



# UNIVERSIDAD CENTRAL DEL ECUADOR

## FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

### CARRERA DE COMPUTACIÓN

**NOMBRE:** Diego Borja – GRUPO 1

**FECHA:** 10 de febrero de 2026

**TEMA:** Pregunta 4.

#### Uso del servicio msginsert

```
interface ApiService {
    // --- PRODUCTOS ---
    @POST("value = \"producto\"")
    suspend fun syncProduct(@Body product: ProductDto): Response<SyncResponse>

    // REQUERIMIENTO EXAMEN: Insertar y notificar por correo
    @POST("value = \"msginsert\"")
    suspend fun insertAndNotify(@Body product: ProductDto): Response<SyncResponse>

    // --- USUARIOS ---
    @GET("value = \"usuario\"")
    suspend fun syncUser(@Body user: User): Response<Unit>

    @GET("value = \"usuario\"")
    suspend fun getUser(@Query("value = \"nombre\") nombre: String): Response<User?>
}

data class SyncResponse(val message: String, val url: String?)
```

Dentro de **ApiService** se puso el endpoint *msginsert* esta es la señal para el servidor de que, además de guardar, debe enviar el correo electrónico con los datos del producto.

```
class SyncWorker(
    override suspend fun doWork(): Result {
    return try {
        val localProducts = repository.getProducts()

        for (product in localProducts) {
            try {
                val productDto = product.toDto(applicationContext)

                // REQUERIMIENTO PROFESOR: Usar msginsert para notificar por correo al insertar
                // Si el producto no tiene URL remota (es nuevo en la base), usamos msginsert
                val response = if (product.imageUri?.startsWith("http") != true) {
                    RetrofitClient.instance.insertAndNotify(productDto)
                } else {
                    RetrofitClient.instance.syncProduct(productDto)
                }

                if (response.isSuccessful) {
                    val body = response.body()
                    if (body?.url != null && body.url.startsWith("http")) {
                        repository.updateProduct(updatedProduct = product.copy(imageUri = body.url))
                    }
                }
            } catch (e: Exception) {
                Log.e(tag = "SyncWorker", msg = "Error sincronizando producto ${product.id}", tr = e)
            }
        }
        val totalLocalProducts = repository.getProducts().size
        showNotification(totalLocalCount = totalLocalProducts)
    } finally {
        Result.success()
    }
}
```

Dentro de la clase **SyncWorker** llamaremos al servicio creado para poder notificar el producto creado.



# UNIVERSIDAD CENTRAL DEL ECUADOR

## FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

### CARRERA DE COMPUTACIÓN

The screenshot shows the AWS SNS console with the following details:

**Nombre:** NotificacionesProductoCorreo

**ARN:** arn:aws:sns:us-east-1:324607928077:NotificacionesProductoCorreo

**Nombre para visualización:** -

**Tipo:** Estándar

**Propietario del tema:** 324607928077

**Suscripciones (1):**

ID	Punto de enlace	Estado	Protocolo
Eliminada	losininternetapp@gmail.com	Confirmada	EMAIL

Implementación del servicio dentro AWS.