**Análisis de Gestión del Riesgo**

**Interacción en Flutter - Basado en norma ISO 31000**

La interacción en el aplicativo desarrollado en Flutter de la información y los registros almacenados en AWS, introduce varios riesgos específicos, los cuales, aplicando la norma ISO 31000, se pueden identificar y gestionar los riesgos para asegurar la funcionalidad de lo implementado en el patrón diseñado en AWS para la suscripción y publicación de la información generada por el dispositivo IoT y mostrarla en el desarrollo realizado en Flutter.

La interacción entre el dispositivo IoT, AWS, y la aplicación Flutter introduce una nueva capa de riesgos. A continuación, presentamos la matriz de riesgos enfocada a la funcionalidad del dispositivo IoT con AWS basados en la norma ISO 31000.

**Matriz de Riesgos para la Interacción del Dispositivo IoT con la Aplicación Flutter**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Descripción del Riesgo** | **Impacto** | **Probabilidad** | **Nivel de Riesgo** | **Estrategia de Mitigación** |
| IF1 | Fallos en la Comunicación entre AWS y Flutter | Alto | Media | Alto | Implementar pruebas de conectividad y monitoreo continuo. Configurar alertas para fallos de comunicación. |
| IF2 | Retrasos en la Actualización de Datos en Flutter | Alto | Media | Alto | Optimizar el uso de WebSockets o Firebase para actualizaciones en tiempo real. Implementar mecanismos de caché. |
| IF3 | Problemas de Compatibilidad entre Flutter y AWS | Alto | Media | Alto | Realizar pruebas exhaustivas de integración y usar SDKs oficiales de AWS para Flutter. |
| IF4 | Pérdida de Datos durante la Transmisión | Alto | Media | Alto | Implementar mecanismos de reenvío y verificación de datos. Utilizar colas de mensajes como SQS. |
| IF5 | Seguridad en la Transmisión de Datos desde AWS a Flutter | Alto | Media | Alto | Utilizar HTTPS/TLS para todas las comunicaciones. Implementar autenticación y autorización robustas. |
| IF6 | Malfuncionamiento de la Aplicación Flutter | Alto | Media | Alto | Realizar pruebas unitarias y de integración, y pruebas de usuario. Tener un plan de recuperación ante fallos. |
| IF7 | Gestión Ineficiente de Estados en la Aplicación Flutter | Alto | Media | Alto | Utilizar soluciones de gestión de estados robustas como Provider o Riverpod. |
| IF8 | Problemas de Desempeño en Flutter | Alto | Media | Alto | Optimizar el código y los recursos utilizados en la aplicación. Monitorear y ajustar el rendimiento continuamente. |
| IF9 | Falta de Notificaciones en Tiempo Real | Alto | Media | Alto | Implementar servicios de notificaciones push y configurar actualizaciones en tiempo real. |
| IF10 | Errores en la Representación de Datos (LEDs) en la UI de Flutter | Alto | Media | Alto | Realizar pruebas exhaustivas de la interfaz de usuario y validar que los datos se representen correctamente. |
| IF11 | Falta de Sincronización entre el Dispositivo IoT y la App Flutter | Alto | Media | Alto | Implementar mecanismos de sincronización y validación de datos entre el dispositivo y la app. |

**DESCRIPCIÓN DETALLADA DE ESTRATEGIAS DE MITIGACIÓN**

**IF1: Fallos en la comunicación entre AWS y Flutter**

**Estrategia para Mitigar:**

* Implementar pruebas de conectividad y monitoreo continuo.
* Configurar alertas para fallos de comunicación.

**Acciones:**

* Utilizar servicios de monitoreo como CloudWatch para rastrear la conectividad.
* Configurar alertas para detectar y responder rápidamente a problemas de comunicación.

**IF2: Retrasos en la actualización de datos en Flutter**

**Estrategia para Mitigar:**

* Optimizar el uso de WebSockets o Firebase para actualizaciones en tiempo real.
* Implementar mecanismos de caché.

**Acciones:**

* Integrar WebSockets o Firebase para recibir actualizaciones instantáneas.
* Utilizar almacenamiento en caché para reducir la latencia.

**IF3: Problemas de compatibilidad entre Flutter y AWS**

**Estrategia para Mitigar:**

* Realizar pruebas exhaustivas de integración y usar SDKs oficiales de AWS para Flutter.

**Acciones:**

* Probar la aplicación con diferentes versiones de Flutter y AWS SDK.
* Utilizar SDKs oficiales y documentados por AWS.

**IF4: Pérdida de datos durante la transmisión**

**Estrategia para Mitigar:**

* Implementar mecanismos de reenvío y verificación de datos.
* Utilizar colas de mensajes como SQS.

**Acciones:**

* Configurar colas de mensajes para asegurar la entrega de datos.
* Implementar checksums y mecanismos de verificación de integridad de datos.

**IF5: Seguridad en la transmisión de datos desde AWS a Flutter**

**Estrategia para Mitigar:**

* Utilizar HTTPS/TLS para todas las comunicaciones.
* Implementar autenticación y autorización robustas.

**Acciones:**

* Configurar HTTPS/TLS para asegurar la transmisión de datos.
* Implementar autenticación mediante tokens seguros y políticas de acceso.

**IF6: Mal funcionamiento de la aplicación Flutter**

**Estrategia para Mitigar:**

* Realizar pruebas unitarias y de integración, y pruebas de usuario.
* Tener un plan de recuperación ante fallos.

**Acciones:**

* Desarrollar y ejecutar pruebas automatizadas.
* Realizar pruebas de usuario para detectar problemas potenciales.
* Definir y documentar un plan de recuperación ante fallos.

**IF7: Gestión ineficiente de estados en la aplicación Flutter**

**Estrategia para Mitigar:**

* Utilizar soluciones de gestión de estados robustas como Provider o Riverpod.

**Acciones:**

* Implementar un patrón de gestión de estado adecuado como Provider, Riverpod, o Bloc.
* Revisar y optimizar la gestión de estado en la aplicación.

**IF8: Problemas de desempeño en Flutter**

**Estrategia para Mitigar:**

* Optimizar el código y los recursos utilizados en la aplicación.
* Monitorear y ajustar el rendimiento continuamente.

**Acciones:**

* Perfilar la aplicación para identificar y solucionar cuellos de botella de rendimiento.
* Optimizar el uso de recursos y el renderizado de la UI.

**IF9: Falta de notificaciones en tiempo real**

**Estrategia para Mitigar:**

* Implementar servicios de notificaciones push y configurar actualizaciones en tiempo real.

**Acciones:**

* Integrar servicios de notificaciones push como Firebase Cloud Messaging (FCM).
* Configurar actualizaciones en tiempo real utilizando WebSockets o Firebase.

**IF10: Errores en la representación de datos (LEDs) en la UI de Flutter**

**Estrategia para Mitigar:**

* Realizar pruebas exhaustivas de la interfaz de usuario y validar que los datos se representen correctamente.

**Acciones:**

* Desarrollar pruebas de UI para validar la representación correcta de los datos.
* Revisar y ajustar el diseño de la interfaz de usuario según sea necesario.

**IF11: Falta de Sincronización entre el Dispositivo IoT y la App Flutter**

**Estrategia para Mitigar:**

* Implementar mecanismos de sincronización y validación de datos entre el dispositivo y la app.

**Acciones:**

* Utilizar colas de mensajes y mecanismos de sincronización para asegurar la consistencia de datos.
* Realizar pruebas de sincronización para detectar y solucionar discrepancias.

Este levantamiento de información basado en la norma ISO 31000 inicia con la identificación y evaluación de los riesgos asociados en el segundo sprint definido para nuestro proyecto. Los riesgos aquí mencionados pueden variar según la usabilidad y las configuraciones necesarias para la conexión pot MQTT desde la cuenta de AWS para la transmisión de información desde el dispositivo IoT.