpty()) {
```

5. Actualizar generateDifferentialJSON:

```
kotlin

private fun generateDifferentialJSON(
    weightChanges: List<Map<String, Any?>>,
    exerciseChanges: List<Map<String, Any?>>,
    sleepChanges: List<Map<String, Any?>>,
    bloodPressureChanges: List<Map<String, Any?>>,  // AGREGAR
    deletions: List<String>
): String {
    // Agregar sección en el JSON
}
```

E) Agregar función helper para leer records

```
kotlin
```

```
private suspend fun readBloodPressureRecords(
    client: HealthConnectClient,
    startTime: Instant,
    endTime: Instant
): List<Map<String, Any?>> {
    return try {
        val request = ReadRecordsRequest(
            recordType = BloodPressureRecord::class,
            timeRangeFilter = TimeRangeFilter.between(startTime, endTime)
        )
        val response = client.readRecords(request)

        response.records.map { record ->
            mapOf(
                "timestamp" to record.time.toString(),
                "systolic" to record.systolic.inMillimetersOfMercury,
                "diastolic" to record.diastolic.inMillimetersOfMercury,
                "source" to record.metadata.dataOrigin.packageName
            )
        }
    } catch (e: Exception) {
        Log.e(TAG, "Error reading blood pressure records", e)
        emptyList()
    }
}
```

Paso 7: JsonExportWorker.kt

Si tienes export filtrado, agregar la lectura similar a AutoExportWorker.

Checklist de Verificación

- [ ] **AndroidManifest.xml** → Permisos READ/WRITE
- [ ] **InputReadingsViewModel.kt** → Import + Permiso
- [ ] **HealthConnectManager.kt** → Función read[Metrica]Records()
- [ ] **HealthDataSerializer.kt** → Función serialize + Modificar generateHealthJSON
- [ ] **WelcomeScreen.kt** → Leer + Filtrar + Pasar al serializer (Manual)
- [ ] **AutoExportWorker.kt** → Import + Permisos + Token + Changes + JSON (Auto)
- [ ] **JsonExportWorker.kt** → Función filtrada + Pasar al serializer (si aplica)

Métricas Comunes y sus Tipos

Métrica	Record Type	Campos Principales
Blood Pressure	BloodPressureRecord	systolic, diastolic
Heart Rate	HeartRateRecord	beatsPerMinute

Métrica	Record Type	Campos Principales
Body Temperature	BodyTemperatureRecord	temperature
Blood Glucose	BloodGlucoseRecord	level
Oxygen Saturation	OxygenSaturationRecord	percentage
Steps	StepsRecord	count
Hydration	HydrationRecord	volume
Nutrition	NutritionRecord	calories, protein, fat, carbs

Notas Importantes

1. **Siempre agregar la métrica en TODOS los modos de exportación** para mantener consistencia
2. **El export Manual filtra por Samsung Health**, los Auto no filtran
3. **Differential Export necesita el record type en ChangesTokenRequest** o no detectará cambios
4. **Verificar que los permisos estén otorgados** antes de intentar leer
5. **Manejar excepciones** en las funciones de lectura para evitar crashes

Ejemplo Completo: Agregar Heart Rate

1. AndroidManifest.xml

```xml
<uses-permission android:name="android.permission.health.READ_HEART_RATE"/>
```

2. HealthConnectManager.kt

```kotlin
suspend fun readHeartRateRecords(
    startTime: Instant,
    endTime: Instant
): List<HeartRateRecord> {
    val request = ReadRecordsRequest(
        recordType = HeartRateRecord::class,
        timeRangeFilter = TimeRangeFilter.between(startTime, endTime)
    )
    return healthConnectClient.readRecords(request).records
}
```

3. AutoExportWorker.kt - Differential

```kotlin
```

```
is HeartRateRecord -> {
    record.samples.forEach { sample ->
        heartRateChanges.add(mapOf(
            "timestamp" to sample.time.toString(),
            "bpm" to sample.beatsPerMinute,
            "source" to record.metadata.dataOrigin.packageName,
            "change_type" to "UPSERT"
        ))
    }
    Log.d(TAG, "  + Heart Rate change detected")
}
```

Solución de Problemas

Problema: La métrica no aparece en el export diferencial

- **Solución:** Verificar que el record type esté en ChangesTokenRequest

Problema: Permisos denegados

- **Solución:** Verificar AndroidManifest.xml y solicitar permisos en runtime

Problema: JSON malformado

- **Solución:** Verificar que todas las comas estén correctas en generateHealthJSON

Problema: No se detectan cambios

- **Solución:** Limpiar el token con ChangeTokenManager.clearToken() para forzar un full export