****

**Grupo conformado por:**

* Tania Michelle Agredo Hurtado
* Nelly Jordan Ordoñez
* Valeria Sibaja Ledesma
* Carlos Alberto Toledo Ramirez

**Problema:** Seguridad de poblaciones vulnerables como mujeres, niños y población LGBTIQ+.

**Estadìsticas:**

Cada 11 minutos, una mujer o niña es asesinada por un familiar en el mundo (ONU Mujeres, 2022). Pero la violencia no termina ahí: en Colombia, una mujer es víctima de violencia sexual cada 30 minutos (Forensis 2022), y en 2023, se reportaron 728 feminicidios (Medicina Legal). Mientras tanto, en Europa, 1 de cada 5 mujeres sufre acoso o agresión en discotecas (UE Fundamental Rights Agency, 2022).

¿Sabías que en México, el 56% de las mujeres evita salir de noche por miedo? (INEGI, 2023). La violencia de género no solo mata; también encarcela, silencia y limita derechos básicos.

El secuestro en Colombia suele asociarse a delitos como trata de personas o desapariciones. Por ejemplo, en 2023, la Fiscalía reportó 1,452 mujeres víctimas de trata (Fiscalía General, 2023). La Policía Nacional reportó un aumento del 66% en secuestros a menores en los primeros nueve meses de 2023 (245 casos) en comparación con 2022 (147 casos). Entre 2016 y 2021, hubo 2032 víctimas registradas de homicidios, amenazas y actos de violencia policial contra la población lgbtq+ en Colombia (Colombia Diversa, 2022).

Para hablar más de lo micro, en nuestra ciudad, Cali en los últimos cinco años (2020-2024) han habido reportados 36622 de violencia intrafamiliar hacia mujeres y 5544 hacia menores de edad (Policía Nacional, 2024). Entre 2019 y 2024 hubo 45 feminicidios en la ciudad (Gobernación del Valle del Cauca, 2024). Además, entre 2016 y 2021 hubo 107 hechos registrados de homicidios, amenazas y actos de violencia policial contra la población lgtbq+ (Colombia Diversa, 2022).

**Solución:** Desarrollo de un aplicativo de alerta temprana integrado a blockchain, que permite compartir la ubicación con contactos de emergencias y entes responsables (policía, bomberos, etc) y realizar el reporte del caso en blockchain.

**Documentación del desarrollo - App JB PanicApp**

**Proyecto: Aplicación Móvil de Botón de Pánico (React Native, en un inicio)**

Una app pensada para ayudar en situaciones de emergencia, permitiendo enviar la ubicación a contactos previamente guardados, registrar eventos y notificar a través de WhatsApp y blockchain.

FASE 1: CRUD de contactos (red de apoyo)

Pantalla con formulario para agregar nombre + teléfono.

Lista de contactos guardados localmente (usaremos AsyncStorage).

Permitir editar y eliminar.

FASE 2: Doble toque en el ícono para activar el botòn de pánico

Detectar doble tap sobre un botón o zona visual discreta.

Al detectar doble toque:

Obtener ubicación.

Enviar mensaje a cada contacto (SMS o WhatsApp).

Registrar evento en la blockchain (usando ethers.js + Infura/Alchemy).

FASE 3: Integración con SMS/WhatsApp y Smart Contract

SMS: usar librerías nativas como react-native-sms o integrar con Twilio vía backend.

WhatsApp: lanzar un enlace whatsapp://send?text=....

Smart contract: conectar con Ethereum (testnet) usando ethers.js y una wallet tipo WalletConnect.

**Herramientas utilizadas**

* **React Native**: framework para crear la app móvil.
  + crear proyecto : npx react-native init PanicoApp --template react-native@0.74.0
* **@react-native-async-storage/async-storage**: para guardar información local (contactos, wallet, historial).
* **@react-native-community/geolocation**: para obtener la ubicación del usuario.
* **react-native-send-intent**: para enviar mensajes SMS o WhatsApp.
* **React Navigation**: para la navegación entre pantallas.
* **JavaScript**: lenguaje principal para la lógica.
* **Flutter:** Para desarrollar la aplicación, reemplazó a React debido a que esta presentaba problemas con las dependencias.

**Validaciones iniciales (En trabajo con React)**

Antes de avanzar, se validó:

* Que todas las dependencias estuvieran instaladas.
* Que Geolocation y AsyncStorage funcionaran correctamente.
* Se hicieron pruebas de lectura y escritura en almacenamiento local.
* Se habilitaron permisos de ubicación para Android.

**Funcionalidades implementadas**

**1. CRUD de contactos**

* Agregar, listar y eliminar contactos de emergencia.
* Los datos se guardan en AsyncStorage como JSON.

**2. Configuración de Wallet**

* Inicialmente fue editable.
* Luego, se "quemó" el valor en código porque todos los usuarios usarán la misma wallet debido a que los costos de gas por transacciones serán asimilados por el ente patrocinador del proyecto, para que el usuario final pueda hacer uso de la app de manera gratuita.

**3. Activación de alerta**

* Botón dorado que al ser presionado:
  + Verifica que existan contactos y una wallet configurada.
  + Obtiene la ubicación.
  + Envía un mensaje por WhatsApp con la ubicación.
  + Registra un historial local con la fecha, ubicación y contactos.
  + Simula un registro en blockchain mostrando un mensaje de consola.

**4. Compartir ubicación manualmente**

* Desde el menú principal.
* Se valida si hay contactos.
* Se obtiene la ubicación.
* Se muestra un botón para enviar a cada contacto por separado o a todos a la vez.

**Problemas encontrados y soluciones**

**Problema: El botón de alerta no hacía nada.**

**Solución**: Se detectó que la pantalla AlertScreen no se estaba llamando desde el botón de la pantalla de inicio. Se corrigió el onPress.

**Problema: El flujo de alerta continuaba aunque faltaran contactos o wallet.**

**Solución**: Se agregaron return inmediatos tras mostrar la alerta de error, para evitar que el flujo continuara.

**Problema: El mensaje "obteniendo ubicación" aparecía aunque no hubiera contactos.**

**Solución**: Se reestructuró el useEffect para validar primero los contactos antes de intentar obtener la ubicación.

**Problema: No se sabía si la ubicación estaba en proceso o lista.**

**Solución**: Se agregó un estado loading y un ActivityIndicator para mostrar un mensaje mientras se obtiene la ubicación.

**Problema: Errores debido a dependencias.**

**Solución:** Se migra el proyecto a Flutter.

**Historial de mejoras y decisiones**

* Se eliminó temporalmente el acceso directo a la configuración de wallet desde el menú.
* Se agregó control de errores visuales con Alert.alert() en cada acción importante.
* Se guarda el historial de alertas localmente para futura reportería.
* Se agregó una alerta de confirmación para saber que la acción fue ejecutada.

## **Para revisar los reporte de los casos de activaciòn de la alerta, se decide desarrollar una Landing para consulta y reporte**

**Credenciales de la landing:** usuario y contraseña de landind hackathon

admin y 1234

### **Objetivo:**

Mostrar las alertas previas de una wallet y permitir **subir información adicional como evidencia visual y textual**.

### **Componentes del Frontend (Landing):**

1. Conexión Web3 (MetaMask).
2. Consulta de alertas por dirección.
3. Visualización de alertas en lista o tabla.
4. Formulario para agregar evidencia:
   * Campo de descripción
   * Carga de imágenes (subida a IPFS)
   * Campo de declaración
   * Botón "Agregar evidencia"

## **Funcionalidades de la Landing**

### **1. Conexión con MetaMask**

* Detecta MetaMask y solicita conexión.
* Obtiene la dirección de la wallet para usarla como identificador del usuario.

### **2. Listado de alertas por dirección**

* Usa la función del contrato getTotalAlertsBySender(address) y getAlert(index) para obtener datos.
* Muestra: nombre, ubicación, fecha, contactos.

### **3. Visualización de evidencia**

* Muestra description, declarationText, y lista de imageHashes (como enlaces si están en IPFS).

### **4. Agregar evidencia a una alerta**

* Solo si msg.sender es el creador de la alerta.
* Llama a addEvidenceToAlert(...) del contrato.

### **5. Subida a IPFS**

* Usa ipfs-http-client para subir imágenes y obtener hashes para almacenar en la blockchain.

## **Estilo visual**

* Interfaz limpia y responsiva.
* Compatible con dispositivos móviles.
* Accesible para personal de seguridad, policía y ciudadanía.

**Funcionalidades Blockchain**

**1. Registro descentralizado de alertas**

Cada vez que alguien active el botón de pánico, se guarda una transacción en la blockchain como prueba de la alerta (timestamp, ubicación, hash del usuario).

**2. Historial inalterable de incidentes**

Crea un historial semiprivado donde se registren eventos previos como evidencia verificable.

**3. Red de apoyo validada en blockchain**

Permite que los contactos de confianza estén registrados en un contrato inteligente, garantizando que solo esas personas reciban alertas.

**4. Acceso compartido con autoridades**

Se puede generar un hash verificable del incidente y dar acceso a autoridades o entidades legales con la aprobación del usuario.

**5. Geolocalización encriptada y accesible bajo permiso**

El smart contract guarda el hash de la ubicación, pero solo los contactos autorizados pueden acceder a la información real, usando una capa extra de encriptación fuera de la blockchain.