PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

Unidad Académica de Formación Técnica y Tecnológica – PUCE TEC

SOS PUCE: APLICACIÓN DE DENUNCIAS Y PREVENCION CONTRA EL ACOSO UNIVERSITARIO EN LA PUCE

Proyecto de titulación previo a la obtención del título de: Tecnología Superior En Desarrollo De Software

Autores: Josué Anthony Tipan Bolaños y Alvaro Daniel Zumba Proaño

Tutor: Carlos Miguel Cárdenas Riofrio

Quito, Ecuador

2025

Dedicatoria

Josué Tipan:

Dedico este trabajo con todo mi corazón a mi familia, quienes siempre estuvieron para mí, dándome su cariño y apoyo en cada paso de mi vida. Han sido mi mayor fuente de fuerza, amor y motivación. Gracias por su apoyo incondicional, por sus palabras de aliento en los momentos de duda, por su paciencia infinita y por nunca dejarme caer. Cada logro que he alcanzado es también suyo, porque detrás de cada paso que he dado han estado ustedes, sosteniéndome y empujándome a seguir adelante. Gracias por su cariño constante y por enseñarme el valor del esfuerzo y la perseverancia. Sin ustedes, nada de esto habría sido posible.

También quiero dedicar esta tesis a mis amigos, que me acompañaron en todo momento, que me animaron cuando sentía que no podía más y celebraron cada pequeño avance conmigo. Gracias por las risas que me ayudaron en esos momentos de estrés y por hacer que esta etapa fuera mucho más llevadera. Su amistad ha sido un regalo invaluable durante estos años.

A cada persona que de una u otra forma fueron parte de este proceso, les agradezco profundamente. Esta dedicatoria es un homenaje a todo el amor, la amistad y el apoyo que recibí. Este logro no es solo mío, es de todos ustedes que confiaron en mí cuando más lo necesitaba.

Y en especial, con mucho amor, dedico este trabajo a mi abuelito. Aunque ya no estás físicamente conmigo, tu presencia vive en cada paso que doy. Sé que este logro te haría sentir orgulloso, porque fuiste una luz en mi vida, un ejemplo de humildad, esfuerzo y amor.
Gracias por tus consejos, tus historias y por enseñarme a no rendirme nunca. Tu apoyo siempre fue mi guía y esta tesis también es tuya.

Te llevo siempre en mi corazón, abuelito.

Dedicatoria

Alvaro Zumba:

Quisiera dedicar este trabajo a mis padres y hermanas, quienes desde el primer día han estado para mí, brindándome su apoyo incondicional en cada etapa de mi vida. A mis padres, por haberme criado con paciencia, cariño y cuidado. A mis hermanas, por ser compañeras de vida y por siempre compartir conmigo.

Tabla de contenidos

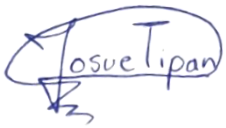
DECLARACIÓN y AUTORIZACIÓN

Yo, Josué Anthony Tipan Bolaños con C.I. 1753016342 autor(a) del trabajo de titulación intitulado: SOS PUCE: Aplicación de denuncias y prevención contra el acoso Universitario en la PUCE, previa a la obtención del título de e Tecnología en desarrollo de software en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, 18 de agosto del 2025



Josué Anthony Tipan Bolaños

C.I. 1753010342

DECLARACIÓN y AUTORIZACIÓN

Yo, Alvaro Daniel Zumba Proaño con C.I. 1725025454 autor(a) del trabajo de titulación intitulado: SOS PUCE: Aplicación de denuncias y prevención contra el acoso Universitario en la PUCE, previa a la obtención del título de e Tecnología en desarrollo de software en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, 18 de agosto del 2025

Alvaro Daniel Zumba Proaño

C.I. 1725025454

Introducción

El acoso en el entorno universitario representa una problemática persistente que afecta directamente el bienestar y el rendimiento académico de los estudiantes. A pesar de los protocolos institucionales existentes, muchas veces las víctimas no cuentan con canales accesibles, confidenciales y efectivos para reportar estos casos. Frente a esta realidad, surge la necesidad de desarrollar herramientas tecnológicas que fortalezcan la prevención y la denuncia de situaciones de acoso.

En respuesta a esta problemática, el presente proyecto propone el diseño y desarrollo de una aplicación móvil llamada SOS PUCE, dirigida a la comunidad universitaria de la Pontificia Universidad Católica del Ecuador. Esta aplicación tiene como objetivo principal facilitar el proceso de denuncia de forma segura y confidencial, así como brindar acceso a recursos informativos y apoyo institucional.

A lo largo de este documento se describen las etapas del proyecto, desde el levantamiento de requisitos y el diseño del sistema, pasando por la construcción de la solución tecnológica, hasta las pruebas realizadas para asegurar su correcto funcionamiento. También se analizan las herramientas empleadas y los beneficios que aporta esta solución digital a la comunidad universitaria.

Este trabajo busca no solo aportar una solución técnica, sino también fomentar una cultura de respeto, prevención y acción frente al acoso, apoyando los esfuerzos institucionales en la defensa de los derechos y el bienestar de todos los miembros de la PUCE. A lo largo de este documento, se van a explicar las metodologías, tecnologías usadas, los requisitos y las pruebas realizadas dentro de la aplicación.

Capítulo I

Levantamiento de Requisitos y Diseño del Sistema

En este capítulo se describe el proceso llevado a cabo para recopilar, analizar y definir los requisitos funcionales y no funcionales del sistema, así como el diseño general de la aplicación móvil SOS PUCE, destinada a facilitar la denuncia, el seguimiento y la prevención de casos de acoso dentro de la Pontificia Universidad Católica del Ecuador (PUCE).

1.1 Levantamiento de requisitos

Para determinar las necesidades específicas del sistema, se realizó un análisis del contexto universitario, tomando en cuenta los lineamientos institucionales, los procesos actuales para reportar casos de acoso y el objetivo de brindar un canal más accesible y seguro. Los requisitos fueron recopilados mediante entrevistas al personal del Departamento de Bienestar Estudiantil, y revisión de los protocolos institucionales existentes para la atención de casos de acoso. Este análisis permitió establecer los principales requisitos que debe cumplir la aplicación:

Requisitos funcionales:

- **Gestión de denuncias:**

RF01: La aplicación permite al usuario registrar una denuncia mediante un formulario que incluye descripción, lugar, fecha y evidencia.

RF02: El sistema registra de manera automática la fecha y hora de creación de cada denuncia.

RF03: El sistema permite al usuario visualizar el historial de denuncias realizadas

- **Acceso institucional y gestión de casos**

RF04: Los usuarios con permisos institucionales pueden acceder a las denuncias, para dar seguimiento mediante un sistema de autenticación seguro.

RF05: La aplicación notifica automáticamente al personal autorizado mediante correo electrónico cada vez que se registra una nueva denuncia.

RF06: Facilitar el contacto directo al Departamento de Bienestar Estudiantil.

- **Recursos y prevención**

RF07: La aplicación ofrece acceso a material informativo sobre el protocolo institucional de actuación frente al acoso.

Requisitos no funcionales:

- **Rendimiento**

RNF01: El sistema debe responder a las solicitudes de registro de denuncias en menos de 3 segundos.

RNF02 El backend debe ser capaz de manejar múltiples registros de forma simultánea sin pérdida de datos.

RF03: El sistema debe garantizar la entrega del correo de denuncia al personal institucional sin errores, asegurando confirmación de envío exitoso.

- **Mantenibilidad**

RNF04: El código fuente debe estar documentado para facilitar futuras actualizaciones y mantenimiento técnico.

- **Usabilidad**

RNF05: La aplicación debe ser intuitiva y de fácil navegación para cualquier usuario.

RNF06: El sistema debe presentar retroalimentación visual inmediata ante acciones del usuario, como confirmación de envío o advertencias de error.

- **Seguridad**

RNF07: Se debe asegurar la confidencialidad de los datos mediante autenticación segura y uso de tokens JWT para proteger las rutas del backend.

RNF08: La información personal y los archivos adjuntos deben ser almacenados de forma segura en la base de datos Supabase.

- **Compatibilidad**

RNF09: Ser compatible con dispositivos móviles iOS y Android

1.2 Diseño del sistema

En esta sección se describe cómo fue diseñado el sistema SOS PUCE a partir de los requisitos definidos previamente. Se detallan la arquitectura general, el modelado del sistema mediante diagramas, así como los prototipos de las interfaces principales. Todo esto con el objetivo de construir una aplicación funcional, segura y fácil de usar para la comunidad universitaria.

1.2.1 Arquitectura general

La solución propuesta se diseñó con una arquitectura cliente-servidor. La aplicación móvil funciona como cliente, mientras en un servidor en la nube se gestiona el almacenamiento y procesamiento de datos. Esto incluye la base de datos que almacena las denuncias.

1.2.2 Modelado del sistema

Se elaboraron los siguientes diagramas para representar el funcionamiento del sistema:

Diagrama de casos de uso:

Identificar las principales interacciones del usuario con la aplicación, como los recursos que están a disposición del usuario para que se informe y conozca un poco más del protocolo que se lleva a cabo en estos casos, registrar una denuncia, ver el historial de denuncias que se han realizado de manera detallada, la zona encargada recibe al correo la denuncia y se comunica con la victima lo más pronto posible.

Diagrama de clases:

Describe la estructura lógica del software, detallando las entidades principales como Usuario, Denuncia y Notificación.

Prototipo de interfaces:

Se diseñaron pantallas preliminares que muestran cómo se visualizarán los formularios de denuncia dentro de la aplicación. Estas interfaces permiten al usuario ingresar de manera sencilla y guiada toda la información necesaria para realizar una denuncia, como la descripción del incidente, la fecha. Además, el prototipo incluye pantallas para la confirmación del envío. Con esto busco asegurar una experiencia de usuario intuitiva y eficiente durante todo el proceso de denuncia.

1.3 Herramientas utilizadas

Para el desarrollo del proyecto se emplearon las siguientes tecnologías:

Visual Studio Code: Ayudó a editar y desarrollar todo el código fuente de la aplicación de manera eficiente

Node.js: Para la creación del servidor y la lógica del backend mediante JavaScript.

React Native: Librería que permite crear aplicaciones móviles tanto para Android como para IOS usando la misma base de código hecho en JavaScript y React en lugar de usar lenguajes nativos.

Se seleccionó React Native en lugar de lenguajes nativos (Java/Kotlin para Android y Swift para iOS) debido a su capacidad de desarrollar aplicaciones para ambas plataformas con una sola base de código, por lo que se optimiza el tiempo de desarrollo y a la vez se reducen los

costos. Además, la integración con Expo permitió acelerar las pruebas y la implementación de funcionalidades como autenticación o carga de archivos de una manera cercana a la nativa.

Supabase: Es una plataforma de backend como servicio (BaaS) que ayuda a crear el backend de una forma rápida y sencilla. Está construida sobre PostgreSQL y ofrece herramientas listas para usar como bases de datos, autenticación de usuarios, almacenamiento de archivos y APIs en tiempo real, para no tener que desarrollar desde cero.

Expo: Un conjunto de herramientas que facilita el desarrollo y prueba de aplicaciones móviles hechas en react native, sin necesidad de escribir con código nativo.

Balsamiq: Software de diseño que permite a los usuarios crear wireframes para interfaces de usuario como sitios web o aplicaciones móviles.

Railway: Es una plataforma como servicio (PAAS), que permite alojar y desplegar aplicaciones de backend en la nube, sin la necesidad de tener que gestionar servidores de forma manual.

1.4 Metodología

Scrum es un enfoque ágil que se aplica para la administración y creación de proyectos de alta complejidad. En lugar de hacer un esfuerzo por entregar un producto final completo de una vez, Scrum facilita la entrega incremental y reiterativa, permitiendo responder rápidamente a los cambios y mejorar continuamente. (atlassian, 2024).

Para el desarrollo de esta aplicación se utilizará SCRUM como metodología debido a su capacidad para gestionar proyectos tecnológicos de forma eficiente y colaborativa. SCRUM se adapta de forma perfecta a proyectos que requieren flexibilidad y entregas incrementales,

permitiendo ajustar la solución tecnológica a las necesidades y cambios que puedan surgir durante el desarrollo.

Por lo tanto, SCRUM nos ofrece las siguientes ventajas:

1. Organización del trabajo en equipo: Define roles claros (Scrum Master, Product Owner y equipo de desarrollo) que optimizan la colaboración y responsabilidad de cada miembro.
2. Ciclos iterativos (Sprints): Permiten planificar, desarrollar y entregar partes funcionales del proyecto en periodos cortos, asegurando avances constantes y medibles.
3. Adaptabilidad: Su enfoque flexible facilita responder a cambios en los requerimientos o prioridades del proyecto.
4. Mejora continua: Mediante reuniones como los dailys y sprint retrospectives se analizan los objetivos de cada día y los resultados de cada sprint para identificar áreas de mejora.

1.5 Limitantes de la aplicación

Para el correcto uso de la aplicación SOS PUCE, es importante tomar en cuenta las siguientes condiciones de funcionamiento:

- Acceso institucional: La autenticación de usuarios está habilitada únicamente para direcciones de correo electrónico con el dominio @puce.edu.ec, asegurando que el uso de la aplicación sea exclusivo para la comunidad universitaria.

- Carga de evidencias: En cada denuncia se pueden adjuntar hasta cinco archivos como evidencia, esto permite una gestión ordenada y eficiente de la información.
- Conectividad: El registro de denuncias y el acceso a los recursos requieren conexión a internet, ya que la aplicación interactúa en tiempo real con el servidor y la base de datos.

Capítulo II

Construcción del Sistema

En este capítulo se describe el proceso de desarrollo de la aplicación móvil SOS PUCE, orientada a la prevención y denuncia de casos de acoso en el entorno universitario. Se detallan las fases de implementación del sistema, el uso de tecnologías específicas, y la forma en que se integraron los diferentes componentes para lograr una solución funcional, segura y accesible.

2.1 Estructura general del desarrollo

La aplicación se desarrolló siguiendo el patrón cliente-servidor, en este tipo de diseño de software las tareas se reparten entre dos tipos de entidades: los clientes, que solicitan recursos o servicios, y los servidores, que los procesan y responden. En este enfoque, el cliente es una aplicación móvil construida con React Native y el servidor es el backend, que fue desarrollado con Node.js y Express, y también fue desplegado en Railway para garantizar disponibilidad en la nube.

El backend se conecta con Supabase, que se utiliza exclusivamente como servicio de base de datos PostgreSQL y almacenamiento de archivos adjuntos. La autenticación de usuarios mediante OAuth2 con Google, así como la lógica de negocio y el manejo de sesiones, son gestionados directamente desde el backend personalizado.

Este enfoque permitió una integración rápida, segura y escalable de las funcionalidades requeridas, manteniendo una clara separación entre la interfaz móvil, la lógica del servidor y la persistencia de datos.

2.2 Organización del proyecto

El sistema se organizó en tres capas principales:

Frontend móvil (React Native + Expo)

- Autenticación mediante Google OAuth2.
- Formulario de denuncias accesible e intuitivo.
- Historial de denuncias registradas por el usuario.
- Acceso a protocolos institucionales y material informativo.

Generación local de archivos PDF con los datos ingresados.

- Backend personalizado (Node.js + Express)
- API REST para operaciones CRUD sobre las denuncias.
- Gestión de autenticación y autorización con tokens JWT.
- Envío automático de correos electrónicos a Bienestar Estudiantil.
- Generación dinámica de archivos PDF.
- Conexión con Supabase para persistencia de datos.

Servicios Backend externos (Supabase)

- Base de datos relacional en PostgreSQL.
- Almacenamiento seguro de archivos (evidencias).
- Módulo de autenticación externo integrado vía backend.

2.3 Desarrollo del frontend

La construcción del frontend de la aplicación SOS PUCE se llevó a cabo utilizando React Native, una biblioteca de JavaScript que permite desarrollar aplicaciones móviles tanto para Android como IOS con una sola base de código. Para facilitar la configuración, ejecución y pruebas durante el desarrollo, se empleó el entorno de herramientas Expo que ofrece una experiencia más ágil para desarrolladores al eliminar la necesidad de compilar código nativo.

El enfoque principal fue ofrecer una interfaz amigable, intuitiva y centrada en la experiencia del usuario, especialmente para quienes reportan situaciones delicadas como el acoso. Se implementaron pantallas con navegación fluida que permiten:

- Iniciar sesión mediante la cuenta institucional de Google, utilizando OAuth2 como protocolo de autenticación segura.
- Acceder a un formulario diseñado para que el usuario pueda registrar una denuncia detallada. Este formulario incluye campos como la descripción del incidente, la fecha y lugar del hecho, así como la opción de adjuntar evidencia en formato de imagen o archivo.
- Visualizar un historial de denuncias presentadas, facilitando el seguimiento personal de los casos reportados.
- Consultar información útil sobre los tipos de violencia, el protocolo institucional de actuación y los canales de apoyo disponibles.

Figura 1

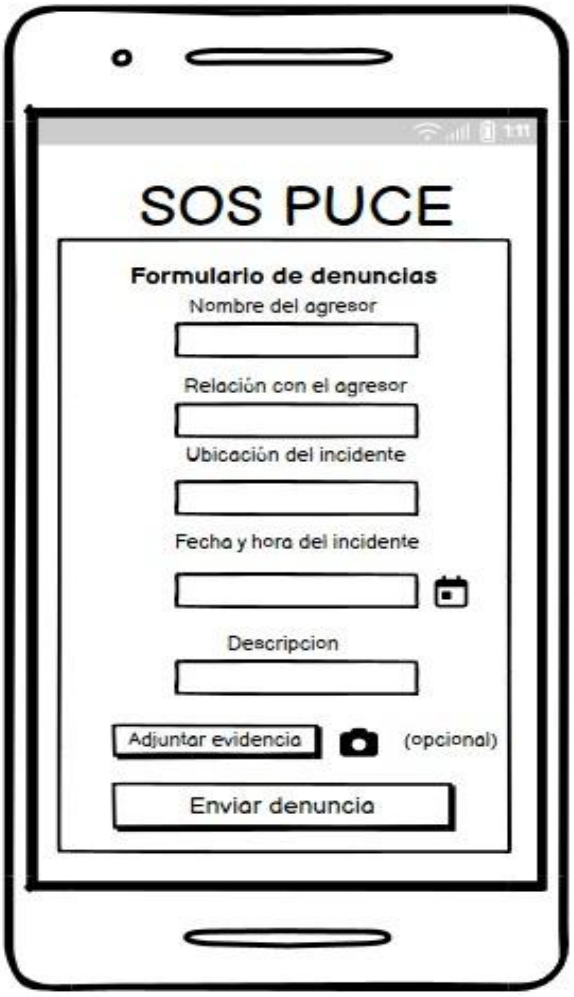
Interfaz de login

Figura 2

Interfaz de verificación de login

Figura 3

Interfaz formulario de denuncias



Cada parte de la aplicación fue desarrollada procurando reutilizar el código siempre que fue posible, validando los datos de forma local antes de enviarlos al servidor. Además, se agregó retroalimentación inmediata al usuario, por ejemplo, mostrando mensajes cuando una denuncia se registra correctamente o si ocurre un error al enviarla. Para la navegación entre pantallas se utilizó React Navigation, esto permitió un flujo más ordenado y sin interrupciones. También se

trató de mantener un diseño claro y fácil de entender, evitando poner demasiada información en una sola pantalla para no confundir al usuario.

2.4 Desarrollo del backend

El backend fue desarrollado con Node.js, utilizando el framework Express para la creación de un conjunto de rutas RESTful que permiten manejar toda la lógica del servidor. Esta capa se encarga de procesar las solicitudes provenientes del cliente, aplicar validaciones, manejar la autenticación y gestionar la conexión con la base de datos alojada en Supabase.

Uno de los elementos clave del backend fue la implementación de un sistema de autenticación seguro basado en OAuth2 con Google, el cual permite que los usuarios ingresen con su cuenta institucional sin necesidad de crear una cuenta desde cero. Luego de autenticarse, se genera un token JWT (JSON Web Token) que se utiliza para validar las sesiones y proteger las rutas sensibles de la API.

Entre las tareas más importantes que maneja el backend están:

- Recibir y verificar los datos que se envían desde el formulario de denuncias.
- Guardar la información en Supabase, que funciona como base de datos en PostgreSQL.
- Crear un archivo PDF con los datos del reporte
- Enviar automáticamente un correo electrónico al Departamento de Bienestar Estudiantil cada vez que se registra una nueva denuncia.
- Guardar en Supabase cualquier archivo adjunto que el usuario suba como evidencia (por ejemplo, imágenes o documentos).

Toda esta lógica está organizada en funciones separadas dentro de controladores, lo que ayuda a tener un código más limpio, fácil de entender y mantener. Esta estructura también permite que el sistema sea más escalable en caso de que en el futuro se quiera ampliar con nuevas funciones.

2.5 Integración y conexión entre capas

La comunicación entre el frontend y el backend se realizó mediante llamadas HTTP al servidor desarrollado con Node.js y Express, desplegado en la nube. Estas llamadas se encapsularon en funciones dentro de un archivo api.js, lo que permite mantener un código organizado, reutilizable y fácil de mantener.

El archivo api.js contiene funciones específicas para operaciones como el envío y verificación de códigos de autenticación, el registro de denuncias, la descarga de archivos PDF generados por el backend, y la consulta del historial de denuncias.

Para proteger los datos y controlar el acceso, se utiliza autenticación basada en tokens JWT. Los tokens se almacenan localmente en el dispositivo del usuario mediante AsyncStorage y se incluyen en las cabeceras de las peticiones al backend.

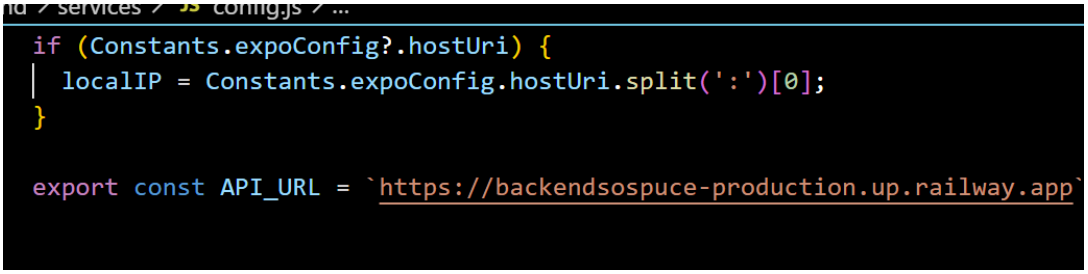
El backend, a su vez, se encarga de procesar las solicitudes, interactuar con la base de datos alojada en Supabase, generar archivos PDF cuando corresponde, y enviar correos electrónicos automáticos a las autoridades universitarias responsables.

Se utilizaron tokens JWT para autenticar las peticiones y asegurar que los datos sensibles solo sean visibles para los usuarios autorizados.

2.5.1. Ejemplos de Integración entre capas

Figura 4

Conexión entre el frontend y backend



Este código permite que la aplicación móvil (frontend) envíe la información de una denuncia al servidor (backend) utilizando el protocolo HTTP y autenticación mediante token JWT

Figura 5

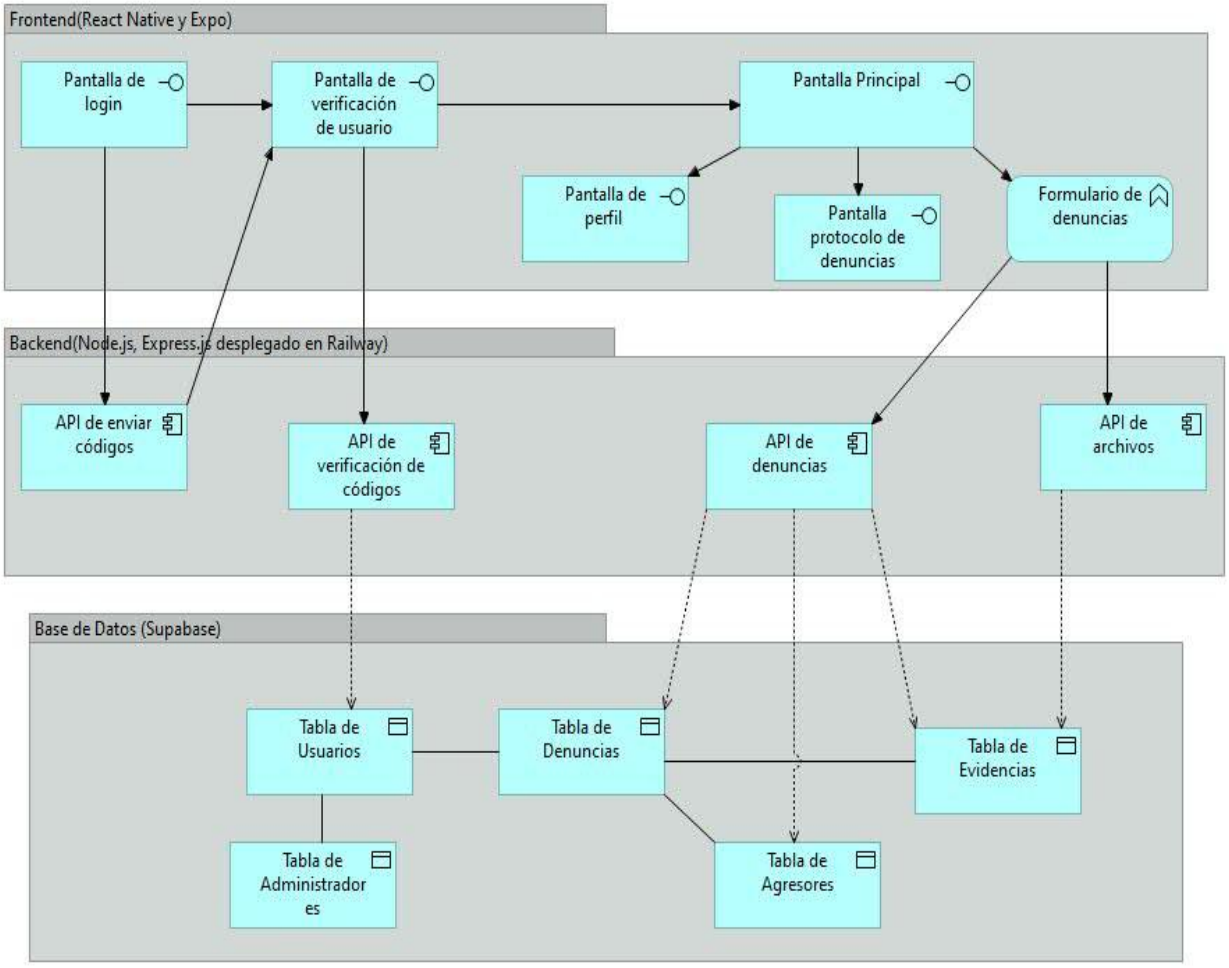
Conexión del backend a la base de datos

```
require('dotenv').config();

if (!process.env.SUPABASE_URL || !process.env.SUPABASE_SERVICE_ROLE_KEY) {
  console.error('Faltan SUPABASE_URL o SUPABASE_SERVICE_ROLE_KEY');
  process.exit(1);
}

const supabase = createClient(
  process.env.SUPABASE_URL,
  process.env.SUPABASE_SERVICE_ROLE_KEY
);

console.log('Conectado a Supabase');

module.exports = supabase;
```

Este código permite que el backend pueda interactuar con la base de datos PostgreSQL alojada en la nube y con el servicio de almacenamiento de evidencias.

Figura 6

Arquitectura de capas



2.6 Control de versiones y pruebas iniciales

Durante el desarrollo de la aplicación se utilizó Git como herramienta para llevar el control de versiones. Esto permitió organizar el trabajo en equipo, guardar un historial de cambios y mantener la estabilidad del proyecto. Se trabajó con ramas separadas según el tipo de cambio, como nuevas funcionalidades o correcciones, lo cual facilitó una colaboración más ordenada y segura.

Las primeras pruebas de funcionamiento se realizaron en dispositivos Android usando la aplicación Expo Go, que ayudó a verificar rápidamente cómo se comportaba la app en un entorno real. Se revisó que las pantallas funcionen bien, que la navegación sea fluida, que los formularios validen correctamente los datos y que toda la información se envíe sin errores al backend. También se comprobó que los PDF se generen con la información ingresada y que los correos lleguen correctamente a las autoridades correspondientes.

Gracias a estas pruebas se pudieron detectar pequeños errores y hacer ajustes antes de tener una versión estable. Esto fue clave para asegurar que el sistema funcione correctamente y que la experiencia del usuario sea fluida y confiable.

Capítulo III

Pruebas y Estabilización

En esta sección se van a explicar las pruebas a las que se le sometió a la app para garantizar la funcionalidad y calidad. Se implementaron pruebas que abarcaron tanto el frontend como el backend y las distintas herramientas que se utilizaron.

Para el backend y la conexión de las APIs, se emplearon herramientas especializadas como Postman y JMeter. Estas permitieron realizar pruebas exhaustivas de rendimiento, carga y funcionalidad, asegurando que las APIs respondieran correctamente ante diferentes tipos de solicitudes y volúmenes de usuarios.

Además, se llevaron a cabo pruebas en dispositivos móviles con sistemas operativos iOS y Android para detectar posibles errores o inconsistencias en la interfaz y la experiencia del usuario. Esto permitió identificar y corregir fallos antes del lanzamiento final.

3.1 Pruebas del backend

Para las pruebas del backend se emplearon las herramientas Postman y JMeter debido a sus características que permiten garantizar la calidad y el correcto funcionamiento de las APIs.

Postman se utilizó principalmente para realizar pruebas funcionales ya que facilita laconstrucción y ejecución de solicitudes HTTP, permitiendo validar que las APIs respondan correctamente a distintos tipos de peticiones y parámetros. Además, postman ofrece una interfaz intuitiva que facilita la verificación de respuestas, lo que puede ayudar a detectar errores en la comunicación con el servidor.

JMeter se empleó para llevar a cabo pruebas de rendimiento y carga, simulando múltiples usuarios concurrentes que interactúan con las APIs. Esto permitió evaluar cómo se comporta el backend bajo diferentes niveles de estrés y garantizar que la aplicación sea capaz de manejar distintos números de solicitudes.

3.1.1 Pruebas funcionales

Figura 7

Prueba en Postman para enviar códigos de verificación

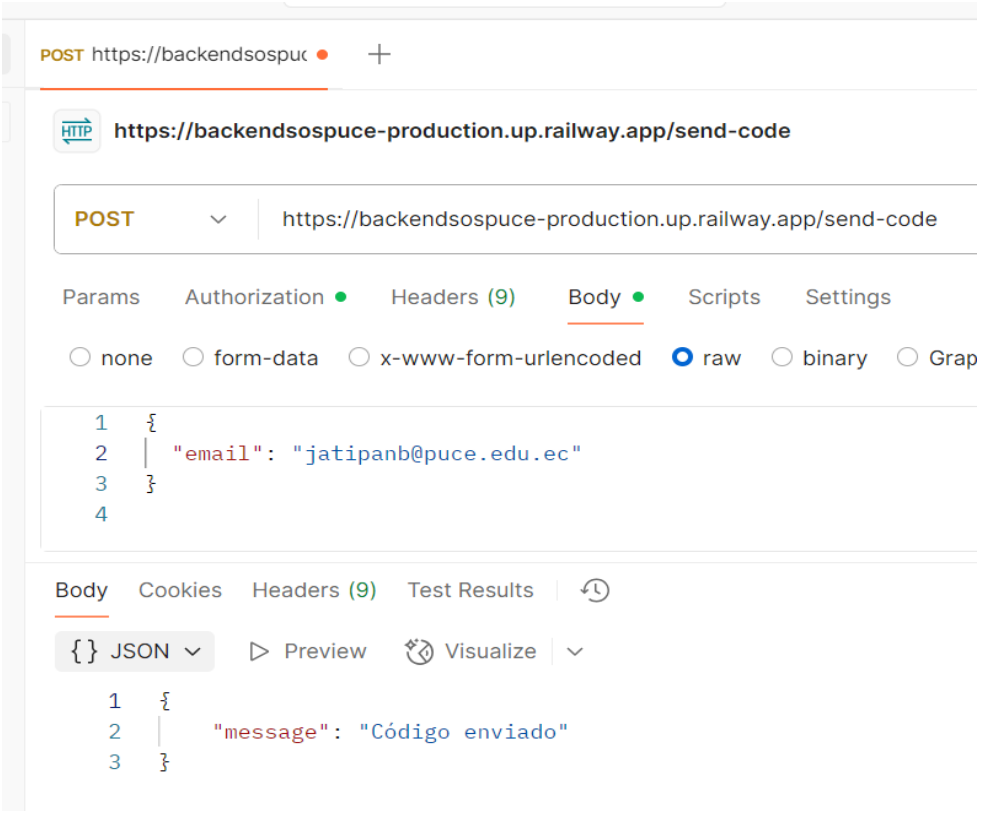


Figura 8

Prueba en postman para comprobar el código de inicio de sesión y la generación de un token para el usuario

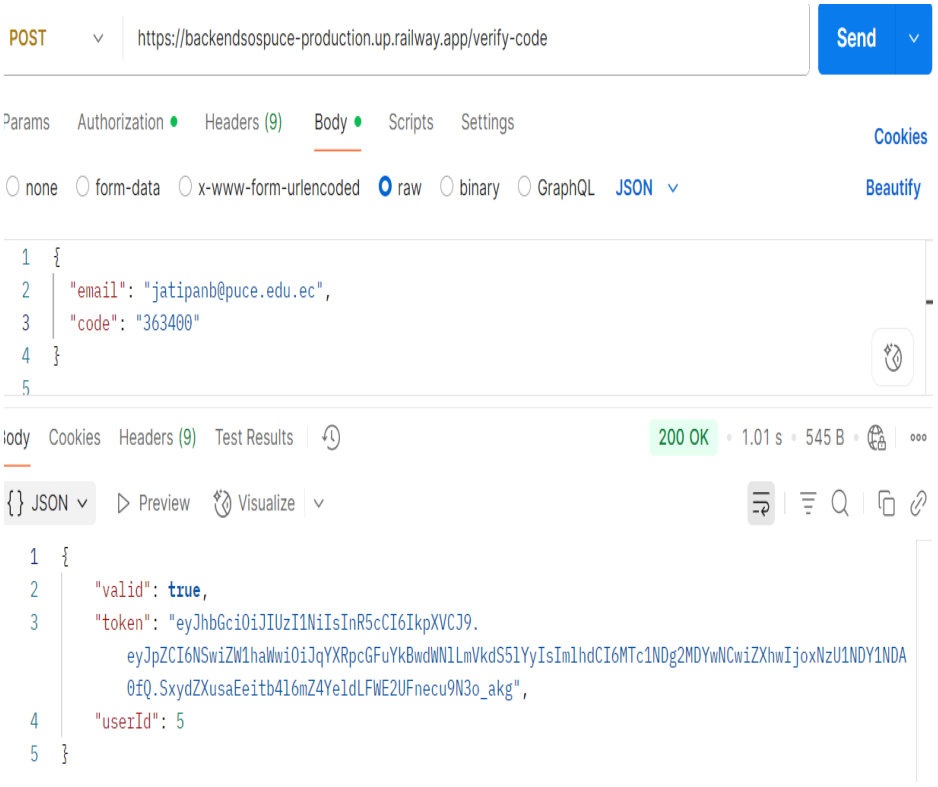
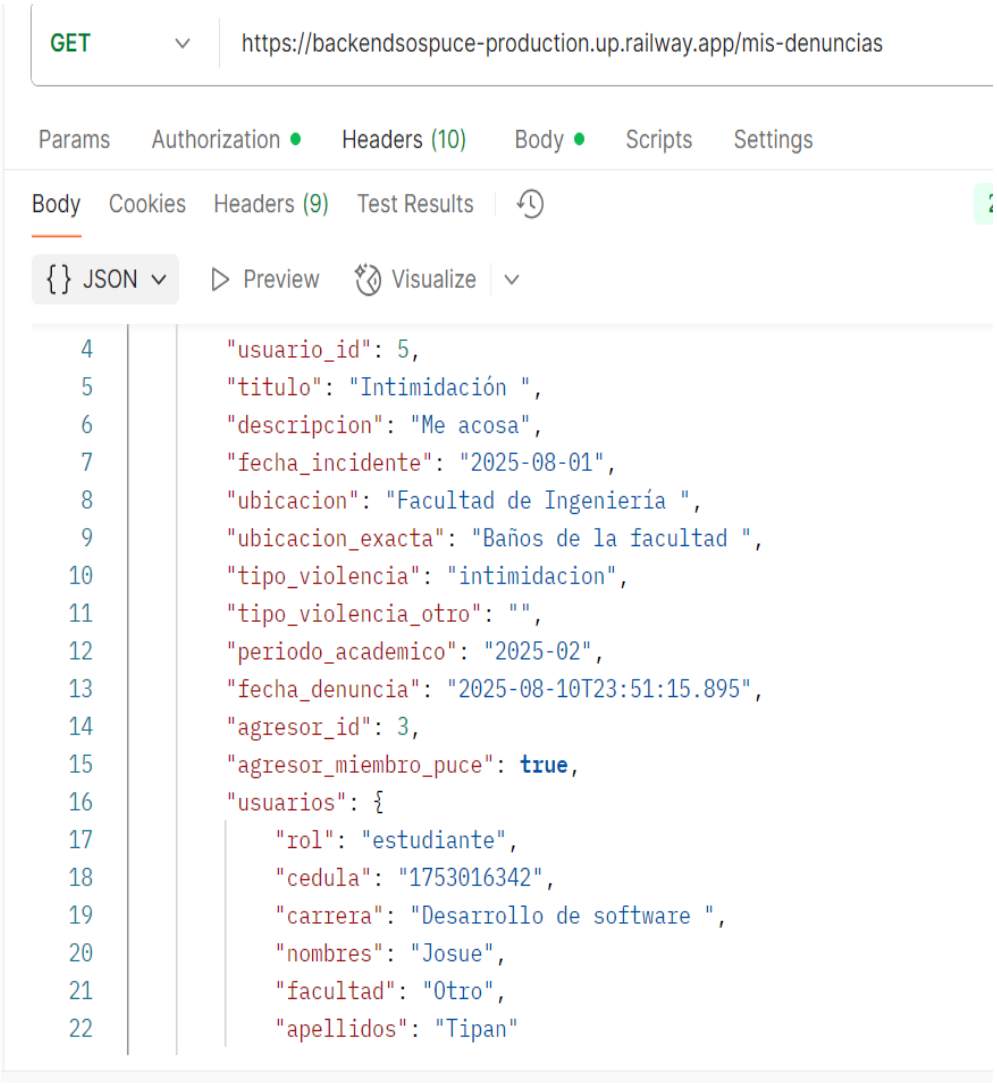


Figura 9

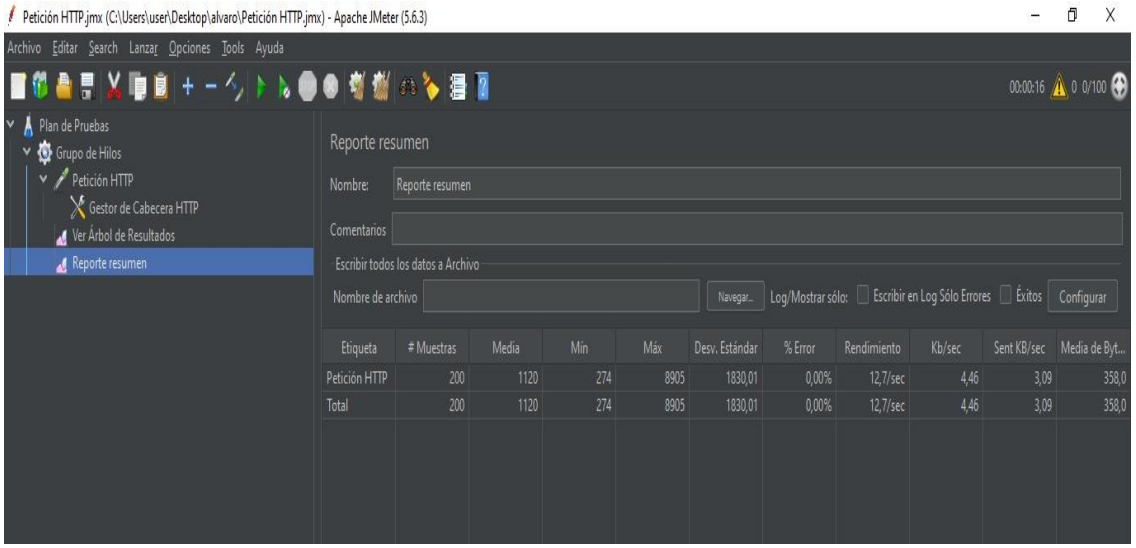
Prueba en postman para probar la api de denuncias de un usuario específico



3.2.2 Pruebas de rendimiento

Figura 10

Estructura de pruebas en Apache JMeter



Nota: En esta imagen se muestra la estructura utilizada para las pruebas en Jmeter.

- Grupo de Hilos: Contenedor principal de Jmeter donde se define como se ejecutará la prueba de carga.

Figura 11

Panel de configuración solicitudes http

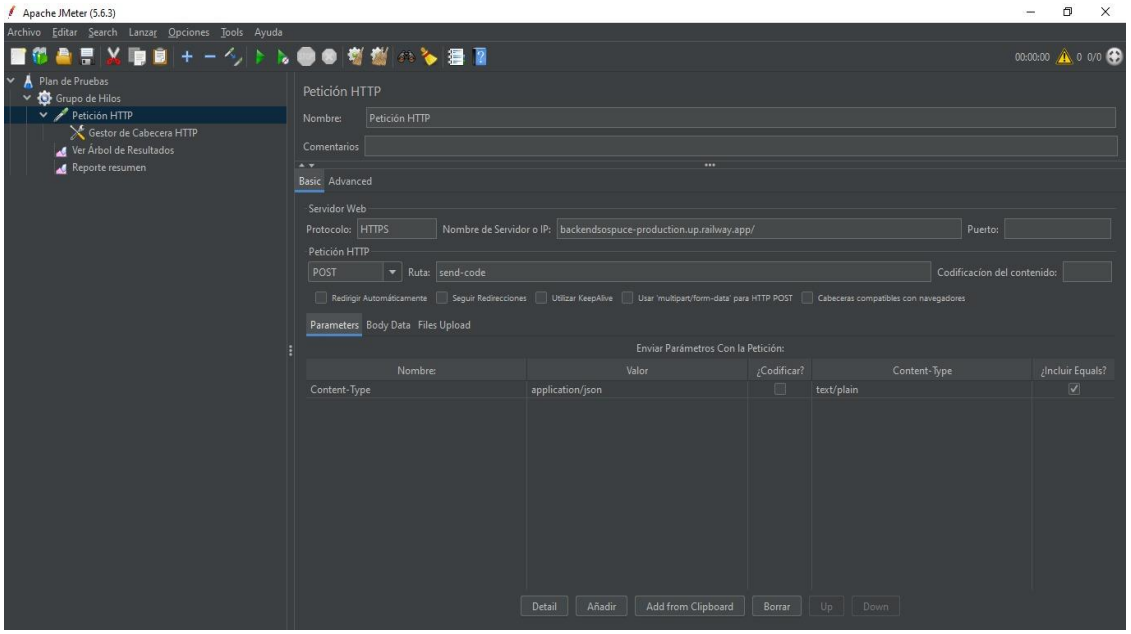
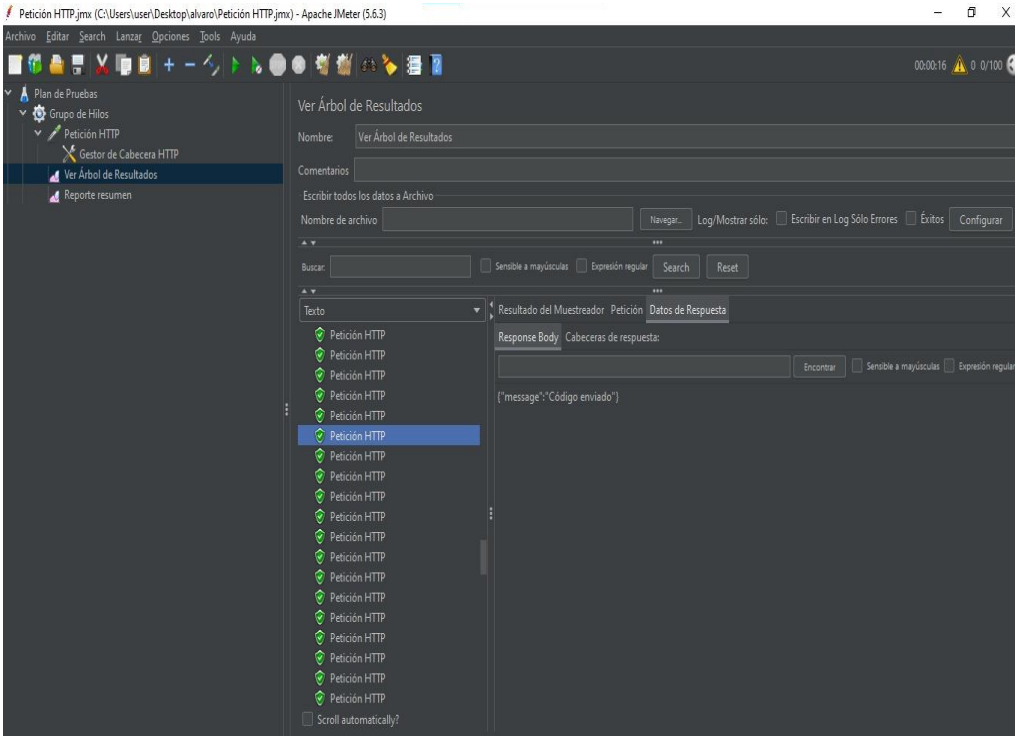


Figura 12

Panel de configuración árbol de resultados



- **Árbol de resultados:** Estructura visual que muestra solicitudes y respuestas de las peticiones http

3.2 Validación en dispositivos iOS y Android

Figura 13

App instalada en dispositivo móvil

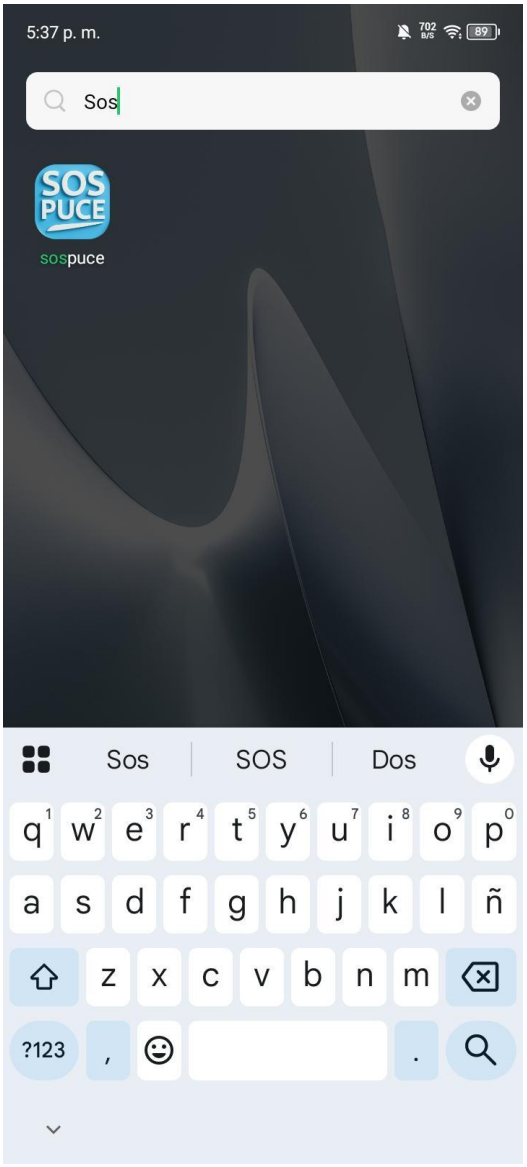


Figura 14

Captura de login

Figura 15

Vista pantalla de inicio

Figura 16

Vista de pantalla de reportar denuncia

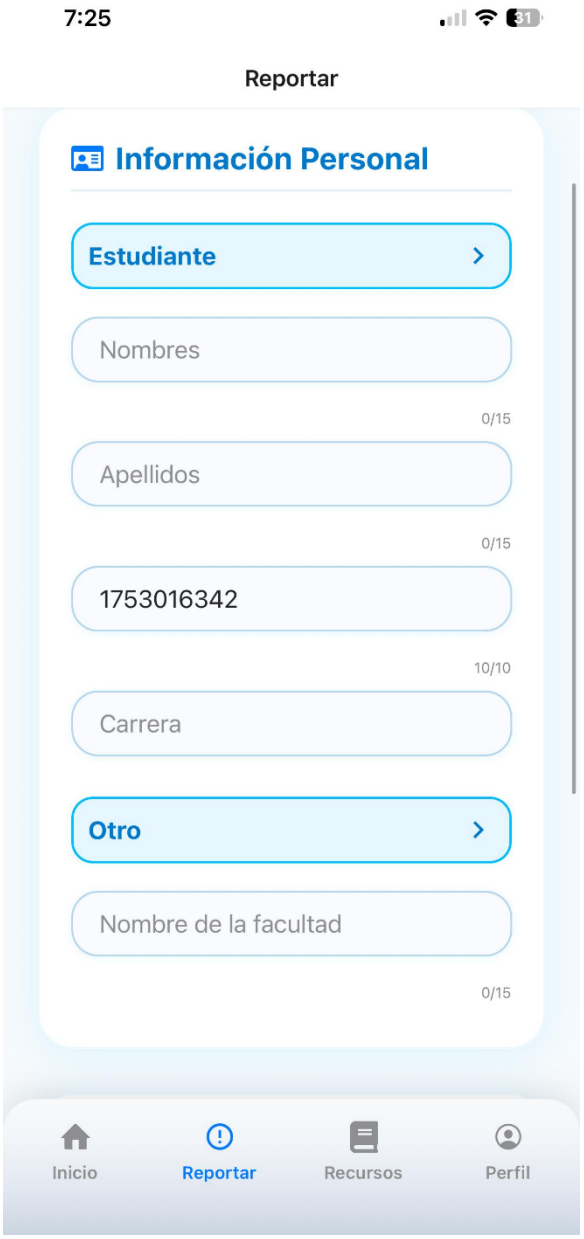


Figura 17

Vista pantalla de recursos

Figura 18

Vista de pantalla de perfil de usuario

3.3 Pruebas de rendimiento en un dispositivo móvil

Figura 19

Consumo de recursos de la aplicación en un dispositivo Android 12

C:\Windows\system32\cmd.exe

```
C:\Users\User>adb shell dumpsys meminfo com.alvaropr.sospuce
Applications Memory Usage (in Kilobytes):
Uptime: 352553195 Realtime: 655437641

** MEMINFO in pid 5105 [com.alvaropr.sospuce] **
                   Pss  Private  Private  SwapPss      Rss     Heap     Heap     Heap
                 Total    Dirty    Clean    Dirty    Total     Size    Alloc     Free
                ------   ------   ------   ------   ------   ------   ------   ------
  Native Heap    42037    42008        4       76    43184    55804    51829     3974
  Dalvik Heap     7507     7440        8       87     8408    15194     7597     7597
 Dalvik Other     1393     1360        4        0     1760
       Stack      1036     1036        0        0     1044
      Ashmem         7        0        0        0      440
     Gfx dev      9540     9540        0        0     9540
   Other dev         9        0        8        0      280
    .so mmap      2127      228      136        2    28676
   .jar mmap      1436        0      200        0    23004
   .apk mmap     14830     1464    11024        0    23780
   .ttf mmap       744        0      576        0     1228
   .dex mmap     15096       36    14664        0    16852
   .oat mmap       751        0       80        0    14260
   .art mmap      5631     5256       40      119    20500
  Other mmap       421        8      252        0     1764
  EGL mtrack     41104    41104        0        0    41104
   GL mtrack       384      384        0        0      384
     Unknown     12037    11940       88        1    12396
       TOTAL    156375   121804    27084      285   248604    70998    59426    11571

 App Summary
                       Pss(KB)                        Rss(KB)
                        ------                         ------
           Java Heap:    12736                          28908
         Native Heap:    42008                          43184
                Code:    28412                         107872
               Stack:     1036                           1044
            Graphics:    51028                          51028
       Private Other:    13668
              System:     7487
             Unknown:                                   16568
```

Figura 20

Tabla de resultados de consumo de recursos de la aplicación en un dispositivo Android 12

Concepto	**Consumo / Cantidad**	**Explicación**
Memoria total usada	153 MB	Espacio que ocupa la app mientras funciona.
Java Heap	12 MB	Parte de la memoria donde la app guarda información mientras maneja datos.
Graphics	50 MB	Memoria usada para mostrar imágenes y la interfaz que ves en pantalla.
Vistas	265 elementos	Cantidad de elementos activos en pantalla (botones, imágenes, etc.).
Activities	1 pantalla	Solo hay una pantalla principal abierta de la app mientras se usa.
Native Heap	41 MB	Memoria para procesos internos del sistema y

		motor de React Native; se libera al cerrar la app.

De forma resumida el consumo de recursos de la app es normal, no es bajo ni demasiado alto, funciona de forma normal en la mayoría de dispositivos.

Si la app creciera en complejidad (más imágenes, más elementos en pantalla), estas cifras podrían subir, pero mientras tanto son normales.

Conclusiones

- La aplicación SOS PUCE demostró ser una solución tecnológica eficaz para la denuncia y prevención del acoso universitario, ofreciendo un canal seguro, accesible y confidencial para la comunidad PUCE.
- La arquitectura cliente-servidor, combinada con React Native, Node.js y Supabase, permitió desarrollar un sistema escalable, multiplataforma y con una experiencia de usuario que es intuitiva.
- Las pruebas funcionales y de rendimiento realizadas con Postman y JMeter, validaron que el backend es capaz de manejar múltiples solicitudes de manera estable y segura.
- La integración de autenticación con Google OAuth2 y el uso de tokens JWT reforzó la seguridad y protección de datos sensibles.
- El material informativo y los recursos preventivos incluidos en la sección de recursos en la aplicación fortalecen la denuncia y fomentan la concientización y educación de los usuarios.

Recomendaciones

- Implementar una sección de chat en tiempo real para que las víctimas puedan comunicarse directamente con el personal de Bienestar Estudiantil.
- Ampliar la compatibilidad de la aplicación para incluir versiones offline que permitan registrar denuncias sin conexión y enviarlas cuando haya internet.
- Realizar campañas periódicas de difusión en la universidad para que toda la comunidad conozca la existencia y uso de la aplicación.
- Implementar un sistema de notificaciones para informar a los usuarios sobre actualizaciones de sus denuncias y nuevos recursos disponibles.
- Mantener actualizadas las dependencias y librerías del proyecto para prevenir vulnerabilidades de seguridad.

Referencias bibliográficas

Auth. (s. f.). JSON Web Tokens. Auth0 Docs. https://auth0.com/docs/secure/tokens/json-web-tokens

Balsamiq: Fast, focused wireframing tools. (s. f.). https://balsamiq.com/

Expo documentation. (s. f.). Expo Documentation. https://docs.expo.dev/

Express - Node.js web application framework. (s. f.). https://expressjs.com/

Introduction · React native. (2025, 17 junio). https://reactnative.dev/docs/getting-started

atlassian. (2024). Qué es scrum y cómo empezar. Obtenido de atlassian: https://www.atlassian.com/es/agile/scrum

Node.js — Introduction to Node.js. (s. f.). https://nodejs.org/en/learn/getting-started/introduction-to-nodejs

Postman: The World's Leading API Platform | Sign Up for Free. (s. f.). https://www.postman.com/

Supabase Docs. (2025, 8 agosto). Supabase Docs. https://supabase.com/docs

Anexos

Manual de Usuario